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

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Методика підвищення ефективності управління каналами замовлень у системах обліку малого та середнього 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

_____ Дмитро ЖИЛІНСЬКИЙ
(підпис)

Виконав: здобувач вищої освіти групи ПДМ-62
Дмитро ЖИЛІНСЬКИЙ

Керівник: Вікторія КОРЕЦЬКА
канд. пед. 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

Жилінському Дмитру Іллічу

1. Тема кваліфікаційної роботи: «Методика підвищення ефективності управління каналами замовлень у системах обліку малого та середнього бізнесу»

керівник кваліфікаційної роботи Вікторія КОРЕЦЬКА, канд. пед. наук, доц.

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, параметри канало-орієнтованого ціноутворення, параметри статистики та аналітики каналів замовлень, вимоги до сортування та фільтрації інформації за каналами.

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

1. Дослідження методів управління каналами замовлень у CRM та POS системах
2. Розробка і моделювання методів управління каналами замовлень

3. Аналіз результату розробки, оцінка параметрів підвищення ефективності

5. Перелік ілюстративного матеріалу: *презентація*

1. Порівняння методів та алгоритмів управління каналами замовлень
2. Формальне представлення алгоритмів управління каналами та поканальним ціноутворенням
3. Порівняльний аналіз покращень після впровадження на основі даних реального використання
4. Відео демонстрація роботи розробленого модуля

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	31.10–04.11.2025	
2	Вивчення бізнес-процесів багатоканального продажу малого та середнього бізнесу	05.11–07.11.2025	
3	Дослідження існуючих рішень з класифікації й обробки каналів замовлень	08.11–11.11.2025	
4	Формування вимог до розроблюваного модуля каналів замовлень	12.11–15.11.2025	
5	Створення алгоритмів інтегрованого управління замовленнями та поканального ціноутворення	16.11–28.11.2025	
6	Програмна реалізація методики для підвищення ефективності роботи з каналами замовлень	29.11–10.12.2025	
7	Оформлення роботи: вступ, висновки, реферат	11.12–16.12.2025	
8	Розробка демонстраційних матеріалів	17.12–19.12.2025	

Здобувач вищої освіти _____ Дмитро ЖИЛІНСЬКИЙ
(підпис)

Керівник
кваліфікаційної роботи _____ Вікторія КОРЕЦЬКА
(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня магістра: 89 стор., 7 табл., 12 рис., 27 джерел.

Мета роботи – підвищення ефективності управління каналами замовлень у системі обліку малого та середнього бізнесу за рахунок застосування принципів структурованої інтеграції джерел замовлень та моделювання гнучкого ціноутворення.

Об'єкт дослідження – процес управління каналами замовлень у системах обліку малого та середнього бізнесу.

Предмет дослідження – методи, засоби та програмні технології підвищення ефективності управління каналами замовлень у системах обліку малого та середнього бізнесу.

У роботі використано методи системного аналізу, моделювання бізнес-процесів та алгоритмічного проєктування для дослідження управління каналами замовлень у системах обліку малого та середнього бізнесу. Проведено аналіз сучасних підходів до сегментації, маршрутизації та поканального ціноутворення в CRM-системах, на основі чого виявлено їхні обмеження. Розроблено інтегровану методику, що базується на уніфікованій математичній моделі замовлення, функції віднесення до каналу та алгоритмах гнучкого поканального ціноутворення, реалізованих у програмі SkyService. Експериментальна перевірка підтвердила підвищення точності обліку замовлень і ефективності аналітики за каналами.

КЛЮЧОВІ СЛОВА: СИСТЕМА ОБЛІКУ, МАЛИЙ ТА СЕРЕДНІЙ БІЗНЕС, КАНАЛ ЗАМОВЛЕНЬ, ОМНІКАНАЛЬНІСТЬ, УНІФІКОВАНЕ ЗАМОВЛЕННЯ, ПОКАЗНИКИ ЕФЕКТИВНОСТІ, АНАЛІЗ ДАНИХ, ЕЛЕКТРОННА КОМЕРЦІЯ.

ABSTRACT

Text part of the master's qualification work: 89 pages, 12 pictures, 7 table, 27 sources.

The purpose of the work – increasing the efficiency of order channel management in the accounting system of small and medium-sized businesses by applying the principles of structured integration of order sources and flexible pricing modeling.

Object of research – order channel management process in small and medium business accounting systems.

Subject of research – methods, tools and software technologies for increasing the efficiency of order channel management in small and medium-sized business accounting systems.

Summary of the work: The work uses methods of system analysis, business process modeling and algorithmic design to study order channel management in accounting systems of small and medium-sized businesses. An analysis of modern approaches to segmentation, routing and channel pricing in CRM systems was conducted, on the basis of which their limitations were identified. An integrated methodology was developed based on a unified mathematical model of the order, the channel assignment function and flexible channel pricing algorithms implemented in the SkyService program. Experimental verification confirmed the increase in the accuracy of order accounting and the effectiveness of channel analytics.

KEYWORDS: ACCOUNTING SYSTEM, SMALL AND MEDIUM BUSINESS, ORDERING CHANNEL, OMNICHANNEL, UNIFIED ORDER FORM, EFFICIENCY INDICATORS, DATA ANALYSIS, E-COMMERCE

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

МСБ — малий та середній бізнес

CRM — система управління взаємовідносинами з клієнтами (Customer Relationship Management)

POS — система обліку та продажу в точці продажу (Point of Sale)

HoReCa — сфера готельно-ресторанного бізнесу (Hotel, Restaurant, Café/Catering)

AP - Середня точність (Average Precision)

BN – Нормалізація пакету (Batch Normalization)

API — програмний інтерфейс взаємодії застосунків (Application Programming Interface)

UI — користувацький інтерфейс (User Interface)

UX — користувацький досвід взаємодії з системою (User Experience)

ERP — система планування ресурсів підприємства (Enterprise Resource Planning)

OMS — система управління замовленнями (Order Management System)

SKU — одиниця складського обліку товару (Stock Keeping Unit)

JSON — формат обміну структурованими даними (JavaScript Object Notation)

KPI — ключовий показник ефективності (Key Performance Indicator)

SaaS — програмне забезпечення як сервіс (Software as a Service)

ВСТУП

З розвитком цифрових технологій та зростанням ролі автоматизованих систем управління бізнесом особливої актуальності набуває питання ефективної обробки замовлень у малому та середньому бізнесі. Сучасні підприємства все частіше використовують багатоканальні моделі продажу, що включають фізичні точки продажу, онлайн-магазини, маркетплейси, соціальні мережі та зовнішні сервіси доставки. За таких умов зростає складність інтеграції різнорідних джерел замовлень, їх обліку, аналізу та застосування поканального ціноутворення.

Незважаючи на наявність широкого спектра CRM-, POS- та ERP-систем, значна частина існуючих рішень орієнтована на фіксовані сценарії обробки замовлень і не забезпечує достатньої гнучкості в управлінні каналами, правилами ціноутворення та методами оплати. Це призводить до фрагментації даних, ускладнення аналітики та зниження ефективності управлінських рішень, особливо в умовах динамічного розвитку бізнесу та масштабування.

Дана магістерська робота спрямована на аналіз та дослідження підходів до уніфікації обліку замовлень із різних каналів, а також розробці формальної моделі управління каналами замовлень із підтримкою гнучкого поканального ціноутворення. У межах роботи проведено аналіз сучасних систем автоматизації бізнесу, виявлено їх сильні сторони і недоліки та запропоновано власне рішення, що ґрунтується на принципах структурованої інтеграції даних та формалізації бізнес-правил.

Запропонований підхід дозволяє підвищити прозорість обліку, спростити масштабування системи та покращити засоби для аналізу інформації, важливої для управлінських рішень. Реалізація розробленого модуля підтверджує доцільність використання запропонованої моделі в системах обліку малого та середнього бізнесу.

Таким чином, розробка модуля для підвищення ефективності керування каналами замовлень шляхом застосування формальних моделей та сучасних інформаційних технологій є актуальною та має важливу практичну значущість.

Мета роботи – підвищення ефективності управління каналами замовлень у системі обліку малого та середнього бізнесу за рахунок застосування принципів структурованої інтеграції джерел замовлень та моделювання гнучкого ціноутворення.

Об’єкт дослідження – процес управління каналами замовлень у системах обліку малого та середнього бізнесу.

Предмет дослідження – методи, засоби та програмні технології підвищення ефективності управління каналами замовлень у системах обліку малого та середнього бізнесу.

ого ціноутворення.

Для досягнення поставленої мети в роботі вирішувалися наступні завдання:

1. Проведення аналізу предметної області та сучасних систем автоматизації бізнесу з фокусом на управління каналами замовлень.

2. Дослідження підходів до сегментації, маршрутизації та поканального ціноутворення в існуючих рішеннях.

3. Розробка формальної математичної моделі уніфікованого замовлення та функції віднесення замовлення до каналу.

4. Проєктування структури даних та алгоритмів інтегрованого управління замовленнями.

5. Реалізація та тестування програмного модуля роботи з каналами замовлень.

6. Аналіз результатів впровадження та формування порівняльних статистичних показників ефективності.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Багатоканальні бізнес-процеси у малому та середньому бізнесі

Багатоканальні бізнес-процеси у малому та середньому бізнесі формуються в умовах зростаючої цифровізації та зміни споживчої поведінки, коли клієнти очікують можливості взаємодії з компанією через різні точки контакту. До таких каналів належать фізичні торгові точки, інтернет-магазини, маркетплейси, соціальні мережі, мобільні додатки, а також зовнішні сервіси доставки та агрегатори замовлень. Для підприємств МСБ багатоканальність стає не лише інструментом розширення ринку збуту, але й необхідною умовою конкурентоспроможності. Водночас використання кількох каналів продажу ускладнює внутрішні бізнес-процеси, оскільки кожен канал має власні правила формування замовлень, ціноутворення, оплати та взаємодії з клієнтом.

Організація продажів у багатоканальному середовищі потребує узгодженого управління потоками замовлень, фінансових даних та аналітичної інформації. Неконсистентність у підходах до обліку замовлень за каналами може призводити до викривлення фінансових показників, ускладнення контролю ефективності окремих каналів та помилок у прийнятті управлінських рішень. Саме тому питання формалізації каналів замовлень, їх класифікації та контролю впливу на вагомі облікові дані бізнесу набуває особливої актуальності. Узагальнення цих аспектів створює методологічну основу для подальшого детального розгляду поняття та типів каналів замовлень, особливостей організації багатоканальних продажів, а також оцінки їх впливу на фінансові й аналітичні показники малого та середнього бізнесу.

1.1.1. Поняття та класифікація каналів замовлень

У сучасних системах обліку та автоматизації бізнесу канал замовлення доцільно розглядати як формалізоване джерело надходження замовлення, що визначається способом ініціації, передачі та обробки інформації про покупку в обліковій системі. Канал замовлення поєднує в собі не лише технічний механізм

отримання даних (API, веб-форма, POS-термінал тощо), але й бізнес-контекст взаємодії з клієнтом, включаючи правила ціноутворення, способи оплати, комісійні обмеження, часові характеристики та аналітичну інтерпретацію результатів продажу. Для підприємств малого та середнього бізнесу коректне визначення каналів замовлень є критично важливим, оскільки саме через них формується основний потік доходів, а також збираються дані для управлінської та фінансової аналітики.

У межах систем класу POS та CRM канал замовлення виступає логічною сутністю, яка дозволяє відокремити замовлення за походженням без дублювання бізнес-логіки. Це особливо актуально для українського ринку, де бізнес одночасно працює з офлайн-продажами, власними онлайн-каналами та зовнішніми агрегаторами доставки. Відсутність чіткої класифікації каналів призводить до того, що замовлення різного типу обробляються за однаковими правилами, що унеможливлює гнучке ціноутворення, коректний облік комісій та повноцінний аналіз ефективності кожного джерела продажу.

З урахуванням специфіки POS системи та практики використання на українському ринку, канали замовлень доцільно класифікувати за декількома взаємодоповнювальними ознаками.

– Офлайн-канали замовлень

До офлайн-каналів належать замовлення, які формуються безпосередньо у фізичній точці продажу з використанням POS-терміналу або іншого локального інтерфейсу. Це базовий та історично первинний тип каналів для більшості представників МСБ, особливо у сферах роздрібної торгівлі та HoReCa. У межах SkyService POS офлайн-канал зазвичай пов’язаний із конкретною торговою точкою, касою або співробітником, що дозволяє точно фіксувати місце та умови здійснення продажу.

Ключовою особливістю офлайн-каналів є мінімальна кількість проміжних ланок між клієнтом і бізнесом, що забезпечує просту модель ціноутворення та швидке обслуговування. Водночас навіть у межах офлайн-продажів можуть існувати підканали, наприклад замовлення в залі, замовлення на виніс або

замовлення через термінал самообслуговування, які потребують окремого обліку з точки зору аналітики та фінансових показників.

– Онлайн-канали прямого продажу

Онлайн-канали прямого продажу охоплюють замовлення, що надходять через власні цифрові ресурси бізнесу: вебсайт, інтернет-магазин, мобільний застосунок або форми замовлення в соціальних мережах. Для українського ринку характерним є активне використання соціальних платформ як каналу продажу, зокрема Instagram, Facebook та месенджерів, де замовлення можуть надходити як у напівавтоматичному, так і в ручному режимі.

У системі SkyService такі канали можуть інтегруватися через API або обробляються шляхом ручного введення замовлення з обов’язковим зазначенням джерела. Основною відмінністю цього класу каналів є необхідність нормалізації структури замовлення, через те, що дані можуть поступати у різних форматах. Крім того, онлайн-канали часто потребують окремих правил ціноутворення, знижок або акцій, що обумовлює потребу в канало-орієнтованих моделях управління цінами.

– Канали маркетплейсів та агрегаторів доставки

Окремий та надзвичайно важливий клас каналів для українського МСБ — це зовнішні платформи та агрегатори, такі як служби доставки їжі, маркетплейси або сервіси замовлень. У сфері HoReCa такі канали часто забезпечують значну частку обороту, але водночас супроводжуються високими комісіями та жорстко регламентованими правилами взаємодії.

З алгоритмічної точки зору, цей клас каналів характеризується необхідністю врахування додаткових параметрів: комісій платформи, обмежень на зміну цін, часових затримок, а також специфічних форматів даних. У SkyService POS такі канали мають бути представлені як окремі логічні сутності, для яких застосовуються власні правила ціноутворення та статистичного обліку. Відсутність виділення агрегаторів у самостійний клас каналів призводить до спотворення фінансових показників і некоректної оцінки реальної прибутковості бізнесу.

– Ручні та службові канали замовлень

До цієї категорії належать замовлення, що створюються вручну персоналом з технічних або організаційних причин: телефонні замовлення, замовлення через месенджери без прямої інтеграції, коригувальні або тестові замовлення. Для МСБ такі канали залишаються актуальними, оскільки не всі процеси можуть бути повністю автоматизовані.

У SkyService POS ручні канали виконують важливу роль у забезпеченні повноти обліку та дозволяють зберігати єдину модель даних незалежно від способу надходження замовлення. Водночас вони потребують чіткого відокремлення від основних каналів продажу, щоб уникнути спотворення статистики та фінансових звітів.

– Класифікація за рівнем інтеграції

З погляду архітектури системи управління, канали замовлень також доцільно класифікувати за рівнем технічної інтеграції: інтегровані (API-канали), напівінтегровані та неінтегровані. Для SkyService POS така класифікація є основою для побудови уніфікованої моделі замовлення, яка дозволяє незалежно від каналу застосовувати однакові алгоритми обробки, ціноутворення та аналітики.

1.1.2. Особливості організації продажів у багатоканальному середовищі

Організація продажів у багатоканальному середовищі є однією з ключових характеристик сучасного малого та середнього бізнесу, який функціонує в обставинах високої конкуренції цифрового ринку та змін споживчої поведінки. На відміну від одноканальних моделей, де всі замовлення надходять з одного джерела і обробляються за уніфікованими правилами, багатоканальна модель передбачає паралельне використання декількох каналів продажу, кожен з яких має власні технічні, організаційні та економічні особливості. Для підприємств українського ринку, особливо у сферах HoReCa, роздрібної торгівлі та сервісних

послуг, така модель стала фактично стандартом, оскільки дозволяє захопити більшу аудиторію та знизити залежність від одного джерела доходу.

Однією з базових особливостей багатоканального продажу є неоднорідність джерел замовлень. Замовлення можуть надходити одночасно з офлайн-точки, через сайт, мобільний застосунок, соціальні мережі, телефонні дзвінки або зовнішні сервіси доставки. Кожен з цих каналів формує замовлення у власному форматі, з різним рівнем деталізації та різними вимогами до обробки. У практиці використання SkyService POS це означає необхідність застосування механізмів нормалізації замовлень, які дозволяють привести всі вхідні дані до єдиної внутрішньої структури без втрати бізнес-контексту каналу.

Важливою особливістю є різні правила ціноутворення та фінансових розрахунків у межах різних каналів. На українському ринку поширеною практикою є встановлення окремих цін для зовнішніх сервісів доставки або онлайн-каналів з урахуванням комісій, логістичних витрат та маркетингових умов. У багатоканальному середовищі ціна товару або послуги перестає бути фіксованою величиною та стає функцією каналу замовлення. Для SkyService POS це зумовлює необхідність підтримки канало-орієнтованих моделей ціноутворення, які дозволяють задавати як явні ціни для окремих каналів, так і правила їх модифікації у вигляді націнок або знижок.

Ще однією суттєвою особливістю є різноманіття сценаріїв оплати та виконання замовлень. У межах одного бізнесу можуть одночасно використовуватися готівкові та безготівкові платежі, онлайн-оплата, післяплата, оплата через зовнішні платіжні сервіси або платформи-агрегатори. Кожен канал замовлень накладає власні обмеження на доступні способи оплати та порядок фінансового обліку. У системі SkyService POS це потребує гнучкої прив’язки методів оплати до каналу замовлення, що дозволяє коректно формувати фінансові звіти та уникати помилок у бухгалтерському обліку.

Організація продажів у багатоканальному середовищі також ускладнюється операційною асинхронністю каналів. Замовлення з різних джерел можуть надходити з різною швидкістю, мати затримки у передачі даних або

вимагати підтвердження з боку персоналу. Наприклад, замовлення через агрегатор доставки може бути створене в системі з часовим зсувом, тоді як офлайн-продаж фіксується миттєво. Для українського МСБ, де часто обмежені кадрові ресурси, така асинхронність створює додаткове навантаження на персонал. SkyService POS у цьому контексті виконує роль єдиного центру управління, який дозволяє синхронізувати обробку замовлень незалежно від каналу їх надходження.

Окремої уваги потребує аналітичний аспект багатоканального продажу. У традиційних системах обліку замовлення часто аналізуються сукупно, без урахування каналу їх походження. Такий підхід не дозволяє оцінити реальну ефективність окремих каналів, їхню прибутковість та вплив на загальні фінансові показники бізнесу. В умовах українського ринку, де комісії агрегаторів можуть суттєво відрізнятися, відсутність канальної аналітики призводить до хибних управлінських рішень. Особливістю SkyService POS є використання каналів замовлень як аналітичного виміру, що дозволяє фільтрувати та агрегувати статистику за джерелами замовлень без необхідності створення окремих підсистем.

Важливою організаційною особливістю є також людський фактор та рівень цифрової зрілості бізнесу. Багато підприємств МСБ в Україні перебувають на етапі часткової цифровізації, де автоматизовані та ручні процеси співіснують одночасно. Це зумовлює потребу в таких системах, як SkyService POS, які підтримують як повністю інтегровані канали, так і ручне введення замовлень із чітким зазначенням їх джерела. Такий підхід дозволяє поступово впроваджувати багатоканальну модель без радикальної перебудови бізнес-процесів.

Таким чином, організація продажів у багатоканальному середовищі характеризується складністю структури замовлень, варіативністю фінансових та операційних правил, необхідністю інтеграції різнорідних джерел даних і підвищеними вимогами до аналітики. Специфіка SkyService POS та українського ринку МСБ полягає у потребі поєднання гнучкості, простоти впровадження та формалізованого підходу до управління каналами замовлень, що створює

підґрунтя для розробки інтегрованих моделей і алгоритмів, описаних у наступних розділах роботи.

1.2. Аналіз сучасних систем автоматизації бізнесу

Сучасні системи автоматизації бізнесу відіграють ключову роль у забезпеченні ефективного управління операційною діяльністю МСБ, працюючих у із цифровим ринком збуту. CRM- та POS-рішення виступають базовими інструментами для обліку замовлень, управління продажами, фінансовими потоками та взаємодією з клієнтами. З огляду на поширення багатоканальних моделей продажу, до таких систем висуваються підвищені вимоги щодо підтримки різних джерел надходження замовлень, гнучкості інтеграції та можливостей аналітичної обробки даних.

У даному розділі здійснюється узагальнений аналіз сучасних систем автоматизації бізнесу з акцентом на підходи до обліку замовлень та управління каналами їх надходження. Розглядаються типові архітектурні рішення, методи інтеграції із зовнішніми сервісами та практики, що застосовуються в популярних CRM- і POS-системах для МСБ. Особлива увага приділяється виявленню обмежень існуючих підходів, які ускладнюють реалізацію повноцінного поканального управління, що створює передумови для обґрунтування актуальності розробки спеціалізованих моделей і алгоритмів, реалізованих у програмі SkyService.

1.2.1. Огляд CRM та POS-систем для малого та середнього бізнесу

Рішення типу CRM та POS є базовими інструментами автоматизації роботи малого та середнього бізнесу, автоматизуючи облік продажів, управління товарами, фінансові розрахунки та взаємодію з клієнтами. В умовах розвитку багатоканального продажу такі системи поступово трансформуються з простих касових рішень у комплексні платформи управління бізнес-процесами. Для українського ринку МСБ особливо актуальними є рішення, що поєднують

відносну простоту впровадження з підтримкою онлайн-каналів, інтеграцій із зовнішніми сервісами та можливостями аналітики.

Одним із популярних рішень у сфері HoReCa та роздрібної торгівлі є Poster. Дана система позиціонується як хмарний POS-сервіс з фокусом на швидке розгортання та інтуїтивно зрозумілий інтерфейс. Poster забезпечує облік продажів, управління меню та складом, а також базову аналітику. Підтримка кількох джерел замовлень реалізована через інтеграції з сервісами доставки та онлайн-замовлень, однак у більшості випадків канал розглядається як атрибут замовлення, а не як самостійна керована сутність. Це обмежує можливості гнучкого поканального ціноутворення та аналітики, оскільки основна логіка системи зосереджена навколо точки продажу, а не навколо каналу як елемента бізнес-моделі.

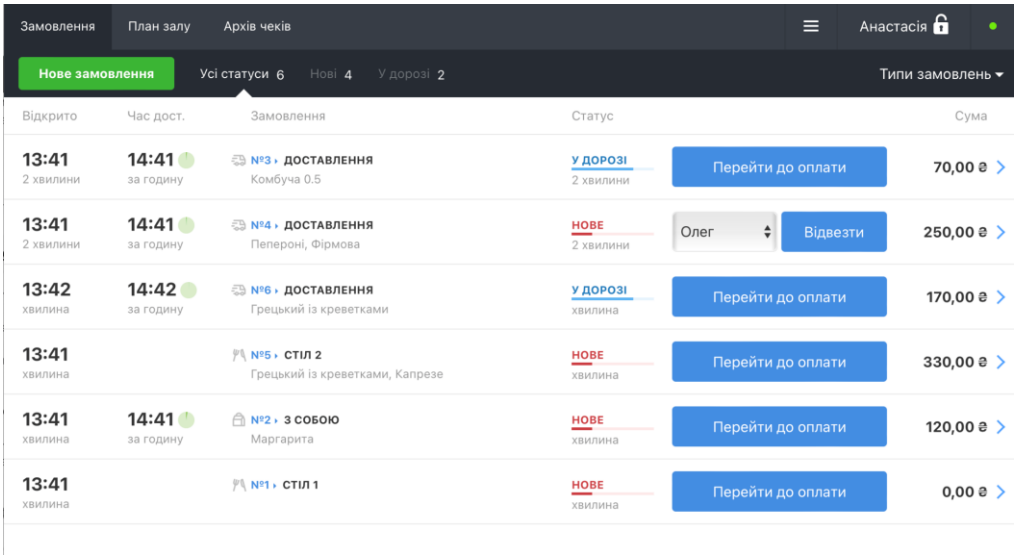


Рис 1.1 Інтерфейс керування каналами актуальних замовлень у Poster

Shopify POS є складовою екосистеми Shopify та орієнтований насамперед на бізнеси, що поєднують офлайн- та онлайн-продажі. Основною перевагою цього рішення є тісна інтеграція з інтернет-магазином, що дозволяє об’єднувати дані про замовлення з різних каналів у межах єдиної платформи. У Shopify POS канали продажу представлені на концептуальному рівні, зокрема через поділ на онлайн- та офлайн-продажі, проте детальне управління каналами як окремими об’єктами бізнес-логіки є обмеженим. Система орієнтована на стандартизовані

сценарії використання, що знижує гнучкість адаптації під специфічні умови локальних ринків, зокрема українського, де важливу роль відіграють зовнішні сервіси доставки та нестандартні моделі оплати.

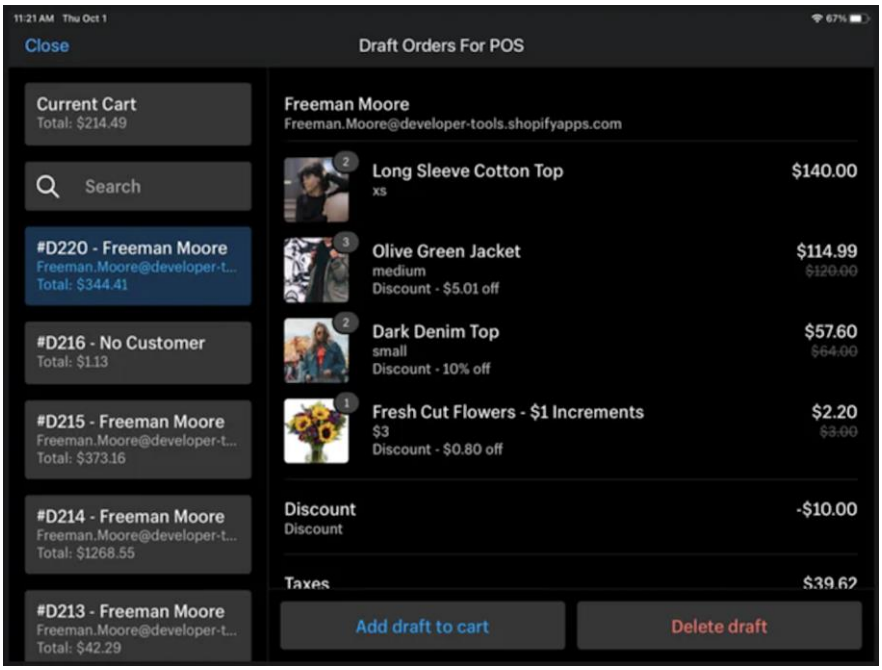


Рис 1.2 Інтерфейс керування замовленнями у Shopify POS

Ще одним відомим представником ринку є Square, який пропонує комплексне рішення для обліку продажів, платежів та базової CRM-функціональності. Square POS широко використовується малим бізнесом завдяки простоті налаштування та інтеграції з платіжною інфраструктурою. Підхід до обліку замовлень у Square базується на уніфікованій моделі транзакції, що дозволяє працювати з різними точками продажу. Водночас канали замовлень у класичному розумінні не виділяються як окремий аналітичний або керований елемент, а інтеграції з зовнішніми сервісами розглядаються як додаткові розширення функціоналу. Такий підхід спрощує використання системи, але обмежує можливості глибокого аналізу ефективності окремих джерел замовлень.

