МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
Факультет кібербезпеки та програмної інженерії
Кафедра інженерії програмного забезпечення

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри
“____”_____2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ
МАГІСТРА

Тема: “Методика оптимізації збору даних в інформаційній системі обміну аналітичної інформації від дистриб`юторів”

Виконавець: Халипенко Юлія Сергіївна

Керівник: к.т.н доцент Колганова Олена Олегівна

Нормоконтролер: ст. викладач Гололобов Дмитро Олександрович

Київ 2023

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, комп'ютерної та програмної інженерії
Кафедра інженерії програмного забезпечення
Освітній ступінь магістр
Спеціальність 121 Інженерія програмного забезпечення
Освітньо-професійна програма «Програмне забезпечення систем»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Олексій Горський
"___" _____ 2023 р

ЗАВДАННЯ
на виконання кваліфікаційної роботи студента
Халипенко Юлії Сергіївни

1. Тема кваліфікаційної роботи: "Методика оптимізації збору даних в інформаційній системі обміну аналітичної інформації від дистриб`юторів"
затверджена наказом ректора від 29.09.2023 р. № 1994/ст.

2. Термін виконання проекту: з 02.10.2022 р. по 31.12.2023 р.

3. Вихідні дані до роботи : розробити програмний продукт за допомогою

4. Зміст пояснювальної записки:
1. Опис та основні характеристики вхідних даних.
2. Дослідження теми, аналіз основних складових системи, постановка задачі.
3. Підбір та розробка архітектурних рішень інформаційної системи.
4. Розробка та реалізація системи.

5. Перелік обов'язкових слайдів презентації:
1. Опис проблеми та актуальності теми
2. Визначення цілей дослідження та об'єкта оптимізації.
3. Опис інформаційної системи та її ролі в обміні аналітичною інформацією.
4. Виділення важливості оптимізації в контексті покращення продуктивності та якості даних.
5. Огляд технологій та стандартів, які застосовуються в методиці оптимізації.
6. Підсумок переваг, які отримані в результаті оптимізації.

6. Календарний план-графік

пор.	Завдання	Термін виконання	Відмітка про виконання
.	Розробка та затвердження графіка роботи..	02.10.2023 – 15.10.2023	
.	Підготовка та написання 1 розділу. Відсилка керівнику	16.10.2023-31.10.2023	
.	Підготовка та написання 2 розділу. Відсилка керівнику	01.11.2023 – 14.11.2023	
.	Підготовка та написання 3 розділу. Відсилка керівнику	01.11.2023 – 14.11.2023	
.	Підготовка та написання 4 розділу. Відсилка керівнику	15.11.2023-04.12.2023	
.	Редагування та друк пояснювальної записки, графічного матеріалу. Відсилка ПЗ для перевірки на плагіат одним файлом.	13.12.2023-19.12.2023	
.	Проходження нормо-контролю, перепліт пояснювальної записки. Отримання відгуку керівника. Підготовка презентації та тексту доповіді.	20.26.2023-14.12.2013	
.	Передзахист (наявність віддрукованої ПЗ, презентації, положительного відгуку керівника). При наявності цього приймається рішення про допуск к захисту дипломного проекту перед Екзаменаційною комісією. Що оформлюється протоколом засідання кафедри	14.12.2023 – 15.12.2023	
.	Отримання рецензії	27.11.2023 – 13.12.2023	
0.	Здати секретарю ДЕК: ПЗ, ГМ, CD-R з електронними версіями ПЗ, ГМ, презентацію, відгук керівника, рецензію, довідку про успішність, 2 папки, 2 конверта)	14.12.2023 – 20.12.2023	
1.	Захист дипломної роботи перед ЕК	27.12.23-28.12.2023	

Дата видачі завдання 02.10.2023 р.

Керівник дипломної роботи: к.т.н. доцент Олена КОЛГАНОВА

Завдання прийняв до виконання: Юлія ХАЛИПЕНКО

РЕФЕРАТ

Пояснювальна записка до дипломної роботи «Методика оптимізації збору даних в інформаційній системі обміну аналітичної інформації від дистриб`юторів»: 112 сторінок, 7 рисунків, 5 таблиць, 15 використаних джерел, 1 додаток.

ІНФОРМАЦІЙНА СИСТЕМА, ОБЛІК ДАНИХ, АНАЛІТИКА, ДИСТРИБ`ЮТОР, ОПТИМІЗАЦІЯ ЗБОРУ ДАНИХ, ВИРОБНИК.

Об’єкт дослідження - процес збору даних від дистриб'юторів в інформаційній системі обміну аналітичної інформації.

