КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БУДІВНИЦТВА І АРХІТЕКТУРИ

Автоматизації і інформаційних технологій

(факультет)

Кібербезпеки та комп'ютерної інженерії

(назва випускової кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

ЗДОБУВАЧА СТУПЕНЯ ВИЩОЇ ОСВІТИ БАКАЛАВР

на тему:

Метод синхронізації дропшипінг-каталогу на основі версійних XML-потоків

Савчук Дмитро Андрійович

(прізвище, ім'я та по батькові здобувача повністю)

Київ 2026 р.

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БУДІВНИЦТВА І АРХІТЕКТУРИ

Автоматизації і інформаційних технологій

(факультет)

Кібербезпеки та комп'ютерної інженерії

(назва кафедри)

ЗАТВЕРДЖУЮ

Завідувач кафедри

Делембовський М. М.

„__" _____20___року

КВАЛІФІКАЦІЙНА РОБОТА

ЗДОБУВАЧА СТУПЕНЯ ВИЩОЇ ОСВІТИ БАКАЛАВР

Метод синхронізації дропшипінг-каталогу на основі версійних XML-потоків

(назва)

Я як здобувач вищої освіти КНУБА розумію і підтримую політику закладу з академічної доброчесності. Я не надавав(-ла) і не одержував(-ла) недозволену допомогу під час підготовки цієї роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

Здобувач Савчук Дмитро Андрійович

(прізвище, ім'я та по батькові повністю)

123 "Комп'ютерна інженерія"

(спеціальність)

Комп'ютерні системи та мережі

(освітня програма)

Група КСМ – 22

Керівник Гуменний Д. О.

(прізвище та ініціали)

Кандидат технічних наук, доцент

(вчене звання, науковий ступінь)

Рецензент Гончаренко Т. А.

(прізвище та ініціали)

Ідентичність підтверджую

Київ 2026 р.

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БУДІВНИЦТВА І АРХІТЕКТУРИ

Факультет: Автоматизації і інформаційних технологій
Випускова кафедра: Кібербезпеки та комп'ютерної інженерії
Ступінь вищої освіти: Бакалавр
Спеціальність: 123 "Комп'ютерна інженерія"
Освітня програма: Комп'ютерні системи та мережі

ЗАТВЕРДЖУЮ

Завідувач кафедри
Делембовський М. М.
„___" _____ 20___ року

З А В Д А Н Н Я

ДО ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ ЗДОБУВАЧА СТУПЕНЯ ВИЩОЇ ОСВІТИ БАКАЛАВР

Савчук Дмитро Андрійович

(прізвище, ім'я та по батькові здобувача)

1. Тема роботи

Метод синхронізації дропшипінг-каталогу на основі версійних XML-потоків

затверджена наказом ректора КНУБА №577/52-15/13.2/26 від «14» травня 2026 року.

Керівник роботи

Гуменний Д. О. кандидат технічних наук, доцент

(прізвище, ім'я та по батькові, науковий ступінь, вчене звання)

3. Термін подання здобувачем роботи до захисту _____

4. Зміст пояснювальної записки за розділами:

Р. 1. Аналіз предметної галузі та постановка задачі.

Р. 2. Розроблення методу інкрементальної синхронізації дропшипінг-каталогу.

Р. 3. Програмна реалізація та тестування методу інкрементальної синхронізаціїдропшипінг-каталогу.

7. Календарний план виконання роботи:

Види робіт та їх зміст	Дата виконання
Розділ 1	09.05.2026
Розділ 2	17.05.2026
Розділ 3	30.05.2026
Розділ 4	-
Розділ 5	-
Остаточне оформлення роботи	
Направлення роботи для перевірки на плагіат	
Попередній захист роботи	
Направлення роботи на рецензування	04.06.2026

8. Дата видачі завдання _____

Керівник _____ _____

Здобувач _____ _____

РЕЗЮМЕ (SUMMARY) *до кваліфікаційної випускової роботи здобувача*	ПІБ *здобувача українською та англійською мовами* *Савчук Дмитро Андрійович* *Dmytro Savchuk Andriyovych*		
ЗВО	Київський національний університет будівництва і архітектури		
Тема *(українською та англійською)*	Метод синхронізації дропшипінг-каталогу на основі версійних XML-потоків A method for synchronizing dropshipping catalogs using XML feeds with versions		
Освітній ступінь	Бакалавр		
Факультет	Автоматизації і інформаційних технологій		
Випускова кафедра	Кібербезпеки та комп'ютерної інженерії		
Спеціальність	123 "Комп'ютерна інженерія"		
Освітня програма	Комп'ютерні системи та мережі		
Керівник	Гуменний Д. О. кандидат технічних наук, доцент		
Обсяг роботи:	*Пояснювальна записка, стор.*	*Розділів*	*Презентація, кількість слайдів*
	82	3	10
Розділ 1	23 стор.		
Розділ 2	22 стор.		
Розділ 3	17 стор.		
Розділ 4	-		
Розділ 5	-		
Висновки по роботі	3 стор.		
Ключові слова: Keywords:	Ключові слова: дропшипінг, XML/YML-фід, товарний каталог, інкрементальна синхронізація, версійний XML-потік, контрольовані поля, хешування, база даних, веб-система, EasyDrop. Keywords: dropshipping, XML/YML feed, product catalog, incremental synchronization, versioned XML flow, controlled fields, hashing, database, web system, EasyDrop.		

Здобувач _____/ _____

Керівник _____/ _____

АНОТАЦІЯ

Кваліфікаційна випускна робота присвячена дослідженню та розробленню методу інкрементальної синхронізації дропшипінг-каталогу на основі версійних XML/YML-потоків. Актуальність теми обумовлена необхідністю підтримання актуального стану товарного каталогу в умовах, коли зовнішній XML/YML-фід надходить як повний знімок даних і не містить готового сигналу про зміни. У роботі проаналізовано структуру товарних XML/YML-каталогів, визначено контрольовані поля товарної позиції, розроблено математичну модель, алгоритм інкрементальної синхронізації та концептуальну модель бази даних. Практичним результатом роботи є web-система, що виконує отримання XML/YML-фіду, парсинг товарних позицій, формування контрольного хешу, порівняння з попереднім станом локальної бази даних, фіксацію нових, оновлених, незмінених і відсутніх товарів, а також відображення результатів синхронізації в інтерфейсі.

Ключові слова: дропшипінг, XML/YML-фід, товарний каталог, інкрементальна синхронізація, контрольовані поля, хешування, база даних, web-система.

ABSTRACT

The qualification graduation work is devoted to the research and development of a method for incremental synchronization of a dropshipping catalog based on versioned XML/YML flows. The relevance of the topic is determined by the need to maintain an up-to-date product catalog when an external XML/YML feed is received as a full snapshot of data and does not contain a ready-made signal about changes. The work analyzes the structure of product XML/YML catalogs, defines controlled fields of a product item, and develops a mathematical model, an incremental synchronization algorithm, and a conceptual database model. The practical result of the work is a web system that performs XML/YML feed retrieval, product item parsing, control hash generation, comparison with the previous state of the local database, detection of new, updated, unchanged, and missing products, and visualization of synchronization results in the interface.

Keywords: dropshipping, XML/YML feed, product catalog, incremental synchronization, controlled fields, hashing, database, web system

ЗМІСТ

СЛОВНИК ТЕРМІНІВ ТА СКОРОЧЕНЬ

Термін / скорочення	**Пояснення**
Дропшипінг	Модель електронної комерції, за якої продавець реалізує товар без його зберігання на власному складі, а відвантаження здійснює постачальник.
Товарний каталог	Структурований набір даних про товари, що містить назви, ціни, наявність, кількість, категорії, зображення та характеристики.
XML	Розширювана мова розмітки, яка використовується для подання структурованих даних.
YML	XML-орієнтований формат товарного фіду, що використовується для передавання товарних пропозицій.
XML/YML-фід	Файл або потік структурованих товарних даних, який передається від постачальника до системи продавця.
Версійний XML-потік	Послідовність оновлених версій XML/YML-фіду, що відображають зміну стану товарного каталогу в часі.
Синхронізація каталогу	Процес приведення локальної бази даних продавця до актуального стану зовнішнього товарного фіду.
Інкрементальна синхронізація	Підхід до оновлення даних, за якого обробляються лише нові, змінені або відсутні записи, а незмінені позиції не перезаписуються.
Контрольовані поля	Поля товару, зміна яких безпосередньо впливає на можливість продажу: ціна, наявність і кількість.
Описові поля	Поля товару, що використовуються для відображення, пошуку та фільтрації: назва, бренд, категорія, зображення, характеристики.
Унікальний ідентифікатор	Поле id або external_id, за яким товар зіставляється між новою версією фіду та попереднім станом бази даних.
group_id	Ідентифікатор групи товарів, що поєднує варіанти однієї моделі, наприклад різні розміри.
Контрольний хеш	Компактне контрольне значення, сформоване на основі контрольованих полів товару.
MD5	Хеш-функція, яка в роботі використовується як технічний механізм компактного порівняння контрольованих полів, а не як засіб криптографічного захисту.
API	Програмний інтерфейс для обміну даними між компонентами системи.
REST API	Тип API, у якому взаємодія між клієнтом і сервером здійснюється через HTTP-запити.
SQLite	Реляційна база даних, використана для зберігання товарів, історії синхронізацій і журналу змін.
Dashboard	Інтерфейсна сторінка для перегляду статистики синхронізації та аналітичних показників.
NEW	Стан товару, який з’явився у новому фіді, але був відсутній у попередньому стані бази даних.

Термін / скоро-чення	**Пояснення**
UPDATED	Стан товару, для якого змінився контрольний хеш.
UNCHANGED / SKIP	Стан товару, який не змінив контрольованих полів і не потребує повторного запису.
MISSING / DELETED	Стан товару, який був у попередній базі даних, але відсутній у новому фіді.

ВСТУП

У сучасних умовах розвитку електронної комерції значна частина інтернет-магазинів працює з великими обсягами товарних даних, які надходять від зовнішніх постачальників. Особливо це характерно для дропшипінг-моделі, у якій продавець не зберігає товари на власному складі, а використовує каталог постачальника для формування власної товарної пропозиції. За таких умов актуальність ціни, наявності та кількості товарів безпосередньо впливає на коректність роботи веб-каталогу, можливість приймання замовлень і якість обслуговування покупців.

Товарні дані у дропшипінг-системах часто передаються у вигляді структурованих XML/YML-фідів. Такий фід може містити тисячі товарних позицій, категорії, ідентифікатори, назви, ціни, залишки, зображення, бренди та додаткові характеристики. Проблема полягає в тому, що зовнішній XML/YML-фід зазвичай надходить як повний знімок поточного стану каталогу, а не як готовий перелік конкретних змін. Тому система має самостійно визначати, які товари були додані, оновлені, залишилися без змін або зникли з нового фіду.

Повний перезапис каталогу є простим у реалізації, однак для великих товарних баз він створює надлишкові операції запису та ускладнює збереження історії змін. Підхід на основі службового поля modified_date залежить від того, чи передає постачальник таке поле та чи оновлює його коректно. Якщо у фіді немає надійного службового сигналу про зміни, доцільно застосовувати власний механізм інкрементальної синхронізації на основі порівняння нової версії XML/YML-фіду з попереднім станом локальної бази даних.

Актуальність теми роботи зумовлена необхідністю підвищення ефективності оновлення дропшипінг-каталогів, які регулярно надходять у вигляді XML/YML-фідів. Для таких каталогів важливо не лише завантажити новий файл, а й визначити, які саме товарні позиції потребують додавання, оновлення або деактивації, щоб зменшити кількість зайвих операцій запису та підтримувати локальний каталог в актуальному стані.

Метою роботи є підвищення ефективності оновлення веб-каталогу товарів шляхом розроблення та реалізації методу синхронізації дропшипінг-каталогу на основі версійних XML-потоків.

Для досягнення поставленої мети необхідно виконати такі завдання:

1) проаналізувати особливості обміну товарними даними в дропшипінг-моделі електронної комерції;

2) дослідити структуру XML/YML-каталогів товарів та визначити контрольовані поля для синхронізації;

3) проаналізувати існуючі підходи до оновлення товарних каталогів і виявлення змін у XML-даних;

4) обґрунтувати доцільність інкрементального підходу до синхронізації каталогу;

5) розробити математичну модель і алгоритм визначення змін товарів на основі хешування контрольованих полів;

6) розробити концептуальну модель даних і структури даних, необхідні для підтримки запропонованого методу;

7) реалізувати програмну веб-систему з серверною частиною, базою даних, REST API та клієнтським інтерфейсом;

8) провести тестування системи на даних XML-каталогу EasyDrop;

9) оцінити ефективність запропонованого підходу за кількістю уникнених операцій запису та часом виконання синхронізації.

Об’єктом дослідження є процес синхронізації товарних даних між зовнішнім XML/YML-каталогом постачальника та локальною базою даних веб-каталогу.

Предметом дослідження є метод інкрементального виявлення змін у товарному XML/YML-каталозі на основі хешування контрольованих полів товару.

У роботі використано методи аналізу предметної області, формалізації, теорії множин, хешування контрольованих полів, проєктування баз даних і експериментального тестування програмної системи.

Практична цінність роботи полягає у створенні методу та програмної веб-системи для автоматизованого оновлення дропшипінг-каталогу на основі XML/YML-

фіду. Запропонований підхід дозволяє зіставляти товари між версіями каталогу за унікальним ідентифікатором, визначати зміну контрольованих полів через хешування ціни, наявності та кількості, фіксувати нові, оновлені, незмінені й відсутні позиції та зберігати результати синхронізації для подальшого аналізу.

Робота складається зі вступу, трьох розділів, загальних висновків, списку використаних джерел і додатків. У першому розділі проаналізовано предметну область, структуру XML/YML-каталогів і підходи до виявлення змін. У другому розділі розроблено математичну модель, алгоритм, структури даних і концептуальну модель бази даних. У третьому розділі описано програмну реалізацію системи, її архітектуру, API, тестування й аналіз результатів роботи на XML-каталозі EasyDrop.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Особливості дропшипінг-моделі та роль товарних каталогів в електронній комерції

Дропшипінг є моделлю електронної комерції, за якої роздрібний продавець організовує продаж товару кінцевому покупцю, але не зберігає його на власному складі. Після оформлення замовлення дані передаються постачальнику, який забезпечує відвантаження безпосередньо покупцю. У науковій літературі такий підхід розглядається як варіант організації електронної комерції, де складське зберігання та виконання замовлення бере на себе постачальник [1].

На відміну від класичного інтернет-магазину, у дропшипінгу значна частина операційної інформації перебуває на стороні постачальника. Тому продавець залежить від якості, повноти та актуальності товарних даних, які отримує із зовнішнього джерела. У дослідженнях електронної комерції якість інформації розглядається як чинник, що впливає на поведінку користувачів і ефективність роботи платформи [2].

Товарний каталог у дропшипінг-моделі є не лише переліком назв і зображень, а структурованим набором даних для прийняття прикладних рішень у процесі продажу. На його основі відображається товар на сайті, розраховується роздрібна ціна, перевіряється можливість виконання замовлення та формується інформаційне наповнення картки товару. Отже, актуальність каталогу безпосередньо впливає на коректність роботи інформаційної системи продавця.

Враховуючи роль якості інформації в електронній комерції [2], у межах цієї роботи до контрольованих полів доцільно віднести параметри, які безпосередньо впливають на можливість продажу товару: ціну, наявність і кількість. Ціна визначає основу для формування роздрібної вартості; наявність показує, чи може товар бути запропонований покупцю; кількість уточнює доступний залишок. Неактуальність цих даних призводить до помилок у ціноутворенні, прийняття замовлень на відсутні товари та зниження довіри покупців.

Наявність і кількість визначають, чи може продавець приймати та виконувати замовлення на відповідну позицію. Якщо товар показано як доступний, але у постачальника він фактично відсутній або його залишок менший за замовлений обсяг, виникають скасування, додаткова комунікація з клієнтом і репутаційні ризики. Якщо ж доступний залишок занижений, продавець може втратити потенційні продажі.

Окрім контрольованих полів, каталог містить описові дані: назву, бренд, категорію, характеристики, зображення, варіанти товару та опис. Вони важливі для картки товару, пошуку й фільтрації, але не кожна така зміна потребує такого самого оперативного реагування, як зміна ціни, наявності або кількості. Тому описові поля можуть зберігатися в локальній базі даних, але не завжди входять до базового контрольованого набору.

Проблема підтримання актуального каталогу посилюється зі зростанням асортименту. Для невеликої кількості товарів ручна перевірка теоретично можлива, але для каталогів із сотнями або тисячами позицій вона стає неефективною, збільшує кількість ручних операцій і підвищує ймовірність помилок.

Синхронізація товарного каталогу означає приведення локальної бази даних продавця до стану, що відповідає актуальній версії каталогу постачальника. Вона не зводиться до простого завантаження нового файлу: система має визначити, які товари з’явилися вперше, які були видалені або стали недоступними, які залишилися без змін, а які потребують оновлення через зміну контрольованих параметрів.

Спрощену схему взаємодії постачальника, продавця та покупця в дропшипінг-моделі наведено на рисунку 1.1.

Рисунок 1.1 — Спрощена схема взаємодії учасників дропшипінг-моделі

Як видно з рисунка 1.1, продавець отримує товарні дані від постачальника, приймає замовлення покупця, передає його постачальнику, а постачальник забезпечує відвантаження товару.

Саме тому задача синхронізації у дропшипінгу має не лише технічне, а й прикладне значення для роботи продавця: вона пов’язана з ціноутворенням, достовірністю складської інформації, зменшенням ручних перевірок і стабільною роботою веб-каталогу. У межах цієї роботи основна увага зосереджується на методі інкрементального виявлення змін, який дає змогу обробляти не весь каталог повністю, а лише позиції зі змінами контрольованих полів.

1.2. XML/YML-каталоги як засіб передавання товарних даних

Для передавання товарних даних між постачальниками, CRM-системами, маркетплейсами, інтернет-магазинами та локальними базами даних використовуються структуровані файли. Одним із поширених підходів є XML- або YML-каталоги, де дані подаються у вигляді ієрархічної структури з визначеними елементами та атрибутами. Це зручно для машинної обробки, оскільки кожне значення прив’язане до конкретного тегу або атрибута.

XML (Extensible Markup Language) є універсальною мовою розмітки для опису структурованих даних. XML-документ складається з елементів, атрибутів і

вкладених вузлів, що дозволяє описувати як прості, так і складні ієрархічні структури. Специфікація XML 1.0 визначена як рекомендація W3C і описує синтаксис подання структурованих документів у мережевому середовищі [3].

YML у практиці електронної комерції використовується як XML-орієнтований формат товарного фіду для структурованого подання товарних пропозицій. У таких файлах визначаються блоки магазину, категорій і товарів, а кожна позиція має типові поля: ідентифікатор, назву, ціну, валюту, категорію, наявність, зображення та додаткові параметри. Офіційні довідки платформ використовують позначення XML/YML або XML(YML) для опису форматів імпорту й експорту товарних даних [4; 5].