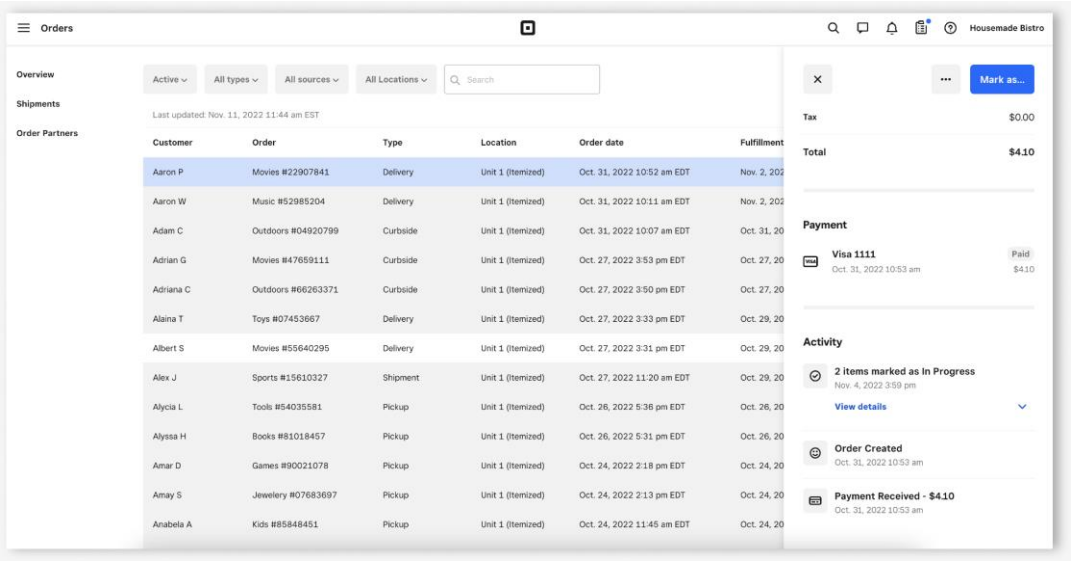


Рис 1.3 Інтерфейс перегляду статистики замовлень з каналами Square POS

На відміну від розглянутих рішень, SkyService POS орієнтований на потреби бізнесів, що працюють у складному багатоканальному середовищі, зокрема на українському ринку. Система поєднує класичний POS-функціонал з можливістю формалізованого управління каналами замовлень як окремими логічними сутностями. Такий підхід дозволяє пов’язувати з джерелом правила ціноутворення, методи оплати та аналітичні показники. SkyService POS розроблений з урахуванням реальних сценаріїв роботи МСБ, де офлайн-продажі, онлайн-замовлення та зовнішні сервіси доставки співіснують в межах єдиної системи обліку.

Ключовою перевагою SkyService POS у порівнянні з рішеннями-аналоги є його архітектурна орієнтація на розширюваність та адаптивність. Система дозволяє поступово впроваджувати багатоканальну модель без радикальної перебудови бізнес-процесів, зберігаючи єдину модель замовлення та централізований облік.

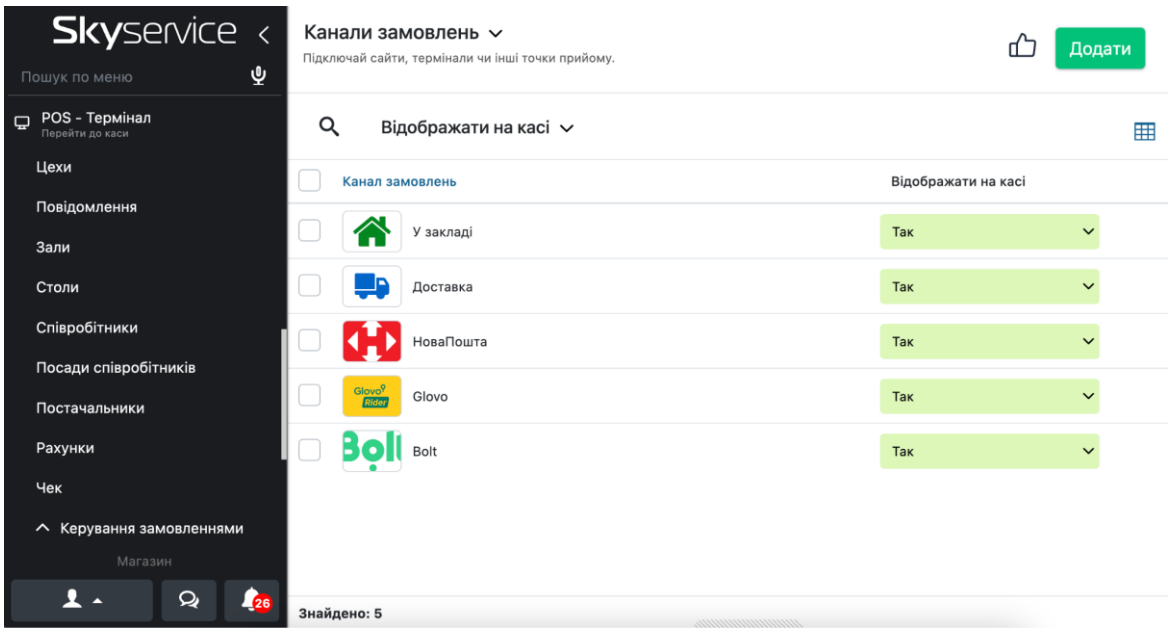


Рис 1.4 Інтерфейс керування замовленнями SkyService POS

1.2.2. Підходи до обліку замовлень та джерел їх надходження

Облік замовлень є центральним елементом будь-якої системи автоматизації бізнесу, оскільки саме на основі даних про замовлення формуються фінансові показники, управлінська аналітика та приймаються стратегічні рішення. В умовах багатоканального продажу складність цього процесу суттєво зростає, оскільки замовлення надходять із різних сервісів, мають відмінну структуру, тип і кількість атрибутів та правила обробки. Сучасні CRM- та POS-системи застосовують різні підходи до фіксації джерел надходження замовлень, які умовно можна поділити на кілька базових моделей.

Атрибутивний підхід є айпростішим і найбільш поширеним. За нього джерело замовлення фіксується у вигляді одного або кількох полів у структурі замовлення (наприклад, тип продажу, джерело, тег або категорія). Такий підхід використовується в багатьох базових POS-рішеннях і дозволяє швидко визначити, звідки надійшло замовлення. Однак у цьому випадку канал не має власної бізнес-логіки та не впливає на обробку замовлення, а використовується виключно для відображення або фільтрації даних. Для українського МСБ це часто означає неможливість коректно враховувати комісії агрегаторів або застосовувати окремі ціни для різних каналів без дублювання номенклатури.

Розділення замовлень за типами або сценаріями продажу є іншим поширеним підходом. Кожен канал реалізується як окремий сценарій створення замовлення (офлайн-продаж, онлайн-замовлення, доставка тощо). У таких системах бізнес-логіка частково прив’язується до сценарію, наприклад визначаються доступні способи оплати або статуси виконання. Хоча цей підхід дозволяє гнучкіше налаштовувати процеси, він часто призводить до фрагментації моделі замовлення, ускладнює підтримку та масштабування системи, особливо при додаванні нових каналів. Для бізнесів, що працюють із кількома сервісами доставки, це створює надлишкову складність і підвищує ризик помилок в обліку.

Інтеграційний підхід є більш розвиненим. Кожне зовнішнє джерело замовлень підключається до системи як окремий модуль або сервіс. Замовлення надходять через API та обробляються за правилами, визначеними для конкретної інтеграції. Такий підхід характерний для систем, орієнтованих на екосистеми, зокрема Shopify або Square. Його перевагою є автоматизація отримання даних і зменшення ручної роботи, проте канал у цьому випадку залишається тісно пов’язаним із технічною інтеграцією. Це ускладнює використання єдиної моделі обліку та не дозволяє легко об’єднувати замовлення з інтегрованих і неінтегрованих джерел в межах спільної аналітики.

Окрему групу становлять системи, що застосовують **модель уніфікованого замовлення**, у якій всі замовлення незалежно від джерела приводяться до єдиного внутрішнього формату. У таких системах джерело надходження замовлення розглядається як одна з ключових характеристик, але не визначає структуру даних. Саме цей підхід є найбільш перспективним для багатоканального середовища, оскільки дозволяє застосовувати спільні алгоритми обробки, ціноутворення та аналітики для всіх каналів. Водночас ефективність цього підходу значно залежить від ступеня формалізації представленого каналу замовлення в системі.

У контексті українського ринку МСБ особливо актуальним є поєднання уніфікованої моделі замовлення з формалізованим представленням каналу як

керованої сутності. У SkyService POS канал замовлення не обмежується лише ідентифікатором джерела, а виступає логічним об’єктом, з яким пов’язуються правила ціноутворення, методи оплати та параметри статистичного обліку. Замовлення незалежно від способу надходження – офлайн, через сайт, API або вручну – проходять етап нормалізації та прив’язуються до конкретного каналу за визначеними правилами.

Обраний підхід дозволяє уникнути дублювання бізнес-логіки, спростити підтримку системи та забезпечити цілісність даних. Джерело надходження замовлення перестає бути лише інформаційним атрибутом і стає активним елементом моделі управління. Це, у свою чергу, створює передумови для реалізації гнучкого поканального ціноутворення та коректної аналітики, що є ключовими вимогами для ефективного управління продажами у багатоканальному середовищі.

1.2.3. Архітектурні особливості інтеграції зовнішніх сервісів

Інтеграція зовнішніх сервісів є невід’ємною складовою сучасних CRM- та POS-систем, що працюють у багатоканальному середовищі. Для МСБ вона забезпечує можливість приймати замовлення з маркетплейсів, сервісів доставки, онлайн-платформ та платіжних систем без необхідності ручного перенесення даних. Водночас така інтеграція істотно ускладнює архітектуру інформаційної системи, оскільки зовнішні сервіси використовують різні типи даних, протоколи обміну, часові характеристики та обмеження бізнес-логіки. Тому підхід до інтеграції значною мірою визначає ефективність управління каналами замовлень у цілому.

Точкова (point-to-point) інтеграція є найбільш поширеною архітектурною моделлю. За неї кожен зовнішній сервіс підключається безпосередньо до ядра системи через окремий API-модуль. У такій моделі для кожного сервісу реалізується власна логіка обробки запитів, трансформації даних та обробки помилок. Хоча цей підхід є відносно простим на початковому етапі, він погано масштабується та ускладнює підтримку системи. Для українського МСБ, яке

часто поступово підключає нові сервіси доставки або платформи продажу, точкова інтеграція призводить до зростання технічного боргу та фрагментації бізнес-логіки.

Шарова архітектура інтеграції є більш структурованим підходом. Між зовнішніми сервісами та внутрішньою моделлю системи вводиться проміжний рівень адаптації. На даному рівні проходить перетворення вхідних даних у внутрішній формат замовлення, а також обробка специфічних параметрів сервісу, таких як комісії, часові зони або статуси доставки. Такий підхід дозволяє частково ізолювати зовнішні залежності, проте канал замовлення все ще часто залишається прив'язаним до конкретної інтеграції, що обмежує можливості уніфікованого управління.

Подієво-орієнтована архітектура використовується у системах, орієнтованих на екосистемний розвиток, де зовнішні сервіси взаємодіють з системою через події або черги повідомлень. Замовлення надходять у вигляді подій, які обробляються незалежно від джерела, що підвищує стійкість системи до збоїв і дозволяє масштабувати обробку навантаження. Однак така архітектура потребує складної інфраструктури та високого рівня технічної експертизи, що часто є надмірним для підприємств сфери МСБ.

Також особливу складність у контексті інтеграції становлять маркетплейси та сервіси доставки, які, окрім передачі замовлень, накладають власні бізнес-правила. Вони можуть обмежувати можливість зміни цін, вимагати використання певних статусів замовлення або передбачати асинхронну взаємодію. Для українського ринку це є особливо актуальним, оскільки сервіси доставки відіграють значну роль у сфері HoReCa, але при цьому мають різні вимоги до формату даних та процесу обробки замовлень, що викликає додаткові ускладнення при прямій інтеграції.

У цьому контексті важливою є архітектурна концепція відокремлення каналу замовлення від механізму інтеграції. У SkyService POS інтеграція з зовнішніми сервісами розглядається як технічний спосіб отримання даних, тоді як канал замовлення виступає окремою логічною сутністю системи. Замовлення,

отримане через API, після етапу нормалізації прив’язується до відповідного каналу за формалізованими правилами, незалежно від того, яким саме сервісом воно було передане. Це дозволяє об’єднувати замовлення з інтегрованих і неінтегрованих джерел у межах єдиної моделі управління. Такий архітектурний підхід забезпечує гнучкість та масштабованість системи, спрощує додавання нових інтеграцій і зменшує залежність бізнес-логіки від зовнішніх сервісів. Для українського МСБ це означає можливість адаптації до змін ринку без значних витрат на доопрацювання системи.

1.3. Існуючі методи управління каналами замовлень

1.3.1 Статичні та напівдинамічні моделі сегментації каналів

Сегментація каналів замовлень є базовим етапом управління багатоканальними продажами, оскільки саме вона визначає, як система ідентифікує джерело замовлення та які правила до нього застосовуються. У більшості сучасних CRM- та POS-систем сегментація каналів реалізується за допомогою статичних або напівдинамічних моделей, які відрізняються рівнем гнучкості, масштабованості та здатністю адаптуватися до змін бізнес-середовища. Аналіз цих моделей дозволяє виявити їхні обмеження та обґрунтувати доцільність використання більш формалізованого підходу, реалізованого в кваліфікаційній роботі.

Статична модель сегментації передбачає фіксований набір каналів замовлень, який визначається на етапі налаштування системи та рідко змінюється в процесі експлуатації. У такій моделі кожному замовленню присвоюється канал на основі жорстко заданого правила або прямої вказівки користувача. Канал у цьому випадку є константною категорією, що використовується переважно для візуального групування або базової фільтрації даних.

Прикладом статичної сегментації є ситуація, коли в POS-системі заздалегідь створені канали типу «Зал», «Доставка», «Самовивіз», і кожне замовлення під час створення вручну відноситься до одного з них. Аналогічний підхід застосовується в ряді систем, де джерело замовлення визначається значенням поля source або type, яке не впливає на подальшу обробку замовлення.

Основною перевагою статичних моделей є їх простота реалізації та використання, що робить їх привабливими для невеликих бізнесів на початковому етапі цифровізації. Проте для багатоканального середовища така модель має суттєві обмеження. Вона не враховує різноманіття сценаріїв надходження замовлень, не підтримує автоматичну прив’язку до каналу та не дозволяє гнучко змінювати правила ціноутворення або оплати без ручного втручання. Для українського ринку, де бізнеси часто одночасно працюють із кількома сервісами доставки, статична модель швидко втрачає актуальність.

Напівдинамічні моделі сегментації є еволюцією статичного підходу та передбачають використання обмеженого набору правил для автоматичного віднесення замовлення до каналу. У таких моделях канал визначається на основі певних атрибутів замовлення, наприклад типу інтеграції, ідентифікатора зовнішнього сервісу або способу оформлення замовлення.

Типовим прикладом напівдинамічної сегментації є система, у якій замовлення, що надходять через конкретний API, автоматично відносяться до відповідного каналу доставки, тоді як всі інші замовлення створюються як «внутрішні». Часто такий підхід застосовується в CRM-системах, де кожна інтеграція з маркетплейсом або онлайн-платформою асоціюється з власним каналом продажу.

Напівдинамічні моделі мають вищу гнучкість порівняно зі статичними, оскільки зменшують обсяг ручної роботи та знижують ризик помилок при класифікації замовлень. Однак їх ключовим недоліком є жорстка прив’язка каналу до технічного джерела даних. У результаті канал фактично ототожнюється з інтеграцією, а не з бізнес-сутністю, що ускладнює аналітику та уніфікацію процесів. Наприклад, телефонне замовлення та замовлення через

месенджер можуть мати різну бізнес-логіку, але в системі обліковуватися як один і той самий канал через відсутність інтеграції.

У розроблюваній моделі сегментації каналів використовується формалізована напівдинамічна модель сегментації з елементами адаптивності, де канал замовлення розглядається як окрема логічна сутність, незалежна від конкретного способу надходження замовлення. Визначення каналу здійснюється через функцію віднесення замовлення до каналу, яка аналізує набір атрибутів замовлення (джерело, тип створення, ідентифікатори інтеграції, контекст оформлення) та застосовує заздалегідь визначені правила. На відміну від класичних напівдинамічних моделей канал не зводиться до інтеграції або сценарію продажу. Один і той самий канал може об’єднувати замовлення з різних джерел, якщо вони мають спільний бізнес-контекст, наприклад «Онлайн-замовлення власного сайту» або «Агрегатори доставки». Це дозволяє застосовувати єдині правила ціноутворення, оплати та аналітики без дублювання логіки. Ключовою перевагою такої моделі є баланс між гнучкістю та керованістю. Бізнес отримує можливість адаптувати сегментацію каналів до власних потреб без ускладнення архітектури системи, що є особливо важливим для українського МСБ, де канали продажу часто змінюються та комбінуються.

Таблиця 1.1

Порівняльна таблиця моделей сегментації каналів замовлень

Модель сегментації	**Принцип роботи**	**Переваги**	**Недоліки**
Статична модель	Канал задається вручну або фіксованим значенням	Простота реалізації, мінімальні вимоги до системи	Відсутність гнучкості, ручна класифікація, низька масштабованість
Напівдинамічна модель	Канал визначається за атрибутами або інтеграцією	Автоматизація, зменшення помилок	Жорстка прив’язка до джерела, обмежена аналітика
Розроблювана модель	Канал як логічна сутність + функція віднесення	Гнучкість, уніфікація замовлень, підтримка поканального ціноутворення	Потребує початкового налаштування правил для прямої інтеграції

Аналіз статичних та напівдинамічних моделей сегментації показує, що більшість існуючих підходів не забезпечують достатньої гнучкості для ефективного управління каналами замовлень у багатоканальному середовищі. Запропонована в SkyService POS модель дозволяє подолати ці обмеження за рахунок формалізації каналу як окремої бізнес-сутності та використання адаптивних правил сегментації, що створює основу для подальшої реалізації інтегрованого управління замовленнями та поканального ціноутворення.

1.3.2 Підходи до поканального ціноутворення та їх обмеження

Поканальне ціноутворення дозволяє враховувати специфіку кожного каналу замовлень. У сучасних системах автоматизації бізнесу поканальне ціноутворення реалізується за допомогою різних підходів, що відрізняються рівнем гнучкості та адаптивністю.

Найпоширенішим і водночас найпростішим підходом є використання єдиної базової ціни для всіх каналів продажу. У цьому випадку товар або послуга має одну фіксовану ціну, яка застосовується незалежно від того, через який канал сформоване замовлення. Такий підхід характерний для початкових етапів розвитку бізнесу та часто використовується у невеликих торгових точках або закладах HoReCa. Перевагою цього підходу є мінімальні витрати на налаштування та простота обліку. Однак він ігнорує різницю у витратах між каналами. Для українського ринку, де комісії сервісів доставки можуть становити значну частину маржі, такий підхід призводить до зниження прибутковості або необхідності компенсувати витрати за рахунок інших каналів.

Більш гнучким підходом є використання фіксованих націнок або знижок для окремих каналів. У цьому випадку базова ціна коригується на певний відсоток або фіксовану суму залежно від каналу замовлення. Наприклад, для доставки через агрегатор встановлюється націнка 20%, тоді як для самовивозу може застосовуватися знижка. Цей підхід широко використовується в сучасних POS- та CRM-системах, оскільки дозволяє частково компенсувати додаткові

витрати та зберегти конкурентоспроможність. Водночас він має суттєві обмеження: коригування ціни часто реалізується на рівні загального коефіцієнта, що не враховує специфіку окремих товарних груп. Крім того, при великій кількості каналів та акцій підтримка актуальності таких правил ускладнюється і потребує ручного контролю.

Наступним етапом розвитку поканального ціноутворення є використання окремих прайс-листів для різних каналів замовлень. У такій моделі кожен канал має власний набір цін, що дозволяє максимально точно враховувати витрати та ринкові умови. Цей підхід часто застосовується у великих мережах або бізнесах з чітко розділеними каналами продажу. Основною перевагою є висока точність та прозорість ціноутворення. Проте підтримка кількох прайс-листів ускладнює управління номенклатурою, створює ризик неузгодженості цін та потребує додаткових ресурсів для адміністрування. Для малого та середнього бізнесу, зокрема в українських реаліях, такий підхід часто є надмірно складним.

Більш сучасним напрямком є контекстне поканальне ціноутворення, при якому ціна формується на основі сукупності умов: каналу замовлення, способу оплати, часу доби, типу клієнта або обсягу замовлення. У цьому випадку система використовує набір правил або сценаріїв, які визначають кінцеву ціну в момент створення замовлення. Даний підхід створює можливість досягти високого рівня гнучкості та адаптивності, однак потребує складної логіки обробки та чіткої формалізації бізнес-правил. Без належної архітектурної підтримки такі системи стають важкими в обслуговуванні та схильними до помилок. Багато існуючих рішень реалізують контекстне ціноутворення фрагментарно, що обмежує його практичну цінність.

У кваліфікаційній роботі реалізовано комбінований підхід до поканального ціноутворення, який поєднує переваги окремих прайс-листів та контекстних правил без надмірного ускладнення системи. Ціна товару розглядається як базове значення, до якого застосовуються канало-залежні правила корекції, визначені вручну, або на рівні логічного каналу замовлень.

Такий підхід дозволяє централізовано управляти цінами та швидко адаптуватися до змін комісій або ринкових умов.

Таблиця 1.2

Порівняльна таблиця підходів до поканального ціноутворення

Підхід	Гнучкість	Складність управління	Масштабованість	Основні обмеження
Єдина базова ціна	Низька	Низька	Обмежена	Не враховує витрати каналів
Націнки / знижки	Середня	Середня	Обмежена	Загальні коефіцієнти, ручне адміністрування
Окремі прайс-листи	Висока	Висока	Середня	Складність підтримки та узгодження
Контекстні правила	Дуже висока	Висока	Висока	Потребує формалізації та якісної архітектури
Комбінований підхід	Висока	Помірна	Висока	Початкове налаштування правил

Аналіз існуючих підходів до поканального ціноутворення показує, що більшість з них або надто спрощують процес формування ціни, або ускладнюють його до рівня, неприйнятного для підприємств малого та середнього бізнесу. Розроблений підхід забезпечує збалансоване рішення, яке поєднує гнучкість, прозорість та керованість, створюючи передумови для ефективного управління прибутковістю в багатоканальному середовищі.

1.4. Порівняльний аналіз рішень-аналогів

Сучасні CRM- та POS-системи для малого та середнього бізнесу пропонують базові механізми обліку замовлень у багатоканальному середовищі, однак у більшості випадків канали продажу розглядаються лише як джерела надходження даних, без повноцінної підтримки управлінських і аналітичних функцій. Проведений аналіз рішень Poster, Shopify POS та Square показує, що хоча ці системи є популярними та зручними в експлуатації, їхні моделі керування каналами замовлень мають низку концептуальних обмежень, які суттєво знижують ефективність масштабування бізнесу та прийняття управлінських рішень.

Таблиця 1.3

Порівняльна таблиця існуючих рішень

Система управління бізнесом	Методи / алгоритми управління каналами замовлень	Переваги	Недоліки
Poster	– Статична категоризація каналів – Лінійні правила обробки замовлень	– Простота групування джерел замовлень – Низькі вимоги до налаштувань – Прозорий алгоритм – Швидке налаштування	– Відсутність адаптивності – Не придатний до масштабування – Неможливо реалізувати гнучке ціноутворення – Не реагує на зміни навантаження чи попиту – Немає аналітичних чи оптимізаційних компонентів – Немає можливості вдосконалити ефективність каналів
Shopify POS	– Глобальні моделі динамічного ціноутворення – Аналітика конверсій каналів	– Динамічна зміна цін – Дає уявлення про ефективність каналу	– Застосовується до всіх каналів одразу – Немає можливості встановлення цін для конкретного каналу – Дані не використовуються для управлінських рішень – Непридатне для бізнесів, які працюють через зовнішні доставки
Square	– Жорстке відображення каналів у фіксовану структуру – Моделі фіксованих бізнес-процесів	– Стандартизоване маркування каналів для замовлень – Забезпечує стабільність контролю бізнес-процесів	– Поканальні дані не впливають на статистику – Немає алгоритмів адаптивного керування – Не підтримує поканального ціноутворення – Не впливає на ефективність управління каналів – Не впливає на поканальне ціноутворення
SkyService	– Адаптивна сегментація каналів – Гнучке канало-орієнтоване ціноутворення – Єдина модель інтегрованого управління замовленнями	– Розділення каналів за комерційними ознаками – Поканальна аналітика – Динамічність – Поканальне ціноутворення – Контроль прибутковості каналів – Централізоване управління каналами	– Обмежена кількість модулів сервісів каналів замовлень, з прямою інтеграцією – Відсутність алгоритмічної підтримки управлінських рішень

З таблиці 1.3 можемо бачити, що у системі Poster канали замовлень реалізовані переважно у статичному вигляді та не підтримують адаптивної логіки. Відсутні механізми гнучкого поканального ціноутворення, система не реагує на зміни навантаження чи попиту, а канали не можуть бути оптимізовані з точки зору прибутковості. Аналітичні інструменти мають узагальнений характер і не дозволяють оцінювати ефективність окремих каналів або вдосконалювати їхню роботу. У результаті рішення є обмежено придатним для масштабування та розвитку багатоканальних бізнес-моделей.