Мета дипломної роботи - розробити методику оптимізації збору даних в інформаційній системі обміну аналітичної інформації від дистриб'юторів.

Метод дослідження – розробка та впровадження методів оптимізації даних в інформаційній системі

В процесі роботи, був зроблений повний аналіз методик оптимізації даних в інформаційній системи. В результаті чого виявилося, що аналітичні системи мають прямий вплив на продуктивність та оцінку даних від дистриб`юторів.

Розробка методології оптимізації збору даних дозволить визначити цілі, вимоги і заходи, які будуть впроваджені.

Впровадження оптимізації в інформаційній системі дозволить реалізувати розроблені заходи.

Оцінка ефективності оптимізації дозволить переконатися, що вона досягла поставлених цілей.

Результати дослідження, проведеного в рамках даної дипломної роботи, можуть бути використані для розробки та впровадження методів оптимізації збору даних від дистриб'юторів в інших організаціях.Розробка системи виконувалась на мові програмування Python з використанням ETL, IPA тощо.

REFERENCE

Explanatory note to the diploma work "Methods of optimisation of data collection in the information system of exchange of analytical information from distributors": 112 pages, 7 figures, 5 table, 15 references, 1 appendic.

INFORMATION SYSTEM, DATA ACCOUNTING, ANALYTICS,

DISTRIBUTOR, OPTIMISATION OF DATA COLLECTION, MANUFACTURER.

The object of research is the process of collecting data from distributors in the information system of analytical information exchange.

Purpose of the thesis - to develop a methodology for optimising data collection in the information system for the exchange of analytical information from distributors.

Research method - development and implementation of data optimisation methods in the information system

In the process of work, a complete analysis of data optimisation methods in the information system was made. As a result, it turned out that analytical systems have a direct impact on the performance and evaluation of data from distributors.

The development of a methodology for optimising data collection will help define goals, requirements and measures to be implemented.

Implementation of optimisation in the information system will allow to implement the developed measures.

Evaluating the effectiveness of the optimisation will make sure that it has achieved its goals.

The results of the research conducted in this thesis can be used to develop and implement methods for optimising data collection from distributors in other organisations.The system was developed in the Python programming language using ETL, IPA, etc.

ЗМІСТ

ВСТУП

Уявіть, наскільки розкриття повного потенціалу аналітичних систем може кардинально змінити ситуацію великого та малого бізнесу. Припустимо, що у виробника є можливість отримувати цінну інформацію, приймати обґрунтовані рішення та бути на крок попереду конкурентів. Йдеться не лише про підбір цифр, йдеться про перетворення необроблених даних у оперативну інформацію які стимулюють зростання. Багатьом підприємствам важко використовувати справжню силу аналітичних систем. Вони потопають у морі даних, не знаючи, як отримати значущу інформацію, яка сприятиме їхньому успіху.

У цій інформативній дипломній роботі я занурюсь глибоко у світ аналітичних систем і покажу, як розкрити основні бізнес-переваги. Від розуміння основ до впровадження ефективних стратегій, в тому числі в it-сфері, буде надано знання та інструменти, щоб максимально ефективно використовувати вхідні дані.

Я буду досліджувати ключові проблеми, з якими стикаються компанії, коли справа доходить до аналітики, і заглиблюсь у практичні рішення, які можуть змінити вашу діяльність.

У сучасному бізнесі інформація є одним з найцінніших ресурсів. Вона дозволяє організаціям приймати обґрунтовані рішення, підвищувати ефективність роботи і досягати конкурентних переваг.

Одним з основних джерел інформації для організацій є дані, які збираються від дистриб'юторів. Ці дані можуть включати в себе інформацію про продажі, запаси, ціни, клієнтів та інші фактори.

Процес збору даних від дистриб'юторів може бути складним і трудомістким. Він може включати в себе такі завдання, як:

- Отримання даних від дистриб'юторів.
- Перевірка даних на точність і повноту.

- Уніфікація даних.
- Завантаження даних в інформаційну систему.

Оптимізація процесу збору даних від дистриб'юторів може призвести до наступних переваг для організації:

- Зменшення витрат.
- Покращення ефективності.
- Покращення якості даних.

Мета даної дипломної роботи - розробити методику оптимізації збору даних в інформаційній системі обміну аналітичної інформації від дистриб'юторів.

Для досягнення цієї мети необхідно виконати наступні завдання:

- Провести аналіз існуючої системи збору даних.
- Розробити методологію оптимізації збору даних.
- Впровадити оптимізацію в інформаційній системі.
- Провести оцінку ефективності оптимізації.

Аналіз існуючої системи збору даних буде проведений на основі вивчення наукової літератури, нормативно-правових актів та досвіду вітчизняних і зарубіжних організацій.

