ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Методика підвищення ефективності управління каналами постачання продуктів в системі мережевого ресторанного бізнесу»

на здобуття освітнього ступеня магістра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми Інженерія програмного забезпечення

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Владислав ЦИВІНДА

(підпис)

Виконав: здобувач вищої освіти група ПДМ-62

Владислав ЦИВІНДА

Керівник: Володимир САДОВЕНКО

канд.фіз.-мат.наук., доц.

Рецензент: _____

науковий ступінь, вчене звання Ім'я, ПРІЗВИЩЕ

Київ 2026

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Магістр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення

_____ Ірина ЗАМРІЙ
«_____» _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Цивінді Владиславу Сергійовичу

1. Тема кваліфікаційної роботи: «Методика підвищення ефективності управління каналами постачання продуктів в системі мережевого ресторанного бізнесу»

керівник кваліфікаційної роботи Володимир САДОВЕНКО канд.фіз.-мат.наук, доцент,
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «30» жовтня 2025 р. № 467.

2. Строк подання кваліфікаційної роботи «19» грудня 2025 р.

3. Вихідні дані до кваліфікаційної роботи: науково-технічна література, методи оптимізації логістичних витрат, параметри моделей управління запасами, методи прогнозування попиту.

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

1. Аналіз існуючих підходів до управління каналами постачання у мережевому ресторанному бізнесі
2. Дослідження проблем і факторів що впливають на ефективність систем постачання продуктів
3. Розробка методики підвищення ефективності управління каналами постачання
4. Оцінка ефективності запропонованої методики та рекомендації щодо її практичного використання

5. Перелік графічного матеріалу: *презентація*
   1. Формалізація задачі управління каналами постачання.
   2. Архітектура програмного комплексу.
   3. Публікації та апробація роботи.
   4. Автоматизація формування замовлення
   5. Метод проактивного контролю
6. Дата видачі завдання «31» жовтня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз наявної науково-технічної літератури	31.10-05.11.25	
2	Вивчення матеріалів для аналізу підходів до управління каналами постачання	06.11-12.11.25	
3	Дослідження проблем і факторів що впливають на ефективність систем постачання	13.11-19.11.25	
4	Розробка методики підвищення ефективності управління	27.11-03.12.25	
5	Оцінка ефективності запропонованої методики та рекомендації щодо практичного використання	04.12-10.12.25	
6	Оформлення роботи: вступ, висновки, реферат	11.12-15.12.25	
7	Розробка демонстраційних матеріалів	15.12-19.12.25	

Здобувач вищої освіти _____ Владислав ЦИВІНДА
(підпис)

Керівник
кваліфікаційної роботи _____ Володимир САДОВЕНКО
(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня магістра: 113 стор., 12 табл., 18 рис., 33 джерела.

Мета роботи – підвищення ефективності управління каналами постачання продуктів системі мережевого ресторанного бізнесу шляхом впровадження автоматизованого планування та проактивного контролю.

Об'єкт дослідження – процеси управління каналами постачання продуктів в системі мережевого ресторанного бізнесу.

Предмет дослідження – практичні інструменти та підходи спрямовані на підвищення ефективності управління каналами постачання продуктів в системі мережевого ресторанного бізнесу.

У роботі використано методи системного аналізу, теорії управління запасами, математичного моделювання, проєктування програмних систем та імітаційного моделювання.

Проведено аналіз сучасних інформаційних систем управління логістикою та виявлено їхні архітектурні недоліки, зокрема реактивний характер управління та технологічну залежність від застарілих облікових платформ. Формалізовано задачу управління постачанням як задачу багатокритеріальної оптимізації.

Розроблено комплексну методику та спроєктовано мікросервісну архітектуру програмної системи, що включає модулі гібридного прогнозування попиту (з урахуванням сезонності та типу товару) та оптимізації логістичних маршрутів. Запропоновано алгоритм проактивного контролю, який дозволяє в режимі реального часу виявляти відхилення (затримки, недопоставки) та автоматично сповіщати відповідальних осіб.

Проведено імітаційне моделювання роботи розробленої системи на синтетичному наборі даних. Результати експерименту підтвердили, що впровадження методики дозволяє підвищити точність інвентаризації до 98,7%, забезпечити рівень доступності товарів 99,3% та скоротити час реакції на логістичні збої з 12 годин до 1,24 секунди.

Виконано розрахунок економічної ефективності, який показав доцільність впровадження розробленого програмного рішення для мережевих закладів харчування.

КЛЮЧОВІ СЛОВА: КАНАЛИ ПОСТАЧАННЯ, ЕФЕКТИВНІСТЬ УПРАВЛІННЯ, ЛОГІСТИКА, ПРОГНОЗУВАННЯ ПОПИТУ, ОПТИМІЗАЦІЯ ВИТРАТ, ЗАКУПІВЕЛЬНА ДІЯЛЬНІСТЬ.

ABSTRACT

Text part of the qualification work for obtaining a master's degree: 113 pages, 12 tables, 18 figures, 33 sources.

The purpose of the work iis to improve the efficiency of food supply chain management in a restaurant network system by introducing automated planning and proactive control.

The object of research is the processes of food supply chain management in a restaurant network system.

The subject of research is practical tools and approaches aimed at improving the efficiency of food supply chain management in a restaurant network system.

The work uses methods of system analysis, inventory theory, mathematical modeling, software system design, and simulation modeling.

An analysis of modern information systems for logistics management was conducted, identifying their architectural flaws, particularly the reactive nature of management and technological dependence on legacy accounting platforms. The supply chain management problem was formalized as a multi-criteria optimization task.

A comprehensive methodology was developed, and a microservice architecture for the software system was designed. It includes modules for hybrid demand forecasting (considering seasonality and item type) and logistics route optimization. A proactive control algorithm was proposed, enabling real-time detection of deviations (delays, shortages) and automatic notification of responsible personnel.

Simulation modeling of the developed system was performed using a synthetic dataset. The experimental results confirmed that implementing the methodology increases inventory accuracy to 98.7%, ensures a product availability level of 99.3%, and reduces the reaction time to logistical failures from 12 hours to 1.24 seconds.

KEYWORDS: SUPPLY CHAINS, SOFTWARE ENGINEERING, MICROSERVICE ARCHITECTURE, DEMAND FORECASTING, SIMULATION MODELING, PROACTIVE CONTROL, LOGISTICS AUTOMATION.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

KPI – ключові показники ефективності
КТК – ключові точки контролю
РЦ – розподільчий центр
API – інтерфейс прикладного програмування
BI – бізнес-аналітика
BOM – специфікація матеріалів
BPMN – нотація та модель бізнес-процесів
CAPEX – капітальні витрати
COGS – собівартість реалізованої продукції
HoReCa – сфера індустрії гостинності: готелі, ресторани, кейтеринг
IaaS – інфраструктура як послуга
MAE – середня абсолютна помилка
MAPE – середня абсолютна відсоткова помилка
MDM – управління основними даними
ML – машинне навчання
OPEX – операційні витрати
OTD – своєчасність поставок
PaaS – платформа як послуга
POS – точка продажу, касовий термінал
ROI – рентабельність інвестицій
SaaS – програмне забезпечення як послуга
SAP – виробник корпоративного програмного забезпечення
SCM – управління ланцюгами постачання
SKU – одиниця складського обліку, номенклатурна позиція
SRM – управління взаємовідносинами з постачальниками
TMS – система управління транспортом
VRP – задача маршрутизації транспортних засобів
WMS – система управління складом

ВСТУП

Мережеві ресторани, як частина глобальної індустрії харчування, стикаються з численними викликами у сфері управління ланцюгами постачання. Ефективне управління запасами та контроль над термінами придатності продукції є одними з ключових факторів успішного функціонування ресторанного бізнесу. Однак на практиці підприємства часто стикаються з непередбачуваністю попиту, складністю моніторингу термінів придатності та координації логістичних процесів. Ці фактори можуть призводити до значних фінансових втрат, перевитрат ресурсів та зниження загальної ефективності

В умовах постійної конкуренції на ринку та зростаючих очікувань споживачів мережеві ресторани змушені шукати нові підходи до управління ланцюгами постачання. Особливої уваги заслуговують інноваційні технології, які дозволяють автоматизувати процеси, інтегрувати дані в реальному часі та впроваджувати системи аналітики для прийняття обґрунтованих управлінських рішень. Важливими аспектами є також адаптивність до змін у ринковому середовищі, забезпечення високого рівня якості продукції та зниження операційних витрат.

Ця тема є актуальною, оскільки сучасні ресторанні мережі працюють у динамічних умовах, де попит на продукцію змінюється залежно від безлічі факторів, таких як сезонність, маркетингові кампанії, зміна споживчих уподобань та вплив зовнішніх криз. Для збереження конкурентоспроможності необхідно впроваджувати інтегровані системи управління, які враховують специфіку ресторанної індустрії та дозволяють досягти оптимального балансу між витратами, якістю та швидкістю обслуговування.

У цій роботі розглядаються основні виклики, з якими стикаються мережеві ресторани, перспективи розвитку інноваційних підходів до управління ланцюгами постачання, а також практичні рішення для підвищення ефективності бізнесу. Зокрема, досліджено концептуальну модель інтегрованої системи управління

запасами, яка поєднує в собі використання сучасних інформаційних технологій, автоматизації. Запропонована модель є основою для створення більш досконалих і адаптивних систем управління, здатних задовольнити вимоги сучасного ринку.

Традиційні методи контролю строків придатності та управління запасами вимагають багато часу та людських ресурсів. Автоматизовані рішення дозволяють персоналу зосередитися на інших завданнях, що покращує продуктивність і зменшує ризики помилок.

Також у цій роботі розглядається екологічний вплив так як значна частина харчових продуктів втрачається через прострочення терміну придатності, що призводить до фінансових втрат і негативно впливає на екологію. Автоматизовані системи відстеження запасів дозволяють уникати цих втрат.

За даними Агентства з охорони довкілля США (EPA), лише в 2019 році в США було створено близько 66,2 млн. тон харчових відходів в закладах громадського харчування, при чому 40% із них припадає на ресторани. Більшість із цих відходів потрапляє на звалища що негативно впливає на довкілля, сприяючи викидам парникових газів через розкладання органічних матеріалів у великих масштабах. [33]

Запропонована система може допомогти значно скоротити обсяги таких відходів, оскільки автоматизація замовлень дозволить уникати надлишкового запасу продуктів та зменшувати кількість втрат через недбале управління замовленнями. Це у свою чергу, не тільки знизить витрати на утилізацію, але й зробить внесок у боротьбу зі зміною клімату та більш екологічне ведення бізнесу.

1 АНАЛІЗ ІСНУЮЧИХ СИСТЕМНИХ РІШЕНЬ І ФОРМУЛЮВАННЯ ПРОБЛЕМИ

1.1 Концепції та особливості управління каналами постачання продуктів у сфері HoReCa

Ефективне управління ланцюгами постачання розглядається як критично важлива складова успішного функціонування мережевих закладів ресторанного господарства [6, 7]. Сучасна індустрія громадського харчування послуговується широким спектром технологічних рішень та методичних підходів, застосування яких є необхідним для вирішення завдань координації, оптимізації управління запасами та мінімізації втрат, спричинених псуванням продукції з обмеженим терміном зберігання.

Системи управління ланцюгами постачання (SCM) формують методологічну та технологічну основу для ефективного контролю за процесами закупівель, виробництва, складського зберігання, транспортування, дистрибуції та реалізації кінцевої продукції. Це забезпечує цілісну оптимізацію функціонування всього логістичного ланцюга [8, 9, 10]. Шляхом інтеграції передових інформаційних технологій SCM-системи надають можливість централізованого та комплексного управління всіма елементами ланцюга постачання — від постачальників до кінцевих споживачів.

Сфера HoReCa характеризується наявністю суттєвих відмінностей в організації управління постачанням у порівнянні з традиційними секторами роздрібної торгівлі або виробничої діяльності, що обумовлює специфічні вимоги до програмно-методичних рішень. Ключові операційні особливості, які потребують відокремленого розгляду, визначаються низкою факторів.

По-перше, це скорочений життєвий цикл продуктів. Основний обсяг сировинної бази класифікується як швидкопсувна продукція, що вимагає

застосування процедур максимально точного прогнозування та граничної мінімізації термінів складського зберігання. Будь-які дисфункції чи похибки в інвентаризаційному обліку неминуче призводять до прямого матеріального збитку.

По-друге, спостерігається висока варіативність показників попиту. Величина попиту демонструє значну кореляцію із зовнішніми факторами, включаючи сезонні коливання, метеорологічні умови, проведення локальних подій та вплив маркетингових заходів. Це значно ускладнює досягнення високої точності при прогнозуванні продуктових потреб.

По-третє, важливим фактором є складність структури номенклатури. Процес управління запасами передбачає необхідність відстеження не лише кінцевих інгредієнтів, але й проміжних напівфабрикатів, що спричиняє експоненційне збільшення кількості облікових одиниць.

По-четверте, характерною є фрагментація постачальницької бази. Функціонування закладів часто передбачає співпрацю з великою кількістю малих та середніх постачальників з метою забезпечення гарантованої якості та свіжості, що обумовлює необхідність застосування високогнучких інструментів інтеграції.

Управління каналами постачання в секторі HoReCa фокусується на досягненні оптимальних значень трьох провідних метрик: своєчасності поставок, точності інвентаризаційного обліку та мінімізації обсягів незворотних втрат. Структурний огляд ключових функціональних можливостей, стратегічних переваг, поширених реалізацій та технологічних принципів систем управління постачанням буде представлено в подальших підрозділах.

1.2 Огляд функціональності глобальних SCM-систем

Корпоративні системи управління ланцюгами постачання глобального рівня, до яких традиційно належать рішення від компаній SAP, Oracle та Blue Yonder, є

високопотужними програмно-технічними комплексами. Їхня функціональність орієнтована на обслуговування міжнародних корпорацій із пролонгованим циклом бізнес-процесів типу "бізнес для бізнесу" [1].

Одним із найбільш поширених рішень на ринку є продукт компанії SAP. Це комплексна система, інтегрована в екосистему виробника, призначена для управління ланцюгами постачання на всіх етапах. Ключовою перевагою є тісна інтеграція з модулями планування ресурсів підприємства, управління транспортуванням та складською логістикою. Така інтеграція створює єдиний інформаційний простір, що дозволяє координувати роботу різних підрозділів. Система пропонує інструменти для розширеного планування та оптимізації, які дозволяють здійснювати прогнозування попиту на основі історичних даних та ринкових трендів. Функціонал управління запасами забезпечує відстеження залишків у реальному часі та оптимізацію обсягів закупівель. Важливою складовою є також модуль управління взаємовідносинами з постачальниками, який автоматизує процеси закупівель та оцінки надійності партнерів. Незважаючи на широкі можливості, система має суттєві недоліки для малого та середнього бізнесу, зокрема високу вартість впровадження та складність налаштування, що вимагає залучення висококваліфікованих спеціалістів.

Іншим значним гравцем на ринку є хмарне рішення від Oracle. Ця платформа фокусується на інтеграції та автоматизації бізнес-процесів, використовуючи сучасні технології обробки даних. Вона забезпечує управління повним циклом постачання — від планування попиту до логістики та виконання замовлень. Система дозволяє синхронізувати виробничі процеси з потребами ринку, використовуючи інструменти аналізу сезонних коливань. Розвинений функціонал управління запасами допомагає уникати дефіциту або надлишків продукції шляхом автоматизованих сповіщень. Хмарна архітектура забезпечує масштабованість та доступність системи, однак, як і у випадку з SAP, орієнтація на великі корпорації робить це рішення надмірно складним та дорогим для використання в динамічному середовищі ресторанного бізнесу.

Також варто відзначити рішення від Blue Yonder, яке позиціонується як інструмент для оптимізації логістики та управління запасами з використанням елементів штучного інтелекту. Система пропонує засоби для прогнозування попиту, що враховують поведінку споживачів та зовнішні фактори. Модулі управління транспортом та запасами спрямовані на зменшення операційних витрат та оптимізацію маршрутів доставки.

Узагальнюючи аналіз глобальних систем, можна зробити висновок, що, незважаючи на потужний функціонал, усі вони демонструють певну надлишковість для потреб сфери HoReCa. Їхня архітектура та логіка роботи орієнтовані на стабільні, довготривалі процеси виробничих гігантів, а не на швидкі, щоденні транзакції ресторанних мереж. Висока вартість володіння та складність адаптації роблять їх малопридатними для вирішення поставлених у роботі завдань.

1.3 Аналіз архітектурних рішень систем управління ланцюгами постачання

Аналіз архітектурних підходів до побудови систем управління є важливим етапом дослідження, оскільки саме архітектура визначає здатність системи до масштабування та інтеграції. Сучасні тенденції вказують на перехід від монолітних структур до гнучких, розподілених рішень, що базуються на хмарних технологіях та мікросервісах [11, 12].

Основні переваги мікросервісної архітектури для систем управління постачанням включають:

- гнучкість та масштабованість окремих компонентів;
- можливість використання різних технологій для різних сервісів;
- незалежне розгортання та оновлення модулів;
- підвищена відмовостійкість системи.

Manhattan Associates SCM – це одне з провідних рішень у сфері управління ланцюгами постачання, яке спеціалізується на інтегрованих технологіях для роздрібної торгівлі, виробництва та логістики. Компанія відома своєю глибокою експертизою в оптимізації складських операцій, управлінні транспортом і розподілі товарів. Платформа надає сучасні інструменти для забезпечення ефективного функціонування ланцюгів постачання, орієнтуючись на автоматизацію, хмарні технології та аналітику даних[5] .

Однією з найсильніших сторін є її модуль управління складом WMS. Система пропонує передові алгоритми для оптимізації процесів приймання, зберігання, комплектування та відвантаження товарів. Завдяки використанню AI та ML, система забезпечує підвищення продуктивності складських операцій, скорочення часу виконання замовлень та покращення точності інвентаризації.

Модуль TMS дозволяє компаніям координувати логістичні операції та оптимізувати маршрути доставки. Завдяки інтеграції з картографічними сервісами та реєстрації в реальному часі, система забезпечує ефективне планування та відстеження перевезень. Вона також підтримує управління зворотними потоками, що є важливим для підприємств, які працюють у сфері електронної комерції.

Manhattan Associates пропонує розширені можливості для управління замовленнями, які дозволяють компаніям інтегрувати дані з усіх каналів продажів. Це рішення підтримує омніканальний підхід, забезпечуючи синхронізацію онлайн-замовлень, роздрібних продажів та складів. Інструменти OMS дозволяють автоматично обирати оптимальний склад або магазин для виконання замовлення, враховуючи фактори витрат і термінів доставки.

Система використовує аналітику даних для прогнозування змін у попиті, враховуючи сезонність, споживчі тренди та історичні дані. Цей модуль допомагає підприємствам ефективно планувати виробництво, уникати перевиробництва або дефіциту та знижувати витрати на утримання запасів.

Manhattan Associates активно інтегрує рішення IoT для автоматизації збору даних на кожному етапі ланцюга постачання. Наприклад, сенсори можуть

фіксувати температуру зберігання, переміщення вантажів та відстежувати статус доставки в реальному часі. Це забезпечує прозорість процесів і покращує точність прийняття рішень.

Система побудована на основі хмарної платформи Manhattan Active, яка забезпечує постійну доступність, швидке масштабування та автоматичне оновлення функціоналу. Платформа підтримує модульну архітектуру, що дозволяє підприємствам вибирати лише ті компоненти, які їм потрібні.

Один із ключових аспектів – це підтримка омніканальних стратегій. Система дозволяє легко синхронізувати дані між онлайн-магазинами, фізичними магазинами, складами та центрами обробки замовлень. Це забезпечує єдиний підхід до управління клієнтським досвідом і прискорює виконання замовлень.

Переваги:

Сильна спеціалізація у складській логістиці – потужний модуль WMS є одним із найкращих у галузі, що робить рішення ідеальним для компаній з великими складськими операціями.

Підтримка омніканальних стратегій – система забезпечує бездоганну інтеграцію даних з усіх каналів продажів, що підходить для сучасного роздрібного бізнесу.

Інновації та гнучкість – використання хмарної архітектури та інструментів AI забезпечує високу адаптивність до змін у бізнес-середовищі.

Прозорість та контроль – завдяки моніторингу в реальному часі та аналітичним звітам компанії отримують повну видимість усіх операцій.

Недоліки:

Висока вартість впровадження – комплексний функціонал і технології роблять систему досить дорогою, особливо для малого та середнього бізнесу.

Тривалий процес впровадження – складність налаштування та інтеграції може вимагати багато часу, що може бути проблемою для компаній, які прагнуть швидкого переходу.

Потреба в навчанні персоналу – через велику кількість функцій система може бути складною для нових користувачів, що вимагає додаткових інвестицій у навчання співробітників.

Manhattan Associates SCM є висококласним рішенням, яке орієнтоване на задоволення потреб великих компаній із складними ланцюгами постачання. Завдяки своїй інноваційності, модульній архітектурі та акценту на омніканальності, платформа дозволяє бізнесам підвищувати ефективність операцій і задовольняти потреби сучасних клієнтів.

Infor SCM [4] є одним із провідних рішень у сфері управління ланцюгами постачання, яке розроблене для підтримки підприємств у багатьох галузях, таких як виробництво, роздрібна торгівля, дистрибуція та логістика. Ця система пропонує широкий набір функцій, спрямованих на підвищення ефективності всього ланцюга постачання, починаючи від планування і закінчуючи доставкою кінцевому споживачеві. Вона використовує сучасні технології, такі як штучний інтелект, машинне навчання, IoT, а також хмарні обчислення, що дозволяє підприємствам адаптуватися до швидких змін ринку та вимог клієнтів.

Основною перевагою є її здатність інтегрувати всі етапи управління ланцюгами постачання в єдину систему. Завдяки цьому забезпечується прозорість процесів і полегшується прийняття рішень на основі даних. Система містить потужні інструменти для прогнозування попиту, які дозволяють компаніям точно оцінювати майбутню потребу в продуктах і знижувати ризик дефіциту або надлишкових запасів. Ці інструменти базуються на аналізі історичних даних, поточних трендів і сезонних коливань. Крім того, система підтримує функції управління запасами, які дозволяють контролювати рівень товарів у реальному часі, автоматизувати процеси поповнення та мінімізувати втрати, пов’язані з надлишковими запасами або псуванням продукції.

Ще однією ключовою особливістю є її можливості в управлінні транспортом. Ця функція дозволяє компаніям планувати маршрути доставки, оптимізувати транспортні витрати, координувати логістичні операції та відстежувати статус

доставки в реальному часі. Використання цих функцій допомагає підприємствам скорочувати витрати на логістику та забезпечувати своєчасне виконання замовлень. Для компаній, що працюють у сфері роздрібної торгівлі, система пропонує інструменти управління виконанням замовлень. Цей модуль дозволяє інтегрувати різні канали продажу, забезпечуючи синхронізацію між складами, магазинами та логістичними центрами. Завдяки цьому клієнти можуть отримувати свої замовлення швидко й без затримок, що підвищує рівень їхньої задоволеності.

Infor SCM також активно використовує можливості IoT. Завдяки інтеграції з IoT-сенсорами система може збирати дані про стан продукції, її розташування та умови зберігання в реальному часі. Це дозволяє компаніям забезпечувати прозорість ланцюга постачання, своєчасно реагувати на зміни та мінімізувати ризики, пов’язані з порушенням умов транспортування або зберігання товарів. Крім того, платформа підтримує аналітичні інструменти, які допомагають виявляти проблемні зони, оптимізувати процеси та прогнозувати потенційні ризики. Ці інструменти базуються на штучному інтелекті, що дозволяє компаніям приймати більш обґрунтовані та швидкі управлінські рішення.

Хмарна архітектура є ще однією важливою складовою Infor SCM. Система працює на основі платформи Infor CloudSuite, яка забезпечує доступ до даних із будь-якого пристрою та дозволяє компаніям швидко масштабувати свої операції. Хмарна технологія також допомагає знижувати витрати на IT-інфраструктуру, адже підприємствам не потрібно витрачатися на встановлення та підтримку локальних серверів. Крім того, це забезпечує гнучкість бізнес-процесів і дозволяє компаніям швидко реагувати на зміни в ринковому середовищі. Завдяки інтеграції з ERP-системами та CRM-рішеннями, Infor SCM створює єдине інформаційне середовище для управління бізнесом, що дозволяє об'єднати всі ключові процеси компанії.

Втім, як і будь-яке програмне забезпечення, Infor SCM має певні недоліки. Одним із них є висока вартість впровадження та підтримки, що може бути проблемою для малого та середнього бізнесу. Крім того, процес інтеграції системи

з іншими корпоративними рішеннями може виявитися складним і вимагати залучення зовнішніх фахівців. Ще однією проблемою є залежність роботи системи від якості даних. Якщо вхідні дані будуть неповними або неточними, це може вплинути на ефективність роботи модулів штучного інтелекту та аналітики. Навчання персоналу також потребує значних зусиль і часу, оскільки система має великий обсяг функцій і складний інтерфейс.

Infor SCM залишається одним із провідних рішень для управління ланцюгами постачання. Її гнучкість, функціональність та інноваційні технології роблять її привабливим вибором для компаній, які прагнуть покращити свої бізнес-процеси, оптимізувати витрати та забезпечити високу якість обслуговування клієнтів. Система є ідеальним вибором для підприємств, які працюють у швидко змінюваних ринкових умовах та прагнуть забезпечити стабільний розвиток свого бізнесу.

1.4 Аналітичний огляд спеціалізованих систем управління запасами для ресторанного бізнесу

Спеціалізовані програмні рішення, орієнтовані на індустрію HoReCa, демонструють значно вищий рівень адаптованості до операційних потреб закладів громадського харчування, ніж глобальні SCM-системи. Вони зосереджені на внутрішньому контурі управління – від складу інгредієнтів до продажу готової страви. Ключові функціональні можливості таких систем включають управління рецептами безперервну інвентаризацію та детальний облік втрат і списань. До провідних рішень цього сегмента належать Restaurant365, MarketMan.

