ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
Кафедра інженерії програмного забезпечення

Пояснювальна записка

до бакалаврської роботи
на ступінь вищої освіти бакалавр

на тему: «**АВТОМАТИЗАЦІЯ ПРОЦЕСУ ОНОВЛЕННЯ ДАНИХ В ІНФОРМАЦІЙНИХ СИСТЕМАХ ЗА ДОПОМОГОЮ ІНТЕГРАЦІЙНОЇ ШИНИ IBM MESSAGE BROKER**»

Виконала: студентка 4 курсу, групи __ПД-41___
спеціальності
121____Інженерія програмного забезпечення_____
(шифр і назва спеціальності)

_____Андрющенкова С.О._____
(прізвище та ініціали)

Керівник___Жебка В.В._____
(прізвище та ініціали)

Рецензент_____
(прізвище та ініціали)

Нормоконтроль _____
(прізвище та ініціали)

Київ – 2021

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра - Інженерії програмного забезпечення
Ступінь вищої освіти - «Бакалавр»
Спеціальність - 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри
інженерії програмного забезпечення
_____О. В. Негоденко
“____” _____ 2021 року

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

Андрющенкова Стелла Олександрівна
(прізвище, ім’я, по батькові)

1.Тема роботи: «АВТОМАТИЗАЦІЯ ПРОЦЕСУ ОНОВЛЕННЯ ДАНИХ В ІНФОРМАЦІЙНИХ СИСТЕМАХ ЗА ДОПОМОГОЮ ІНТЕГРАЦІЙНОЇ ШИНИ IBM MESSAGE BROKER»
Керівник роботи Жебка Вікторія Вікторівна к.т.н., доцент ,
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від “12”березня 2021 року №65.
2. Строк подання студентом роботи __1.06.2021_____
3. Вихідні дані до роботи: Функціональні можливості інтеграційної шини IBM Message Broker, мова програмування ESQL

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

1. Визначення поняття ціни та процесу ціноутворення.

2. Дослідження функціональних можливостей IBM Integration Bus.

3. Розробка бізнес-процесу та модуля по оновленню даних в інформаційних системах.

5. Перелік графічного матеріалу

1.

2.

3.

4.

6. Дата видачі завдання _____12.03.2021_____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Підбір науково-технічної літератури	12.03-19.03.2021	
2	Визначення та функціональні можливості IBM Message Broker	19.03-29.03.2021	
3	Дослідження функціональних можливостей IBM Message Broker	29.03-14.04.2021	
4	Розробка бізнес-процесу та специфікації по передачі переоцінки	14.04-28.04.2021	
5	Розробка ESQL коду	28.04-08.05.2021	
6	Розробка обов’язкових демонстраційних матеріалів	15.05.2021	
7	Попередній захист роботи	17.05-20.05.2021	
8	Здача роботи в деканат	24.05.2021	

Студент _____ _____
(підпис) (прізвище та ініціали)

Керівник роботи _____ _____
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Текстова частина бакалаврської роботи 44 с., 9 рис., 16 джерел.

Ключові слова: INTEGRATION, IBM, MESSAGE BROKER, АВТОМАТИЗАЦІЯ ДАНИХ, АНАЛІЗ, БІЗНЕС-ПРОЦЕС, РОБОЧИЙ ПРОЦЕС, СТВОРЕННЯ ЧЕРГ, ЦІНОУТВОРЕННЯ.

Об'єкт дослідження – передача даних між інформаційними системами.

Предмет дослідження – методи та засоби автоматизації на основі IBM Message Broker.

Мета роботи – автоматизація процесу передачі цін між різними інформаційними системами за допомогою IBM Message Broker.

Актуальність роботи – визначається постійно зростаючим обсягом даних у різних сферах діяльності. У зв'язку з розвитком направлення електронної комерції з'являється необхідність в оптимізації роботи по формуванню цінових пропозицій та їх передачі в реальному часі в різні інформаційні системи.

Завдання дослідження, які дозволяють досягти поставленої мети:

1. Аналіз існуючої ситуації та пошук рішень для оптимізації роботи компанії
2. Порівняння способів передачі даних
3. Аналіз функціональних можливостей IBM Message Broker.
4. Розробка бізнес-процесу та специфікації по передачі переоцінок на сайт та у POS-систему.
5. Розробка ESQL коду по запису даних в буферні таблиці.
6. Впровадження модуля в проекту експлуатацію.

Стислий опис результатів дослідження: реалізовано програмний модуль, який дозволяє автоматично оновлювати дані в різних інформаційних системах.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ERP (Enterprise Resource Planning) — планування ресурсів підприємства.

IBM (International Business Machines) — назва компанії.

CMS (Content Management System) — система управління контентом.

ESB (Enterprise Service Bus або Сервісна шина підприємства) — корпоративна шина даних.

POS (Post of sale) — місце продажу.

API (application programming interface) — інтерфейс прикладного програмування.

AI (Artificial Intelligence) — штучний інтелект.

ESQL (Embedded SQL) — розширена мова програмування.

ВСТУП

У сучасному світі необхідність автоматизації різних процесів стала вже звичним явищем. Бізнес зростає кожен день, шукаючи різні варіанти збуту своєї продукції. Тож, працюючи над продуктом, буду виявлено, проблему, що існує багато інформаційних систем, де потрібно швидко оновлювати дані. Крім того, всі вони ще і зберігаються у різних форматах, які не підтримують системи приймачі. Це не дозволяло швидко та якісно обробляти інформацію, саме тому було вирішено використовувати сполучне програмне забезпечення, що забезпечує централізований та уніфікований, орієнтований на події обмін повідомленнями між різними інформаційними системами на принципах сервіс-орієнтованої архітектури

Це рішення допоможе:

1. Мінімізувати трату людського часу.
2. Оновлювати дані водночас в різних інформаційних системах.
3. Застосувати це рішення для інших об’єктів передачі.

Об’єкт дослідження – передача даних між інформаційними системами.

Предмет дослідження – методи та засоби автоматизації на основі IBM Message Broker.

Мета роботи – автоматизація процесу передачі цін між різними інформаційними системами за допомогою IBM Message Broker.

Актуальність роботи – визначається постійно зростаючим обсягом даних у різних сферах діяльності. У зв’язку з розвитком направлення електронної комерції з'являється необхідність в оптимізації роботи по формуванню цінових пропозицій та їх передачі в реальному часі в різні інформаційні системи.

1. ВИЗНАЧЕННЯ ПОНЯТТЯ ЦІНИ ТА ПРОЦЕСУ ЦІНОУТВОРЕННЯ

1.1. Визначення та приклади ціноутворення

Ціноутворення відноситься до процесу прийняття рішень, який передбачає встановлення вартості товару чи послуги. Існує багато різних стратегій, які бізнес може використовувати при встановленні цін, але всі вони є формою ціноутворення. Ціна, встановлена в процесі ціноутворення, –це те, що платить клієнт за цей товар або послугу.

Існує багато методів ціноутворення, але здебільшого всі вони зводяться до якогось варіанту з трьох загальних підходів. При встановленні ціни власнику бізнесу потрібно враховувати широкий спектр факторів, включаючи витрати на виробництво та розподіл, пропозиції конкурентів, ідентифікацію ринку та цільову базу клієнтів.

Ціноутворення на продукцію – одна з найпоширеніших проблем, з якою стикаються підприємці та дрібні підприємці. І це завжди збиває їх з пантелику, бо вони не знають, як встановити ціну на товар, і на якій основі ці продукти цінують. А також стає проблема з передачею цих даних. Багато з них оцінюють свою продукцію, не вивчаючи ринок і не знаючи, які ціни конкурентів. Деякі з них встановлюють високі ціни, щоб заробити більше, і результатом є те, що вони не можуть нічого продати, оскільки їх ціни вищі, ніж у інших конкурентів. Деякі з них оцінюють свою продукцію за нижчими цінами, ніж конкуренти, щоб продати більше, і результат також полягає в тому, що вони не можуть нічого продати, оскільки споживачі вважають, що їх ціна низька, оскільки їх продукція має нижчу якість, ніж конкуренти.

Ціноутворення на продукцію є мистецтвом і не робиться випадковим чином, але воно повинно здійснюватися у науково правильному порядку, щоб досягти цілей власника проекту.

Потрібно взяти до уваги кілька речей перед ціноутворенням на свою продукцію, наприклад: характер товару, конкурентів, ринок, цільову аудиторію та визначити, яка мета бізнесу, і на якому етапі розвитку знаходиться бізнес.

1.2. Автоматизація процесу ціноутворення та оновлення даних

1.2.1. Огляд проблеми

Неможливо підвищити дохід підприємства без постійного пошуку проблем, які можна вирішити автоматизацією.

Призначення автоматизації даних – це процес завантаження та маніпулювання даними за допомогою автоматизованих інструментів, а не виконання всіх цих завдань людськими ресурсами. Автоматизація процесу пошуку даних не тільки економить час і гроші, але й підвищує ефективність бізнесу. Автоматизація даних також допомагає зменшити кількість помилок під час перевірки даних та забезпечує завантаження даних в організованому форматі.

Автоматизація даних пропонує великі стимули для бізнесу. Це економічно ефективне та продуктивне рішення для підприємства. Організація може отримати вигоду, заощаджуючи витрати та підвищуючи ефективність бізнесу за рахунок автоматизації деяких своїх процесів. Це також може бути корисним для працівників, які можуть зосередитися на складних та сильно стимулюючих заходах, а не виконувати нудні повторювані завдання. Більше того, автоматизація даних забезпечує узгодженість. Підтримка якості роботи є критично важливою для компаній, яку можна поставити під загрозу шляхом впровадження ручних процесів

Ситуація була така: в зв'язку з зростанням кількості SKU, кількості конкурентів та постачальників з’явилася необхідність автоматизувати цей процесс. Будь яке утворення ціни все одно береться враховуючи зовнішні чинники, так чому би не використовувати для цього спеціальні системи.

Як виявилось, створення ціни – це процес, який може вирішувати штучний інтелект.

1.2.2. Застосування штучного інтелекту в електронній комерції

Електронна комерція базується на принципі використання обчислювальних та комунікаційних технологій для консолідації економічної діяльності між деякими або всіма комерційними організаціями та їх клієнтами всіх рівнів. Технології штучного інтелекту почали впроваджуватись у багато секторів електронної комерції, таких як сектор торгівлі та бізнесу – проти споживчого B2C та торгово-діловий сектор проти торгівлі та бізнесу B2B.

Штучний інтелект використовується в першому секторі для полегшення завдання вибору продуктів. За умови забезпечення (автоматизованого) механізму прийняття рішень щодо цін на товари, що розміщуються на світовому ринку.

У другому секторі штучний інтелект використовується у Департаменті управління та організації ланцюгів поставок для розширення.

Сучасні програми штучного інтелекту засновані на аналізі історичних даних та даних у реальному часі, які будуть використовуватися для прогнозування того, як клієнти та конкуренти будуть мати справу з будь-якою зміною цін, що дає їм уявлення про динаміку ринку, часто перевершуючи можливості людського розуму.

Фірмам діагностується більше, ніж одна перешкода, пов’язана з визначенням ціни реалізації продукції у світлі існуючих цін. І характер пропонованих цін у тому випадку, якщо більше одного товару розміщується в одному продажу.

Надання розумних інструментів з хорошою здатністю автоматизувати відповідь допомагає усунути потребу у працівнику з передовим досвідом, який дозволяє йому відповідати на запитання клієнтів. Для того, щоб виробник гарантував задоволення споживача, він повинен мати чітке уявлення про характер товарів, які він може запропонувати.

Є також деякі програми, які дозволяють знати цінову політику конкурентів, а алгоритми також можуть знати, які товари купуються разом, що призводить до появи деяких продуктів, пропонованих сайтом, через який ви купуєте (наприклад, Amazon або Alibaba), як тільки ви завершите покупку товару. Ці алгоритми також підвищують ціни на деякі товари, коли конкуренти виявляють, що вони закінчуються

Динамічне ціноутворення (зазвичай його називають особистим ціноутворенням) – це цінова стратегія, при якій ціна визначається відповідно до попиту, товарних запасів та профілю замовника. Програми штучного інтелекту можуть аналізувати ваш профіль, використовуючи файли cookie, відвідувати історію, пошуки та інші цифрові дії, і відповідно динамічно визначатимуть ціни на товари. Прикладом використання динамічного ціноутворення є сайти бронювання готелів, оскільки ціни динамічно падають і зростають залежно від рівня заповненості номерів, туристичного сезону.

1.2.3. Мета автоматизації та її принципи

Основна мета автоматизації – поліпшення якості виконання процесу. Автоматичний процес є більш стабільним, ніж ручний. У багатьох випадках автоматизація процесів може підвищити продуктивність, скоротити час виконання процесу, зменшити витрати та підвищити точність та стабільність процесу.

Сьогодні автоматизація процесів охопила багато галузей та сфер діяльності: від виробничих процесів до магазинів. Незалежно від розміру та сфери діяльності організації, майже кожна компанія має автоматизовані процеси. Процесний підхід забезпечує стандартизовані принципи автоматизації для всіх процесів.