XML/YML-фід є джерелом даних, яке може регулярно оновлюватися постачальником. У дропшипінг-сценарії продавець періодично отримує нову версію файлу й порівнює її з попереднім станом локальної бази даних. Фід може бути доступний за посиланням, формуватися на вимогу або експортуватися з CRM-платформи, але в усіх випадках передає актуальний стан асортименту.

Типовий XML/YML-каталог може містити кореневий елемент, блок категорій та блок товарних позицій; у деяких форматах також **можуть передаватися** службові дані про магазин або джерело каталогу. Блок категорій задає структуру групування, а блок товарних позицій містить елементи offer, item або product залежно від формату. Офіційні вимоги до XML-прайс-листів передбачають такі дані, як ідентифікатор, ціна, статус наявності, кількість, назва, опис, характеристики й категорія [5].

Ключову роль у синхронізації виконує унікальний ідентифікатор товару: id, external_id, offer id або інше подібне поле. За цим значенням система визначає, що товар із нової версії XML/YML-фіду відповідає певному запису в попередній версії або в локальній базі даних. Без стабільного ідентифікатора зіставлення ускладнюється, оскільки назва, опис або зображення можуть змінюватися [5].

Поле group_id використовується для зв’язування товарів, що є варіантами однієї моделі, наприклад різними розмірами, кольорами або комплектаціями. Для

оперативної синхронізації воно не завжди є контрольованим полем, але важливе для побудови структури каталогу та коректного відображення товарів на веб-сайті.

Поля price або priceuah відображають вартість товару. У межах задачі синхронізації ціна є контрольованим полем, оскільки її зміна безпосередньо впливає на фінансовий результат продажу та має бути зафіксована в локальному каталозі.

Поле available позначає доступність товару, а quantity або quantity_in_stock деталізує кількість одиниць у постачальника. Ці параметри пов'язані, але не тотожні: товар може бути доступним, проте мати обмежений залишок. У межах цієї роботи обидва поля належать до контрольованих, оскільки визначають можливість виконання замовлення.

Описові поля, зокрема name, vendor, categoryId, picture або image, формують інформаційне наповнення товару для відображення, пошуку, фільтрації та SEO-опису. Для оперативного визначення можливості продажу найбільше значення мають контрольовані поля: ціна, наявність і кількість; описові дані можуть оновлюватися окремо [2; 5].

Узагальнену структуру XML/YML-фіду, побудовану з урахуванням використаного в роботі каталогу, наведено на рисунку 1.2.

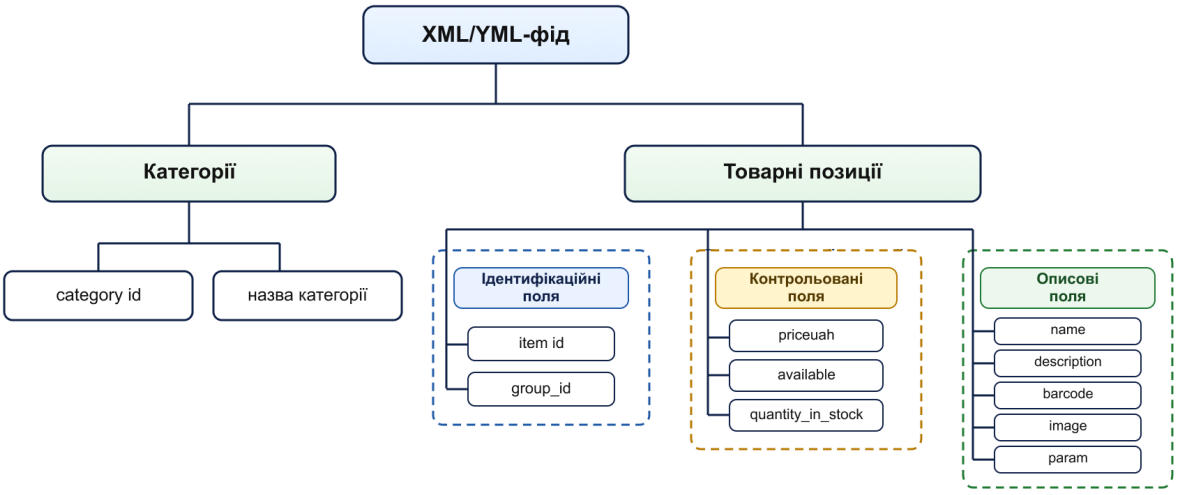

Рисунок 1.2 — Узагальнена структура XML/YML-фіду для синхронізації товарів

Як видно з рисунка 1.2, XML/YML-фід містить блок категорій і блок товарних позицій. Пунктирні рамки на рисунку не відображають окремі службові вузли XML/YML-файлу, а позначають аналітичне групування полів товарної позиції за їх роллю у синхронізації: ідентифікаційні поля використовуються для зіставлення товарів між версіями фіду, контрольовані — для визначення зміни операційного стану, а описові — для відображення інформації в каталозі. Приклад фрагмента XML/YML-фіду товарного каталогу наведено в додатку А.

У таблиці 1.1 наведено основні поля товару, що можуть зустрічатися у типових XML/YML-каталогах, а також їхню роль у процесі синхронізації.

Таблиця 1.1 — Основні поля товару в XML/YML-каталозі

Поле	Призначення	Роль у синхронізації
id / external_id / offer id	Унікальний ідентифікатор товарної позиції у зовнішньому каталозі.	Ключ для зіставлення товару з локальним записом у базі даних.
group_id	Ідентифікатор групи товарів, що є варіантами однієї моделі.	Допоміжне поле для збереження логічного зв’язку між варіантами товару.
name	Назва товару, яка відображається у каталозі продавця.	Описове поле; може зберігатися, але не завжди потребує оперативного оновлення.
price / priceuah	Ціна товару або ціна у відповідній валюті.	Контрольоване поле, зміна якого впливає на ціноутворення та маржинальність.
available	Статус наявності товару на складі постачальника.	Контрольоване поле, що визначає можливість приймання замовлення.
quantity / quantity_in_stock	Кількість одиниць товару, доступних у постачальника.	Контрольоване поле, що деталізує доступний залишок.
vendor / brand	Бренд, виробник або постачальник товару.	Описове поле для фільтрації, пошуку та формування картки товару.
categoryId	Ідентифікатор категорії, до якої належить товар.	Описове або структурне поле, важливе для групування товарів.
picture / image	Посилання на зображення товару.	Описове поле; може оновлюватися окремо від контрольованих параметрів продажу.
param / characteristics	Набір додаткових характеристик товару.	Описове поле, що впливає на повноту картки товару та фільтрацію.

Таблиця 1.1 показує, що поля XML/YML-каталогу мають різну роль: ідентифікатор забезпечує зіставлення товарів між версіями, описові поля формують

картку товару, а контрольовані поля визначають потребу в оперативному оновленні локальної системи.

Для цієї роботи контрольованими полями доцільно вважати ціну, наявність і кількість, оскільки вони найтісніше пов’язані з можливістю продажу товару та виконанням замовлення. За потреби набір може бути розширений: категорія, зображення або опис можуть додаватися до контрольованого набору чи оброблятися окремим сценарієм.

Отже, XML/YML-каталог є зручним джерелом структурованих товарних даних, але сам файл не вирішує задачу синхронізації. Для практичного використання потрібно визначити ключові поля, правила зіставлення записів між версіями та ознаки появи, зникнення або зміни товару.

У межах цієї роботи версійний XML-потік розглядається як послідовність актуалізованих версій XML/YML-фіду, що надходять від зовнішнього джерела та відображають зміну стану товарного каталогу в часі.

1.3. Проблема оновлення товарного каталогу

Після отримання XML/YML-каталогу система продавця має оновити локальну базу даних. На перший погляд достатньо завантажити новий файл, однак у практичних умовах потрібно визначити конкретні зміни між версіями даних.

Нова версія каталогу може містити кілька типів змін: частина товарів залишається без змін, частина змінює ціну, наявність або кількість, з’являються нові позиції, а окремі товари зникають із фіду або стають недоступними. Для ефективної синхронізації кожен товар потрібно класифікувати за характером змін.

Найпростішим підходом є ручне оновлення, коли менеджер порівнює дані постачальника з локальним каталогом і вносить зміни вручну. Такий підхід дає можливість людської перевірки, але для каталогів із сотнями або тисячами позицій він потребує значних витрат часу, не гарантує відсутності помилок і не забезпечує регулярного оновлення з потрібною частотою.

Іншим підходом є повний перезапис каталогу: система під час кожного циклу синхронізації перезаписує всі записи та завантажує дані з нового XML/YML-файлу.

Він простий у реалізації й може бути прийнятним для невеликих каталогів або систем, де історія змін не має значення.

Недоліком повного перезапису є надлишковість: навіть якщо змінилися лише кілька десятків товарів, система виконує операції для всіх позицій. Це збільшує навантаження на базу даних, створює зайві записи, ускладнює аудит змін і може впливати на пов’язані дані — історію цін, картки товарів або прив’язки до замовлень.

Часткове оновлення за метаданими передбачає, що постачальник додає службове поле modified_date, updated_at або аналогічний параметр, який показує дату останньої зміни. Тоді система може обробляти лише товари, дата зміни яких є новішою за дату попередньої синхронізації.

Однак підхід на основі modified_date залежить від постачальника: поле має бути наявним у фіді та коректно оновлюватися під час кожної зміни. Якщо дата зміни відсутня або заповнюється непослідовно, система не може надійно визначати фактичні зміни. В аналізованому XML-каталозі modified_date не визначається на рівні зовнішнього фіду, тому потрібен власний механізм порівняння.

Оновлення через API є гнучким способом інтеграції, оскільки дозволяє отримувати або змінювати дані програмно, працювати з пагінацією, фільтрацією та службовими статусами. Проте API-інтеграція прив’язана до конкретної платформи, авторизації, лімітів запитів і формату відповіді, тому не може бути базовою передумовою запропонованого методу.

Інкрементальне оновлення на основі контрольованих полів передбачає, що система аналізує нову версію каталогу, зіставляє товари за унікальним ідентифікатором і визначає зміну контрольованих параметрів. Новий товар відсутній у попередньому стані; відсутній у новому фіді товар може бути позначений як видалений або недоступний; товар зі зміненою ціною, наявністю чи кількістю класифікується як оновлений.

Перевага інкрементального підходу полягає в обробці лише фактично змінених записів. Це зменшує кількість операцій запису до бази даних, дає змогу зберігати історію змін і контролювати результат кожного циклу синхронізації. Такий

підхід потребує реалізації механізму порівняння, але відповідає задачі роботи, оскільки не залежить від modified_date і не вимагає повного перезапису каталогу.

У таблиці 1.2 наведено порівняння основних підходів до оновлення товарного каталогу з урахуванням їхнього механізму, переваг, обмежень і придатності для каталогів із великою кількістю товарних позицій.

Таблиця 1.2 — Порівняння підходів до оновлення товарного каталогу

Підхід	**Механізм оновлення**	**Чи оновлюються лише змінені товари**	**Переваги**	**Обмеження**
Ручне оновлення	Менеджер вручну переглядає каталог постачальника та змінює локальні записи.	Так, але лише за умови коректної ручної перевірки.	Гнучкість і можливість людського контролю.	Не масштабується для великих каталогів; висока ймовірність помилок.
Повний перезапис каталогу	Видалення або перезапис усіх записів із подальшим завантаженням нових даних.	Ні.	Простота реалізації та відсутність складного порівняння.	Надлишкові операції запису; ускладнення збереження історії змін.
Оновлення за modified_date	Обробка лише товарів із датою зміни, новішою за попередню синхронізацію.	Так, якщо постачальник коректно передає дату зміни.	Ефективність за наявності якісних метаданих.	Залежність від постачальника та коректності оновлення службового поля.
Оновлення через API	Отримання або оновлення товарів через програмні HTTP-запити до платформи.	Залежить від можливостей API.	Гнучкість і можливість точкових операцій.	Прив’язка до конкретної платформи, авторизації та лімітів запитів.
Інкрементальне оновлення	Зіставлення товарів між новою версією XML/YML-фіду та попереднім станом локальної бази даних за унікальним ідентифікатором із подальшим порівнянням контрольованих полів. У межах цієї роботи зміни визначаються власним механізмом порівняння, а не готовим дельта-сигналом платформи.	Так.	Зменшення кількості операцій запису; можливість збереження історії змін; незалежність від службового поля modified_date.	Потребує реалізації алгоритму порівняння, зберігання попереднього стану та визначення набору контрольованих полів.

Таблиця 1.2 показує, що жоден підхід не є універсальним: ручне оновлення не масштабується, повний перезапис створює надлишкове навантаження, а modified_date та API залежать від постачальника або платформи. Інкрементальне

оновлення також не є автоматично доступним для будь-якого XML/YML-фіду, оскільки потребує або службового сигналу про зміну, або власного механізму порівняння. У межах цієї роботи такий механізм будується на зіставленні товарів за унікальним ідентифікатором і порівнянні контрольованих полів із попереднім локальним станом.

Для дропшипінг-каталогу, який регулярно надходить як XML/YML-файл, доцільний підхід, що не потребує спеціального службового поля з боку постачальника. Система може розглядати кожну нову версію фіду як повний знімок поточного стану, але виконувати запис до бази даних лише для товарів зі зміненими контрольованими параметрами.

Таким чином, ключова складність оновлення товарного каталогу полягає не лише у виборі способу завантаження XML/YML-файлу, а в тому, що зовнішній фід надходить як повний знімок поточного стану каталогу без службового сигналу про те, які саме позиції змінилися. Оскільки в аналізованому фіді відсутнє поле modified_date, а універсальний дельта-інтерфейс не є базовою умовою методу, система має самостійно порівнювати кожну нову версію каталогу з попереднім станом локальної бази даних. Саме це обґрунтовує доцільність інкрементального підходу на основі контрольованих полів для задачі цієї роботи.

1.4. Аналіз платформ та інструментів для роботи з товарними каталогами

Для обґрунтування актуальності задачі необхідно розглянути використання товарних даних у сучасних рішеннях електронної комерції. У цьому підрозділі платформи подаються не як об’єкти вибору оптимального сервісу, а як приклади практичного використання структурованих фідів для імпорту, експорту й оновлення каталогів.

Рішення для роботи з каталогами можна згрупувати за типами. До першої групи належать маркетплейси та платформи електронної комерції, зокрема Prom.ua і Rozetka. Prom.ua у роботі розглядається як приклад української платформи, де XML/YML-файли можуть використовуватися для імпорту товарних даних [4]. Rozetka має вимоги до XML-прайс-листів і товарних фідів, що підтверджує

використання структурованої товарної інформації в українській електронній комерції [5].

Друга група охоплює SaaS/CMS-платформи для створення та адміністрування інтернет-магазинів. Хорошоп доцільно розглядати не як маркетплейс, а як платформу для власного інтернет-магазину, у якій можливе наповнення та оновлення каталогу через структуровані файли. Офіційна довідка Хорошоп описує імпорт товарів за допомогою XML, YML або Excel-файлів [6].

Третю групу становлять дропшипінг CRM-платформи, які поєднують постачальників і продавців. EasyDrop позиціонується як дропшипінг-платформа у форматі CRM-системи, що об’єднує постачальників, продавців, склади та комунікацію [7]. У межах цієї роботи вона використовується як практичне джерело XML-даних для реалізації та перевірки методу.

До цієї ж предметної області належить MyDrop та подібні рішення для взаємодії продавців із постачальниками. Офіційний сайт MyDrop позиціонує сервіс як систему обліку для постачальників і дропшиперів, що підтверджує його прикладний контекст [8]. Водночас метод цієї роботи не прив’язується до внутрішньої логіки MyDrop або іншої закритої системи.

Дропшипінг-платформи або агрегатори постачальників, зокрема Dropshipping.ua, також використовують файловий обмін. На офіційному сайті зазначено можливість завантаження товарів у форматах XML, YML, XLSX і CSV, а також оновлення наявності та цін [9].

Окремо існують плагіни для імпорту товарів у CMS, наприклад інструменти для WooCommerce. Сторінка WP All Import для WooCommerce описує імпорт товарів із XML, CSV, Excel або Google Sheets-файлів та зіставлення даних із полями WooCommerce [10]. Такі рішення прив’язані до конкретної CMS і не завжди дозволяють дослідити внутрішній алгоритм визначення оновлених позицій.

До міжнародних SaaS-аналогів належать Spark Shipping та AutoDS. Вони позиціонуються як інструменти автоматизації дропшипінгу, роботи з постачальниками, оновлення інвентарю та інтеграції з каналами продажу [11; 12].

У таблиці 1.3 наведено порівняння платформ та джерел товарних XML/YML-даних, які відображають різні сценарії роботи з товарними фідами.

Таблиця 1.3 — Порівняння платформ та джерел товарних XML/YML-даних

Рішення	Тип рішення	Формат або джерело товарних даних	Механізм оновлення або імпорту	Значення для цієї роботи
Prom.ua	Маркетплейс / платформа електронної комерції	XML/YML-файли, прайс-листи або дані кабінету продавця [4].	Імпорт товарів через файл; API не розглядається як універсальна умова методу.	Приклад українського контексту застосування XML/YML-форматів.
Rozetka	Маркетплейс / платформа електронної комерції	XML-прайс-лист або товарний фід продавця [5].	Підготовка й передавання фіду згідно з вимогами платформи.	Підтверджує використання структурованих товарних фідів у практиці продажів.
Хорошоп	SaaS/CMS-платформа для інтернет-магазинів	XML, YML або Excel-файли для наповнення й оновлення каталогу [6].	Імпорт товарів із файлу в адміністративному середовищі платформи.	Приклад використання товарних файлів у власному інтернет-магазині.
EasyDrop	Дропшипінг CRM-платформа	Дропшипінг CRM-платформа [7]; XML-фід використовується у роботі як практичне джерело даних.	Використання зовнішнього XML-фіду; часткове оновлення потребує власного механізму порівняння.	Практичне джерело XML-даних для реалізації та перевірки методу.
MyDrop	Система обліку / дропшипінг CRM-платформа	Каталог і дані постачальників у середовищі дропшипінгу [8].	Функції взаємодії продавців і постачальників залежать від платформи.	Додатковий приклад предметної області дропшипінгу.
Dropshipping.ua	Дропшипінг-платформа / агрегатор	XML, YML, XLSX, CSV-файли для товарних даних [9].	Оновлення цін і наявності у межах можливостей платформи.	Приклад файлового обміну товарними даними.
WooCommerce / WP All Import	CMS-плагін для імпорту товарів	XML, CSV, Excel або Google Sheets-файли [10].	Зіставлення полів фіду з полями Woo-Commerce.	Показує наявність готових інструментів імпорту, але не замінює дослідження методу.
Spark Shipping / AutoDS	Міжнародні SaaS-сервіси	Дані постачальників, фіди, інтеграції, інструменти автоматизації [11; 12].	Закрита комерційна автоматизація дропшипінг-процесів.	Контекст комерційних аналогів, без використання як наукової основи методу.