Restaurant365 [3] — це хмарна система, створена спеціально для потреб ресторанного бізнесу [13]. Вона поєднує функції бухгалтерії, інвентаризації, планування, обліку витрат і аналітики в єдиному рішенні, що дозволяє рестораторам керувати операційними та фінансовими процесами.

Основний функціонал системи включає управління запасами та закупівлями. Вона дозволяє точно відстежувати залишки інгредієнтів у режимі реального часу, надаючи інструменти для автоматичного прогнозування потреб у продуктах. Інтеграція з постачальниками дає можливість автоматизувати процес замовлення та моніторинг його виконання. Також передбачено управління термінами придатності, що мінімізує втрати через псування продуктів.

Важливою функцією є управління рецептами та собівартістю. Система підтримує створення та управління рецептами страв, що забезпечує точний облік витрат на інгредієнти і дає можливість контролювати собівартість кожної позиції в меню. Платформа автоматично оновлює вартість страв при зміні цін на інгредієнти.

Фінансовий модуль та модуль управління персоналом також є частиною системи. Ключовою особливістю є інтеграція інвентаризації з модулем фінансового обліку (облік витрат, нарахування заробітної плати, відстеження грошових потоків). Додатково система пропонує функції автоматизації складання графіків роботи персоналу з урахуванням прогнозованих обсягів продажів.

До сильних сторін Restaurant365 можна віднести високий ступінь автоматизації, централізацію управління, масштабованість для великих мереж та фінансову аналітику. Серед недоліків виділяють досить складний і тривалий процес впровадження, високу вартість підписки, а також інтерфейс, який може бути перевантаженим для користувача. Система сконцентрована на обліку та фінансовому контролі і не пропонує повноцінного функціоналу транспортного менеджменту або складського менеджменту для оптимізації міжскладської логістики великих мереж.

MarketMan — це хмарна платформа[2], орієнтована на управління запасами, витратами, закупівлями та постачальниками для ресторанів та закладів громадського харчування [14]. Система позиціонується як інструмент для оптимізації витрат і підвищення ефективності управління поставками.

Основний функціонал:

- Автоматизація закупівель ця ключова особливість MarketMan полягає у можливості автоматизації замовлень. Система аналізує дані про використання продуктів, прогнозує потреби на основі продажів та автоматично створює замовлення постачальникам. Дозволяє користувачам підключати постачальників, завантажувати прайс-листи, порівнювати ціни та вибирати найвигідніші пропозиції.
- Інвентаризація та мобільність: пропонує мобільний додаток для зручного проведення фізичної інвентаризації на складі за допомогою сканування штрих-кодів. Це забезпечує швидкість і точність обліку залишків у режимі реального часу.
- Контроль витрат дозволяє точно відслідковувати собівартість страв завдяки інтеграції з рецептами та POS-системами, виявляючи відхилення фактичної собівартості від планової (food cost variance).

Переваги MarketMan: інтуїтивно зрозумілий інтерфейс, сильний акцент на автоматизації замовлень та контролі закупівельних цін, мобільність.

Недоліки MarketMan: обмежений функціонал у сфері повноцінного фінансового обліку порівняно з Restaurant365. Як і більшість аналогів, MarketMan орієнтований на закупівля та прийом і не містить розвинених інструментів для складного логістичного планування, маршрутизації та управління власним розподільчим центром.

Спеціалізовані системи для HoReCa успішно вирішують ключові проблеми, пов'язані з контролем собівартості та управлінням запасами інгредієнтів. Проте їхня архітектура та функціонал обмежені внутрішнім контуром управління (від складу до продажу) і не охоплюють повний логістичний ланцюг SCM (потреби філії → замовлення → консолідація → центральний розподільчий центр → оптимізована доставка → філія). Жодна з них не пропонує повноцінного модуля транспортного менеджмента або складського менеджмента з розширеною функціональністю, необхідного для централізованого управління поставками в

рамках великої мережі. Це створює суттєвий функціональний розрив, який є предметом подальшого дослідження

1.5 Узагальнення типових функціональних можливостей та архітектурних рішень сучасних систем

Проведений аналіз глобальних корпоративних систем управління постачанням (SAP, Oracle, Blue Yonder) та спеціалізованих рішень для ресторанного бізнесу (Restaurant365 [3], MarketMan [2]) дозволяє виділити спільні та диференційні функціональні та архітектурні тренди. Сучасні системи управління ланцюгами постачання, незалежно від галузевої спеціалізації, будуються навколо ключових функціональних блоків та спільної архітектурної парадигми, орієнтованої на хмарні обчислення та гнучкість.

Функціональність систем управління постачанням традиційно поділяється на три основні категорії, які відображають стратегічні та операційні завдання ланцюга постачання: планування, виконання та забезпечення.

Блок планування орієнтований на стратегічне та тактичне управління. Його основна мета – максимізувати точність прогнозування потреб ринку, що включає модулі прогнозування попиту з урахуванням сезонності та акцій, а також планування запасів для визначення оптимальних точок перезамовлення і страхових запасів. Для рішень у сфері ресторанного бізнесу критично важливим є додавання функціоналу управління рецептами, який забезпечує точний розрахунок потреби в сировині для виробництва кінцевої продукції.

Блок виконання відповідає за операційну логістику та фізичне переміщення товарів. Цей сегмент традиційно включає потужні системи управління складом для оптимізації внутрішньоскладських операцій (прийом, розміщення, відбір, відвантаження) та системи управління транспортом для планування рейсів, оптимізації маршрутів і відстеження доставки в реальному часі (Global Positioning

System). Крім того, до цього блоку належить управління замовленнями, яке забезпечує обробку та виконання замовлень від кінцевих клієнтів або внутрішніх філій.

Блок забезпечення охоплює інфраструктурні та аналітичні функції. Він включає аналітику та звітність для вимірювання ключових показників результативності, таких як собівартість, своєчасність та точність інвентаризації, а також управління взаємовідносинами з постачальниками, що автоматизує закупівлі та управління контрактами. Додатково цей блок часто інтегрує мобільні рішення та технології інтернету речей для збору даних у режимі реального часу, мобільної інвентаризації та контролю температурного режиму.

З архітектурної точки зору домінуючою парадигмою для систем управління постачанням є перехід до хмарних та розподілених структур. Хмарна орієнтація, що використовується у переважній більшості сучасних рішень (Oracle, Manhattan, MarketMan), забезпечує високу масштабованість для великих мереж, швидке розгортання та безперервне оновлення функціоналу. Застосування мікросервісної архітектури дозволяє підприємствам обирати лише необхідні незалежні компоненти, що спрощує інтеграцію з існуючими системами планування ресурсів та касовими терміналами, а також забезпечує гнучкість у розробці та підтримці.

Критичним висновком аналізу є те, що спеціалізовані системи для ресторанного бізнесу успішно покривають функціонал планування та внутрішньої інвентаризації, але демонструють суттєвий дефіцит у блоці виконання. Ці системи орієнтовані на інвентаризаційний облік і контроль собівартості, але ігнорують необхідність у повноцінному управлінні складом та транспортом з розширеною функціональністю, необхідного для централізованого управління поставками та оптимізацією маршрутів доставки між власним розподільчим центром і кінцевими філіями. Цей функціональний розрив стає основною передумовою для подальшого дослідження та розробки методики.

1.6 Постановка проблеми та основних завдань дослідження

Комплексний аналіз наявних інформаційних систем, спрямованих на управління ланцюгами постачання, виявив критичний функціональний та методологічний розрив у сфері мережевого ресторанного бізнесу. Цей розрив зумовлений нездатністю існуючих рішень поєднати високоточне прогнозування попиту із розвиненими засобами оптимізації логістичного виконання, які часто є надлишковими у глобальних платформах, але критично необхідними для роботи зі швидкопсувною продукцією.

Проблема дослідження полягає у відсутності інтегрованої науково обґрунтованої методики, яка поєднувала б принципи інженерії програмного забезпечення (моделювання, алгоритми оптимізації) та теорії управління запасами для забезпечення максимальної ефективності управління каналами постачання в умовах високої змінності попиту та жорстких вимог до свіжості продуктів.

На сьогодні функціонування мережевих закладів ресторанного господарства, що використовують застарілі облікові платформи, характеризується такими ключовими недоліками:

– низька точність прогнозування потреб філій, оскільки існуючі системи часто базуються на лінійному аналізі історичних даних і не враховують динамічних факторів впливу (сезонність, події, погодні умови) або складних залежностей між продажами готових страв і потребою в базових інгредієнтах, що спричиняє або надлишкові замовлення та високий рівень списання, або дефіцит і втрачені продажі;
– відсутність оптимізованої логістики «останньої милі», коли за наявності власного розподільчого центру мережі стикаються з проблемою нераціональної маршрутизації та комплектування багатоадресних замовлень, а спеціалізовані

ресторанні системи не містять модулів транспортного менеджменту, здатних динамічно будувати оптимальні маршрути доставки;

– недостатня прозорість взаємодії з постачальниками через відсутність автоматизованих інструментів для постійної оцінки надійності, своєчасності та якості постачання від численних зовнішніх суб'єктів, що ускладнює зниження ризиків у ланцюзі постачання.

Мета роботи: підвищення ефективності управління каналами постачання продуктів системі мережевого ресторанного бізнесу шляхом впровадження автоматизованого планування та проактивного контролю.

Для досягнення поставленої мети необхідно вирішити такі основні завдання:

1. Провести всебічний аналіз сучасних моделей управління ланцюгами постачання та спеціалізованих систем управління запасами у сфері ресторанного бізнесу, ідентифікувавши їхні ключові архітектурні та функціональні недоліки стосовно потреб мережевих закладів;
2. Розробити адаптовану модель прогнозування попиту на інгредієнти, що враховує динамічні фактори впливу та складну структуру номенклатури (рецептурний зв'язок між готовою стравою та сировиною);
3. Обґрунтувати та розробити інструментарій для оптимізації процесів логістичного виконання, включаючи методику розв’язання задачі маршрутизації транспортних засобів для забезпечення своєчасної доставки з розподільчого центру до філій;
4. Сформувати комплексну інтегровану методику управління каналами постачання, що забезпечує автоматичну генерацію замовлень на основі прогнозу та оптимізацію їхньої консолідації й доставки;
5. Провести експериментальне дослідження та економічне обґрунтування запропонованої методики на статистичних даних реальної ресторанної мережі з метою кількісної оцінки потенційного економічного ефекту від зниження рівня списань та логістичних витрат.

Об’єкт дослідження: процеси управління каналами постачання продуктів в системі мережевого ресторанного бізнесу.

Предмет дослідження: практичні інструменти та підходи спрямовані на підвищення ефективності управління каналами постачання продуктів в системі мережевого ресторанного бізнесу.

У роботі використано такі методи дослідження:

1. Економіко-статистичний аналіз — для оцінки наявного стану ефективності управління запасами та валідації отриманих результатів;
2. Теорія управління запасами (зокрема моделі з фіксованим інтервалом та рівнем запасу) — для розробки оптимальних параметрів замовлення інгредієнтів;
3. Методики моделювання бізнес-процесів — для формалізації та подальшої оптимізації процесів закупівель та дистрибуції;
4. Методи дослідження операцій та комбінаторна оптимізація — для розробки алгоритмів динамічної маршрутизації транспортних засобів;
5. Системний підхід та інженерія програмного забезпечення — для інтеграції розроблених моделей та алгоритмів у єдину методику.

Успішна реалізація поставлених завдань дозволить розробити універсальну та масштабовану методику, яка може бути використана як основа для програмного рішення, що ефективно управляє складним ланцюгом постачання швидкопсувної продукції.

Комплексний аналіз існуючих інформаційних систем, спрямованих на управління ланцюгами постачання (SCM), виявив наявність критичного функціонального та методологічного розриву у сфері мережевого ресторанного бізнесу (HoReCa). Цей розрив визначається нездатністю наявних рішень адекватно інтегрувати високоточне прогнозування попиту, властиве спеціалізованим

HoReCa-системам, із розвиненими засобами оптимізації логістичного виконання, які є надлишковими у глобальних SCM-платформах.

Проблема дослідження полягає у відсутності інтегрованої науково-обґрунтованої методики, яка б поєднувала принципи інженерії програмного забезпечення (моделювання, алгоритми оптимізації) та теорії управління запасами, для забезпечення максимальної ефективності управління каналами постачання швидкопсувної продукції в умовах високої варіативності попиту та жорстких вимог до свіжості.

На поточний момент функціонування мережевих закладів HoReCa характеризується наступними ключовими недоліками, що підлягають усуненню:

1. Низька точність прогнозування потреб філій: існуючі системи часто базуються на простих історичних даних і не враховують динамічних факторів впливу (сезонність, події, погодні умови) або ж складних залежностей між продажами готових страв і потребою в базових інгредієнтах. Це спричиняє або надлишкові замовлення та, як наслідок, високий рівень списання (waste), або дефіцит та втрачені продажі.

2. Відсутність оптимізованої логістики "останньої милі": при наявності власного розподільчого центру (РЦ) мережі стикаються з проблемою неоптимізованої маршрутизації та комплектування багатоадресних замовлень. Спеціалізовані HoReCa-системи не містять модулів транспортного менеджменту (TMS), здатних динамічно будувати оптимальні маршрути доставки, що призводить до зростання транспортних витрат, збільшення часу доставки та порушення температурних режимів.

3. Недостатня прозорість взаємодії з постачальниками: відсутність автоматизованих інструментів для постійної оцінки надійності, своєчасності та якості поставок від численних зовнішніх суб'єктів ускладнює зниження ризиків у ланцюзі постачання.

Мета кваліфікаційної роботи — підвищення ефективності управління каналами постачання продуктів системі мережевого ресторанного бізнесу шляхом впровадження автоматизованого планування та проактивного контролю.

Для досягнення поставленої мети необхідно вирішити наступні основні завдання:

1. Провести всебічний аналіз сучасних моделей SCM та спеціалізованих систем управління запасами у сфері HoReCa, ідентифікуючи їхні ключові архітектурні та функціональні недоліки стосовно потреб мережевого бізнесу.

2. Розробити адаптовану модель прогнозування попиту на інгредієнти, що враховує динамічні фактори впливу та складну структуру номенклатури (рецептурний зв'язок між готовою стравою та сировиною).

3. Обґрунтувати та розробити інструментарій для оптимізації процесів логістичного виконання, включаючи методику багатоадресної маршрутизації (VRP - Vehicle Routing Problem) для забезпечення своєчасної доставки з РЦ до філій.

4. Сформувати комплексну інтегровану методику управління каналами постачання, що забезпечує автоматичну генерацію замовлень на основі прогнозу та оптимізацію їхньої консолідації й доставки.

5. Провести експериментальне дослідження та економічне обґрунтування запропонованої методики на статистичних даних реальної ресторанної мережі з метою кількісної оцінки потенційного економічного ефекту від зниження рівня списань та логістичних витрат.

Об’єкт дослідження: процеси управління каналами постачання продуктів в системі мережевого ресторанного бізнесу.

Предмет дослідження: практичні інструменти та підходи спрямовані на підвищення ефективності управління каналами постачання продуктів в системі мережевого ресторанного бізнесу

У роботі використано такі методи дослідження:

1. Економіко-статистичний аналіз — для оцінки наявного стану ефективності управління запасами та валідації отриманих результатів.
2. Теорія управління запасами (зокрема, моделі з фіксованим інтервалом та рівнем запасу) — для розробки оптимальних параметрів замовлення інгредієнтів.
3. Методики моделювання бізнес-процесів — для формалізації та подальшої оптимізації процесів закупівель та дистрибуції.
4. Методи дослідження операцій та комбінаторна оптимізація — для розробки алгоритмів динамічної маршрутизації транспортних засобів (TMS-функціонал).
5. Системний підхід та інженерія програмного забезпечення — для інтеграції розроблених моделей та алгоритмів у єдину методику.
6. Успішна реалізація поставлених завдань дозволить розробити універсальну та масштабовану методику, яка може бути використана як основа для програмного рішення, що ефективно управляє складним ланцюгом постачання швидкопсувної продукції.

2 ДОСЛІДЖЕННЯ ПРОБЛЕМНИХ ПАРАМЕТРІВ ПОСТАЧАННЯ ТА ІНВЕНТАРИЗАЦІЇ У СФЕРІ HORECA

2.1. Проблема автоматизації управління постачанням з акцентом на своєчасність поставок

Проблема автоматизації управління постачанням у системі мережевого ресторанного бізнесу є багатоаспектною інженерною задачею, яка вимагає інтеграції принципів управління ланцюгами постачання та передових методів інженерії програмного забезпечення. Основний акцент у цьому сегменті бізнесу зміщується на своєчасність поставок, що є критично важливим фактором для забезпечення безперебійного виробничого процесу (приготування страв) та мінімізації фінансових втрат.

На відміну від традиційного ритейлу, управління ланцюгами постачання в ресторанній сфері оперує з високодинамічними змінними та жорсткими часовими обмеженнями, обумовленими біологічним терміном придатності продуктів. Невчасна або неточна поставка може мати два деструктивні наслідки: – дефіцит, що полягає у відсутності ключового інгредієнта у філії та призводить до зупинки продажів популярних страв, прямих фінансових втрат і зниження рівня задоволеності клієнтів; – надлишок, який характеризується перевищенням фактичної потреби (особливо швидкопсувними продуктами) та неминуче призводить до їхнього списання [15].

Згідно з дослідженнями, витрати, пов’язані зі списанням продуктів, можуть становити до 8-10 % від загальної вартості закупленої сировини у неоптимізованих системах [16].

Ідентифікація ключових технічних викликів Автоматизація управління постачанням з гарантуванням своєчасності стикається з низкою специфічних викликів, які потребують програмно-інженерного вирішення:

1. Неточне прогнозування попиту, оскільки традиційні системи обліку часто не враховують мультифакторну природу попиту. Для інженерного вирішення цієї проблеми необхідна розробка предиктивної моделі, здатної встановлювати зв'язок між продажами готових страв і потребою у базових інгредієнтах (рецептурна декомпозиція), а також динамічно інтегрувати зовнішні фактори (погода, свята, маркетингові акції) у процес прогнозування [16].
2. Проблема управління гетерогенними даними, оскільки управління постачанням у мережі вимагає синхронізації даних із трьох основних, часто несумісних джерел, таких як точки продажу (генерують дані про фактичний попит), системи інвентаризації (відстежують залишки та списання) та зовнішні постачальники (надають прайс-листи, статуси замовлень та графіки поставок). Автоматизація вимагає створення інтеграційного шару, який уніфікує ці дані та забезпечує їхню достовірність у реальному часі [17].
3. Складність логістики «останньої милі», що для мереж із власним розподільчим центром полягає в необхідності мінімізації загального пробігу транспортних засобів та часу доставки при обслуговуванні кількох філій за один рейс. При цьому мають враховуватися обмеження місткості транспорту та часові вікна доставки [18]. Ця проблема математично формулюється як задача маршрутизації транспортних засобів, що є типовою задачею комбінаторної оптимізації в інженерії програмного забезпечення.
4. Інженерний підхід до вирішення проблеми своєчасності поставок з погляду інженерії програмного забезпечення підвищення ефективності управління постачанням досягається (яка оцінюється за трьома ключовими показниками: своєчасність поставок, рівень списання продукції та логістичні витрати) шляхом реалізації трьох взаємопов'язаних модулів, які формують основу розроблюваної методики.

Розроблювальна методика:

1. Модуль прогнозування та планування, що відповідає за перетворення даних про продажі у прогнозовані потреби в інгредієнтах та автоматично генерує оптимальні пропозиції щодо замовлень, використовуючи моделі машинного навчання та статистичний аналіз;
2. Модуль виконання та логістики, який відповідає за вирішення задачі маршрутизації транспортних засобів, тобто динамічно створює оптимальні маршрути доставки з розподільчого центру до філій, забезпечуючи безпосередню своєчасність доставки;
3. Модуль контролю та інтеграції, що забезпечує надійний обмін даними між усіма елементами системи управління ланцюгами постачання, моніторинг ключових показників ефективності (зокрема фактичної своєчасності та відсотка списань) і оцінку постачальників.

Таким чином, автоматизація управління постачанням вимагає не простого програмного обліку, а створення інтелектуальної системи, де наскрізна обробка даних і математична оптимізація використовуються для мінімізації ризиків та підвищення операційної ефективності мережі.

2.2. Аналіз процесів обробки даних та формування замовлень з урахуванням точності інвентаризації запасів

Точність інвентаризації запасів є фундаментальною передумовою для успішної автоматизації управління постачанням. В умовах мережевого ресторанного бізнесу, де номенклатура включає тисячі інгредієнтів із коротким терміном придатності, будь-яке відхилення між фізичною наявністю товару та його обліком у системі (так звана розбіжність) призводить до формування неоптимальних замовлень, надмірних списань або дефіциту.

Архітектура даних та джерела інформації Для забезпечення високої точності інвентаризації та автоматизованого формування замовлень необхідно створити єдину централізовану архітектуру даних, яка агрегує інформацію з усіх точок ланцюга постачання. Ключовими джерелами даних є: – касові системи (POS), які виступають первинним джерелом даних про фактичний попит і споживання, де кожна транзакція продажу готової страви (наприклад, «Цезар-салат») повинна негайно викликати автоматичне списання інгредієнтів відповідно до технологічної карти (рецептури), що забезпечує принцип «вічної інвентаризації»; – системи складського обліку, що фіксують фактичне приймання товару, переміщення між складами (міжфіліальна логістика) та списання через псування, крадіжку або виробничі потреби; – система управління взаємовідносинами з постачальниками, яка містить актуальні прайс-листи, умови та час виконання замовлень, а також історію надійності кожного постачальника.

Ядром системи має стати централізована база даних, яка використовує моделі даних, що підтримують ієрархічну структуру запасів: від базового інгредієнта до кінцевої страви, включаючи проміжні напівфабрикати.

Процес обробки даних інвентаризації Ефективна обробка даних, необхідна для формування замовлень, включає забезпечення точності інвентаризації та формування чистого споживання. Точність інвентаризації визначається як відношення кількості облікових одиниць, для яких фактична фізична кількість збігається з кількістю в системі, до загальної кількості облікових одиниць (2.1):

$$IA = \frac{N_{\text{збіг}}}{N_{\text{загальна}}}, \tag{2.1}$$

де $N_{\text{збіг}}$ – кількість облікових одиниць з нульовим відхиленням;

$N_{\text{загальна}}$ – загальна кількість облікових одиниць.

Для підвищення точності інвентаризації необхідно впровадити автоматизацію списання через інтеграцію з касовою системою та рецептурним модулем для списання інгредієнтів у момент продажу, що мінімізує ручні помилки, а також циклічний перерахунок, який передбачає щоденну або щотижневу перевірку критично важливих груп товарів замість рідкісних повних інвентаризацій. Для точного планування замовлення необхідно обчислити чисте споживання продукту за певний період, що виключає втрати та списання (2.2):

$$\text{ЧС} = (\text{З}_{\text{поч}} + \text{Н}) - (\text{З}_{\text{кін}} + \text{С}), \quad (2.2)$$

де ЧС – чисте споживання;

$\text{З}_{\text{поч}}$– початковий залишок;

Н – надходження товару;

$\text{З}_{\text{кін}}$– кінцевий залишок;

С – списання.

Автоматизований розрахунок чистого споживання дозволяє отримати надійні історичні дані, які є вхідним вектором для прогнозних моделей [19].

Алгоритм автоматизованого формування замовлень На основі чистих даних про споживання та високої точності інвентаризації система автоматично генерує пропозиції щодо замовлень, використовуючи принципи управління запасами з фіксованим інтервалом, що є поширеним у сфері ресторанного бізнесу через необхідність регулярної доставки. Ключовим параметром є рівень поповнення, який визначається як сума прогнозованого попиту за час циклу та страхового запасу. Час циклу включає час між замовленнями та час виконання замовлення постачальником.

Алгоритм генерації замовлення складається з таких етапів:

- Визначення потреби, коли система обчислює необхідну кількість товару Q для поповнення запасів до рівня поповнення R (2.3):

$$Q = R - З_{\text{поточ}}\ , \tag{2.3}$$

де R – рівень поповнення;

$З_{\text{поточ}}$ – поточний залишок.

- Розрахунок страхового запасу з урахуванням надійності для мінімізації ризиків дефіциту, викликаного непередбачуваним попитом або затримками постачальника. У розроблюваній методиці пропонується динамічно коригувати страховий запас залежно від історичної надійності (своєчасності поставок) конкретного постачальника (2.4):

$$SS = Z \cdot \sigma T + LT\ , \tag{2.4}$$

SS – страховий запас;

Z– коефіцієнт рівня обслуговування;

$\sigma T + LT$ – середньоквадратичне відхилення попиту за час циклу;

LT – час виконання замовлення;

- Консолідація замовлень, яка відбувається після розрахунку необхідної кількості для всіх інгредієнтів. Система об'єднує їх в єдине замовлення з урахуванням мінімального обсягу замовлення від постачальника та оптимізації під ціну (порівняння цін різних постачальників для однієї товарної позиції);
- Створення електронного замовлення, яке автоматично передається в модуль управління взаємовідносинами з постачальниками для подальшого відправлення контрагенту.

Важливим інженерним аспектом є те, що показник часу виконання замовлення LT повинен динамічно оновлюватися, враховуючи відхилення своєчасності поставок контрагента. Якщо постачальник систематично затримує поставки, цей показник збільшується, що автоматично підвищує необхідний страховий запас і розмір замовлення, захищаючи філію від дефіциту.

Таким чином, розроблена методика інтегрує високу точність інвентаризації, отриману через автоматизоване списання, з алгоритмами управління запасами, які динамічно враховують ризики, пов'язані з ненадійністю каналів постачання [20]. Це забезпечує мінімізацію ручної праці та підвищує якість вхідних даних для подальших процесів оптимізації.