Хоча автоматизацію процесів можна здійснювати на різних рівнях, принципи автоматизації для всіх рівнів та всіх типів процесів залишатись однаковими. Це загальні принципи, що визначають умови для ефективного виконання операцій в

автоматичному режимі та встановлюють правила автоматичного управління процесом.

Основними принципами автоматизації процесів є:

1. Принцип узгодженості: усі дії в автоматизованому процесі повинні узгоджуватися між собою, а також з вхідними та вихідними даними процесу. Якщо процедури не збігаються, процес може не працювати.

2. Принцип інтеграції: автоматизований процес повинен мати можливість інтегруватися в загальне середовище підприємства. Інтеграція реалізується по-різному на різних рівнях автоматизації, але суть принципу залишається незмінною. Автоматизація процесів повинна забезпечувати взаємодію автоматизованого процесу із зовнішнім середовищем (стосовно цього процесу).

3. Принцип незалежності реалізації: процес, що підлягає автоматизації, повинен здійснюватися самостійно, без втручання людини або з найменшим контролем людини. Людина не повинна брати участь у процесі, якщо операція проводиться відповідно до зазначених вимог.

Перелічені загальні принципи деталізуються залежно від рівня, що враховується для автоматизації, і конкретних операцій. Наприклад, автоматизація виробничих процесів включає такі принципи, як принцип спеціалізації, принцип пропорційності, принцип безперервності тощо.

1.2.4. Управління робочим часом

Ефективне управління і облік робочого часу роблять позитивний вплив на всі бізнес-процеси компанії. Основним ресурсом будь-якої організації є її працівники. Прагнення максимально ефективно використовувати цей ресурс призвело до широкого використання систем автоматизації для управління операціями співробітників.

1.3. Огляд систем

В цьому розділі проведено огляд архітектури систем, які є ключовими в роботі Інтернет магазину. Нижче наведена схема передачі переоцінки між інформаційними системами:

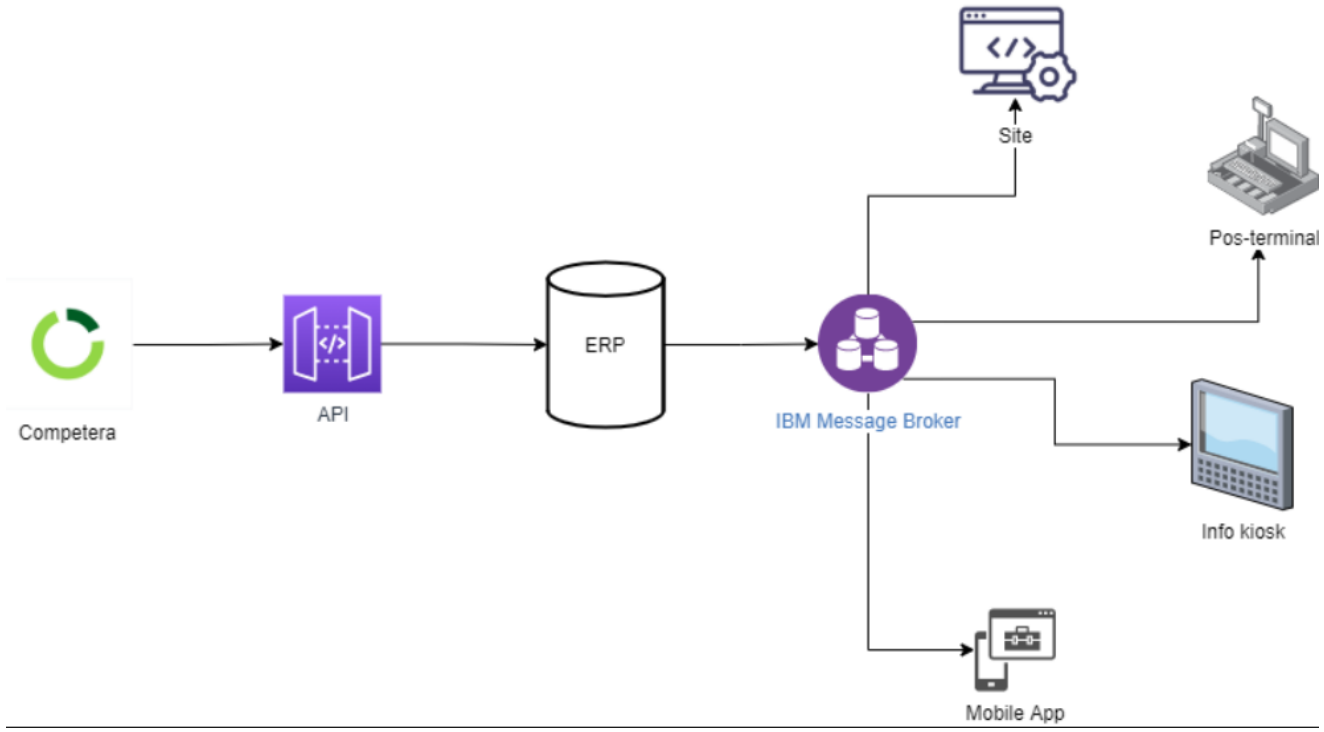

Рисунок 1.1 – Архітектура систем

В системі Competera відбувається ціноутворення за допомогою штучного інтелекту, далі ціни завантажуються в ERP систему використовуючи API, з майстер системи ERP ціни надходять в інформаційні системи, які позначені на малюнку: сайт, POS система, інфокіоск та мобільний додаток за допомогою IBM Message Broker. Нижче хочу розібрати детальніше основні системи, які задіяні в процесі.

1.3.1. ERP - система

Окремі функціональні сфери компанії, такі як продаж, управління матеріалами та управління людськими ресурсами, тривалий час працювали окремо одна від одної

та підтримувались так званими острівними рішеннями. Проблемою була ізольована обробка окремих підрозділів, так що невідповідності між даними зустрічалась доволі часто. Розробка ERP-систем змогла протистояти цьому. Завдяки взаємозв'язку між всіма процесами та процедурами вдалося створити одну майстер систему.

ERP-системи – це інтегровані програми, в основі яких лежить центральне управління даними. Тим самим інформація протікає в межах бізнес-процесу, утворюючи при цьому єдину майстер систему.

Система ERP зазвичай складається з декількох програмних підрозділів, які задовольняють потреби компанії. Кожен підрозділ системи ERP зосереджується на певній галузі. Нижче представлені прикладами найпоширеніших підрозділів:

- матеріали;
- контроль запасів;
- розподіл;
- бухгалтерський облік;
- фінанси та людські ресурси.

Бізнес також використовує групу різних підрозділів для управління офісною діяльністю та завданнями, включаючи наступне:

- управління поставками,ціноутворення;
- підвищення точності фінансової звітності;
- автоматизувати життєвий цикл працівника;
- стандартизувати важливі ділові процедури;
- скоротити зайві завдання;
- оцінити потреб бізнесу;
- формувати бухгалтерські та фінансові заяви;
- управляти людськими ресурсами та оплатою праці.

Функціональність та розмір ERP-систем може сильно різнитися між собою в залежності від профілю компанії. Однак більшість програм ERP мають такі характеристики:

1. Інтеграція на рівні підприємства:

Інтеграція між бізнес-процесами може проходити у різних бізнес-підрозділах. Наприклад, коли встановлюється ціна, то система може одразу перевірити, чи не менша вона від ціни закупівлі.

2. Операції в режимі реального часу:

Якщо розглянути приклад, який наведено вище, то ми можемо помітити, що оновлення цін в реальному часі є досить важливим, оскільки, це допоможе уникнути суперечливих ситуацій з клієнтами.

3. Одна загальна база даних:

Спільна база даних є головною перевагою ERP-систем. Оскільки ERP - це майстер система, то вона визначає розраховує дані, а потім використовує однакове значення для всіх підрозділів. Розділи повинні відповідати затвердженим стандартам спільних даних, а також правилам модифікації спільних даних. Хоча деякі ERP-системи продовжують покладатися на єдину базу даних, інші розділяють фактичну базу даних для підвищення продуктивності.

4. Послідовний зовнішній вигляд та інтерфейс:

Дуже важливо, щоб програмне забезпечення ERP мало стабільний користувальницький інтерфейс, це дозволить зменшити витрати на навчання працівників користуватися системою. Нові системи з ERP-системи зосереджені на стандартах безпеки, сучасних стандартах, розподілі службових обов'язків та підтримці сучасного законодавства [1].

ERP може підтримувати різноманітні структури ціноутворення, щоб задовольнити унікальні вимоги кожного бізнесу, які можуть принести прибуток. Функція управління ціноутворенням підтримує різні сценарії ціноутворення,

включаючи управління цінами, роздрібне ціноутворення, франшизне ціноутворення, оптове ціноутворення.

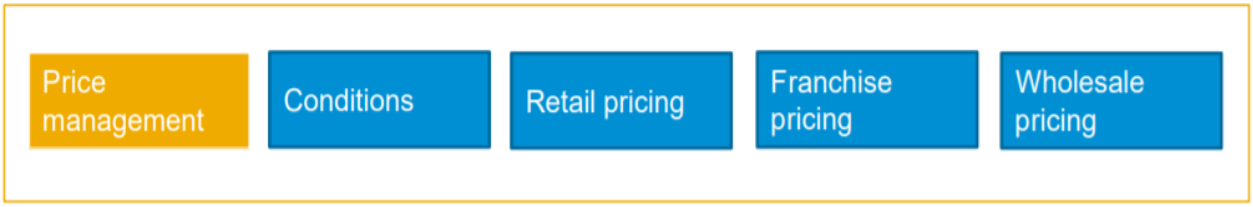

Рисунок 1.2 – Основні елементи ціноутворення в ERP

ERP може підтримувати нескінченну кількість електронних прайс-листів із унікальними датами активності, конкретні товари або групи товарів можуть одночасно знаходитись на нескінченній кількості прайс-листів, які будуть передаватися у різні системи також є можливість встановлювати знижки на основі відсотка, суми валюти або формули.

1.3.2. IBM Message broker

Шина інтеграції IBM Integration Advanced, раніше відома як WebSphere Message Broker – це службова шина обслуговування (ESB), що забезпечує підключення та універсальне перетворення даних для сервісно-орієнтованої архітектури (SOA) та середовищ, що не належать до SOA. Тепер підприємства будь-якого розміру можуть усунути з'єднання "точка-точка" та пакетну обробку незалежно від платформи, протоколу або формату даних.

Брокер повідомлень – це програмне забезпечення, яке дозволяє програмам, системам та послугам взаємодіяти між собою та обмінюватися інформацією. Брокер повідомлень робить це, перекладаючи повідомлення між офіційними протоколами обміну повідомленнями. Це дозволяє взаємозалежним службам «спілкуватися» безпосередньо між собою, навіть якщо вони написані різними мовами або реалізовані на різних платформах.

Брокери повідомлень мають можливість перевіряти, направляти, зберігати, та доставляти повідомлення до відповідних пунктів призначення. Вони служать посередниками між іншими програмами, дозволяючи відправникам видавати повідомлення, не знаючи, де знаходяться приймачі, чи вони активні чи ні, чи скільки їх існує. Це полегшує роз'єднання процесів та послуг у системах.

Брокери повідомлень – це програмні модулі в рамках проміжного програмного забезпечення для обміну повідомленнями або орієнтованого на повідомлення проміжного програмного забезпечення (MOM). Цей тип проміжного програмного забезпечення надає розробникам стандартизовані засоби обробки потоку даних між компонентами програми, щоб вони могли зосередитися на його основній логіці. Він може служити розподіленим комунікаційним рівнем, що дозволяє програмам, що охоплюють кілька платформ, взаємодіяти внутрішньо.

Щоб забезпечити гарантовану доставку та надійніше зберігання повідомлень, брокери повідомлень часто покладаються на структуру або компонент, який називається чергою повідомлень, який зберігає та впорядковує повідомлення, поки програми-споживачі не здатні їх обробити. У черзі повідомлення зберігаються в точному порядку, в якому вони були передані, і залишаються в черзі до підтвердження отримання [2].

Існує асинхронний обмін повідомленнями. Він відноситься до типу взаємодії між програмами, яку роблять можливі брокери повідомлень. Це запобігає втраті цінних даних і дозволяє системам продовжувати функціонувати навіть в умовах переривчастих проблем підключення або затримки, поширених у загальнодоступних мережах. Асинхронний обмін повідомленнями гарантує, що повідомлення будуть доставлені один раз (і лише один раз) у правильному порядку відносно інших повідомлень.

Брокери повідомлень мають можливість використовувати менеджерів черг для обробки взаємодій між декількома чергами повідомлень, а також служб, що

забезпечують переклад повідомлень, маршрутизацію даних, збереження та функціональність управління станом клієнта.