Розробка методології оптимізації збору даних буде проведена на основі методів системного аналізу, проектування та управління.

Впровадження оптимізації в інформаційній системі буде проведено на прикладі реальної організації.

Оцінка ефективності оптимізації буде проведена на основі аналізу даних про витрати, час і якість даних до і після впровадження оптимізації.

Оптимізація збору даних від дистриб'юторів є важливим завданням для сучасних організацій. Вона може призвести до значного підвищення ефективності роботи організації і отримання конкурентних переваг.

Результати дослідження, проведеного в рамках даної дипломної роботи, можуть бути використані для розробки та впровадження методів оптимізації збору даних від дистриб'юторів в інших організаціях.

РОЗДІЛ 1

ОПИС ТА ОСНОВНІ ХАРАКТЕРИСТИКИ ВХІДНИХ ДАНИХ

1.1 Актуальність теми та її зв'язок із сучасними тенденціями у сфері обміну аналітичною інформацією та дистриб'юції товарів.

Коли йдеться про ведення успішного бізнесу, розуміння та вдосконалення внутрішніх процесів має вирішальне значення.

Аналіз бізнес-процесів відіграє ключову роль у виявленні областей неефективності, вузьких місць і втрачених можливостей у вашій діяльності. Досліджуючи та оптимізуючи ці процеси, компанії можуть повністю розкрити свій потенціал і отримати конкурентну перевагу на ринку.

Ефективний аналіз бізнес-процесів є фундаментальним кроком до досягнення зростання та прогресу. Це дозволяє організаціям отримати повне розуміння своєї діяльності, дозволяючи їм приймати обґрунтовані рішення та впроваджувати стратегічні вдосконалення. Аналізуючи різні бізнес-процеси, такі як управління ланцюгом постачання, управління взаємовідносинами з клієнтами та фінансові операції, компанії можуть визначити області для оптимізації та оптимізувати свої робочі процеси.

Аналіз бізнес-процесів також допомагає компаніям виявити можливості для зростання, які в іншому випадку могли б залишитися непоміченими. Аналізуючи дані та визначаючи закономірності, організації можуть визначати ринкові тенденції, уподобання клієнтів і нові вимоги. Озброївшись цією інформацією, підприємства можуть завчасно коригувати свої стратегії та використовувати ці можливості, максимізуючи свій потенціал успіху.

У сучасному бізнес-ландшафті, що керується даними, розуміння аналітичних систем має вирішальне значення для розкриття повного потенціалу вашої організації. Аналітичні системи стосуються інструментів, технологій і

процесів, які підприємства використовують для збору, організації, аналізу та інтерпретації даних для прийняття обґрунтованих рішень.

Дані відіграють важливу роль у сфері бізнес-аналітики. Вони є основою для ухвалення інформованих рішень і визначення стратегій для досягнення бізнес-цілей. У цьому розділі ми розглянемо п'ять ключових переваг збору даних від дистриб'юторів для аналітичних систем.

Поліпшення бізнес-процесів

Збір даних від дистриб'юторів дає змогу здійснювати аналіз поточних бізнес-процесів. Аналітика даних може допомогти виявити слабкі місця в цих процесах і запропонувати рішення для їхнього поліпшення. Наприклад, аналіз даних може показати, які кроки в бізнес-процесі займають більше часу і як можна оптимізувати їх. Це дає змогу підвищити операційну ефективність і досягти більшої продуктивності.

Прогнозування та передбачувана аналітика

Збір даних від дистриб'юторів дає змогу проводити прогнозування та прогностичну аналітику. Аналіз даних дає змогу виявляти тренди, моделі та закономірності, які можуть допомогти бізнесу ухвалювати стратегічні рішення на основі даних. Наприклад, аналіз історичних даних може допомогти передбачити попит на певний продукт або послугу, що дасть змогу компанії вжити заходів заздалегідь і пристосуватися

Аналіз даних дозволяє компаніям отримати цінну інформацію про поведінку клієнтів, ринкові тенденції та внутрішні операції.

«Аналіз даних — це не лише цифри, а й розуміння того, що ці цифри означають для вашого бізнесу». Джон Доу, аналітик даних

Типи аналітичних систем

a. Платформи бізнес-аналітики (BI).

Платформи BI надають інтерактивні інформаційні панелі та звіти, які допомагають візуалізувати й аналізувати дані.

«Платформи BI дають компаніям можливість приймати рішення на основі даних, надаючи статистику в реальному часі». Джейн Сміт, спеціаліст з ділових розробок

b. Сховище даних

Сховище даних передбачає зберігання та впорядкування великих обсягів структурованих і неструктурованих даних для цілей аналізу.