Таблиця 1.3 показує, що товарні фіди використовуються у маркетплейсах, SaaS/CMS-платформах, дропшипінг CRM-платформах, агрегаторах постачальників і міжнародних SaaS-сервісах. Офіційні джерела підтверджують практичний контекст функцій імпорту, експорту або автоматизації в конкретних платформах, тоді як технічна доцільність методу обґрунтовується у підрозділі 1.5 на основі наукових і технічних джерел.

Наведені рішення не знімають потреби у власному методі інкрементальної синхронізації. Маркетплейси та CMS-платформи мають власні правила імпорту, дропшипінг-платформи можуть не надавати відкритого механізму визначення змін, а комерційні SaaS-сервіси зазвичай є закритими. Тому доцільно розробити прозорий метод, який можна пояснити, реалізувати й адаптувати до різних XML/YML-каталогів.

Prom.ua у роботі не є центральним об’єктом дослідження, а використовується лише як приклад українського контексту застосування XML/YML-файлів. Основним джерелом даних для перевірки методу є XML-каталог EasyDrop.

Отже, аналіз платформ показує прикладний контекст проблеми оновлення товарних даних. Головним у роботі є не дослідження конкретної платформи, а побудова методу визначення змін у версіях XML/YML-каталогу за контрольованими полями.

1.5. Огляд технічних підходів до виявлення змін у XML-даних

Виявлення змін між двома версіями XML-документа є технічною задачею, що може розв’язуватися різними способами. Вибір підходу залежить від структури документа, типу потрібних змін, вимог до швидкодії, доступних метаданих і подальших дій системи. Для синхронізації каталогу важливо не лише знайти відмінності між файлами, а й перетворити їх на прикладні дії: додавання, оновлення, позначення як недоступного або ігнорування незмінених позицій.

XML-документ з погляду алгоритмічної обробки можна розглядати як дерево вузлів: теги є вузлами, вкладені елементи формують дочірні зв’язки, а атрибути й текстові значення уточнюють зміст. Універсальні алгоритми порівняння XML

працюють із такою моделлю та можуть визначати вставки, видалення, оновлення і переміщення вузлів.

Повне порівняння XML-документів доцільне для складних структур, де змінюється не лише значення поля, а й ієрархія документа. У науковій літературі відомі підходи XyDiff та X-Diff, призначені для порівняння XML-дерев і пошуку структурних відмінностей [13; 14]. Порівняльні дослідження XML change detection підкреслюють, що вибір алгоритму залежить від моделі документа, представлення змін, продуктивності та семантики вузлів [15].

Для товарного XML/YML-каталогу повне деревоподібне порівняння часто є надлишковим. Каталог зазвичай має стабільну структуру: блок категорій, блок товарів і повторюваний набір полів. Прикладно важливі зміни найчастіше відбуваються на рівні значень полів товару, тому універсальний XML-diff алгоритм може створювати зайву обчислювальну складність без суттєвого виграшу для оновлення ціни, наявності та кількості.

Другий підхід полягає у використанні службового поля дати зміни, наприклад modified_date або updated_at. Якщо постачальник передає й коректно оновлює таке поле для кожного товару, система може відфільтрувати записи, змінені після попередньої синхронізації.

Обмеження цього підходу полягає в залежності від якості даних постачальника. Якщо modified_date відсутнє або оновлюється непослідовно, система не може покладатися на нього як на єдине джерело істини. Крім того, дата не завжди показує, яке поле було змінене: товар може отримати нову дату через зміну опису або зображення, хоча ціна й залишок залишилися незмінними.

Третій підхід передбачає порівняння всіх полів товару. Система зчитує позицію з нового XML/YML-файлу, знаходить відповідний запис у попередньому стані та порівнює назву, бренд, категорію, характеристики, ціну, наявність, кількість, зображення та інші параметри.

Перевага такого підходу — висока чутливість до змін, зокрема редагування опису або зображення. Проте не кожна зміна має однакову бізнес-вагу: опис може бути важливим для картки товару, але не завжди потребує негайного оновлення в

тому самому циклі, що й ціна або наявність. До того ж порівняння великої кількості полів збільшує обсяг обробки.

Четвертий підхід полягає у хешуванні контрольованих полів. У цій роботі хешування використовується для формування компактного контрольного значення з ціни, наявності та кількості. Якщо хоча б одне з цих значень змінюється, змінюється і дайджест, тому товар класифікується як оновлений; якщо дайджест збігається з попереднім, запис можна не оновлювати. У базовій реалізації використовується алгоритм MD5, описаний у RFC 1321 як такий, що формує 128-бітний дайджест повідомлення. У межах роботи MD5 застосовується не як засіб криптографічного захисту, а як технічний механізм компактного порівняння контрольованих полів [17]. За потреби хеш-функцію можна замінити без зміни загальної логіки методу.

Порівняння за хешем зводить перевірку змін контрольованих полів до зіставлення попереднього й нового контрольних значень, без окремого зберігання кожного попереднього значення поля. Ідея контрольних сум для виявлення відмінностей між версіями даних застосовується і в алгоритмах синхронізації, зокрема в rsync, де вони використовуються для пошуку збіжних і відмінних блоків [16]. Такий підхід не залежить від modified_date та не потребує універсального XML-diff алгоритму.

Важливим етапом є нормалізація значень перед хешуванням. Наприклад, ціна може бути подана як 100, 100.0 або 100.00; поле наявності — як true, 1 або текстовий еквівалент; кількість може бути відсутньою або дорівнювати нулю. Без нормалізації система може фіксувати зміни там, де з погляду бізнес-логіки їх немає, тому значення потрібно приводити до єдиного формату.

Хешування контрольованих полів доцільне для підтримання актуального операційного стану каталогу. Ціна, наявність і кількість безпосередньо впливають на продаж і виконання замовлення; якщо змінюється хоча б один із цих параметрів, локальний каталог має бути оновлений. Зміни описових полів можуть оброблятися окремо або не входити до базового циклу синхронізації.

Обмеження підходу полягає в тому, що він виявляє зміни лише в полях, включених до контрольного хешу. Якщо потрібно контролювати опис, зображення або категорію, ці поля слід додати до контрольованого набору або обробляти окремо. Точність підходу означає коректне виявлення змін у межах обраних контрольованих полів, а не всіх можливих змін у картці товару.

Для практичної реалізації потрібно врахувати появу та зникнення товарів. Хешування визначає зміну вже відомої позиції, а нові й відсутні товари визначаються за унікальним ідентифікатором. Якщо ідентифікатор є у новому XML-фіді, але відсутній у локальній базі, товар є новим; якщо ідентифікатор був у попередньому стані, але зник із нової версії фіду, товар позначається як видалений або недоступний залежно від бізнес-логіки.

Узагальнену логіку інкрементального виявлення змін у XML/YML-каталозі наведено на рисунку 1.3.

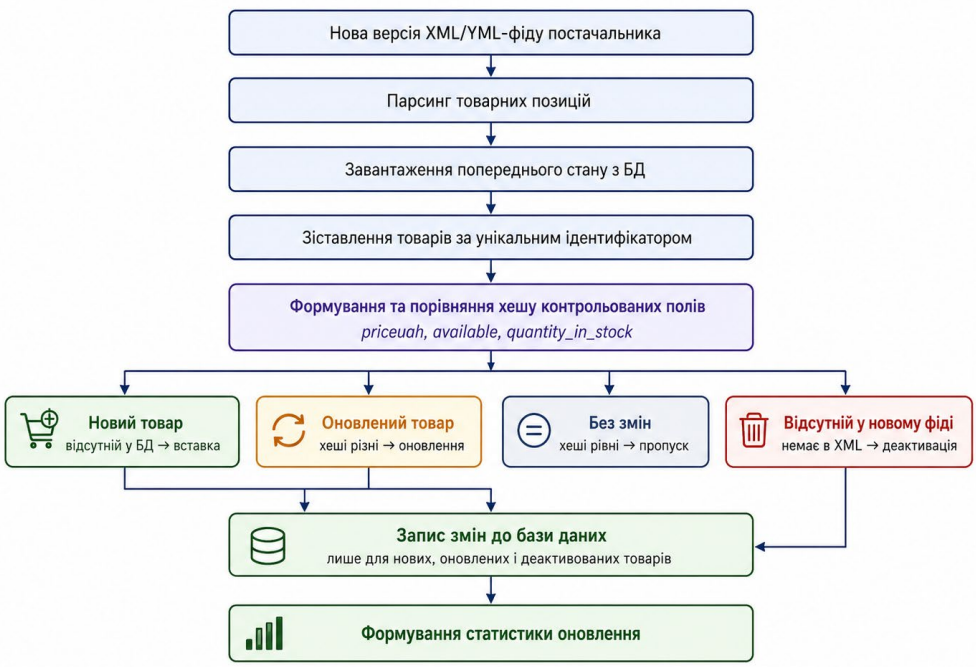

Рисунок 1.3 — Узагальнена логіка інкрементального виявлення змін у XML/YML-каталозі

Як видно з рисунка 1.3, запис до бази даних виконується лише для нових, оновлених і деактивованих товарів, тоді як незмінені позиції пропускаються.

У таблиці 1.4 наведено порівняння технічних підходів до виявлення змін у XML-каталозі.

Таблиця 1.4 — Порівняння технічних підходів до виявлення змін у XML-каталозі

Підхід	Що порівнюється	Залежність від постачальника	Орієнтовна складність	Доцільність для задачі роботи	Обмеження
Повне XML-diff порівняння	Структура XML-дерева, вузли, атрибути, вставки, видалення та переміщення.	Ні.	Поліноміальна; залежить від реалізації та розміру дерева.	Надлишкове для каталогу зі стабільною структурою.	Дає більше інформації, ніж потрібно для оновлення ціни, наявності та кількості.
Порівняння за modified_date	Службова дата останньої зміни товару.	Так, потребує наявності та коректного оновлення поля.	Лінійна відносно кількості товарів або нижча за наявності фільтрації.	Обмежено придатне; залежить від якості метаданих.	Не застосовується, якщо поле відсутнє або оновлюється непослідовно.
Порівняння всіх полів товару	Усі доступні атрибути та елементи товарної позиції.	Ні.	$O(n \cdot m)$, де n — кількість товарів, m — кількість полів.	Можливе, але може бути надлишковим.	Фіксує навіть зміни, які не належать до контрольованого набору для оперативного продажу.
Хешування контрольованих полів	Хеш значень ціни, наявності та кількості.	Ні.	$O(n)$ за умови зіставлення товарів за унікальним ідентифікатором.	Доцільно для задачі роботи.	Виявляє зміни лише в полях, включених до контрольного хешу.

Таблиця 1.4 показує, що для задачі роботи збалансованим є хешування контрольованих полів: воно не потребує зміни формату фіду, не залежить від службової дати й не виконує повного структурного порівняння XML-дерева.

Технічно підхід передбачає зберігання попереднього стану товару або принаймні його ідентифікатора й контрольного хешу. Під час синхронізації зчитується новий стан товару, формується новий хеш контрольованих полів і порівнюється з попереднім. Якщо хеші відрізняються, товар додається до набору оновлених позицій, а система виконує відповідні операції запису.

Підхід також створює основу для історії змін: для оновленого товару можна зафіксувати попередні й нові значення ціни, наявності та кількості, час синхронізації й тип події. Це дає змогу не лише оновити каталог, а й сформувати журнал змін для контролю коректності оновлень.

Отже, технічний аналіз підтверджує доцільність методу, що поєднує зіставлення товарів за унікальним ідентифікатором, визначення нових і відсутніх позицій та порівняння контрольованих полів через хешування. Математична модель, структури даних і детальний алгоритм такого методу розглядаються у Розділі 2.

1.6. Обґрунтування вибору EasyDrop як джерела XML-даних

Для реалізації та перевірки методу потрібне практичне джерело XML-даних, що відповідає предметній області дропшипінгу та містить поля для аналізу змін. У межах роботи таким джерелом обрано XML-каталог EasyDrop. Це не означає прив’язку методу лише до цієї платформи: EasyDrop використовується як практичний приклад зовнішнього джерела товарних даних.

EasyDrop належить до предметної області дропшипінгу, оскільки позиціонується як CRM-середовище для взаємодії постачальників, продавців, складів і комунікаційних процесів [7]. У такому середовищі актуальність каталогу має практичне значення, адже ціни, доступність і залишки можуть змінюватися незалежно від локального веб-каталогу продавця.

Використаний XML-фід має структуру, придатну для перевірки методу: у ньому виділено унікальний ідентифікатор товару, назву, ціну, наявність, кількість, бренд, зображення, group_id та інші характеристики. Ідентифікатор дозволяє

зіставляти позиції між версіями, а ціна, доступність і кількість формують набір контрольованих полів для визначення зміни стану товару.

EasyDrop у межах роботи розглядається саме як зовнішнє джерело даних. Завдання полягає не у дослідженні внутрішніх алгоритмів платформи, а у побудові локального механізму, який отримує XML-фід, аналізує його як нову версію каталогу та визначає зміни відносно попереднього стану локальної бази даних.

Метод може бути адаптований до інших XML/YML-каталогів зі схожою структурою. Головна умова адаптації — наявність стабільного унікального ідентифікатора товару та полів, які можна визначити як контрольовані. Якщо ціна, наявність або кількість мають інші назви, вони можуть бути зіставлені з внутрішніми полями системи під час парсингу.

Якщо у фіді відсутнє поле quantity, але наявне лише available, базовий варіант методу може бути спрощений до контролю ціни та статусу доступності. Якщо ж бізнес-логіка потребує відстеження опису, зображення або категорії, ці поля можуть бути додані до контрольованого набору. Це підтверджує гнучкість методу щодо різних XML/YML-структур.

Використання EasyDrop дозволяє перевірити повний цикл методу: отримання нової версії фіду, парсинг товарних позицій, зіставлення за ідентифікатором, формування хешу контрольованих полів і визначення нових, оновлених, незмінних або відсутніх товарів. Такий цикл відповідає практичній задачі підтримання актуального дропшипінг-каталогу.

Отже, вибір EasyDrop обґрунтований предметною областю, структурою даних і можливістю практичної перевірки методу. Водночас метод не обмежується EasyDrop і може використовуватися для інших XML/YML-фідів із потрібними ідентифікаційними та контрольованими полями.

1.7. Постановка задачі дослідження

Проведений аналіз показав, що оновлення дропшипінг-каталогу не зводиться до простого завантаження XML/YML-файлу. Для ефективної роботи веб-каталогу потрібно визначати появу, зникнення та зміну товарів, а також зберігати попередній стан для порівняння з новою версією фіду.

Для досягнення поставленої мети необхідно виконати такі завдання:

1) проаналізувати особливості обміну товарними даними в дропшипінг-моделі електронної комерції;

2) дослідити структуру XML/YML-каталогів товарів та визначити контрольовані поля для синхронізації;

3) проаналізувати існуючі підходи до оновлення товарних каталогів і виявлення змін у XML-даних;

4) обґрунтувати доцільність інкрементального підходу до синхронізації каталогу;

5) розробити математичну модель і алгоритм визначення змін товарів на основі хешування контрольованих полів;

6) розробити концептуальну модель даних і структури даних, необхідні для підтримки запропонованого методу;

7) реалізувати програмну веб-систему з серверною частиною, базою даних, REST API та клієнтським інтерфейсом;

8) провести тестування системи на даних XML-каталогу EasyDrop;

9) оцінити ефективність запропонованого підходу за кількістю уникнених операцій запису та часом виконання синхронізації.

Перше завдання пояснює предметну область і вимоги дропшипінгу до актуальності товарних даних. Друге випливає з потреби визначити структуру XML/YML-фіду та контрольовані поля. Третє й четверте пов’язані з аналізом підходів до оновлення каталогу та вибором інкрементальної синхронізації як основи методу.

П’яте та шосте завдання належать до Розділу 2, де розробляються математична модель, алгоритм, структури даних і концептуальна модель зберігання результатів синхронізації. Сьоме, восьме й дев’яте завдання реалізуються у Розділі 3, де метод перевіряється у веб-системі на XML-даних EasyDrop та оцінюється за кількістю уникнених операцій запису і часом виконання синхронізації.

Практична цінність роботи полягає у створенні механізму синхронізації, який може бути реалізований у веб-системі для автоматизованого оновлення

дропшипінг-каталогу. Метод дає змогу зменшити кількість операцій запису, зберігати історію змін і підтримувати актуальність локального каталогу без повного перезапису всіх товарів.

Отже, постановка задачі випливає з аналізу предметної області. Розділ 1 визначає проблему, обґрунтовує використання XML/YML-каталогів, показує обмеження наявних підходів і підводить до розроблення математичної моделі та алгоритму в Розділі 2.

Висновки до розділу 1

Проведений аналіз предметної області, структури XML/YML-каталогів, підходів до оновлення товарних даних і технічних методів виявлення змін дозволяє сформулювати такі висновки:

1. Дропшипінг-модель потребує регулярного оновлення товарних даних, оскільки продавець не контролює склад постачальника безпосередньо, а ціна, наявність і кількість товарів можуть змінюватися з боку зовнішнього джерела. Неактуальність цих даних може призводити до помилок у ціноутворенні, скасування замовлень, збільшення ручного адміністрування та зниження довіри клієнтів.

2. XML/YML-каталоги є поширеним способом передавання структурованої товарної інформації між постачальниками, маркетплейсами, CRM-системами, SaaS/CMS-платформами та інтернет-магазинами. Вони дозволяють описувати товарні позиції через набір полів, серед яких особливе значення для синхронізації мають унікальний ідентифікатор, ціна, наявність і кількість.

3. Українська електронна комерція використовує різні джерела товарних даних: маркетплейси, SaaS/CMS-платформи, дропшипінг CRM-платформи, агрегатори постачальників і товарні файли у форматах XML/YML, XLSX або CSV. Розгляд цих рішень підтверджує практичну актуальність задачі, але не змінює головного фокусу роботи — розроблення методу інкрементальної синхронізації.

4. Повний перезапис каталогу є простим, але надлишковим підходом, оскільки обробляє всі товари незалежно від того, чи відбулися в них фактичні зміни. Для великих каталогів це збільшує кількість операцій запису, ускладнює

збереження історії змін і не дозволяє ефективно відокремити змінені позиції від незмінених.

5. Рішення на основі modified_date можуть бути ефективними лише за умови, що постачальник передає відповідне поле та коректно оновлює його при кожній суттєвій зміні товару. Якщо такого поля немає або його значення ненадійні, система потребує власного механізму порівняння версій каталогу.

6. Для товарних XML/YML-каталогів доцільно застосувати інкрементальний підхід, який виявляє зміни у ключових контрольованих полях. Хешування ціни, наявності та кількості дозволяє визначати оновлені товари без повного XML-diff порівняння та без залежності від метаданих постачальника. При цьому підхід коректно виявляє зміни саме в межах полів, включених до контрольного хешу.