Брокери повідомлень пропонують дві основні схеми розподілу повідомлень або стилі обміну повідомленнями:

1. Повідомлення "точка-точка": це шаблон розподілу, який використовується в чергах повідомлень із взаємозв’язком один до одного між відправником та одержувачем повідомлення. Кожне повідомлення в черзі надсилається лише одному одержувачу і споживається лише один раз. Повідомлення "точка-точка" вимагається, коли повідомлення має діяти лише один раз. Приклади підходящих випадків використання для цього стилю обміну повідомленнями включають нарахування заробітної плати та обробку фінансових операцій. У цих системах і відправникам, і одержувачам потрібна гарантія того, що кожен платіж буде надісланий один раз і лише один раз.

2. Опублікувати/підписати повідомлення: У цьому шаблоні розповсюдження повідомлень, який часто називають "pub/sub", виробник кожного повідомлення публікує його в темі, і багато споживачів повідомлень підписуються на теми, з яких вони хочуть отримувати повідомлення. Усі повідомлення, опубліковані в темі, розподіляються серед усіх підписаних на неї програм. Це метод розповсюдження в стилі мовлення, при якому між видавцем повідомлення та його споживачами існує взаємозв'язок "один до багатьох". Наприклад, якщо авіакомпанія повинна поширювати оновлення про час посадки або статус затримки своїх рейсів, кілька сторін можуть скористатися цією інформацією: наземні екіпажі, що виконують технічне обслуговування та заправку літаків, обробки багажу, бортпровідники та пілоти, які готуються до польоту літака. наступної поїздки, а оператори візуальних дисплеїв сповіщають громадськість [3].

Сервісної шини підприємства (ESB) являє собою архітектурний шаблон, який використовується в сервіс-орієнтованих архітектурах, реалізованих між підприємствами. У ESB централізована програмна платформа поєднує протоколи

зв'язку та формати даних у "загальну мову", якою можуть спільно користуватися всі служби та додатки в архітектурі. Наприклад, він може перекладати запити, які він отримує, з одного протоколу (наприклад, XML) на інший (наприклад, JSON). ESB трансформують свої корисні навантаження повідомлень за допомогою автоматизованого процесу. Централізована програмна платформа також обробляє інші логічні схеми, такі як підключення, маршрутизація та обробка запитів.

Брокери повідомлень - це "легка" альтернатива ESB, яка забезпечує подібну функціональність - механізм міжвідомчих комунікацій - простіше і за нижчою вартістю. Вони добре підходять для використання в архітектурах мікропослуг, які стали більш поширеними, оскільки ESB не потрапили в немилість.

1.3.3. CMS

Система управління контентом (CMS) - це програмне забезпечення, призначене для допомоги користувачам у створенні та редагуванні веб-сайту. Важливо зазначити, що система управління вмістом робить набагато більше, ніж допомагає керувати текстовим та графічним вмістом, що відображається на веб-сторінках. Вони еволюціонували, щоб допомогти розробити зовнішній вигляд веб-сайтів, відстежувати сесії користувачів, обробляти пошуки, збирати коментарі відвідувачів, розміщувати форуми та багато іншого.

1.3.4. POS - система

Якщо почати інтегрувати свої офлайн-та онлайн-канали в одну торгову точку, то можна отримати цілісне уявлення про своїх клієнтів та бізнес. Саме тут нам і допоможе інтеграція в POS систему.

POS-програмне забезпечення зазвичай комплектується сумісним обладнанням для офлайн-магазину, наприклад, це може бути каса. POS-система може посилатись не тільки на платформу електронної комерції, але і на обслуговування клієнтів, управління запасами, бухгалтерський облік та встано [4].

Інтеграція з POS системою дозволяє не вводити великі масиви даних вручну. Можливість керувати інформацією про товар, його залишками та ціною в одному місці - це величезна користь для бізнесу.

1.4. Автоматизація процесу оновлення даних у системах

В цьому розділі я хочу оглянути декілька видів інтеграції, які найбільше підходять саме для електронної комерції. Найпопулярнішим звісно залишається інтеграція завдяки API. Хоча цей метод використовується досить давно, ще невідомо скільки він буде популярним, завдяки своїй швидкості та прототі у використанні [5].

У контексті електронної комерції API представляє інтерфейс, через який користувач отримує доступ до функціональних можливостей електронної комерції. Таким чином, цей інтерфейс дозволяє отримувати дані про товари, залишки, функції кошика, реєстрацію користувачів та багато іншого [6]. Але коли ми говоримо саме пропередачу великих масивів даних, таких як переоцінки, то хочется розглянути ще один інтерфейс передачі даних, а саме за допомогою інтеграційної шини IBM.

Тож для вирішення проблем передачі даних можемо розглянути передачу за допомогою повідомлень [7]. Продукти які можуть представити такі послуги можна розділити на два типи: послуги черг повідомлень (SQS Amazon, MQ) та сервісні шини підприємств (ESB).

Рисунок 1.3 - Принцип роботи SQS Amazon

Загальний сценарій інтеграції зазвичай виглядає так: система підключається до інтеграційної шини за допомогою конекторів. Задачею системи-джерела є передача даних у коннектор, а от вже трансформація, маршрутизація та доставка повідомлень

у системи-споживачі здійснюються [8] без її участі. Головною задачею конектора - забезпечення каналу прийому даних у систему та передача даних з неї.

Таким чином, я можу стверджувати, що обмін даними за допомогою механізму обробки повідомлень та складання їх у чергу, реалізований сервісними шинами, є вартим уваги [9]. Окрім переваг зазначених вище, шина даних забезпечує конфіденційність інформації завдяки тому, що шифрує дані. Також сервісна шина може гарантувати доставку даних, завдяки тому що ім просто нема куди дітися, якщо повідомлення не буде доставлено, то воно може просто попасти в чергу “невідправлених” повідомлень.

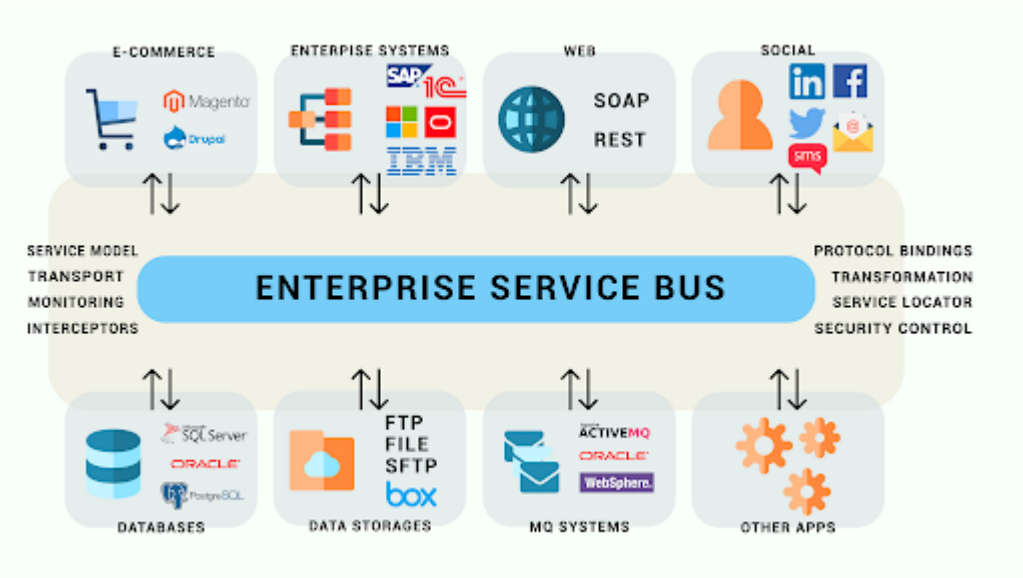

Рисунок 1.4 - Можливості сервісної шини

У наступному розділі ми розглянемо функціональні можливості IBM Integration Bus. Такий вибір може істотно заощадити час, витрати та швидкість оновлення даних.

2. ДОСЛІДЖЕННЯ ФУНКЦІОНАЛЬНИХ МОЖЛИВОСТЕЙ IBM INTEGRATION BUS

2.1. Огляд шина інтеграції IBM

IBM Integration Bus використовується для з'єднання програм разом, незалежно від форматів повідомлень або протоколів, які вони підтримують.

Це підключення означає, що різноманітні програми можуть взаємодіяти та обмінюватися даними з іншими програмами в гнучкій, динамічній та розширювальний інфраструктурі. Шина IBM Integration Bus маршрутизує, перетворює та збагачує повідомлення з одного місця в будь-яке інше:

Продукт підтримує широкий спектр протоколів: WebSphere MQ, JMS 1.1 та 2.0, HTTP та HTTPS, веб-сервіси (SOAP та REST), файлові, корпоративні інформаційні системи (включаючи ERP та Siebel) та TCP / IP.

Він підтримує широкий спектр форматів даних: двійкові формати (C та COBOL), XML та галузеві стандарти (включаючи SWIFT, EDI та HIPAA). Також можливо визначити власні формати даних.

Він підтримує багато операцій, включаючи маршрутизацію, перетворення, фільтрацію, збагачення, моніторинг, розподіл, збір, кореляцію та виявлення. Взаємодії з IBM Integration Bus можна розглянути в двох широких категоріях [10].

Розробка, тестування та розгортання додатків. Ми можете використовувати один або декілька з наданих варіантів для програмування своїх програм:

1. Шаблони надають багаторазові рішення, які містять перевірений підхід до вирішення архітектурної, дизайнерської або загальної задачі в певному контексті. Можна використовувати їх без змін або модифікувати відповідно до власних вимог.

2. Потоки повідомлень описують логіку підключення програми, яка визначає точний шлях, який проходять дані у вузлі інтеграції, а отже, обробку, яка застосовується до нього вузлами повідомлень у цьому потоці.

3. Вузли повідомлень інкапсулюють необхідну логіку інтеграції, яка діє на дані, коли вони обробляються через вузол інтеграції.

4. Дерева повідомлень описують дані ефективно, незалежно від формату. Можна вивчити та змінити вміст дерев повідомлень у багатьох наданих вузлах, а також можливо надати додаткові вузли для власного проекту.

5. Перетворення можна здійснити, використовуючи графічне відображення, Java, ESQL та XSL, і можна зробити вибір, виходячи з навичок робочої сили, не вимагаючи перепідготовки.

Оперативне управління та продуктивність. Шина інтеграції IBM включає такі функції та функціональність, які підтримують роботу та продуктивність розгортання:

1. Широкий спектр варіантів адміністрування та управління системами для розроблених рішень.

2. Підтримка широкого кола операційних систем та апаратних платформ.

3. Масштабована, високопродуктивна архітектура, заснована на вимогах традиційних середовищ обробки транзакцій.

4. Тісна інтеграція з програмними продуктами від IBM та інших постачальників, які надають відповідні послуги з управління та підключення.

Додатки для обробки повідомлень, які можна запускати на більш ніж 30 галузевих платформах, можуть підключатися до вузла інтеграції, використовуючи один із підтримуваних протоколів, уже перерахованих. Підтримуються платформи від IBM, Microsoft, Oracle та інших.

Різноманітні програми можуть обмінюватися інформацією в дуже різних форматах, при цьому вузли інтеграції обробляють обробку, необхідну для надходження інформації у потрібне місце у правильному форматі, відповідно до визначених вами правил. Додатки повинні розуміти лише власні формати та протоколи, а не стандарти, що використовуються програмами, до яких вони підключені.

Додатки також мають набагато більшу гнучкість у виборі повідомлень, які вони хочуть отримувати, оскільки можна застосовувати фільтри для управління повідомленнями, які їм надаються.

Шина інтеграції IBM забезпечує структуру, яка містить широкий спектр основних функцій, що постачаються, разом із визначеними користувачем удосконаленнями, щоб забезпечити швидку побудову та зміну правил обробки повідомлень.

Програми можна інтегрувати, забезпечуючи перетворення повідомлень та даних в одному місці, на вузлі інтеграції [11]. Ця інтеграція допомагає зменшити витрати на оновлення та модифікації програм. Також можна розширити свої системи, щоб охопити своїх постачальників та клієнтів, задовольнивши їхні вимоги щодо інтерфейсу у ваших вузлах інтеграції. Ця здатність може допомогти вам поліпшити якість вашої взаємодії та дозволити більш швидке реагування на зміни або додаткові вимоги [12].

Повідомлення обробляються відповідно до правил, які ви визначаєте за допомогою IBM Integration Toolkit.

Шина інтеграції IBM підтримує вибір інтерфейсів для роботи та адміністрування вузлів інтеграції:

1. Набір інструментів інтеграції IBM.
2. Веб-інтерфейс користувача, який можна використовувати для адміністрування вузлів інтеграції.
3. Програми, що використовують IBM Integration API.
4. Комплексний набір команд, які можна запускати інтерактивно або за допомогою сценаріїв.
5. Representational State Transfer API (REST) дозволяє розробляти адміністративні додатки без необхідності установки клієнтського програмного забезпечення і веб - браузери можуть управляти інтеграційними вузлами за допомогою призначеного для користувача інтерфейсу.

2.2. Технічний огляд шини інтеграції IBM

Шина інтеграції IBM дозволяє передавати інформацію, упаковану у вигляді повідомлень, між різними бізнес-додатками, починаючи від великих традиційних систем і закінчуючи безпілотними пристроями, такими як датчики на трубопроводах.