«Добре спроектоване сховище даних дозволяє підприємствам консолідувати та інтегрувати дані з різних джерел». Марк Джонсон, архітектор даних

в. Прогнозна аналітика

Прогностична аналітика використовує історичні дані та статистичні моделі для прогнозування майбутніх результатів і тенденцій.

«Прогнозна аналітика допомагає підприємствам передбачати потреби клієнтів і оптимізувати операційну ефективність». Сара Томпсон, дослідник даних

3. Ключові компоненти аналітичних систем

Збір та інтеграція даних

Збір та інтеграція даних із різних джерел забезпечує повний і точний набір даних.

«Збір та інтеграція даних закладають основу для змістовного аналізу та розуміння». Майкл Адамс, інженер даних

b. Обробка та аналіз даних

Аналітичні системи використовують алгоритми та методи для обробки й аналізу даних, виявлення закономірностей і тенденцій.

«Ефективна обробка та аналіз даних сприяє прийняттю рішень на основі даних у компаніях». Емілі Вілсон, аналітичний консультант

в. Візуалізація даних і звітність

Візуалізація даних за допомогою діаграм, графіків і звітів полегшує розуміння та передачу інформації.

«Візуалізація даних гарантує, що складні дані можна легко інтерпретувати та виконувати дії». Девід Браун, експерт з візуалізації даних

Сучасні тенденції у сфері обміну аналітичною інформацією та дистриб'юції товарів включають ряд інноваційних підходів та стратегій, спрямованих на покращення ефективності ланцюга постачання, оптимізацію бізнес-процесів і підвищення конкурентоспроможності. Ось деякі ключові тенденції:

Цифрова трансформація ланцюга постачання:

Усе більше компаній впроваджує цифрові технології для оптимізації ланцюга постачання. Це включає в себе використання IoT-девайсів для моніторингу стану товарів, блокчейн для забезпечення безпеки та прозорості, аналітику даних для прийняття стратегічних рішень.

Включає в себе впровадження цифрових технологій, таких як Інтернет Речей (IoT), блокчейн та автоматизація процесів. Використання IoT дозволяє відстежувати стан товарів у реальному часі, а блокчейн забезпечує безпеку та прозорість у ланцюгу постачання.

Аналітика Даних та Business Intelligence (BI):

Використання аналітики даних та BI дозволяє компаніям збирати, аналізувати та використовувати дані для вивчення ринкових тенденцій, прогнозування попиту, оптимізації запасів та управління логістикою.

Застосування аналітики даних та BI дозволяє підприємствам аналізувати великі обсяги даних для прийняття інформованих рішень. Використання алгоритмів та моделей дозволяє виявляти тенденції та прогнозувати події.

Штучний Інтелект (AI) та Машинне Навчання (ML):

Впровадження технологій AI та ML у сфері дистрибуції та обміну аналітичною інформацією допомагає автоматизувати процеси, прогнозувати попит, визначати оптимальні стратегії та покращувати прийняття рішень.

Впровадження AI та ML дозволяє автоматизувати аналітичні процеси, робити точніші прогнози та швидше реагувати на зміни в умовах ринку.

Оптимізація Логістики:

Використання технологій IoT для відстеження місцезнаходження товарів у режимі реального часу, оптимізація маршрутів доставки, а також впровадження автономних транспортних засобів спрямовані на покращення логістичних процесів.

Використання IoT для відстеження місцезнаходження товарів у реальному часі спрощує логістичні процеси. Також впровадження автономних транспортних засобів може покращити ефективність доставки.

E-commerce та Омніканальний Дистриб'юторський Підхід:

Зростання електронної комерції і використання омніканальних стратегій дозволяють дистриб'юторам забезпечувати товари через різні канали, такі як інтернет-магазини, платформи соціальних мереж, мобільні додатки та традиційні магазини.

Розвиток електронної комерції і омніканальних стратегій дозволяє клієнтам отримувати товари через різні канали, що сприяє збільшенню продажів та поліпшенню взаємодії з клієнтами.

Підвищення Кількості Данних від Дистриб'юторів:

Зростання обсягу та різноманітності даних, які надходять від дистриб'юторів, дозволяє виробникам отримувати більше інформації про ринкові умови, поведінку споживачів та інші фактори, що впливають на бізнес.

Зростання обсягу та різноманітності даних від дистриб'юторів дозволяє виробникам отримувати більше інформації про ринкові умови, що впливає на виробничі стратегії та планування.