7. EasyDrop обрано як практичне джерело XML-даних для реалізації та перевірки запропонованого підходу, оскільки платформа належить до предметної області дропшипінгу, а використаний у роботі XML-фід містить потрібні для синхронізації поля. Водночас метод не є жорстко прив’язаним до EasyDrop і може бути адаптований до інших XML/YML-каталогів зі схожою структурою.

8. Проведений аналіз створює основу для Розділу 2, у якому розробляються математична модель, алгоритм інкрементальної синхронізації, структури даних і концептуальна модель зберігання результатів синхронізації.

РОЗДІЛ 2
РОЗРОБЛЕННЯ МЕТОДУ ІНКРЕМЕНТАЛЬНОЇ СИНХРОНІЗАЦІЇ ДРОП-ШИППІНГ-КАТАЛОГУ

2.1. Формалізація задачі синхронізації версійного XML/YML-потоку

У першому розділі обґрунтовано, що товарний каталог у дропшипінг-сценарії надходить від зовнішнього джерела у вигляді XML/YML-фіду. Такий фід не є набором окремих операцій додавання, оновлення або деактивації товарів. Він розглядається як чергова повна версія стану каталогу, яку потрібно зіставити з попереднім локальним станом.

Перед формальним описом методу необхідно визначити основні об’єкти, які беруть участь у синхронізації. Нова версія XML/YML-фіду розглядається як актуальний стан товарного каталогу на певний момент часу, а локальна база даних — як попередній стан, з яким цей фід порівнюється. Результатом такого порівняння є набір змін, що показує, які товари потрібно додати, оновити, залишити без змін або позначити як відсутні у новому фіді.

У межах цього розділу версійний XML/YML-потік подається як послідовність актуалізованих версій каталогу: $X_1, X_2, \ldots, X_k$. Кожна версія X_k містить множину товарних позицій P_k. Локальна база даних перед виконанням k-го циклу синхронізації зберігає попередній стан каталогу D_{k-1}. Результатом циклу є множина класифікованих змін R_k і оновлений стан локального каталогу D_k.

$$X = \{X_1, X_2, \ldots, X_k\} \quad (2.1)$$

де X — послідовність версій XML/YML-каталогу; X_k — k-та актуалізована версія фіду; k — номер циклу синхронізації.

$$P_k = \{p_1, p_2, \ldots, p_n\} \quad (2.2)$$

де P_k — множина товарних позицій, отриманих із версії X_k; p_i — окрема товарна позиція; n — кількість товарів у новій версії фіду.

Задача синхронізації полягає не у простому завантаженні X_k до локальної бази, а у визначенні різниці між новою версією фіду та станом D_{k-1}. Це пов’язано з тим, що зовнішній фід надходить як повний знімок каталогу, не містить надійного

службового поля modified_date та не надає універсального дельта-інтерфейсу. Тому зміни мають визначатися власним алгоритмом порівняння.

$R_k = compare(X_k, D_{k-1})$ (2.3)

де R_k — результат k-го циклу синхронізації; compare — операція порівняння нової версії XML/YML-фіду з попереднім локальним станом; D_{k-1} — стан локальної бази даних перед синхронізацією. Множина R_k містить результати класифікації товарів на нові, оновлені, незмінені та відсутні у новій версії фіду.

Після виконання циклу синхронізації локальний стан D_k формується на основі попереднього стану D_{k-1} і результатів порівняння R_k. При цьому незмінені товари не потребують операцій запису, тоді як нові, оновлені та відсутні у фіді товари мають бути зафіксовані відповідно до правил методу.

$D_k = update(D_{k-1}, R_k)$ (2.4)

де update — логічна операція оновлення локального стану каталогу за результатами порівняння. У цій роботі фізичне видалення товарів не вважається базовою дією: товари, що відсутні у новому фіді, доцільно позначати як неактивні або недоступні для збереження історії.

2.2. Модель товарної позиції та контрольованих полів

Товарна позиція XML/YML-каталогу розглядається як структурований об’єкт, що містить ідентифікаційні, контрольовані, описові та допоміжні поля. Для запропонованого методу принципово важливими є не всі поля товару, а ті, що дають змогу зіставити позицію між версіями каталогу й визначити зміну її операційного стану.

$p_i = \{id_i, group\_id_i, name_i, price_i, available_i, quantity_i, description_i, image_i, params_i\}$ (2.5)

де p_i — i-та товарна позиція; id_i — унікальний ідентифікатор товару; $group\_id_i$ — ідентифікатор групи варіантів; $name_i$ — назва; $price_i$ — ціна; $available_i$ — статус доступності; $quantity_i$ — кількість; $description_i$, $image_i$, $params_i$ — описові характеристики товару.

Ідентифікаційні поля використовуються для зіставлення товарів між новою версією фіду та попереднім станом бази даних. Найважливішим з них є id_i, оскільки

саме стабільний ідентифікатор дозволяє встановити, що товар із нового фіду відповідає певному локальному запису. Поле $group\_id_i$ має допоміжний характер і може використовуватися для збереження зв'язку між варіантами однієї моделі.

Контрольований стан товару визначається трьома параметрами, безпосередньо пов'язаними з можливістю продажу: ціною, статусом наявності та кількістю. Саме ці поля є основою для визначення того, чи змінився операційний стан товарної позиції.

$$C_i = (price_i, available_i, quantity_i) \quad (2.6)$$

де C_i — контрольований стан товару p_i; $price_i$ — ціна; $available_i$ — ознака доступності; $quantity_i$ — кількість товару. Якщо змінюється хоча б один елемент C_i, товар потребує оновлення в локальній базі даних.

Таблиця 2.1 — Класифікація полів товарної позиції для задачі синхронізації

Група полів	**Приклади полів**	**Призначення**	**Роль у методі**
Ідентифікаційні	id, external_id, item id	Однозначне визначення товарної позиції	Використовуються для зіставлення товарів між фідом і локальною базою
Допоміжні структурні	group_id, category_id	Збереження зв'язків між варіантами та категоріями	Допомагають підтримувати структуру каталогу, але не завжди входять до контрольного стану
Контрольовані	price / priceuah, available, quantity / quantity_in_stock	Визначення операційного стану товару	Використовуються для формування контрольного хешу та виявлення оновлень
Описові	name, description, image, params, barcode	Формування картки товару та опису	Можуть зберігатися в базі, але не є базовою умовою оперативної синхронізації

Розмежування груп полів дозволяє уникнути надлишкового оновлення товарів через зміни, які не впливають безпосередньо на продаж. Наприклад, зміна опису або зображення може бути важливою для повноти картки товару, але в межах базового методу вона не прирівнюється до зміни ціни, наявності або кількості.

2.3. Математична модель визначення змін між версіями каталогу

Для класифікації товарів використовується порівняння множини ідентифікаторів у новій версії фіду з множиною ідентифікаторів у попередньому локальному стані. Нехай $ID(X_k)$ — множина ідентифікаторів товарів у новій версії XML/YML-фіду, а $ID(D_{k-1})$ — множина ідентифікаторів товарів у локальній базі даних перед синхронізацією.

$$P\_new = ID(X_k) \setminus ID(D_{k-1}) \quad (2.7)$$

де P_new — множина нових товарів, тобто таких, ідентифікатори яких є у новому XML/YML-фіді, але відсутні у попередньому локальному стані.

$$P\_missing = ID(D_{k-1}) \setminus ID(X_k) \quad (2.8)$$

де P_missing — множина товарів, які були у локальній базі даних перед синхронізацією, але відсутні у новій версії фіду. Такі товари доцільно позначати як відсутні у новому фіді або деактивовані.

$$P\_common = ID(X_k) \cap ID(D_{k-1}) \quad (2.9)$$

де P_common — множина товарів, що існують і в новій версії фіду, і в попередньому локальному стані. Саме для цих товарів потрібно визначити, чи змінився контрольований стан.

$$P\_updated = \{p_i \in P\_common \mid H_k(p_i) \neq H_{k-1}(p_i)\} \quad (2.10)$$

де P_updated — множина оновлених товарів; $H_k(p_i)$ — контрольний хеш товару в новій версії фіду; $H_{k-1}(p_i)$ — попереднє контрольне значення, збережене в локальній базі даних.

$$P\_unchanged = \{p_i \in P\_common \mid H_k(p_i) = H_{k-1}(p_i)\} \quad (2.11)$$

де P_unchanged — множина товарів без змін. Якщо контрольний хеш не змінився, операційний стан товару в межах обраних контрольованих полів вважається незмінним.

$$R_k = \{P\_new, P\_updated, P\_unchanged, P\_missing\} \quad (2.12)$$

Отже, кожен товар у межах циклу синхронізації належить лише до одного з чотирьох станів: новий, оновлений, без змін або відсутній у новому фіді. Така класифікація є взаємовиключною, оскільки вона базується на наявності ідентифікатора у відповідних множинах і на результаті порівняння контрольного хешу.

Отримані множини результатів класифікації є попарно неперетинними, оскільки кожна товарна позиція в межах одного циклу синхронізації може належати лише до одного стану: $P\_new \cap P\_updated = \emptyset$, $P\_new \cap P\_unchanged = \emptyset$, $P\_updated \cap P\_unchanged = \emptyset$, $P\_missing \cap P\_common = \emptyset$. Це забезпечує однозначність подальшої дії системи для кожної товарної позиції: додавання, оновлення, пропуск або деактивацію.

2.4. Формування контрольного хешу товарної позиції

Для компактного порівняння контрольованого стану товару використовується контрольний хеш. Його перевага полягає в тому, що попередній стан контрольованих полів можна зберігати як одне компактне значення, а не як окремий набір попередніх значень кожного поля. Для запобігання неоднозначності конкатенації нормалізовані значення контрольованих полів об’єднуються із використанням службового роздільника.

$$S_i = \text{norm}(\text{price}_i) \,||\, "|" \,||\, \text{norm}(\text{available}_i) \,||\, "|" \,||\, \text{norm}(\text{quantity}_i) \quad (2.13)$$

де S_i — нормалізований рядок контрольованого стану товару; символ "|" використовується як роздільник між значеннями контрольованих полів.

Нормалізація потрібна для уникнення помилкової класифікації змін. Наприклад, значення 100, 100.0 і 100.00 мають трактуватися як однакова ціна; значення true, 1 або інше еквівалентне подання доступності мають бути приведені до єдиного формату; відсутнє значення кількості та нуль мають оброблятися за заздалегідь визначеним правилом.

$$H_i = \text{hash}(S_i) \quad (2.14)$$

де H_i — контрольний хеш товару p_i; hash — функція формування дайджесту на основі нормалізованого рядка контрольованого стану.

У базовій реалізації для функції hash використовується MD5, описаний у RFC 1321 як алгоритм формування 128-бітного дайджесту повідомлення [17]. У межах цієї роботи хеш використовується як контрольне значення для порівняння станів товарної позиції, а не як засіб криптографічного захисту. Тому MD5 не розглядається як механізм безпеки, цифрового підпису або перевірки цілісності в

криптографічному сенсі. За потреби функція hash може бути замінена іншою хеш-функцією без зміни загальної логіки методу.

2.5. Алгоритм інкрементальної синхронізації каталогу

Запропонований метод передбачає повне читання нової версії XML/YML-фіду, але не передбачає повного перезапису локального каталогу. Усі товари нового фіду аналізуються, однак операції запису виконуються лише для нових, оновлених і деактивованих товарів. Це є важливим уточненням: метод не зменшує необхідність прочитати фід, але зменшує кількість операцій зміни локальної бази даних.

Основний алгоритм складається з таких етапів: отримання нової версії XML/YML-фіду; парсинг товарних позицій; завантаження попереднього стану з локальної бази; побудова індексу за ідентифікатором; обхід товарів нового фіду; формування контрольного хешу; класифікація товарів; визначення відсутніх позицій; запис результатів синхронізації.

Алгоритм 2.1 — Інкрементальна синхронізація XML/YML-каталогу

Вхідні дані: X_k — нова версія XML/YML-фіду; D_{k-1} — попередній стан локальної бази даних.

Вихідні дані: P_new, P_updated, P_unchanged, P_missing, оновлений стан D_k.

1. $P_k \leftarrow \text{parse}(X_k)$
2. D_prev ← loadPreviousState(D_{k-1})
3. M_prev ← buildIndex(D_prev, key = id)
4. S_current ← порожня множина
5. P_new, P_updated, P_unchanged, P_missing ← порожні множини
6. for each product p in P_k do
7. додати p.id до S_current
8. C_new ← (norm(p.price), norm(p.available), norm(p.quantity))
9. H_new ← hash(C_new)
10. if p.id not in M_prev then
11. p.hash ← H_new
12. додати p до P_new

```
13.    else
14.        p_prev ← M_prev[p.id]
15.        if H_new ≠ p_prev.hash then
16.            p.hash ← H_new
17.            додати p до P_updated
18.        else
19.            додати p до P_unchanged
20.        end if
21.    end if
22. end for
23. for each product q in D_prev do
24.     if q.id not in S_current then
25.         q.status ← DEACTIVATED
26.         додати q до P_missing
27.     end if
28. end for
29. writeChanges(P_new, P_updated, P_missing)
30. saveSyncResult(P_new, P_updated, P_unchanged, P_missing)
31. D_k ← update(D_{k−1}, R_k)
32. return R_k, D_k
```

У наведеному алгоритмі parse позначає логічну операцію перетворення XML/YML-фіду на множину товарних позицій, buildIndex формує відображення ідентифікатора товару на попередній стан, C_new позначає нормалізований контрольований стан товару, а writeChanges виконує запис лише для нових, оновлених і деактивованих позицій. Для нових і оновлених товарів разом із записом зберігається нове контрольне значення H_new, що використовується у наступному циклі синхронізації.

Порядок виконання етапів у алгоритмі має принципове значення. Спочатку формується індекс попереднього стану M_prev, оскільки саме він забезпечує швидке зіставлення товарів за ідентифікатором. Без такого індексу для кожної

позиції нового фіду довелося б виконувати пошук серед усіх записів локального каталогу, що створювало б надлишкові операції порівняння.

Контрольний хеш H_new формується для кожного товару з нового фіду, зокрема і для нових позицій. Для наявних товарів це значення використовується для порівняння з попереднім хешем, а для нових товарів — зберігається як початковий контрольний стан, який буде використаний у наступних циклах синхронізації. Тому хешування є не окремою допоміжною дією, а частиною механізму переходу від одного стану каталогу до наступного.

Товари, відсутні у новій версії фіду, визначаються після завершення обходу всіх позицій X_k. Це пов’язано з тим, що лише після формування множини S_current можна коректно встановити, які товари були присутні у попередньому стані D_{k-1}, але не потрапили до нової версії каталогу. Саме тому деактивація відсутніх позицій винесена в окремий завершальний етап алгоритму.

Узагальнену блок-схему запропонованого алгоритму інкрементальної синхронізації наведено на рисунку 2.1.

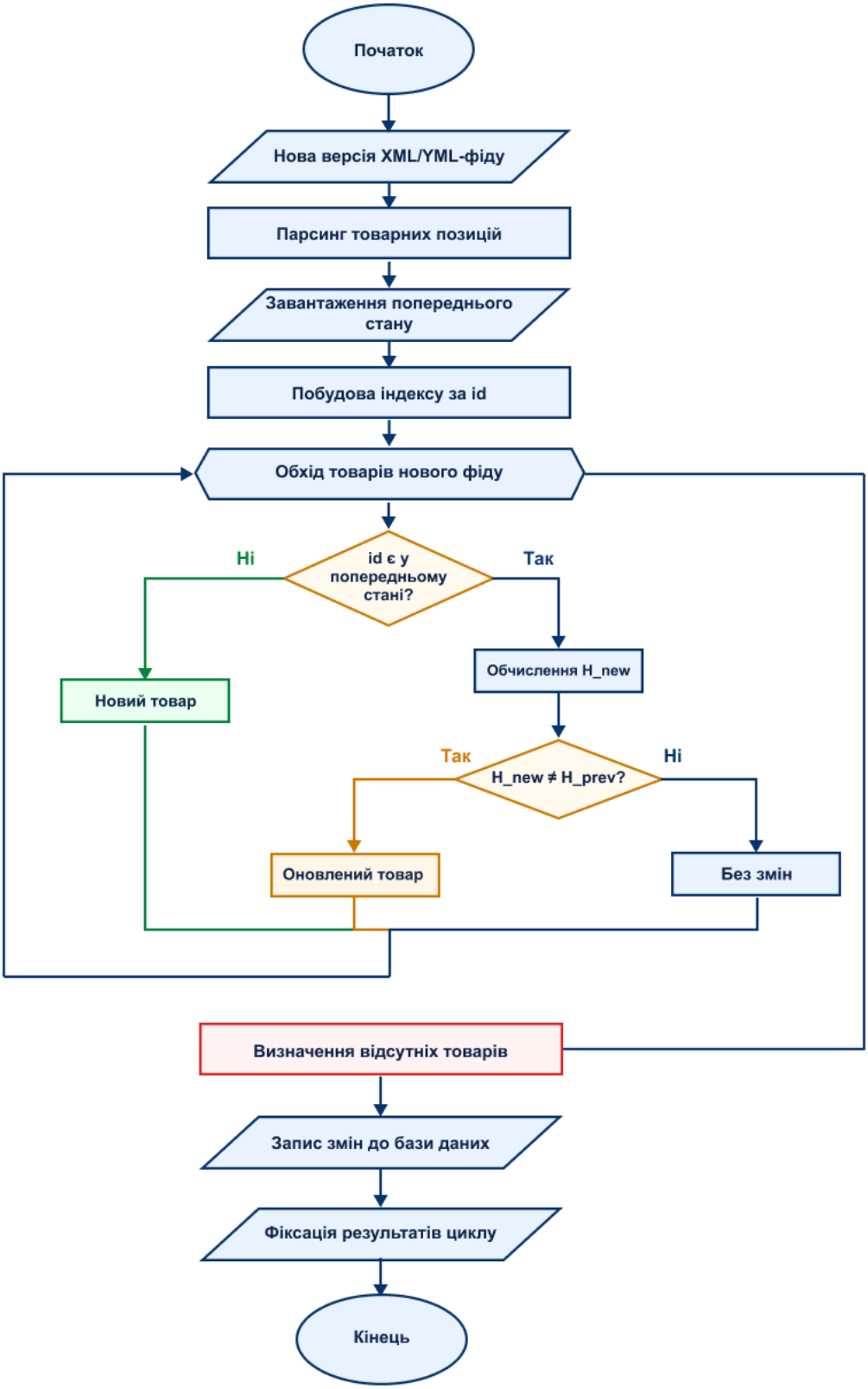

Рисунок 2.1 — Блок-схема алгоритму інкрементальної синхронізації XML/YML-каталогу

Як видно з рисунка 2.1, алгоритм аналізує всі позиції нового XML/YML-фіду, але операції запису виконуються лише для нових, оновлених і деактивованих товарів. Незмінені позиції враховуються у статистиці циклу, але не потребують оновлення запису в базі даних.