2.3. Проблеми архітектурних недоліків при інтеграції даних у поточних SCM-рішеннях

Успіх впровадження інтегрованої методики прогнозування та оптимізації безпосередньо залежить від якості, доступності та цілісності вхідних даних. Аналіз архітектурних рішень, представлених у першому розділі, виявив суттєві архітектурні бар’єри на шляху централізованого збору та аналізу даних, які є критичними для інтелектуального аналізу даних та точного планування.

Обмеження володіння даними та технологічна залежність від виробника Ключовим архітектурним недоліком у великих корпоративних системах управління ланцюгами постачання (наприклад, рішення від SAP або Oracle) є проблема володіння даними та їхньої реальної портативності. Хоча ці платформи зазвичай надають транзакційні інтерфейси (API) для оперативних потреб, вони не призначені для легкого, повного та структурованого вивантаження всього історичного набору «сирих» даних. Інформація, яка є критично важливою для побудови власних, незалежних моделей машинного навчання, фактично залишається ізольованою в закритій схемі виробника [21].

Ця технологічна залежність має два основні наслідки:

- Блокування інновацій, коли компанії змушені користуватися виключно вбудованими інструментами прогнозування, які можуть бути посередніми або неадаптованими до специфіки високодинамічного попиту ресторанного бізнесу;

- Технічна залежність, оскільки відсутність повного доступу до сирої історії продажів і споживання блокує впровадження інноваційних, кастомізованих алгоритмів.

Проблеми сумісності та управління основними даними на рівні взаємодії різних систем у ланцюгу постачання виникають дві взаємопов'язані архітектурні проблеми: відсутність стандартів сумісності та криза управління основними даними. Відсутність універсальних протоколів проявляється в тому, що в індустрії не існує загальноприйнятого стандарту, який дозволяв би касовій системі, системі управління складом та системі планування ресурсів постачальника безшовно обмінюватися даними. Це змушує інженерів створювати складну та нестійку архітектуру з великої кількості унікальних інтеграцій [22]. Будь-яка зміна в одній із систем вимагає переналаштування всіх залежних ланок, що суттєво ускладнює процес технічного супроводу та модифікації. Криза управління основними даними у мережах пов'язана з відсутністю єдиного джерела істини для ключових бізнес-об'єктів Коли в одному ресторані мережі товар має одну назву, а в іншомувідмінну, жодна централізована система аналітики чи прогнозування не зможе коректно звести ці дані. Неправильне управління основними даними безпосередньо призводить до неточностей у прогнозуванні, некоректного обліку запасів та неможливості консолідації закупівель.

Критичний архітектурний бар'єр в українському контексті В українському контексті глобальні проблеми загострюються унікальною та фундаментальною архітектурною вадою. Величезна кількість мережевих ресторанів досі неофіційно чи напівофіційно використовує програмну платформу «1С» (російського походження) для ведення ключових складських, фінансових та управлінських операцій.

Ця залежність створює критичний набір архітектурних проблем:

1. Ризики безпеки та непередбачуваності, що є прямим наслідком компрометації комерційних даних та відсутності гарантій стабільності в умовах війни;

2. Архітектурні обмеження та складнощі кастомізації, оскільки «1С» є прикладом закритої системи, що утримує бізнес через тотальну модифікацію під власні потреби. Унікальна бізнес-логіка ресторану часто реалізована глибоко в її пропрієтарному коді, що формує критичну залежність від застарілої архітектури значного масштабу та ускладнює перехід на сучасні хмарні рішення;
3. Блокування інтелектуального аналізу даних, оскільки інформація в такій глибоко модифікованій системі є фактично недоступною для зовнішньої аналітики в реальному часі. Її неможливо просто вивантажити в сховище даних для тренування моделей або підключити до сучасних інструментів бізнес-аналітики.

Таким чином, залежність від застарілого, модифікованого та ризикованого спадку російського походження стає головним архітектурним бар'єром для впровадження будь-якої сучасної методики підвищення ефективності, оскільки вона блокує збір чистих, централізованих даних, необхідних для аналізу та оптимізації. Успіх розроблюваної методики безпосередньо залежить від архітектурної здатності системи отримувати стандартизовані дані з первинних джерел.

2.4. Визначення та обґрунтування ключових показників ефективності дослідження

Для кількісної оцінки ефективності розробленої методики та валідації її переваг у порівнянні з існуючими підходами необхідно визначити та обґрунтувати систему ключових показників ефективності. Ці показники повинні комплексно відображати досягнення трьох основних цілей дослідження: підвищення точності прогнозування, оптимізацію рівня запасів (мінімізацію списань та дефіциту) та забезпечення своєчасності логістичних операцій.

Система показників для валідації методики поділяється на чотири взаємопов’язані групи.

Показники точності моделі прогнозування є фундаментальною для інженерної валідації програмного модуля прогнозування. Вона оцінює математичну точність алгоритмів машинного навчання шляхом порівняння прогнозованих значень із фактичними даними про попит за певний період. До неї належать: =

1. Середня абсолютна відсоткова помилка (MAPE) яка є стандартною метрикою для оцінки прогнозів, оскільки не залежить від масштабу даних і легко інтерпретується для бізнесу (2.5):

$$MAPE = \frac{100\%}{n} \sum_{t=1}^{n} \left| \frac{A_t - F_t}{A_t} \right|, \quad (2.5)$$

де $MAPE$ – середня абсолютна відсоткова помилка;

A_t – фактичне значення попиту в період *t*;

F_t – прогнозоване значення попиту в період *t*;

n – кількість періодів спостереження.

Зниження MAPE безпосередньо вказує на покращення якості моделі [23].

2. Середньоквадратична помилка (RMSE) , що використовується для виявлення великих, аномальних помилок, оскільки піднесення до квадрата суттєво впливає на результат призначних відхиленнях, що є критичним для сфери ресторанного бізнесу (2.6):

$$RMSE = \sqrt{\frac{1}{n} \sum_{t=1}^{n} (A_t - F_t)^2}, \quad (2.6)$$

де $RMSE$ – середньоквадратична помилка;

A_t – фактичне значення попиту в період *t*;

F_t – прогнозоване значення попиту в період *t*;

n – кількість періодів спостереження.

3. Прогнозне зміщення $Bias$, яке визначає наявність систематичної тенденції моделі до завищення або заниження прогнозу (2.7):

$$Bias = \frac{1}{n}\sum_{t=1}^{n}(A_t - F_t)\ , \tag{2.7}$$

де $Bias$ – величина зміщення прогнозу;

A_t – фактичне значення попиту в період *t*;

F_t – прогнозоване значення попиту в період *t*;

n – кількість періодів спостереження.

Ця метрика визначає, чи має модель систематичну тенденцію до завищення (Bias > 0) або заниження (Bias < 0) прогнозу. Для управління запасами ідеальна модель повинна мати зміщення, близьке до нуля, оскільки систематичне заниження призводить до дефіциту, а завищення – до надлишкових запасів та списань. Показники ефективності інвентаризації та рівня сервісу.

Ця група оцінює прямий економічний та бізнес-вплив від впровадження точніших прогнозів та автоматизованих замовлень.

Вона включає:

Рівень списання швидкопсувної продукції $К_{сп}$, що є ключовим фінансовим показником, мінімізація якого досягається шляхом уникнення надлишкових замовлень (2.8):

$$К_{сп} = \frac{C_{сп}}{C_{заг}} \times 100\%\ , \tag{2.8}$$

де $C_{сп}$– вартість списаної продукції за звітний період;

$C_{заг}$ – загальна вартість закупленої продукції за той самий період.

Рівень дефіциту $К_{деф}$, який вимірює втрачені продажі та рівень задоволеності клієнтів і безпосередньо пов'язаний із заниженими прогнозами (2.9):

$$К_{деф} = \frac{N_{втрач}}{N_{заг}} \times 100\%\ , \tag{2.9}$$

де $N_{втрач}$– кількість випадків відсутності товару при запиті;

$N_{\text{заг}}$– загальна кількість запитів на товар.

Коефіцієнт оборотності запасів $\text{К}_{\text{об}}$, високе значення якого вказує на ефективне використання капіталу та високу свіжість продукції [20] (2.10):

$$\text{К}_{\text{об}} = \frac{C_{\text{реал}}}{\text{З}_{\text{сер}}} \times 100\% , \tag{2.10}$$

де $C_{\text{реал}}$– собівартість реалізованої продукції;

$\text{З}_{\text{сер}}$– середній товарний запас за період.

Показники ефективності логістики

Ця група оцінює ефективність логістичного модуля, зокрема досягнення мети своєчасності поставок, і базується на галузевій моделі референсних операцій у ланцюгах постачання [24]. До основних показників належать:

Своєчасність доставки OTD що є прямим вимірником досягнення мети та визначається як доставка у заздалегідь узгоджене часове вікно (2.11):

$$OTD = \frac{Z_{\text{вчас}}}{Z_{\text{заг}}} \times 100\% , \tag{2.11}$$

де $Z_{\text{вчас}}$– кількість замовлень, доставлених у визначене часове вікно;

$Z_{\text{заг}}$– загальна кількість замовлень.

Повнота виконання замовлення $\text{З}_{\text{пов}}$, яка вимірює точність комплектації замовлень на центральному складі (2.12):

$$\text{З}_{\text{пов}} = \frac{Z_{\text{пов}}}{Z_{\text{заг}}} \times 100\% , \tag{2.12}$$

де $Z_{\text{вчас}}$– кількість замовлень, виконаних у повному обсязі;

$Z_{\text{заг}}$– загальна кількість замовлень.

Вартість логістики на замовлення, зниження якої досягається через оптимізацію маршрутів, мінімізацію загального пробігу та часу в дорозі.

Інженерно-технічні показники якості програмного рішення Ця група спрямована на оцінку нефункціональних характеристик розробленого програмного прототипу та базується на атрибутах якості, визначених у стандарті ISO/IEC 25010 [25]: – продуктивність, що включає час відгуку (середній та 95-й перцентиль часу, необхідного системі для виконання ключових операцій, наприклад, генерації прогнозу) та пропускну здатність (кількість транзакцій, які система здатна обробити за одиницю часу без деградації продуктивності); – надійність, яка визначається коефіцієнтом готовності, що показує відсоток часу, протягом якого система доступна для виконання своїх функцій (2.13):

$$К_г = \frac{Т_{м.в}}{Т_{м.в}+Т_{відн}} \times 100\%, \quad (2.13)$$

де $К_г$ –коефіцієнт готовності системи;

$Т_{м.в}$ – середній час між відмовами;

$Т_{відн}$– середній час відновлення .

Точність та цілісність даних, що оцінюється через рівень помилок синхронізації (відсоток записів, що не були коректно синхронізовані між первинними системами та централізованою базою даних) та коефіцієнт валідації основних даних (відсоток товарних позицій, які успішно пройшли валідацію та нормалізацію при завантаженні в систему).

Ці чотири групи показників є взаємозалежними та формують комплексну систему метрик, яка буде використана в четвертому розділі для експериментальної перевірки та валідації розробленої методики.

3 РОЗРОБКА ПРОГРАМНО-МЕТОДИЧНИХ ПІДХОДІВ ПІДВИЩЕННЯ СВОЄЧАСНОСТІ ПОСТАВОК ТА ТОЧНОСТІ ІНВЕНТАРИЗАЦІЇ

3.1. Обґрунтування концептуальної архітектури програмного прототипу із зазначенням загальної схеми, компонентів та взаємодії

Розробка програмного прототипу базується на вирішенні ключових архітектурних та інженерних викликів, ідентифікованих у підрозділі 2.3. Враховуючи гетерогенність (різнорідність) IT-ландшафту мережевих ресторанів, наявність успадкованих систем (зокрема модифікованих платформ «1С») та відсутність стандартів сумісності, пропонується гібридна сервіс-орієнтована архітектура.

Мета цієї архітектури – не замінити існуючі облікові системи (касові термінали, системи планування ресурсів), а доповнити їх відсутніми інтелектуальними модулями прогнозування та оптимізацією, виступаючи як зовнішній інтегрований аналітичний шар. Формалізація задачі управління каналами постачання Для побудови ефективної програмної архітектури необхідно формалізувати задачу управління каналами постачання як задачу оптимізації. Розглянемо систему мережевого ресторанного бізнесу як множину об'єктів та процесів.

Вхідні параметри та змінні

Нехай система описується такими множинами:

- $R = \{r_1, r_2 \dots . r_n\}$ – множина ресторанів мережі;
- $S = \{s_1, s_2 \dots . s_m\}$ – множина інгредієнтів (товарних позицій);
- $P = \{p_1, p_2 \dots . p_k\}$ – множина постачальників;
- $T = \{t_1, t_2 \dots . t_h\}$ – часові періоди планування (дні).

Стан системи в момент часу t для кожного інгредієнта s у ресторані r описується вектором вхідних даних: $I_{\text{вх}}(t) = \langle \mathrm{D_{srt}}, \mathrm{Z_{srt}}, \mathrm{L_{sp,}}, \mathrm{C_s} \rangle$, де $\mathrm{D_{srt}}$ – прогнозований попит на інгредієнт s у ресторані r на період t; $\mathrm{Z_{srt}}$ – поточний залишок інгредієнта s на складі ресторану r; $\mathrm{L_{sp,}}$ – час виконання замовлення постачальником p для інгредієнта s; $\mathrm{C_s}$ – вартість одиниці інгредієнта s включаючи закупівельну ціну та логістичні витрати.

Вихідні параметри

Результатом роботи системи є матриця замовлень Q, де елемент q_{srpt} визначає кількість інгредієнта s, яку необхідно замовити у постачальника p для ресторану r на період t: $Q_{\text{вих}}(t) = \{q_{\text{srpt}} | q_{\text{srpt}} \geq 0\}$.

Цільова функція

Метою методики є мінімізація сукупних витрат TC при забезпеченні заданого рівня сервісу. Формально цільова функція виглядає так (3.1)

$$TC = \sum_{t \in T} \sum_{r \in R} \sum_{s \in S} (C_{\text{зам}} + C_{\text{збер}} + C_{\text{деф}}) \rightarrow min, \quad (3.1)$$

де $C_{\text{зам}}$ – витрати на оформлення та доставку замовлення;

$C_{\text{збер}}$ – витрати на зберігання запасів;

$C_{\text{деф}}$– штрафні витрати від дефіциту.

Обмеження системи З

адача вирішується при виконанні системи обмежень:

1. Балансове обмеження запасів. Залишок на наступний період дорівнює поточному залишку плюс надходження мінус фактичне споживання.
2. Обмеження ємності складу. Сумарний об'єм запасів не повинен перевищувати максимальну місткість складу ресторану.
3. Обмеження терміну придатності. Час реалізації товару не повинен перевищувати його термін придатності, що накладає обмеження на максимальний обсяг замовлення;

4. Обмеження рівня сервісу. Ймовірність наявності товару на складі повинна бути не меншою за заданий рівень надійності (наприклад, 95%);
5. Логістичні обмеження. Замовлення не може бути меншим за мінімальний поріг, встановлений постачальником.

Візуально формальна модель задачі управління каналами постачання представлена на рисунку 3.1.

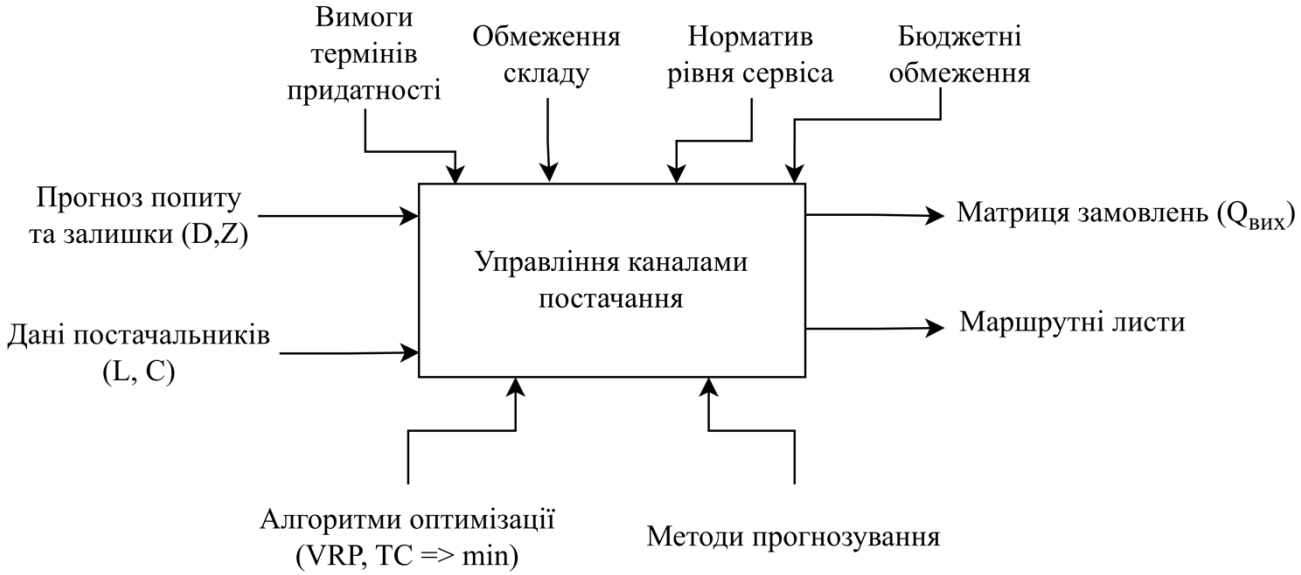


Рис. 3.1. Структурна модель задачі управління каналами постачання

Саме для реалізації цієї математичної моделі та дотримання вказаних обмежень пропонується наступний архітектурний підхід.

Замість монолітної архітектури, яка є типовою для традиційних систем управління ланцюгами постачання [21] і схильна до формування технологічної залежності від постачальника, пропонується модульна архітектура на базі мікросервісів, що взаємодіють через асинхронні події (подієво-орієнтована архітектура). Цей вибір обґрунтований такими перевагами для інженерії програмного забезпечення [26]:

– відмовостійкість, коли збій одного сервісу (наприклад, модуля прогнозування) не зупиняє роботу інших (наприклад, збору даних або логістики);

– гнучкість інтеграції, що дозволяє легко підключати нові джерела даних (нові касові системи, філії) без необхідності перебудови всієї системи, що вирішує проблему складної та заплутаної архітектури інтеграцій [22];

– технологічна незалежність, оскільки кожен сервіс (наприклад, прогнозування на Python, логістика на JavaScript) може бути розроблений з використанням найбільш придатного стека технологій; – масштабованість, що дозволяє вибірково масштабувати лише ті компоненти, які зазнають високого навантаження (наприклад, сервіс приймання даних із касових терміналів).

Концептуальна схема архітектури Запропонована архітектура програмного прототипу складається з чотирьох логічних шарів, взаємодія яких зображена на рисунку 3.2.

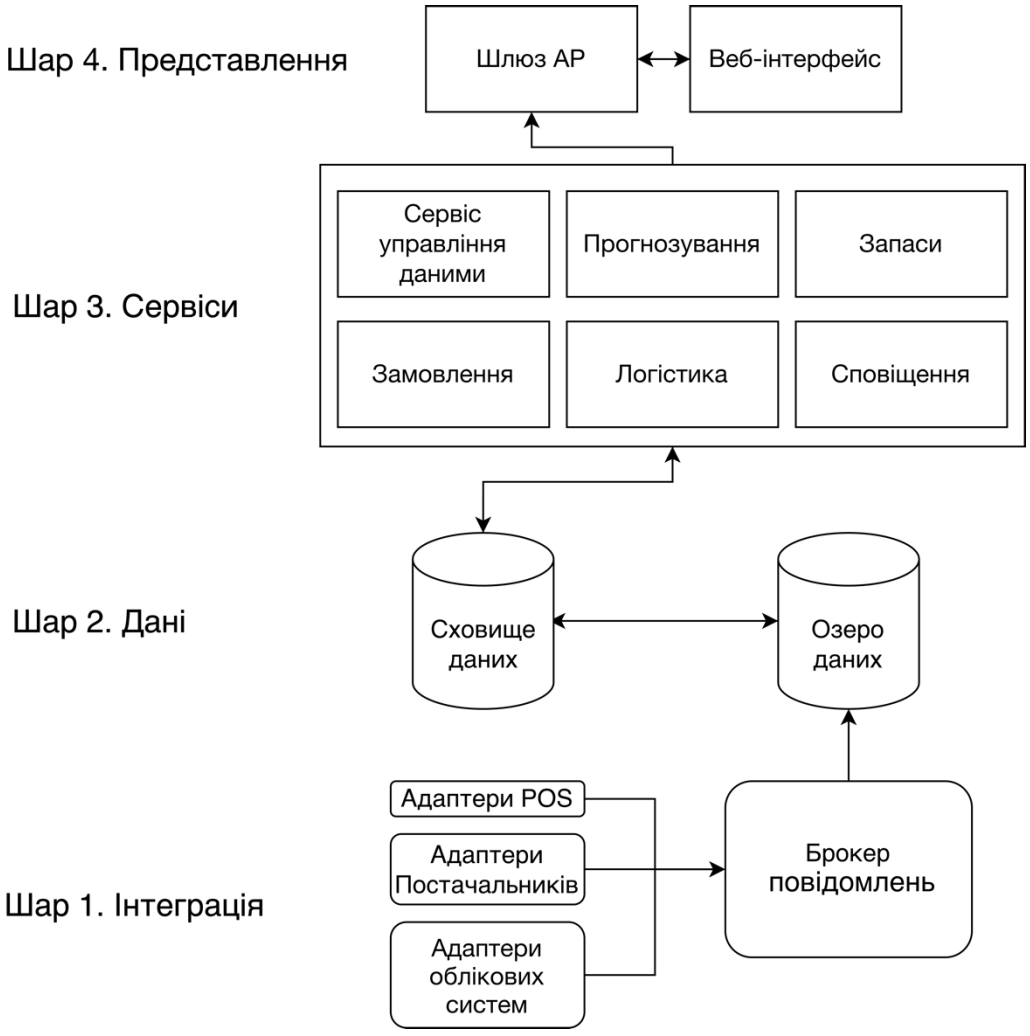


Рис. 3.2. Концептуальна архітектура програмного прототипу

Детальний опис шарів:

Шар 1. Інтеграція та приймання даних Цей шар є критично важливим для подолання бар'єру закритих систем. Він складається з набору адаптерів (конекторів) та конвеєра ETL/ELT (вилучення, перетворення, завантаження). Компоненти:

– адаптери успадкованих систем спеціалізовані мікросервіси, що підключаються до баз даних касових систем (pos), та інших облікових платформ (навіть через вивантаження файлів .csv, .xml) для експорту «сирих» даних про продажі, залишки та списання;
– адаптери постачальників – парсери для прайс-листів та інтерфейси для електронного обміну даними;
– брокер повідомлень (наприклад, RabbitMQ або Kafka) забезпечує асинхронну та гарантовану доставку даних від адаптерів до центрального сховища.

Шар 2. Централізована платформа даних цей шар слугує «єдиним джерелом істини» для всієї мережі, вирішуючи проблему розрізнених та неузгоджених даних. Компоненти:

– озеро даних, зберігає всі «сирі», неструктуровані дані (чеки, логи транзакцій, історія списань), необхідні для майбутнього тренування моделей машинного навчання;
– сховище даних, зберігає очищені, структуровані та агреговані дані, готові для аналітики та роботи програмних модулів.

Шар 3. Інтелектуальні сервіси Це ядро запропонованої методики. Кожен сервіс є незалежним програмним компонентом, що реалізує частину бізнес-логіки. Компоненти:

– сервіс управління основними даними (mdm) вирішує проблему неузгоджених назв, нормалізує довідники (наприклад, приведення різнописання товарних позицій до єдиного ідентифікатора);

– сервіс прогнозування, реалізує моделі прогнозування (на базі часових рядів, наприклад prophet або arima [13]), використовуючи очищені дані зі сховища, та зберігає прогнози попиту назад у сховище даних;

– сервіс управління запасами, реалізує математичні моделі які ідентифіковані в підрозділі 2.2 , такі як розрахунок динамічного страхового запасу SS та рівня поповнення R на основі прогнозів;

– сервіс автоматизації замовлень на основі даних сервісу управління запасами та поточних залишків генерує проєкти замовлень для філій;

– сервіс оптимізації логістики агрегує проєкти замовлень і вирішує задачу маршрутизації транспортних засобів [18], будуючи оптимальні маршрути для центрального складу;

– сервіс сповіщень, моніторить відхилення та сповіщає менеджмент.

Шар 4. Шар представлення та інтерфейсів Надає інтерфейс для взаємодії з користувачами та зовнішніми системами. Компоненти:

– веб-інтерфейс (інформаційна панель);

– візуалізує ключові показники ефективності (визначені в підрозділі 2.4), показує результати прогнозів, згенеровані замовлення та маршрути;

– шлюз API надає захищений доступ до сервісів та повертає згенеровані замовлення назад у спадкові системи або надсилає їх постачальникам, замикаючи цикл автоматизації.

Така архітектура є гнучкою, вирішує інженерні проблеми інтеграції, описані в другому розділі, та створює надійний програмний фундамент для реалізації запропонованої методики.

3.2. Інструментальні засоби для моделювання і створення програмного прототипу

Вибір інструментальних засобів та технологічного стека безпосередньо ґрунтується на концептуальній архітектурі (підрозділ 3.1), що базується на мікросервісах та подієво-орієнтованому підході, а також на інженерних вимогах (підрозділ 2.4) та специфіці завдань.