Співпраця та Відкриті API:

Збільшення обсягу обміну даними між різними учасниками ланцюга постачання, використовуючи відкриті API, сприяє більшій прозорості та швидкому реагуванню на зміни в умовах ринку.

Спрощення Замовлення та Постачання:

Використання електронних систем для автоматизації процесів замовлення та постачання спрощує взаємодію між виробниками та дистриб'юторами.

Використання електронних систем дозволяє автоматизувати процеси замовлення та постачання, зменшуючи час та помилки.

Стратегії Сталого Розвитку:

Зростаюча увага до питань сталого розвитку впливає на стратегії дистрибуції, зокрема використання екологічно чистих транспортних засобів та оптимізація упаковки.

Підвищення Кількості Реального Часу та Необхідності Відповіді:

Зростання значення реального часу у зборі та обміні аналітичною інформацією дозволяє бізнесу швидше реагувати на зміни та оптимізувати бізнес-процеси в реальному часі.

Ці тенденції відображають стрімкий розвиток технологій та стратегій у сфері дистрибуції та обміну аналітичною інформацією, спрямований на досягнення більшої ефективності та конкурентоспроможності на ринку.

1.2 Ретельний огляд існуючих методик та підходів до збору даних в системах обміну аналітичною інформацією.

Огляд існуючих методик та підходів до збору даних в системах обміну аналітичною інформацією може включати розгляд різних технологій, підходів та інструментів, які використовуються для збору, обробки та аналізу даних в контексті обміну аналітичною інформацією між виробниками та їхніми дистриб'юторами. Нижче наведено ключові аспекти:

Електронний обмін даними (EDI) – процес передавання данних шляхом створення стуктурованих (форматXML/JSON) та неструктурованих (PDF, PNG, JPG, DOCX, XLSX) документів організацями.

Ці документи можуть включати в себе інформацію про торгові, фінансові трансопртні операції,

EDI є стандартом для електронного обміну структурованими даними між компаніями. Використовується для автоматизації бізнес-процесів, включаючи замовлення та обмін іншою бізнес-інформацією.

Переваги: Автоматизація, зменшення помилок, ефективний обмін даними.

API-інтеграція:

Використання інтерфейсів програмування додатків (API) для забезпечення взаємодії між інформаційними системами виробників і дистриб'юторів.

Використання API для забезпечення взаємодії між інформаційними системами. Це може включати RESTful, SOAP або інші протоколи.

API дозволяють виробникам та дистриб'юторам обмінюватися даними в реальному часі, забезпечуючи ефективний та надійний обмін інформацією.

Переваги: Реальний час обмін даними, структуровані запити та відповіді.

Інтернет речей (IoT):

Використання сенсорів та з'єднаних пристроїв для збору даних, таких як рух товарів, рівень запасів тощо.

Переваги: Реальний час моніторинг, автоматизований збір даних.

Системи управління відносинами з клієнтами (CRM):

Використання CRM для відстеження та управління взаємодією з дистриб'юторами, а також для аналізу важливих даних.

Використання CRM для відстеження та управління взаємодією з дистриб'юторами, а також для аналізу важливих даних.

Деталі: CRM забезпечує інструменти для збору та обробки інформації про клієнтів, включаючи замовлення та попит.

Переваги: Зберігання даних про клієнтів, історію взаємодії та можливість аналізу та прогнозування замовлень.

Автоматизовані системи управління ланцюгом постачання (SCM):

Використання SCM-систем для відстеження руху товарів, управління запасами та взаємодії з дистриб'юторами.

Переваги: Оптимізація ланцюга постачання, прогнозування потреб виробництва.

Використання платформ електронної комерції:

Використання електронних платформ для замовлення та обміну інформацією між виробником і дистриб'юторами.

Переваги: Зручна система замовлення, вбудована аналітика, простота інтеграції.

Звітність та аналітика:

Використання спеціальних інструментів для збору, аналізу та візуалізації даних, які надходять від дистриб'юторів.

Використання спеціальних інструментів для збору, аналізу та візуалізації даних від дистриб'юторів.

Деталі: Інструменти аналітики надають можливість витягти цінні інсайти з накопичених даних, сприяючи прийняттю обґрунтованих стратегічних рішень.

Переваги: Систематизований аналіз та звітність, можливість прийняття обґрунтованих стратегічних рішень.

В даному випадку важливо виділити метод, який є найбільш вдалим для налаштування та здійснення передачі даних. Це електронний обмін даними (EDI). Тож розглянемо більш дедально вказану методику.