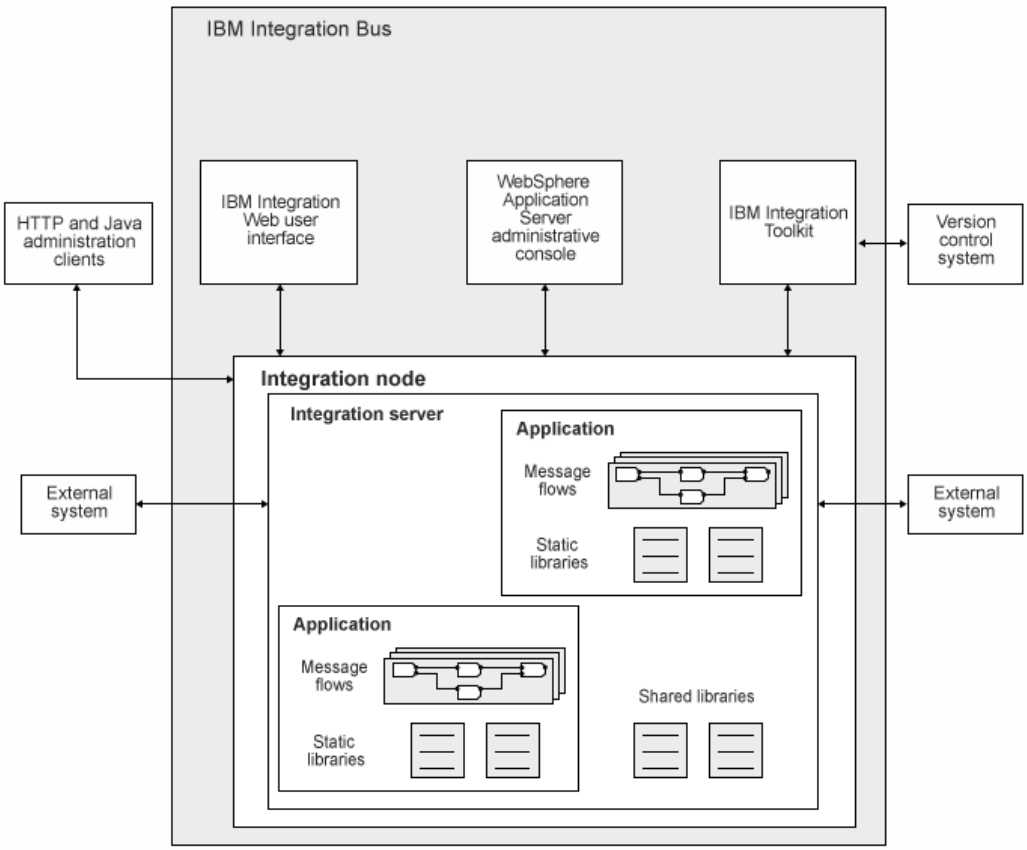

Рисунок 2.1 - Основні компоненти шини інтеграції IBM та їх взаємодія.

Шина інтеграції IBM обробляє повідомлення двома способами: маршрутизацією повідомлень та перетворенням повідомлень.

2.2.1. Маршрутизація повідомлень

Повідомлення можна перенаправляти від відправника до одержувача на основі вмісту повідомлення. Повідомлення протікає, а ми можемо керування

маршрутизацією повідомлень. Потік повідомлення описує операції, що виконуються над вхідним повідомленням, і послідовність, в якій вони виконуються.
Кожен потік повідомлень складається з таких частин:

1. Серія кроків, що використовуються для обробки повідомлення.
2. Зв'язки між вузлами, що визначають маршрути через обробку.

IBM постачає вбудовані вузли та зразки для багатьох загальних функцій.

2.2.2. Перетворення повідомлень

Повідомлення можна трансформувати перед доставкою:

1. Вони можуть трансформуватися з одного формату в інший, можливо, з урахуванням різних вимог відправника та одержувача.
2. Вони можуть бути перетворені шляхом модифікації, комбінування, додавання або видалення полів даних, можливо, з використанням інформації, що зберігається в базі даних.

Інформація може бути зіставлена між повідомленнями та базами даних. Більш складна маніпуляція з даними повідомлень може бути досягнута шляхом написання коду, наприклад, у розширеному SQL (ESQL) або Java, в межах конфігурованих вузлів [12].

Перетворення можуть здійснюватися різними вузлами в потоці повідомлень. Перш ніж вузол потоку повідомлень зможе оперувати вхідним повідомленням, він повинен зрозуміти структуру цього повідомлення.

1. Деякі повідомлення містять визначення власної структури та формату. Ці повідомлення відомі як самовизначальні повідомлення, які можна обробляти без потреби в додатковій інформації про структуру та формат.
2. Інші повідомлення не містять інформації про їх структуру та формат. Для їх обробки потрібно створити модель їх структури.

Як і потоки повідомлень, моделі повідомлень створюються в IBM Integration Toolkit. Вони можуть містити два типи інформації:

- логічна структура: абстрактне розташування та характеристики даних, представлені у вигляді деревної структури;
- один або кілька фізичних форматів: спосіб представлення та розмежування даних у фізичному потоці бітів.

2.2.3. Особливості інтеграції IBM

Деякі функції шини інтеграції IBM мають конкретні вимоги, такі як певний режим роботи або наявність додаткового продукту, наприклад бази даних або доступу до WebSphere MQ.

Додаткові умови та положення також застосовуються до використання деяких функцій IBM Integration Bus [13], включаючи вимоги щодо інших ліцензій на продукт.

Можна вибрати режим роботи для своїх вузлів інтеграції, виходячи з необхідних вам функціональних можливостей та потужності:

- у експрес- режимі вузол інтеграції працює з обмеженим набором вузлів для використання з одним сервером інтеграції;
- у масштабному режимі вузол інтеграції працює з обмеженим набором вузлів для використання з необмеженими серверами інтеграції.
- у стандартному режимі вузол інтеграції працює з усіма увімкненими функціями, але ви обмежені створенням лише одного сервера інтеграції;
- у розширеному режимі вузол інтеграції працює з усіма увімкненими функціями, і немає обмеження на кількість серверів інтеграції або кількість потоків повідомлень, розгорнутих на кожному сервері інтеграції.

Нижче наведено схеми режимів роботи, яку підтримує IBM Integration BUS. Основній режими же: стандартний режим, експрес режим, розширений режим та режим, який можна масштабувати.

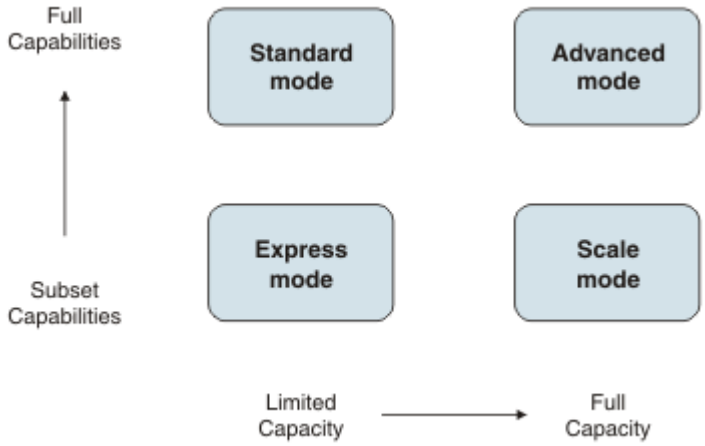

Рисунок 2.2 - Режими роботи

Огляд основних можливостей шини інтеграції IBM та показано режими роботи, в яких вони доступні представлені в таблиці 2.1.

Таблиця 2.1 - основні можливості шини інтеграції IBM та їх режими роботи

Можливість	Опис	Стандартний режим	Розширений режим	Режим інтеграції додатків
Вертикальне масштабування	Можна використовувати необмежену кількість серверів інтеграції.	Немає	Так	Так

Підключення до веб-стандартів	Надаються такі вузли: - HTTPAsyncRequest - HTTPAsyncResponse - HTTPHeader - HTTPInput - HTTPReply - HTTPRequest - SOAPsyncRequest - SOAPAsyncResponse - SOAPEnvelope - SOAPExtract - SOAPInput - SOAPReply - SOAPRequest Також надається підтримка REST, XML та JSON.	Так	Так	Так
REST зв’язок	Надаються такі вузли: - RESTRequest - RESTAsyncRequest - RESTAsyncResponse	Так	Так	Так
Підключення до електронної пошти,	Надаються такі вузли: - EmailOutput - FileInput - FileOutput	Так	Так	Так

файлів та (S) FTP				
SQL	Надаються такі вузли: - Обчислювальний - Бази даних - Фільтра Також надається підтримка розгортання бази даних ODBC та ESQL.	Так	Так	Так

Всі вузли повідомлень, що наведені нижче можуть використовуватися для різних цілей, а також мати своє призначення при використанні їх в потоці повідомлень.

2.3. Середовище інтеграційної IBM шини

Робота маршрутизації та перетворення повідомлень відбувається у вузлі інтеграції. У межах вузла інтеграції можна визначити один або кілька серверів інтеграції, які є процесами, в яких виконуються потоки повідомлень.

Режим, в якому вузол інтеграції працює може вплинути на кількість інтеграції серверів і потоки повідомлень, які можна розгорнути, і тип вузлів, які можна використовувати.

Можна встановити та створити один або кілька вузлів інтеграції на одному або кількох комп’ютерах, на яких працює підтримувана операційна система. Якщо створювати кілька вузлів інтеграції, можна налаштувати своє середовище таким чином, щоб забезпечити захист від збоїв, також можливо розподілити роботу між різними підрозділами в бізнесі.

Адміністрюються вузли інтеграції за допомогою команд продукту або IBM Integration API у власних програмах [14].

Вузол інтеграції являє собою сукупність процесів виконання, що хости один або кілька потоків повідомлень для маршрутизації, перетворення і збагачують в повідомленнях польоту.

Створення декількох вузлів інтеграції може забезпечити балансування навантаження або розподіл відповідальності. Наприклад, у вас може бути один вузол інтеграції, який обробляє всі ваші фінансові програми, а інший - обробку та виконання ваших замовлень.

Прикладні програми підключаються та надсилають повідомлення до вузла інтеграції та отримують повідомлення від вузла інтеграції. Кодування своїх програм потрібно, щоб використовувати один із підтримуваних протоколів для взаємодії з вузлом інтеграції; наприклад, черги та підключення WebSphere MQ, веб-служби або адаптери WebSphere. Вузол інтеграції маршрутизує кожне повідомлення за допомогою правил, визначених у потоках повідомлень та файлах схем моделі повідомлень, і перетворює дані у структуру, необхідну для програми-одержувача.

Встановити IBM Integration Bus можна на одній або декількох підтримуваних платформах, перелічених у системних вимогах IBM Integration Bus. Створити вузол інтеграції можна лише на комп’ютері, на якому встановлено шину інтеграції IBM. Використовується IBM Integration Toolkit або командний рядок для створення локальних вузлів інтеграції.

Адміністрування вузла інтеграції за допомогою інтеграції Nodes уявлення в IBM Integration Toolkit або команді продукту. Крім того, можна написати власні програми для використання IBM Integration API.

Керувати ресурсами програми вузла інтеграції, які включають потоки повідомлень та файли схем моделі повідомлень можна за допомогою IBM Integration Toolkit або веб-інтерфейсу користувача.

На наступному малюнку показано взаємозв'язок між ресурсами, які існують під час виконання, та тим, як вони взаємодіють з інтеграційним набором IBM та веб-інтерфейсом користувача [15].

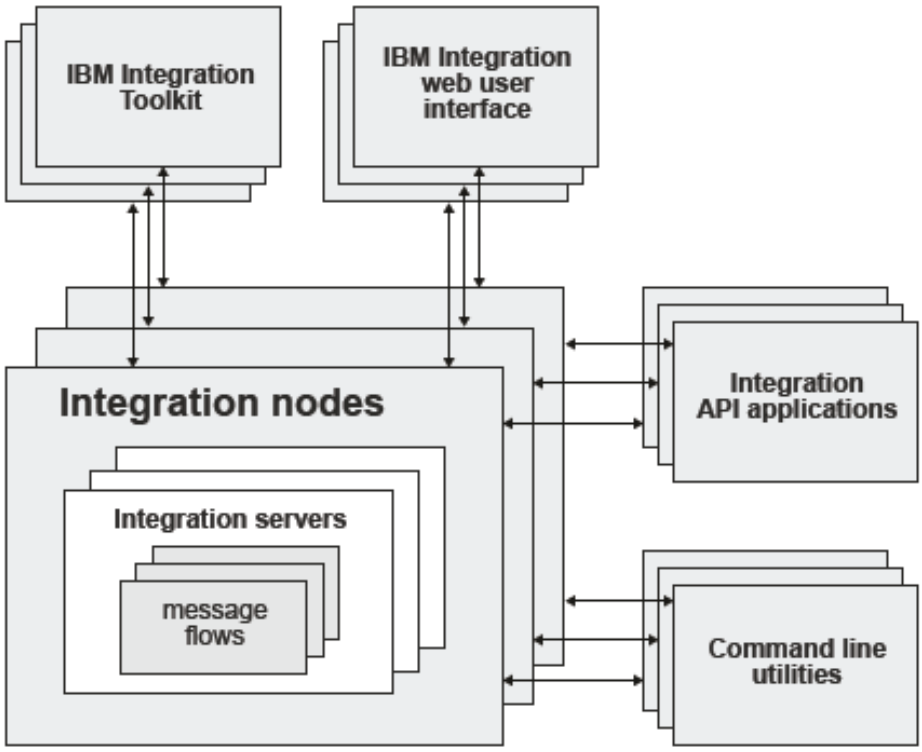

Рисунок 2.3 - Взаємозв'язок між ресурсами

2.3.1. Перетворення повідомлень

Вузли інтеграції надають послугу для незалежних агентів управління системою. Ця послуга надає центральному засобу управління доступ до інформації про мережу, яка включає один або кілька вузлів інтеграції. Таким чином, можна розширити свої існуючі агенти управління системою, включивши ресурси IBM Integration Bus.