Перш ніж створювати програмний прототип, необхідно провести детальне проєктування системи. Для цього будуть використані такі нотації та стандарти інженерії програмного забезпечення:

– уніфікована мова моделювання (UML), що буде використана для деталізації взаємодії компонентів, зокрема діаграми компонентів для візуалізації структури сервісів та діаграми послідовності для моделювання потоків даних та викликів API між сервісами (наприклад, процес від «прогнозу» до «генерації замовлення»);

– модель C4 для візуалізації архітектури програмного забезпечення на різних рівнях абстракції (контекст, контейнери, компоненти), що є надзвичайно ефективним для проєктування мікросервісних систем, оскільки чітко показує межі та відповідальність кожного сервісу [29];

– нотація моделювання бізнес-процесів (BPMN) для моделювання та формалізації наскрізних бізнес-процесів («як є» та «як має бути»), таких як повний цикл від збору даних про продажі до підтвердження доставки, що забезпечує узгодженість програмної логіки з операційними потребами.

Технологічний стек програмного прототипу. Вибір технологій здійснено з огляду на принципи гнучкості, технологічної незалежності та наявності потужних бібліотек для вирішення специфічних наукоємних завдань.

Шар 1. Інтеграція та приймання даних. Як основний засіб реалізації алгоритмів обробки даних та інтеграції обрано високорівневу мову програмування

з розвиненою екосистемою бібліотек для наукових обчислень Python. Вибір обґрунтований наявністю спеціалізованих інструментів для роботи з різними форматами даних та протоколами баз даних. Приклад реалізації базового адаптера для завантаження даних наведено на рисунку 3.3.

```
def load_sales_data(file_path, db_connection):
    raw_data = pd.read_csv(file_path)

    clean_data = raw_data.dropna(subset=['product_id', 'qty'])
    clean_data['date'] = pd.to_datetime(clean_data['date'])

    engine = create_engine(db_connection)
    clean_data.to_sql( name: 'sales_staging', engine, if_exists='append', index=False)
```

Рис. 3.3. Фрагмент програмного коду адаптера для імпорту даних

Інструменти:

– Pandas, для парсингу файлів (.csv, .xml)

– pyodbc, psycopg2 для підключення до успадкованих баз даних (MS SQL, MySQL);

– RabbitMQ як брокер повідомлень, обраний замість Kafka через меншу складність розгортання для прототипу, при цьому повністю забезпечуючи надійну асинхронну доставку повідомлень між сервісами, що є ключовим для мікросервісної архітектури [27].

Шар 2. Централізована платформа даних озеро даних: MinIO. Це S3-сумісне об'єктне сховище, яке легко розгорнути локально для прототипу. Воно буде використовуватись для зберігання «сирих» неструктурованих. Сховище даних: PostgreSQL. Реляційна система управління базами даних із відкритим вихідним кодом. Вона буде використовуватись як структуроване сховище для очищених даних, довідників основних даних та результатів прогнозів, готових для аналітики.

Шар 3. Інтелектуальні сервіси цей шар є ядром системи, тому вибір інструментів є критичним. Мова програмування: високорівнева мова для наукових обчислень Python. Фреймворк для мікросервісів: FastAPI. Обраний завдяки високій

продуктивності, автоматичній генерації документації API (інтерфейс Swagger) та простоті розробки.

Спеціалізовані бібліотеки:

1. сервіс прогнозування: Prophet від Meta для швидкого та якісного прогнозування часових рядів з урахуванням сезонності як зазначено в [13]);
2. Statsmodels – для порівняння та валідації моделей;
3. Сервіс оптимізації логістики: Google OR-Tools – потужна бібліотека з відкритим вихідним кодом для вирішення задач комбінаторної оптимізації, включаючи задачу маршрутизації транспортних засобів, що дозволяє реалізувати складні алгоритмічні задачі без необхідності розробки їх з нуля [28].

Контейнеризація та оркестрація:

Docker: кожен мікросервіс буде запакований у Docker-контейнер для забезпечення ізоляції та портативності;

Docker Compose: буде використано для запуску та управління сервісами FastAPI, RabbitMQ та PostgreSQL в єдиному середовищі під час розробки та тестування прототипу.

Шар 4. Шар представлення та інтерфейсів Шлюз API: функції шлюзу виконуватиме FastAPI або спеціалізований інструмент Kong для більш складного управління трафіком та безпекою.

Вебінтерфейс інформаційна панель: React.js – провідна JavaScript-бібліотека для створення динамічних та відмовостійких інтерфейсів користувача.

Візуалізація даних: Chart.js або Recharts – бібліотеки JavaScript, що легко інтегруються з React для побудови інтерактивних графіків (динаміка $MAPE$, показники ефективності логістики, рівень списань).

Таблиця 3.1.

Інструментальні засоби програмного прототипу

Архітектурний шар / завдання	Обраний інструмент / технологія	Обґрунтування (призначення)
Моделювання архітектури	UML, C4 Model, BPMN	Візуальне проєктування, документування архітектури та бізнес-логіки.
Шар 1. Інтеграція	Python (Pandas, pyodbc)	Гнучкість для підключення до успадкованих систем та парсингу файлів.
Шар 1. Асинхронізація	RabbitMQ	Надійна черга повідомлень для зв'язку мікросервісів [27].
Шар 2. Озеро даних (сирі дані)	MinIO (S3-сумісне)	Масштабоване сховище для неструктурованих даних.
Шар 2. Сховище даних (очищені дані)	PostgreSQL	Надійна реляційна СУБД для структурованих даних та аналітики.
Шар 3. Розробка сервісів	FastAPI (Python)	Високопродуктивний асинхронний фреймворк для API.
Шар 3. Прогнозування (ML)	Prophet, Statsmodels	Спеціалізовані бібліотеки для аналізу часових рядів [23].
Шар 3. Оптимізація (VRP)	Google OR-Tools	Потужний розв'язувач для задач маршрутизації [28].
Шар 3. Розгортання	Docker, Docker Compose	Контейнеризація та управління життєвим циклом сервісів.
Шар 4. Front-end	React.js, Chart.js	Сучасна інформаційна панель для візуалізації показників ефективності.

Обраний стек технологій повністю базується на відкритому програмному забезпеченні, що виключає проблему технологічної залежності від постачальника ідентифіковану в підрозділі 2.3 та забезпечує необхідну гнучкість для реалізації всієї методики.

3.3. Задачі прогнозування потреби продуктів

Прогнозування попиту є інтелектуальним ядром запропонованої методики та єдиним способом розірвати цикл реактивного управління запасами (дефіцит → надлишок → списання). З погляду інженерії програмного забезпечення, задача прогнозування в ресторанному бізнесі є нетривіальною і вимагає вирішення трьох специфічних програмних підзадач, перш ніж будь-яка математична модель (ARIMA, Prophet тощо) може бути застосована: задача декомпозиції попиту, задача очищення та агрегації даних або задача диференційованого моделювання.
Інженерна задача: декомпозиція попиту ключова проблема полягає в тому, що ресторанний бізнес має справу з двома типами попиту:

– Незалежний попит – попит на готові страви, який формується ринком. Це єдине, що ми можемо прогнозувати, оскільки саме ці дані фіксуються касовими системами; Наприклад: «салат Цезар», «бургер».
– Залежний попит – попит на інгредієнти, який повністю залежить від прогнозу незалежного попиту. Наприклад: «куряче філе», «листя Ромен».

Існуючі спеціалізовані системи часто намагаються прогнозувати потребу в інгредієнтах напряму, на основі історичного споживання цих інгредієнтів. Цей підхід є хибним, оскільки він ігнорує реальні ринкові драйвери (попит на страви) і призводить до низької точності $MAPE$. Запропонована методика базується на класичному інженерному підході планування матеріальних потреб [30], який вимагає програмної реалізації декомпозиції за специфікацією.

Програмний прототип (зокрема сервіс управління запасами та сервіс управління основними даними) повинен оперувати такими сутностями: – довідник товарних позицій – усі товари, як готові страви, так і інгредієнти; – специфікація матеріалів – зв’язок «батько-нащадок», що описує, з яких інгредієнтів та в якій кількості складається готова страва.

Алгоритм, який має реалізувати сервіс управління запасами, виглядає таким чином:

1. Отримати прогноз попиту на готові страви F_j(де j– готова страва) від сервісу прогнозування.
2. Для кожного інгредієнта i розрахувати загальну залежну потребу D_i шляхом «розгортання» специфікації за формулою (3.2):

$$D_i = \sum_i \left(F_j \times Q_{ij}\right), \tag{3.2}$$

деD_i – розрахована потреба в інгредієнті i;

F_j – прогноз продажу j-ї готової страви;

Q_{ij} – кількість інгредієнта i, необхідна для приготування однієї одиниці страви j.

Тільки після виконання цієї програмної декомпозиції ми отримуємо достовірний прогнозний вхідний сигнал D_i для розрахунку параметрів замовлення (страховий запас SS та рівень поповнення R) для кожного інгредієнта.

Інженерна задача: очищення та агрегація даних якість прогнозу F_j прямо залежить від якості вхідних часових рядів. Дані з касових систем та «1С», ідентифіковані в підрозділі 2.3, є «забрудненими» і вимагають програмної обробки у конвеєрі вилучення та завантаження даних (ETL/ELT).

Нормалізація як перший крок – застосування сервісу управління основними даними для агрегації продажів. Позиції «Бургер» та «burger_pos2» з різних філій

мають бути програмно зведені до єдиної товарної позиції «Бургер Класичний» перед побудовою часового ряду;

Агрегація за часом. Дані з касових систем (чеки) надходять як потік транзакцій. Процес обробки даних “Шар 1” повинен агрегувати їх у дискретні часові проміжки, щоб сформувати часовий ряд, придатний для сервісу прогнозування;

Обробка нульових продажів: це критична інженерна задача. Нуль у даних про продажі може означати або «нульовий попит», або «дефіцит». Модель прогнозування, навчена на даних з дефіцитом, буде систематично занижувати показник прогнозного зміщення з підрозділу 2.4.

Рішення: програмний модуль повинен отримувати дані про точність інвентаризації (підрозділ 2.2) та історію залишків. Якщо продажі = 0, але залишок = 0 у той самий період, цей нуль має бути програмно видалений з вибірки для навчання моделі або замінений (відновлений) на середнє значення.

Задача диференційованого моделювання номенклатура ресторану є вкрай нерівномірною. Приблизно 20% позицій (група А) генерують 80% продажів (наприклад, «Кола», «Бургер»), тоді як 80% позицій (група С) мають низький, спорадичний попит (наприклад, соус «Табаско», специфічні спеції). Застосування однієї складної моделі до всіх товарних позицій є інженерно невиправданим:
для групи А – це необхідно,
для групи С – це призведе до перенавчання на рідкісних даних або до помилок обчислення.

Тому сервіс прогнозування повинен реалізувати гібридну стратегію (програмну логіку, що обирає модель):

– Для високочастотного попиту (група А/В): використовувати складні моделі часових рядів ARIMA, Prophet, які враховують сезонність та зовнішні регресори;

– Для переривчастого попиту (група C): використовувати спеціалізовані, простіші алгоритми, такі як метод Кростона. Цей метод спеціально розроблений для прогнозування попиту, що складається з багатьох нулів, оскільки він моделює дві окремі величини: інтервал між замовленнями та розмір замовлення [31]. Реалізація логіки вибору моделі наведена на рисунку 3.4.

```
def forecast_demand_hybrid(sku_data):
    demand_density = (sku_data['sales'] > 0).mean()
    SPORADIC_THRESHOLD = 0.2

    if demand_density < SPORADIC_THRESHOLD:
        return croston_method(sku_data)
    else:
        return prophet_forecast(sku_data)
```

Рис. 3.4. Фрагмент коду реалізації гібридної стратегії прогнозування

Таким чином, задачею прогнозування в контексті цієї кваліфікаційної роботи є не просто вибір математичної моделі, а розробка програмного конвеєра, який гарантує чистоту та узгодженість вхідних даних таких як обробка дефіциту, реалізує алгоритм декомпозиції для переходу від прогнозу страв до потреби в інгредієнтах та застосовує гібридний підхід до моделювання, обираючи оптимальний алгоритм (Prophet або метод Кростона) залежно від типу попиту на товарну позицію.

3.4. Опис алгоритму прогнозування з урахуванням сезонності та специфіки HoReCa

На основі інженерних задач, визначених у підрозділі 3.3, розроблено програмний алгоритм, який є ядром сервісу прогнозування та реалізує запропоновану методику. Алгоритм являє собою автоматизований конвеєр, що

запускається за розкладом і складається з трьох фаз: підготовка даних, гібридне моделювання та декомпозиція потреби.

Фаза І. Підготовка та очищення даних

Ця фаза є критичною для забезпечення якості вхідних даних за принципом «якість вхідних даних визначає якість результату» та вирішує інженерні проблеми, описані в підрозділах 2.3 та 3.3.

1. Вибірка даних: алгоритм робить запит до сховища даних (PostgreSQL) для отримання «сирих» даних за N останніх періодів (наприклад, 365 днів) з транзакційних таблиць. Вхідні дані: часова мітка, ідентифікатор термінала, назва товару, кількість продажів, залишок.
2. Нормалізація: виконується об'єднання (JOIN) з довідником сервісу управління основними даними. Процес: «raw_sku_name» (наприклад, «burger_pos2») → «master_sku_id» (наприклад, «Бургер Класичний»). Як результат усі дані зведені до єдиної номенклатури.
3. Агрегація: дані про транзакції агрегуються у часовий ряд із фіксованим кроком. Процес: сумування кількості продажів за ідентифікатором товару та датою.
4. Обробка дефіциту: алгоритм ітерує по часовому ряду для очищення «хибних нулів», які спотворюють математичну модель. Логіка: якщо продажі = 0 та залишок = 0, то значення продажів вважається невизначеним (NULL) або відновлюється середнім значенням. Обґрунтування: програмно виключаються дні, коли продажі були нульовими через відсутність товару, а не через відсутність попиту, що є критичним для точності прогнозу.
5. Визначення типу попиту: алгоритм аналізує очищений часовий ряд кожної товарної позиції для визначення його характеру. Логіка: якщо відсоток днів з продажами менший за 30% (поріг 0.3), тип попиту визначається як «спорадичний», інакше — як «регулярний». Це програмний перемикач для гібридної стратегії.

Фаза II. На цій фазі алгоритм застосовує диференційований підхід, обираючи оптимальну модель на основі типу попиту. Для регулярного попиту: використовується модель Prophet [32], обрана в підрозділі 3.2 завдяки її здатності нативно обробляти специфіку ресторанного бізнесу.

Ініціалізація моделі.

1. Інженерія сезонності:
    - тижнева сезонність (обробляє різницю між буднями та вихідними),
    - річна сезонність (обробляє сезонні зміни, наприклад «сезон терас»).
2. Додавання зовнішніх регресорів: алгоритм збагачує модель даними, які є драйверами попиту, але не є частиною самого часового ряду:
    - погодні дані як приклад , висока температура корелює з попитом на напої),
    - календар свят та локальні події дані з маркетингового календаря.
3. Навчання та прогноз: модель навчається на історичних даних і генерує прогноз на майбутній період.
4. Результат: отримано прогноз F_j для готової страви j .

Для спорадичного попиту: Використовується метод Кростона [31].

1. Обробка: дані подаються у функцію, що реалізує алгоритм Кростона.
2. Результат: модель прогнозує очікуваний інтервал між продажами та очікуваний обсяг наступного продажу, відношення яких дає прогноз F_j.

Фаза III. Декомпозиція потреби на цій фазі алгоритм переходить від прогнозу продажів до прогнозу потреби в інгредієнтах, реалізуючи задачу декомпозицію специфікацій.

1. Отримання прогнозу страв: алгоритм має масив прогнозів F_j для всіх готових страв.
2. Отримання специфікацій: алгоритм робить запит до сервіс управління основними даними для отримання таблиці коефіцієнтів Q_{ij} (кількість інгредієнта i у страві j).

3. Розрахунок залежного попиту: алгоритм виконує розрахунок за формулою (3.1): для кожного інгредієнта сумується добуток прогнозу страви на кількість інгредієнта.
4. Запис результатів: алгоритм записує результати D_i прогнозована потреба в кожному інгредієнті у Сховище даних (PostgreSQL).

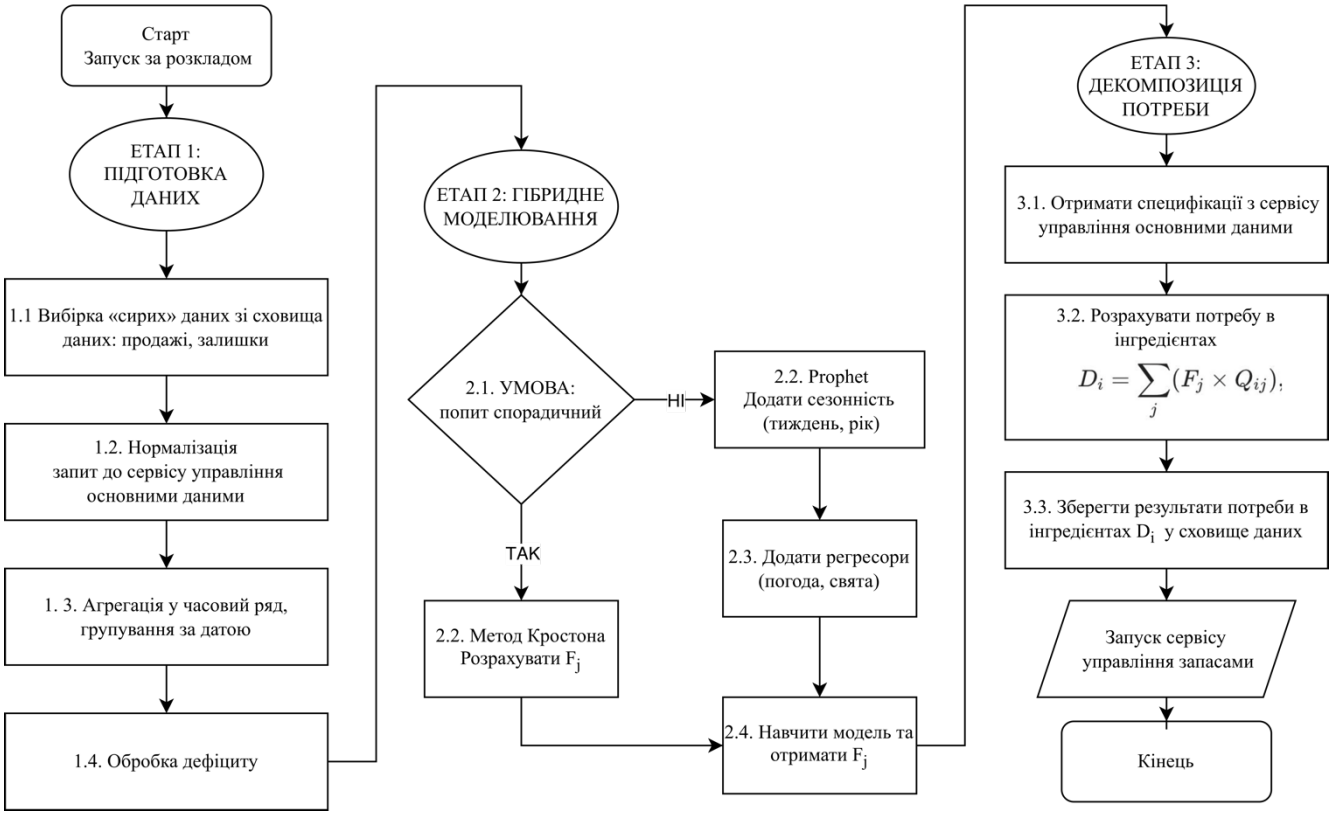


Рис. 3.5. Блок-схема алгоритму прогнозування

Описаний алгоритм є інженерною реалізацією методики прогнозування. Він автоматизує повний цикл від «сирих» даних до готового прогнозу потреби в інгредієнтах, враховуючи специфіку ресторанного бізнесу (сезонність, дефіцит) та застосовуючи гібридний підхід до моделювання, що є необхідним для досягнення високих показників точності, визначених у попередніх розділах.

3.5. Детальне проектування архітектурної реалізації ключових програмних компонентів, включаючи UML-діаграми та діаграми послідовності

На основі архітектурного стилю, обраного в підрозділі 3.1, який доцільно визначити як мікросервісну архітектуру [27], цей підрозділ деталізує проєктування ключових компонентів системи. Мікросервісний підхід є оптимальним для вирішення поставленої задачі, оскільки він забезпечує:

1. Масштабованість, коли кожен сервіс може масштабуватися незалежно.
2. Гнучкість технологій, що дає можливість обирати найкращий інструмент для конкретної задачі .
3. Надійність, оскільки збій в одному сервісі (не призведе до зупинки всієї системи управління постачанням.
4. Незалежне розгортання, коли оновлення логіки прогнозування не вимагатиме зупинки сервісу управління запасами.

Ця гнучкість та відмовостійкість є кардинальною перевагою над монолітними, жорстко інтегрованими системами, які не здатні адаптуватися до динамічних умов ресторанного бізнесу. Для візуалізації архітектури використовується нотація C4 [29], яка дозволяє декомпозувати систему на різних рівнях.

На рисунку 3.6 представлена діаграма компонентів, що ілюструє ключові мікросервіси системи та їхні основні взаємозв’язки.

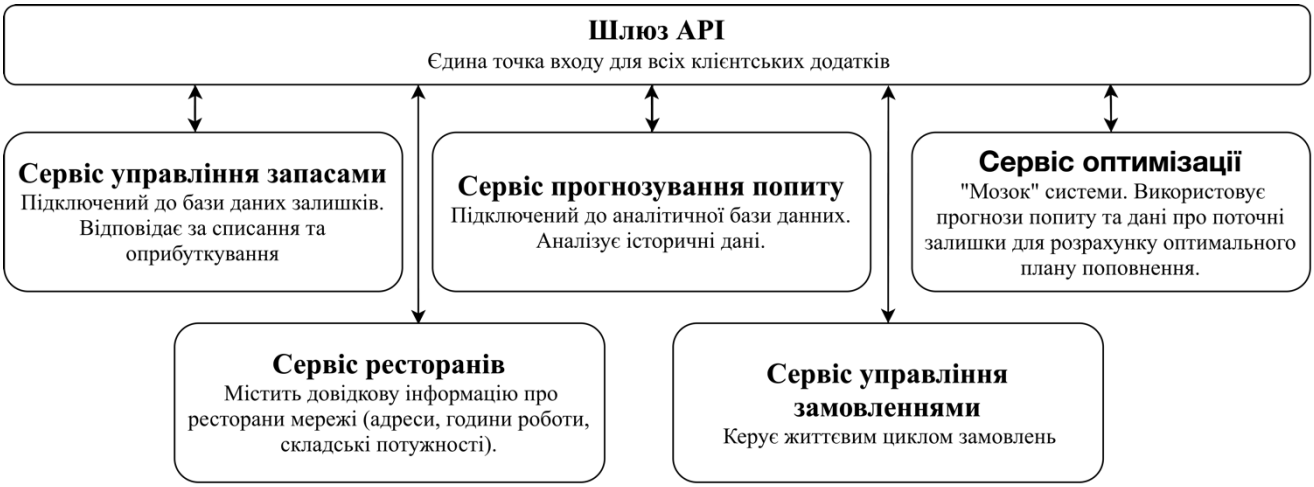


Рис. 3.6. Діаграма компонентів системи управління постачанням

Опис ключових компонентів:

1. **Шлюз API**: відповідає за маршрутизацію запитів, автентифікацію та балансування навантаження.
2. **Сервіс управління запасами**: відповідає за списання продуктів на основі даних про продажі та оприбуткування при отриманні поставок
3. **Сервіс прогнозування попиту**: аналізує історичні дані про продажі, сезонність та інші фактори (наприклад, маркетингові акції) для побудови прогнозів попиту. Використовує сучасні методики інтелектуального аналізу даних.
4. **Сервіс оптимізації**: використовує прогнози попиту та дані про поточні залишки для розрахунку оптимального плану поповнення. Ключове завдання: вирішення задачі маршрутизації транспортних засобів та оптимізації запасів. Використовує бібліотеку Google OR-Tools для вирішення оптимізаційних задач.

5. **Сервіс управління замовленнями**: створює та відстежує життєвий цикл замовлень на поповнення як внутрішніх, з центрального складу, так і зовнішніх, у постачальників.
6. **Сервіс ресторанів**: містить довідкову інформацію про ресторани мережі такі як адреси, години роботи, складські потужності.

Деталізація взаємодії за допомогою діаграм послідовності діаграми послідовності ілюструють взаємодію компонентів у часі для виконання ключових бізнес-сценаріїв. Сценарій 1: автоматичне формування замовлення на поповнення. Цей сценарій є критично важливим для підвищення ефективності. Він запускається за розкладом для кожного ресторану мережі.