2.6. Алгоритм класифікації товарів за результатами порівняння

Класифікація товарів є окремим логічним етапом синхронізації. Вона визначає, які дії мають бути виконані після порівняння нової версії фіду з попереднім станом локальної бази даних. У методі передбачено чотири взаємовиключні стани товару.

Таблиця 2.2 — Правила класифікації товарів під час синхронізації

Стан товару	Умова визначення	Дія системи	Значення для історії змін
Новий товар	id є у новому фіді, але відсутній у попередньому стані	Додати товар до локального каталогу та зберегти його контрольний хеш	Фіксується подія NEW
Оновлений товар	id є у фіді та базі, але контрольний хеш змінився	Оновити контрольовані поля та новий хеш	Фіксується подія UPDATED із попередніми та новими значеннями
Товар без змін	id є у фіді та базі, а контрольний хеш не змінився	Не виконувати операцію запису для товару	Може враховуватися лише у статистиці циклу
Відсутній у новому фіді / деактивований	id був у попередньому стані, але відсутній у новому фіді	Позначити товар як неактивний або недоступний	Фіксується подія DEACTIVATED або DELETED без фізичного видалення

Товар, відсутній у новій версії фіду, не обов’язково має фізично видалятися з бази даних. Для збереження історії коректніше позначати його як неактивний або недоступний. Такий підхід дозволяє відновити інформацію про попередній стан товару, сформувати статистику деактивацій і уникнути втрати історичних записів.

$$\text{status}(p_i) \in \{\text{NEW}, \text{UPDATED}, \text{UNCHANGED}, \text{MISSING}\} \quad (2.15)$$

Формула (2.15) відображає множину допустимих станів товару за результатами одного циклу синхронізації. Наявність чітких правил класифікації дозволяє відокремити етап порівняння від етапу запису змін до бази даних.

2.7. Структури даних для порівняння станів каталогу

Ефективність запропонованого методу залежить від способу організації даних під час порівняння. Якщо кожен товар із нового фіду порівнювати з кожним товаром попереднього стану, алгоритм матиме квадратичний характер. Для уникнення цього використовується індекс попереднього стану за унікальним ідентифікатором.

$$M\_prev[id] \rightarrow p\_prev \quad (2.16)$$

де M_prev — індекс попереднього стану; id — зовнішній ідентифікатор товару; p_prev — відповідний товар із локальної бази даних. Такий індекс дає змогу швидко перевірити, чи існував товар раніше, і отримати його попередній хеш.

$$S\_current = \{id_1, id_2, \ldots, id_n\} \quad (2.17)$$

де S_current — множина ідентифікаторів товарів, що містяться у новій версії XML/YML-фіду. Вона використовується для визначення позицій, які були в попередньому стані, але відсутні в новій версії фіду.

Крім індексу та множини ідентифікаторів, алгоритм використовує множини результатів: P_new, P_updated, P_unchanged і P_missing. Їхнє призначення полягає у відокремленні різних типів результатів порівняння до моменту запису змін у базу даних.

Використання індексу за ідентифікатором дозволяє уникнути повного попарного порівняння. Без індексу для n товарів нового фіду та m товарів попереднього стану можливе порівняння $O(n \cdot m)$. За наявності індексу пошук товару за ідентифікатором у середньому виконується за $O(1)$, а загальний прохід виконується за $O(n + m)$.

2.8. Обґрунтування вибору типу моделі даних для методу синхронізації

Після формалізації алгоритму необхідно визначити, яка модель даних найкраще відповідає задачі інкрементальної синхронізації. Це питання розглядається до вибору конкретної системи керування базами даних або мови програмування, оскільки тип моделі даних впливає на спосіб зберігання поточного стану товарів, історії циклів синхронізації та журналу змін.

Для запропонованого методу база даних виконує не лише роль сховища поточного каталогу. Вона також забезпечує зіставлення нової версії XML/YML-фіду з попереднім станом, збереження контрольного хешу, фіксацію появи нових товарів, оновлення контрольованих полів і деактивацію товарів, відсутніх у новій версії фіду.

Під час вибору типу моделі даних для такого методу доцільно враховувати наявність стабільного унікального ідентифікатора товару, повторювану структуру контрольованих полів, потребу зберігати зв’язок між товаром і циклами синхронізації, необхідність фіксувати попередні та нові значення контрольованих полів, а також можливість формувати статистику за результатами кожного запуску алгоритму.

XML/YML-фід має ієрархічну природу, тому для зберігання повного документа могла б розглядатися документно-орієнтована модель. Водночас запропонований метод не потребує постійного зберігання повних XML-файлів: його основою є структуроване зіставлення товарів за ідентифікатором, порівняння контрольованих полів і журналювання змін.

Таблиця 2.3 — Порівняння типів моделей даних для задачі синхронізації каталогу

Тип моделі даних	**Можливості для задачі**	**Переваги**	**Обмеження**	**Доцільність для цієї роботи**
Реляційна модель	Зберігання товарів, циклів синхронізації та журналу змін у пов’язаних таблицях.	Підтримує структуровані поля, зв’язки між сутностями, цілісність даних і формування статистики.	Менш зручна для зберігання повністю довільної вкладеної структури XML без попереднього перетворення.	Найбільш доцільна для ядра методу, оскільки товари, цикли та зміни мають чіткі зв’язки.
Документно-орієнтована модель	Зберігання товарів або повних фрагментів фіду як документів.	Підходить для гнучких і вкладених структур, зокрема параметрів товару.	Складніше забезпечувати строгі зв’язки між товаром, циклом синхронізації та журналом змін.	Може бути корисною для додаткових описових параметрів, але не є оптимальною

Тип моделі даних	Можливості для задачі	Переваги	Обмеження	Доцільність для цієї роботи
				основою журналу змін.
Ключ-значення	Швидкий доступ до товару за ідентифікатором або збереження контрольного хешу.	Проста й швидка модель для перевірки наявності запису за ключем.	Має обмежені можливості для історії змін, аналітики та зв’язків між сутностями.	Доцільна як допоміжна структура індексації, але не як повна модель зберігання.
Графова модель	Опис складних зв’язків між товарами, категоріями, постачальниками та іншими об’єктами.	Корисна для задач із великою кількістю взаємозв’язків.	Надлишкова для базової задачі порівняння станів товарів за ідентифікатором.	Не є доцільною для базового методу інкрементальної синхронізації.

З аналізу таблиці 2.3 видно, що для базового ядра запропонованого методу найбільш доцільною є реляційна модель даних. Вона дозволяє відокремити поточний стан товару від історії циклів синхронізації та журналу змін, а також забезпечує зв’язки між цими сутностями через ідентифікатори.

На рівні розроблення методу для задачі інкрементальної синхронізації обрано структуровану реляційну концептуальну модель даних. Такий вибір зумовлений тим, що поточний стан товарів, цикли синхронізації та журнал змін мають чіткі зв’язки між собою й потребують збереження структурованих полів, ідентифікаторів, контрольних хешів та результатів класифікації змін. Конкретна програмна реалізація цієї моделі має відповідати визначеній у цьому розділі логіці зберігання даних і не змінювати математичну та алгоритмічну основу запропонованого методу.

Реляційна модель у цьому випадку є доцільною не через вибір конкретного програмного продукту, а через характер самих даних. Товар має стабільний зовнішній ідентифікатор і набір контрольованих полів; цикл синхронізації фіксує результат окремого запуску алгоритму; журнал змін пов’язує конкретний товар із конкретним циклом і зберігає попередні та нові значення. Така структура природно

описується зв’язками «один товар — багато змін» і «один цикл синхронізації — багато змін».

Окремо слід зазначити, що деякі описові або додаткові характеристики товару можуть мати змінну структуру. Наприклад, параметри товару з XML/YML-фіду можуть відрізнятися залежно від категорії. Тому в межах реляційної моделі допускається гнучке зберігання таких полів, якщо вони не є основою контрольного стану.

При цьому змінність описових параметрів не суперечить вибору реляційної концептуальної моделі, оскільки такі поля не визначають базовий контрольований стан товару. Їх можна зберігати як допоміжні атрибути або окрему групу параметрів, тоді як основна логіка синхронізації залишається прив’язаною до стабільних полів: ідентифікатора, ціни, доступності, кількості, контрольного хешу та статусу активності.

Таким чином, алгоритм інкрементальної синхронізації у цій роботі спирається на структуровану модель даних, у якій поточний стан товарів, цикли синхронізації та журнал змін є окремими, але пов’язаними сутностями.

2.9. Концептуальна модель бази даних для зберігання результатів синхронізації

Для зберігання результатів синхронізації потрібна концептуальна модель даних, яка відокремлює поточний стан каталогу, історію циклів синхронізації та журнал окремих змін. У цьому підрозділі описується логічна модель без прив’язки до конкретної системи керування базами даних або SQL-реалізації.

Таблиця 2.4 — Концептуальні сутності моделі даних

Сутність	Основні поля	Призначення	Зв’язок з методом синхронізації
Товар / поточний стан товару	внутрішній id, зовнішній id, назва, group_id, ціна, available, quantity, описові поля, hash, active, updated_at	Зберігає актуальний локальний стан товарної позиції	Використовується для зіставлення з новою версією фіду та порівняння контрольного хешу

Сутність	Основні поля	Призначення	Зв’язок з методом синхронізації
Цикл синхронізації	id циклу, дата запуску, дата завершення, кількість оброблених, нових, оновлених, незмінених і деактивованих товарів, тривалість, статус	Фіксує результат одного запуску синхронізації	Дозволяє аналізувати ефективність і результат кожного циклу
Журнал змін	id зміни, id товару, id циклу, тип зміни, попередні та нові значення контрольованих полів, дата фіксації	Зберігає історію змін контрольованого стану товарів	Забезпечує простежуваність змін між версіями XML/YML-потоку

Сутність «Товар» зберігає поточний стан локального каталогу. Вона повинна містити зовнішній ідентифікатор, контрольовані поля, описові поля, контрольний хеш і статус активності. Статус активності потрібний для випадків, коли товар відсутній у новому фіді, але його історію не потрібно втрачати.

Сутність «Цикл синхронізації» фіксує метадані одного запуску алгоритму. Вона дозволяє визначити, скільки товарів було прочитано, скільки позицій додано, оновлено, пропущено або деактивовано, а також чи завершився цикл успішно.

Сутність «Журнал змін» пов’язує конкретний товар із конкретним циклом синхронізації. Один товар може мати багато записів у журналі змін, а один цикл синхронізації може містити багато змін. Поточний стан товару оновлюється за результатами циклу, тоді як журнал зберігає історичну інформацію про зміну контрольованих полів.

Запропонована концептуальна модель дозволяє зберігати поточний стан каталогу, відстежувати історію синхронізацій, аналізувати кількість змін і не втрачати дані про деактивовані товари. Вона також підтримує ідею версійного XML-потоку, оскільки кожен цикл синхронізації відображає перехід від одного стану каталогу до іншого.

Концептуальну модель зберігання результатів синхронізації наведено на рисунку 2.2.

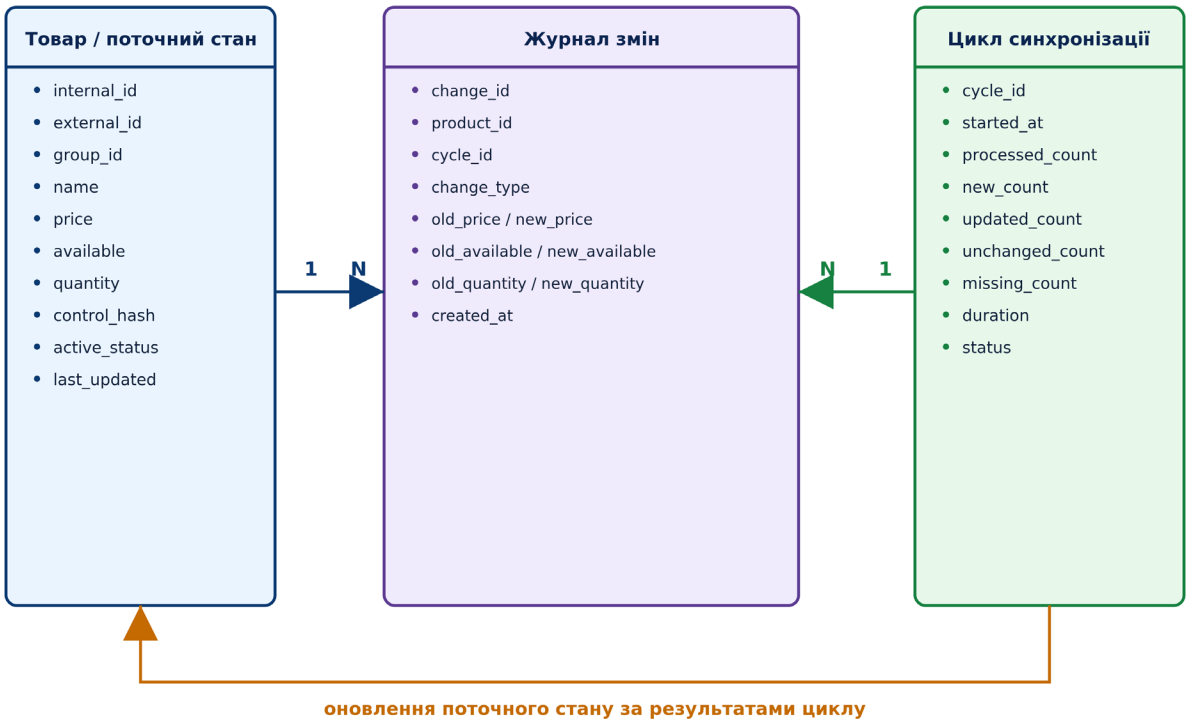

Рисунок 2.2 — Концептуальна модель даних для зберігання результатів синхронізації

Як видно з рисунка 2.2, сутність товару зберігає поточний стан каталогу, сутність циклу синхронізації фіксує результат окремого запуску алгоритму, а журнал змін забезпечує простежуваність переходів між послідовними версіями XML/YML-фіду.

2.10. Алгоритмічний каркас методу до етапу програмної реалізації

Після визначення математичної моделі, алгоритму класифікації та моделі даних доцільно описати алгоритмічний каркас майбутньої комп'ютерної системи. Такий каркас не є описом конкретного web-додатку або вибором мови програмування. Його призначення полягає в тому, щоб показати, з яких логічних компонентів складається запропоноване рішення до етапу практичної реалізації.

Ключова складова роботи полягає саме в алгоритмі обробки XML/YML-даних. Програмний додаток у подальшому лише реалізує цей алгоритм, забезпечує його запуск, зберігання результатів і відображення статистики. Тому на рівні Розділу 2 потрібно відокремити алгоритмічне ядро від майбутньої прикладної оболонки.

Отже, розроблення методу починається не з екранів користувача чи програмних бібліотек, а з визначення того, які дані надходять на вхід, які проміжні структури потрібні для порівняння та який результат має бути отриманий після кожного циклу синхронізації. Така послідовність відповідає логіці інженерного проєктування: спочатку формується алгоритмічний каркас, а потім він реалізується програмними засобами.

Узагальнений каркас методу складається з кількох логічних компонентів: отримання нової версії XML/YML-фіду, парсингу, нормалізації контрольованих полів, завантаження попереднього стану, порівняння та класифікації, запису результатів до моделі даних і формування статистики циклу.

Окремим компонентом є завантаження попереднього стану локального каталогу та побудова індексу за унікальним ідентифікатором товару. Саме цей індекс дозволяє перейти від надлишкового попарного порівняння до лінійного проходу по товарах.

Результат порівняння передається до компонента класифікації. Він визначає, до якого стану належить товар: новий, оновлений, незмінений або відсутній у новій версії фіду. Після класифікації компонент збереження результатів виконує запис лише для тих позицій, які потребують зміни локального стану.

Узагальнений алгоритмічний каркас методу до етапу програмної реалізації наведено на рисунку 2.3.

Рисунок 2.3 — Логічний каркас методу інкрементальної синхронізації до програмної реалізації

Як видно з рисунка 2.3, метод можна подати як послідовність логічних компонентів: отримання фіду, парсинг, нормалізація, порівняння, класифікація, збереження результатів і формування статистики. Такий поділ дозволяє реалізувати метод у вигляді програмної системи без зміни його математичної та алгоритмічної основи.

Таблиця 2.5 — Відповідність логічних компонентів методу етапам обробки даних

Логічний компонент	Вхідні дані	Основна дія	Вихідні дані	Значення для методу
Отримання фіду	URL або файл XML/YML-фіду	Отримання нової версії каталогу	XML/YML-документ	Забезпечує початкові дані для циклу синхронізації.
Парсинг	XML/YML-документ	Перетворення структури фіду на множину товарів	P_k	Надає уніфіковане представлення товарних позицій.
Нормалізація	price, available, quantity	Приведення значень до єдиного формату	Нормалізовані контрольовані поля	Зменшує ризик помилкового визначення змін.
Попередній стан	Поточний локальний каталог	Побудова індексу M_prev за id	M_prev	Забезпечує швидке зіставлення товарів між станами.
Порівняння	P_k та M_prev	Формування H_new і зіставлення з попереднім хешем	Ознака зміни контрольованого стану	Визначає факт оновлення товарної позиції.
Класифікація	Результати зіставлення id і хешів	Розподіл товарів за станами	P_new, P_updated, P_unchanged, P_missing	Визначає подальшу дію для кожного товару.
Збереження	Множини результатів класифікації	Запис нових, оновлених і деактивованих товарів	D_k та журнал змін	Забезпечує актуалізацію каталогу й історію.
Статистика	Результати циклу	Підрахунок кількості товарів за станами	Статистика циклу	Дозволяє оцінювати результат синхронізації.

Таблиця 2.5 показує, що запропонований метод має модульну логіку ще до вибору конкретних програмних засобів. Кожен компонент виконує окрему функцію й має чітко визначені вхідні та вихідні дані.

На цьому рівні мова програмування, бібліотеки та серверна платформа не є предметом розгляду, оскільки вони не змінюють логіку запропонованого методу. Будь-яка програмна реалізація має відтворювати визначені компоненти: отримання фіду, парсинг, нормалізацію контрольованих полів, індексацію попереднього стану, порівняння, класифікацію та фіксацію результатів синхронізації.

Такий поділ дозволяє оцінювати метод незалежно від прикладної оболонки. Якщо змінити інтерфейс, спосіб запуску або конкретну платформу реалізації,

алгоритмічна основа залишається незмінною: система все одно має отримати новий стан фіду, зіставити його з попереднім станом, визначити чотири групи результатів і зберегти тільки необхідні зміни.

Такий підхід відповідає логіці проєктування комп'ютерної системи: спочатку задається математична модель і алгоритм обробки даних, потім визначається модель зберігання, а після цього формується програмна архітектура додатку.

2.11. Оцінка обчислювальної складності запропонованого методу

Оцінка обчислювальної складності виконується для основних етапів алгоритму. Нехай n — кількість товарів у новій версії XML/YML-фіду, m — кількість товарів у попередньому стані локальної бази даних, c — кількість контрольованих полів. У базовому варіанті c = 3, оскільки використовуються ціна, наявність і кількість.