Вузли інтеграції публікують повідомлення про події з використанням фіксованих тем у відповідь на зміни конфігурації, зміни стану та дії користувачів, такі як реєстрація передплати. Вузли інтеграції також використовують архітектурні

повідомлення для публікації подій, пов’язаних з їхнім робочим станом та змінами в цьому статусі. Ці повідомлення публікуються із використанням зарезервованого кореня теми $SYS на кодовій сторінці 1208. Лише для видавця чи допоміжного брокера MQTT коренем теми є IntegrationBus.

Приклад фіксованої теми з брокером MQ pub/sub:

```
$SYS/Broker/integrationNodeName/Status/ExecutionGroup/integrationServerName
```

Приклад фіксованої теми з пабом / допоміжним брокером MQTT:

```
IBM/IntegrationBus/integrationNodeName/Status/ExecutionGroup/integrationServerName
```

Прикладом фактичних даних повідомлень для цієї публікації є:

```
<Broker uuid="01234567-1234-1234-1234-1234567890">
        <ExecutionGroup uuid="01234567-1234-1234-1234-1234567890">
                <Stop>
                        <AllMessageFlows/>
                </Stop>
        </ExecutionGroup>
</Broker>
```

Повідомлення про події мають фіксовану структуру, яка визначена в XML (Розширювана мова розмітки). Формат цих повідомлень, які побудовані у форматі XML, детально описується в тілі повідомлення XML. Деякі повідомлення про події також публікуються у форматі JSON (JavaScript Object Notation), що містить ту саму інформацію, що і відповідне повідомлення XML. Повідомлення охоплюють зміни конфігурації, зміни стану, сповіщення про помилки, а також детальну інформацію про передплату та тему (наприклад, реєстрацію передплати).

2.3.2. Сервери інтеграції

Сервер інтеграції - це іменоване групування потоків повідомлен, призначених вузлу інтеграції. Вузол інтеграції забезпечує ступінь ізоляції між потоками повідомлень на різних серверах інтеграції, забезпечуючи їх роботу в окремих адресних просторах або як унікальні процеси.

Кожен сервер інтеграції запускається як окремий процес операційної системи, що забезпечує ізольоване середовище виконання для набору розгорнутих потоків повідомлень. На сервері інтеграції призначені потоки повідомлень виконуються в різних пулах потоків. Також можна вказати розмір пулу потоків (тобто кількість потоків), призначених для кожного потоку повідомлень, вказавши кількість додаткових екземплярів кожного потоку повідомлень.

Режим, коли ваш вузол інтеграції працює в може вплинути на кількість серверів інтеграції, які можна використовувати.

Налаштувавши додаткові сервери інтеграції, можна ізолювати потоки повідомлень, які обробляють конфіденційні дані, такі як записи про заробітну плату, або інформацію про безпеку, або неоголошену інформацію про товар, від інших нечутливих потоків повідомлень.

Якщо ви створюєте додаткові сервери інтеграції, ви повинні дати кожній групі унікальне ім'я у вузлі інтеграції та призначити та розгорнути один або кілька потоків повідомлень кожному.

Також можна створювати та розгортати сервери інтеграції або в IBM Integration Toolkit, або за допомогою команд [16].

Процес сервера інтеграції також відомий як DataFlowEngine (DFE), цей термін зазвичай використовується у сценаріях визначення проблем (вміст трасування, діагностичні повідомлення тощо). DFE створюється як процес операційної системи і має взаємозв'язок "один на один" із названим сервером інтеграції. Якщо на сервері

інтеграції працює більше одного потоку повідомлень, у процесі DFE створюється кілька потоків.

2.3.3. API інтеграції IBM

IBM Integration API є віддаленим інтерфейсом програмування, який користувальницькі додатки інтеграції можуть використовувати для управління інтеграцією вузлів і їх ресурсами.

IBM Integration API складається тільки з Java ™ реалізації. Ваші користувальницькі програми інтеграції мають повний доступ до функцій вузла інтеграції та ресурсів через набір класів Java, що становлять API IBM Integration.

Використовується IBM Integration API для взаємодії з вузлом інтеграції для виконання таких завдань:

- розгортання файлів BAR;
- змініть властивості конфігурації вузла інтеграції;
- створюйте, змінюйте та видаляйте сервери інтеграції;
- перегляньте журнал адміністрування;
- перегляньте журнал активності;
- створювати та модифікувати програми потоку повідомлень.

2.3.4. Веб-інтерфейс користувача IBM Integration Bus

Веб-інтерфейс IBM Integration Bus призначений для користувача дозволяє користувачам вузла ресурсів інтеграції доступу за допомогою веб - браузера, і надає адміністраторам інтеграції з альтернативою використання команди для вузла ресурсів інтеграції адмініструванні.

За допомогою веб-інтерфейсу користувача можна виконувати такі адміністративні завдання:

- Керування ресурсами вузла інтеграції:

1. Переглядати, створювати, перейменовувати, запускати, зупиняти та видаляти сервери інтеграції.
2. Розгортати ресурси.
3. Переглядати, запускати, зупиняти, керувати та видаляти розгорнуті ресурси.
4. Переглядати, створювати, редагувати та видаляти настроювані послуги.
5. Створювати, отримувати, оновлювати та видаляти операційні політики.
6. Редагувати деякі властивості вузлів інтеграції, серверів інтеграції та потоків повідомлень.
7. Працювати зі статистикою та даними бухгалтерського обліку для ваших потоків повідомлень. Також можна розпочати та зупинити збір статистичних даних знімків та даних бухгалтерського обліку, а потім відобразити дані у форматі, який допоможе вам проаналізувати та налаштувати ефективність ваших потоків повідомлень та програм.
8. Працювати зі статистикою для ресурсів, які використовуються серверами інтеграції. Також можна розпочати та зупинити збір статистики ресурсів, а потім відобразити дані у форматі, який допоможе проаналізувати та налаштувати ефективність вузлів інтеграції. Для отримання додаткової інформації потрібно дивитись аналіз продуктивності ресурсів та налаштування вузла інтеграції.
9. Видалити (експортувати) визначення одного або декількох розгорнутих REST API до IBM API Connect.

Бізнес-користувачі можуть використовувати веб-інтерфейс користувача для визначення та моніторингу ділових операцій.

2.4. Взаємодія з базами даних за допомогою ESQL

Оператори та функції ESQL можна використовувати для читання, запису та редагування баз даних із потоків повідомлень.

ESQL має ряд операторів та функцій для доступу до баз даних:

1. Оператор CALL викликає збережену процедуру.
2. Оператор DELETE FROM видаляє рядки з таблиці бази даних.
3. Оператор INSERT додає рядок до таблиці бази даних.
4. Для здійснення складних виборів можна використовувати функцію PASSTHRU.
5. Також PASSTHRU можна використовувати для виклику адміністративних операцій (наприклад, створення таблиці).
6. Функція SELECT отримує дані з таблиці.
7. Оператор UPDATE змінює одне або кілька значень, що зберігаються в нулі або більше рядків.

Отримати доступ до баз даних можна із вузлів Compute, Database, DatabaseInput та Filter. У всіх цих вузлах підтримуються однакові функції та процедури ESQL. Дані в базах можна використовувати для оновлення або створення повідомлень.

ESQL підтримує шість типів даних:

1. Boolean.
2. Datetime.
3. Null.
4. Numeric.
5. Reference.
6. String.

Дані, які отримуються з баз даних, отримуються в самовизначальному повідомленні або визначаються в моделі повідомлень (із використанням типів даних MRM), відображаються в один із цих основних типів ESQL, коли вони обробляються у виразах ESQL.

У межах вузла інтеграції поля повідомлення містять дані, які мають певний тип даних. Також можна використовувати проміжні змінні, які допомагають обробляти повідомлення, полегшуючи процесс. Потрібно оголосити всі такі змінні з типом даних перед їх використанням, як і в інших мовах. Поля повідомлень не мають

фіксованого типу даних, і можна призначити значення якогось іншого типу. Тоді поле приймає нове значення та тип. Не завжди можливо передбачити тип даних, який є результатом обчислення виразу. Це тому, що вирази компілюються без посилання на будь-яку схему повідомлень, і тому в процесі роботи можуть виникнути деякі помилки.

3. РОЗРОБКА БІЗНЕС-ПРОЦЕСУ ТА МОДУЛЯ ПО ОНОВЛЕННЮ ДАНИХ В ІНФОРМАЦІЙНИХ СИСТЕМАХ

3.1. Концепція оновлення даних у режимі реального часу

Дані в реальному часі - це дані, які оновлюються та переглядаються в той момент, коли вони доступні. Існує багато загальних застосувань даних у режимі реального часу.

Обробка в режимі реального часу виконується для потоків даних, одержаних в реальному часі і оброблюваних з мінімальною затримкою для створення звітів або автоматизованого редагування в режимі реального часу (або наближення до реальної часу). Наприклад, рішення для оновлення цін в режимі реального часу дозволяє уникнути неприємних ситуацій з боку клієнта. Уявімо, що клієнт прийшов до магазину за певною річчю, яка сподобалася йому в інтернеті, але в магазині на неї зовсім інша ціна - це може викликати неприємну реакцію у клієнтів магазину. Також швидке оновлення цін позитивно впливає на прибуток компанії, оскільки дає можливість швидко оновлювати дані при зміні курсу валют.

Обробкою в режимі реального часу вважається обробка неприв’язаного потоку вхідних даних з мінімальним часом затримки - кілька мілісекунд або секунд. Ці вхідні дані зазвичай надходять в структурованому або напівструктурованому форматі, наприклад JSON. Вимоги до обробки тут такі ж, як і в разі пакетної обробки, крім додаткового обмеження по часу для підтримки використання в режимі реального часу.

Оброблені дані зазвичай зберігаються в сховищі аналітичних даних, яке оптимізовано для аналізу і візуалізації. Також оброблені дані можна безпосередньо приймати на шарах аналітичної обробки і звітності для аналізу, бізнес-аналітики і візуалізації на панелях моніторингу в режимі реального часу.

Архітектура обробки в режимі реального часу складається з наступних логічних компонентів.

1. Прийом повідомлень в режимі реального часу. Архітектура повинна включати кошти збору і збереження повідомлень в режимі реального часу, доступні для об'єкта-одержувача, який займається обробкою потік. У простих випадках роль такої служби може виконувати звичайне сховище даних, в одній з папок якого розміщуються нові повідомлення. Але частіше для цього компонента потрібні спеціалізовані брокери обміну повідомленнями, наприклад Центри подій Azure, які виконують роль буфера для вхідних повідомлень. Брокер обміну повідомленнями повинен підтримувати масштабовану обробку і надійну доставку.
2. Передача потокового обробка. Зберігши повідомлення, що надходять в режимі реального часу, система виконує для них фільтрацію, статистичну обробку та інші процеси підготовки даних до аналізу.
3. Сховище аналітичних даних. Багато рішень по обробці великих спроектовано так, щоб готувати дані до аналізу і надавати їх в структурованому форматі для запитів через засоби аналітики.
4. Аналіз і звітність, більшість рішень по обробці великих даних призначені для аналізу та складання звітів, що дозволяє отримати важливу інформацію.

3.2. Основні аспекти запропонованої розробки

В час стрімкого розвитку електронної комерції зростає кількість торгових пропозицій. Пояснюється це розвитком асортименту компанії, співпраця з партнерами marketplace та збільшенням каналів збуту. Велику цінність має час, затрачений спеціалістом на виконання певних задач. Ціноутворення та передача цін в облікові системи потребує великої кількості людських ресурсів, саме тому постає задача в автоматизації цих процесів.

Метою дослідження є автоматизація процесу передачі даних між різними інформаційними системами за допомогою IBM Message Broker. Працюючи над продуктом, буду виявлено, що існує багато джерел інформації, які зберігають дані у своїх певних форматах, а система приймач їх не підтримує. Це не дозволяє швидко та якісно обробляти інформацію, саме тому було вирішено використовувати сполучне програмне забезпечення, що забезпечує централізований та уніфікований, орієнтований на події обмін повідомленнями між різними інформаційними системами на принципах сервіс-орієнтованої архітектури.