Електронний обмін даними (EDI) - це процес обміну структурованими електронними документами між різними комп'ютерними системами. EDI дозволяє підприємствам ефективно обмінюватися бізнес-документами, такими як замовлення, накладні, рахунки, листи на оплату тощо, без необхідності ручного введення даних. Ось кроки, які зазвичай здійснюються в рамках процесу EDI:

Установлення Зв'язку:

Підприємства, які планують обмінюватися даними за допомогою EDI, спочатку встановлюють електронний зв'язок між своїми комп'ютерними системами. Це може бути внутрішньокорпоративна мережа або захищений канал Інтернету.

Стандартизація Документів:

Сторони, які беруть участь у обміні даними, домовляються про стандартизований формат для представлення різних типів документів. Найбільш відомими стандартами для EDI є ANSI X12, EDIFACT, та XML.

Розробка EDI-Мапування:

Кожен тип документа, такий як замовлення або накладна, має відповідний формат в стандарті EDI. Для кожного партнера розробляється EDI-мапування, яке визначає відповідність між структурою документів від кожного партнера та форматом EDI.

Створення EDI-Повідомлення:

При виникненні події, що викликає потребу в обміні даними (наприклад, створення замовлення), відправник створює EDI-повідомлення відповідного формату, яке містить інформацію про цю подію.

Передача EDI-Повідомлення:

EDI-повідомлення передається по електронному каналу від відправника до отримувача. Це може бути здійснено через протоколи передачі файлів (FTP), електронну пошту, веб-сервіси або інші засоби забезпечення електронного обміну даними.

Прийом та Обробка EDI-Повідомлення:

Отримувач приймає EDI-повідомлення та використовує власні системи для обробки інформації. Система отримувача розбирає EDI-повідомлення, витягує необхідні дані та інтегрує їх у внутрішні системи.

Підтвердження Доставки та Отримання:

Відправник може отримати підтвердження про те, що EDI-повідомлення було успішно отримано та оброблено. Це допомагає партнерам бути впевненими в доставці та обробці важливих документів.

EDI використовується в різноманітних галузях, особливо в сфері логістики, виробництва, торгівлі тощо, де швидкий та точний обмін даними є важливим елементом ефективного бізнесу.

Електронний обмін даними (EDI) є технологією, яка дозволяє компаніям обмінюватися даними та документами електронним шляхом. Цей метод має численні переваги, серед яких:

Ефективність:

EDI дозволяє автоматизувати процеси обміну даними, що призводить до значного підвищення ефективності порівняно з традиційними методами обміну.

Швидкість:

Обмін даними в режимі реального часу допомагає підприємствам отримувати актуальну інформацію миттєво, що полегшує швидке реагування на зміни та прийняття стратегічних рішень.

Зменшення помилок:

Автоматизований процес EDI мінімізує ручне втручання, що допомагає уникнути помилок, що можуть виникнути при ручному обміні документами.

Економія часу та ресурсів:

Замість того, щоб витрачати час на обробку паперових документів або обробку електронних даних вручну, EDI дозволяє автоматизовано обмінюватися даними, що економить час та ресурси.

Прозорість та слідування:

EDI забезпечує прозорість у ланцюгу постачання, оскільки всі сторони мають доступ до однакової інформації. Це полегшує слідування за процесами та вирішення можливих проблем.

Безпека даних:

EDI використовує різні методи шифрування та аутентифікації для забезпечення безпеки обміну даними. Це дозволяє уникнути несанкціонованого доступу та зберегти конфіденційність даних.

Економія фінансових ресурсів:

Зменшення витрат на обробку, друку та пересилання документів сприяє економії фінансових ресурсів підприємства.

Відповідність стандартам:

EDI регулюється міжнародними стандартами, що сприяє відповідності та сумісності обміну даними між різними компаніями.

Зменшення паперового обігу:

Використання EDI дозволяє уникнути великої кількості паперових документів, що полегшує управління та зменшує вплив на навколишнє середовище.

Гнучкість та скасування:

EDI може легко адаптуватися до змін у бізнес-процесах та вимогах, що забезпечує гнучкість у використанні.

Ці переваги роблять EDI ефективним інструментом для оптимізації процесів обміну даними в сучасних бізнес-середовищах.

Розуміння аналітичних систем є життєво важливим для компаній, щоб використовувати потужність даних і приймати обґрунтовані рішення. Використовуючи різні типи аналітичних систем і зосереджуючись на ключових компонентах, організації можуть отримати цінну інформацію та досягти конкурентної переваги на сучасному динамічному ринку.

1.3 Огляд існуючих програмних рішень та аналіз існуючих практик аналітичних систем

В цьому підпункті розглянемо та проаналізуємо існуючі програмні рішення, виділимо їх переваги та недоліки. Також сформуємо основний перелік вимог до розроблюваної системи.