Оскільки кількість контрольованих полів є сталою, формування нормалізованого рядка та хешу для одного товару можна вважати операцією O(1). Основна складність алгоритму визначається кількістю товарів у фіді та кількістю записів у попередньому стані.

Таблиця 2.6 — Оцінка складності етапів алгоритму

Етап	Позначення	Складність	Пояснення
Парсинг нового фіду	P_k	O(n)	Кожна товарна позиція нового XML/YML-фіду читається один раз
Завантаження попереднього стану	D_{k-1}	O(m)	Зчитуються активні товари, що зберігаються в локальній базі
Побудова індексу за ідентифікатором	M_prev	O(m)	Кожен запис попереднього стану додається до індексу
Обхід товарів нового фіду	P_k	O(n)	Кожен товар нового фіду аналізується один раз
Формування та порівняння хешів	H_i	O(n)	Кількість контрольованих полів є сталою, тому хеш одного товару формується за O(1)

Етап	Позначення	Складність	Пояснення
Визначення відсутніх товарів	S_current	O(m)	Кожен товар попереднього стану перевіряється на наявність у новій множині ідентифікаторів

$$T(n, m) = O(n + m) \quad (2.18)$$

Формула (2.18) показує загальну асимптотичну складність запропонованого методу за умови використання індексу за ідентифікатором. Якщо кількість товарів у новому фіді та попередньому стані приблизно однакова, тобто $n \approx m$, тоді складність можна подати як лінійну відносно кількості товарів.

$$T(n) = O(n) \quad (2.19)$$

Для порівняння, повне попарне зіставлення товарів без індексу потребувало б перевірки кожного товару нового фіду з кожним товаром попереднього стану.

$$T_{pairwise}(n, m) = O(n \cdot m) \quad (2.20)$$

Запропонований метод не усуває необхідності прочитати весь новий XML/YML-фід, оскільки фід надходить як повний знімок стану каталогу. Метод аналізує всі товари нового фіду, але виконує операції запису лише для товарів, стан яких змінився або які потребують додавання чи деактивації. Саме це зменшує надлишкові операції запису порівняно з повним перезаписом каталогу.

Практична реалізація запропонованого методу, вибір програмних засобів, побудова серверної частини, бази даних, API та клієнтського інтерфейсу розглядаються у Розділі 3. У цьому розділі визначено математичну та алгоритмічну основу, на якій базується подальша програмна реалізація.

Висновки до розділу 2

У результаті розроблення математичної моделі, алгоритму інкрементальної синхронізації, правил класифікації товарів і концептуальної моделі даних можна сформулювати такі висновки:

1. У розділі формалізовано задачу синхронізації версійного XML/YML-потоку як порівняння нової повної версії каталогу з попереднім локальним станом бази даних.

2. Визначено модель товарної позиції та контрольований стан товару, який складається з ціни, статусу наявності та кількості.

3. Обґрунтовано розмежування ідентифікаційних, контрольованих, описових і допоміжних структурних полів товарної позиції.

4. Розроблено математичну модель класифікації товарів на нові, оновлені, незмінені та відсутні у новій версії XML/YML-фіду.

5. Запропоновано формування контрольного хешу для компактного порівняння станів товарної позиції; MD5 розглядається лише як технічний механізм порівняння, а не як засіб криптографічного захисту.

6. Розроблено алгоритм інкрементальної синхронізації, який аналізує всі товари нового фіду, але виконує запис лише для нових, оновлених і деактивованих позицій.

7. Обґрунтовано доцільність використання структурованої реляційної концептуальної моделі даних для зберігання поточного стану товарів, циклів синхронізації та журналу змін без прив’язки до конкретної СУБД на етапі розроблення методу.

8. Запропоновано концептуальну модель даних для зберігання поточного стану каталогу, історії циклів синхронізації та журналу змін, що забезпечує простежуваність переходів між версіями XML/YML-потоку.

9. Сформовано алгоритмічний каркас методу до етапу програмної реалізації, у якому виділено логічні компоненти отримання фіду, парсингу, нормалізації, порівняння, класифікації, збереження результатів і формування статистики.

10. Проведено оцінку обчислювальної складності, яка показує, що за умови використання індексу за ідентифікатором загальна складність методу становить $O(n + m)$, а при $n \approx m$ має лінійний характер $O(n)$.

РОЗДІЛ 3
ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ МЕТОДУ ІНКРЕМЕНТАЛЬНОЇ СИНХРОНІЗАЦІЇ ДРОПШИПІНГ-КАТАЛОГУ

3.1. Призначення та функціональні можливості програмної системи

У Розділі 2 було розроблено метод інкрементальної синхронізації дропшипінг-каталогу, що ґрунтується на порівнянні нової версії XML/YML-фіду з попереднім станом локальної бази даних. Було визначено математичну модель, контрольовані поля товарної позиції, алгоритм класифікації змін, структури даних і концептуальну модель зберігання результатів синхронізації. У цьому розділі розглядається практична реалізація запропонованого методу у вигляді програмної системи та перевірка її роботи на товарному XML/YML-фіді.

Розроблена програмна система призначена для автоматизованого опрацювання товарного каталогу постачальника, який надходить у вигляді XML/YML-фіду. Система забезпечує отримання фіду, перетворення XML-структури у внутрішнє представлення товарних позицій, порівняння нового стану каталогу з попереднім локальним станом, фіксацію змін і відображення результатів синхронізації у web-інтерфейсі.

Основною метою програмної реалізації є підтвердження працездатності методу, розробленого у Розділі 2. На відміну від повного перезапису товарного каталогу, система аналізує всі позиції нової версії XML/YML-фіду, але операції запису виконує лише для тих товарів, стан яких потребує зміни: нових, оновлених або деактивованих. Незмінені товари не перезаписуються повторно, однак враховуються у статистиці циклу синхронізації.

Програмна система реалізує завантаження актуальної версії фіду, парсинг товарних позицій, нормалізацію контрольованих полів, формування контрольного хешу, порівняння з попереднім станом, класифікацію товарів, збереження актуального стану каталогу, ведення історії синхронізацій, фіксацію змін у журналі, пошук товарів, фільтрацію за брендом та відображення статистики у Dashboard.

Система має прикладне значення для дропшипінг-моделі, оскільки дозволяє підтримувати локальний каталог товарів в актуальному стані без повного

перезапису всіх позицій під час кожного оновлення. Це особливо важливо для каталогів із великою кількістю товарів, де між двома послідовними циклами синхронізації значна частина позицій може залишатися без змін.

За результатами контрольного запуску в актуальній версії програмної системи у локальному каталозі відображено 5655 активних товарних позицій, зафіксовано 23 цикли синхронізації та 7490 записів змін. Останній цикл синхронізації було виконано за 18635 мс, при цьому система визначила 59 нових товарів, 461 оновлений товар, 62 деактивовані товари та 5155 незмінених позицій.

Загальний вигляд сторінки каталогу товарів після виконання синхронізації наведено на рисунку 3.1.

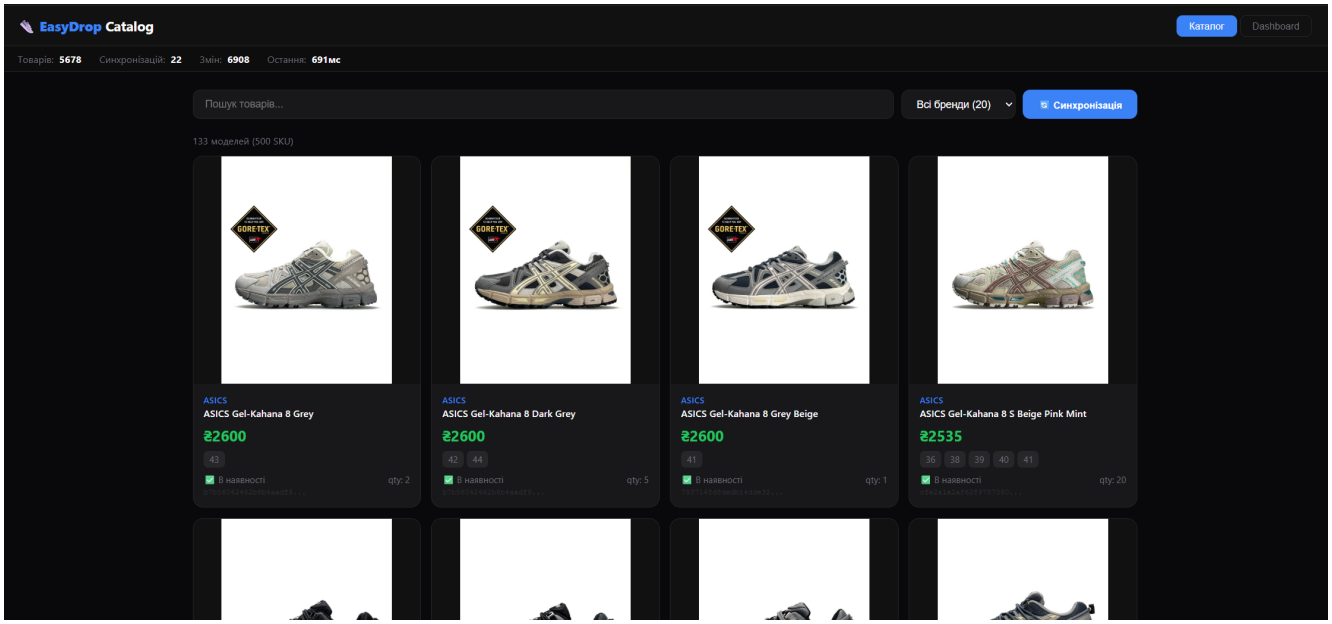

Рисунок 3.1 — Інтерфейс каталогу товарів після виконання синхронізації

Як видно з рисунка 3.1, клієнтський інтерфейс відображає список товарних позицій, їхні зображення, бренд, назву, ціну, статус наявності, кількість і скорочений ідентифікатор. У верхній частині сторінки показано загальну статистику: кількість товарів, кількість синхронізацій, кількість зафіксованих змін і тривалість останнього запуску.

3.2. Обґрунтування вибору програмних засобів реалізації

Вибір програмних засобів для реалізації системи здійснювався з урахуванням алгоритмічної логіки, розробленої у Розділі 2. Основними вимогами до програмної реалізації були: можливість отримання XML/YML-фіду із зовнішнього джерела,

перетворення XML-структури у внутрішнє представлення товарів, виконання інкрементального порівняння з попереднім станом, збереження результатів синхронізації та відображення статистики у web-інтерфейсі.

Програмна система має працювати з даними, що надходять у вигляді повного знімка товарного каталогу. Тому серверна частина повинна забезпечувати отримання фіду, його обробку, формування контрольних хешів, порівняння з локальною базою даних і запис результатів синхронізації. Для реалізації цієї логіки було обрано середовище Node.js, оскільки воно підходить для задач, пов’язаних з обробкою мережевих запитів, асинхронним отриманням даних і побудовою серверних застосунків [18; 19]..

Для організації взаємодії між серверною та клієнтською частинами використано Express. Його роль у системі полягає не лише у створенні HTTP-маршрутів, а й у забезпеченні доступу до основних функцій програмної системи: запуску синхронізації, отримання каталогу товарів, перегляду статистики, історії синхронізацій і списку брендів. Таким чином, Express використовується як проміжний рівень між алгоритмічним ядром синхронізації та web-інтерфейсом користувача [20].

Клієнтську частину реалізовано з використанням React, TypeScript і Vite. React обрано для побудови компонентного інтерфейсу, у якому окремими логічними частинами виступають каталог товарів, панель статистики, історія синхронізацій, фільтр брендів і елементи запуску синхронізації. TypeScript використано для підвищення надійності клієнтського коду, оскільки інтерфейс працює з різними структурами даних: товарними позиціями, статистикою, результатами циклів синхронізації та відповідями API. Vite застосовано як інструмент швидкого запуску й збирання клієнтської частини під час розроблення [21; 22; 23].

Для зберігання даних використано реляційну модель, яка була обґрунтована у Розділі 2. У програмній реалізації її реалізовано засобами SQLite через бібліотеку sql.js. Такий варіант є доцільним для дослідної реалізації, оскільки дозволяє зберігати структуровані таблиці товарів, історії синхронізацій і журналу змін без необхідності розгортання окремого серверу бази даних. При цьому логічна структура

зберігання відповідає концептуальній моделі: поточний стан товарів, цикли синхронізації та журнал змін зберігаються як пов’язані сутності [24; 25].

Для обробки XML/YML-фіду використано бібліотеку xml2js. Вона забезпечує перетворення XML-документа у структуру об’єктів, після чого серверна частина приводить отримані дані до внутрішньої моделі товарної позиції. Це дає змогу надалі працювати з товарами не як із сирим XML-документом, а як із масивом структурованих об’єктів, що містять ідентифікатор, назву, бренд, ціну, наявність, кількість, категорію, зображення та інші поля [26].

Для отримання XML/YML-фіду із зовнішнього джерела використано axios. Його застосування дозволяє серверній частині виконувати HTTP-запит до джерела даних і отримувати актуальну версію товарного каталогу перед запуском циклу синхронізації. Це відповідає логіці методу, за якою кожен цикл починається з отримання нової версії фіду [27; 28].

Формування контрольного хешу реалізовано з використанням стандартного модуля crypto. У межах цієї роботи хеш не використовується як засіб криптографічного захисту. Його призначення полягає у компактному порівнянні контрольованого стану товару, який складається з ціни, статусу наявності та кількості. Такий підхід дозволяє швидко визначити, чи змінилися ключові для синхронізації поля товарної позиції [30].

Для автоматичного запуску синхронізації може використовуватися node-cron. Його застосування дозволяє виконувати синхронізацію за розкладом, що є корисним для дропшипінг-каталогів, які регулярно оновлюються постачальником. Водночас система також підтримує ручний запуск синхронізації через web-інтерфейс [29].

Таблиця 3.1 — Відповідність програмних засобів задачам реалізації системи

Програмний засіб	**Задача, яку він вирішує у системі**
Node.js	Виконання серверної логіки отримання, обробки та синхронізації XML/YML-даних
Express	Надання API для запуску синхронізації, отримання товарів, статистики та історії

React	Побудова клієнтського інтерфейсу для перегляду каталогу й результатів синхронізації
TypeScript	Типізація даних клієнтської частини та зменшення кількості помилок під час роботи з API
Vite	Запуск і збирання клієнтської частини під час розроблення
SQLite / sql.js	Локальне зберігання товарів, циклів синхронізації та журналу змін
xml2js	Перетворення XML/YML-фіду у внутрішнє представлення товарних позицій
axios	Отримання актуальної версії XML/YML-фіду із зовнішнього джерела
crypto	Формування контрольного хешу товарної позиції
node-cron	Запуск синхронізації за розкладом

Отже, обрані програмні засоби не є самостійною метою роботи, а використовуються для практичної реалізації методу, розробленого у Розділі 2. Серверна частина забезпечує виконання алгоритму синхронізації, база даних зберігає поточний стан і результати змін, а клієнтська частина надає засоби перегляду каталогу, статистики та історії синхронізацій.

3.3. Загальна архітектура програмної системи

Програмна система інкрементальної синхронізації дропшипінг-каталогу побудована за web-орієнтованою архітектурою, у якій логіка обробки XML/YML-фіду відокремлена від клієнтського інтерфейсу. Такий підхід дозволяє розділити відповідальність між модулями системи: серверна частина виконує отримання, обробку та синхронізацію даних, база даних зберігає поточний стан каталогу й історію змін, а клієнтська частина забезпечує відображення результатів роботи алгоритму.

Зовнішнє джерело XML/YML-фіду надає актуальну версію товарного каталогу. У межах роботи такий фід розглядається як повний знімок поточного стану каталогу, що містить товарні позиції, їхні ідентифікатори, назви, ціни, статус наявності, кількість, зображення, бренд, категорію та інші допоміжні характеристики.

Серверна частина є центральним компонентом системи. Вона отримує XML/YML-фід, запускає алгоритм синхронізації, взаємодіє з базою даних і надає результати клієнтській частині через API. Саме на серверному рівні реалізовано основну логіку методу.

Локальна база даних зберігає поточний стан каталогу, історію циклів синхронізації та журнал змін. Клієнтський web-інтерфейс забезпечує перегляд каталогу, пошук, фільтрацію за брендом, запуск синхронізації, аналіз історії запусків і перегляд розподілу товарів за брендами.

Загальну архітектуру програмної системи наведено на рисунку 3.2.

Рисунок 3.2 — Загальна архітектура програмної системи інкрементальної синхронізації

Як видно з рисунка 3.2, система реалізує послідовний цикл обробки даних: отримання XML/YML-фіду, перетворення його у внутрішню структуру, порівняння з попереднім станом, запис змін до бази даних і відображення результатів у клієнтському інтерфейсі.

3.4. Реалізація бази даних програмної системи

Для зберігання результатів роботи системи використовується локальна реляційна модель даних, яка відображає концептуальну модель, розроблену у Розділі 2. Фізична структура бази даних містить три основні таблиці: products, sync_history і changes_log.

Таблиця products використовується для зберігання поточного стану товарного каталогу. Кожен запис у цій таблиці відповідає окремій товарній позиції, отриманій із XML/YML-фіду. Основним полем для зіставлення товарів між версіями фіду є external_id.

У таблиці products також зберігаються контрольовані поля товару: ціна, статус наявності та кількість. На основі цих полів формується контрольний хеш, який надалі використовується для визначення змін. Якщо товар відсутній у новому фіді, він не видаляється фізично, а позначається як неактивний.

Таблиця sync_history використовується для зберігання інформації про кожен цикл синхронізації: час початку й завершення, кількість оброблених товарів, кількість нових, оновлених, деактивованих і незмінених позицій, тривалість виконання та статус.

Таблиця changes_log призначена для фіксації окремих змін, виявлених під час синхронізації. Вона зберігає тип зміни, ідентифікатор товару, ідентифікатор циклу синхронізації, попередні та нові значення контрольованих полів.

Таблиця 3.2 — Основні таблиці бази даних програмної системи

Таблиця	Призначення	Ключові поля
products	Зберігання поточного стану товарного каталогу	external_id, name, price, available, quantity, vendor / brand, hash, is_active
sync_history	Зберігання статистики циклів синхронізації	id, started_at, finished_at, status, total_products, new_products, updated_products, deleted_products, unchanged_products, duration_ms
changes_log	Фіксація окремих змін товарних позицій	sync_id, product_external_id, change_type, old_price, new_price, old_available, new_available, old_quantity, new_quantity

Поле vendor / brand використовується для збереження бренду або назви постачальника товару та застосовується під час фільтрації каталогу в клієнтському інтерфейсі.

У базі даних також створено індекси за зовнішнім ідентифікатором товару, контрольним хешем, статусом активності та брендом. Це прискорює пошук товару під час синхронізації, вибірку активних позицій і фільтрацію каталогу в клієнтському інтерфейсі.