Виникла потреба в реалізації рішення, яке б дозволяло як отримувати від майстер системи файли великого розміру (> 100 Mb), так і передавати їх в інші системи. З причини ненадійного мережевого каналу між мережею клієнта і мережею додатки використовується система обміну повідомленнями, що забезпечує гарантовану доставку між ними.

З причини великого розміру переданих файлів відсутня можливість розміщення його повністю в оперативній пам'яті, більш того, з боку MQ також накладається обмеження - максимальний розмір одного повідомлення в MQ не може перевищувати 100 Mb. Таким чином моє рішення буде грунтуватися на наступних принципах:

- отримання файлу і збереження його в MQ черзі має виконуватися в потоковому режимі, без приміщення в пам'ять повністю файлу;
- у черзі MQ файл буде розміщуватися у вигляді набору невеликих повідомлень.

3.2.1. Бізнес процес

Бізнес-процес або бізнес-метод - це сукупність пов’язаних, структурованих видів діяльності чи завдань людей чи обладнання, які в певній послідовності виробляють послугу чи товар (служать певній бізнес-меті) для конкретного клієнта чи клієнтів.

Нижче представлений бізнес процес передачі переоцінки на сайт та у POS систему, який розроблений у системі Creatio:

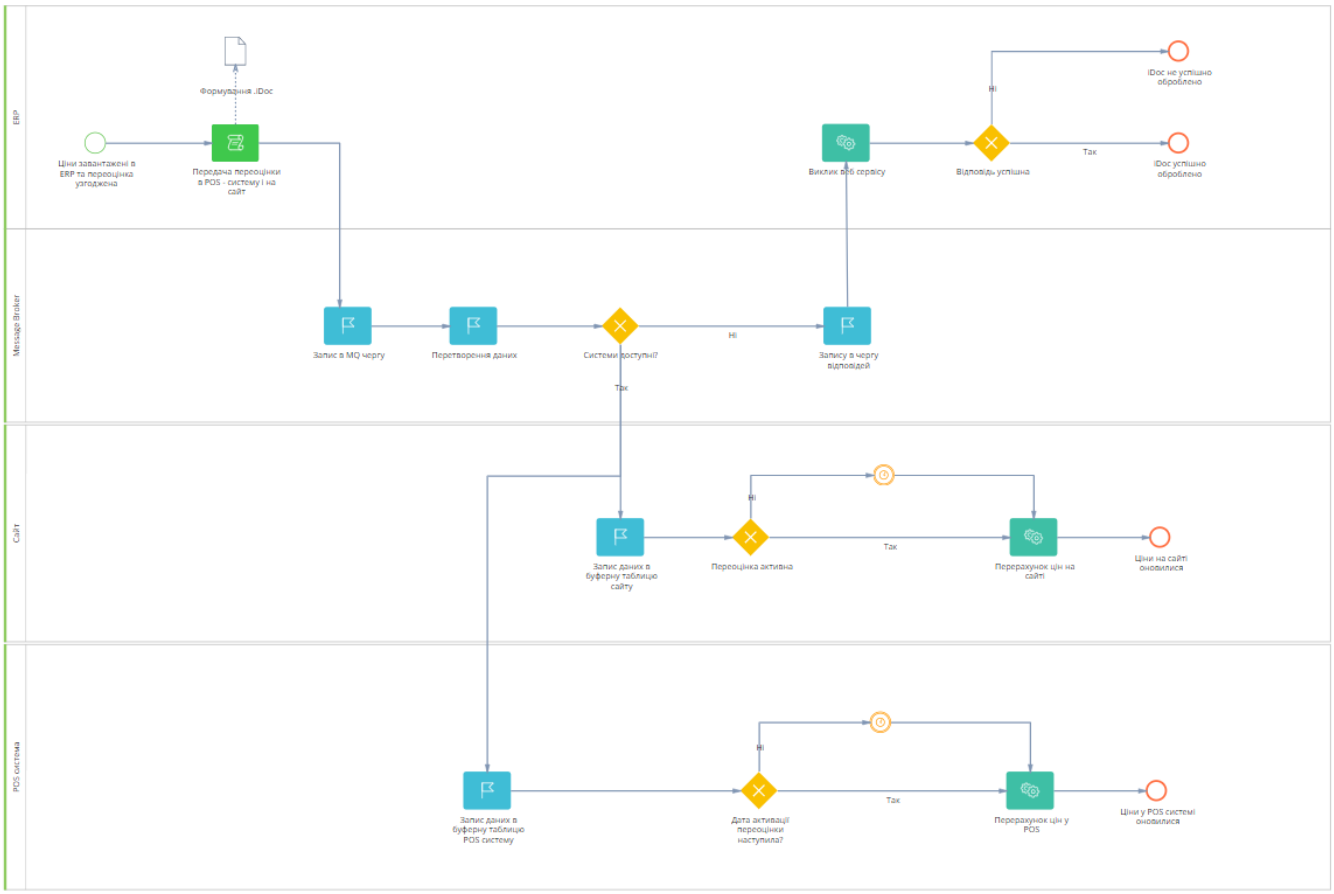

Рисунок 3.1 - Бізнес процес оновлення цін в інформаційних системах

Опис кроків бізнес процесу:

1. Переоцінка угоджена. Ціни завантажені в ERP систему.
2. Завдання призначене для користувача. Запуск передачі переоцінки. На цьому етапі формується .iDoc.
3. Запис файлу у MQ чергу.
4. Перетворення даних.
5. Перевірка доступності систем. Якщо системи доступні п.6., якщо ні - запис даних у чергу відповідей, виклик веб сервісу по поверненню файлу.

6. Завантаження даних у базу даних сайту та POS системи.
7. Перевірка дати активності переоцінки.
8. Ціни активовані на сайті.

3.3. Розробка специфікації

Обмін даними - це процес взяття даних, структурованих за вихідною схемою, і перетворення їх у цільову схему, так що цільові дані є точним поданням вихідних даних. Обмін даними дозволяє обмінюватися даними між різними комп’ютерними програмами.

Це схоже на пов'язану з цим концепцію інтеграції даних, за винятком того, що дані фактично реструктуруються (втрата вмісту) при обміні даними. Можливо, не існує способу перетворити екземпляр з урахуванням усіх обмежень. І навпаки, може існувати безліч способів перетворення екземпляру (можливо, нескінченно багато), і в цьому випадку має бути визначений та обґрунтований "найкращий" вибір рішень.

В рамках інтеграції підприємств специфікація обміну даними - це архітектурний артефакт, який розвивається разом із бізнесом. Розробка та підтримка узгодженої семантичної моделі обміну даними є важливим, але не тривіальним завданням. Узгоджена семантична модель специфікацій обміну даними підтримує повторне використання, сприяє сумісності та, як наслідок, зменшує витрати на інтеграцію. Компоненти специфікацій обміну даними повинні бути послідовними та дійсними з точки зору узгоджених стандартів та керівних принципів.

У роботі описано модель обміну даними, а саме акту переоцінки, яка є узгодженою та підтримує масштабовану інтеграцію підприємств на основі стандартів.

3.3.1. Інформація про інтерфейс

Інтерфейс обміну даними включає в собі основні положення про призначення та роботу інтерфейсу. Інтерфейс обміну файлу з цінами представлен в таблиці 2.1.

Таблиця 2.1 - інформація про інтерфейс

Найменування і призначення інтерфейсу	Акт переоцінки - документ, який формується в ERP, коли необхідно провести переоцінку товару.
Частота обміну даними через інтерфейс	В реальному часі
Система-джерело	ERP
Система (ми)-приймач (і)	POS система, сайт

Опис інтерфейсу є обов’язковим в заповненні специфікації обміну даними.

3.3.2 Об’єкт

Нижче наведено опис об'єкта передачі переоцінки, вказано його дії, номер вибірки з черги, а також ключові поля.

Таблиця 2.2 - Опис об’єкту

Назва об’єкту	oERP_PRICE_RETAIL
Дія	(x) Create () Update () Del
Пріоритет вибірки з черги	() Высокий (x) Средний () Низкий
Структура ключа події	BPURAB;BPURCD

3.3.3. Структура вихідних даних (Response)

Маппінг розробляється для визначення відповідності даних між послідовними елементами.

Таблиця 2.3 - Маппінг 1 для POS системи заголовок

Система – джерело	**Система – приймач**	**Алгоритм перетворення**	**Назва поля** **Приклад заповнення**	**Тип поля**
	POS_ID	Послідовно	id запису “01”	Numeric
HEADER-REV_DAT	DOC_DATE		Дата створення документу переоцінки “04.03.2021”	Datetime
HEADER-NUM_DAT	DOC_NUM		Номер документу переоцінки “123”	Numeric
	STORE		Номеру магазину “store1”	String
	TYPES_ID	"type1” - відправка на сайт “type2” - відправка у POS систему	Тип переоцінки	String
ITEMS_DATE_BEG	start_date	Дата береться з першого рядка масиву	Дата активації документу переоцінки “04.03.2021”	Datetime

ITEMS_DATE_END	over_date	Дата береться з першого рядка масиву	Дата деактивації документу переоцінки “04.03.2021”	Datetime
	POS_ID	POS. POS_ID		Numeric
ITEMS_COST_NEW	COST_NEW		Нова ціна “799”	Numeric
ITEMS_OUTPUT	OUTPUT			String
ITEMS_ FULCOST	COMMENT	заповнення в разі помилки	Коментар “Error”	String
Алгоритм	TOVAR_ID	TOVAR_ID заповнюється процедурою Якщо товари не знайдені, процедура заповнює наступні поля: POS.comment = 'Size not found in table TOVAR' POS.status = 5		Numeric
Алгоритм	OLD_PRICE_GRN	Якщо не заповнено - нова ціна	Стара ціна “799”	Numeric

ITEMS_PRICE	PRICE_GRN	Цена в гривні	Ціна “899”	Numeric

У додатку Б розміщений ESQL код, який записує дані з переоцінку в буферну таблицю сайту.

Таблиця 2.5 - Маппінг 2 для сайту

Для подальшого запису даних в буферну таблицю сайту нижче наведено маппінг для системи сайт.

Система – источник	**Система – приемник**	**Алгоритм преобразования**	**Назва поля** **Приклад заповнення**	**Тип поля**
ITEMNUM	id_of_product		id товару “345”	Numeric
PRICE	promotional_cost	Если CONDITION= 'VKP0, promotional_cost = PRICE	Акційна ціна “799”	Numeric
	first_cost	Если CONDITION= 'VKP0, first_cost= 0	Перша ціна “1299”	Numeric

FULCOST	full_cost	Якщо FULCOST = 'YES': full_cost = 1. Якщо FULCOST = null: full_cost = 0. Якщо FULCOST відсутня: full_cost = null.	Признак повної ціни “1”	Boolean
	date_activation		Дата активації документу переоцінки “04.03.2021”	Datetime
REVNUM	revaluation_number		Номер документу переоцінки “123”	Numeric
	User	AGENT	Назва користувача “AGENT”	String

У додатку А розміщений ESQL код, який записує дані з переоцінку в буферну таблицю POS-системы

3.4. Створення MQ черги за помогою мови програмування ESQL

Необхідно розібрати перетворення повідомлень WebSphere MQ в шині інтеграції IBM в односторонній шаблон повідомлення.

У цьому випадку для підключення до WebSphere MQ на вузлах MQ використовуються прив’язки серверів. Тому WebSphere MQ слід встановлювати на тому ж комп'ютері, що і вузол інтеграції. Крім того, можна налаштувати вузли MQ для віддалених підключень.

Сценарій створення одностороннього шаблону обміну повідомленнями:

У існуючій топології обміну повідомленнями WebSphere MQ можна використовувати IBM Integration Bus для перетворення повідомлень WebSphere MQ і передачі його назад в іншу чергу. Підтвердження відповіді не надсилається; повідомлення обробляється і негайно передається до вказаної черги.

У цьому сценарії перевірка вмісту повідомлень або обробка помилок не налаштована для вузлів потоку повідомлень, оскільки відповідні дії залежать від загальної архітектури системи обміну повідомленнями.

Нижче описана процедура створення потоку повідомлень, щоб повідомлення з WebSphere MQ можна було перетворити, а потім перенести в іншу чергу.

1. Створюємо потік повідомлень з назвою MQ_PRICES_SITE_B_PRICE з такими вузлами:
   - MQInput вузол;
   - обчислювальний вузол;
   - MQOutput вузол.
2. Підключаємо вихідний термінал вузла MQInput до вхідного терміналу вузла Compute.
3. Підключаємо вихідний термінал вузла Compute до терміналу In вузла MQOutput.

Рисунок 3.2 - Зображення потоку повідомлень та вузлів, їх підключення.

Тепер потрібно налаштувати вузли потоку повідомлень.