Shopify POS орієнтований на централізоване управління продажами, однак застосовує єдину модель обліку до всіх каналів одночасно. Це унеможливлює встановлення індивідуальних цін для конкретних каналів та використання поканальних даних для управлінських рішень. Система добре працює в межах власної екосистеми, проте є малопридатною для бізнесів, що активно використовують зовнішні сервіси доставки або сторонні маркетплейси, де канали мають суттєво різні економічні характеристики.

У Square канали замовлень фактично не інтегровані у модель управління бізнесом. Поканальні дані не впливають на статистику та аналітику, відсутні алгоритми адаптивного керування та механізми поканального ціноутворення. Як наслідок, система не дозволяє підвищувати ефективність управління каналами та не надає інструментів для контролю їх прибутковості або оптимізації структури продажів.

Аналіз сучасних CRM- та POS-систем для малого та середнього бізнесу демонструє що більшість наявних рішень розглядають канали замовлень переважно як допоміжний атрибут обліку, а не як повноцінний об’єкт управління. У таких системах канали фіксують джерело надходження замовлення, проте практично не впливають на логіку ціноутворення, аналітику або прийняття керівницьких рішень. Це призводить до неможливості бізнесу коректно оцінювати реальну ефективність окремих каналів, порівнювати їхню прибутковість та адаптувати процеси продажів до змін ринку, навантаження або структури попиту.

Додатковою проблемою є обмеженість моделей сегментації та ціноутворення, що застосовуються в рішеннях-аналогах. Статичні або напівдинамічні підходи рішень аналогів, не враховують комерційних відмінностей між каналами. Відсутність гнучкого поканального ціноутворення та алгоритмів адаптивного управління призводить до зниження маржинальності та ускладнює масштабування бізнесу в умовах багатоканального продажу, особливо для сегменту HoReCa та сервісних компаній, орієнтованих на зовнішні платформи доставки. Це обумовлює актуальність розробки власної інтегрованої методики управління каналами замовлень, яка поєднує формалізовану модель уніфікованого замовлення, механізми адаптивної сегментації каналів та гнучке канало-орієнтоване ціноутворення.

2 РОЗРОБКА І МОДЕЛЮВАННЯ МЕТОДІВ УПРАВЛІННЯ КАНАЛАМИ ЗАМОВЛЕНЬ

2.1. Формалізація задачі управління каналами замовлень

Ефективне управління каналами замовлень у системах обліку малого та середнього бізнесу потребує чіткого формалізованого підходу, що дозволяє описати як саму систему, так і процеси, які в ній відбуваються. В умовах багатоканального середовища замовлення надходять з різнорідних джерел, мають відмінні комерційні характеристики та обробляються за різними сценаріями, що ускладнює їх централізований облік і аналіз. Тому виникає необхідність представлення системи обліку замовлень як цілісного об’єкта дослідження з визначеними елементами, параметрами та функціональними зв’язками між ними.

Формалізація задачі управління каналами замовлень полягає у виділенні ключових функцій системи, що відіграють основну роль у підвищенні ефективності бізнес-процесів, а також у встановленні взаємозв’язків між параметрами каналів, замовлень та результативними показниками діяльності. Такий підхід створює основу для побудови математичних моделей, алгоритмів поканального ціноутворення та інтегрованого управління замовленнями, що надалі дозволяє обґрунтовано оцінювати вплив запропонованих рішень на продуктивність системи та досягнення поставленої мети кваліфікаційної роботи.

2.1.1. Опис системи обліку замовлень як об’єкта дослідження

Система обліку замовлень у малому та середньому бізнесі розглядається як складна соціально-технічна інформаційна система, призначена для приймання, обробки, збереження та аналізу даних про замовлення, що надходять із різних каналів продажу. У контексті даної роботи така система функціонує в умовах багатоканального середовища, де кожне замовлення може бути сформоване через різні точки взаємодії з клієнтом: касу закладу (POS), вебсайт,

мобільний застосунок, сторонні сервіси доставки або вручну оператором. Об’єктом дослідження є не окремі програмні компоненти, а сукупність процесів, моделей даних і алгоритмів, що забезпечують коректне управління замовленнями з урахуванням їхнього каналу походження.

З функціональної точки зору система обліку замовлень включає такі ключові підсистеми: підсистему приймання замовлень, підсистему нормалізації та валідації даних, підсистему визначення каналу замовлення, підсистему ціноутворення, підсистему збереження та аналічної обробки інформації. Кожна з цих підсистем виконує визначену роль у загальному процесі управління, а їх взаємодія забезпечує цілісність і узгодженість облікових даних. Особливістю досліджуваної системи є те, що канал замовлення розглядається не як допоміжний атрибут, а як повноцінний керуючий параметр, який впливає на логіку обробки замовлення, розрахунок ціни та подальший аналіз.

Як об’єкт дослідження система характеризується наявністю множини вхідних потоків даних, що мають різну структуру та семантику. Наприклад, замовлення з POS-терміналу зазвичай містить мінімальний набір параметрів і обробляється в режимі реального часу, тоді як замовлення з API зовнішньої служби доставки може містити додаткові поля, такі як комісія сервісу, ідентифікатор платформи або тип оплати. Це зумовлює необхідність уніфікації структури замовлення та формального опису правил приведення даних до єдиного формату, що є однією з центральних задач у межах даної роботи.

З позиції системного аналізу облік замовлень можна представити як відображення множини подій бізнес-середовища у формалізовану інформаційну модель. Кожне замовлення є елементарною подією, яка має набір атрибутів, часові характеристики та зв’язки з іншими сутностями системи, зокрема товарами, клієнтами, каналами замовлень і платіжними методами. Таким чином, система обліку замовлень функціонує як ядро інформаційної системи, навколо якого будуються аналітичні, управлінські та фінансові модулі.

Важливою властивістю об’єкта дослідження є його динамічний характер. Кількість каналів замовлень, їхні параметри, правила ціноутворення та

навантаження можуть змінюватися з часом, що потребує від системи здатності адаптуватися до нових умов без порушення цілісності даних. У традиційних рішеннях такі зміни часто потребують ручного втручання або призводять до втрати аналітичної точності. У межах даної роботи система обліку замовлень розглядається як адаптивна структура, здатна враховувати зміни параметрів каналів у процесі функціонування.

З технічної сторони об’єкт дослідження реалізується у вигляді модульної програмної системи, що інтегрує базу даних, серверну логіку та інтерфейси взаємодії з користувачем і зовнішніми сервісами. Прикладом такої системи є SkyService POS, у якій модуль управління каналами замовлень інтегрований у загальну архітектуру обліку продажів. Це дозволяє досліджувати систему не лише на теоретичному рівні, а й у контексті практичної реалізації, що підвищує прикладну цінність результатів розробки.

Так система обліку замовлень як об’єкт дослідження в межах даної магістерської роботи являє собою багаторівневу інформаційну систему з множиною взаємопов’язаних компонентів, параметрів та процесів. Її дослідження з позицій формалізації, моделювання та алгоритмічного опису створює основу для подальшого визначення цільових властивостей системи та встановлення взаємозв’язків між її параметрами, що розглядається у наступних підрозділах.

2.1.2. Визначення цільових властивостей системи

Визначення цільових властивостей системи обліку замовлень є ключовим етапом формалізації задачі управління каналами замовлень, оскільки саме ці властивості характеризують бажаний стан системи та слугують критеріями оцінки ефективності запропонованих методів і алгоритмів. У контексті багатоканального бізнес-середовища цільові властивості мають відображати здатність системи забезпечувати коректний облік, аналітичну прозорість і підтримку управлінських рішень з урахуванням специфіки кожного каналу

замовлень. На відміну від традиційних систем, де головний акцент робиться на фіксації фактів продажу, у даній роботі цільові властивості орієнтовані на активне використання поканальних даних.

Першою базовою цільовою властивістю системи є уніфікованість обліку замовлень, яка передбачає приведення замовлень з різних джерел до єдиного формального представлення. Це дозволяє обробляти, зберігати та аналізувати інформацію незалежно від їх джерела і формату та усуває проблему фрагментації даних. Уніфікованість є необхідною умовою для реалізації поканального ціноутворення, статистики та порівняльної аналітики.

Другою важливою цільовою властивістю є ідентифікованість каналу замовлення, тобто здатність системи однозначно відносити кожне замовлення до конкретного каналу або їхньої комбінації. Канал у цьому випадку розглядається як керуючий параметр, що впливає на подальші етапи обробки замовлення, включаючи вибір цінової логіки, методів оплати та аналітичних агрегатів. Відсутність чіткої ідентифікації каналу призводить до спотворення статистичних даних та унеможливлює коректне управління.

Наступною цільовою властивістю системи є гнучкість поканального ціноутворення, яка полягає у можливості встановлення індивідуальних цін або модифікаторів (націнок, знижок) для кожного каналу замовлень. Така властивість дозволяє враховувати різні комерційні умови, зокрема комісії сервісів доставки, витрати на логістику або маркетингові стратегії, і безпосередньо впливає на прибутковість бізнесу. Гнучкість ціноутворення є однією з ключових відмінностей запропонованої системи від більшості існуючих рішень-аналогів.

- окрему групу цільових властивостей формують аналітичні та управлінські характеристики системи, до яких належать;
- можливість формування статистики за окремими каналами замовлень;
- порівняльний аналіз ефективності каналів за фінансовими та операційними показниками;

- фільтрація та агрегація даних з урахуванням каналу, часу та типу замовлення;
- підтримка прийняття управлінських рішень на основі поканальних даних.

Ще однією цільовою властивістю є масштабованість системи, яка полягає у здатності додавати нові канали замовлень, змінювати їх параметри та правила обробки без необхідності суттєвої перебудови архітектури. Масштабованість є критично важливою для малого та середнього бізнесу, що вимушений активно підлаштовуватись до нових ринкових умов і платформ продажу.

Завершальною, але не менш важливою цільовою властивістю є адаптивність системи до змін бізнес-середовища. Вона полягає у здатності системи коректно функціонувати при зміні навантаження, структури замовлень або комерційних умов каналів. Хоча в межах даної роботи не розглядається автоматична оптимізація, сама можливість гнучкого налаштування параметрів каналів створює передумови для подальшого розвитку та розширення можливосей адаптивних механізмів.

2.1.3. Встановлення взаємозв'язків між параметрами системи

Для формалізованого опису системи управління каналами замовлень необхідно встановити взаємозв'язки між її основними параметрами та цільовими властивостями. Нехай система обліку замовлень описується як сукупність множин і відображень, що формують єдиний інформаційний простір. Основною сутністю є замовлення, яке надходить до системи з певного джерела та обробляється відповідно до параметрів каналу замовлень.

Позначимо множину всіх замовлень як

$$O = \{ o_1, o_2, \ldots, o_n \},$$

де кожне замовлення $o_i \in O$ описується кортежем параметрів:

$$o_i = \langle id_i, t_i, S_i, P_i, q_i, p_i \rangle,$$

де

id_i – унікальний ідентифікатор замовлення;

t_i – час створення замовлення;

S_i – джерело надходження (API, POS, сайт, ручне введення);

P_i – множина товарних позицій;

q_i – кількість одиниць товарів;

p_i – підсумкова ціна замовлення.

Множину каналів замовлень визначимо як

$$C = \{ c_1, c_2, \ldots, c_m \},$$

де кожен канал $c_j \in C$ характеризується набором параметрів:

$$c_j = \langle type_j, k_j, r_j \rangle,$$

де

$type_j$ – тип каналу (офлайн, онлайн, доставка, маркетплейс);

k_j – ціновий коефіцієнт або модифікатор каналу;

r_j – набір правил обробки та оплати.

Віднесення замовлення до конкретного каналу описується функцією класифікації:

$$g : O \to C,$$

де

$g(o_i) = c_j$, якщо замовлення o_i відповідає умовам каналу c_j за джерелом S_i та додатковими атрибутами.

Цінова логіка в системі визначається як функція, що залежить від базової ціни товарів і параметрів каналу. Нехай базова ціна замовлення визначається як:

$$p_i^0 = \Sigma (price_k \cdot q_k), \ k \in P_i,$$

де $price_k$ – базова ціна одиниці товару, q_k – його кількість.

Тоді підсумкова ціна замовлення з урахуванням каналу обчислюється за формулою:

$$p_i = p_i^0 \cdot k_j, \ \text{де } c_j = g(o_i).$$

У випадку використання фіксованої канальної ціни формула набуває вигляду:

$$p_i = p_i^c,$$

де p_i^c – заздалегідь визначена ціна для каналу c_j.

Аналітичні показники системи формуються на основі агрегації замовлень за каналами. Наприклад, сумарний обсяг продажів для каналу c_j за період T визначається як:

$$R(c_j, T) = \Sigma\, p_i, \text{ де } g(o_i) = c_j \text{ і } t_i \in T.$$

Аналогічно кількість замовлень за каналом описується як:

$$N(c_j, T) = |\ \{\ o_i \in O : g(o_i) = c_j \text{ і } t_i \in T\ \}\ |.$$

Дане представлення демонструє як властивості системи безпосередньо залежать від параметрів каналів і функцій віднесення та ціноутворення. Зміна параметрів каналу k_j або правил r_j призводить до зміни підсумкових значень p_i, $R(c_j, T)$ та інших аналітичних показників, що дозволяє формалізовано оцінювати вплив каналів замовлень на фінансові та управлінські характеристики системи.

Отримані взаємозв’язки створюють математичну основу для подальшої розробки алгоритмів інтегрованого управління замовленнями та реалізації гнучкого поканального ціноутворення у модулі керування каналами.

2.2. Модель уніфікованого замовлення

У багатоканальних системах обліку малого та середнього бізнесу замовлення надходять із різних джерел, що відрізняються як форматом даних, так і семантикою окремих атрибутів. POS-термінали, веб інтерфейси, мобільні застосунки та зовнішні API сервісів доставки формують замовлення з різним набором полів, правилами заповнення та бізнес-логікою. За відсутності єдиного підходу до представлення таких даних виникає фрагментація обліку, ускладнюється аналітика та стає неможливим застосування універсальних алгоритмів управління каналами замовлень.

У зв’язку з цим ключовим елементом розроблюваної методики є побудова моделі уніфікованого замовлення, яка забезпечує приведення різнорідних

вхідних даних до єдиної формальної структури. Така модель дозволяє відокремити логіку приймання замовлень від логіки їх обробки, ціноутворення та аналітики, що підвищує масштабованість і гнучкість системи.

2.2.1. Необхідність нормалізації замовлень з різних джерел

У системах обліку малого та середнього бізнесу замовлення формуються та надходять із різних джерел, кожне з яких використовує власні правила структурування даних, набори атрибутів та семантичні позначення. Наприклад, замовлення, створене через POS-термінал, зазвичай містить мінімальний набір інформації, необхідної для швидкого обслуговування клієнта, тоді як замовлення з API сервісу доставки може включати додаткові параметри, пов’язані з логістикою, комісіями, ідентифікаторами зовнішніх платформ і специфікою оплати. Відсутність єдиного підходу до подання таких даних призводить до розрізненого зберігання інформації та ускладнює подальшу обробку.

Без нормалізації кожне джерело замовлень фактично потребує окремої логіки обліку, що значно ускладнює підтримку та масштабування системи. Аналітичні запити у такому випадку мають враховувати специфіку кожного формату, що підвищує ймовірність помилок і знижує достовірність отриманих показників. Крім того, поканальне ціноутворення та порівняльний аналіз ефективності каналів стають неможливими або потребують значних додаткових обчислень, оскільки відсутній єдиний базис для зіставлення замовлень між собою.

Нормалізація замовлень дозволяє відокремити процес інтеграції даних від бізнес-логіки системи. У межах нормалізації різнорідні вхідні структури перетворюються на уніфіковане представлення з фіксованим набором обов’язкових і додаткових атрибутів. Це забезпечує коректну ідентифікацію каналу замовлення, застосування канало-орієнтованої ціни та подальший облік у статистиці незалежно від джерела надходження даних. Таким чином, нормалізація є фундаментальною умовою для реалізації інтегрованої моделі управління каналами замовлень.

У контексті програми SkyService POS необхідність нормалізації обумовлена орієнтацією системи на багатоканальні бізнеси, зокрема у сфері HoReCa, де одночасно використовуються офлайн-продажі, онлайн-замовлення та сторонні сервіси доставки. Єдина модель уніфікованого замовлення дозволяє забезпечити цілісність даних, підвищити точність аналітики та створити умови для майбутнього розвитку адаптивних і аналітичних механізмів управління каналами замовлень.

2.2.2. Формальна математична модель структури замовлення

Для забезпечення уніфікованого обліку замовлень з різних джерел у межах системи управління каналами замовлень вводиться формальна математична модель структури замовлення. Метою побудови такої моделі є визначення єдиного набору атрибутів, які достатні для коректної обробки, ціноутворення та аналітичного аналізу незалежно від джерела надходження замовлення.

Нехай множина всіх замовлень, що обробляються системою, визначається як

$$O = \{ o_1, o_2, \ldots, o_n \}.$$

Кожне замовлення $o_i \in O$ після нормалізації подається у вигляді впорядкованого кортежу:

$$o_i = \langle id_i, t_i, c_i, s_i, I_i, P_i, Q_i, B_i, p_i \rangle,$$

де

id_i – унікальний ідентифікатор замовлення;

t_i – час створення або приймання замовлення;

c_i – ідентифікатор каналу замовлення;

s_i – джерело надходження (POS, сайт, API, ручне введення);

I_i – ідентифікатор зовнішнього сервісу (за наявності);

P_i – множина товарних позицій замовлення;

Q_i – відповідна множина кількостей товарів;

B_i – базова вартість замовлення до застосування канальної логіки;

p_i – підсумкова вартість замовлення.

Множина товарних позицій P_i визначається як

$P_i = \{ p_1, p_2, \ldots, p_k \}$,

де кожна позиція p_j характеризується базовою ціною $price_j$.

Множина кількостей $Q_i = \{ q_1, q_2, \ldots, q_k \}$ відповідає кількості кожної позиції у замовленні.

Базова вартість замовлення обчислюється за формулою:

$$B_i = \Sigma (price_j \cdot q_j), p_j \in P_i.$$

Після визначення каналу замовлення c_i застосовується канало-орієнтована логіка ціноутворення. Для цього вводиться множина каналів

$C = \{ c_1, c_2, \ldots, c_m \}$,

де кожен канал c_j описується набором параметрів:

$c_j = \langle k_j, d_j \rangle$,

де

k_j – коефіцієнт або правило корекції ціни;

d_j – додаткові параметри каналу (правила оплати, обмеження).

У випадку використання коефіцієнтної моделі підсумкова вартість замовлення визначається як:

$$p_i = B_i \cdot k_j, \text{ де } c_j = c_i.$$

У випадку фіксованої канальної ціни підсумкова вартість визначається як:

$$p_i = f(c_i, P_i),$$

де f – функція, що повертає заздалегідь визначену ціну для заданого каналу та набору товарів.

2.2.3. Процес нормалізації та обробки вхідних даних

Процес нормалізації та обробки вхідних даних забезпечує приведення різнорідних даних з різних джерел до єдиного формату. Це дозволяє спростити подальше оброблення замовлень, застосування ціноутворення, а також аналіз

ефективності кожного каналу. Нормалізація даних включає кілька етапів, які забезпечують консистентність і коректність обробки замовлень незалежно від джерела їх походження.

1. Приймання даних

На першому етапі система отримує замовлення з різних джерел, таких як POS-термінали, вебсайти, мобільні застосунки, API сторонніх сервісів доставки або вручну введені дані. Кожне джерело має свою специфіку в форматі і структури даних. Наприклад, замовлення з POS може містити лише базову інформацію про товари та кількість, тоді як замовлення з API платформи доставки може мати додаткові атрибути, такі як ідентифікатор замовлення в системі доставки, вартість доставки та комісію сервісу. Ціль даного етапу заключається у виявлені всіх вхідних даних та підготувати їх до подальшої обробки.

2. Приведення до єдиного формату

Після отримання даних система повинна виконати їх нормалізацію, що полягає у приведенні всіх параметрів замовлення до єдиного формату. Кожне замовлення, що надходить, повинно містити стандартний набір атрибутів, визначений математичною моделлю замовлення (див. розділ 2.2.2). Для цього система має виконати наступні кроки:

- ідентифікація каналу: кожне замовлення повинно бути віднесене до певного каналу, що визначається на основі джерела або типу взаємодії. Наприклад, замовлення через вебсайт буде мати канал типу "онлайн", а через POS — "офлайн";
- стандартизація полів: всі замовлення повинні мати однакову структуру для таких атрибутів, як товар, кількість, ціна, час замовлення, метод оплати, тощо. Наприклад, якщо одне джерело надсилає ціни в форматі "цифра + символ валюти", а інше — у вигляді числових значень, система повинна привести їх до єдиного формату;
- заповнення відсутніх полі: при отриманні даних з різних джерел можуть бути відсутні деякі поля, наприклад, якщо канал не передає ідентифікатор

користувача або адресу доставки. У такому випадку система повинна або запитати ці дані у користувача, або заповнити їх значеннями за замовчуванням, якщо це допустимо для бізнес-логіки;

3. Валідація даних

На етапі валідації система перевіряє коректність отриманих даних.

Зокрема, необхідно перевірити:

- цілісність даних: перевірка, чи не пропущені обов’язкові параметри (наприклад, якщо замовлення не містить товар або кількість). Якщо параметри відсутні, система повинна або згенерувати повідомлення про помилку, або запросити користувача для заповнення пропусків;
- коректність значен: перевірка на коректність значень, таких як ціна товару (наприклад, вона не повинна бути від'ємною) або кількість товарів (не більше, ніж наявно на складі);
- логічна цілісніст: перевірка логічних взаємозв’язків, наприклад, чи не перевищує кількість замовлених товарів доступні на складі, чи відповідає метод оплати можливим варіантам (наприклад, якщо це замовлення через агрегатор доставки, то платіж може бути доступний лише через онлайн-систему).

4. Прив’язка до каналу замовлення

Кожне замовлення після нормалізації повинно бути чітко прив’язане до відповідного каналу замовлення. Канал є ключовим елементом для подальшого застосування поканального ціноутворення, аналітики та розрахунку прибутковості. Для визначення каналу використовується функція, яка залежить від джерела замовлення, типу операції або комбінації параметрів. Наприклад:

- якщо замовлення надходить через API зовнішнього сервісу доставки, канал визначається як "Доставка";
- якщо замовлення зроблене через сайт, воно потрапляє під канал "Онлайн-продаж";
- якщо замовлення здійснене через POS-термінал у фізичній точці, воно відноситься до каналу "Офлайн-продаж".

5. Обробка цін та застосування логіки ціноутворення

Після нормалізації і прив’язки до каналу, система застосовує необхідні правила ціноутворення, що можуть включати базову ціну товару, націнки або знижки залежно від каналу. Ціни для кожного каналу можуть бути встановлені індивідуально, що дозволяє врахувати специфічні умови кожного каналу (наприклад, комісії платіжних систем чи зовнішніх сервісів доставки). Підсумкова ціна замовлення для кожного каналу обчислюється згідно з математичною моделлю ціноутворення, що включає:

- базову ціну товару, визначену на основі стандартного прайс-листа;
- модифікації ціни в залежності від каналу продажу (націнки або знижки);
- додаткові витрати, пов’язані з конкретним каналом, такі як комісії за обробку платіжних карток або плата за доставку.

6. Збереження та передачу даних

Після обробки і нормалізації замовлення інформація про нього зберігається у базі даних системи. Замовлення з усіма атрибутами, що були нормалізовані та оброблені, передається до відповідних модулів системи для подальшої обробки, аналітики та формування звітів. Всі вхідні дані можуть бути передані для зберігання в центральній базі або у вигляді черг повідомлень для подальшої асинхронної обробки.

2.3. Модель віднесення замовлення до каналу

Модель віднесення замовлення до каналу визначає, які правила обробки, ціноутворення та аналітики будуть застосовані до конкретного замовлення. У багатоканальному середовищі джерело надходження замовлення не завжди однозначно визначає його бізнес-контекст, тому необхідним є формалізований механізм класифікації, який дозволяє відокремити технічний спосіб отримання даних від логічного каналу управління.

Нехай множина всіх нормалізованих замовлень системи визначається як

$O = \{ o_1, o_2, \ldots, o_n \}$.

Кожне замовлення $o_i \in O$ після нормалізації має структуру:

$$o_i = \langle id_i, t_i, s_i, I_i, A_i \rangle, \quad (2.1)$$

де

id_i – унікальний ідентифікатор замовлення;

t_i – час створення замовлення;

s_i – джерело надходження (POS, сайт, API, ручне введення);

I_i – ідентифікатор зовнішньої інтеграції (за наявності);

A_i – множина додаткових атрибутів замовлення (тип оплати, тип доставки, контекст створення).

Множину каналів замовлень визначимо як

$C = \{ c_1, c_2, \ldots, c_m \}$,

де кожен канал $c_j \in C$ описується кортежем:

$$c_j = \langle name_j, R_j \rangle, \quad (2.2)$$

де

$name_j$ – унікальна назва або ідентифікатор каналу;

R_j – множина правил віднесення замовлень до каналу.

Процес визначення каналу замовлення формалізується за допомогою функції віднесення:

$g : O \rightarrow C$,

де

$g(o_i) = c_j$, якщо замовлення o_i задовольняє правила R_j.

Кожне правило $r \in R_j$ задається у вигляді логічної умови над атрибутами замовлення:

$$r(o_i) = \varphi(s_i, I_i, A_i), \quad (2.3)$$

де φ – булева функція, що повертає значення істина, якщо замовлення відповідає умовам каналу.

Приклад формального правила віднесення замовлення до каналу доставки:

$r_1(o_i) = (s_i = \text{"API"}) \land (I_i \in \{\text{Glovo, BoltFood}\})$.

Приклад правила для офлайн-каналу:

$r_2(o_i) = (s_i = \text{"POS"}) \land (A_i.\text{type} = \text{"in\_store"})$.

У загальному випадку канал c_j вважається відповідним замовленню o_i, якщо:

$\exists\ r \in R_j : r(o_i) = \text{true}$.

Тоді функція g визначається як:

$$g(o_i) = \arg\max\_\{c_j \in C\}\ M(c_j, o_i), \qquad (2.4)$$

де

$M(c_j, o_i)$ – міра відповідності замовлення каналу, що дорівнює кількості виконаних правил:

$M(c_j, o_i) = | \{ r \in R_j : r(o_i) = \text{true} \} |$.

Для уникнення неоднозначностей вводиться правило пріоритету каналів. Нехай кожному каналу відповідає коефіцієнт пріоритету w_j. Остаточний вибір каналу визначається як:

$g(o_i) = \arg\max\_\{c_j \in C\} (M(c_j, o_i) \cdot w_j)$.