3.5. Реалізація серверної частини та модулів синхронізації

Серверна частина реалізує основну логіку інкрементальної синхронізації. Вона складається з кількох модулів, кожен із яких відповідає за окремий етап

обробки: отримання фіду, парсинг XML, формування хешу, порівняння станів, запис до бази даних і надання API.

Файл app.js виконує роль точки входу серверної частини. У ньому налаштовується Express-додаток, підключається CORS, статична директорія клієнтської частини, API-маршрути та, за потреби, автоматичний запуск синхронізації через node-cron.

Модуль parser.js перетворює XML/YML-документ у масив товарних позицій. Під час парсингу з фіду виділяються ідентифікатор, назва, ціна, статус наявності, кількість, категорія, бренд, зображення, group_id, barcode і параметри товару. Якщо бренд явно не вказаний, система може визначити його з першого слова назви товару.

Модуль hasher.js реалізує формування контрольного хешу товарної позиції. Значення ціни нормалізується до двох знаків після коми, статус наявності приводиться до true або false, після чого контрольовані поля об’єднуються через роздільник і передаються до MD5-функції модуля crypto.

Модуль comparator.js реалізує порівняння нової версії фіду з попереднім станом бази даних. Попередній стан індексується за externalId за допомогою Map, а ідентифікатори нового фіду зберігаються у Set. Така структура дозволяє класифікувати товари без попарного порівняння кожного запису з кожним.

Модуль sync-service.js координує повний цикл синхронізації: отримує XML-фід через axios або з локального файлу, викликає парсер, формує хеші, завантажує попередній стан, запускає порівняння, записує нові, оновлені та деактивовані товари, фіксує журнал змін і завершує запис статистики циклу.

Центральною частиною серверної логіки є саме модуль sync-service.js, оскільки він поєднує окремі модулі в єдиний цикл синхронізації. parser.js перетворює XML/YML-фід у структуровані дані, hasher.js формує контрольне значення, comparator.js визначає тип зміни, а database.js фіксує результат у базі даних. Таким чином, серверна частина безпосередньо реалізує алгоритмічний каркас, розроблений у Розділі 2.

Таблиця 3.3 — Основні модулі серверної частини

Модуль	Роль у системі
app.js	Налаштування Express API, запуск серверу, підключення cron-синхронізації
database.js	Створення таблиць, робота з products, sync_history і changes_log
parser.js	Перетворення XML/YML-фіду у внутрішнє представлення товарних позицій
hasher.js	Нормалізація контрольованих полів і формування MD5-хешу
comparator.js	Виявлення нових, оновлених, незмінених і деактивованих товарів
sync-service.js	Оркестрація повного циклу синхронізації

Фрагмент реалізації формування контрольного хешу наведено нижче.

```
static computeHash(price, available, quantity = null) {
  const p = parseFloat(price);
  const pr = isNaN(p) ? '0.00' : p.toFixed(2);
  const av = available ? 'true' : 'false';
  const input = quantity != null ? `${pr}|${av}|${quantity}` : `${pr}|${av}`;
  return crypto.createHash('md5').update(input, 'utf8').digest('hex');
}
```

У цьому фрагменті видно, що перед хешуванням значення приводяться до стабільного формату, а між ними використовується роздільник. Це відповідає моделі формування контрольного рядка, наведений у Розділі 2.

Фрагмент логіки порівняння товарів наведено нижче.

```
const dbIndex = new Map();
for (const p of dbProducts) dbIndex.set(p.externalId, p);
const xmlIds = new Set();

for (const xp of xmlProducts) {
  xmlIds.add(xp.externalId);
  const newHash = Hasher.hashProduct(xp);
  const dbp = dbIndex.get(xp.externalId);
  if (!dbp) newProducts.push({ ...xp, hash: newHash });
  else if (newHash !== dbp.hash) updatedProducts.push({ ...xp, hash: newHash });
  else unchangedCount++;
}
```

Наведений код демонструє практичне використання індексу попереднього стану за externalId. Завдяки цьому кожен товар нового фіду перевіряється один раз, а пошук відповідного попереднього запису виконується через Map.

3.6. Реалізація клієнтської частини програмної системи

Клієнтська частина програмної системи реалізована як web-інтерфейс, що забезпечує перегляд каталогу, пошук товару, фільтрацію товарів, запуск

синхронізації та аналіз статистики. Інтерфейс побудований у вигляді двох основних режимів: Catalog і Dashboard.

Режим Catalog призначений для перегляду товарних позицій. На сторінці відображаються картки товарів із зображенням, брендом, назвою, ціною, статусом наявності, розмірами, кількістю та скороченим контрольним хешем. Для зручності користувача реалізовано пошук за назвою та фільтрацію за брендом.

Для зручності роботи з великим каталогом у режимі Catalog реалізовано пошук товарів за назвою. Користувач може ввести повну назву товару або її частину, після чого інтерфейс відображає лише ті товарні позиції, які відповідають пошуковому запиту. Це дає змогу швидко знаходити потрібний товар без ручного перегляду всього списку та спрощує перевірку результатів синхронізації після оновлення XML/YML-фіду.

Використання пошуку у каталозі показано на рисунку 3.3.

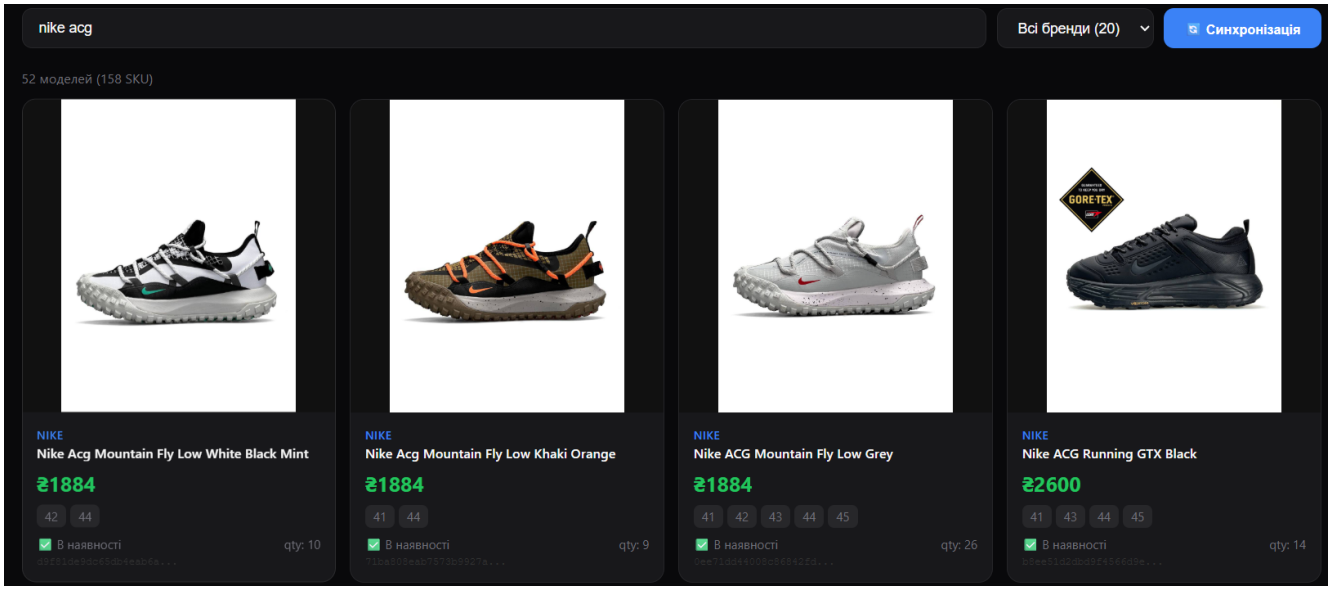

Рисунок 3.3 —Пошук товару за назвою у клієнтському інтерфейсі

Також у клієнтському інтерфейсі реалізовано фільтрацію товарів за брендом. Пошук за назвою та фільтрація за брендом можуть використовуватися як допоміжні інструменти для аналізу локального каталогу, оскільки дозволяють звузити перелік відображених товарних позицій за потрібними користувачу ознаками.

Система групує товари за моделями, використовуючи group_id або зовнішній ідентифікатор. Завдяки цьому кілька SKU, які належать до однієї моделі, можуть відображатися як одна картка з набором розмірів і сумарною кількістю.

Використання фільтрації за брендом у каталозі показано на рисунку 3.4.

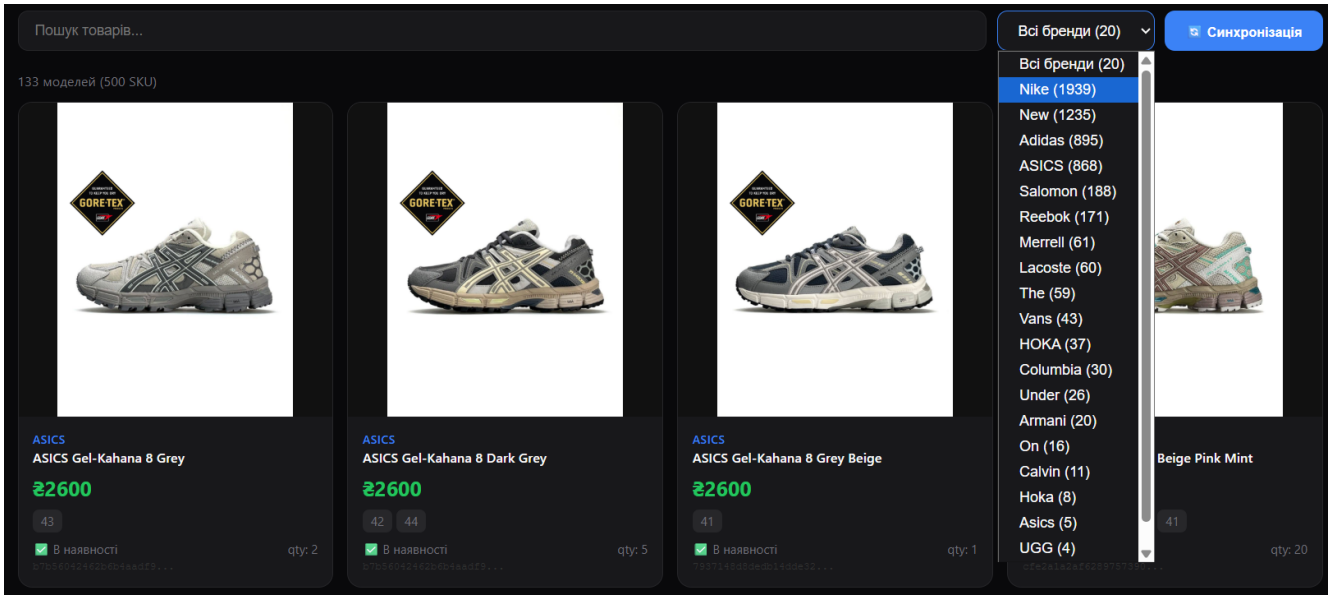

Рисунок 3.4 — Фільтрація товарів за брендом у клієнтському інтерфейсі

Як видно з рисунка 3.4, список брендів формується разом із кількістю товарних позицій для кожного бренду. Це підтверджує, що клієнтська частина використовує не статичні дані, а результати запитів до серверного API.

Режим Dashboard призначений для аналізу результатів синхронізації. На ньому відображаються ключові показники системи: кількість активних товарів, кількість виконаних циклів синхронізації, кількість записів змін і кількість брендів. Крім того, Dashboard містить таблицю історії синхронізацій і розподіл товарів за брендами.

Історія синхронізацій відображає номер циклу, статус виконання, тривалість, кількість нових, оновлених, деактивованих і пропущених товарів. Це дозволяє перевірити, як система реагувала на різні версії XML/YML-фіду.

Загальний вигляд Dashboard із історією синхронізацій наведено на рисунку 3.5.

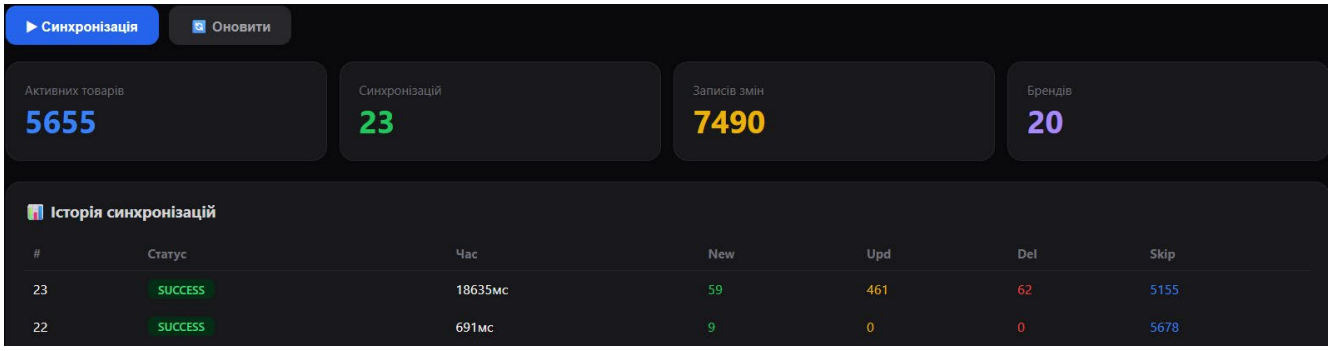

Рисунок 3.5 — Dashboard та історія циклів синхронізації

Розподіл товарів за брендами в інтерфейсі наведено на рисунку 3.6.

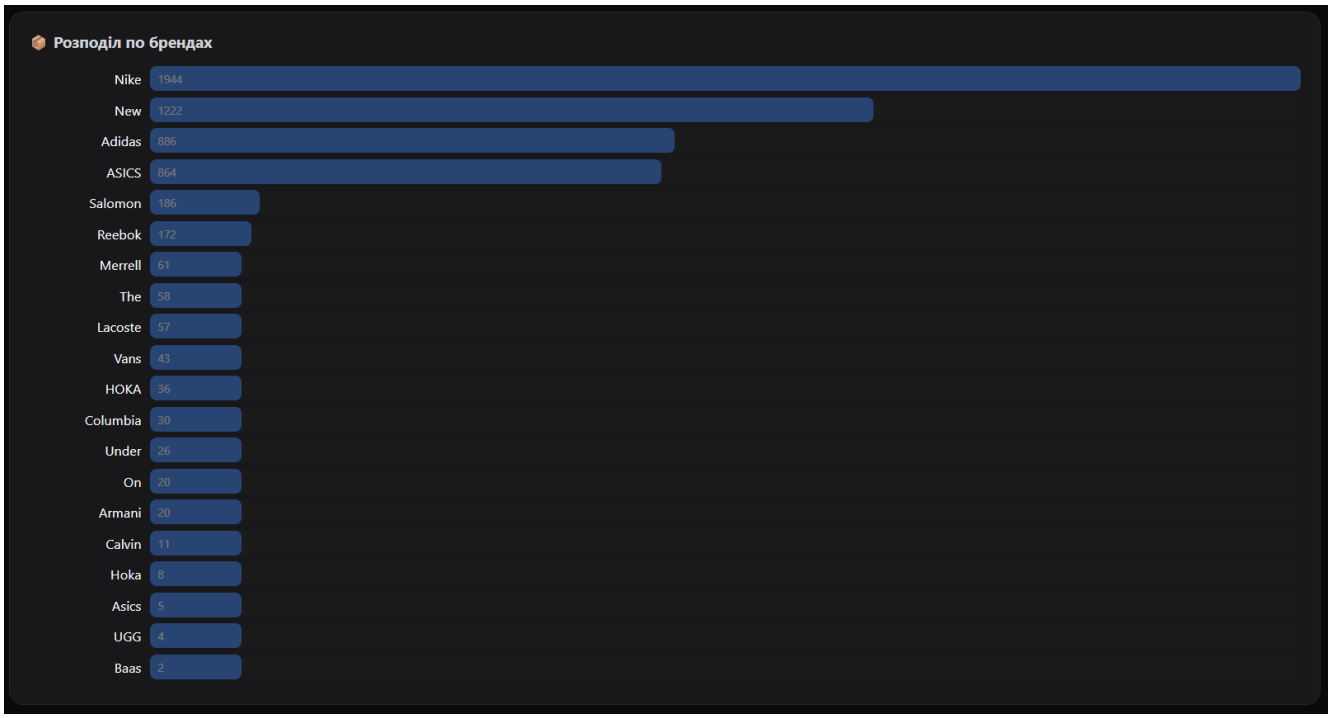

Рисунок 3.6. — Розподіл товарів за брендами

На рисунку 3.6 видно, що найбільшу кількість активних товарів мають бренди Nike, New, Adidas та ASICS. Таке подання дозволяє швидко оцінити структуру локального каталогу після синхронізації.

3.7. API програмної системи

Взаємодія між клієнтською та серверною частинами здійснюється через HTTP API. API надає клієнтській частині доступ до товарів, статистики, списку брендів, історії синхронізацій, журналу змін і запуску нового циклу синхронізації.

Маршрути API реалізовано у серверному файлі app.js. Усі основні операції виконуються через префікс /api. Частина маршрутів працює у режимі читання даних, а запуск синхронізації виконується через POST-запит.

Таблиця 3.4 — Основні API-маршрути програмної системи

Метод	Маршрут	Призначення
GET	/api/stats	Отримання загальної статистики системи
GET	/api/products	Отримання списку активних товарів з урахуванням пошуку та фільтра бренду
GET	/api/vendors	Отримання списку брендів і кількості товарів за кожним брендом
GET	/api/history	Отримання історії циклів синхронізації
GET	/api/changes/:id	Отримання журналу змін для конкретного циклу синхронізації
GET	/api/sync/status	Перевірка стану виконання синхронізації
POST	/api/sync	Запуск нового циклу синхронізації

Використання окремих API-маршрутів забезпечує розділення відповідальності між серверною логікою та клієнтським інтерфейсом. Клієнтська частина не виконує синхронізацію самостійно, а лише надсилає запити до серверної частини та відображає отримані результати.

3.8. Тестування роботи системи на XML/YML-фіді

Тестування програмної системи виконувалося для перевірки коректності реалізації методу інкрементальної синхронізації. Основна мета тестування полягала в тому, щоб переконатися, що система правильно класифікує товари як нові, оновлені, незмінені або деактивовані та коректно фіксує результати у базі даних.

Перевірка проводилася на товарному XML/YML-фіді, що містить декілька тисяч товарних позицій. Під час першого запуску локальна база даних ще не містила попереднього стану, тому всі товари були класифіковані як нові. Під час наступних запусків система вже порівнювала нову версію фіду з попереднім локальним станом.

Окремо перевірялися сценарії повторного запуску без істотних змін, появи нових товарів, оновлення контрольованих полів і зникнення товарів із нового фіду. Результати таких запусків відображалися в таблиці історії синхронізацій і в журналі змін.