4. Встановлюємо такі властивості вузла MQInput:

- на вкладці “Основні” встановлює для властивість Ім'я черги значення MQ.TASK1.PRICES;
- на вкладці “MQ Connection” встановлюю для властивості Connection значення Локальний менеджер черг, а потім для властивості “Менеджер черг” призначення TASK1_QUEUE_MANAGER;
- на вкладці “Розбір вхідного повідомлення” встановлюємо для властивості домену повідомлення значення XMLNSC.

5. Встановлюємо такі властивості вузла Compute:

Модуль ESQL MQ_PRICES_SITE_B_PRICE_Compute автоматично створюється. У поданні “Розробка додатків” відкриємо ESQL, а потім MQ_PRICES_SITE_B_PRICE_Compute модуль.

Вставляємо наступний ESQL в модуль:

```
CREATE COMPUTE MODULE MQ_Task1_One_Way_Compute
      CREATE FUNCTION Main() RETURNS BOOLEAN
      BEGIN
            SET OutputRoots = InputRoot;
            SET OutputRoots.XML.Price.TimeStamp = CURREMESTAMP;
            RETURN TRUE;
      END;
END MODULE;
```

Обчислювальний вузол являє перетворення рішення в цьому сценарії.

Вузол "Обчислення" додає до повідомлення значення часової позначки, яке показує час, коли воно було оброблено.

6. Встановлюємо такі властивості вузла MQOutput:

- на вкладці “Основні” встановлюємо для властивості Ім'я черги значення MQ.TASK1.OUT1;
- на вкладці MQ Connection встановлюємо для властивості Connection значення Локальний менеджер черг, а потім для властивості Менеджер черг призначення TASK1_QUEUE_MANAGER.

7. Зберігаємо потік повідомлень.

Тепер стає можливим перевірка потоку повідомлень. Створюємо повідомлення WebSphere MQ, надсилаємо повідомлення у чергу введення та спостерігаємо за змінами повідомлення у черзі виводу.

8. Створюємо повідомлення WebSphere MQ із корисним навантаженням, яке містить XML для черги введення у наступному форматі:

```
<price>
 <product>123</product>
 <cost>799</cost>
 <currency>grn</currency>
</price>
```

9. Запускаємо потік повідомлень.

Результати:

Під час запуску потоку повідомлень відбуваються такі дії:

- MQInput читає WebSphere MQ повідомлення з програми;
- обчислювальний вузол перетворює повідомлення;
- MQOutput вузол поміщає перетворене повідомлення на вказаний вихідний.

3.5. Приклад реалізації запропонованого алгоритму

Оновлення даних в інформаційних системах відбувається одразу, як тільки в черзі з'явиться файл переоцінок. Нижче розглянуло приклад реалізації обробки переоцінки на сайті.

≡	1 254 979	04.05.2021 00:00:00	14.05.2021 00:00:00	123	AGENT	345	799	1 299	1

Рисунок 3.3 - Запис даних переоцінки в буферну таблицю

В таблиці бачимо id запису, дату вставки запису в таблицю, дату активації ціни, бачимо, що дата записана майбутнім числом - це потрібно на період акцій, які повинні опускатися у певний проміжок часу, далі бачимо номер переоцінки “123”, ім’я користувача, який здійснив запис в таблицю, id товару, акційна ціна та перша ціна. Після того, як дата активації почне діяти ціна з переоцінки починає

відображатися на сайті. Щоб розуміти принцип встановлення ціни в системі сайт було спроектовано блок схему. Блок-схема це представлення процесу (схематичне), системи або комп'ютерного алгоритму. Блок-схеми часто застосовуються в різних сферах діяльності, щоб вивчати, планувати, удосконалювати, документувати і пояснювати складні процеси за допомогою простих логічних діаграм. Нижче наведено блок схему з принципом роботи модуля по встановленню ціни на сайті встановлення ціни на сайті.

Рисунок 3.4 - Блок-схема “Оновлення ціни на сайті”

Блок схема має початок та кінець та 3 процеси, які йдуть послідовно один за одним:

1. Перевірка дати активації акту переоцінки.
2. Встановлення нової ціни на товар в картці товару.
3. Запуск регламентної події по перерахунку ціни на сайті для коректного відображення ціни в каталозі.

Оскільки сайт інтернет магазину зазвичай має велике навантаження на сторінки каталогу, то їх необхідно кешувати, саме для того була розроблена регламентна подія, яка б оновлювала ціни в каталозі. Нижче наведено скріншот з сайту, де встановлена ціна на товар з буферної таблиці:

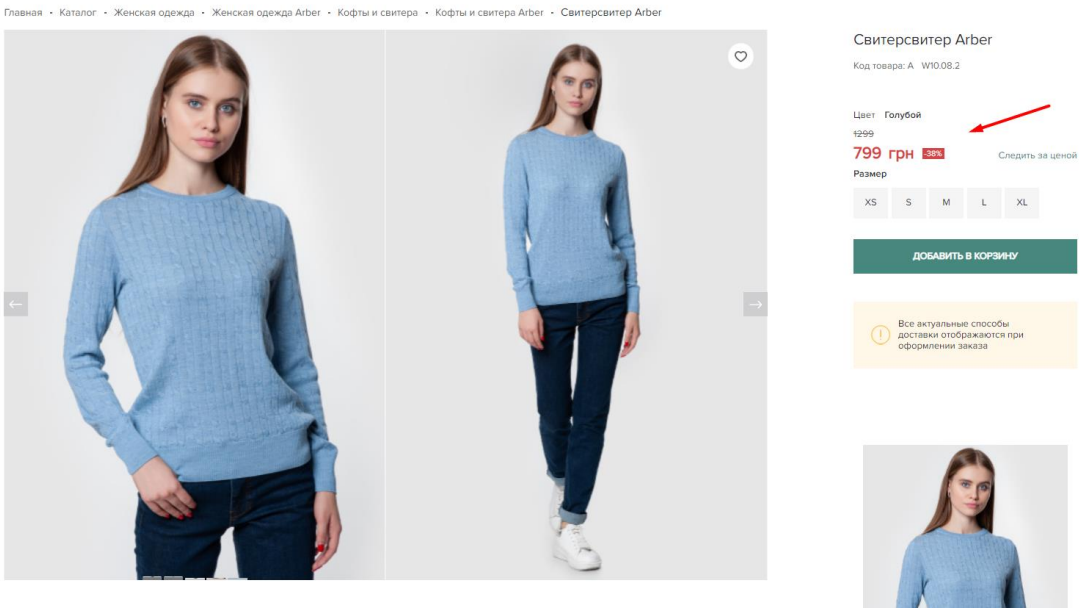

Рисунок 3.5 - Відображення ціни на сайті

Швидке оновлення даних є важливою річчю в роботі інтернет магазину, саме тому використання запропонованого алгоритму є гарним рішенням, яке дозволяє якісно маніпулювати цінам, що дозволить інтернет магазину отримувати ще більший прибуток.

ВИСНОВКИ

В дипломній роботі в повному обсязі розглянуті основні функціональні можливості IBM Integration Bus, що використовується для покращення автоматизації інтеграційних процесів між системами. Завдяки сучасним технологіям ціноутворення та оновлення даних в режимі реального часу робота спеціаліста стає більш легкою. IBM Message broker допомагає полегшити процес інтеграції між системами, робить його більш зручним для роботи з великими об’ємами даних, автоматизації процесів. Під час виконання дипломної роботи, була розроблена специфікація обміну даними, завдяки якій можно було розробити маппінг даних для інформаційних систем.

Після переходу на такий інструмент як IBM шина, було реалізовано максимальну швидку, зручну та цілісну передачу даних із майстер-системи. Функціональні можливості IBM Message broker дозволили максимально зручно автоматизувати процес обробки, передачі та інтеграції даних між досліджуваними POS-системами, що значно вплинуло на показники швидкості та цілісності передачі даних між різними системами. Реалізація архітектури ESQL допомогла організувати зв'язок між системами, зробити процес передачі автоматизованим та усунути помилки, що могли завадити передачі цілісних даних.

Для подальшої розробки ефективного плану дій з автоматизації процесу ціноутворення, було опрацьовано необхідні теоретичні матеріали, обрано відповідні інструментальні засоби, викладено теоретичні основи та суть досліджуваної проблеми, визначені основні завдання функціональних можливостей IBM Message Broker.

На основі проведеного дослідження опубліковано тези доповіді в збірниках “ЗАСТОСУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ В ІНФОКОМУНІКАЦІНИХ ТЕХНОЛОГІЯХ”

ПЕРЕЛІК ПОСИЛАНЬ

1. Discovering Statistics Using IBM SPSS Statistics — SAGE Publications Ltd; Fifth edition, 2018.
2. SPSS Survival Manual: A Step by Step Guide to Data Analysis Using IBM — Open University Press; 7th ed. edition, 2020.
3. Websphere Business Integration Message Broker Basics — IBM Publishing, 2004.
4. IBM Integration Bus A Complete Guide — The Art of Service; IBM Integration Bus Publishing, 2020.
5. Message Broker A Clear and Concise Reference — 5STARCooks; 2018.
6. ERP Systems for Manufacturing Supply Chains: Applications, Configuration, and Performance — Auerbach Publications; 1st edition, 2020.
7. Martin B., How to implement a manufacturing system: Best practices and pitfalls when implementing an MRP/ERP system – Macmillan Publishers, 2019.
8. Deborah S., Senior ERP Consultant Guru Notebook: Customized 100 Page Lined Journal Gift For A Busy Senior ERP Consultant — Adchops, 2021.
9. IBM Power Systems: Сервер віртуального вводу-виводу — IBM Corporation Publishing, 2009.
10. Progi, Розділ про створення MQ черг, [Електронний ресурс]. — Режим доступу: https://progi.pro/messagebroker-t9672
11. Appian, Business Process Definition, [Електронний ресурс]. — Режим доступу: https://appian.com/bpm/business-process-definition.html
12. Migration to Websphere Business Integration Message Broker V5 — IBM Corporation Publishing, 2004.
13. Bhushan B., Sook C., Carolyn E., High Availability in WebSphere Messaging Solutions — 2010.
14. Kareem Y., Enterprise Messaging Using JMS and IBM WebSphere — Hachette Book Group, 2004.

15. Боббі Вульф, Грегор Х., Шаблони інтеграції корпоративних додатків — BookPrix, 2019.
16. Jennifer F., Kentaroh K., IBM MB Managed File Transfer Integration with WebSphere — IBM Redbooks, 2012.

ДОДАТОК А

Програмний код для запису переоцінки в буферну таблицю сайту

```
SCHEMA erp.prices.retail

Подключение к сайту
CREATE COMPUTE MODULE SUB_PRICES_SITE_B_PRICE
CREATE FUNCTION Main() RETURNS BOOLEAN
BEGIN
-- CALL CopyMessageHeaders();
-- CALL CopyEntireMessage();
-- VKP0 тогда вставляем в action
--max вставляем в base


CASE WHEN
InputRoot.XMLNSC.NS:ErpCondA04CondA04.ErpCondA04CondA04IDocBO.ErpIDocControlRecord.IDOCT
YP = 'COND_A04' THEN
DECLARE price DECIMAL
CAST(InputRoot.XMLNSC.NS:ErpCondA04CondA04.ErpCondA04CondA04IDocBO.ErpCondA04CondA04
DataRecord.ErpCondA04CondA04E2komg005.ErpCondA04CondA04E2konh.ErpCondA04CondA04E2konp
003.KBETR AS DECIMAL);
DECLARE price_grn DECIMAL
CAST(InputRoot.XMLNSC.NS:ErpCondA04CondA04.ErpCondA04CondA04IDocBO.ErpCondA04CondA04
DataRecord.ErpCondA04CondA04E2komg005.ErpCondA04CondA04E2konh.ErpCondA04CondA04E2konp
003.ErpCondA04CondA04Ze1konp000.PRICE AS DECIMAL);
DECLARE revaluationNumber CHAR
InputRoot.XMLNSC.NS:ErpCondA04CondA04.ErpCondA04CondA04IDocBO.ErpCondA04CondA04DataRe
cord.ErpCondA04CondA04E2komg005.ErpCondA04CondA04E2konh.REVNUM;
SET Environment.HEADER.EVENT_ID = InputRoot.XMLNSC.NS:ErpCondA04CondA04.ERPTransactionID;
DECLARE name_of_user CHAR 'AGENT';
DECLARE promotional_cost INTEGER;
DECLARE first_cost INTEGER;

CASE WHEN
InputRoot.XMLNSC.NS:ErpCondA04CondA04.ErpCondA04CondA04IDocBO.ErpCondA04CondA04DataRe
cord.ErpCondA04CondA04E2komg005.CONDITION= 'ZMAX' THEN
SET promotional_cost = 0;
SET first_cost = price_grn;
ELSE
SET promotional_cost = price_grn ;
SET first_cost = 0;
END CASE;

INSERT INTO Database.db_site.table_cost (id_of_product,
promotional_cost,first_cost,full_cost,date_activation,revaluation_number,user)
```

```
VALUES (
InputRoot.XMLNSC.NS:ErpCondA04CondA04.ErpCondA04CondA04IDocBO.ErpCondA04CondA04DataRecord.ErpCondA04CondA04E2komg005.ITEMNUM,
promotional_cost,
first_cost,
null,
cast(CURRENT_DATE as date format 'yyyy-MM-dd'),
revaluationNumber,
name_of_user);


ELSE
SET Environment.HEADER.EVENT_ID = InputRoot.XMLNSC.NS12:ErpZsd1900.ERPTransactionID;
-- SET OutputRoot.XMLNSC.REQUEST.BODY.OBJECT_KEY =
InputRoot.XMLNSC.NS12:ErpZsd1900.ErpZsd1900IDocBO.ErpZsd1900DataRecord.ErpZsd1900Zsd19Header000.ErpZsd1900Zsd19Items000.BISMT ;
-- SET OutputRoot.XMLNSC.REQUEST.HEADER.EVENT_ID = Environment.HEADER.EVENT_ID;
declare full_cost CHARACTER '0' ;
declare ref REFERENCE TO
InputRoot.XMLNSC.NS12:ErpZsd1900.ErpZsd1900IDocBO.ErpZsd1900DataRecord.ErpZsd1900Zsd19Header000.ErpZsd1900Zsd19Items000;

WHILE LASTMOVE(ref) DO
CASE WHEN ref.FULCOST ='YES' THEN
SET full_cost = '1';
END CASE;
INSERT INTO Database.db_site.table_cost(id_of_product,
promotional_cost,first_cost,full_cost,date_activation,revaluation_number,user) VALUES
(rtrim(ref.BISMT),
ref.ZZ_PRICE,0,full_cost,cast(cast(InputRoot.XMLNSC.NS12:ErpZsd1900.ErpZsd1900IDocBO.ErpZsd1900DataRecord.ErpZsd1900Zsd19Header000.ErpZsd1900Zsd19Items000.VKKAB AS DATE FORMAT
'yyyyMMdd') as date format 'yyyy-MM-
dd'),InputRoot.XMLNSC.NS12:ErpZsd1900.ErpZsd1900IDocBO.ErpZsd1900DataRecord.ErpZsd1900Zsd19Header000.NUM_DAT,'AGENT');
MOVE ref NEXTSIBLING REPEAT TYPE NAME;
END WHILE;
END CASE;
SET OutputLocalEnvironment.Destination.RouterList.DestinationData[1].labelName =
'PRICES_RESPONSE';
RETURN TRUE;
END;
CREATE PROCEDURE CopyMessageHeaders() BEGIN
DECLARE I INTEGER 1;
DECLARE J INTEGER;
SET J = CARDINALITY(InputRoot.*[]);
WHILE I < J DO
SET OutputRoot.*[I] = InputRoot.*[I];
SET I = I + 1;
END WHILE;
```

```
END;
CREATE PROCEDURE CopyEntireMessage() BEGIN
SET OutputRoot = InputRoot;
END;
END MODULE;
```