Нижче розглянемо декілька аналітичних систем, які використовують виробники для моніторингу та оцінки. Виявимо їхні переваги та недоліки, проаналізуємо способи передачі даних. На основі цього складемо основний список вимог до розроблюваної системи, враховуючи всі недоліки проаналізованих систем.

Oracle Fusion Cloud Supply Chain Planning

Oracle Fusion Cloud Supply Chain Planning - це потужне програмне забезпечення для планування ланцюжка поставок, яке пропонує широкий спектр функцій для прогнозування попиту, управління запасами та планування виробництва. Інтерфейс програми зображений на рис. 1.1.

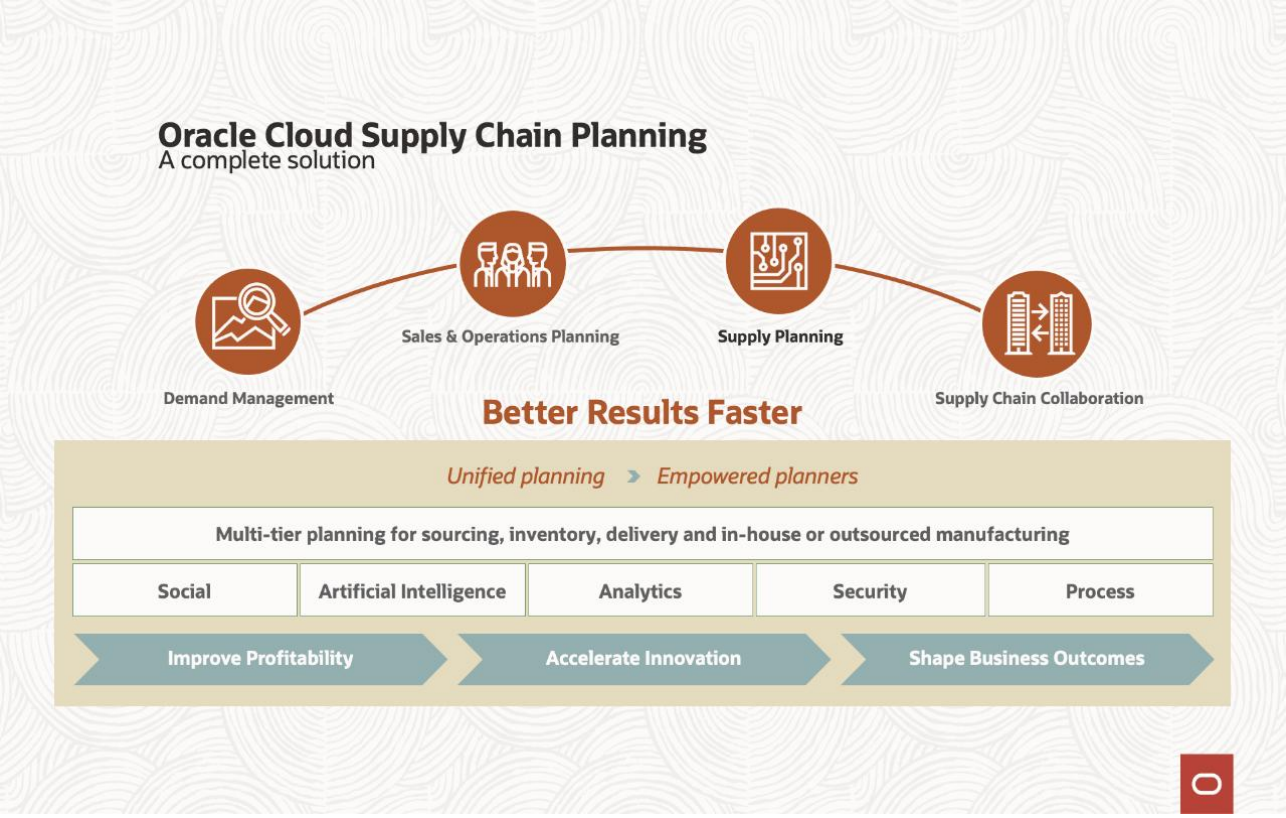


Рис 1.1 Oracle Fusion Cloud Supply Chain Planning

Основні переваги Oracle Fusion Cloud Supply Chain Planning:

Гнучке планування попиту та постачання: Oracle Fusion Cloud Supply Chain Planning пропонує широкий спектр моделей прогнозування попиту, які можна налаштувати відповідно до конкретних потреб бізнесу. Крім того, програмне забезпечення забезпечує гнучкість у плануванні постачання, дозволяючи користувачам прогнозувати потреби в запасах на основі різних сценаріїв попиту.