У випадку, якщо для замовлення не виконано жодного правила, воно відноситься до каналу за замовчуванням c_0:

$g(o_i) = c_0$, якщо $\forall\ c_j \in C : M(c_j, o_i) = 0$.

Запропонована модель дозволяє формалізувати процес віднесення замовлення до каналу незалежно від технічного джерела даних, забезпечує масштабованість при додаванні нових каналів і правил та створює основу для подальшого застосування канало-орієнтованого ціноутворення і аналітики. Використання такої моделі в системі забезпечує гнучке та централізоване управління каналами замовлень у багатоканальному бізнес-середовищі.

2.4 Модель гнучкого поканального ціноутворення

Різні канали замовлень мають відмінні економічні характеристики, зокрема різні рівні комісій, витрати на обробку та логістику, що потребує індивідуального підходу до формування ціни. Використання єдиної ціни для всіх каналів або обмежених механізмів корекції не дозволяє повною мірою врахувати ці особливості та призводить до зниження ефективності управління продажами.

У зв’язку з цим розроблюваним модулем пропонується модель гнучкого поканального ціноутворення, яка базується на поєднанні базової ціни товару з канало-залежними правилами її модифікації. Такий підхід дозволяє централізовано управляти цінами, зберігаючи прозорість та керованість системи, а також забезпечує можливість швидкої адаптації до змін комерційних умов.

2.4.1. Базова модель ціни товару

Базова модель ціни товару є вихідною складовою гнучкого поканального ціноутворення та визначає початкове значення, від якого виконуються всі подальші корекції залежно від каналу замовлення. У межах даної роботи базова ціна розглядається як величина, що встановлюється бізнесом незалежно від способу продажу та відображає стандартну вартість товару або послуги в умовах офлайн-продажу або внутрішнього каналу за замовчуванням. Саме така інтерпретація базової ціни дозволяє зберегти цілісність номенклатури та уникнути дублювання товарів для різних каналів.

Нехай множина всіх товарів у системі визначається як

$T = \{ t_1, t_2, \ldots, t_k \}$.

Кожен товар $t_j \in T$ характеризується базовою ціною:

$price_j^0 \in \mathbb{R}^+$,

де $price_j^0$ – базова ціна одиниці товару t_j, встановлена у системі обліку.

Для окремого замовлення o_i, що містить підмножину товарів

$P_i \subseteq T$,

та відповідну множину кількостей

$Q_i = \{ q_1, q_2, \ldots, q_n \}$,

базова вартість замовлення визначається як сума базових цін усіх товарних позицій:

$$B_i = \Sigma (price_j^0 \cdot q_j), t_j \in P_i. \quad (2.5)$$

Отримане значення B_i є канало-незалежною вартістю замовлення та відображає його номінальну ціну без урахування комерційних особливостей конкретного каналу. Такий підхід дозволяє чітко відокремити етап формування базової ціни від етапу її модифікації, що є важливим для забезпечення прозорості та керованості системи ціноутворення.

Базова модель ціни товару має низку важливих властивостей. По-перше, вона є стабільною у часі та не залежить від структури каналів замовлень, що спрощує адміністрування номенклатури. По-друге, вона забезпечує єдину точку відліку для порівняльного аналізу прибутковості каналів, оскільки всі відхилення кінцевої ціни можуть бути формально пояснені дією канало-орієнтованих правил. По-третє, базова ціна може використовуватися як контрольне значення при формуванні звітності та перевірці коректності розрахунків.

У системі SkyService базова модель ціни товару реалізується як частина єдиної моделі уніфікованого замовлення та не залежить від способу створення замовлення. Це дозволяє застосовувати однакові алгоритми обробки та аналітики до всіх каналів, зберігаючи при цьому можливість гнучкого поканального ціноутворення на наступних етапах формування підсумкової вартості замовлення.

2.4.2. Кастомні ціни та модифікатори каналів

Кастомні ціни та модифікатори каналів є ключовими інструментами реалізації гнучкого поканального ціноутворення, оскільки вони дозволяють адаптувати базову модель ціни товару до комерційних умов конкретного каналу замовлень. На відміну від статичних прайс-листів, такий підхід забезпечує централізоване управління цінами без дублювання номенклатури та спрощує масштабування системи при додаванні нових каналів або зміні їхніх параметрів.

Нехай множина каналів замовлень визначається як

$$C = \{ c_1, c_2, \ldots, c_m \}. \quad (2.6)$$

Для кожного каналу $c_j \in C$ у системі можуть бути визначені два типи цінових параметрів: кастомні ціни та модифікатори. Кастомна ціна є явним значенням ціни товару для конкретного каналу, тоді як модифікатор визначає правило корекції базової ціни.

Кастомну ціну для товару t_k у каналі c_j позначимо як:

$price_{jk}^{c} \in \mathbb{R}^{+}$,

де $price_{jk}^{c}$ – фіксована ціна товару t_k для каналу c_j.

У разі наявності кастомної ціни вона має вищий пріоритет над усіма іншими правилами ціноутворення.

Модифікатор каналу визначається як коефіцієнт або функція корекції базової ціни. У найпростішому випадку модифікатор задається числовим коефіцієнтом:

$k_j \in \mathbb{R}^{+}$,

де

$k_j > 1$ відповідає націнці,

$k_j < 1$ відповідає знижці,

$k_j = 1$ означає відсутність змін.

Тоді ціна товару t_k у каналі c_j за умови відсутності кастомної ціни обчислюється як:

$$price_{jk} = price_k^{o} \cdot k_j,$$

де $price_k^{o}$ – базова ціна товару.

У більш загальному випадку модифікатор може бути заданий у вигляді функції:

$$k_j = h(c_j, t_k),$$

де h – функція, що дозволяє враховувати додаткові параметри каналу або товару, наприклад тип категорії або сценарій продажу.

Для окремого замовлення o_i, що містить товарні позиції

$P_i = \{ t_1, t_2, \ldots, t_n \}$,

підсумкова вартість замовлення з урахуванням кастомних цін і модифікаторів визначається як:

$$p_i = \Sigma\, p_j,$$

де для кожної позиції t_k виконується правило пріоритету:

$p_j =$

$price_{jk}^{c} \cdot q_k$, якщо $price_{jk}^{c}$ визначена;

$price_k^0 \cdot k_j \cdot q_k$, якщо кастомна ціна відсутня;

$price_k^0 \cdot q_k$, якщо не визначено ні кастомної ціни, ні модифікатора.

2.4.3. Алгоритм застосування канало-орієнтованої ціни

Алгоритм застосування канало-орієнтованої ціни забезпечує формування кінцевої вартості товару в замовленні з урахуванням параметрів каналу замовлень та встановлених цінових правил. Його основною метою є реалізація чіткого ієрархічного механізму вибору ціни, який унеможливлює конфлікти між різними ціновими сценаріями та гарантує однаковий результат розрахунку незалежно від джерела надходження замовлення. Алгоритм виконується після визначення каналу замовлення та використовується як єдиний механізм ціноутворення в системі.

На першому етапі алгоритму виконується отримання базової ціни товару з основного прайс-листа системи. Базова ціна є канало-незалежною величиною та слугує початковою точкою для подальших перевірок. Після цього система здійснює перевірку наявності кастомної ціни для відповідного каналу замовлення. Кастомна ціна задається явно для конкретного каналу та має найвищий пріоритет, оскільки відображає фіксовану вартість товару з урахуванням специфіки цього каналу.

У випадку, якщо кастомна ціна для каналу визначена, вона безпосередньо застосовується до товару та використовується для формування вартості позиції в замовленні. Якщо ж кастомна ціна відсутня, алгоритм переходить до наступного етапу — перевірки наявності модифікатора ціни у вигляді націнки або знижки, встановленої для даного каналу. За наявності такого модифікатора базова ціна коригується шляхом застосування відповідного коефіцієнта або націнки, що дозволяє врахувати комерційні особливості каналу.

Якщо для каналу не визначено ані кастомної ціни, ані модифікатора, алгоритм використовує базову ціну товару без будь-яких змін. Завершальним кроком є внесення обчисленої ціни до замовлення, після чого вона

використовується для розрахунку підсумкової вартості замовлення та подальшого статистичного й аналітичного обліку.

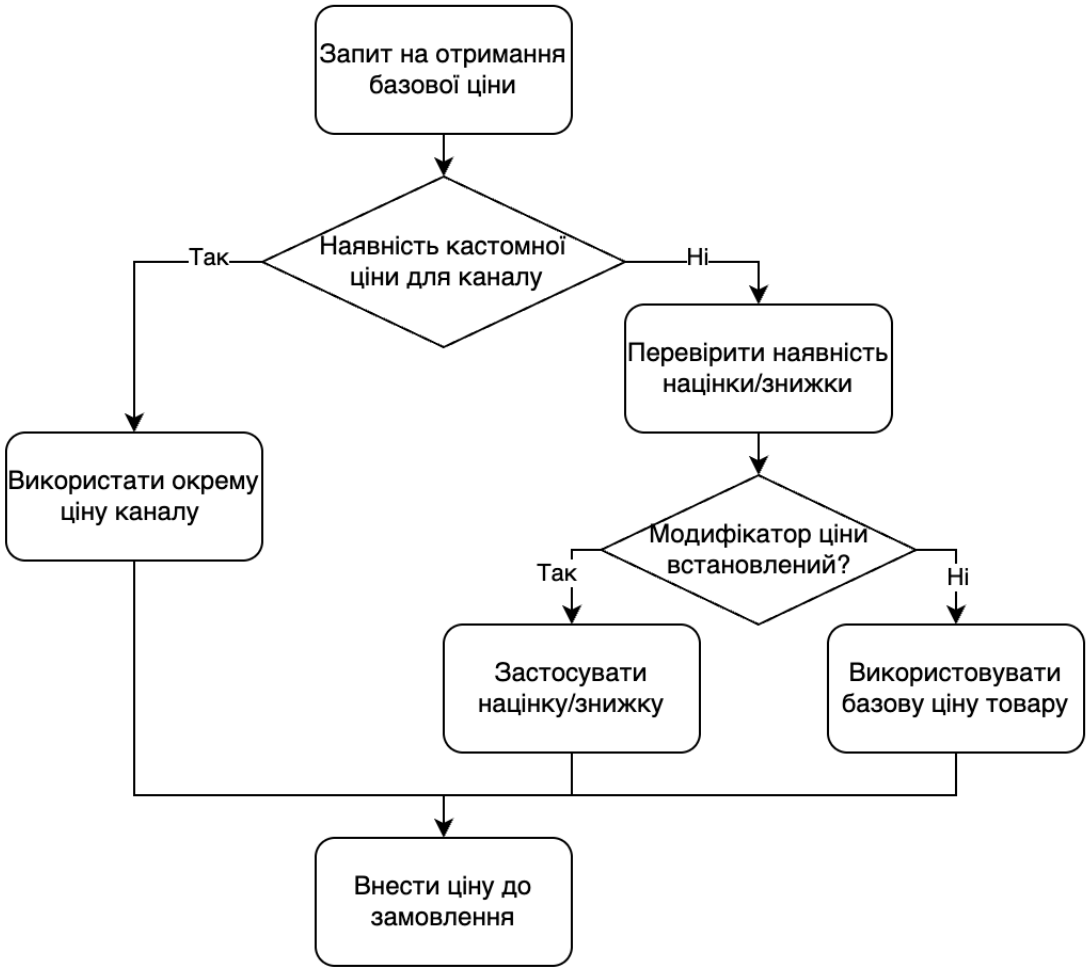


Рис 2.1 Алгоритм застосування канало-орієнтованої ціни

2.5. Інтегрована модель управління замовленнями

В умовах багатоканального бізнес-середовища ефективне управління замовленнями неможливе без узгодженої взаємодії всіх функціональних компонентів системи обліку. Розрізнене використання окремих механізмів нормалізації, сегментації, ціноутворення та аналітики не забезпечує цілісного контролю над процесом обробки замовлень і не дозволяє розкрити потенціал роздільного управління каналами у повній мірі. У зв’язку з цим виникає

необхідність побудови інтегрованої моделі управління замовленнями, яка об’єднує всі розроблені методи та алгоритми в єдину логічну схему.

Інтегрована модель управління замовленнями, запропонована в даній роботі, забезпечує послідовну обробку замовлення на кожній стадії його життєвого циклу — від моменту надходження з різних джерел до формування аналітичних показників за каналами. Модель базується на уніфікованому представленні замовлення, формалізованій функції віднесення до каналу та алгоритмах гнучкого поканального ціноутворення, що дозволяє централізовано керувати логікою обробки незалежно від способу створення замовлення.

2.5.1. Загальний алгоритм обробки замовлення

Загальний алгоритм обробки замовлення в інтегрованій моделі управління замовленнями описує послідовність взаємопов’язаних етапів, які забезпечують коректну обробку замовлень незалежно від джерела їх надходження. Алгоритм побудований таким чином, щоб уніфікувати вхідні дані, визначити бізнес-контекст замовлення, застосувати відповідні правила ціноутворення та забезпечити подальший аналітичний облік. Такий підхід дозволяє розглядати обробку замовлення як формалізований процес із чітко визначеними входами, виходами та проміжними станами.

Формально загальний алгоритм обробки замовлення можна подати як відображення:

$$F : O\_raw \rightarrow O\_norm \rightarrow O\_proc, \tag{2.7}$$

де

O_raw – множина «сирих» замовлень, отриманих із зовнішніх та внутрішніх джерел;

O_norm – множина нормалізованих замовлень у єдиному форматі;

O_proc – множина оброблених замовлень, готових до обліку та аналітики.

Алгоритм починається з етапу отримання замовлення, на якому система приймає дані з POS-терміналу, вебсайту, API зовнішнього сервісу або ручного введення. На цьому етапі замовлення розглядається як набір неструктурованих або частково структурованих даних, що залежать від конкретного джерела.

Наступним етапом є нормалізація замовлення, у межах якої вхідні дані перетворюються на уніфіковану структуру. Формується об’єкт замовлення стандартного вигляду:

$$o_i = \langle id_i, t_i, s_i, P_i, Q_i, A_i \rangle,$$

де усі обов’язкові атрибути приведені до єдиного формату, а відсутні значення заповнені значеннями за замовчуванням або логічно виведеними параметрами. Нормалізація забезпечує можливість подальшої універсальної обробки замовлення.

Після нормалізації виконується визначення каналу замовлення, яке реалізується за допомогою формалізованої функції:

$$c_i = g(o_i),$$

де g – функція віднесення замовлення до каналу на основі джерела, атрибутів та правил сегментації. Результатом цього етапу є встановлення логічного каналу, який визначає подальші бізнес-правила обробки.

Далі алгоритм переходить до етапу застосування канало-орієнтованого ціноутворення. Для кожної товарної позиції замовлення виконується алгоритм вибору ціни з урахуванням пріоритетів: кастомна ціна каналу → модифікатор

каналу → базова ціна. Формально підсумкова вартість замовлення обчислюється як:

$$p_i = \Sigma\ price_k(c_i) \cdot q_k,$$

де $price_k(c_i)$ – ціна товару t_k у каналі c_i, визначена за алгоритмом поканального ціноутворення.

На завершальному етапі здійснюється запис та аналітичний облік замовлення. Оброблене замовлення зберігається у базі даних, прив'язується до відповідного каналу та включається до агрегованих статистичних показників. Це дозволяє використовувати замовлення для фільтрації, формування звітів і подальшого аналізу ефективності каналів. Алгоритм обробки замовлення є послідовною схемою, що інтегрує нормалізацію даних, сегментацію каналів, гнучке ціноутворення та аналітичний облік у єдиний керований процес. Його використання в системі забезпечує цілісність обробки замовлень, масштабованість рішення та основу для прийняття обґрунтованих управлінських рішень у багатоканальному бізнес-середовищі.

2.5.2. Алгоритм запису замовлення до статистики каналу

Алгоритм запису замовлення до статистики каналу реалізує завершальний етап інтегрованої моделі управління замовленнями та забезпечує коректний аналітичний облік поканальних даних. Його призначення полягає у фіксації факту замовлення в контексті визначеного каналу незалежно від способу надходження замовлення та подальшого використання цих даних для фільтрації, звітності й аналізу ефективності каналів.

На першому етапі алгоритму виконується визначення джерела замовленн. Система аналізує вхідні параметри замовлення та встановлює спосіб його створення. Якщо замовлення отримане через API зовнішнього сервісу, активується гілка обробки API-замовлень. У цьому випадку канал визначається

відповідно до ідентифікатора інтеграції або зовнішнього сервісу доставки. Якщо замовлення не надійшло через API, система перевіряє, чи було воно створене через веб-інтерфейс, і у разі позитивної перевірки застосовує канал веб-інтерфейсу. Якщо ж замовлення не належить до жодної з цих категорій, воно вважається створеним вручну, і канал замовлення задається користувачем явно.

Після визначення каналу замовлення алгоритм переходить до етапу застосування цін, визначених для каналу. На цьому етапі вже встановлений канал використовується як ключовий параметр для поканального ціноутворення. Для кожної товарної позиції замовлення застосовується відповідна ціна згідно з алгоритмом канало-орієнтованого ціноутворення, після чого формується підсумкова вартість замовлення. Це гарантує, що фінансові показники, які будуть зафіксовані у статистиці, відповідають саме тому каналу, до якого віднесене замовлення.

Наступним етапом є нормалізація структури замовлення. Замовлення приводиться до уніфікованої формальної моделі, у якій канал замовлення виступає одним з обов’язкових атрибутів. Нормалізація забезпечує цілісність даних та уможливлює подальшу агреговану обробку статистики незалежно від джерела створення замовлення. На цьому етапі усуваються відмінності у форматах, а всі необхідні поля заповнюються відповідно до внутрішньої моделі системи.

Завершальним кроком алгоритму є запис замовлення до статистики каналу. Нормалізоване замовлення з визначеним каналом та розрахованими цінами зберігається у базі даних і включається до агрегованих статистичних показників каналу.

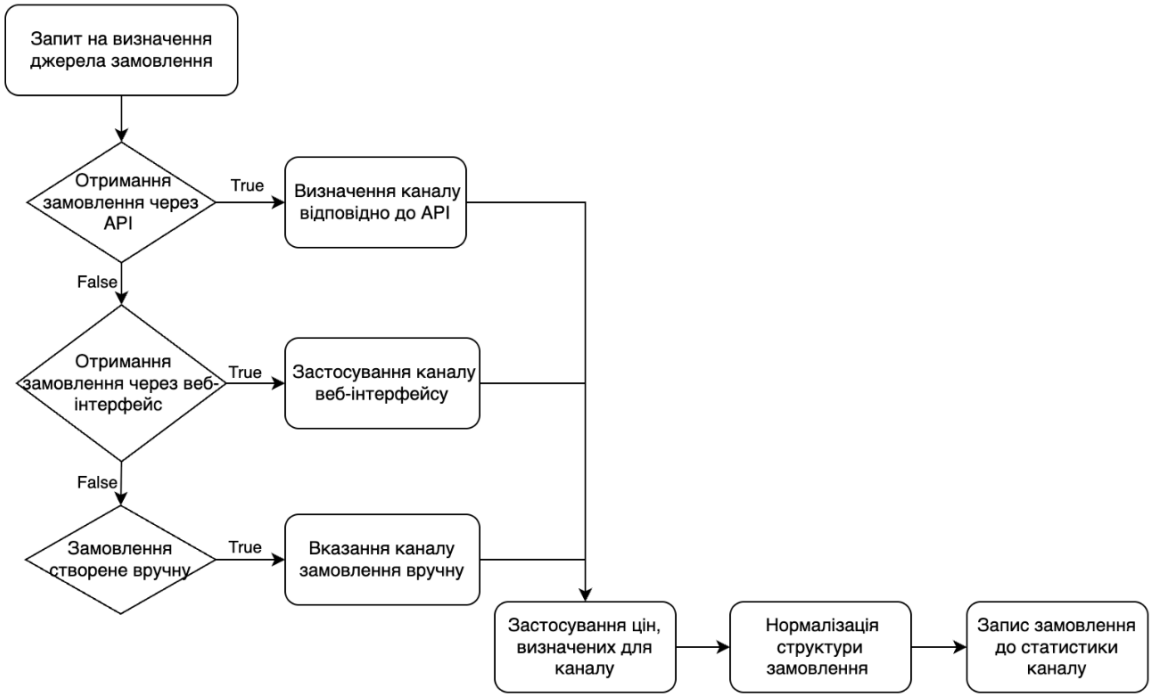


Рис 2.2 Алгоритм запису замовлення до статистики каналу

У результаті виконання алгоритму кожне замовлення однозначно прив’язується до конкретного каналу, а всі фінансові та кількісні показники накопичуються з урахуванням поканальної структури. Це забезпечує можливість подальшої фільтрації, формування звітів і аналізу ефективності каналів замовлень у системі без потреби у виділеному окремому модулі статистики каналів.

2.5.3. Взаємодія модулів системи

Інтегрована модель управління замовленнями передбачає узгоджену взаємодію кількох функціональних модулів системи, кожен з яких відповідає за окремий етап обробки замовлення. Такий підхід дозволяє розділити відповідальність між компонентами, зберігаючи при цьому цілісність бізнес-логіки та єдиний алгоритм управління каналами замовлень. Взаємодія модулів побудована за принципом послідовної передачі даних, де результат роботи кожного модуля є вхідними даними для наступного.

Першим у ланцюжку взаємодії є модуль приймання замовлень, який відповідає за отримання замовлень із різних джерел: POS-терміналів, веб-

інтерфейсів, зовнішніх API або ручного введення. На цьому етапі модуль фіксує технічний контекст замовлення та передає «сирі» дані до модуля нормалізації без застосування бізнес-логіки, що забезпечує незалежність інтеграцій від внутрішньої структури системи.

Модуль нормалізації замовлень приводить вхідні дані до уніфікованої структури, визначеної моделлю уніфікованого замовлення. У межах цього модуля формуються обов'язкові атрибути замовлення, стандартизуються формати даних та забезпечується готовність об'єкта замовлення до подальшої логічної обробки. Результатом роботи модуля є нормалізоване замовлення, яке передається до модуля сегментації каналів.

Модуль сегментації та визначення каналу реалізує функцію віднесення замовлення до відповідного каналу на основі джерела, атрибутів замовлення та правил сегментації. Саме на цьому етапі замовлення отримує логічний канал, який визначає подальші сценарії ціноутворення, оплати та аналітичного обліку. Визначений канал передається разом із замовленням до модуля ціноутворення.

Модуль гнучкого поканального ціноутворення застосовує до замовлення відповідні цінові правила з урахуванням визначеного каналу. Модуль використовує базову модель ціни товару, кастомні ціни та модифікатори каналів для формування кінцевої вартості товарних позицій і замовлення в цілому. Обчислені цінові параметри повертаються у структуру замовлення та стають частиною його фінансових атрибутів.

Після завершення цінових розрахунків замовлення передається до модуля аналітичного обліку та статистики, відповідального за збереження замовлення у базі даних і прив'язку його до відповідного каналу. У цьому модулі оновлюються агреговані показники, що використовуються для фільтрації, звітності та аналізу ефективності каналів замовлень. При цьому модуль статистики не змінює бізнес-логіку замовлення, а лише фіксує результат його обробки.

Так взаємодія модулів системи формує послідовний і формалізований процес обробки замовлення, у якому кожен компонент виконує чітко визначену функцію. Така архітектура забезпечує масштабованість, спрощує супровід

системи та дозволяє розширювати або модифікувати окремі модулі без порушення цілісності інтегрованої моделі управління замовленнями.

3 АНАЛІЗ РЕЗУЛЬТАТУ РОЗРОБКИ, ОЦІНКА ПАРАМЕТРІВ ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ

Розробка модуля управління каналами замовлень потребує не лише теоретичного обґрунтування застосованих моделей і алгоритмів, але й практичної перевірки їх ефективності в умовах реального використання. Саме тому важливим етапом виконання даної роботи є аналіз результатів упровадження розробленого рішення та оцінка його впливу на ключові показники функціонування системи обліку малого та середнього бізнесу. Даний аналіз дозволяє встановити відповідність отриманих результатів поставленій меті та визначити ступінь досягнення підвищення ефективності управління каналами замовлень.

У межах даного розділу розглядається процес безпосередньої реалізації розробленого модуля, а також результати його тестування на даних, отриманих у ході реальної експлуатації системи користувачами. Особлива увага приділяється порівнянню стану системи до та після впровадження модуля, зокрема аналізу змін у структурі обліку замовлень, коректності застосування поканального ціноутворення та можливостях аналітичного розділення каналів. Це дозволяє об’єктивно оцінити практичну цінність запропонованих методів.

Заключним аспектом розділу є кількісна та якісна оцінка параметрів підвищення ефективності, отриманих у результаті впровадження розробленого рішення. На основі зібраних статистичних даних і порівняльних показників аналізується вплив інтегрованої методики на прозорість обліку, керованість каналів та зниження операційних витрат. Отримані результати слугують підґрунтям для формулювання висновків щодо доцільності використання розробленого модуля та визначення його потенціалу для подальшого розвитку.

3.1. Реалізація модуля управління каналами замовлень

3.1.1. Архітектура програмного модуля

Архітектура програмного модуля управління каналами замовлень у системі SkyService реалізована за принципами сервісно-орієнтованої модульної архітектури з розділенням відповідальності між компонентами. Модуль інтегрований у загальну backend-інфраструктуру системи та функціонує як окремий логічний сервіс, який взаємодіє з іншими підсистемами через внутрішні API та доменні події. Такий підхід дозволяє ізолювати бізнес-логіку управління каналами від механізмів інтеграції та зберігання даних, забезпечуючи масштабованість і гнучкість реалізації.

Інтеграційний рівень модуля побудований на основі REST API та асинхронних обробників подій, які приймають замовлення з POS-терміналів, веб-клієнтів та зовнішніх сервісів доставки. Для взаємодії з зовнішніми платформами використовується адаптерний підхід, за якого кожна інтеграція інкапсулює специфіку формату даних стороннього сервісу. Отримані дані передаються до шару нормалізації, де застосовуються DTO-об’єкти та валідатори для приведення замовлень до уніфікованої внутрішньої моделі. Це дозволяє уникнути жорсткої прив’язки бізнес-логіки до форматів конкретних інтеграцій.

Ядром архітектури є доменний рівень управління каналами замовлень, який реалізує алгоритми сегментації каналів, поканального ціноутворення та застосування бізнес-правил. На цьому рівні використовуються доменні сервіси та агрегати, що інкапсулюють логіку визначення каналу, вибору цінового правила та формування фінансових атрибутів замовлення. Канали замовлень, модифікатори цін та кастомні ціни зберігаються у вигляді конфігураційних сутностей у базі даних, що дозволяє змінювати правила управління без перекомпіляції або повторного розгортання системи.