Таблиця 3.5 — Сценарії тестування програмної системи

Сценарій	Очікуваний результат	Підтвердження в системі
Перший запуск синхронізації	Усі товари визначаються як нові	У першому циклі зафіксовано 5752 нові товари та 0 пропущених
Повторний запуск без змін	Товари визначаються як незмінені	У кількох циклах зафіксовано 0 нових, 0 оновлених, 0 деактивованих і понад 5700 пропущених позицій
Поява нових товарів	Нові товари додаються до локального каталогу	У циклі № 23 зафіксовано 59 нових товарів
Зміна ціни, наявності або кількості	Товар класифікується як оновлений	В історії є цикли з оновленими позиціями, наприклад № 14 з 186 оновленнями
Відсутність товарів у новому фіді	Товари позначаються як деактивовані	наприклад, у циклі № 14 зафіксовано 47 деактивованих товарів

Результати тестування підтверджують, що система реалізує саме інкрементальний підхід: усі товари фіду аналізуються, але повторний запис незмінених позицій не виконується. Це видно з історії синхронізацій, де для стабільних запусків кількість пропущених позицій значно перевищує кількість записів змін.

3.9. Аналіз результатів синхронізації та продуктивності

За результатами актуального тестового запуску програмної системи у Dashboard відображено 5655 активних товарів, 23 виконані цикли синхронізації, 7490 записів змін і 20 брендів. Останній цикл синхронізації мав статус SUCCESS і був виконаний за 18635 мс.

Під час циклу № 23 система визначила 59 нових товарів, 461 оновлений товар, 62 деактивовані товари та 5155 незмінених позицій. Це свідчить про те, що система коректно відокремлює різні типи змін у каталозі: нові товари додаються до локальної бази даних, змінені позиції оновлюються, відсутні у новому фіді товари деактивуються, а незмінені позиції пропускаються без повторного запису.

Особливе значення має показник skip, який у циклі № 23 становить 5155 товарних позицій. Це означає, що для більшої частини каталогу система не виконувала зайвих операцій оновлення, оскільки контрольований стан цих товарів не змінився. Таким чином, інкрементальний підхід дозволяє уникати повного перезапису каталогу та виконувати операції запису лише для тих позицій, які дійсно потребують змін.

Порівняно з попередніми циклами синхронізації, останній запуск демонструє роботу системи не лише у випадку появи нових товарів, а й у випадку значної кількості оновлень і деактивацій. Зокрема, у циклі № 23 було зафіксовано 461 оновлену позицію та 62 деактивовані товари. Це підтверджує, що реалізований алгоритм здатний обробляти не тільки додавання нових записів, а й зміну контрольованих полів та зникнення товарів із нового XML/YML-фіду.

Загальна кількість записів змін становить 7490, що підтверджує накопичення історії синхронізацій і можливість подальшого аналізу змін каталогу. Наявність історії запусків дозволяє відстежувати номер циклу, статус виконання, тривалість синхронізації, кількість нових, оновлених, деактивованих і пропущених товарів.

Розподіл товарів за брендами показує структуру локального каталогу після синхронізації. Ці дані використовуються не для безпосередньої оцінки ефективності методу синхронізації, а для демонстрації можливостей аналітичного відображення результатів у клієнтському інтерфейсі. Завдяки Dashboard користувач може оцінити загальний стан каталогу, кількість брендів, історію синхронізацій і співвідношення між різними типами змін.

Таблиця 3.6 — Результати контрольного запуску системи

Показник	Значення
Активних товарів у локальному каталозі	5655
Кількість циклів синхронізації	23
Кількість записів змін	7490
Кількість брендів	20
Тривалість останнього запуску	18635 мс
Нових товарів у циклі № 23	59
Оновлених товарів у циклі № 23	461
Деактивованих товарів у циклі № 23	62
Незмінених товарів у циклі № 23	5155

Отримані результати підтверджують працездатність програмної реалізації методу інкрементальної синхронізації. Система здатна обробляти XML/YML-фід із тисячами товарних позицій, зберігати поточний стан каталогу, вести історію синхронізацій, формувати журнал змін і відображати аналітичні показники у клієнтському інтерфейсі. Результати циклу № 23 показують, що система не перезаписує

весь каталог повністю, а виконує оновлення лише для нових, змінених і відсутніх товарів, тоді як незмінені позиції пропускаються. Фрагмент журналу змін, отриманий через API програмної системи за результатами циклу синхронізації № 23, наведено в додатку Б.

Висновки до розділу 3

Проведена програмна реалізація та тестування запропонованого методу дозволяють узагальнити отримані результати та сформулювати такі висновки:

1. У розділі реалізовано програмну систему для інкрементальної синхронізації дропшипінг-каталогу на основі XML/YML-фіду.

2. Обґрунтовано вибір програмних засобів реалізації: Node.js, Express, React, TypeScript, Vite, SQLite/sql.js, xml2js, axios, crypto та node-cron.

3. Розроблено серверну частину, яка виконує отримання фіду, парсинг XML/YML-даних, формування контрольного хешу, порівняння станів і запис результатів синхронізації.

4. Реалізовано локальну базу даних із таблицями products, sync_history і changes_log, що забезпечують зберігання поточного стану каталогу, історії запусків і журналу змін.

5. Реалізовано клієнтський web-інтерфейс для перегляду каталогу, пошуку, фільтрації за брендами, запуску синхронізації та аналізу результатів у Dashboard.

6. Проведено тестування системи на XML/YML-фіді, у межах якого перевірено перший запуск, повторні запуски без змін, появу нових товарів, оновлення контрольованих полів і деактивацію відсутніх позицій.

7. За результатами контрольного запуску система обробила каталог із 5655 активними товарними позиціями, зафіксувала 23 цикли синхронізації та 7490 записів змін.

8. Підтверджено, що реалізована система аналізує всі товари нового фіду, але виконує операції запису лише для нових, оновлених і деактивованих позицій, що відповідає методу, розробленому у Розділі 2.

ЗАГАЛЬНІ ВИСНОВКИ

У кваліфікаційній роботі вирішено задачу підвищення ефективності оновлення веб-каталогу товарів у дропшипінг-моделі шляхом розроблення та програмної реалізації методу інкрементальної синхронізації на основі версійних XML/YML-потоків. Отримані результати дозволяють сформулювати такі загальні висновки:

1. У результаті аналізу предметної області встановлено, що дропшипінг-модель потребує регулярного оновлення товарних даних, оскільки продавець не контролює склад постачальника безпосередньо. Найбільше прикладне значення для роботи веб-каталогу мають ціна, наявність і кількість товару, оскільки ці параметри визначають можливість приймання та виконання замовлень.

2. Досліджено структуру XML/YML-каталогів як поширеного способу передавання товарних даних. Визначено, що зовнішній фід у межах роботи розглядається як повний знімок поточного стану каталогу, який не містить універсального дельта-інтерфейсу або надійного службового сигналу про зміни.

3. Проаналізовано підходи до оновлення товарного каталогу: ручне оновлення, повний перезапис, оновлення за modified_date, API-інтеграцію та інкрементальне оновлення. Обґрунтовано, що для задачі роботи доцільним є інкрементальний підхід, за якого система самостійно зіставляє нову версію фіду з попереднім локальним станом і виконує запис лише для позицій, які справді потребують зміни.

4. Розроблено математичну модель інкрементальної синхронізації, у якій нова версія XML/YML-фіду, попередній стан локальної бази даних і результат циклу синхронізації описуються через відповідні множини. У межах моделі визначено нові, оновлені, незмінені та відсутні у новому фіді товари.

5. Запропоновано використання контрольного хешу для компактного порівняння стану товарної позиції. До базового набору контрольованих полів віднесено ціну, наявність і кількість. Хешування цих значень після нормалізації дозволяє визначати зміну операційного стану товару без повного порівняння всіх описових полів і без залежності від поля modified_date.

6. Розроблено алгоритм інкрементальної синхронізації XML/YML-каталогу, що передбачає отримання нової версії фіду, парсинг товарних позицій, завантаження попереднього стану, побудову індексу за ідентифікатором, формування контрольного хешу, класифікацію товарів і запис результатів до бази даних. Оцінка складності показала, що за умови використання індексу за ідентифікатором загальна складність методу становить $O(n + m)$, а при $n \approx m$ має лінійний характер.

7. Спроєктовано концептуальну модель даних, яка охоплює поточний стан товарів, історію циклів синхронізації та журнал змін. Така структура забезпечує не лише оновлення локального каталогу, а й простежуваність результатів кожного запуску, збереження статистики та можливість аналізу змін між версіями XML/YML-фіду.

8. Виконано програмну реалізацію запропонованого методу у вигляді веб-системи. Серверна частина забезпечує отримання XML/YML-фіду, парсинг даних, формування контрольних хешів, порівняння станів і збереження результатів. Клієнтський інтерфейс дає змогу переглядати каталог, виконувати пошук і фільтрацію, запускати синхронізацію та аналізувати результати через Dashboard і історію синхронізацій.

9. Проведено тестування системи на XML-каталозі EasyDrop. За результатами контрольного запуску в локальному каталозі відображено 5655 активних товарних позицій, зафіксовано 23 цикли синхронізації та 7490 записів змін. Останній цикл синхронізації тривав 18635 мс і виявив 59 нових товарів**,** 461 оновлений товар**,** 62 деактивовані товари та 5155 незмінених позицій. Це підтверджує працездатність реалізації та здатність системи обробляти всі основні типи змін каталогу.

10. Мету роботи досягнуто: розроблено та реалізовано метод інкрементальної синхронізації дропшипінг-каталогу, який дозволяє визначати зміни між версіями XML/YML-фіду без повного перезапису всіх товарів і без залежності від службового поля modified_date. Запропонований підхід може бути використаний як основа для систем актуалізації товарних каталогів у дропшипінг-проєктах і адаптований до інших XML/YML-фідів за наявності стабільного ідентифікатора товару та визначеного набору контрольованих полів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Menaouer B., Khalissa S., Belayachi M. E. A., Amine B. The Role of Drop Shipping in E-Commerce. International Journal of E-Business Research. 2021. Vol. 17, No. 4. P. 1–19. DOI: 10.4018/IJEBR.2021100104.

2. Liu C.-T., Guo Y. M., Hsu J.-L. Creating and Validating an Information Quality Scale for E-Commerce Platforms. Journal of Electronic Commerce in Organizations. 2023. Vol. 21, No. 1. P. 1–28. DOI: 10.4018/JECO.327350.

3. Extensible Markup Language (XML) 1.0 (Fifth Edition) : W3C Recommendation. W3C. 26 November 2008. URL: https://www.w3.org/TR/xml/ (дата звернення: 02.04.2026).

4. Імпорт через YML — формат файлу : офіційна довідка для продавців. Prom.ua. URL: https://support.prom.ua/hc/uk/articles/360004963538 (дата звернення: 02.04.2026).

5. Підготовка XML прайс-листа з товарами : офіційна довідка для продавців. Rozetka Seller Help. URL: https://sellerhelp.rozetka.com.ua/p274-create-price-list.html (дата звернення: 03.04.2026).

6. Імпорт товарів із файлу : центр допомоги Хорошоп. Хорошоп. URL: https://help.horoshop.ua/uk/articles/1684865 (дата звернення: 03.04.2026).

7. EasyDrop : офіційний сайт дропшипінг-платформи у форматі CRM-системи. URL: https://easydrop.one/ (дата звернення: 03.04.2026).

8. MyDrop : система обліку для постачальників і дропшиперів. URL: https://mydrop.com.ua/ (дата звернення: 03.04.2026).

9. Dropshipping.ua : офіційний сайт дропшипінг-платформи. URL: https://www.dropshipping.ua/ (дата звернення: 03.04.2026).

10. WP All Import — Product Import for WooCommerce : сторінка плагіна. WordPress.org. URL: https://uk.wordpress.org/plugins/woocommerce-xml-csv-product-import/ (дата звернення: 05.04.2026).

11. eCommerce Dropshipping Automation Software. Spark Shipping. URL: https://www.sparkshipping.com/ (дата звернення: 05.04.2026).

12. Dropshipping Automation Platform. AutoDS. URL: https://www.autods.com/ (дата звернення: 05.04.2026).
13. Cobéna G., Abiteboul S., Marian A. Detecting Changes in XML Documents. Proceedings of the 18th International Conference on Data Engineering (ICDE). San Jose, CA, USA, 2002. P. 41–52. DOI: 10.1109/ICDE.2002.994696.
14. Wang Y., DeWitt D. J., Cai J.-Y. X-Diff: An Effective Change Detection Algorithm for XML Documents. Proceedings of the 19th International Conference on Data Engineering (ICDE). Bangalore, India, 2003. P. 519–530. DOI: 10.1109/ICDE.2003.1260818.
15. Cobéna G., Abdessalem T., Hinnach Y. A Comparative Study for XML Change Detection. INRIA / ENST Research Report. URL: https://abiteboul.com/gemoReports/GemoReport-221.pdf (дата звернення: 10.05.2026).
16. Tridgell A., Mackerras P. The rsync algorithm : Technical Report TR-CS-96-05. Australian National University. 1996. URL: https://www.andrew.cmu.edu/course/15749/READINGS/required/cas/tridgell96.pdf (дата звернення: 10.05.2026).
17. Rivest R. The MD5 Message-Digest Algorithm : RFC 1321. 1992. URL: https://www.rfc-editor.org/rfc/rfc1321 (дата звернення: 13.05.2026).
18. About Node.js. Node.js. URL: https://nodejs.org/en/about (дата звернення: 18.05.2026).
19. Node.js API documentation. Node.js. URL: https://nodejs.org/docs/latest/api/ (дата звернення: 18.05.2026).
20. Express — Node.js web application framework. Express. URL: https://expressjs.com/ (дата звернення: 18.05.2026).
21. Quick Start. React. URL: https://react.dev/learn (дата звернення: 18.05.2026).
22. The TypeScript Handbook. TypeScript. URL: https://www.typescriptlang.org/docs/handbook/intro.html (дата звернення: 18.05.2026).
23. Vite — Next Generation Frontend Tooling. Vite. URL: https://vite.dev/ (дата звернення: 18.05.2026).
24. SQLite Documentation. SQLite. URL: https://sqlite.org/docs.html (дата звернення: 20.05.2026).

25. SQLite compiled to JavaScript : project repository. sql.js. URL: https://github.com/sql-js/sql.js/ (дата звернення: 20.05.2026).
26. Simple XML to JavaScript object converter : npm package documentation. xml2js. URL: https://www.npmjs.com/package/xml2js (дата звернення: 20.05.2026).
27. Axios documentation. Axios. URL: https://axios-http.com/docs/intro (дата звернення: 21.05.2026).
28. HTTP. MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Web/HTTP (дата звернення: 21.05.2026).
29. A cron-like job scheduler for Node.js : npm package documentation. node-cron. URL: https://www.npmjs.com/package/node-cron (дата звернення: 22.05.2026).
30. Crypto API. Node.js. URL: https://nodejs.org/api/crypto.html (дата звернення: 22.05.2026).

ДОДАТКИ

ДОДАТОК А

Фрагмент XML/YML-фіду товарного каталогу

У додатку А наведено скорочений фрагмент XML/YML-фіду товарного каталогу, який демонструє структуру вхідних даних для синхронізації. Фрагмент містить приклади ідентифікаційних, контрольованих та описових полів товарної позиції.

Приклад фрагмента XML/YML-фіду:

```
<shop>
  <categories>
    <category id="174059649784">Asics</category>
    <category id="174059649794">New Balance</category>
    <category id="174059649795">Nike</category>
    <category id="174059649796">Adidas</category>
    <category id="174059649803">Merrell</category>
  </categories>
  <items>
    <item id="894732127649" selling_type="r" group_id="127649" available="true">
      <categoryId>1740596416038</categoryId>
      <name>ASICS Gel-Kahana 8 Dark Grey</name>
      <priceuah>2600</priceuah>
      <description><p>Матеріал : шкіра , термо</p><p>Виробник : В’єт-
    нам</p></description>
      <barcode>A3247-894732</barcode>
      <image>https://easydrop.one/media/uploads/images/base/49ed4555-934a-4f6f-870a-
    e36714c63e04.png</image>
      <param name="Розмір" unit="">42</param>
      <available>true</available>
      <quantity_in_stock>2</quantity_in_stock>
    </item>
    <item id="894734127649" selling_type="r" group_id="127649" available="true">
      <categoryId>1740596416038</categoryId>
      <name>ASICS Gel-Kahana 8 Dark Grey</name>
      <priceuah>2600</priceuah>
      <description><p>Матеріал : шкіра , термо</p><p>Виробник : В’єт-
    нам</p></description>
      <barcode>A3247-894734</barcode>
      <image>https://easydrop.one/media/uploads/images/base/49ed4555-934a-4f6f-870a-
    e36714c63e04.png</image>
      <param name="Розмір" unit="">44</param>
      <available>true</available>
      <quantity_in_stock>3</quantity_in_stock>
     </item>
    </items>
</shop>
```

ДОДАТОК Б

Фрагмент журналу змін за результатами синхронізації

У додатку Б наведено фрагмент реальних записів журналу змін, отриманих через API програмної системи за маршрутом /api/changes/23. Записи сформовано за результатами циклу синхронізації № 23. Журнал змін містить лише ті товарні позиції, для яких було зафіксовано фактичну зміну стану: додавання нового товару, оновлення контрольованих полів або деактивацію товару.

Таблиця Б.1 — Фрагмент журналу змін за результатами циклу синхронізації № 23

cycle_id	product_id	change_type	old_price	new_price	old_avail.	new_avail.	old_qty	new_qty	changed_at
23	1623083232015	NEW	-	2600	-	0	-	1	2026-06-03 21:09:45
23	1325990193833	NEW	-	2600	-	0	-	2	2026-06-03 21:09:45
23	1532396220521	UPDATED	2730	2730	0	0	4	5	2026-06-03 21:09:45
23	1532373220518	UPDATED	2730	2730	0	0	5	4	2026-06-03 21:09:45
23	1455702210570	UPDATED	2652	2470	0	0	2	2	2026-06-03 21:09:45
23	1455703210570	UPDATED	2652	2470	0	0	3	3	2026-06-03 21:09:45
23	1520003218749	DELETED	2587	-	0	-	1	-	2026-06-03 21:09:45
23	990133143600	DELETED	2535	-	0	-	1	-	2026-06-03 21:09:45

Як видно з таблиці Б.1, записи з типом NEW не мають попередніх значень, оскільки відповідні товари були відсутні у попередньому стані локальної бази даних. Записи з типом UPDATED містять старі та нові значення контрольованих полів, що дозволяє визначити, яке саме поле змінилося. Наприклад, у частини товарів змінилася кількість, а в частини — ціна. Записи з типом DELETED мають попередні значення, але не мають нових значень, оскільки ці товари були відсутні у новій версії XML/YML-фіду.

За результатами циклу синхронізації № 23 у журналі змін було сформовано 582 записи: 59 нових товарів, 461 оновлений товар і 62 деактивовані товари. Незмінені товарні позиції не потрапляють до журналу змін, оскільки для них не виконується повторний запис до бази даних. Це підтверджує практичну логіку інкрементальної синхронізації: система фіксує лише фактичні зміни, а незмінені товари пропускає.