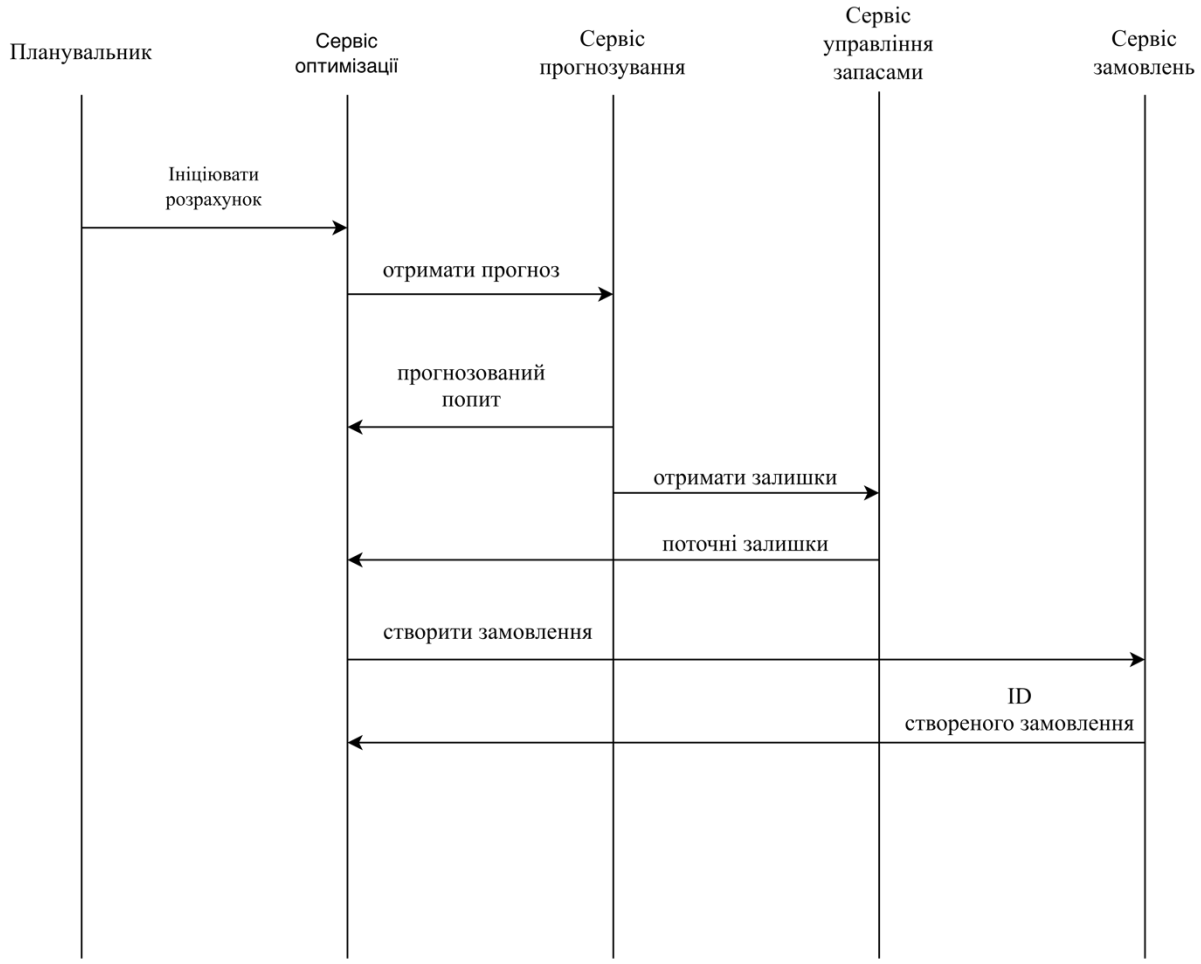


Рис. 3.7. Діаграма послідовності для сценарію «Автоматичне формування замовлення»

Проєктування структури ключових компонентів (UML Class Diagram) Для деталізації внутрішньої структури сервісів розглянемо спрощену діаграму класів для сервісу управління запасами та сервісу управління замовленнями (рис. 3.8).

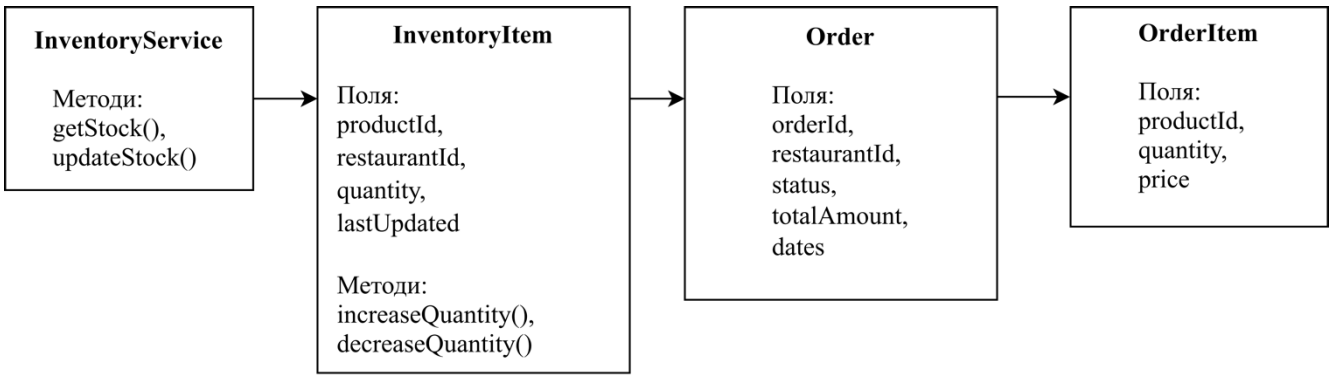


Рис. 3.8. Спрощена діаграма класів для сервісів управління запасами та замовленнями

Обґрунтування технологічного стека для реалізації описаної архітектури пропонується технологічний стек, що відповідає сучасним практикам інженерії програмного забезпечення .

Таблиця 3.2.

Технологічний стек проекту

Категорія	Компонент	Технологія	Обґрунтування
Бекенд (Data Science)	Сервіс оптимізації, Сервіс прогнозування	FastAPI	Високорівнева мова програмування, що є стандартом для аналізу даних та машинного навчання, забезпечує легку інтеграцію з бібліотеками Prophet та OR-Tools.
Бекенд (Core)	Сервіс управління запасами, Сервіс замовлень	Node.js	Висока швидкість обробки I/O запитів, єдина мова з фронтендом, нативна підтримка JSON.

Продовження таблиці 3.2.

Технологічний стек проекту

Категорія	Компонент	Технологія	Обґрунтування
Бази даних	Сховище сервісів	PostgreSQL	Надійна реляційна система управління базами даних з підтримкою JSONB, що забезпечує гнучкість.
Кешування	Кеш залишків, прогнозів	Redis	Використовується для кешування «гарячих» даних, щоб зменшити навантаження на сервіс управління запасами.
Комунікація	Обмін подіями	RabbitMQ	Для асинхронної взаємодії. Наприклад, сервіс управління запасами публікує подію зміни рівня залишків, на яку підписаний сервіс оптимізації.
Інструмент оптимізації	Сервіс оптимізації	Google OR-Tools	Потужна бібліотека з відкритим вихідним кодом для вирішення комбінаторних задач, включаючи задачу маршрутизації та лінійне програмування.
Фронтенд	Веб-додаток	React	Сучасний фреймворк для побудови динамічних інтерфейсів користувача.

3.6. Розробка алгоритму контролю та оповіщення про відхилення в логістичних операціях

Для реального підвищення ефективності система повинна не лише створювати оптимальні плани, але й контролювати їхнє виконання. Логістичні операції в ресторанному бізнесі характеризуються високою динамікою та ризиками: затримки поставок, невідповідність якості, помилки при прийманні

товару тощо. Метою розробки цього алгоритму є проактивне виявлення та сповіщення про суттєві відхилення (аномалії) між запланованими логістичними показниками та фактичними даними, що надходять із системи. Завдання алгоритму:

1. ідентифікувати ключові точки контролю в логістичному ланцюгу;
2. порівнювати планові та фактичні показники в режимі реального часу;
3. класифікувати відхилення за ступенем критичності;
4. генерувати цільові сповіщення для відповідальних менеджерів. Такий підхід відповідає принципам управління операціями, де контроль відхилень є ключовим для підтримки стабільності процесу [30].

Архітектурна реалізація: сервіс моніторингу та сповіщень. Для реалізації цього функціоналу в рамках мікросервісної архітектури пропонується введення нового компонента — сервіс моніторингу та сповіщень (Monitoring & Alerting Service). Цей сервіс діє за подієво-орієнтованим принципом. Він не втручається в основні бізнес-процеси (як-от створення замовлення), а підписується на події, що генеруються іншими сервісами (наприклад, сервіс замовлень, сервіс управління запасами) через асинхронну шину повідомлень (RabbitMQ).

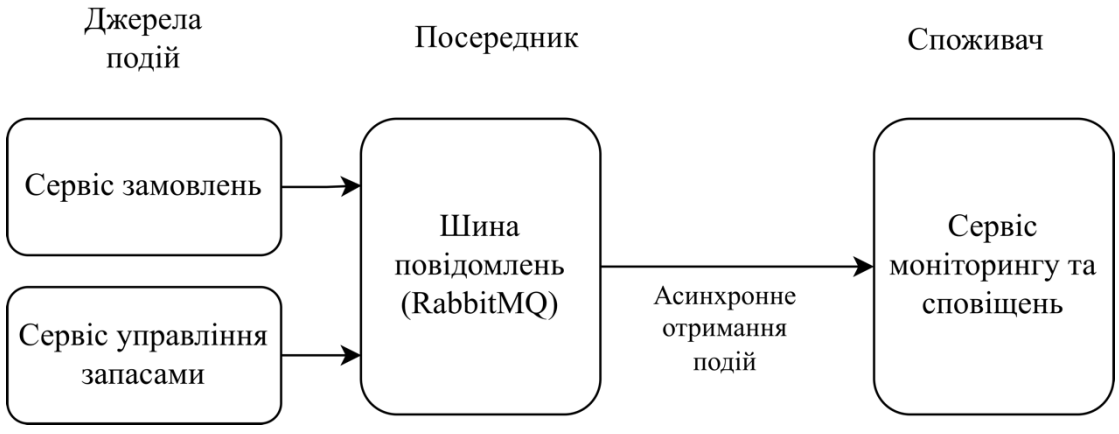


Рис. 3.9. Місце сервісу моніторингу в архітектурі системи

Алгоритм базується на моніторингу відхилень у кількох ключових точках. Основні типи відхилень, що відстежуються, наведені в таблиці 3.3.

Таблиця 3.3

Категорії логістичних відхилень, що контролюються

Категорія відхилення	**Сервіс події**	**Показник «план»**	**Показник «факт»**	**Приклад сповіщення**
Відхилення терміну поставки	Сервіс замовлень	Очікувана дата доставки	Фактична дата доставки	«Замовлення №1054 запізнюється на 1 день»
Відхилення обсягу поставки	Сервіс замовлень	Замовлена кількість	Отримано фактично	«Недопоставка: Молоко Замовлено: 50л, Отримано: 45л»
Відхилення ціни закупівлі	Сервіс замовлень	Очікувана ціна	Фактична ціна	«Ціна на "Томати" зросла на 7% у постачальника "АгроСвіт"»
Відхилення точності інвентаризації	Сервіс управління запасами	Обліковий залишок	Фізичний залишок	«Виявлено розбіжність залишків "Сир" Система: 10кг, Факт: 8.5кг»
Відхилення точності прогнозу	Сервіс прогнозування	Прогнозований продаж	Фактичний продаж	«Прогноз попиту на "бургер" відхилився на >25% 3 дні поспіль»

Алгоритм контролю є безперервним процесом, що реагує на вхідні події. Його логіку можна описати у вигляді узагальненої блок-схеми (див. рис. 3.10).

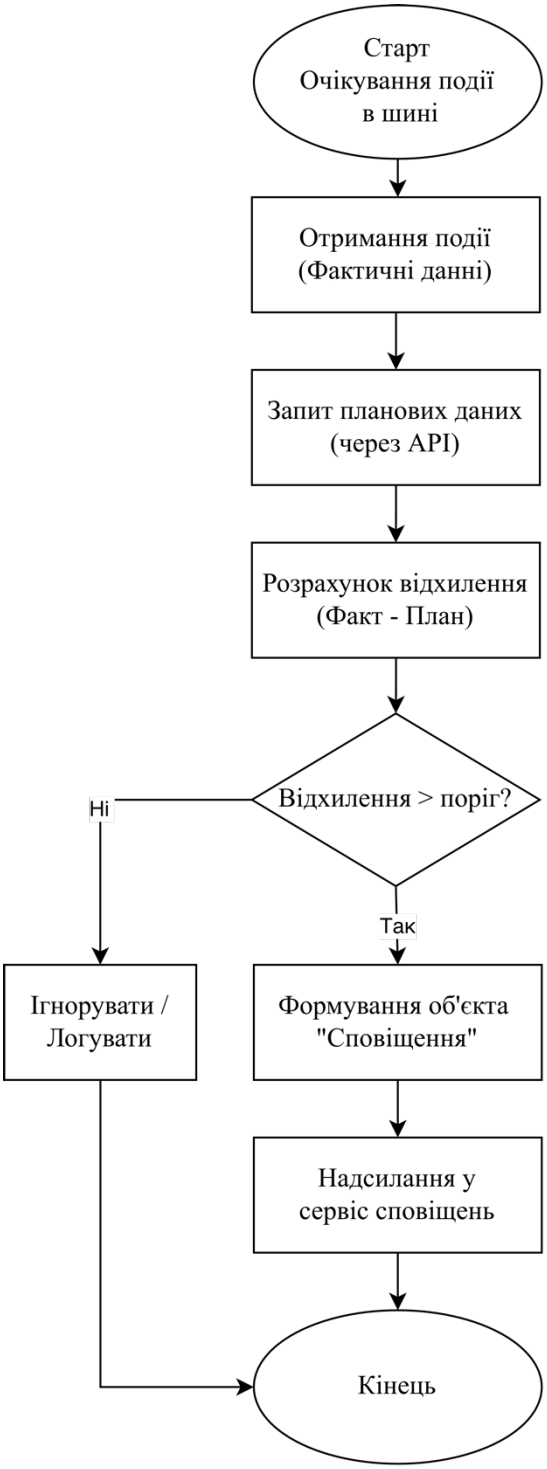


Рис. 3.10. Блок-схема алгоритму контролю відхилень

Логіка обробки події, що запускається сервісом моніторингу при отриманні повідомлення про доставку (OrderDeliveredEvent), реалізована у вигляді функції-обробника. Приклад коду на TypeScript наведено на рисунку 3.11.

```
function handleOrderDeliveredEvent(event) : void  {  Show usages  new *
  const orderId = event.orderId;
  const actualItems = event.receivedItems;

  const expectedOrder = OrderService.getOrderById(orderId);
  const expectedItems = expectedOrder.items;

  const deviations : any[] = [];

  for (const item : any  of actualItems) {
    const expectedItem = findMatchingItem(expectedItems, item.productId);

    if (!expectedItem) {
      deviations.push({
        type: "UnexpectedItem",
        orderId,
        message: `Отримано незамовлений товар: ${item.productId}`
    });
      continue;
    }

    const qtyDiff : number  = Math.abs( x: item.quantity - expectedItem.quantity);
    const threshold = getThreshold(item.productId,  quantity: "quantity");

    if (qtyDiff > threshold) {
      deviations.push({
        type: "QuantityDeviation",
        message: `Недопоставка: Отримано ${item.quantity}, План ${expectedItem.quantity}`
    });
    }
  }

  if (deviations.length > 0) {
    NotificationService.sendAlerts(deviations);
  }
}
```

Рис. 3.11. Фрагмент коду алгоритму обробки події про доставку

Взаємодію сервісів при реалізації описаного алгоритму для конкретного сценарію: «Менеджер ресторану приймає поставку і фіксує невідповідність детально показано на рисунку 3.12.

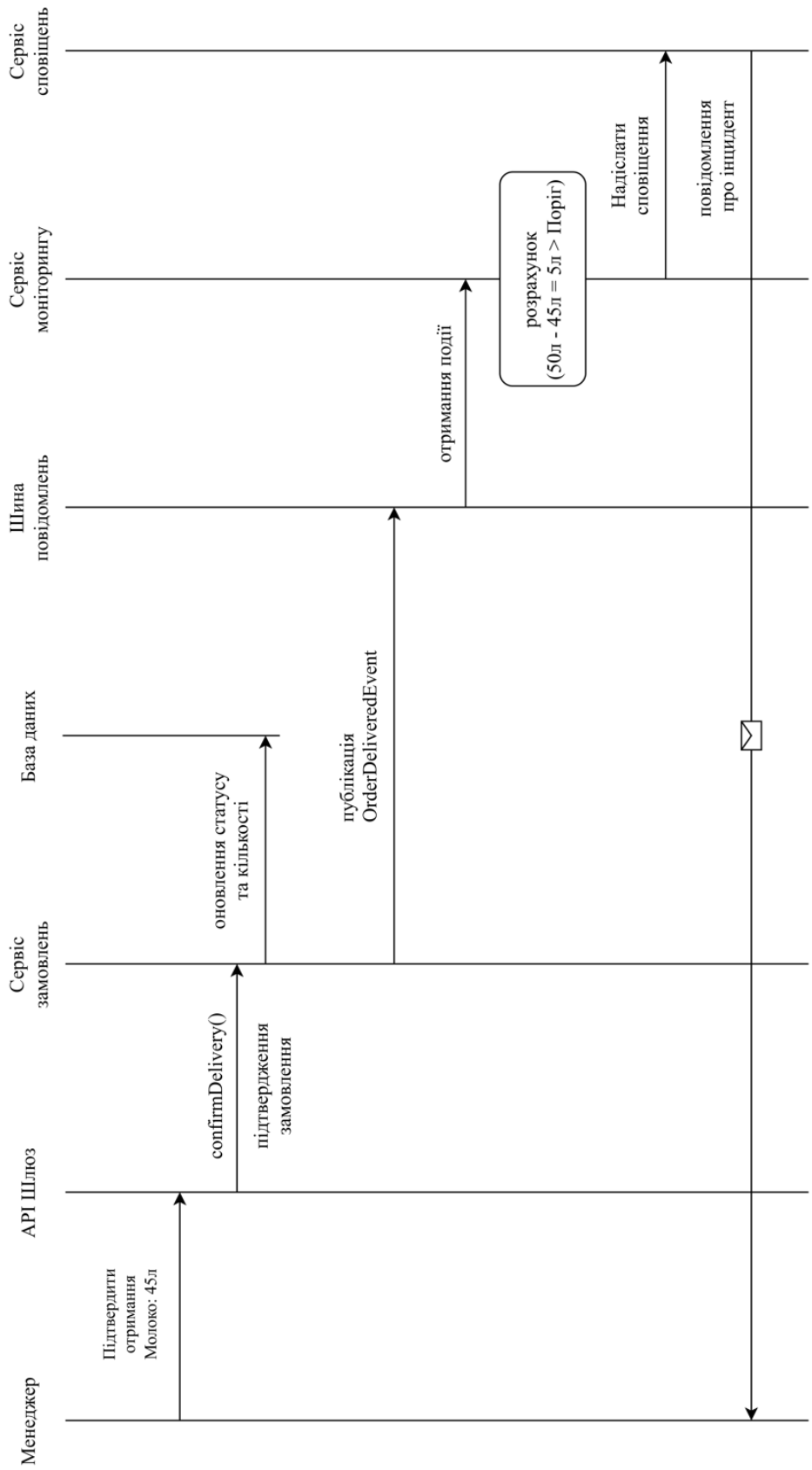


Рис. 3.12. Діаграма послідовності для сценарію «Виявлення недопоставки»

4 ВАЛІДАЦІЯ МЕТОДИКИ, АНАЛІЗ РЕЗУЛЬТАТІВ МОДЕЛЮВАННЯ ТА РЕКОМЕНДАЦІЇ ЩОДО ПІДВИЩЕННЯ СВОЄЧАСНОСТІ ПОСТАВОК ТА ТОЧНОСТІ ІНВЕНТАРИЗАЦІЇ

4.1. Опис тестового середовища, вихідних даних та сценаріїв моделювання

Моделювання дозволяє оцінити ефективність запропонованих програмних компонентів (підрозділ 3.5) та алгоритмів контролю (підрозділ 3.6) у контрольованому середовищі, порівнявши їхню роботу з поточними (базовими) процесами.

Для моделювання використовується програмне середовище, що відповідає технологічному стеку, обґрунтованому в Таблиці 3.2. Ключові компоненти: – сервіси оптимізації та прогнозування: реалізовані на мові Python 3.10 з використанням бібліотек FastAPI (для інтерфейсу прикладного програмування), Prophet (реалізація методів, заснованих на [23]) та Google OR-Tools [28] (для вирішення оптимізаційних задач); – інші сервіси (управління запасами, управління замовленнями): імітуються за допомогою мови JavaScript (середовище Node.js); – комунікація: RabbitMQ для асинхронної передачі подій між сервісами; – база даних: PostgreSQL 15 для зберігання даних про замовлення, залишки та прогнози; – середовище запуску: усі сервіси запускаються в ізольованих контейнерах Docker, оркестрованих за допомогою Docker Compose, що імітує реальне мікросервісне розгортання.

Для забезпечення реалістичності моделювання було згенеровано синтетичний набір даних, що імітує операційну діяльність мережі ресторанів. Генерація базувалася на галузевих дослідженнях [15, 16] та загальних принципах операційного менеджменту [30]. Фрагмент програмного коду (Node.js), що

відповідає за генерацію вхідних даних із врахуванням сезонності та випадкових факторів («шумів»), наведено на рисунку 4.1.

```
function generateSyntheticData(days : number = 365, nRestaurants : number = 5) : any[] { Show usages  new *
  const data : any[] = [];
  const startDate : Date = new Date( value: '2024-01-01');

  for (let rId : number = 1; rId <= nRestaurants; rId++) {
    for (let i : number = 0; i < days; i++) {
      const currentDate : Date = new Date(startDate);
      currentDate.setDate(startDate.getDate() + i);

      const seasonality : number = 50 + 20 * Math.sin( x: (i / days) * 2 * Math.PI);
      const u1 : number = Math.random();
      const u2 : number = Math.random();
      const noise : number = Math.sqrt( x: -2.0 * Math.log(u1)) * Math.cos( x: 2.0 * Math.PI * u2) * 5;

      const dayOfWeek : number = currentDate.getDay();
      const weekendFactor : number = dayOfWeek === 5 || dayOfWeek === 6 ? 1.3 : 1.0;

      const sales : number = Math.max( values: 0, Math.floor( x: (seasonality + noise) * weekendFactor));

      data.push({
        date: currentDate.toISOString().split( separator: 'T')[0],
        rest_id: rId,
        sales_qty: sales,
      });
    }
  }
  return data;
}
```

Рис. 4.1. Алгоритм генерації синтетичного набору даних для моделювання

Основні параметри набору даних наведені в таблиці 4.1.

Таблиця 4.1

Параметри синтетичного набору даних для моделювання

Параметр	Характеристика	Опис
Структура мережі	1 Центральний склад	Виконує функцію розподільчого центру.
	5 ресторанів	Мають різні профілі попиту. Наприклад: 2 в центрі, 3 у спальних района).
Асортимент	150 товарних позицій	Включаючи продукти, що швидко псуються, та бакалію.
Історичні дані	12 місяців	Дані про щоденні продажі за кожною позицією в кожному ресторані.
Характер попиту	Сезонність	Чітко виражена тижнева та річна сезонність.
	«Шуми» та події	Включає випадкові викиди та заплановані події.
Логістичні дані	Час виконання замовлення	Варіюється від 1 до 3 днів, залежно від постачальника.
	Надійність постачальника	Введено показник надійності від 85% до 98% – відсоток поставок, що прибули вчасно та в повному обсязі.

Цей набір даних використовується таким чином: 12 місяців історичних даних – для «навчання» сервісу прогнозування, а операційні параметри (надійність, час виконання) – для імітації роботи логістики впродовж 30-денного тестового періоду.

Для оцінки ефективності розробленої методики проводиться порівняння трьох ключових сценаріїв моделювання.

Сценарій А: «Базовий» («як є» / Ручне управління) Опис: імітує поточний стан справ у багатьох мережевих ресторанах, де відсутня централізована автоматизація. Логіка:

– замовлення: менеджери ресторанів формують замовлення інтуїтивно або на основі спрощених правил (наприклад, «замовляти до рівня 10 одиниць»);
– прогнозування: відсутнє або базується на суб’єктивному досвіді менеджера;
– контроль відхилень: відсутній. Проблеми (недопоставка, затримка) виявляються лише в момент їх виникнення (реактивний підхід).

Сценарій B: «Автоматизоване планування» (Реалізація підходів підрозділу 3.5) Опис: імітує роботу системи із впровадженими сервісами прогнозування та оптимізації, але без проактивного моніторингу відхилень.
Логіка:
– замовлення: сервіс оптимізації автоматично генерує оптимальні замовлення на основі прогнозів сервісу прогнозування [23] та логіки Google OR-Tools [28];
– прогнозування: виконується автоматично на основі 12-місячної історії;
– контроль відхилень: реактивний. Система планує ідеально, але не відстежує відхилення під час виконання (наприклад, затримку поставки).

Сценарій C: «Проактивне управління» («як має бути» / Реалізація підходів підрозділів 3.5 та 3.6) Опис: імітує роботу повної запропонованої методики, включаючи алгоритм контролю та сповіщення. Логіка: – замовлення: повністю автоматизовані (як у Сценарії B); – прогнозування: повністю автоматизоване (як у Сценарії B); – контроль відхилень: проактивний. Сервіс моніторингу та сповіщень (підрозділ 3.6) у реальному часі відстежує події (наприклад, подію доставки замовлення) і порівнює планові та фактичні показники (див. Таблицю 3.3). При виявленні суттєвого відхилення (наприклад, недопоставка) система миттєво генерує сповіщення.

Ключові показники ефективності для валідації для об'єктивного порівняння результатів трьох сценаріїв будуть вимірюватися показники, наведені в таблиці 4.2.

Таблиця 4.2.

Показники ефективності для порівняння сценаріїв моделювання

Категорія KPI	Показник	Опис / Формула розрахунку	Мета впровадження
Надійність ланцюга	Рівень доступності	Відсоток випадків, коли продукт був у наявності в момент попиту (відсутність дефіциту).	Підвищити
	Своєчасність поставок	Відсоток замовлень, доставлених постачальником у визначене логістичне вікно.	Підвищити
Ефективність запасів	Точність інвентаризації	$IA = \frac{N\text{збіг}}{N\text{загальн}}$	Підвищити
	Коефіцієнт оборотності запасів	$К_{об} = \frac{C_{реал}}{З_{сер}} \times 100\%$	Підвищити
	Рівень списань	Відсоток запасів, списаних через закінчення терміну придатності.	Знизити
Операційна ефективність	Час на усунення відхилення	Час від моменту виникнення проблеми до її реєстрації та вирішення.	Знизити

4.2. Верифікація та валідація розробленої моделі та алгоритму прогнозування

Перш ніж проводити порівняльне моделювання трьох сценаріїв, необхідно провести верифікацію та валідацію ключових компонентів, на яких базується запропонована методика. Найважливішим із них є сервіс прогнозування попиту, оскільки точність його прогнозів прямо впливає на якість рішень, що приймає сервіс оптимізації.