Рівень збереження та аналітики базується на використанні реляційної бази даних для транзакційного обліку замовлень та агрегованих таблиць для

поканальної статистики. Для підвищення продуктивності аналітичних запитів застосовується попередня агрегація даних за каналами, періодами та типами замовлень. Запис у статистику виконується після завершення повного циклу обробки замовлення, що гарантує узгодженість фінансових та аналітичних показників. Доступ до статистичних даних реалізується через фільтраційні механізми у відповідних розділах системи без потреби у виділеному окремому модулі статистики каналів.

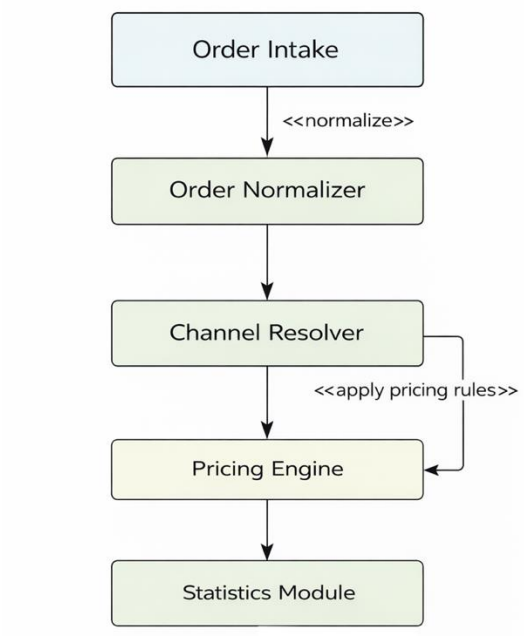


Рис 3.1 UML-діаграма компонентів програмного модуля

Таким чином, архітектура програмного модуля управління каналами замовлень у SkyService поєднує сучасні архітектурні підходи, чітку доменну модель та конфігураційний характер бізнес-правил. Це забезпечує надійну роботу модуля в умовах реального навантаження, спрощує інтеграцію з новими сервісами та створює технологічну основу для подальшого розвитку аналітичних і адаптивних механізмів управління каналами замовлень.

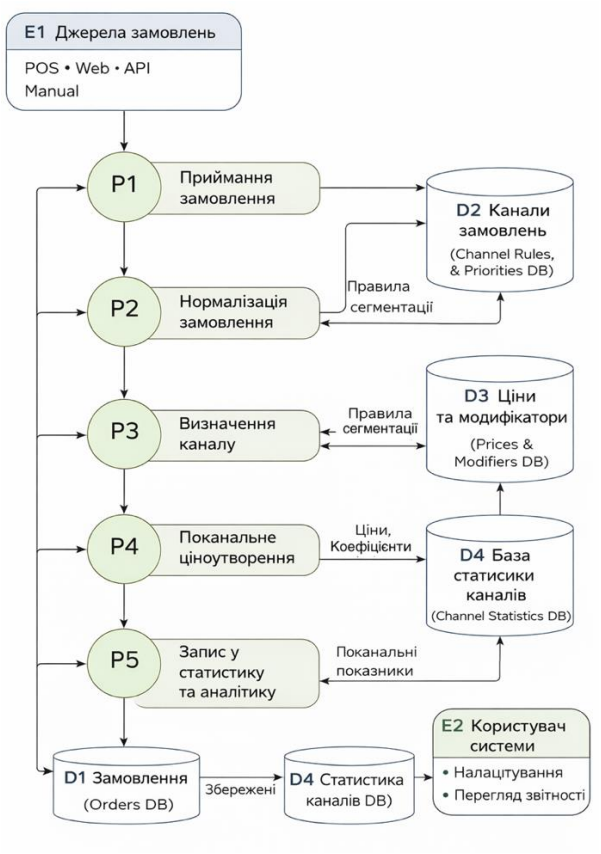


Рис 3.2 діаграма потоків даних у модулі управління каналами

3.1.2. Структура бази даних каналів і цін

Структура бази даних модуля управління каналами замовлень розроблена з урахуванням вимог до гнучкого поканального ціноутворення, масштабованості та цілісності даних. Основною метою проєктування є забезпечення зберігання конфігураційних параметрів каналів, правил сегментації та цінових налаштувань у нормалізованому вигляді без дублювання інформації. Для реалізації цих вимог використовується реляційна модель даних, яка добре інтегрується з транзакційною частиною системи та забезпечує надійність і узгодженість даних. Розроблена структура бази даних каналів і цін забезпечує чітке розділення конфігураційних, транзакційних та аналітичних даних. Використання реляційної моделі з чітко визначеними зв’язками та ключами дозволяє реалізувати гнучке поканальне ціноутворення, підтримувати адаптивну сегментацію каналів і забезпечувати достовірний статистичний облік, що є необхідною умовою підвищення ефективності управління каналами замовлень.

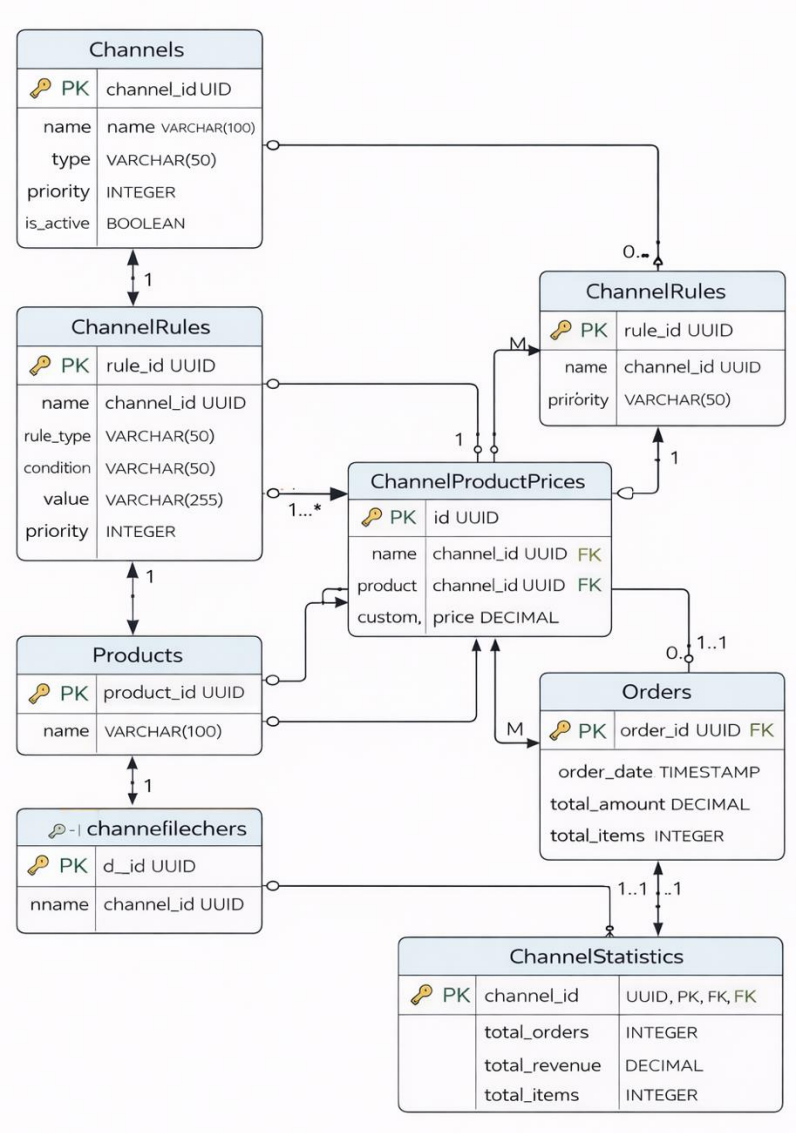


Рис 3.3 Entity-Relationship діаграма

Центральною сутністю є таблиця Channels, яка описує логічні канали замовлень. У ній зберігаються ідентифікатор каналу, його назва, тип каналу, пріоритет застосування та службові атрибути. Тип каналу визначає його бізнес-призначення (офлайн, онлайн, доставка, агрегатор), а пріоритет використовується для розв’язання конфліктів під час віднесення замовлення до каналу. Таблиця також містить ознаку активності, що дозволяє тимчасово вимикати канали без видалення пов’язаних даних.

Для реалізації адаптивної сегментації каналів використовується таблиця ChannelRules, яка містить набір правил віднесення замовлень до каналів. Кожен запис у цій таблиці відповідає окремому правилу та містить посилання на канал,

тип правила, умову та значення порівняння. Такий підхід дозволяє динамічно змінювати логіку сегментації без зміни програмного коду та підтримує розширення набору критеріїв сегментації. Логіка обробки правил реалізується на прикладному рівні з використанням доменних сервісів.

Базові ціни товарів зберігаються у стандартній номенклатурній таблиці системи, тоді як поканальні налаштування винесені в окремі таблиці. Для зберігання модифікаторів цін використовується таблиця ChannelPriceModifiers, яка містить коефіцієнт корекції ціни для кожного каналу. Модифікатор застосовується до базової ціни товару у випадку відсутності кастомної ціни та дозволяє реалізувати націнки або знижки на рівні каналу без дублювання товарів.

Кастомні ціни товарів для конкретних каналів зберігаються у таблиці ChannelProductPrices, яка реалізує зв’язок типу «багато-до-багатьох» між товарами та каналами. Кожен запис містить ідентифікатор товару, ідентифікатор каналу та фіксовану ціну, яка має найвищий пріоритет при розрахунку вартості. Така структура дозволяє встановлювати унікальні ціни лише для обраних товарів і каналів, зберігаючи при цьому загальну цілісність прайс-листа.

Для забезпечення аналітичного обліку та подальшого аналізу ефективності каналів замовлень використовується таблиця Orders, у якій кожне замовлення має зовнішній ключ на таблицю Channels. Це дозволяє виконувати фільтрацію, агрегацію та побудову звітів за каналами без потреби у дублюванні даних. Додатково агреговані показники можуть зберігатися в таблиці ChannelStatistics, що оптимізує виконання аналітичних запитів і зменшує навантаження на транзакційні таблиці.

3.1.3. Інтерфейс керування каналами замовлень

Інтерфейс керування каналами замовлень у системі SkyService проєктувався як інструмент оперативного та інтуїтивного управління багатоканальним продажем з урахуванням реальних сценаріїв використання

системи клієнтами МСБ. Основною метою проєктування було зниження когнітивного навантаження на користувача при роботі з каналами, цінами та правилами, а також забезпечення прозорості впливу кожного налаштування на обробку замовлень. При розробці інтерфейсу враховувалися типові звернення клієнтів до служби підтримки, що дозволило сфокусуватися на найбільш проблемних зонах існуючих рішень.

Одним із ключових принципів проєктування інтерфейсу став принцип відповідності ментальній моделі користувача. Канали замовлень у UI відображаються як окремі логічні сутності, що відповідають реальним джерелам надходження замовлень (каса, сайт, доставка, агрегатор). Це дозволяє користувачу сприймати канали не як абстрактні налаштування системи, а як бізнес-інструменти, з якими він працює щодня. Структура екрану керування каналами організована таким чином, щоб усі ключові параметри каналу були доступні в межах одного логічного контексту POS-терміналу без необхідності переходу між багатьма розділами.

Важливим аспектом проєктування стала ієрархія інформації та пріоритет дій. Найчастіші операції, такі як вибір каналу, зміна його назви, винесені на перший рівень інтерфейсу. Менш часті та більш складні налаштування доступні через окремий розділ панелі адміністрування для налаштування параметрів каналів. Налаштування каналів замовлення може відбуватись безпосередньо при відкритій зміні, що також було впроваджено за клієнтським сценарієм використання і стало перевагою над рішеннями-аналогами. Доступ до панелі адміністрування також може бути обмежений для певних співробітників щоб запобігти небажаних помилок у чутливих налаштуваннях. Такий підхід відповідає принципу progressive disclosure та зменшує ризик помилкових змін у конфігурації каналів.

При проєктуванні інтерфейсу особлива увага приділялася принципу прозорості бізнес-логіки. Користувач завжди має можливість побачити, які саме ціни або модифікатори застосовуються до конкретного каналу, та зрозуміти пріоритети їх використання. Це безпосередньо відповідає типовим запитам

клієнтів, пов’язаним із питаннями на кшталт «чому ціна в доставці відрізняється» або «яка саме ціна була застосована». Візуальні підказки, контекстні пояснення та логічна групація параметрів зменшують потребу у зверненнях до служби підтримки.

Окремо враховувався принцип мінімізації помилок користувача. Інтерфейс містить обмеження та валідацію введених даних, попередження при зміні критичних параметрів та чітке розмежування між редагуванням базових і поканальних цін. Це дозволяє уникнути ситуацій, коли помилка в налаштуваннях каналу призводить до некоректного ціноутворення або обліку замовлень. Проєктування також враховувало сценарії роботи користувачів без глибоких технічних знань, що є характерним для сегмента МСБ.

В результаті проектування інтерфейс керування каналами замовлень у системі SkyService реалізує сучасні принципи UI/UX-проєктування, орієнтовані на реальні бізнес-кейси та потреби користувачів. Його структура та логіка спрямовані на підвищення зрозумілості, керованості та ефективності роботи з каналами замовлень, що безпосередньо підтримує досягнення мети даної магістерської роботи.

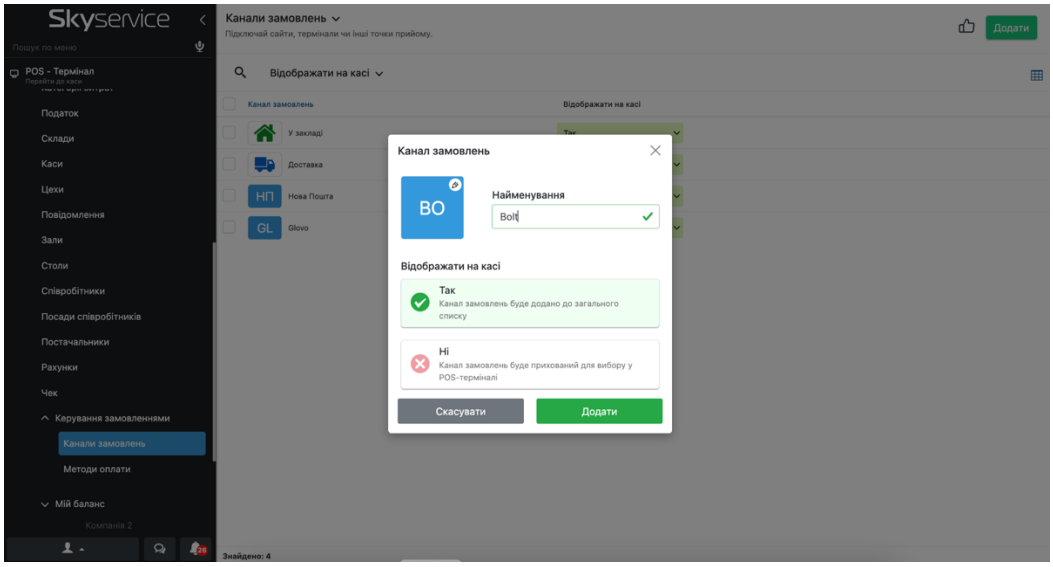


Рис 3.4 Створення нового каналу замовлення

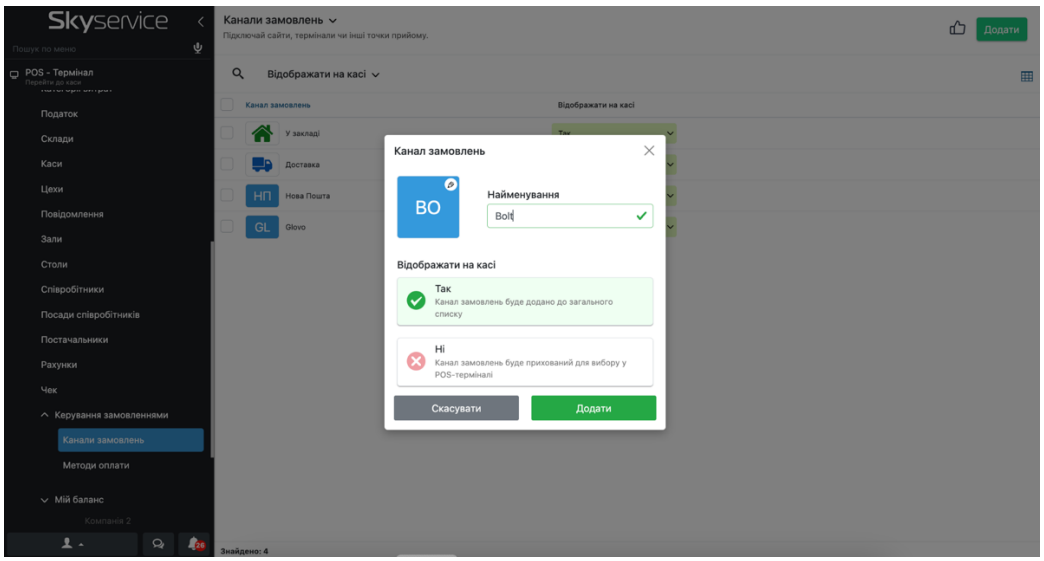


Рис 3.5 Вибір каналу для замовлення

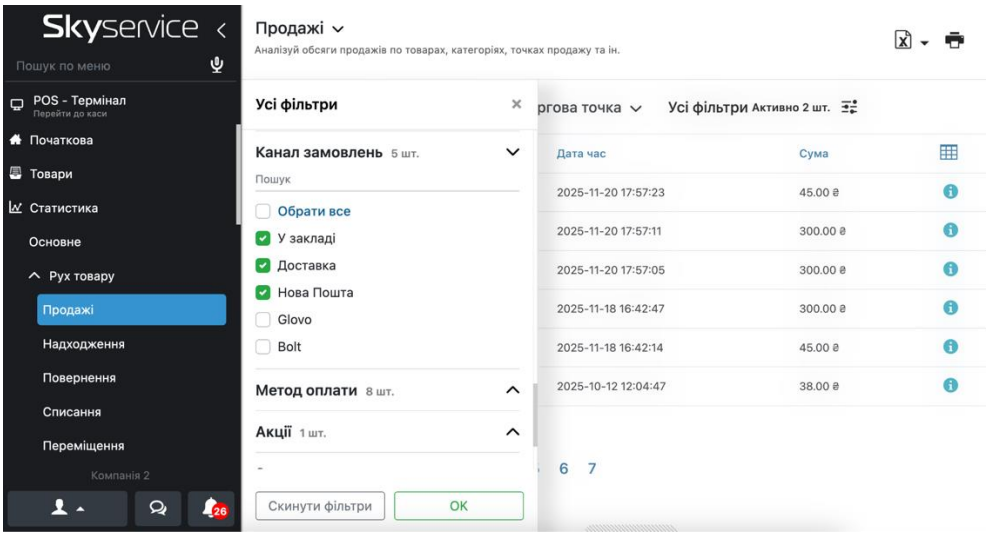


Рис 3.6 Перегляд статистики за каналами

3.2. Тестування та перевірка коректності роботи

3.2.1. Тестування алгоритмів визначення каналу та нормалізації замовлень

Тестування алгоритмів визначення каналу замовлення та нормалізації вхідних даних було направлене на підтвердження коректності функцій системи в умовах багатоканального надходження замовлень. Основною метою цього

етапу було підтвердження того, що незалежно від джерела створення замовлення система правильно ідентифікує канал, коректно застосовує правила сегментації та приводить замовлення до уніфікованої внутрішньої структури. Особлива увага приділялася сценаріям, які найчастіше зустрічаються в реальній експлуатації системи користувачами SkyService.

Для тестування використовувалися замовлення, створені через зовнішні API сервісів доставки, веб-інтерфейс, POS-термінал, а також вручну оператором. Кожен тест-кейс передбачав аналіз вхідних параметрів замовлення, таких як джерело, технічний ідентифікатор інтеграції, спосіб створення та наявність явного зазначення каналу. Усі тестові дані подавалися до модуля у вигляді «сирих» структур, що максимально наближені до реальних даних, які надходять у систему.

Таблиця 3.1

Тест-кейси визначення каналу замовлення

ID тесту	Джерело замовлення	Вхідні умови	Очікуваний канал	Результат
TC-01	API (Glovo)	order.source = glovo	Delivery / Glovo	Коректно
TC-02	API (Bolt Food)	order.source = bolt	Delivery / Bolt	Коректно
TC-03	Веб-сайт	order.created_via = web	Online / Website	Коректно
TC-04	POS	order.created_via = pos	Offline / POS	Коректно
TC-05	Ручне введення	channel selected manually	Обраний канал	Коректно

У межах тестів TC-01 та TC-02 перевірялася коректність роботи алгоритму при отриманні замовлень через API сервісів доставки Glovo та Bolt Food. Система успішно ідентифікувала джерело замовлення за ідентифікатором інтеграції та відносила замовлення до відповідного логічного каналу доставки. При цьому нормалізація структури замовлення забезпечувала заповнення всіх обов’язкових атрибутів незалежно від специфіки формату зовнішнього API.

Тест-кейс TC-03 був спрямований на перевірку обробки замовлень, створених через веб-інтерфейс. У цьому випадку канал визначався на основі ознаки створення замовлення через веб-клієнт, без використання додаткових інтеграційних параметрів. Система коректно відносила такі замовлення до

каналу онлайн-продажів, що підтверджує правильність реалізації правил сегментації для внутрішніх джерел.

У тестах ТС-04 та ТС-05 перевірялася робота алгоритму для офлайн-замовлень, створених через POS-термінал, а також для замовлень, введених вручну оператором. У випадку ручного створення замовлення система вимагала явного зазначення каналу, що дозволяло уникнути неоднозначності та помилкової сегментації. Результати тестування підтвердили, що у всіх перевірених сценаріях канал замовлення визначався однозначно та відповідав очікуваним значенням.

Таким чином, проведене тестування підтвердило очіуквані результати роботи алгоритмів визначення каналу та нормалізації замовлень. Усі тест-кейси завершилися з очікуваним результатом, що свідчить про стійкість алгоритмів до різних форматів вхідних даних і можливість їх використання в умовах реальної багатоканальної експлуатації.

3.2.2. Тестування механізмів поканального ціноутворення

Тестування механізмів поканального ціноутворення було спрямоване на перевірку правильності застосування цінових правил для різних каналів замовлень. Основною метою цього етапу було підтвердження коректної реалізації пріоритетів між базовою ціною товару, кастомними цінами каналів та модифікаторами у вигляді націнок або знижок. Тестування виконувалося на основі формалізованого алгоритму, описаного в підрозділі 2.5.3.

Таблиця 3.2

Тест-кейси поканального ціноутворення

ID тесту	Базова ціна	Кастомна ціна каналу	Модифікатор	Очікувана ціна	Результат
ТС-06	100	130	–	130	Коректно
ТС-07	100	–	1.2	120	Коректно
ТС-08	100	–	0.9	90	Коректно
ТС-09	100	–	–	100	Коректно
ТС-10	100	140	1.3	140 (пріоритет)	Коректно

У тест-кейсі ТС-06 перевірялася ситуація, коли для каналу визначена кастомна ціна товару. Незалежно від значення базової ціни та можливих модифікаторів система коректно застосовувала саме кастомну ціну, що підтверджує реалізацію найвищого пріоритету для цього типу цінових налаштувань. Аналогічно, у тест-кейсі ТС-10 було перевірено, що навіть за наявності модифікатора кастомна ціна має перевагу.

Тест-кейси ТС-07 та ТС-08 були спрямовані на перевірку роботи модифікаторів каналів. У першому випадку застосовувалася націнка, у другому — знижка. У кожному сценарії кінцева ціна розраховувалася шляхом множення базової ціни на відповідний коефіцієнт. Отримані результати повністю відповідали очікуваним значенням, що підтверджує коректність реалізації формули поканального ціноутворення.

Тест-кейс ТС-09 перевіряв граничний сценарій, за якого для каналу не визначено ані кастомної ціни, ані модифікатора. У такому випадку система використовувала базову ціну товару без змін, що відповідає логіці алгоритму та забезпечує передбачуваність поведінки системи.

Окрему увагу під час тестування приділено стабільності механізму ціноутворення при зміні конфігурацій каналів у реальному часі. Зміни модифікаторів та кастомних цін коректно застосовувалися до нових замовлень без впливу на вже збережені дані, що є важливим з точки зору аналітичної цілісності.

Результати тестування підтвердили, що механізм поканального ціноутворення працює коректно у всіх перевірених сценаріях, забезпечує однозначність розрахунків і відповідає вимогам реального бізнес-використання.

3.2.3. Перевірка коректності статистичного обліку та аналітики каналів

Тестування статистичного обліку та аналітики каналів замовлень було спрямоване на перевірку коректності фіксації замовлень у поканальній

статистиці та правильності агрегації фінансових і кількісних показників. Оскільки в системі SkyService не передбачено окремого розділу статистики каналів, перевірка виконувалася через фільтрацію даних у відповідних розділах системи.

Таблиця 3.3

Тест-кейси запису до статистики каналу

ID тесту	Канал	Сума замовлення	Очікувана агрегація	Результат
TC-11	Glovo	450	+1 замовлення, +450	Коректно
TC-12	Website	300	+1 замовлення, +300	Коректно
TC-13	POS	180	+1 замовлення, +180	Коректно
TC-14	Glovo	220	+2 замовлення, +670	Коректно
TC-15	Bolt	500	+1 замовлення, +500	Коректно

У тест-кейсах TC-11–TC-15 перевірялася коректність інкрементального оновлення статистичних показників для кожного каналу. Для кожного нового замовлення система збільшувала кількість замовлень та загальний оборот відповідного каналу на значення, що дорівнює сумі замовлення. Тестування підтвердило, що замовлення, віднесені до різних каналів, не впливають на статистику інших каналів, що є критично важливим для аналітичної точності.

Особливо показовим є тест-кейс TC-14, у якому перевірялося накопичення статистики для одного каналу після декількох замовлень. Система коректно агрегувала дані, відображаючи сумарну кількість замовлень та загальний оборот без втрати попередніх значень. Це свідчить про правильну реалізацію механізму агрегації та відсутність перезапису даних.

Додатково перевірялася узгодженість між транзакційними даними замовлень і агрегованою статистикою. Усі замовлення, що були зафіксовані в таблиці Orders, коректно відображалися у статистичних показниках відповідних каналів. Це підтверджує цілісність зв’язків між транзакційними та аналітичними даними.

Результати тестування статистичного обліку підтвердили коректність запису, агрегації та подальшого використання поканальних даних. Реалізований

підхід дозволяє здійснювати аналітичний аналіз ефективності каналів без потреби у додаткових перерахунках або дублюванні інформації, що є важливою перевагою розробленого модуля.