ДОДАТОК Б

Програмний код для запису переоцінки в буферну таблицю програмного забезпечення POS системи

```
SCHEMA erp.prices.retail

DECLARE NS NAMESPACE ;
CREATE COMPUTE MODULE SUB_PRICES_POS_POS
CREATE FUNCTION Main() RETURNS BOOLEAN
BEGIN
IF InputRoot.XMLNSC.NS12:ErpZsd1900.ErpZsd1900IDocBO.ErpIDocControlRecord.IDOCTYP =
'ZSD1900' THEN
SET Environment.HEADER.EVENT_ID = InputRoot.XMLNSC.NS12:ErpZsd1900.ErpTransactionID;
SET Environment.POS.INSERT = PASSTHRU('INSERT INTO POS (
DOC_DATE,
DOC_NUM,
STORE,
TYPES_ID,
start_date,
over_date) VALUES (?,?,?,?,?,?)' VALUES(
CAST(CAST(InputRoot.XMLNSC.NS12:ErpZsd1900.ErpZsd1900IDocBO.ErpZsd1900DataRecord.ErpZsd1
900Zsd19Header000.ERDAT AS DATE FORMAT 'yyyyMMdd') AS DATE FORMAT 'yyyy-MM-dd'),
CAST(InputRoot.XMLNSC.NS12:ErpZsd1900.ErpZsd1900IDocBO.ErpZsd1900DataRecord.ErpZsd1900Zsd
19Header000.KBELN AS INTEGER),
'store1',
'A1',
CAST(CAST(InputRoot.XMLNSC.NS12:ErpZsd1900.ErpZsd1900IDocBO.ErpZsd1900DataRecord.ErpZsd1
900Zsd19Header000.ErpZsd1900Zsd19Items000.VKKAB AS DATE FORMAT 'yyyyMMdd') AS DATE
FORMAT 'yyyy-MM-dd'),
CAST(CAST(InputRoot.XMLNSC.NS12:ErpZsd1900.ErpZsd1900IDocBO.ErpZsd1900DataRecord.ErpZsd1
900Zsd19Header000.ErpZsd1900Zsd19Items000.VKKBI AS DATE FORMAT 'yyyyMMdd') AS DATE
FORMAT 'yyyy-MM-dd')
));
SET Environment.SEQ = PASSTHRU('SELECT IDENT_CURRENT("POS") as POS_ID');
DECLARE FULCOST CHARACTER 'N';
DECLARE zsditems REFERENCE TO
InputRoot.XMLNSC.NS12:ErpZsd1900.ErpZsd1900IDocBO.ErpZsd1900DataRecord.ErpZsd1900Zsd19Hea
der000.ErpZsd1900Zsd19Items000;

WHILE LASTMOVE(zsditems) DO
IF zsditems.FULCOST = 'YES'
THEN SET FULCOST = 'F';
ELSE
SET FULCOST = 'N';
END IF ;
SET Environment.POS_GDS.INSERT = PASSTHRU('INSERT INTO POS_GDS (
POS_ID,
COST_NEW,
OUTPUT,
COMMENT
) VALUES (?,?,?,?)'
```

```
VALUES(
CAST(CAST(Environment.SEQ.POS_ID AS INTEGER) AS CHARACTER),
zsditems.ENDPR,
CAST(zsditems.BISMT AS CHARACTER),
FULCOST
));
MOVE zsditems NEXTSIBLING REPEAT TYPE NAME ;
END WHILE;
COMMIT;
CALL SP_POSCREATE(
CAST(CAST(Environment.SEQ.POS_ID AS INTEGER) AS CHARACTER),
Environment.ERROR_CODE,
Environment.ERROR_DESCRIPTION
);
IF Environment.ERROR_CODE = 0 THEN
SET Environment.ERROR_CODE = 1;
END IF;
SET Environment.PRICE_RETAIL = NULL;
SET OutputLocalEnvironment.Destination.RouterList.DestinationData[1].labelName =
'PRICES_RESPONSE';
RETURN TRUE;
ELSEIF
InputRoot.XMLNSC.NS:ErpCondA04CondA04.ErpCondA04CondA04IDocBO.ErpIDocControlRecord.IDOCT
YP = 'COND_A04' THEN
SET Environment.HEADER.EVENT_ID = InputRoot.XMLNSC.NS:ErpCondA04CondA04.ErpTransactionID;
DECLARE TYPES_ID CHARACTER;
CASE WHEN
InputRoot.XMLNSC.NS:ErpCondA04CondA04.ErpCondA04CondA04IDocBO.ErpCondA04CondA04DataRe
cord.ErpCondA04CondA04E2komg005.ErpCondA04CondA04E2konh.ErpCondA04CondA04E2konp003.CO
NDITION ='VKP0' THEN
SET Environment.DOC_NUM = cast(CAST(CURRENT_DATE AS CHARACTER FORMAT 'yyyyMMdd') as
INTEGER);
SET TYPES_ID = 'A0';
WHEN
InputRoot.XMLNSC.NS:ErpCondA04CondA04.ErpCondA04CondA04IDocBO.ErpCondA04CondA04DataRe
cord.ErpCondA04CondA04E2komg005.ErpCondA04CondA04E2konh.ErpCondA04CondA04E2konp003.CO
NDITION ='ZMAX' THEN
SET Environment.DOC_NUM =
CAST(InputRoot.XMLNSC.NS:ErpCondA04CondA04.ErpCondA04CondA04IDocBO.ErpCondA04CondA04
DataRecord.ErpCondA04CondA04E2komg005.ErpCondA04CondA04E2konh.REVNUM AS INTEGER);
SET TYPES_ID = 'A2';
ELSE
SET Environment.ERROR_CODE = '-1' ;
SET Environment.ERROR_DESCRIPTION = 'CONDITION NOT IN VKP0, ZMAX';
PROPAGATE ;
RETURN FALSE;
END CASE;
SET Environment.POS.INSERT = PASSTHRU('INSERT INTO POS (
DOC_DATE,
DOC_NUM,
STORE,
TYPES_ID,
start_date,
over_date) VALUES (?,?,?,?,?,?)' VALUES(
CAST(CURRENT_DATE AS DATE FORMAT 'yyyy-MM-dd'),
Environment.DOC_NUM,
```

```
'store1',
TYPES_ID,
CAST(CAST(InputRoot.XMLNSC.NS:ErpCondA04CondA04.ErpCondA04CondA04IDocBO.ErpCondA04Co
ndA04DataRecord.ErpCondA04CondA04E2komg005.ErpCondA04CondA04E2konh.DATAB AS DATE
FORMAT 'yyyyMMdd') AS DATE FORMAT 'yyyy-MM-dd'),
CAST(CAST(InputRoot.XMLNSC.NS:ErpCondA04CondA04.ErpCondA04CondA04IDocBO.ErpCondA04Co
ndA04DataRecord.ErpCondA04CondA04E2komg005.ErpCondA04CondA04E2konh.DATBI AS DATE
FORMAT 'yyyyMMdd') AS DATE FORMAT 'yyyy-MM-dd')
));
SET Environment.SEQ = PASSTHRU('SELECT IDENT_CURRENT("POS") as POS_ID');
CASE WHEN TYPES_ID ='A0'
THEN
SET Environment.POS_GDS.INSERT = PASSTHRU('INSERT INTO POS_GDS (
POS_ID,
COST_NEW,
OUTPUT
) VALUES (?,?,?)'
VALUES(
CAST(CAST(Environment.SEQ.POS_ID AS INTEGER) AS CHARACTER),
InputRoot.XMLNSC.NS:ErpCondA04CondA04.ErpCondA04CondA04IDocBO.ErpCondA04CondA04DataRe
cord.ErpCondA04CondA04E2komg005.ErpCondA04CondA04E2konh.ErpCondA04CondA04E2konp003.KB
ETR,
InputRoot.XMLNSC.NS:ErpCondA04CondA04.ErpCondA04CondA04IDocBO.ErpCondA04CondA04DataRe
cord.ErpCondA04CondA04E2komg005.ITEMNUM
));
ELSE

SET Environment.POS_MODEL.INSERT = PASSTHRU('INSERT INTO POS_MODEL (
POS_ID,
MODEL_CODE,
PRICE_MAX
) VALUES (?,?,?)'
VALUES(
CAST(CAST(Environment.SEQ.POS_ID AS INTEGER) AS CHARACTER),
InputRoot.XMLNSC.NS:ErpCondA04CondA04.ErpCondA04CondA04IDocBO.ErpCondA04CondA04DataRe
cord.ErpCondA04CondA04E2komg005.ITEMNUM,
InputRoot.XMLNSC.NS:ErpCondA04CondA04.ErpCondA04CondA04IDocBO.ErpCondA04CondA04DataRe
cord.ErpCondA04CondA04E2komg005.ErpCondA04CondA04E2konh.ErpCondA04CondA04E2konp003.KB
ETR
));
END CASE;
COMMIT;
CALL SP_POSCREATE(
CAST(CAST(Environment.SEQ.POS_ID AS INTEGER) AS CHARACTER),
Environment.ERROR_CODE,
Environment.ERROR_DESCRIPTION
);
IF Environment.ERROR_CODE = 0 THEN
SET Environment.ERROR_CODE = 1;
END IF;
SET Environment.PRICE_RETAIL = NULL;
SET OutputLocalEnvironment.Destination.RouterList.DestinationData[1].labelName =
'PRICES_RESPONSE';
END IF;
RETURN TRUE;
END;
```

```
CREATE PROCEDURE SP_POSCREATE(
IN POS_ID CHARACTER,
INOUT ERROR_STATUS CHARACTER,
INOUT ERROR_TXT CHARACTER)
LANGUAGE DATABASE
EXTERNAL NAME "dbo.spPOSCreate";

CREATE PROCEDURE CopyMessageHeaders() BEGIN
DECLARE I INTEGER 1;
DECLARE J INTEGER;
SET J = CARDINALITY(InputRoot.*[]);
WHILE I < J DO
SET OutputRoot.*[I] = InputRoot.*[I];
SET I = I + 1;
END WHILE;
END;
CREATE PROCEDURE CopyEntireMessage() BEGIN
SET OutputRoot = InputRoot;
END;
END MODULE;
```

ДОДАТОК В

Демонстраційний матеріал

титульный, актуальность, мета объект пердмет, что использовалось, блок схемы бп, скрины реализации и табличек, выводы - предложение с тезами, дата, как нызвается, дякую за увагу 11-12