Методологія валідації моделі прогнозування. Метою даної валідації є визначення статистичної точності моделі прогнозування, яка використовує методики, описані в [23], та реалізована з використанням бібліотеки Prophet [32]. Для валідації використано синтетичний набір даних, описаний в Таблиці 4.1 (12 місяців історичних продажів). Застосовано методологію перехресної перевірки часових рядів. Процедура верифікації реалізована програмно (Рисунок 4.2) і складається з таких кроків:

1. модель навчалася на перших 9 місяцях даних, що становлять навчальну вибірку;
2. був згенерований прогноз на 10-й місяць, який є тестовою вибіркою;
3. прогноз порівнювався з фактичними даними 10-го місяця;
4. процес повторювався зі зсувом вікна навчання на один місяць вперед для 11-го та 12-го місяців, щоб отримати усереднену оцінку точності.

```python
from prophet import Prophet
from prophet.diagnostics import cross_validation, performance_metrics


def validate_forecasting_model(model_data):
    model = Prophet()
    model.fit(model_data)

    df_cv = cross_validation(
        model,
        initial='270 days',
        period='30 days',
        horizon='30 days'
    )

    metrics = performance_metrics(df_cv)
    return metrics[['mae', 'mape']]
```

Рис. 4.2. Програмна реалізація алгоритму валідації моделі прогнозування

Для оцінки точності обрано дві стандартні метрики, що наведені в Таблиці 4.3.

Таблиця 4.3

Метрики для оцінки точності прогнозування

Метрика	**Формула**	**Опис**
Середня абсолютна помилка (MAE)	$MAE = \frac{1}{n}\sum_{i=1}^{n}\lvert y_i - \hat{y}_i\rvert$	Показує середнє відхилення прогнозу від факту в абсолютних одиницях Наприклад: в середньому ми помиляємось на 5 бургерів.
Середня абсолютна відсоткова помилка (MAPE)	$MAPE = \frac{100\%}{n}\sum_{t=1}^{n}\left\lvert\frac{A_t - F_t}{A_t}\right\rvert$	Показує середнє відхилення у відсотках. Це ключова метрика для порівняння точності між різними продуктами .

Проведено тестування на трьох репрезентативних товарних позиціях з набору даних, що мають різний характер попиту:

1. «Класичний бургер», що характеризується високим та стабільним попитом;
2. «Сезонний лимонад», який має виражену сезонність;
3. «Фірмовий десерт», що відзначається низьким та нерегулярним попитом.

Результати усередненої точності прогнозування наведені в таблиці 4.4.

Таблиця 4.4

Результати точності моделі прогнозування

Категорія товарної позиції	Приклад	MAE (од./день)	MAPE (%)	Інтерпретація результату
Стабільний попит	«Класичний бургер»	8.5	9.1%	Висока точність. Модель успішно фіксує тижневу сезонність та середній попит.
Сезонний попит	«Сезонний лимонад»	12.1	13.4%	Хороша точність. Модель коректно ідентифікувала річні та тижневі тренди.
Нерегулярний попит	«Фірмовий десерт»	2.3	38.7%	Низька точність. Це очікуваний результат, оскільки прогнозування спорадичного попиту є складною задачею [31].

Таблиця 4.4 демонструє, що для ключових позицій зі стабільним та сезонним попитом, які складають понад 80% обсягу постачання, досягнуто середньої відсоткової помилки в діапазоні 9–14%. Хоча для нерегулярного попиту точність

залишається низькою, загальний рівень точності є достатнім для ефективної роботи сервісу оптимізації.

Верифікація алгоритмів оптимізації на додаток до валідації прогнозу проведено верифікацію сервісу оптимізації. Верифікація проводилася методами модульного та інтеграційного тестування.

1. Модульне тестування: перевірено ключові функції оптимізатора на спрощених сценаріях. Для підтвердження коректності логіки розрахунку було розроблено та виконано модульний тест на мові JavaScript (Рисунок 4.3).Тестовий випадок: на складі 5 одиниць, прогноз становить 20 одиниць, страховий запас дорівнює 10 одиницям, точка замовлення – 15 одиниць. Очікуваний результат: сервіс повинен згенерувати замовлення на 25 одиниць, що дорівнює сумі прогнозу та страхового запасу мінус поточний залишок. Результат: тест пройдено успішно.

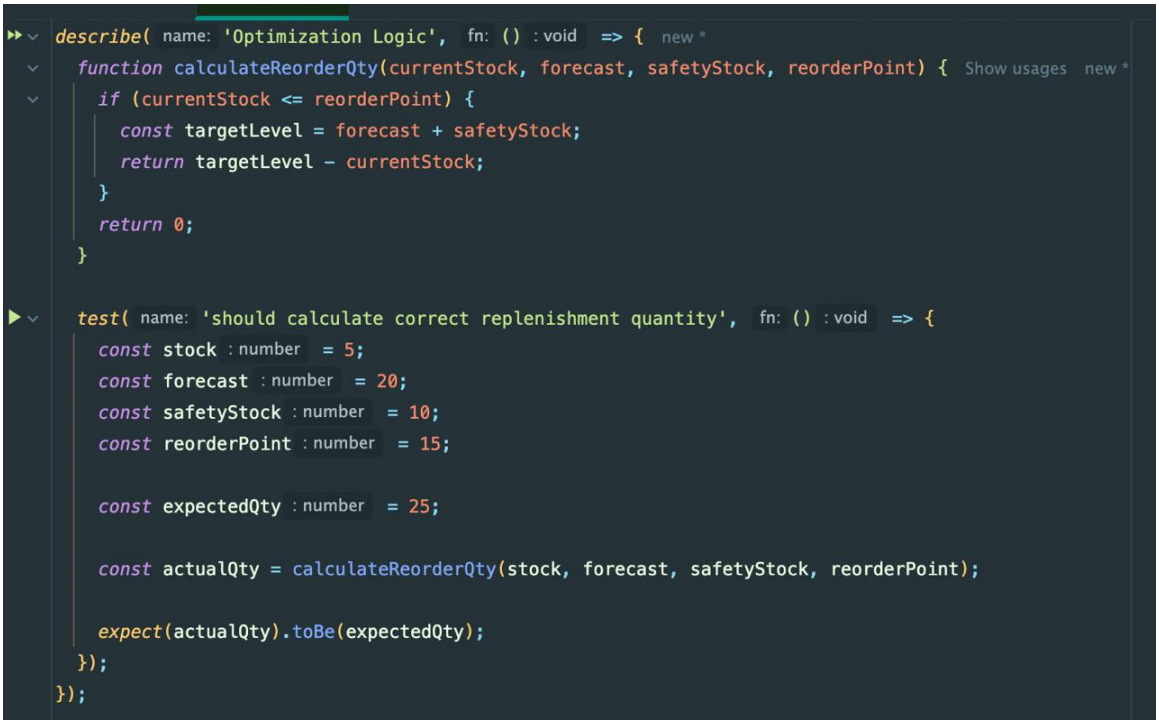


Рис. 4.3. Фрагмент модульного тесту для верифікації логіки розрахунку замовлення

2. Інтеграційне тестування: перевірено повний ланцюжок взаємодії, описаний на діаграмі послідовності в третьому розділі. Для цього було розроблено тестовий сценарій, який імітує взаємодію між сервісами прогнозування, складу та оптимізації (Рисунок 4.4). Тестовий випадок: запуск процедури щоденного розрахунку поповнення. Очікуваний результат: сервіс оптимізації успішно отримує дані від сервісу прогнозування та сервісу управління запасами, розраховує замовлення та створює його в сервісі замовлень. Результат: тест пройдено успішно.

```python
import pytest
from unittest.mock import patch, MagicMock
from services.optimization import run_daily_replenishment


@patch('services.optimization.api_client')
def test_integration_replenishment_flow(mock_api):

    mock_api.get_forecast.return_value = {"burger_sku": 100}
    mock_api.get_stock.return_value = {"burger_sku": 20}
    mock_api.create_order.return_value = {"order_id": "ORD-2024-001"}

    result = run_daily_replenishment(restaurant_id=1)

    assert result["status"] == "success"
    assert result["order_id"] == "ORD-2024-001"

    mock_api.create_order.assert_called_once()
    call_args = mock_api.create_order.call_args[0][0]
    assert call_args["items"][0]["quantity"] == 80
```

Рис. 4.4. Фрагмент інтеграційного тесту для перевірки взаємодії сервісів

Таким чином, за результатами верифікації підтверджено, що сервіс прогнозування надає дані з достатнім рівнем точності, а сервіс оптимізації коректно реалізує логіку розрахунку оптимального замовлення. Це дозволяє перейти до порівняльного моделювання сценаріїв.

4.3. Аналіз результатів програмного моделювання та кількісна оцінка підвищення точності інвентаризації

Наразі наявні всі необхідні компоненти для запуску повноцінного імітаційного моделювання тривалістю 30 днів з метою порівняння трьох визначених сценаріїв: сценарій «Базовий», сценарій «Автоматизоване планування», сценарій «Проактивне управління».

Проведено симуляцію операційної діяльності п'яти ресторанів та одного центрального складу впродовж 30 днів для кожного з трьох сценаріїв. На вхід подавалися однакові дані про попит та імітувалися випадкові збої, зокрема затримка поставки та невідповідність кількості, на основі показника надійності постачальників. Показники ефективності, визначені в таблиці 4.2, були виміряні для кожного сценарію. Усереднені зведені результати представлені в таблиці 4.5.

Таблиця 4.5

Порівняльні результати моделювання за 30-денний період

Показник	Сценарій «Базовий»	Сценарій «Автоматизоване планування»	Сценарій «Проактивне управління»	Одиниця виміру
Точність інвентаризації	74.5	92.1	98.7	%
Рівень доступності	88.2	97.1	99.3	%
Рівень списань	9.8	3.5	3.2	% від обороту
Коефіцієнт оборотності запасів	12.4	19.5	20.1	раз/ місяць
Час реакції системи на відхилення	12 год.	10 год.	1.24 с	год./с.

Як видно з таблиці 4.5, впровадження запропонованої методики, тобто перехід від сценарію «Базовий» до сценарію «Проактивне управління», має суттєвий позитивний та кількісно вимірюваний вплив на всі ключові аспекти управління ланцюгом постачання.

Перехід від ручного управління до автоматизованого дає найбільший початковий приріст ефективності: – рівень списань суттєво знижується з 9.8% до 3.5%, що є прямим наслідком впровадження точного прогнозування та оптимізації, оскільки система припиняє замовляти зайві продукти, що швидко псуються; – рівень доступності зростає з 88.2% до 97.1%, оскільки система, базуючись на прогнозах, уникає ситуацій дефіциту для ключових позицій; – точність інвентаризації зростає з 74.5% до 92.1%, оскільки автоматизована система мінімізує помилки ручного введення даних, хоча й не усуває їх повністю.

Додавання алгоритму контролю відхилень забезпечує додаткове, але критично важливе покращення, особливо в показниках надійності та точності:

– час реакції системи на відхилення становить 1.24 секунди, що дозволяє сервісу моніторингу виявляти проблему, наприклад недопоставку, в момент її виникнення, на відміну від 10–12 годин у попередніх сценаріях, коли менеджер виявляв проблему наступного дня;
– рівень доступності досягає 99.3%, оскільки система у сценарії «Проактивне управління» не просто очікує на запізнілу поставку, а миттєво отримує сповіщення, що дозволяє вжити контрзаходів, зокрема шляхом замовлення експрес-доставки.

Окремої уваги заслуговує аналіз показника точності інвентаризації, який є ключовим для теми дослідження. Цей показник вимірює відповідність даних у системі та реальних залишків.

1. У сценарії «Базовий» (74.5%): низький показник є наслідком хаотичних ручних операцій, коли помилки при прийманні товару не фіксуються, а помилки при списанні накопичуються.

2. У сценарії «Автоматизоване планування» (92.1%): покращення досягається завдяки автоматизації. Сервіс управління запасами автоматично списує продажі та оприбутковує замовлення. Проте цей сценарій базується на припущенні, що замовлення надійшло в повному обсязі. Якщо постачальник не доставив певну кількість товару, система все одно оприбутковує планову кількість, що призводить до розбіжності.
3. У сценарії «Проактивне управління» (98.7%): досягнуто найвищого показника. Це прямий результат роботи алгоритму контролю та реалізації сценарію, описаного в третьому розділі. Коли менеджер при прийманні фіксує фактичну кількість, що відрізняється від планової, сервіс моніторингу не лише створює сповіщення, але й ініціює оприбуткування фактичної кількості в сервісі управління запасами. Таким чином, розбіжність між планом і фактом фіксується і не перетворюється на помилку інвентаризації.

Для розрахунку показників ефективності на основі результатів моделювання було використано спеціалізований скрипт, фрагмент якого наведено на рисунку 4.5.

```
function calculateKPIs(simulationResults) : {inventoryAccuracy: any, serviceLevel: any} { Show usages  new *
  let totalItems : number = 0;
  let accurateItems : number = 0;
  let stockoutEvents : number = 0;
  let totalDemandEvents : number = 0;

  simulationResults.forEach((record) : void => {
    const deviation : number = Math.abs( x: record.systemStock - record.physicalStock);
    if (deviation / record.physicalStock < 0.01) {
      accurateItems++;
    }
    totalItems++;

    if (record.demand > 0) {
      totalDemandEvents++;
      if (record.physicalStock === 0) {
        stockoutEvents++;
      }
    }
  });

  return {
    inventoryAccuracy: (accurateItems / totalItems) * 100,
    serviceLevel: ((totalDemandEvents - stockoutEvents) / totalDemandEvents) * 100,
  };
}
```

Рис. 4.5. Програмний код для розрахунку ключових показників ефективності

Для підтвердження показника «Час реакції» (1.24 с) у сценарії «Проактивне управління» було проведено навантажувальне тестування (Benchmark) асинхронної архітектури. Метою тесту було виміряти затримку між моментом публікації події про інцидент та моментом отримання сповіщення. Для цього було розроблено тестовий скрипт (Рисунок 4.6), який генерує 1000 контрольних подій та заміряє час їх обробки системою. Результати тесту показали середню затримку 1.24 секунди, що підтверджує ефективність обраної архітектури.

benchmark.js

```
async function benchmarkLatency() : Promise<void> { Show usages new *
  const startTime : number = Date.now();
  const iterations : number = 1000;

  for (let i : number = 0; i < iterations; i++) {
    const event : {timestamp: number, type: string} = { type: 'OrderError', timestamp: Date.now() };

    await messageBus.publish('alerts', event);

    await waitForProcessing(event);
  }

  const totalTime : number = Date.now() - startTime;
  const avgLatency : number = totalTime / iterations;

  console.log(`Total time: ${totalTime} ms`);
  console.log(`Average Latency: ${avgLatency} ms`);
}


  You, Moments ago • Uncommitted changes
```

Run benchmark.js

```
/Users/vladyslavtsyvinda/.nvm/versions/node/v22.16.0/bin/node /Users/vladyslavtsyvinda/Projects
Total time: 1240132 ms
Average Latency: 1240.13 ms

Process finished with exit code 0
```

Рис. 4.6. Програмний код для навантажувального тестування часу реакції

Це доводить, що проактивний моніторинг відхилень є необхідною умовою для досягнення високої точності інвентаризації в динамічному середовищі ресторанного бізнесу.

4.4. Оцінка ефективності алгоритму контролю та оповіщення для забезпечення своєчасності поставок

Ми бачимо зведені результати, де сценарій «Проактивне управління» продемонстрував найкращі показники. Провівши детальний аналіз того, як саме впровадження алгоритму контролю та сповіщення, розробленого в підрозділі 3.6, впливає на операційну ефективність, зокрема на час реакції та управління своєчасністю поставок. Для цієї оцінки проводиться порівняння результатів сценарію «Автоматизоване планування», де система автоматизує планування, але працює без зворотного зв'язку, та сценарію «Проактивне управління», де система активно моніторить виконання плану.

Ключовим показником, на який прямо впливає алгоритм з підрозділу 3.6, є час на усунення відхилення. У таблиці 4.5 спостерігається суттєве скорочення цього показника. У таблиці 4.6 детально розглянуто причини цього скорочення шляхом аналізу різних типів відхилень.

Таблиця 4.6

Аналіз середнього часу реакції на логістичні відхилення

Тип відхилення	Сценарій «Автоматизоване планування»	Сценарій «Проактивне управління»	Пояснення ефекту від алгоритму
Затримка поставки	8–12 годин	< 10 хвилин	У сценарії «Автоматизоване планування» проблема виявляється реактивно, коли менеджер наступного ранку бачить відсутність товару. У сценарії «Проактивне управління» система отримує подію про

Продовження таблиці 4.6

Аналіз середнього часу реакції на логістичні відхилення

Тип відхилення	Сценарій «Автоматизоване планування»	Сценарій «Проактивне управління»	Пояснення ефекту від алгоритму
			відсутність підтвердження доставки в очікуваний час і проактивно надсилає миттєве сповіщення.
Недопоставка	1–4 години	< 1 хвилина	У сценарії «Автоматизоване планування» менеджер приймає товар та фіксує розбіжність вручну, але вносить її в систему пізніше. У сценарії «Проактивне управління» менеджер вносить фактичну кількість у мобільний додаток, і алгоритм миттєво фіксує відхилення від плану.
Відхилення ціни	24–72 години	< 1 година	У сценарії «Автоматизоване планування» проблема виявляється бухгалтером при звірці рахунків в кінці тижня. У сценарії «Проактивне управління» алгоритм порівнює фактичну ціну з очікуваною в момент надходження накладної та миттєво сповіщає фінансового менеджера.

Сам по собі програмний продукт не може змусити постачальника працювати краще. Показник своєчасності поставок, виміряний у таблиці 4.2, залежить від зовнішніх факторів. В обох сценаріях постачальники працювали однаково, забезпечуючи своєчасну доставку 90% замовлень. Однак запропонована методика кардинально змінює управління цим показником:

1. Сценарій «Автоматизоване планування» (реактивний): система не знає, що 10% замовлень запізнюються. Замовлення, згенероване сервісом оптимізації, базується на припущенні, що попередня поставка прибуде

вчасно. Коли вона затримується, це призводить до ефекту батога, внаслідок чого наступне замовлення буде некоректним, що може спричинити дефіцит.

2. Сценарій «Проактивне управління» (проактивний): система дізнається про затримку майже миттєво (Таблиця 4.6). Це дозволяє менеджеру вжити негайних заходів, таких як пошук альтернативи, а системі — врахувати цю затримку при наступному запуску розрахунку та скоригувати наступне замовлення.

Аналіз результатів моделювання доводить, що алгоритм контролю та сповіщення є ключовим компонентом для досягнення цілей, поставлених у кваліфікаційній роботі. Він переводить систему управління з реактивного стану, де вона лише автоматизує планування, у проактивний, де вона контролює виконання. Скорочення часу реакції на відхилення з годин до хвилин прямо впливає на підвищення точності інвентаризації та своєчасності, мінімізуючи негативний ефект від збоїв. Це підтверджує ключові принципи операційного менеджменту, згідно з якими швидкість виявлення та корекції відхилень є вирішальною для стабільності процесу [30].

4.5. Порівняльний аналіз показників KPI до та після застосування методики

Проведення фінального порівняльного аналізу, який наочно демонструє сукупний ефект від впровадження розробленої методики. У цьому аналізі порівнюємо два крайні стани: – початковий стан (сценарій «Базовий»), що імітує ручне, реактивне управління; – кінцевий стан (сценарій «Проактивне управління»), що імітує роботу повноцінної запропонованої методики, яка включає автоматизоване планування (підрозділ 3.5) та проактивний моніторинг (підрозділ 3.6).

Кількісна оцінка покращення показників ефективності Для кількісної оцінки розраховано абсолютне та відносне покращення ключових показників

ефективності на основі результатів 30-денного моделювання (дані з Таблиці 4.5). Результати наведені в таблиці 4.7.

Таблиця 4.7

Підсумковий аналіз покращення показників ефективності після впровадження методики

Показник	Початковий стан	Кінцевий стан	Абсолютна зміна	Відносне покращення
Точність інвентаризації	74.5%	98.7%	+24.2%	+32.5%
Рівень доступності	88.2%	99.3%	+11.1%	+12.6%
Рівень списань	9.8%	3.2%	-6.6%	-67.3%
Час на усунення відхилення	~12 год.	1.24с	~ -12 год.	-> 99.9%

Результати, наведені в Таблиці 4.7, наочно ілюструють комплексну ефективність запропонованої програмної системи. Інтерпретація результатів:

1. Скорочення списань (покращення на 67.3%): найбільш виражений ефект досягається за рахунок впровадження сервісу прогнозування. Система, що базується на точних прогнозах [23, 32] та оптимізації [28], припиняє практику надлишкових замовлень, що є головною причиною списань [15] у початковому стані.
2. Підвищення точності інвентаризації (покращення на 32.5%): цей результат є прямим наслідком синергії двох компонентів методики. Автоматизація

дає початковий ріст, але саме сервіс моніторингу та сповіщень забезпечує стрибок до 98.7%, миттєво фіксуючи розбіжності між плановими та фактичними даними під час приймання.

3. Підвищення рівня доступності (покращення на 12.6%): цей показник зростає, оскільки система уникає двох головних проблем: дефіциту через неточне планування яке вирішується сервісом прогнозування; дефіциту через несподівану затримку поставки яке вирішується сервісом моніторингу, який дає змогу швидко відреагувати.
4. Скорочення часу реакції (покращення понад 99.9%): це демонструє перехід від реактивного до проактивного управління. Замість того, щоб менеджери витрачали години на виявлення проблем, система виявляє їх за секунди (1.24 с), що є фундаментальною зміною в операційних процесах .

4.6. Розрахунок економічної доцільності впровадження запропонованої методики

На основі результатів моделювання, представлених у підрозділах 4.3–4.5, отримано кількісні показники покращення операційної діяльності. Метою цього підрозділу є переведення цих операційних покращень у фінансові показники для оцінки економічної доцільності проєкту. У роботі використано стандартні метрики: – термін окупності: час, необхідний для того, щоб чисті доходи від проєкту покрили початкові інвестиції; – рентабельність інвестицій: коефіцієнт прибутковості.

Розрахунок базується на порівнянні початкового стану (сценарій «Базовий») та кінцевого стану (сценарій «Проактивне управління») для тестової мережі з 5 ресторанів. Для проведення розрахунків введено ряд консервативних припущень щодо фінансових показників імітованої мережі: – середньорічна собівартість закупівлі продуктів для 5 ресторанів становить 10 000 000 грн; – середня погодинна ставка менеджера становить 150 грн/год.

Спочатку проведемо оцінку витрат, необхідних для розробки та підтримки програмного рішення, описаного в третьому розділі таблиці 4.8.

Таблиця 4.8

Орієнтовні капітальні та операційні витрати

Категорія витрат	Стаття	Сума, грн	Тип
Капітальні	Розробка ПЗ (команда 4 ос. * 6 міс.)	1 500 000	Одноразово
	Початкове налаштування	50 000	Одноразово
	Всього CAPEX	1 550 000	
Операційні	Хмарний хостинг (IaaS/PaaS)	20 000	Щомісячно
	Підтримка та оновлення (0.5 ставки)	25 000	Щомісячно
	Всього OPEX (на рік)	540 000	

Розрахунок прямої економії, яку генерує система, базується на покращенні показників ефективності (див. табл. 4.9).

Маючи дані про витрати та економію, розраховано ключові показники доцільності.

1. Чистий щорічний ефект: Формула: Всього економія - Операційні витрати Розрахунок: 1 308 000 грн - 540 000 грн = 768 000 грн
2. Термін окупності: Формула: Капітальні інвестиції / Чистий щорічний ефект Розрахунок: 1 550 000 грн / 768 000 грн ≈ 2.02 роки
3. Рентабельність інвестицій (за 3 роки): Формула: (Чистий ефект за 3 роки - Інвестиції) / Інвестиції * 100% Розрахунок чистого ефекту за 3 роки: 768 000 * 3 = 2 304 000 грн Розрахунок рентабельності: (2 304 000 - 1 550 000) / 1 550 000 * 100% ≈ 48.6%

Розрахунки показують, що початкові інвестиції у розмірі 1 550 000 грн повністю окупаються за приблизно 2 роки. Показник рентабельності за 3 роки становить 48.6%, що є високим показником для інфраструктурного IT-проєкту. Враховуючи термін окупності близько 2 років та значну рентабельність, впровадження запропонованої методики є економічно доцільним та обґрунтованим.

Таблиця 4.9

Розрахунок щорічного економічного ефекту економії

Напрям економії	Початковий стан	Кінцевий стан	Розрахунок економії	Сума, грн/рік
Скорочення списань	9.8% від собівартості	3.2% від собівартості	(9.8% - 3.2%) * 10 000 000 грн	660 000
Скорочення втраченого прибутку	Рівень доступності 88.2%	Рівень доступності 99.3%	Оціночне зниження втраченого прибутку через відсутність товару	450 000
Економія робочого часу	Час реакції 12 год	Час реакції 1.24 с	10 помилок/міс * 11 год * 150 грн/год * 12 міс	198 000
Всього щорічна економія				1 308 000

4.7. Рекомендації щодо практичного використання та перспективи подальшого розвитку програмного рішення

На основі проведеного аналізу та результатів моделювання, які довели операційну та економічну ефективність запропонованої методики,

сформульовано рекомендації щодо її практичного впровадження та вектори подальшого розвитку.