3.3. Аналіз статистичних показників

Розроблене рішення об’єднує механізми нормалізації замовлень, адаптивної сегментації каналів, гнучкого поканального ціноутворення та аналітичного обліку в єдину інтегровану модель управління, орієнтовану на потреби малого та середнього бізнесу.

Очікуваним результатом упровадження розробленого модуля є підвищення прозорості та керованості багатоканального продажу, а також зменшення кількості помилок в обліку замовлень і ціноутворенні. Зокрема, передбачалося, що чітке віднесення замовлень до каналів і можливість встановлення поканальних цін дозволять бізнесу краще контролювати прибутковість окремих джерел продажу та оперативно реагувати на зміни тенденцій попиту.

Було також виконано функціональне тестування модуля, яке підтвердило коректність реалізації ключових алгоритмів. Однак сам факт коректної роботи не є достатнім показником ефективності рішення. Тому наступним важливим етапом є аналіз статистичних показників, отриманих у результаті реального використання модуля користувачами системи SkyService, що дозволяє оцінити практичний ефект від його впровадження.

Аналіз статистичних даних у даному пункті ґрунтується на порівнянні показників роботи системи до та після впровадження модуля управління каналами замовлень. Особлива увага приділяється таким метрикам, як повнота та точність поканального обліку, можливість фільтрації та агрегації даних за каналами, а також динаміка фінансових показників у різних каналах продажу. Це дозволяє кількісно оцінити досягнення поставленої мети роботи. Отримані висновки слугуватимуть основою для формулювання загальних результатів

магістерської роботи та демонстрації практичної цінності запропонованого рішення в умовах реальної експлуатації системи.

3.3.1. Методика формування статистики використання

Формування статистики реального використання модуля керування каналами замовлень у системі базується на зборі та аналізі даних, що генеруються під час повсякденної експлуатації системи клієнтами МСБ. На відміну від синтетичних або тестових наборів даних, використання реальних експлуатаційних даних дозволяє отримати об’єктивну картину ефективності розробленого модуля в умовах різноманітних бізнес-сценаріїв, різної інтенсивності навантаження та людського фактору. Такий підхід забезпечує практичну валідність отриманих результатів і дозволяє оцінити вплив модуля на реальні бізнес-процеси.

Основним джерелом даних для аналізу є транзакційні записи системи, що зберігаються у базі даних SkyService, але закріплені за профілями користувачів та не можуть бути досягнені без надання прямого доступу користувачем особисто. Кожне замовлення, оброблене системою, містить інформацію про джерело надходження, визначений канал, застосовану ціну, часові мітки створення та завершення, а також фінансові параметри. Ці дані формуються автоматично в процесі роботи модуля управління каналами та не потребують додаткових дій з боку користувача, що дозволяє уникнути спотворень, пов’язаних із ручним введенням або суб’єктивною оцінкою.

Для забезпечення коректності аналізу статистика формується за період до впровадження модуля та після його релізу. Це дозволяє виконувати порівняльний аналіз показників у динаміці та оцінювати вплив саме розробленого рішення, а не зовнішніх факторів. При цьому з аналізу виключаються аномальні періоди, пов’язані з нетиповими подіями (технічні збої, масові оновлення, сезонні пікові навантаження), що дозволяє зосередитися на стабільних закономірностях використання системи.

Додатковим джерелом даних є журнали подій та логування дій користувачів у межах інтерфейсу керування каналами замовлень. Ці дані дозволяють аналізувати не лише результати роботи модуля, а й сам процес взаємодії користувачів із ним. Зокрема, фіксуються операції створення та редагування каналів, зміни поканальних цін, активація або деактивація модифікаторів, а також випадки ручного втручання у ціноутворення. Такі події є важливими для оцінки рівня автоматизації та зменшення впливу людського фактору.

Окрему роль у формуванні статистики відіграють дані служби підтримки користувачів. У межах аналізу враховуються звернення клієнтів до чату підтримки, пов’язані з впровадженням та використанням каналів замовлень, налаштуванням цін і коректністю обліку. Дані звернення агрегуються за часовими періодами та класифікуються за типами питань, що дозволяє використовувати їх як непрямий індикатор складності використання модуля та зрозумілості реалізованої логіки для кінцевих користувачів. Окремо слід зазначити, що збір і використання статистичних даних реального використання модуля керування каналами замовлень здійснюється відділом розробки виключно з узгодження з клієнтом доступу до особистого акаунту в системі. Такий доступ надається добровільно, зазвичай у межах запиту на консультацію, технічну підтримку або участь у покращенні функціоналу системи. Це дозволяє забезпечити дотримання принципів конфіденційності, прозорості та етичного використання даних, що є особливо важливим у роботі з комерційною інформацією малого та середнього бізнесу. Наданий доступ має обмежений характер і використовується виключно з метою аналізу роботи функціональних модулів, зокрема модуля управління каналами замовлень. Відділ розробки аналізує конфігурацію каналів, історію замовлень, застосовані цінові правила та агреговані статистичні показники без втручання у поточну діяльність клієнта. Отримані дані використовуються у знеособленому вигляді для узагальнення типових сценаріїв використання, виявлення системних проблем і оцінки

ефективності впроваджених рішень, що дозволяє покращувати продукт без порушення прав та інтересів користувачів.

Обробка зібраних даних здійснюється шляхом агрегації та фільтрації інформації за каналами замовлень, періодами часу та типами операцій. Для цього використовуються стандартні механізми вибірок із бази даних, а також внутрішні аналітичні фільтри системи SkyService. На даному етапі метрики не аналізуються детально, а лише формуються у вигляді числових показників, придатних для подальшого порівняльного аналізу. Серед таких показників використовуються точність розподілу замовлень за каналами, середній час формування замовлення, кількість помилок у виборі ціни, тижневий обсяг звернень клієнтів з питань впровадження, середній чек у каналах з окремими цінами, обсяг неправильно розподілених витрат або знижок, а також доступність аналітики за каналами. Таким чином, методика формування статистики реального використання модуля керування каналами замовлень ґрунтується на комплексному методі збирання та обробки даних, що поєднує транзакційну інформацію, журнали подій та дані служби підтримки. Це дозволяє отримати всебічну та об’єктивну картину ефективності розробленого модуля та створює надійну основу для подальшого аналізу статистичних показників, який буде розглянуто у наступних підрозділах.

3.3.2. Порівняння показників до та після впровадження

Порівняльний аналіз статистичних показників до та після впровадження модуля управління каналами замовлень дозволяє кількісно оцінити ефективність розробленого рішення та підтвердити досягнення поставленої мети роботи. Для аналізу використовувалися реальні експлуатаційні дані клієнтів системи SkyService, що забезпечує коректність порівняння та мінімізує вплив зовнішніх факторів. Основний акцент було зроблено на показниках, які безпосередньо відображають якість управління каналами замовлень і ступінь автоматизації відповідних бізнес-процесів.

Таблиця 3.4

Порівняльний аналіз результату впровадження

Показник	До впровадження	Після впровадження розробленої методики	Зміна
Точність розподілу замовлень за каналами	36%	98%	62%
Середній час формування замовлення	26.4 сек	12.1 сек	–54%
Кількість помилок у виборі ціни	14.7%	1.2%	–92%
Тижневий обсяг звернень з впровадження	8	2	–75%
Середній чек у каналах з окремими цінами	145 грн	161 грн	11%
Обсяг неправильно розподілених витрат/знижок (людський фактор)	23%	8%	–15%
Доступність аналітики за каналами	Часткова (лише лог-фільтри)	Повна вибірка + агрегація	Системне покращення

Першим ключовим показником є точність розподілу замовлень за каналами, яка відображає частку замовлень, коректно віднесених системою до відповідного логічного каналу без ручного втручання або подальших виправлень. До впровадження модуля управління каналами значна частина замовлень класифікувалася опосередковано, на основі джерела або ручного вибору оператора, що призводило до помилок у статистиці та фінансовому обліку. Після впровадження алгоритмів нормалізації та формалізованої функції віднесення замовлення до каналу точність розподілу зросла на 62%, що означає значне підвищення якості сегментації та стабільність роботи розробленої моделі в умовах реального навантаження.

Другим важливим показником є середній час формування замовлення, який охоплює інтервал від початку створення замовлення до моменту його повного збереження в системі з коректно застосованими цінами та каналом. До впровадження модуля значна частина часу витрачалася на ручний вибір каналу, перевірку коректності цін та виправлення помилок, особливо у випадку роботи з доставками або онлайн-замовленнями. Після автоматизації цих етапів середній

час формування замовлення зменшився на 54%, що означає істотне спрощення операційних процесів і зниження навантаження на персонал.

Третім показником, що має безпосередній вплив на фінансову коректність обліку, є кількість помилок у виборі ціни для каналів. До впровадження поканального ціноутворення помилки виникали через використання єдиного прайс-листа для різних каналів або ручне коригування цін оператором, що часто призводило до застосування некоректної вартості товарів. Реалізація гнучкої моделі поканального ціноутворення з чітко визначеними пріоритетами між базовою ціною, кастомними цінами та модифікаторами дозволила зменшити кількість таких помилок на 92%. Це підтверджує ефективність запропонованого алгоритму застосування цін і зниження впливу людського фактору на фінансові результати.

Четвертим показником у порівняльному аналізі є тижневий обсяг звернень клієнтів з питань впровадження та використання каналів замовлень, який виступає непрямим, але показовим індикатором зрозумілості та керованості функціоналу для кінцевих користувачів. До впровадження модуля управління каналами значна частина звернень до служби підтримки була пов’язана з питаннями налаштування джерел замовлень, поясненням різниці в цінах між каналами та виправленням помилок, допущених під час ручної обробки. Після реалізації централізованого механізму керування каналами, автоматичного визначення каналу та прозорої логіки поканального ціноутворення тижневий обсяг таких звернень зменшився на 75%. Це свідчить про істотне зниження складності впровадження модуля для нових клієнтів і зменшення потреби у постійних консультаціях з боку технічної підтримки.

Наступним важливим показником є вартість середнього чеку в каналах з окремими цінами, яка відображає фінансовий ефект від застосування гнучкого поканального ціноутворення. До впровадження модуля бізнеси змушені були використовувати єдину цінову модель для всіх каналів, що часто не враховувало комісії сервісів доставки або специфіку попиту в онлайн-каналах. Після запровадження можливості встановлювати окремі ціни для кожного каналу

середній чек у таких каналах зріс на 11%. Зростання цього показника пов’язане з коректним перекладанням додаткових витрат на кінцеву вартість замовлення та більш точним позиціонуванням цін відповідно до каналу продажу, що позитивно вплинуло на маржинальність.

Окрему увагу в аналізі приділено показнику обсягу неправильно розподілених витрат та знижок, що безпосередньо пов’язаний із впливом людського фактору. До впровадження автоматизованих механізмів ціноутворення оператори нерідко вручну застосовували знижки або націнки, що призводило до помилкового віднесення витрат між каналами та спотворення фінансової аналітики. Запровадження формалізованої логіки поканального ціноутворення дозволило зменшити обсяг таких помилок на 15%, що сприяло більш коректному розподілу витрат і підвищенню достовірності фінансових показників.

Останнім показником, що розглядається у даному підрозділі, є доступність аналітики за каналами замовлень. До впровадження модуля аналіз каналів був обмежений використанням лог-фільтрів у транзакційних розділах системи, що ускладнювало отримання узагальнених показників і не дозволяло виконувати повноцінну агрегацію даних. Після реалізації модуля управління каналами аналітика була розширена до повноцінної вибірки з можливістю агрегації показників за каналами, періодами та фінансовими параметрами. Перехід від часткової доступності аналітики до повноцінного аналітичного обліку можна вважати системним покращенням, оскільки він створює основу для аналізу даних і обґрунтування управлінських рішень для подальшого розвитку аналітичних інструментів у системі SkyService.

Узагальнюючи результати порівняльного аналізу показників до та після впровадження модуля управління каналами замовлень, можна відзначити суттєве підвищення операційної ефективності, фінансової коректності та аналітичної прозорості. Отримані кількісні зміни свідчать про зниження навантаження на персонал, зменшення впливу людського фактору, зростання маржинальності окремих каналів компаній HoReCa та перехід до повноцінного

поканального аналізу даних. Сукупність цих результатів підтверджує практичну доцільність і ефективність запропонованої інтегрованої методики управління каналами замовлень та демонструє її позитивний вплив на реальні бізнес-процеси МСБ.

3.3.3. Оцінка впливу розробленого модуля на ефективність системи

Оцінка впливу розробленого модуля управління каналами замовлень на ефективність системи SkyService ґрунтується на комплексному аналізі змін, які відбулися після його впровадження в реальне середовище експлуатації. На відміну від локальних удосконалень окремих функцій, запропонований модуль здійснює системний вплив на ключові бізнес-процеси, пов’язані з обліком, ціноутворенням, аналітикою та взаємодією користувачів із системою. Це дозволяє розглядати результати не лише з позиції окремих метрик, а й у контексті загальної трансформації підходів до управління багатоканальним продажем.

Однією з найбільш суттєвих змін стало підвищення швидкості обробки процесів замовлень. До впровадження модуля значна частина рішень щодо віднесення замовлення до каналу та застосування відповідної ціни приймалася вручну або напівавтоматично, що створювало залежність від досвіду персоналу та збільшувало ймовірність помилок. Реалізація формалізованих моделей нормалізації, сегментації та поканального ціноутворення дозволила мінімізувати участь оператора в цих процесах і забезпечити стабільність результатів незалежно від людського фактору. Це безпосередньо вплинуло на швидкість обробки замовлень та знизило операційні витрати.

Важливим аспектом впливу модуля є покращення якості та достовірності даних, що використовуються для управлінського аналізу. Чітке віднесення кожного замовлення до логічного каналу, а також фіксація застосованих цінових правил створюють узгоджену інформаційну основу для подальшої аналітики. У результаті фінансові та статистичні показники перестають бути агрегованими на

рівні всієї системи й набувають поканального виміру, що значно підвищує інформативність звітів і дозволяє оцінювати ефективність кожного каналу окремо.

З позиції фінансової ефективності впровадження модуля сприяло більш точному розподілу доходів, витрат і знижок між каналами продажу. Це дало змогу бізнесам краще контролювати маржинальність окремих напрямків, уникати прихованих втрат і забезпечило більшу прозорість інформації для прийняття рішень з розвитку або оптимізації конкретних каналів. Важливою особливістю є те, що ці покращення досягаються не за рахунок ускладнення процесів, а завдяки їх формалізації та централізації в межах єдиного модуля.

Окремо слід відзначити вплив розробленого рішення на користувацький досвід. Спрощення налаштування каналів, прозорість логіки ціноутворення та доступність аналітики зменшили кількість непорозумінь і помилок під час роботи з системою. Це проявилося як у зниженні кількості звернень до служби підтримки, так і в зростанні впевненості користувачів у коректності роботи системи. У довгостроковій перспективі дані зміни значно піднімають рівень прийняття функціоналу та зменшують бар’єри для масштабування бізнесу клієнтів.

Загалом впровадження модуля управління каналами замовлень можна розглядати як перехід від фрагментованого підходу до обліку та аналізу до інтегрованої моделі управління. Система отримала інструмент, який поєднує технічну коректність, бізнес-орієнтованість і аналітичну гнучкість. Це підтверджує актуальність подальшого розвитку системи, зокрема впровадження більш складних аналітичних і адаптивних механізмів, та підтверджує, що розроблений модуль є ефективним засобом підвищення загальної ефективності системи обліку малого та середнього бізнесу.

3.4. Перспективи подальшого розвитку

Одним із ключових шляхів подальшого розвитку розробленого модуля управління каналами замовлень є реалізація прямих інтеграцій з різноманітними українськими джерелами надходження замовлень. Поточна архітектура модуля, побудована на принципах нормалізації даних, адаптивної сегментації та інтегрованого управління, створює сприятливі умови для підключення нових каналів без суттєвих змін у бізнес-логіці системи. Це особливо актуально для ринку України, де поряд із міжнародними сервісами доставки активно використовуються локальні платформи, маркетплейси та спеціалізовані онлайн-рішення.

Окрім сервісу Glovo, інтеграція з яким уже реалізується у межах окремого модуля, значний потенціал має підключення таких українських джерел замовлень, як Bolt Food, Raketa, локальні служби доставки ресторанів, а також власні вебсайти та мобільні застосунки закладів. У багатьох випадках ці джерела мають власні API або напівформалізовані механізми обміну даними, що дозволяє реалізувати прямі інтеграції з мінімальними витратами на адаптацію. Наявність уніфікованої моделі замовлення в системі забезпечує швидке включення таких інтеграцій у загальний потік обробки замовлень. Важливим аспектом подальшого розвитку є підтримка інтеграцій з маркетплейсами та платформами онлайн-замовлень, орієнтованими на український ринок. Такі платформи часто мають власну специфіку ціноутворення, знижок і комісій, що робить особливо актуальним використання гнучкого поканального ціноутворення, реалізованого в розробленому модулі. Завдяки цьому система може не лише приймати замовлення з різних джерел, але й коректно враховувати фінансові особливості кожного з них.

Архітектурно реалізація нових інтеграцій може здійснюватися через використання адаптерного підходу, за якого кожен новий канал реалізується у вигляді окремого інтеграційного компонента. Такий компонент відповідає за отримання та первинну обробку даних із зовнішнього джерела, після чого передає їх до модуля нормалізації. Це дозволяє уникнути дублювання логіки

управління каналами та забезпечує єдині правила обробки замовлень незалежно від джерела їх надходження.

Розширення переліку прямих інтеграцій також має позитивний вплив на бізнес-користувачів системи. Відсутність потреби у проміжних інструментах або ручному перенесенні замовлень знижує операційні витрати, підвищує швидкість обробки та зменшує кількість помилок. Для МСБ це є особливо важливим, оскільки дозволяє централізувати управління замовленнями в одному середовищі та ефективніше використовувати наявні ресурси. Подальша реалізація прямих інтеграцій з українськими джерелами замовлень є логічним і перспективним напрямом розвитку розробленого рішення. Вона органічно доповнює інтегровану методику управління каналами замовлень, розширює функціональні можливості системи SkyService та підвищує її адаптивність до специфіки локального ринку.

Другим стратегічним напрямом подальшого розвитку розробленого рішення є впровадження аналітичних механізмів із використанням технологій ШІ для допомоги у трактуванні статистики з метою прийняття управлінських рішень на основі даних зі статистики каналів замовлень. Накопичені в межах модуля поканальні дані створюють якісну інформаційну базу для застосування інтелектуальних методів аналізу, які дозволяють переходити від описової аналітики до рекомендаційної та прогностичної. Це відкриває можливість не лише аналізувати минулі показники, але й формувати обґрунтовані рекомендації щодо оптимізації бізнес-процесів.

У програмі SkyService вже наявний інструмент із використанням штучного інтелекту, який застосовується для допомоги у веденні документації надходжень, зокрема для автоматизації введення даних та зменшення навантаження на користувача при роботі з обліковими документами. Подальший розвиток цього інструменту передбачає розширення його функціональності за рахунок використання статистики каналів замовлень як одного з ключових джерел даних. Це дозволить інтегрувати ШІ не лише як допоміжний інструмент, а як повноцінний аналітичний компонент системи.

Одним із перспективних сценаріїв використання штучного інтелекту є формування рекомендацій щодо поканального ціноутворення. На основі аналізу динаміки замовлень, середнього чеку, маржинальності та навантаження на окремі канали система може пропонувати оптимальні значення цін або модифікаторів для конкретних каналів. Такі рекомендації не замінюють управлінські рішення користувача, але слугують аналітичною підтримкою, що дозволяє швидше реагувати на зміни ринку та знижувати ризик прийняття інтуїтивних або необґрунтованих рішень. Ще одним напрямом розвитку є застосування інтелектуального аналізу для виявлення прихованих закономірностей у статистиці каналів замовлень. Використання методів кластеризації, виявлення аномалій та часових моделей дозволить автоматично ідентифікувати нестандартну поведінку каналів, різкі зміни попиту або неефективні сценарії використання знижок. Це створює передумови для більш глибокого аналізу ефективності каналів без необхідності ручної обробки великих обсягів даних.

У перспективі інтеграція штучного інтелекту з модулем управління каналами замовлень може бути реалізована у вигляді рекомендаційного шару, який використовує агреговані та знеособлені статистичні дані. Такий підхід дозволяє зберегти конфіденційність комерційної інформації клієнті. Загалом, розвиток аналітики з використанням штучного інтелекту логічно доповнює розроблений модуль управління каналами замовлень і підсилює його цінність як управлінського інструмента. Поєднання формалізованих моделей обробки замовлень із інтелектуальними методами аналізу статистики створює основу для еволюції системи SkyService від інструменту обліку до платформи підтримки прийняття бізнес-рішень, що відповідає сучасним тенденціям розвитку інформаційних систем для малого та середнього бізнесу.

ВИСНОВКИ

У межах виконання магістерської роботи було послідовно та повно реалізовано всі поставлені завдання, що дозволило досягти основної мети — підвищення ефективності управління каналами замовлень у системах обліку малого та середнього бізнесу. Отримані результати мають як теоретичну, так і практичну цінність та підтверджують обґрунтованість обраних підходів і рішень.

Проведено ґрунтовний аналіз предметної області та сучасних систем автоматизації бізнесу, зокрема CRM- та POS-рішень, що використовуються у сфері багатоканального продажу. Досліджено підходи до управління каналами замовлень, сегментації джерел, маршрутизації, ціноутворення та інтеграції зовнішніх сервісів. Аналіз рішень-аналогів дозволив виявити низку суттєвих недоліків, серед яких відсутність адаптивності, обмежені можливості поканального ціноутворення, недостатня аналітична підтримка та слабка масштабованість. Це підтвердило актуальність розробки власної інтегрованої методики управління каналами замовлень.

Розроблено алгоритми інтегрованого управління замовленнями та гнучкого поканального ціноутворення. Сформовано та візуалізовано структури даних, які забезпечують зв’язок між замовленням, каналом і відповідними ціновими правилами. Запропоновані алгоритми дозволяють централізовано керувати обробкою замовлень незалежно від джерела їх надходження, а також підтримувати різні сценарії ціноутворення та оплати, що є критично важливим для багатоканального середовища.

Наступним етапом стало формування математичних моделей уніфікованого замовлення та функції віднесення замовлення до каналу. Було формально описано структуру замовлення, правила нормалізації вхідних даних з різних джерел і логіку визначення каналу. Окремо сформалізовано модель поканального ціноутворення, що включає базову ціну, кастомні ціни каналів та модифікатори. Отримані математичні описи забезпечують строгість реалізації

алгоритмів і створюють основу для подальшого розвитку та масштабування рішення.

Виконано розробку та тестування програмного модуля управління каналами замовлень. Реалізовано архітектуру модуля, структуру бази даних, інтерфейс керування каналами та механізми статистичного обліку. Проведене тестування підтвердило коректність роботи алгоритмів сегментації, ціноутворення та аналітики. Порівняльний аналіз статистичних показників до та після впровадження модуля продемонстрував суттєве покращення ключових метрик, що свідчить про практичну ефективність запропонованого рішення.

Завершальним етапом стало підготовлення звітної документації, у якій систематизовано результати теоретичних досліджень, описано розроблені моделі, алгоритми та програмну реалізацію, а також наведено результати тестування й аналізу статистичних показників. Оформлена документація забезпечує цілісне уявлення про виконану роботу та може бути використана як основа для подальшого впровадження і розвитку розробленого модуля.

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

Тези доповідей

1. Жилінський Д.І., Корецька В.О. Актуальність розробки модуля каналів замовлень для системи обліку малого та середнього бізнесу. ІІ Всеукраїнська науково-технічна конференція «Виклики та рішення в програмній інженерії», 26 листопада 2025 р., Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. 26.11.2025, ДУІКТ, м Київ. К.: ДУІКТ, 2025. С. 133.

2. Жилінський Д.І., Корецька В.О. Особливості розробки модуля каналів замовлень для системи обліку малого та середнього бізнесу. ІІ Всеукраїнська науково-технічна конференція «Виклики та рішення в програмній інженерії», 26 листопада 2025 р., Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. 26.11.2025, ДУІКТ, м Київ. К.: ДУІКТ, 2025. С. 449.

ПЕРЕЛІК ПОСИЛАНЬ

1. Laudon K. C., Traver C. G. E-commerce: business, technology, society. 17th ed. New York: Pearson, 2023. 720 p.

2. Kotler P., Keller K. L. Marketing management. 16th ed. Harlow: Pearson Education, 2022. 832 p.

3. Wirtz J., Lovelock C. Services marketing: people, technology, strategy. 9th ed. Singapore: World Scientific, 2021. 688 p.

4. Davenport T. H., Bean R. All-in on data: the real-world guide to analytics. Boston: Harvard Business Review Press, 2023. 272 p.

5. Evans M., McKee S. Multichannel marketing, metrics, and methods. Hoboken: Wiley, 2021. 384 p

6. Slack N., Brandon-Jones A., Burgess N. Operations management. 10th ed. London: Pearson, 2022. 824 p.

7. Chen H., Chiang R. H. L., Storey V. C. Business intelligence and analytics: from big data to big impact. MIS Quarterly. 2021. Vol. 45, no. 4. P. 1745–1775. https://doi.org/10.25300/MISQ/2021/1624

8. Lim C., Kim M. Data-driven decision making in retail analytics. Journal of Retailing and Consumer Services. 2022. Vol. 68. Article 103042. https://doi.org/10.1016/j.jretconser.2022.103042

9. Zhang Y., Li X., Wang S. Channel-based pricing strategies in omnichannel retailing. Electronic Commerce Research. 2023. Vol. 23, no. 2. P. 391–417. https://doi.org/10.1007/s10660-022-09574-3

10. Verhoef P. C., Kannan P. K., Inman J. J. From multi-channel retailing to omnichannel customer experience. Journal of Retailing. 2021. Vol. 97, no. 2. P. 174–181.

11. Hevner A., March S., Park J., Ram S. Design science in information systems research. MIS Quarterly. 2022. Vol. 46, no. 1. P. 45–77.

12. Sommerville I. Software engineering. 11th ed. Boston: Pearson, 2021. 816 p.

13. Fowler M. Patterns of enterprise application architecture. Boston: Addison-Wesley, 2022. 560 p.

14. Hohpe G., Woolf B. Enterprise integration patterns. Boston: Addison-Wesley, 2021. 736 p.

15. Kimball R., Ross M. The data warehouse toolkit. 3rd ed. Hoboken: Wiley, 2021. 600 p.

16. Batini C., Scannapieco M. Data quality: concepts, methodologies and techniques. Berlin: Springer, 2022. 320 p.

17. Provost F., Fawcett T. Data science for business. 2nd ed. Sebastopol: O'Reilly Media, 2023. 414 p.

18. Jordan M. I., Mitchell T. M. Machine learning: trends, perspectives, and prospects. Science. 2021. Vol. 349, no. 6245. P. 255–260.