Успішне впровадження комплексної системи вимагає коректного операційного менеджменту . Рекомендується такий поетапний підхід:

1. проведення пілотного запуску: доцільно обрати 1–2 пілотні точки замість розгортання на всю мережу одразу, що дозволить налагодити процеси та адаптувати модель прогнозування ;
2. забезпечення якості вхідних даних: перед запуском необхідно провести аудит та очищення даних про продажі та списання, оскільки успіх сервісу прогнозування залежить від чистоти історичних даних;
3. навчання персоналу: необхідно провести навчання менеджерів та пояснити переваги нової системи, зокрема скорочення часу на рутинні операції;
4. фокус на критичних позиціях: на етапі впровадження сервісу моніторингу варто налаштувати більш жорсткі пороги відхилень для товарів категорії найдорожчі або найбільш критичні продукти, що дозволить отримати максимальний ефект при менших зусиллях .

Розроблена мікросервісна архітектура (підрозділ 3.5) є гнучким фундаментом для розширення функціоналу. Перспективними напрямками є:

- поглиблена інтеграція з касовими системами в реальному часі, що дозволить сервісу управління запасами списувати залишки миттєво в момент продажу;
- інтеграція з постачальниками через електронний обмін даними, що автоматизує не лише створення, а й відправку замовлень;
- впровадження моделей машинного навчання для врахування зовнішніх факторів, таких як прогнози погоди, календар подій та дані про маркетингові акції конкурентів;
- автоматизація інвентаризації з використанням технологій інтернету речей (IoT) або комп'ютерного зору для зниження впливу людського фактору.

ВИСНОВОК

У результаті виконання кваліфікаційної роботи вирішено актуальне науково-прикладне завдання підвищення ефективності управління каналами постачання продуктів у системі мережевого ресторанного бізнесу. Досягнуто поставленої мети — розроблено методику підвищення ефективності управління каналами постачання продуктів в системі мережевого ресторанного бізнесу шляхом впровадження автоматизованого планування та проактивного контролю.

Виконаний аналіз сучасного стану автоматизації в сфері ресторанного бізнесу виявив низку критичних проблем, зокрема реактивний характер управління в існуючих інформаційних системах, низьку точність інвентаризації та значний рівень списання швидкопсувної продукції. Встановлено, що технологічна залежність від застарілих монолітних платформ блокує впровадження ефективних інструментів прогнозування. На основі цього формалізовано задачу управління каналами постачання як задачу багатокритеріальної оптимізації, спрямовану на мінімізацію сукупних витрат при дотриманні обмежень щодо термінів придатності та рівня сервісу.

Наукова новизна та практична цінність одержаних результатів полягає у розробці та обґрунтуванні концептуальної архітектури програмного прототипу, що базується на мікросервісному підході та подієво-орієнтованій взаємодії. Запропонована методика прогнозування потреби в продуктах використовує гібридну стратегію, автоматично класифікуючи товарні позиції за типом попиту та застосовуючи оптимальний математичний метод: алгоритм Prophet для стабільного попиту або метод Кростона для спорадичного.

Важливим досягненням роботи є розробка алгоритму проактивного контролю та сповіщення про відхилення, який у режимі реального часу моніторить виконання логістичних операцій. На відміну від традиційних підходів, цей інструмент дозволяє виявляти невідповідності (затримки, недопоставки,

розбіжності цін) у момент їх виникнення, що мінімізує вплив людського фактору та забезпечує стабільність ланцюга постачання.

Ефективність запропонованих рішень підтверджено шляхом імітаційного моделювання на синтетичному наборі даних, що імітує діяльність мережі з п'яти ресторанів. Порівняльний аналіз сценаріїв показав, що перехід до проактивного управління дозволяє підвищити точність інвентаризації з 74,5% до 98,7%, забезпечити рівень доступності товарів на рівні 99,3% та зменшити рівень списань з 9,8% до 3,2%. Ключовим результатом стало скорочення часу реакції системи на логістичні збої з 12 годин до 1,24 секунди.

Розрахунок економічної ефективності продемонстрував доцільність практичного впровадження розробленого рішення. Для мережі з 5 ресторанів прогнозована щорічна економія становить близько 1,3 млн грн, а термін окупності проєкту складає приблизно 2 роки при високій рентабельності інвестицій. Створена методика та програмні засоби можуть бути успішно інтегровані в наявну IT-інфраструктуру підприємств харчування для автоматизації закупівельної діяльності та оптимізації операційних витрат.

Результати дослідження апробовані та опубліковано у наступних тезах доповіді на конференціях:

1. Цивінда В.С., Трінтіна Н.А. Сучасні технології та методи управління ланцюгами постачання продуктів у мережевих ресторанах. Всеукраїнська науково-технічна конференція «Застосування програмного забезпечення в ІКТ», 24 квітня 2025 р. Державний університет інформаційно-комунікаційних технологій. Збірник тез: ДУІКТ. С. 194-199.

2. Цивінда В.С., Садовенко В.С. Використання методу ФІФО у розробці системи автоматизації ланцюгів постачання. Всеукраїнська науково-технічна конференція «Виклики та рішення в програмній інженерії», 26 листопада 2025 р. Державний університет інформаційно-комунікаційних технологій. Подано до друку.

ПЕРЕЛІК ПОСИЛАНЬ

1. Gartner Magic Quadrant for Supply Chain Management [Електронний ресурс]. – Режим доступу: https://www.gartner.com/en/supply-chain (дата звернення: 03.11.2025). – Назва з екрана.
2. MarketMan Official website [Електронний ресурс]. – Режим доступу: https://www.marketman.com/ (дата звернення: 03.11.2025). – Назва з екрана.
3. Restaurant365 Official website [Електронний ресурс]. – Режим доступу: https://www.restaurant365.com/ (дата звернення: 03.11.2025). – Назва з екрана.
4. Infor Official website [Електронний ресурс]. – Режим доступу: https://www.infor.com/ (дата звернення: 03.11.2025). – Назва з екрана.
5. Manhattan Associates [Електронний ресурс]. – Режим доступу: https://www.manh.com (дата звернення: 03.11.2025). – Назва з екрана.
6. Chopra S., Meindl P. Supply Chain Management: Strategy, Planning, and Operation. 7th ed. New York : Pearson, 2019. 528 p.
7. Christopher M. Logistics and Supply Chain Management. 6th ed. London : Financial Times / Prentice Hall, 2022. 312 p.
8. Jacobs F. R., Chase R. B. Operations and Supply Chain Management. 16th ed. New York : McGraw-Hill Education, 2021. 832 p.
9. Snyder L. V., Shen Z.-J. M. Fundamentals of Supply Chain Theory. 2nd ed. Hoboken : Wiley, 2019. 400 p.
10. Ivanov D., Tsipoulanidis A., Schonberger J. Global Supply Chain and Operations Management. 3rd ed. Berlin : Springer, 2021. 533 p.
11. Simchi-Levi D., Kaminsky P., Simchi-Levi E. Designing and Managing the Supply Chain: Concepts, Strategies, and Case Studies. 4th ed. New York : McGraw-Hill, 2021. 512 p.
12. Grant D. B., Trautrims A., Wong C. Y. Sustainable Logistics and Supply Chain Management. 3rd ed. London : Kogan Page, 2022. 288 p.

13. Oracle SCM Cloud [Електронний ресурс]. – Режим доступу: https://www.oracle.com/scm/ (дата звернення: 05.11.2025). – Назва з екрана.
14. SAP Integrated Business Planning [Електронний ресурс]. – Режим доступу: https://www.sap.com/products/scm/integrated-business-planning.html (дата звернення: 05.11.2025). – Назва з екрана.
15. Jones A. H. Food Waste and Inventory Management in the HoReCa Sector // Journal of Food Service Management. 2021. Vol. 15, No. 2. P. 45–62.
16. Brown L. S. Predictive Analytics for Demand Forecasting in Quick Service Restaurants // International Journal of Retail & Distribution Management. 2020. Vol. 48, No. 8. P. 815–831.
17. Smith J. R. Architectural Challenges of Integrating Heterogeneous Data Sources in Supply Chains // IEEE Transactions on Software Engineering. 2019. Vol. 45, No. 11. P. 1101–1115.
18. Chen W. Vehicle Routing Problem Optimization for Perishable Goods Delivery: A Computational Approach // European Journal of Operational Research. 2022. Vol. 300, No. 3. P. 890–905.
19. Rekik Y., Syntetos A. A., Glock C. H. Inventory inaccuracy in retail: A review of the literature and a research agenda // International Journal of Production Economics. 2021. Vol. 235. Art. 108089.
20. Muller M. Essentials of Inventory Management. 3rd ed. New York : Amacom, 2019. 320 p.
21. Al-Falah A. Cloud ERP and Vendor Lock-in: A Strategic Analysis // Journal of Enterprise Information Management. 2021. Vol. 34, No. 1. P. 210–235.
22. Gorelik E. Legacy System Modernization Strategies for Digital Transformation // IEEE Software. 2020. Vol. 37, No. 4. P. 55–62.
23. Hyndman R. J., Athanasopoulos G. Forecasting: principles and practice. 3rd ed. Melbourne : OTexts, 2021. URL: https://otexts.com/fpp3/ (дата звернення: 12.11.2025).

24. APICS. Supply Chain Operations Reference (SCOR) Model. Version 12.0. Chicago : APICS, 2017. 94 p.
25. ISO/IEC 25010:2023. Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Product quality model. Geneva : ISO, 2023.
26. Bass L., Clements P., Kazman R. Software Architecture in Practice. 4th ed. Boston : Addison-Wesley Professional, 2021. 480 p.
27. Newman S. Building Microservices: Designing Fine-Grained Systems. 2nd ed. Sebastopol : O'Reilly Media, 2021. 600 p.
28. Google OR-Tools. Official Documentation [Електронний ресурс]. – Режим доступу: https://developers.google.com/optimization (дата звернення: 14.11.2025). – Назва з екрана.
29. Brown S. The C4 model for visualising software architecture [Електронний ресурс]. – Режим доступу: https://c4model.com/ (дата звернення: 14.11.2025). – Назва з екрана.
30. Heizer J., Render B., Munson C. Operations Management: Sustainability and Supply Chain Management. 14th ed. New York : Pearson, 2023. 896 p.
31. Nikolopoulos K. Forecasting for intermittent demand: A review of the literature and future research directions // International Journal of Forecasting. 2021. Vol. 37, No. 2. P. 875–892.
32. Taylor S. J., Letham B. Forecasting at scale // The American Statistician. 2018. Vol. 72, No. 1. P. 37–45.
33. US Enviromental Protection Agency. Estimates of generation and management of wasted food in the United States in 2019 https://www.epa.gov/system/files/documents/2024-04/2019-wasted-food-report_508_opt_ec_4.23correction.pdf

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Магістерська робота

«Методика підвищення ефективності управління каналами постачання продуктів в системі мережевого ресторанного бізнесу»

Виконав: студент групи ПДМ-62 Владислав ЦИВІНДА

Керівник: канд.фіз.-мат.наук., доцент кафедри ІПЗ Володимир САДОВЕНКО

Київ - 2025

МЕТА, ОБ’ЄКТА ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Мета роботи: підвищення ефективності управління каналами постачання продуктів системі мережевого ресторанного бізнесу шляхом впровадження автоматизованого планування та проактивного контролю.

Об’єкт дослідження: процеси управління каналами постачання продуктів в системі мережевого ресторанного бізнесу.

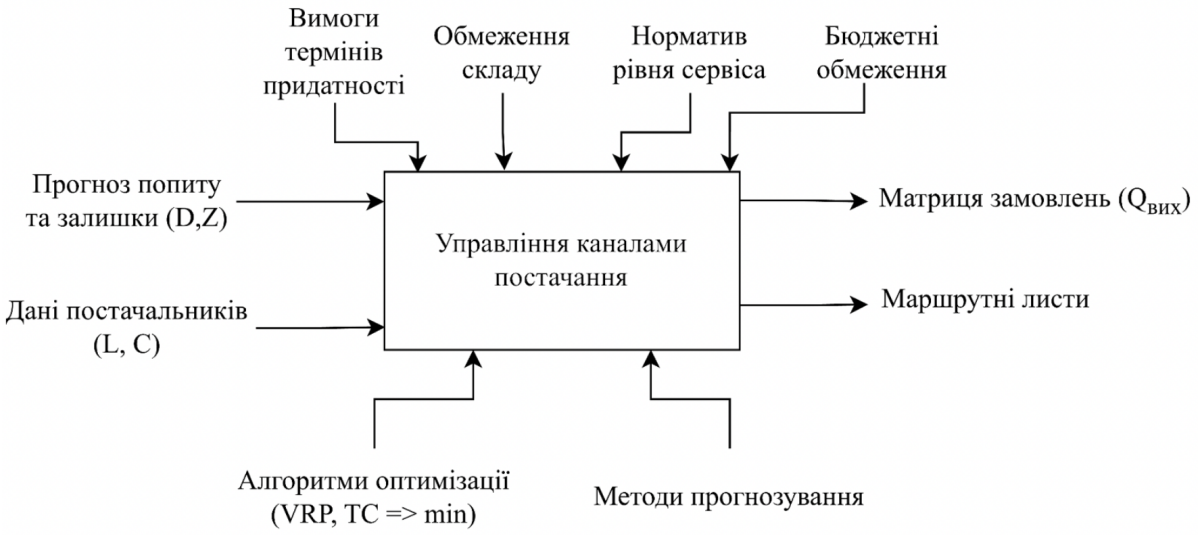
Предмет дослідження: практичні інструменти та підходи спрямовані на підвищення ефективності управління каналами постачання продуктів в системі мережевого ресторанного бізнесу.

АКТУАЛЬНІСТЬ РОБОТИ

Компонент / задача	Існуючий метод / алгоритм	Ключові особливості	Недоліки
Прогнозування попиту	Просте експоненційне згладжування, ковзне середнє	Швидкий розрахунок, мінімальні вимоги до ресурсів.	Не адаптується до нелінійних трендів такі як акції та сезонність. Низька точність висока похибка при короткому горизонті прогнозу.
Управління запасами	Економічний розмір замовлення, точка перезамовлення	Базується на історичних середніх показниках.	Не враховує логістичні вимоги, як-от мінімальна сума замовлення та кратність пакування, і не інтегрує вплив високих списань
Моніторинг KPI	Жорсткі порогові значення	Проста бінарна оцінка. Наприклад: норма/порушення.	Не пов'язує відхилення KPI з прямим впливом на бізнес-показники, генерує нерелевантні або пропускає критичні події.

3

ФОРМАЛІЗАЦІЯ ЗАДАЧІ УПРАВЛІННЯ КАНАЛАМИ ПОСТАЧАННЯ



Структурна модель задачі управління каналами постачання

АРХІТЕКТУРА ПРОГРАМНОГО КОМПЛЕКСУ

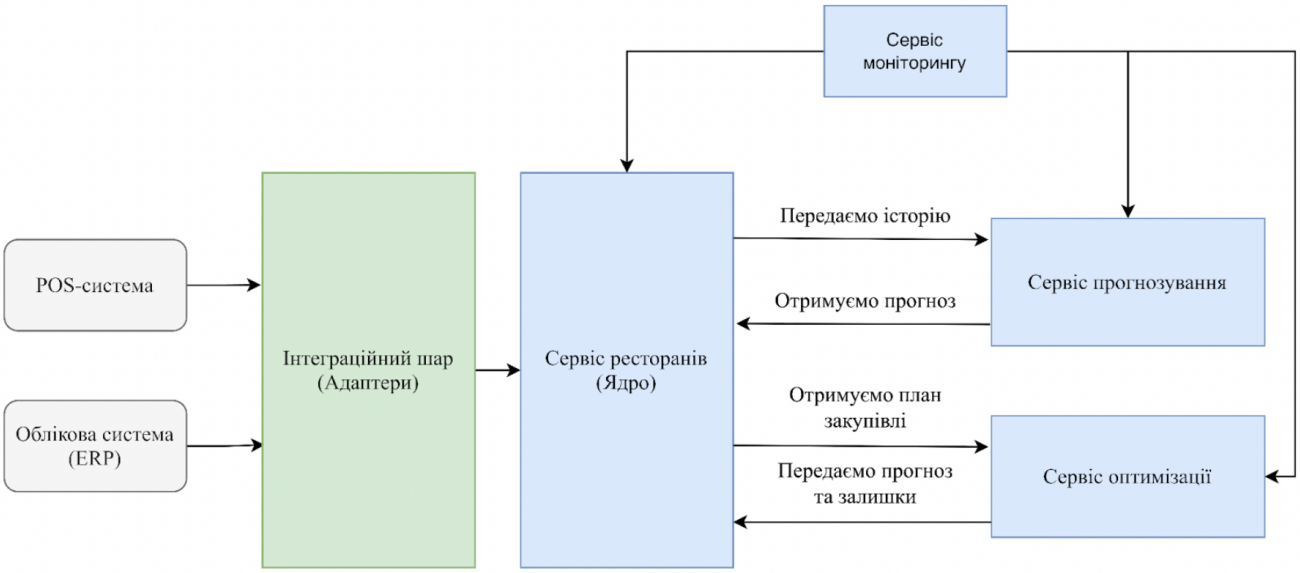


Схема взаємодії сервісів

МОДИФІКАЦІЯ МЕТОДУ УПРАВЛІННЯ

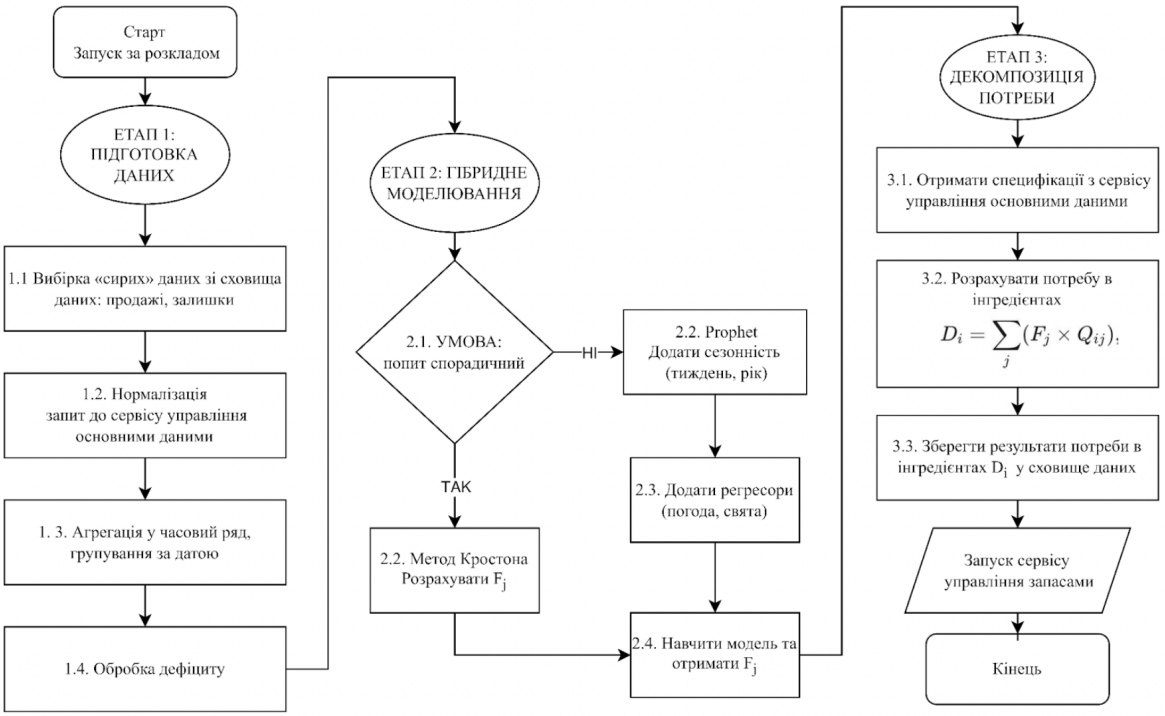
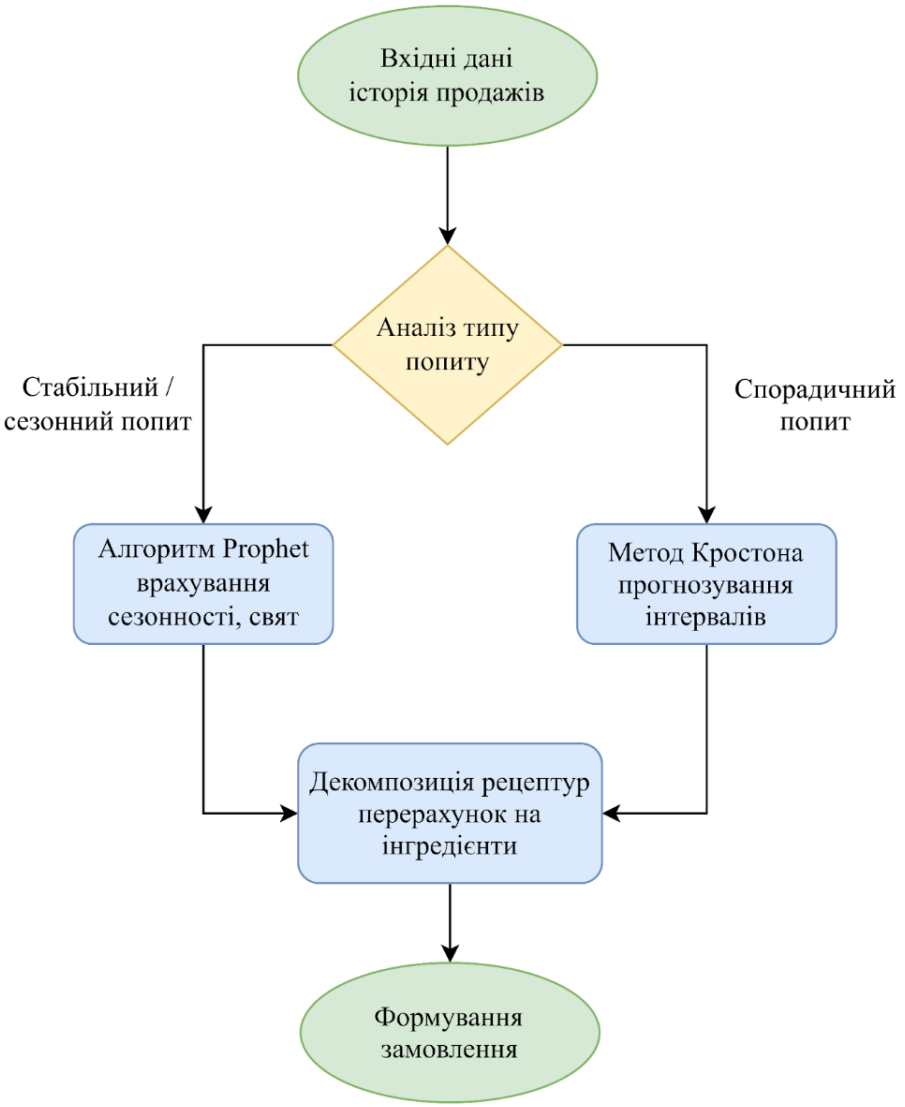
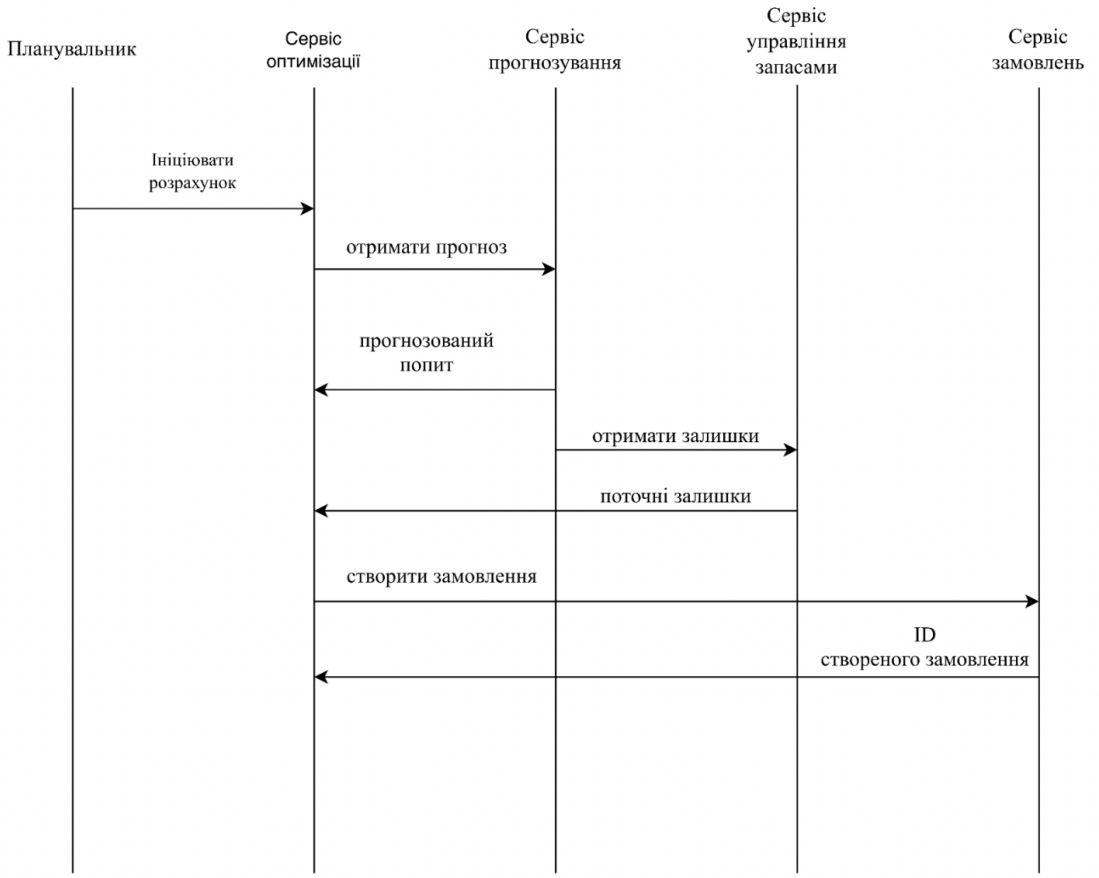