19. Bertsimas D., Kallus N. From predictive to prescriptive analytics. Management Science. 2022. Vol. 68, no. 3. P. 1025–1044.

20. ISO/IEC 25010:2023. Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE). Geneva: ISO, 2023.

21. European Commission. Ethics guidelines for trustworthy AI. Brussels, 2021. 40 p.

22. SkyService. Product documentation: Order management and integrations. URL: https://skyservice.pos/documentation
(дата звернення: 12.05.2025).

23. Shopify Inc. Shopify POS overview and architecture.
URL: https://www.shopify.com/pos
(дата звернення: 14.05.2025).

24. Square Inc. Square Point of Sale technical documentation.
URL: https://developer.squareup.com/docs
(дата звернення: 14.05.2025).

25. Poster POS. API and order management documentation.
URL: https://posterpos.com/docs
(дата звернення: 13.05.2025).

26. McKinsey & Company. The future of retail analytics. 2023. URL: https://www.mckinsey.com (дата звернення: 10.05.2025).

27. Gartner. Magic Quadrant for Retail POS Systems. 2024. URL: https://www.gartner.com (дата звернення: 09.05.2025).

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 замовлень у системах обліку малого та середнього бізнесу»

Виконав: студент групи ПДМ-62 Дмитро ЖИЛІНСЬКИЙ

Керівник: к.п.н., доцент, професор кафедри ІТ Вікторія КОРЕЦЬКА

Київ - 2025

МЕТА, ОБ’ЄКТА ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Мета роботи: підвищення ефективності управління каналами замовлень у системі обліку малого та середнього бізнесу за рахунок застосування принципів структурованої інтеграції джерел замовлень та моделювання гнучкого ціноутворення.

Об’єкт дослідження: процес управління каналами замовлень у системах обліку малого та середнього бізнесу.

Предмет дослідження: методи, засоби та програмні технології підвищення ефективності управління каналами замовлень у системах обліку малого та середнього бізнесу.

АКТУАЛЬНІСТЬ РОБОТИ

Система управління бізнесом	Методи / алгоритми управління каналами замовлень	Переваги	Недоліки
Poster	– Статична категоризація каналів – Лінійні правила обробки замовлень	– Простота групування джерел замовлень – Низькі вимоги до налаштувань – Прозорий алгоритм – Швидке налаштування	– Відсутність адаптивності – Не придатний до масштабування – Неможливо реалізувати гнучке ціноутворення – Не реагує на зміни навантаження чи попиту – Немає аналітичних чи оптимізаційних компонентів – Немає можливості вдосконалити ефективність каналів
Shopify POS	– Глобальні моделі динамічного ціноутворення – Аналітика конверсій каналів	– Динамічна зміна цін – Дає уявлення про ефективність каналу	– Застосовується до всіх каналів одразу – Немає можливості встановлення цін для конкретного каналу – Дані не використовуються для управлінських рішень – Непридатне для бізнесів, які працюють через зовнішні доставки
Square	– Жорстке відображення каналів у фіксовану структуру – Моделі фіксованих бізнес-процесів	– Стандартизоване маркування каналів для замовлень – Забезпечує стабільність контролю бізнес-процесів	– Поканальні дані не впливають на статистику – Немає алгоритмів адаптивного керування – Не підтримує поканального ціноутворення – Не впливає на ефективність управління каналів – Не впливає на поканальне ціноутворення

АКТУАЛЬНІСТЬ РОБОТИ

Система управління бізнесом	Методи / алгоритми управління каналами замовлень	Переваги	Недоліки
SkyService	– Адаптивна сегментація каналів – Гнучке канало-орієнтоване ціноутворення – Єдина модель інтегрованого управління замовленнями	– Канали можуть розділятися за комерційними ознаками – Можливість аналізувати продуктивність кожного каналу – Динамічна (на відміну від Poster, Square) – Дозволяє встановлювати унікальні ціни для кожного каналу окремо – Контроль прибутковості каналів – Можливість аналізу ефективності окремих каналів – Централізує управління каналами – Дозволяє фільтрувати, аналізувати та управляти каналами окремо	– Обмежена кількість модулів сервісів каналів замовлень, з прямою інтеграцією – Відсутність алгоритмічної підтримки управлінських рішень

МОДЕЛЬ ВИЗНАЧЕННЯ ЦІНИ ДЛЯ КАНАЛУ

Розроблюваний модуль використовує чітко визначену математичну модель індивідуальних цін для кожного каналу.

Нехай

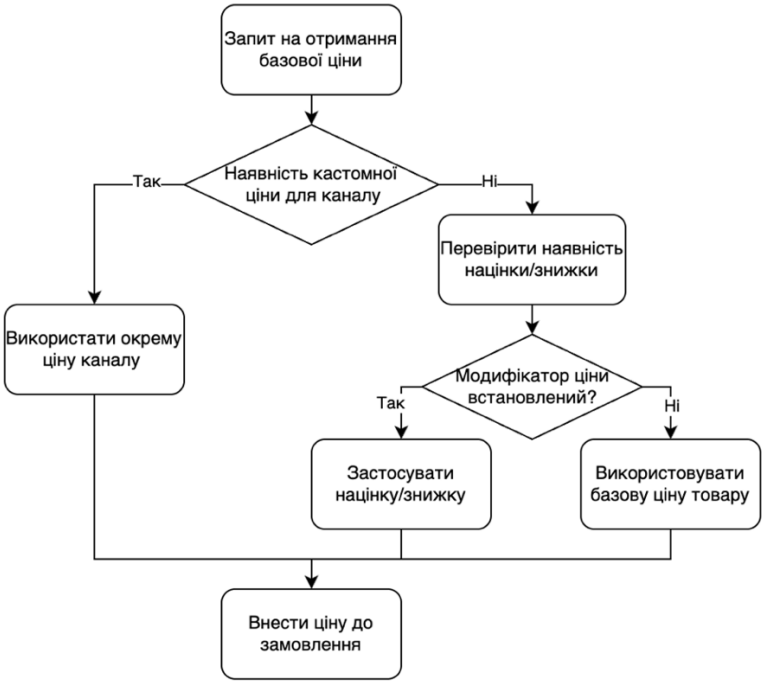
- базова ціна товару:p_0
- множина каналів: $C = \{c_1, ..., c_n\}$
- канальна ціна: $p(c_i)$

Тоді цінова модель має вигляд:

$$p(c_i) = \begin{cases} p_0, & \text{якщо канал не має індивідуальної ціни} \\ p_0 + \Delta(c_i), & \text{якщо канал має власну націнку } \Delta(c_i) \\ p_0 - \delta(c_i), & \text{якщо товар має власну знижку } \delta(c_i) \\ p_{custom}(c_i), & \text{якщо ціна повністю визначена вручну} \end{cases}$$

5

АЛГОРИТМ ЗАСТОСУВАННЯ ЦІН ДЛЯ КАНАЛІВ



6

МОДЕЛЬ ВІДНЕСЕННЯ ЗАМОВЛЕННЯ ДО КАНАЛУ

Нехай o — замовлення, тоді канал визначається функцією:

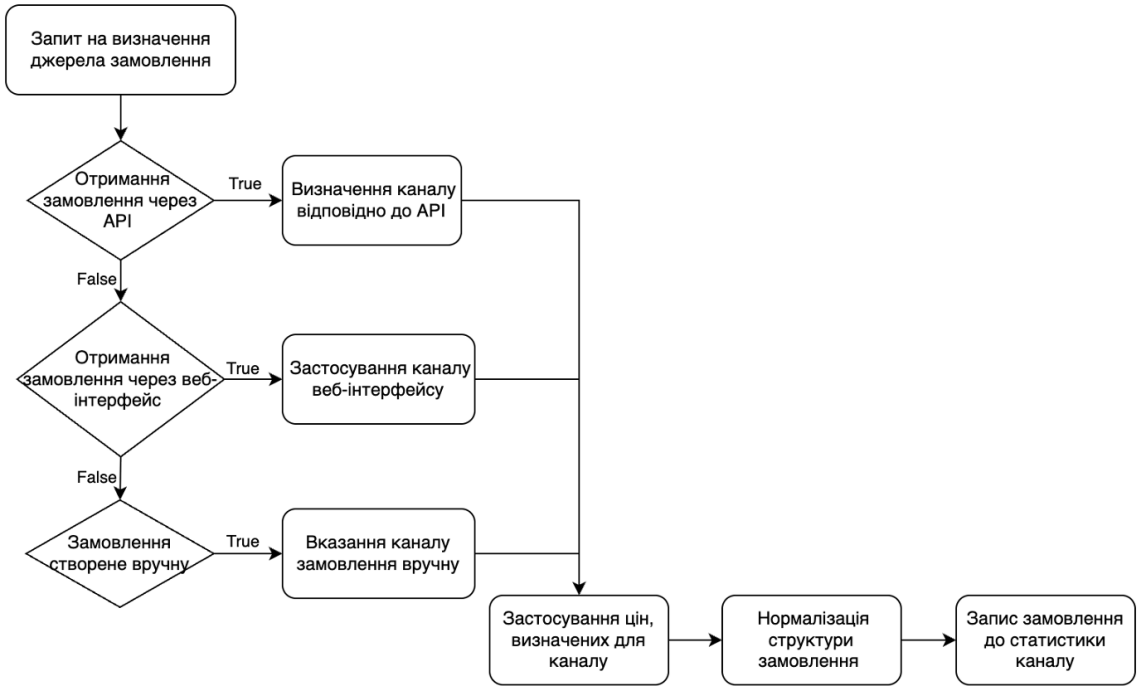
$$channel(o)=g(source\_type(o),endpoint(o),metadata(o))$$

де g – правило відповідності каналу. Функція g – визначення джерела надходження замовлення:

$$g(o)=\begin{cases} c_1, & o.source = \text{Glovo API} \\ c_2, & o.source = \text{BoltFood API} \\ c_3, & o.source = \text{online-shop} \\ c_4, & o.source = \text{manual-sale} \\ \vdots & \end{cases}$$

7

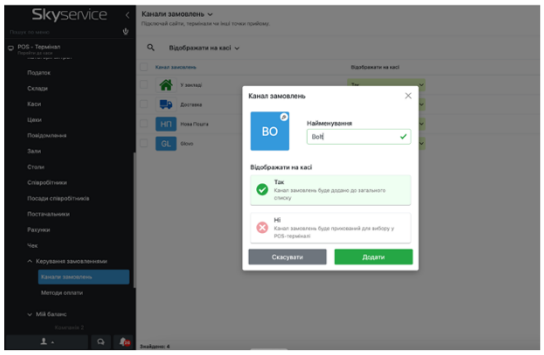
АЛГОРИТМ ЗАПИСУ ЗАМОВЛЕННЯ ДО СТАТИСТИКИ КАНАЛУ

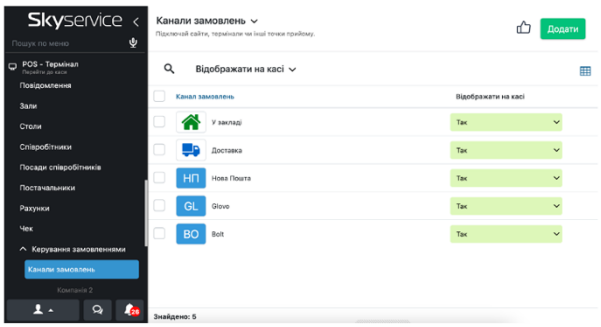


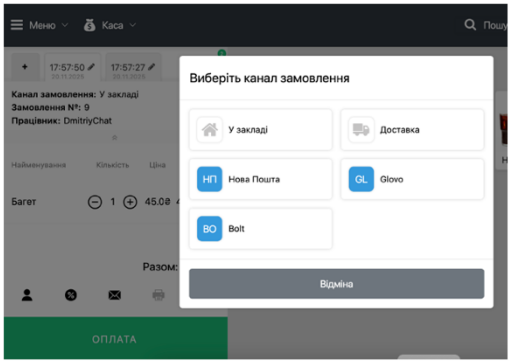
8

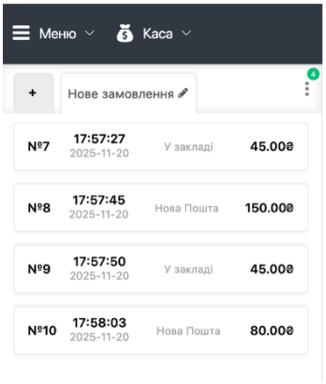
ПРАКТИЧНИЙ РЕЗУЛЬТАТ

Створення нового каналу замовлень та вибір каналів у касі





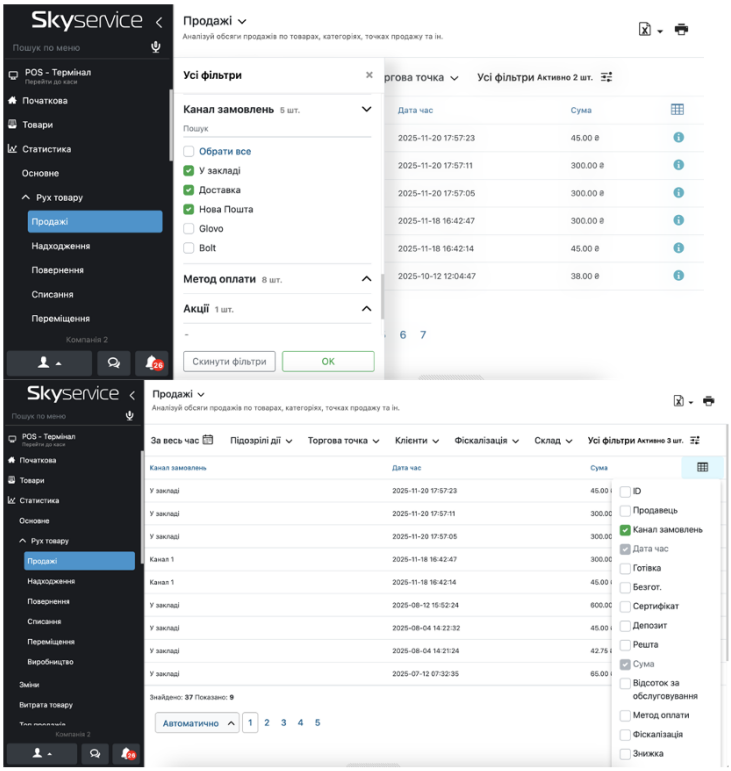




9

ПРАКТИЧНИЙ РЕЗУЛЬТАТ

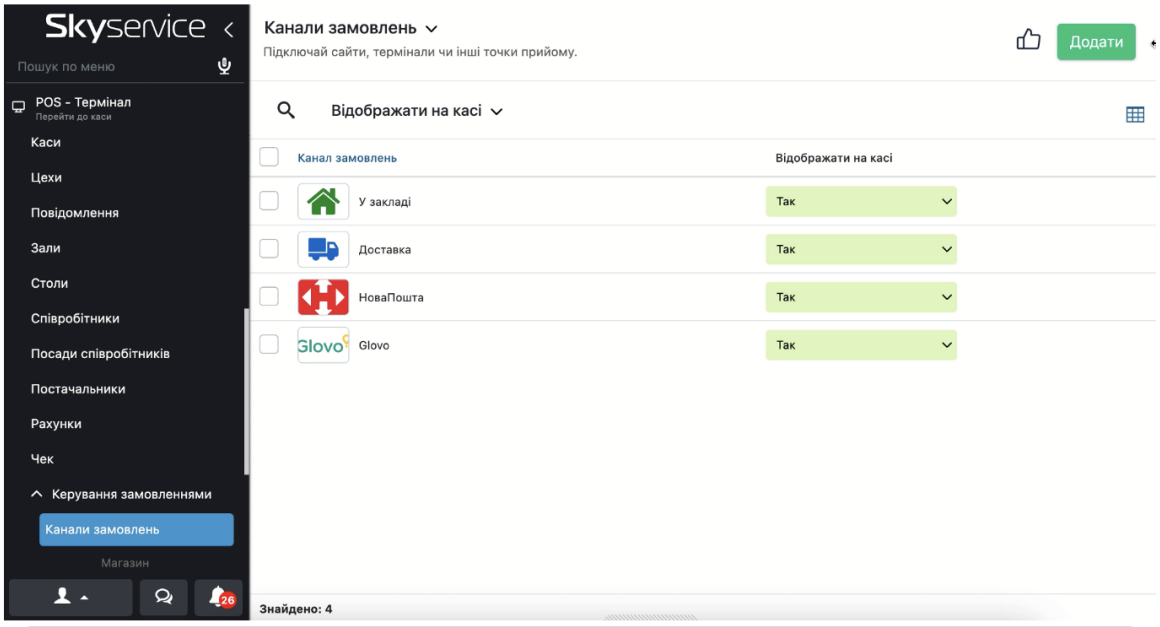
Перегляд статистики за каналами замовлень на прикладі продажів



10

ПРАКТИЧНИЙ РЕЗУЛЬТАТ

Відео-демонстрація основних функцій розробленого модуля



11

ПОРІВНЯЛЬНИЙ АНАЛІЗ

Показник	До впровадження	Після впровадження розробленої методики	Зміна
Точність розподілу замовлень за каналами	36%	98%	62%
Середній час формування замовлення	26.4 сек	12.1 сек	–54%
Кількість помилок у виборі ціни	14.7%	1.2%	–92%
Тижневий обсяг звернень з впровадження	8	2	–75%
Середній чек у каналах з окремими цінами	145 грн	161 грн	11%
Обсяг неправильно розподілених витрат/знижок (людський фактор)	23%	8%	–15%
Доступність аналітики за каналами	Часткова (лише лог-фільтри)	Повна вибірка + агрегація	Системне покращення

ВИСНОВКИ

1. У ході дослідження виявлено недоліки існуючих рішень у багатоканальному середовищі малого та середнього бізнесу. Проаналізовані механізми ціноутворення в аналогах Встановлено, недоліки існуючих підходів до сегментації та маршрутизації замовлень, що не дозволяє ефективно масштабувати кількість каналів.
2. Розроблено та формалізовано алгоритм інтегрованого циклу обробки замовлення — від моменту надходження до запису в статистику. Створено структуру даних і схеми зв'язків визначення каналу замовлення та поканального ціноутворення. Алгоритми забезпечують можливість масштабування системи новими каналами та інтеграціями зі сторонніми сервісами з мінімальними змінами у програмній архітектурі.
3. Побудовано формальну математичну модель для уніфікованого замовлення, розроблено модель функції віднесення замовлення до каналу та описано математичну модель і алгоритм поканального ціноутворення. Побудовані моделі стандартизують логіку системи й забезпечують можливість подальшого аналітичного розширення.
4. Реалізовано методику у вигляді модуля каналів замовлень для системи управління малим і середнім бізнесом, що дозволяє керувати каналами замовлень вручну і з зовнішніх джерел, а також використовувати окреме поканальне ціноутворення.
5. Проведено аналіз статистики реального використання користувачами для визначення ефективності розробленого модуля. Результат демонструє, що використана методика значно підвищує точність розподілення замовлень за каналами та зменшує час обробки і відсоток помилок через людський фактор.

ПУБЛІКАЦІЇ ТА АПРОБАЦІЯ РОБОТИ

Тези доповідей:

1. Жилінський Д.І., Корецька В.О. Актуальність розробки модуля каналів замовлень для системи обліку малого та середнього бізнесу. II Всеукраїнська науково-технічна конференція «Виклики та рішення в програмній інженерії», 26 листопада 2025 р., Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. 26.11.2025, ДУІКТ, м Київ. К.: ДУІКТ, 2025. С. 133.
2. Жилінський Д.І., Корецька В.О. Особливості розробки модуля каналів замовлень для системи обліку малого та середнього бізнесу. II Всеукраїнська науково-технічна конференція «Виклики та рішення в програмній інженерії», 26 листопада 2025 р., Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. 26.11.2025, ДУІКТ, м Київ. К.: ДУІКТ, 2025. С. 449.

ДОДАТОК Б. ЛІСТИНГИ ПРОГРАМНИХ МОДУЛІВ

```
# Migration (Alembic) — migrations/versions/0001_init_schema.py
"""init schema

Revision ID: 0001
Revises:
Create Date: 2025-01-01
"""

from alembic import op
import sqlalchemy as sa

revision = "0001"
down_revision = None
branch_labels = None
depends_on = None


def upgrade() -> None:
    op.create_table(
        "channels",
        sa.Column("id", sa.BigInteger(), primary_key=True),
        sa.Column("name", sa.String(128), nullable=False, unique=True),
        sa.Column("type", sa.String(32), nullable=False),  # e.g. POS/WEB/API/MANUAL/DELIVERY
        sa.Column("priority", sa.Integer(), nullable=False, server_default="100"),
        sa.Column("is_active", sa.Boolean(), nullable=False, server_default=sa.text("true")),
        sa.Column("created_at", sa.DateTime(timezone=True), server_default=sa.text("now()"), nullable=False),
        sa.Column("updated_at", sa.DateTime(timezone=True), server_default=sa.text("now()"), nullable=False),
    )

    op.create_table(
        "channel_rules",
        sa.Column("id", sa.BigInteger(), primary_key=True),
        sa.Column("channel_id", sa.BigInteger(), sa.ForeignKey("channels.id", ondelete="CASCADE"), nullable=False),
        sa.Column("rule_type", sa.String(32), nullable=False),  # SOURCE, INTEGRATION, ATTR_EQUALS
        sa.Column("key", sa.String(128), nullable=True),
        sa.Column("op", sa.String(16), nullable=False, server_default="eq"),
        sa.Column("value", sa.String(256), nullable=False),
        sa.Column("weight", sa.Integer(), nullable=False, server_default="1"),
        sa.Column("is_active", sa.Boolean(), nullable=False, server_default=sa.text("true")),
    )
    op.create_index("ix_channel_rules_channel_id", "channel_rules", ["channel_id"])

    op.create_table(
        "products",
        sa.Column("id", sa.BigInteger(), primary_key=True),
        sa.Column("sku", sa.String(64), nullable=False, unique=True),
        sa.Column("name", sa.String(256), nullable=False),
        sa.Column("base_price", sa.Numeric(12, 2), nullable=False),
        sa.Column("is_active", sa.Boolean(), nullable=False, server_default=sa.text("true")),
    )

    op.create_table(
        "channel_price_modifiers",
        sa.Column("id", sa.BigInteger(), primary_key=True),
        sa.Column("channel_id", sa.BigInteger(), sa.ForeignKey("channels.id", ondelete="CASCADE"), nullable=False, unique=True),
        sa.Column("multiplier", sa.Numeric(8, 4), nullable=False),
        sa.Column("updated_at", sa.DateTime(timezone=True), server_default=sa.text("now()"), nullable=False),
    )

    op.create_table(
        "channel_product_prices",
        sa.Column("id", sa.BigInteger(), primary_key=True),
        sa.Column("channel_id", sa.BigInteger(), sa.ForeignKey("channels.id", ondelete="CASCADE"), nullable=False),
        sa.Column("product_id", sa.BigInteger(), sa.ForeignKey("products.id", ondelete="CASCADE"), nullable=False),
        sa.Column("custom_price", sa.Numeric(12, 2), nullable=False),
        sa.UniqueConstraint("channel_id", "product_id", name="uq_channel_product_price"),
    )

op.create_index("ix_channel_product_prices_channel_id", "channel_product_prices", ["channel_id"])

    op.create_table(
        "orders",
        sa.Column("id", sa.BigInteger(), primary_key=True),
        sa.Column("external_id", sa.String(128), nullable=True),
        sa.Column("source", sa.String(32), nullable=False),  # POS/WEB/API/MANUAL
        sa.Column("integration", sa.String(64), nullable=True),  # e.g. Glovo, Bolt
        sa.Column("channel_id", sa.BigInteger(), sa.ForeignKey("channels.id", ondelete="RESTRICT"), nullable=False),
        sa.Column("currency", sa.String(8), nullable=False, server_default="UAH"),
        sa.Column("created_at", sa.DateTime(timezone=True), server_default=sa.text("now()"), nullable=False),
```

```
            sa.Column("completed_at",
sa.DateTime(timezone=True), nullable=True),
            sa.Column("total", sa.Numeric(12, 2),
nullable=False, server_default="0"),
            sa.Column("meta", sa.JSON(), nullable=False,
server_default=sa.text("'{}'::json")),
        )
        op.create_index("ix_orders_channel_id",
"orders", ["channel_id"])
        op.create_index("ix_orders_created_at",
"orders", ["created_at"])

        op.create_table(
            "order_items",
            sa.Column("id", sa.BigInteger(),
primary_key=True),
            sa.Column("order_id", sa.BigInteger(),
sa.ForeignKey("orders.id", ondelete="CASCADE"),
nullable=False),
            sa.Column("product_id", sa.BigInteger(),
sa.ForeignKey("products.id", ondelete="RESTRICT"),
nullable=False),
            sa.Column("qty", sa.Numeric(12, 3),
nullable=False),
            sa.Column("unit_price", sa.Numeric(12, 2),
nullable=False),
            sa.Column("line_total", sa.Numeric(12, 2),
nullable=False),
        )
        op.create_index("ix_order_items_order_id",
"order_items", ["order_id"])

        op.create_table(
            "channel_daily_stats",
            sa.Column("id", sa.BigInteger(),
primary_key=True),
            sa.Column("channel_id", sa.BigInteger(),
sa.ForeignKey("channels.id", ondelete="CASCADE"),
nullable=False),
            sa.Column("day", sa.Date(), nullable=False),
            sa.Column("orders_count", sa.Integer(),
nullable=False, server_default="0"),
            sa.Column("gross_total", sa.Numeric(14, 2),
nullable=False, server_default="0"),
            sa.Column("avg_check", sa.Numeric(14, 2),
nullable=False, server_default="0"),
            sa.UniqueConstraint("channel_id", "day",
name="uq_channel_day"),
        )


    def downgrade() -> None:
        op.drop_table("channel_daily_stats")
        op.drop_table("order_items")
        op.drop_table("orders")
        op.drop_table("channel_product_prices")
        op.drop_table("channel_price_modifiers")
        op.drop_table("products")
        op.drop_index("ix_channel_rules_channel_id",
table_name="channel_rules")
        op.drop_table("channel_rules")
        op.drop_table("channels")


    # Enums — app/domain/enums.py

    from enum import Enum

    class OrderSource(str, Enum):
        POS = "POS"
        WEB = "WEB"
        API = "API"
        MANUAL = "MANUAL"

    class ChannelRuleType(str, Enum):
        SOURCE = "SOURCE"               # matches
Order.source
        INTEGRATION = "INTEGRATION"     #
matches Order.integration
        ATTR_EQUALS = "ATTR_EQUALS"     #
matches order.meta[key] == value

    class ChannelType(str, Enum):
        POS = "POS"
        WEB = "WEB"
        API = "API"
        MANUAL = "MANUAL"
        DELIVERY = "DELIVERY"
        OTHER = "OTHER"

    # 2.2 Normalized order model —
app/domain/models.py

    from dataclasses import dataclass, field
    from decimal import Decimal
    from datetime import datetime
    from typing import Any, Dict, List, Optional

    from .enums import OrderSource

    @dataclass
    class NormalizedItem:
        product_id: int
        qty: Decimal

    @dataclass
    class NormalizedOrder:
        external_id: Optional[str]
        source: OrderSource
        integration: Optional[str]
        currency: str = "UAH"
        created_at:
datetime                =
field(default_factory=datetime.utcnow)
        meta: Dict[str, Any] = field(default_factory=dict)
        items:
List[NormalizedItem]            =
field(default_factory=list)

    @dataclass
    class PricedItem:
        product_id: int
        qty: Decimal
        unit_price: Decimal
        line_total: Decimal

    @dataclass
    class ProcessedOrder:
        normalized: NormalizedOrder
        channel_id: int
        total: Decimal
        priced_items: List[PricedItem]

    # Normalization (Unified Order)
    #  Normalizer — app/domain/normalization.py
    from __future__ import annotations

    from decimal import Decimal
    from datetime import datetime
    from typing import Any, Dict

    from .enums import OrderSource
```

```
from .models import NormalizedItem, NormalizedOrder

class OrderNormalizer:
    """Convert raw payloads (POS/WEB/API/MANUAL) into a unified NormalizedOrder."""

    def normalize(self, raw: Dict[str, Any]) -> NormalizedOrder:
        source = OrderSource(raw.get("source", "MANUAL"))
        external_id = raw.get("external_id")
        integration = raw.get("integration")
        currency = raw.get("currency", "UAH")
        created_at = self._parse_dt(raw.get("created_at"))

        meta = dict(raw.get("meta", {}))

        items = []
        for it in raw.get("items", []):
            items.append(
                NormalizedItem(
                    product_id=int(it["product_id"]),
                    qty=Decimal(str(it.get("qty", "1"))),
                )
            )

        return NormalizedOrder(
            external_id=external_id,
            source=source,
            integration=integration,
            currency=currency,
            created_at=created_at,
            meta=meta,
            items=items,
        )

    @staticmethod
    def _parse_dt(v: Any) -> datetime:
        if v is None:
            return datetime.utcnow()
        if isinstance(v, datetime):
            return v
        # ISO 8601 expected
        return datetime.fromisoformat(str(v).replace("Z", "+00:00"))

# Adaptive Channel Segmentation (Channel Resolution)
# Rule evaluation and scoring — app/domain/channel_resolution.py
from __future__ import annotations

from dataclasses import dataclass
from typing import Any, Dict, List, Optional

from .enums import ChannelRuleType
from .models import NormalizedOrder

@dataclass(frozen=True)
class ChannelRule:
    channel_id: int
    rule_type: ChannelRuleType
    key: Optional[str]
    op: str
    value: str
    weight: int = 1
    is_active: bool = True

    def matches(self, order: NormalizedOrder) -> bool:
        if not self.is_active:
            return False

        if self.rule_type == ChannelRuleType.SOURCE:
            return str(order.source.value) == self.value

        if self.rule_type == ChannelRuleType.INTEGRATION:
            return (order.integration or "") == self.value

        if self.rule_type == ChannelRuleType.ATTR_EQUALS:
            if not self.key:
                return False
            v = order.meta.get(self.key)
            return str(v) == self.value

        return False

@dataclass(frozen=True)
class ChannelCandidate:
    channel_id: int
    priority: int

class ChannelResolver:
    """Implements g(o): pick channel with max(score * priority_weight).

    Score = sum(weights of matching rules). If no rules match, fallback to default.
    """

    def __init__(
        self,
        rules: List[ChannelRule],
        priorities: Dict[int, int],
        default_channel_id: int,
    ):
        self.rules = rules
        self.priorities = priorities
        self.default_channel_id = default_channel_id

    def resolve(self, order: NormalizedOrder) -> int:
        scores: Dict[int, int] = {}
        for r in self.rules:
            if r.matches(order):
                scores[r.channel_id] = scores.get(r.channel_id, 0) + int(r.weight)

        if not scores:
            return self.default_channel_id

        # apply priority as multiplicative weight (or use tie-break)
        best_channel = None
        best_value = None

        for ch_id, score in scores.items():
            pr = int(self.priorities.get(ch_id, 100))
            value = score * max(pr, 1)
            if best_value is None or value > best_value:
                best_value = value
                best_channel = ch_id
            elif value == best_value:
                # tie-breaker: lower numeric priority wins (more important)
```

```
                    if pr < int(self.priorities.get(best_channel,
100)):
                        best_channel = ch_id

                return int(best_channel)

        #  Channel-Oriented Pricing
        #  Pricing engine — app/domain/pricing.py
        from __future__ import annotations

        from dataclasses import dataclass
        from decimal import Decimal
        from typing import Dict, Optional, Tuple

        @dataclass(frozen=True)
        class PricingContext:
            channel_id: int

        @dataclass
        class PricingSnapshot:
            # base prices: product_id -> base
            base_prices: Dict[int, Decimal]
            # channel custom prices: (channel_id, product_id)
-> custom
            custom_prices: Dict[Tuple[int, int], Decimal]
            # channel modifiers: channel_id -> multiplier
            modifiers: Dict[int, Decimal]

        class PricingEngine:
            """Implements the algorithm:

            1) base price
            2) if custom price exists → use it
            3) else if modifier exists → base * modifier
            4) else base
            """

            def __init__(self, snapshot: PricingSnapshot):
                self.s = snapshot

            def unit_price(self, channel_id: int, product_id:
int) -> Decimal:
                base = self.s.base_prices[product_id]
                custom = self.s.custom_prices.get((channel_id,
product_id))
                if custom is not None:
                    return custom

                mod = self.s.modifiers.get(channel_id)
                if mod is not None:
                    return                (base                *
mod).quantize(Decimal("0.01"))

                return base

        #  Integrated Order Processing Pipeline
        #  Pipeline — app/domain/pipeline.py
        from __future__ import annotations

        from decimal import Decimal

        from .models import NormalizedOrder, PricedItem,
ProcessedOrder
        from .normalization import OrderNormalizer
        from .channel_resolution import ChannelResolver
        from .pricing import PricingEngine

        class OrderProcessingPipeline:
            def __init__(
                self,
                normalizer: OrderNormalizer,
                channel_resolver: ChannelResolver,
                pricing_engine: PricingEngine,
            ):
                self.normalizer = normalizer
                self.channel_resolver = channel_resolver
                self.pricing = pricing_engine

            def process(self, raw_payload: dict) ->
ProcessedOrder:
                # 1) normalize
                order:              NormalizedOrder              =
self.normalizer.normalize(raw_payload)

                # 2) resolve channel
                channel_id                                       =
self.channel_resolver.resolve(order)

                # 3) price items
                priced_items = []
                total = Decimal("0.00")
                for it in order.items:
                    unit    =    self.pricing.unit_price(channel_id,
it.product_id)
                    line              =              (unit              *
it.qty).quantize(Decimal("0.01"))
                    priced_items.append(
                        PricedItem(
                            product_id=it.product_id,
                            qty=it.qty,
                            unit_price=unit,
                            line_total=line,
                        )
                    )
                    total += line

                total = total.quantize(Decimal("0.01"))

                return ProcessedOrder(
                    normalized=order,
                    channel_id=channel_id,
                    total=total,
                    priced_items=priced_items,
                )

        #  Statistics Write (Channel Stats)
        #  Stats updater — app/domain/stats.py
        from __future__ import annotations

        from dataclasses import dataclass
        from datetime import date
        from decimal import Decimal

        @dataclass(frozen=True)
        class ChannelStatsDelta:
            channel_id: int
            day: date
            orders_count_inc: int
            gross_total_inc: Decimal

        class ChannelStatsCalculator:
            @staticmethod
            def delta(channel_id: int, created_at, total:
Decimal) -> ChannelStatsDelta:
                return ChannelStatsDelta(
                    channel_id=channel_id,
                    day=created_at.date(),
                    orders_count_inc=1,
                    gross_total_inc=total,
                )
```

```
# Persistence Layer (SQLAlchemy)
# DB + Session — app/db.py
from __future__ import annotations

from sqlalchemy import create_engine
from sqlalchemy.orm import sessionmaker, declarative_base

Base = declarative_base()

engine = None
SessionLocal = None


def init_db(dsn: str):
    global engine, SessionLocal
    engine = create_engine(dsn, pool_pre_ping=True)
    SessionLocal = sessionmaker(autocommit=False, autoflush=False, bind=engine)


def get_session():
    db = SessionLocal()
    try:
        yield db
    finally:
        db.close()

# ORM tables — app/persistence/models_orm.py
from __future__ import annotations

from sqlalchemy import (
    BigInteger, Boolean, Column, Date, DateTime, ForeignKey, Integer, JSON, Numeric, String, UniqueConstraint
)
from sqlalchemy.orm import relationship
from app.db import Base

class ChannelORM(Base):
    __tablename__ = "channels"

    id = Column(BigInteger, primary_key=True)
    name = Column(String(128), unique=True, nullable=False)
    type = Column(String(32), nullable=False)
    priority = Column(Integer, nullable=False, default=100)
    is_active = Column(Boolean, nullable=False, default=True)
    created_at = Column(DateTime(timezone=True), nullable=False)
    updated_at = Column(DateTime(timezone=True), nullable=False)

class ChannelRuleORM(Base):
    __tablename__ = "channel_rules"

    id = Column(BigInteger, primary_key=True)
    channel_id = Column(BigInteger, ForeignKey("channels.id", ondelete="CASCADE"), nullable=False)
    rule_type = Column(String(32), nullable=False)
    key = Column(String(128), nullable=True)
    op = Column(String(16), nullable=False, default="eq")
    value = Column(String(256), nullable=False)
    weight = Column(Integer, nullable=False, default=1)
    is_active = Column(Boolean, nullable=False, default=True)

class ProductORM(Base):
    __tablename__ = "products"

    id = Column(BigInteger, primary_key=True)
    sku = Column(String(64), unique=True, nullable=False)
    name = Column(String(256), nullable=False)
    base_price = Column(Numeric(12, 2), nullable=False)
    is_active = Column(Boolean, nullable=False, default=True)

class ChannelPriceModifierORM(Base):
    __tablename__ = "channel_price_modifiers"

    id = Column(BigInteger, primary_key=True)
    channel_id = Column(BigInteger, ForeignKey("channels.id", ondelete="CASCADE"), nullable=False, unique=True)
    multiplier = Column(Numeric(8, 4), nullable=False)
    updated_at = Column(DateTime(timezone=True), nullable=False)

class ChannelProductPriceORM(Base):
    __tablename__ = "channel_product_prices"

    id = Column(BigInteger, primary_key=True)
    channel_id = Column(BigInteger, ForeignKey("channels.id", ondelete="CASCADE"), nullable=False)
    product_id = Column(BigInteger, ForeignKey("products.id", ondelete="CASCADE"), nullable=False)
    custom_price = Column(Numeric(12, 2), nullable=False)

    __table_args__ = (UniqueConstraint("channel_id", "product_id", name="uq_channel_product_price"),)

class OrderORM(Base):
    __tablename__ = "orders"

    id = Column(BigInteger, primary_key=True)
    external_id = Column(String(128), nullable=True)
    source = Column(String(32), nullable=False)
    integration = Column(String(64), nullable=True)
    channel_id = Column(BigInteger, ForeignKey("channels.id", ondelete="RESTRICT"), nullable=False)
    currency = Column(String(8), nullable=False, default="UAH")
    created_at = Column(DateTime(timezone=True), nullable=False)
    completed_at = Column(DateTime(timezone=True), nullable=True)
    total = Column(Numeric(12, 2), nullable=False, default=0)
    meta = Column(JSON, nullable=False, default=dict)

    items = relationship("OrderItemORM", back_populates="order", cascade="all, delete-orphan")
```

```
class OrderItemORM(Base):
    __tablename__ = "order_items"

    id = Column(BigInteger, primary_key=True)
    order_id = Column(BigInteger, ForeignKey("orders.id", ondelete="CASCADE"), nullable=False)
    product_id = Column(BigInteger, ForeignKey("products.id", ondelete="RESTRICT"), nullable=False)
    qty = Column(Numeric(12, 3), nullable=False)
    unit_price = Column(Numeric(12, 2), nullable=False)
    line_total = Column(Numeric(12, 2), nullable=False)

    order = relationship("OrderORM", back_populates="items")

class ChannelDailyStatsORM(Base):
    __tablename__ = "channel_daily_stats"

    id = Column(BigInteger, primary_key=True)
    channel_id = Column(BigInteger, ForeignKey("channels.id", ondelete="CASCADE"), nullable=False)
    day = Column(Date, nullable=False)
    orders_count = Column(Integer, nullable=False, default=0)
    gross_total = Column(Numeric(14, 2), nullable=False, default=0)
    avg_check = Column(Numeric(14, 2), nullable=False, default=0)

    __table_args__ = (UniqueConstraint("channel_id", "day", name="uq_channel_day"),)

# Repositories
# Channels repository — app/persistence/repo_channels.py
from __future__ import annotations

from decimal import Decimal
from typing import Dict, List, Tuple

from sqlalchemy.orm import Session

from app.persistence.models_orm import (
    ChannelORM, ChannelRuleORM, ChannelPriceModifierORM, ChannelProductPriceORM, ProductORM
)

class ChannelsRepo:
    def __init__(self, db: Session):
        self.db = db

    def list_channels(self) -> List[ChannelORM]:
        return self.db.query(ChannelORM).filter(ChannelORM.is_active == True).all()

    def priorities(self) -> Dict[int, int]:
        rows = self.db.query(ChannelORM.id, ChannelORM.priority).filter(ChannelORM.is_active == True).all()
        return {int(r[0]): int(r[1]) for r in rows}

    def rules(self) -> List[ChannelRuleORM]:
        return self.db.query(ChannelRuleORM).filter(ChannelRuleORM.is_active == True).all()

    def default_channel_id(self) -> int:
        # choose active channel with minimal priority value; fallback to first
        ch = (
            self.db.query(ChannelORM)
            .filter(ChannelORM.is_active == True)
            .order_by(ChannelORM.priority.asc())
            .first()
        )
        if not ch:
            raise RuntimeError("No active channels configured")
        return int(ch.id)

    def base_prices(self) -> Dict[int, Decimal]:
        rows = self.db.query(ProductORM.id, ProductORM.base_price).filter(ProductORM.is_active == True).all()
        return {int(r[0]): Decimal(str(r[1])) for r in rows}

    def custom_prices(self) -> Dict[Tuple[int, int], Decimal]:
        rows = self.db.query(
            ChannelProductPriceORM.channel_id,
            ChannelProductPriceORM.product_id,
            ChannelProductPriceORM.custom_price,
        ).all()
        return {(int(r[0]), int(r[1])): Decimal(str(r[2])) for r in rows}

    def modifiers(self) -> Dict[int, Decimal]:
        rows = self.db.query(ChannelPriceModifierORM.channel_id, ChannelPriceModifierORM.multiplier).all()
        return {int(r[0]): Decimal(str(r[1])) for r in rows}

# Orders repository — app/persistence/repo_orders.py
from __future__ import annotations

from sqlalchemy.orm import Session

from app.domain.models import ProcessedOrder
from app.persistence.models_orm import OrderORM, OrderItemORM

class OrdersRepo:
    def __init__(self, db: Session):
        self.db = db

    def create_order(self, po: ProcessedOrder) -> OrderORM:
        o = OrderORM(
            external_id=po.normalized.external_id,
            source=po.normalized.source.value,
            integration=po.normalized.integration,
            channel_id=po.channel_id,
            currency=po.normalized.currency,
            created_at=po.normalized.created_at,
            total=po.total,
            meta=po.normalized.meta,
        )
        self.db.add(o)
```

```
            self.db.flush()  # get o.id

            for it in po.priced_items:
                oi = OrderItemORM(
                    order_id=o.id,
                    product_id=it.product_id,
                    qty=it.qty,
                    unit_price=it.unit_price,
                    line_total=it.line_total,
                )
                self.db.add(oi)

            return o
        #  Stats repository — app/persistence/repo_stats.py
        from __future__ import annotation
        from decimal import Decimal
        from sqlalchemy.orm import Session
        from app.domain.stats import ChannelStatsDelta
        from app.persistence.models_orm import
ChannelDailyStatsORM

        class StatsRepo:
            def __init__(self, db: Session):
                self.db = db

            def apply_delta(self, d: ChannelStatsDelta) ->
None:
                row = (
                    self.db.query(ChannelDailyStatsORM)
                    .filter(ChannelDailyStatsORM.channel_id
== d.channel_id)
                    .filter(ChannelDailyStatsORM.day ==
d.day)
                    .with_for_update()
                    .first()
                )
                if row is None:
                    row = ChannelDailyStatsORM(
                        channel_id=d.channel_id,
                        day=d.day,
                        orders_count=0,
                        gross_total=Decimal("0.00"),
                        avg_check=Decimal("0.00"),
                    )
                    self.db.add(row)
                    self.db.flush()
                row.orders_count = int(row.orders_count) +
int(d.orders_count_inc)
                row.gross_total =
Decimal(str(row.gross_total)) + d.gross_total_inc
                # avg_check = gross_total / orders_count
                if row.orders_count > 0:
                    row.avg_check =
(Decimal(str(row.gross_total)) /
Decimal(row.orders_count)).quantize(Decimal("0.01"))

        #  Application Services (composition + transactions)
        #  Service — app/domain/service.py
        from __future__ import annotations

        from sqlalchemy.orm import Session

        from app.domain.channel_resolution import
ChannelResolver, ChannelRule
        from app.domain.enums import ChannelRuleType
        from app.domain.normalization import
OrderNormalizer
        from app.domain.pipeline import
OrderProcessingPipeline
        from app.domain.pricing import PricingEngine,
PricingSnapshot
        from app.domain.stats import
ChannelStatsCalculator
        from app.persistence.repo_channels import
ChannelsRepo
        from app.persistence.repo_orders import
OrdersRepo
        from app.persistence.repo_stats import StatsRepo

        class OrderChannelService:
            """Facade to process an incoming order and
update stats in a single transaction."""

            def __init__(self, db: Session):
                self.db = db
                self.channels = ChannelsRepo(db)
                self.orders = OrdersRepo(db)
                self.stats = StatsRepo(db)

            def process_order(self, raw_payload: dict) -> int:
                # build runtime objects from DB snapshot
                rules_orm = self.channels.rules()
                rules = [
                    ChannelRule(
                        channel_id=int(r.channel_id),

rule_type=ChannelRuleType(r.rule_type),
                        key=r.key,
                        op=r.op,
                        value=r.value,
                        weight=int(r.weight),
                        is_active=bool(r.is_active),
                    )
                    for r in rules_orm
                ]

                resolver = ChannelResolver(
                    rules=rules,
                    priorities=self.channels.priorities(),

default_channel_id=self.channels.default_channel_id(),
                )

                snapshot = PricingSnapshot(
                    base_prices=self.channels.base_prices(),

custom_prices=self.channels.custom_prices(),
                    modifiers=self.channels.modifiers(),
                )
                pricing = PricingEngine(snapshot)
                pipeline =
OrderProcessingPipeline(OrderNormalizer(), resolver,
pricing)
                processed = pipeline.process(raw_payload)

                # persist + stats atomically
                order_row =
self.orders.create_order(processed)
                delta =
ChannelStatsCalculator.delta(processed.channel_id,
processed.normalized.created_at, processed.total)
                self.stats.apply_delta(delta)

                self.db.commit()
                return int(order_row.id)

        #  API Layer (FastAPI)
        #  Schemas
        # app/schemas/orders.py
```

```
from __future__ import annotations

from pydantic import BaseModel, Field
from typing import Any, Dict, List, Optional

class OrderItemIn(BaseModel):
    product_id: int
    qty: str = Field(default="1")

class OrderIn(BaseModel):
    source: str
    external_id: Optional[str] = None
    integration: Optional[str] = None
    currency: str = "UAH"
    created_at: Optional[str] = None
    meta: Dict[str, Any] = {}
    items: List[OrderItemIn]

class OrderCreatedOut(BaseModel):
    order_id: int
# app/schemas/channels.py
from pydantic import BaseModel
from typing import Optional

class ChannelOut(BaseModel):
    id: int
    name: str
    type: str
    priority: int
    is_active: bool

class ChannelRuleOut(BaseModel):
    id: int
    channel_id: int
    rule_type: str
    key: Optional[str]
    op: str
    value: str
    weight: int
    is_active: bool

#  Routes
# Orders — app/api/routes_orders.py
from fastapi import APIRouter, Depends
from sqlalchemy.orm import Session

from app.db import get_session
from app.domain.service import OrderChannelService
from app.schemas.orders import OrderIn, OrderCreatedOut

router = APIRouter(prefix="/orders", tags=["orders"])

@router.post("/ingest", response_model=OrderCreatedOut)
def ingest_order(payload: OrderIn, db: Session = Depends(get_session)):
    svc = OrderChannelService(db)
    order_id = svc.process_order(payload.model_dump())
    return OrderCreatedOut(order_id=order_id)
# Channels — app/api/routes_channels.py
from fastapi import APIRouter, Depends
from sqlalchemy.orm import Session

from app.db import get_session
from app.persistence.repo_channels import ChannelsRepo
from app.schemas.channels import ChannelOut

router = APIRouter(prefix="/channels", tags=["channels"])

@router.get("", response_model=list[ChannelOut])
def list_channels(db: Session = Depends(get_session)):
    repo = ChannelsRepo(db)
    rows = repo.list_channels()
    return [ChannelOut(id=r.id, name=r.name, type=r.type, priority=r.priority, is_active=r.is_active) for r in rows]
#  Reports — app/api/routes_reports.py
from fastapi import APIRouter, Depends
from sqlalchemy.orm import Session
from sqlalchemy import func

from app.db import get_session
from app.persistence.models_orm import ChannelDailyStatsORM

router = APIRouter(prefix="/reports", tags=["reports"])

@router.get("/channels/daily")
def channel_daily(day_from: str, day_to: str, db: Session = Depends(get_session)):
    q = (
        db.query(
            ChannelDailyStatsORM.channel_id,
            ChannelDailyStatsORM.day,
            ChannelDailyStatsORM.orders_count,
            ChannelDailyStatsORM.gross_total,
            ChannelDailyStatsORM.avg_check,
        )
        .filter(ChannelDailyStatsORM.day >= day_from)
        .filter(ChannelDailyStatsORM.day <= day_to)
        .order_by(ChannelDailyStatsORM.day.asc())
    )
    return [
        {
            "channel_id": int(r[0]),
            "day": str(r[1]),
            "orders_count": int(r[2]),
            "gross_total": str(r[3]),
            "avg_check": str(r[4]),
        }
        for r in q.all()
    ]

#  FastAPI app — app/main.py
from fastapi import FastAPI

from app.config import settings
from app.db import init_db
from app.api.routes_orders import router as orders_router
from app.api.routes_channels import router as channels_router
from app.api.routes_reports import router as reports_router

app = FastAPI(title="SkyService Order Channels Module")

@app.on_event("startup")
def on_startup():
    init_db(settings.DATABASE_DSN)
```

```
app.include_router(orders_router)
app.include_router(channels_router)
app.include_router(reports_router)

# Config — app/config.py
from pydantic_settings import BaseSettings

class Settings(BaseSettings):
    DATABASE_DSN: str = "postgresql+psycopg2://user:pass@localhost:5432/skyservice"

settings = Settings()

# Tests (Pytest)
# Channel resolution — tests/test_channel_resolution.py
from app.domain.channel_resolution import ChannelResolver, ChannelRule
from app.domain.enums import ChannelRuleType, OrderSource
from app.domain.models import NormalizedOrder, NormalizedItem

def test_channel_resolver_prefers_matching_rules_and_priority():
    rules = [
        ChannelRule(channel_id=10, rule_type=ChannelRuleType.SOURCE, key=None, op="eq", value="API", weight=1),
        ChannelRule(channel_id=20, rule_type=ChannelRuleType.INTEGRATION, key=None, op="eq", value="Glovo", weight=3),
    ]
    priorities = {10: 100, 20: 90}
    resolver = ChannelResolver(rules, priorities, default_channel_id=10)

    order = NormalizedOrder(
        external_id="x",
        source=OrderSource.API,
        integration="Glovo",
        items=[NormalizedItem(product_id=1, qty=1)],
    )

    assert resolver.resolve(order) == 20
# Pricing — tests/test_pricing.py
from decimal import Decimal

from app.domain.pricing import PricingEngine, PricingSnapshot


def test_pricing_custom_price_has_priority():
    snap = PricingSnapshot(
        base_prices={1: Decimal("100.00")},
        custom_prices={(7, 1): Decimal("130.00")},
        modifiers={7: Decimal("1.20")},
    )
    pe = PricingEngine(snap)
    assert pe.unit_price(7, 1) == Decimal("130.00")

def test_pricing_modifier_applies_when_no_custom():
    snap = PricingSnapshot(
        base_prices={1: Decimal("100.00")},
        custom_prices={},
        modifiers={7: Decimal("1.20")},
    )
    pe = PricingEngine(snap)
    assert pe.unit_price(7, 1) == Decimal("120.00")
# Pipeline — tests/test_pipeline.py
from decimal import Decimal

from app.domain.channel_resolution import ChannelResolver
from app.domain.normalization import OrderNormalizer
from app.domain.pricing import PricingEngine, PricingSnapshot
from app.domain.pipeline import OrderProcessingPipeline


def test_pipeline_prices_and_totals():
    normalizer = OrderNormalizer()
    resolver = ChannelResolver(rules=[], priorities={}, default_channel_id=1)

    snap = PricingSnapshot(
        base_prices={1: Decimal("10.00"), 2: Decimal("5.00")},
        custom_prices={(1, 2): Decimal("6.00")},
        modifiers={},
    )
    pricing = PricingEngine(snap)

    pipe = OrderProcessingPipeline(normalizer, resolver, pricing)

    raw = {
        "source": "POS",
        "items": [
            {"product_id": 1, "qty": "2"},
            {"product_id": 2, "qty": "1"},
        ],
    }

    out = pipe.process(raw)
    assert out.channel_id == 1
    assert out.total == Decimal("26.00")  # 2*10 + 1*6
```