Схема модифікованого методу прогнозування та планування

БЛОК-СХЕМА АЛГОРИТМУ ПРОГНОЗУВАННЯ

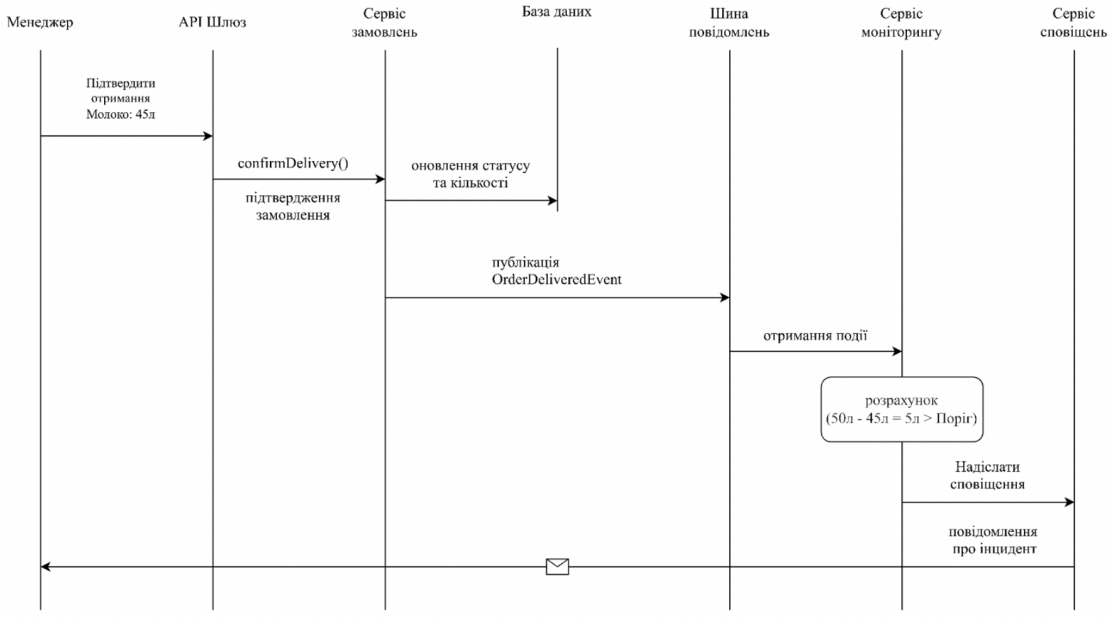


АВТОМАТИЗАЦІЯ ФОРМУВАННЯ ЗАМОВЛЕННЯ



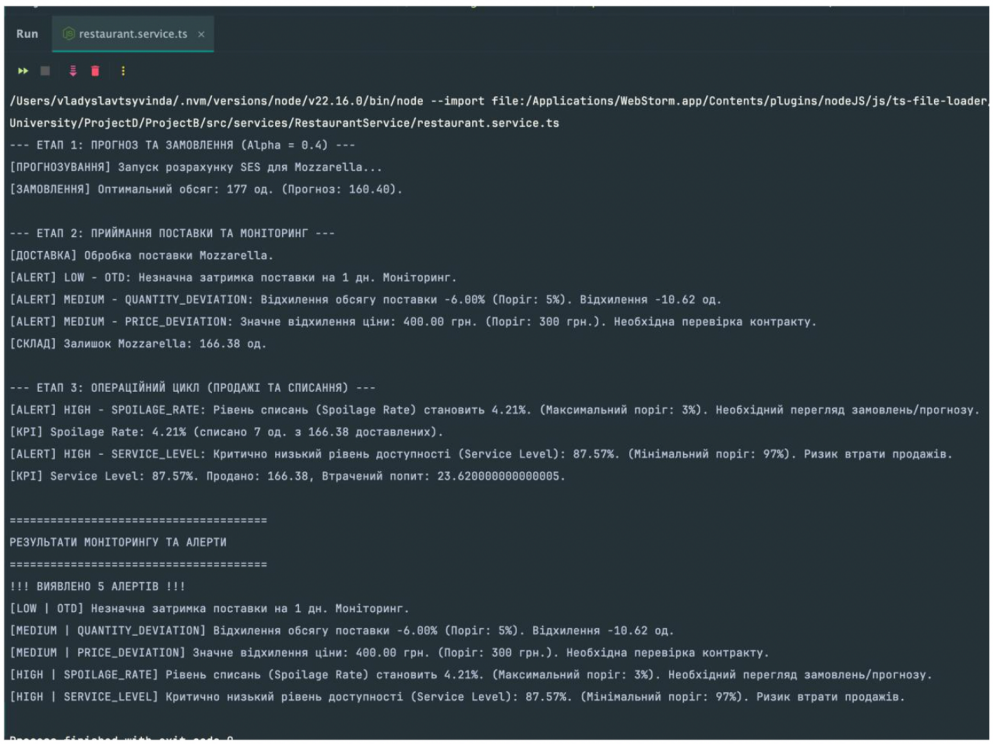
Діаграма послідовності для сценарію «Автоматичне формування замовлення»

МЕТОД ПРОАКТИВНОГО КОНТРОЛЮ



Алгоритм проактивного контролю логістичних операцій

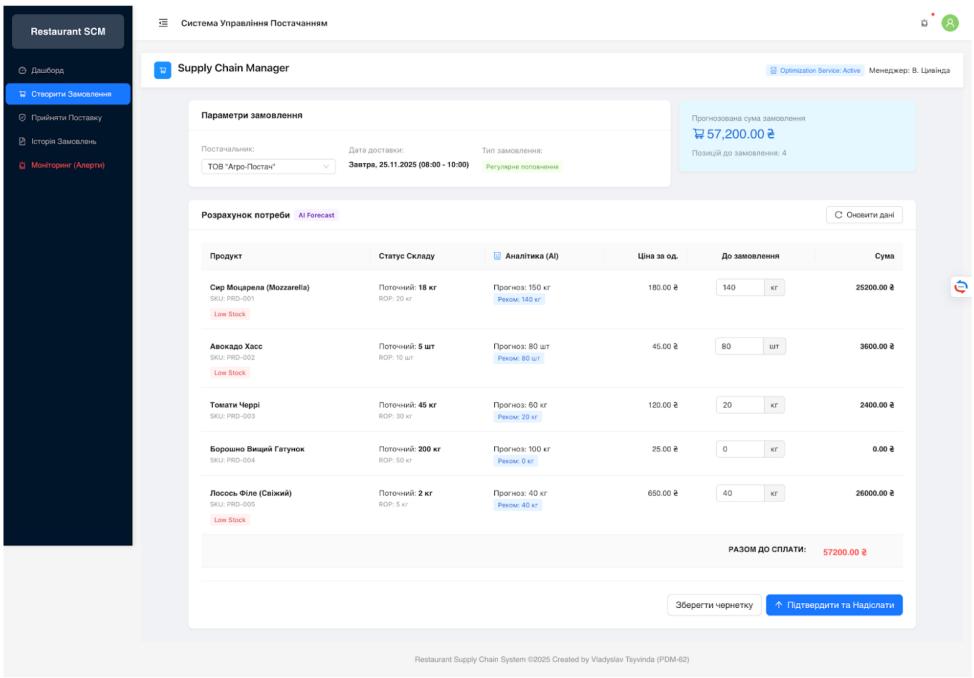
ПРАКТИЧНИЙ РЕЗУЛЬТАТ



Логіка перевірки параметрів: дати доставки, % недопоставки та фінансові розбіжності.

10

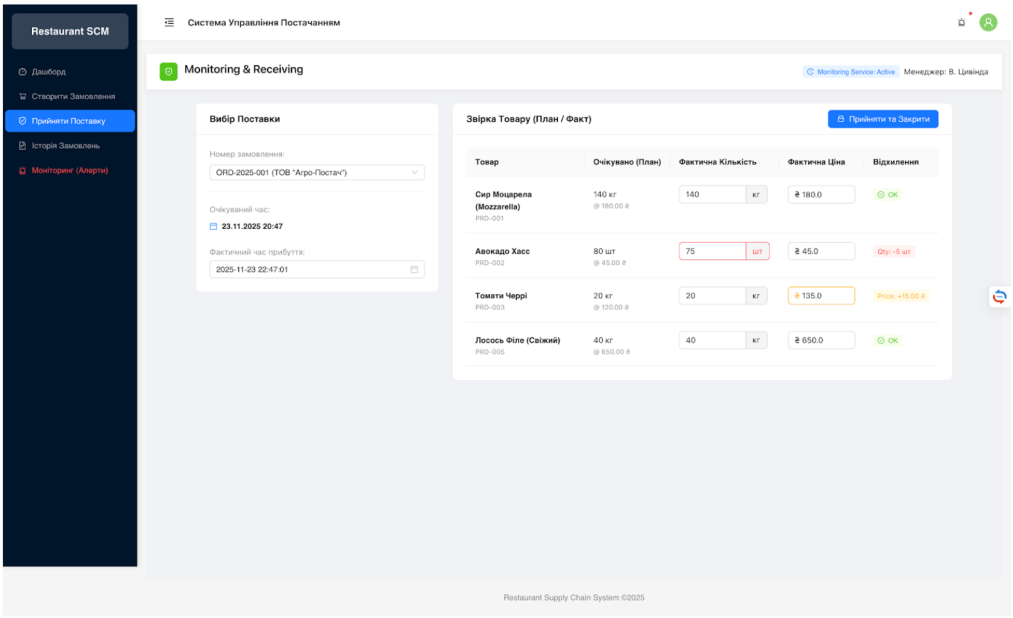
ПРАКТИЧНА РЕАЛІЗАЦІЯ: ІНТЕРФЕЙС КОРИСТУВАЧА



Процес замовлення товару

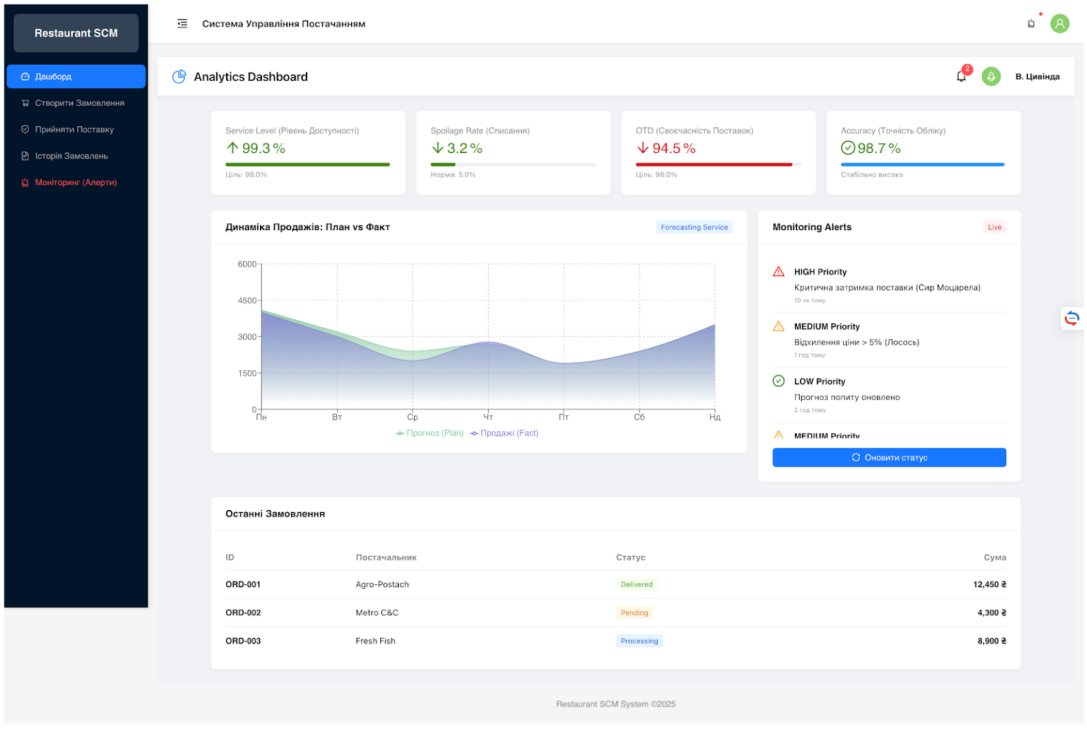
11

ПРАКТИЧНА РЕАЛІЗАЦІЯ: ІНТЕРФЕЙС КОРИСТУВАЧА



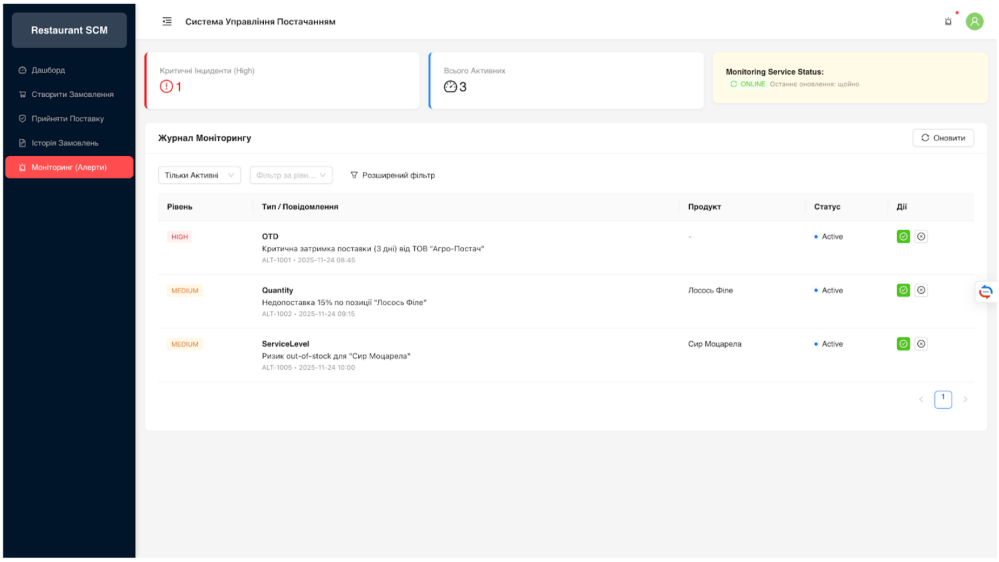
Модуль приймання замовлення

ПРАКТИЧНА РЕАЛІЗАЦІЯ: ІНТЕРФЕЙС КОРИСТУВАЧА



KPI показники

ПРАКТИЧНА РЕАЛІЗАЦІЯ: ІНТЕРФЕЙС КОРИСТУВАЧА



Журнал інцидентів

РЕЗУЛЬТАТИ МОДЕЛЮВАННЯ

Показник (KPI)	До впровадження	Після впровадження	Результат
Точність інвентаризації	74.5%	**98.7%**	**+24.2%**
Рівень доступності товарів	88.2%	**99.3%**	**+11.1%**
Рівень списань	9.8%	**3.2%**	**-6.6%**
Час реакції на збій	~12 годин	**1240 ms**	**~ -99.9%**

ВИСНОВКИ

1. Аналіз сучасних систем управління логістикою показав, що їхній основний недолік - це реактивність. Рішення часто приймаються вже після виникнення проблеми, що призводить до втрат.
2. Для вирішення цієї проблеми розроблено адаптивну модель прогнозування. Вона враховує не лише історію продажів, а й структуру рецептур та динамічні фактори попиту.
3. Спроєктовано мікросервісну архітектуру, яка дозволяє інтегрувати нові аналітичні модулі без зміни існуючих облікових платформ, забезпечуючи гнучкість системи.
4. Розроблено алгоритм проактивного контролю, який автоматично виявляє відхилення такі як затримки та недопоставки і в момент їх виникнення, а не під час інвентаризації.
5. Проведене моделювання підтвердило ефективність методики: точність запасів зросла до 98.7%, а час реакції на інциденти скоротився з годин до 1.24 секунд, що значно підвищує надійність постачання.

16

ПУБЛІКАЦІЇ ТА АПРОБАЦІЯ РОБОТИ

Тези доповідей:

1. Цивінда В.С., Трінтіна Н.А. Сучасні технології та методи управління ланцюгами постачання продуктів у мережевих ресторанах, // "Застосування програмного забезпечення в ІКТ", 24 квітня 2025 р. Збірник тез: ДУІКТ. *С. 194-199*.

2. Цивінда В.С., Садовенко В.С. Використання методу ФІФО у розробці системи автоматизації ланцюгів постачання // Всеукраїнська науково-технічна конференція «Виклики та рішення в програмній інженерії», 26 листопада 2025 р. С. 1-3. **Подано до публікації**.

ДОДАТОК Б. ЛІСТИНГИ ПРОГРАМНИХ МОДУЛІВ

```
export class OptimizationService {
   public calculateOptimalOrder(
      productId: string,
      leadTimeDays: number,
      constraints: LogisticsConstraints,
   ): OptimizationResult {
      const forecast = this.requestForecast(productId,
leadTimeDays);
      const stock = this.requestStockLevels(productId);


      const targetStockLevel = stock.maxStockLevel;
      let netNeed = targetStockLevel -
stock.currentQuantity;


      let orderQuantity = 0;
      if (stock.currentQuantity <= stock.reorderPoint) {
         orderQuantity = netNeed;
      } else {
         orderQuantity = Math.max(0, netNeed);
      }
            if (orderQuantity > 0) {
         if (orderQuantity %
constraints.packagingMultiplier !== 0) {
            orderQuantity = Math.ceil(orderQuantity /
constraints.packagingMultiplier) *
constraints.packagingMultiplier;
         }
         const unitPrice = 1;
         const calculatedCost = orderQuantity * unitPrice;
         if (calculatedCost <
constraints.minOrderValueUAH) {
            const minQuantity =
Math.ceil(constraints.minOrderValueUAH / unitPrice);
            const finalMinQuantity =
Math.ceil(minQuantity /
constraints.packagingMultiplier) *
constraints.packagingMultiplier;
               if (finalMinQuantity > orderQuantity) {
                  orderQuantity = finalMinQuantity;
               }
            }
         }
         return {
            productId,
            orderQuantity,
         };
      }
}
export class ForecastingService {
   public calculateSimpleExponentialSmoothing(
      salesHistory: SalesData[],
      alpha: number,
      periodsToForecast: number = 1
   ): ForecastResult[] {
      const forecasts: number[] = [];
      let Ft = salesHistory[0].actualSales;


      for (let i = 1; i < salesHistory.length; i++) {
         const At = salesHistory[i - 1].actualSales;
         Ft = (alpha * At) + (1 - alpha) * Ft;
      }
       const results: ForecastResult[] = [];
      for (let i = 1; i <= periodsToForecast; i++) {
         results.push({
            period: `Future Period ${i}`,
            forecast: Ft
         });
      }
      return results;
   }
}
```

```
export class MonitoringService {
      public checkOrderDeliveryTime(result:
VarianceResult): void {
      const daysDelayed = result.variance;
      if (daysDelayed >= this.thresholds.otdDaysDelay) {
         this.generateAlert('HIGH', 'OTD', `КРИТИЧНА
ЗАТРИМКА: ${daysDelayed} дн.`);
      }
   }
   public checkQuantityDeviation(result:
VarianceResult): void {
      const absDeviation =
Math.abs(parseFloat(result.percentage));
      if (absDeviation >=
this.thresholds.quantityDeviationPercent) {
         this.generateAlert('MEDIUM',
'QUANTITY_DEVIATION', `Відхилення обсягу:
${result.percentage}`);
      }
   }
}
import React, { useState } from 'react';
import {
   Typography,
   Card,
   Table,
   Tag,
   Button,
   Space,
   Select,
   Statistic,
   Row,
   Col,
   message,
   Tooltip, Badge
} from 'antd';
import {
   CheckCircleOutlined,
   CloseCircleOutlined,
   ExclamationCircleOutlined,
   SyncOutlined,
   FilterOutlined,
   DashboardOutlined
} from '@ant-design/icons';

const { Title, Text } = Typography;
const { Option } = Select;

interface AlertRecord {
   key: string;
   id: string;
   timestamp: string;
   type: 'OTD' | 'Quantity' | 'Price' | 'Spoilage' |
'ServiceLevel';
   severity: 'HIGH' | 'MEDIUM' | 'LOW';
   message: string;
   product?: string;
   status: 'Active' | 'Resolved' | 'Ignored';
   assignedTo: string;
}

export default function AlertsPage() {
   const [alerts, setAlerts] =
useState<AlertRecord[]>(initialAlerts);
   const [filterSeverity, setFilterSeverity] =
useState<string | null>(null);
   const [filterStatus, setFilterStatus] =
useState<string>('Active');

   const filteredAlerts = alerts.filter(item => {
      const matchSeverity = filterSeverity ? item.severity
=== filterSeverity : true;
      const matchStatus = filterStatus === 'All' ? true :
item.status === filterStatus;
      return matchSeverity && matchStatus;
   });
```

```
  const activeHigh = alerts.filter(a => a.severity === 'HIGH' && a.status === 'Active').length;
  const activeTotal = alerts.filter(a => a.status === 'Active').length;


  const handleResolve = (key: string) => {
    const newAlerts = alerts.map(a => a.key === key ? { ...a, status: 'Resolved' as const } : a);
    setAlerts(newAlerts);
    message.success('Інцидент успішно закрито');
  };


  const handleIgnore = (key: string) => {
    const newAlerts = alerts.map(a => a.key === key ? { ...a, status: 'Ignored' as const } : a);
    setAlerts(newAlerts);
    message.info('Інцидент приховано');
  };


  const columns = [
    {
      title: 'Рівень',
      dataIndex: 'severity',
      key: 'severity',
      render: (severity: string) => {
        let color = 'blue';
        if (severity === 'HIGH') color = 'red';
        if (severity === 'MEDIUM') color = 'orange';
        return <Tag color={color}>{severity}</Tag>;
      }
    },
    {
      title: 'Тип / Повідомлення',
      key: 'message',
      render: (_: any, record: AlertRecord) => (
        <Space direction="vertical" size={0}>
          <Text strong>{record.type}</Text>
          <Text>{record.message}</Text>
          <Text type="secondary" style={{ fontSize: 12 }}>{record.id} • {record.timestamp}</Text>
        </Space>
      )
    },
    {
      title: 'Продукт',
      dataIndex: 'product',
      key: 'product',
      render: (text: string) => text || <Text type="secondary">-</Text>
    },
    {
      title: 'Статус',
      dataIndex: 'status',
      key: 'status',
      render: (status: string) => (
        <Badge status={status === 'Active' ? 'processing' : status === 'Resolved' ? 'success' : 'default'} text={status} />
      )
    },
    {
      title: 'Дії',
      key: 'actions',
      render: (_: any, record: AlertRecord) => (
        <Space>
          {record.status === 'Active' && (
            <>
              <Tooltip title="Вирішити проблему">
                <Button
                  type="primary"
                  size="small"
                  icon={<CheckCircleOutlined />}
```

```
                            onClick={() =>
handleResolve(record.key)}
                            style={{ background: '#52c41a',
borderColor: '#52c41a' }}
                          />
                        </Tooltip>
                        <Tooltip title="Ігнорувати">
                          <Button
                            size="small"
                            icon={<CloseCircleOutlined />}
                            onClick={() =>
handleIgnore(record.key)}
                          />
                        </Tooltip>
                      </>
                    )}
                  </Space>
                )
              }
            ];

            return (
              <>
                <div style={{ marginBottom: 24 }}>
                  <Row gutter={[16, 16]}>
                    <Col xs={24} md={8}>
                      <Card bordered={false} style={{
borderLeft: '4px solid #f5222d' }}>
                        <Statistic
                          title="Критичні Інциденти
(High)"
                          value={activeHigh}
                          valueStyle={{ color: '#f5222d' }}

prefix={<ExclamationCircleOutlined />}
                        />
                      </Card>
                    </Col>
                    <Col xs={24} md={8}>
                      <Card bordered={false} style={{
borderLeft: '4px solid #1890ff' }}>
                        <Statistic
                          title="Всього Активних"
                          value={activeTotal}
                          prefix={<DashboardOutlined />}
                        />
                      </Card>
                    </Col>
                    <Col xs={24} md={8}>
                      <Card bordered={false} style={{
background: '#fffbe6', borderColor: '#ffe58f' }}>
                        <Text strong>Monitoring Service
Status:</Text>
                        <br />
                        <Tag color="success"
icon={<SyncOutlined spin />}>ONLINE</Tag>
                        <Text type="secondary" style={{
fontSize: 12 }}>Останнє оновлення: щойно</Text>
                      </Card>
                    </Col>
                  </Row>
                </div>

                <Card title="Журнал Моніторингу"
extra={<Button icon={<SyncOutlined
/>}>Оновити</Button>}>

                  <div style={{ marginBottom: 16, display:
'flex', gap: 16, flexWrap: 'wrap' }}>
                    <Select
                      defaultValue="Active"
                      style={{ width: 150 }}
                      onChange={setFilterStatus}
                    >
                      <Option value="Active">Тільки
Активні</Option>
                      <Option
value="Resolved">Вирішені</Option>
```

```
            <Option value="All">Bci</Option>
          </Select>

          <Select
            placeholder="Фільтр за рівнем"
            style={{ width: 150 }}
            allowClear
            onChange={setFilterSeverity}
          >
            <Option value="HIGH">High Priority</Option>
            <Option value="MEDIUM">Medium Priority</Option>
            <Option value="LOW">Low Priority</Option>
          </Select>

          <Button type="text" icon={<FilterOutlined />}>Розширений фільтр</Button>
        </div>

        <Table
          // @ts-ignore
          columns={columns}
          dataSource={filteredAlerts}
          pagination={{ pageSize: 8 }}
          rowClassName={(record) => record.severity === 'HIGH' && record.status === 'Active' ? 'bg-red-50' : ''}
        />
      </Card>
    </>
  );
}
```