Автоматизоване моделювання та оптимізація: Oracle Fusion Cloud Supply Chain Planning пропонує широкий спектр інструментів для автоматизації моделювання та оптимізації ланцюжка поставок. Ці інструменти можуть

допомогти підприємствам підвищити ефективність своїх операцій та зменшити витрати.

Єдине джерело правди для даних: Oracle Fusion Cloud Supply Chain Planning забезпечує єдине джерело даних для всіх аспектів ланцюжка поставок. Це полегшує користувачам отримувати доступ до точних даних та приймати обґрунтовані рішення.

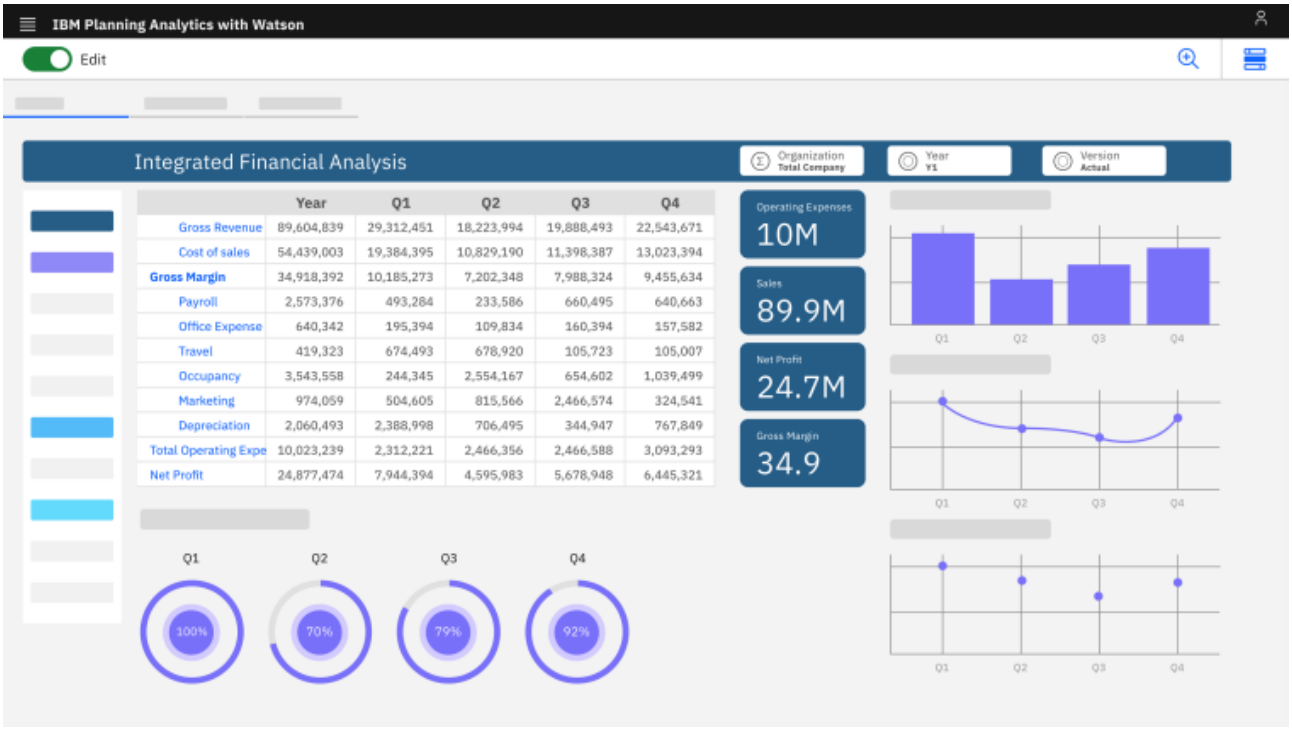
Основні недоліки Oracle Fusion Cloud Supply Chain Planning:

Може бути дорогим: Oracle Fusion Cloud Supply Chain Planning - це висококласне програмне забезпечення, яке може бути дорогим для невеликих підприємств.

Може бути складним для використання: Oracle Fusion Cloud Supply Chain Planning має широкий спектр функцій, які можуть бути складними для вивчення та використання.

2. IBM Planning Analytics

IBM Planning Analytics - це потужне програмне забезпечення для бізнес-прогнозування, яке пропонує широкий спектр функцій для прогнозування попиту, бюджетування, планування та аналізу. Інтерфейс програми зображений на рис. 1.2.



1.2 IBM Planning Analytics

Основні переваги IBM Planning Analytics:

- Широкий спектр функцій: IBM Planning Analytics пропонує широкий спектр функцій для прогнозування попиту, бюджетування, планування та аналізу. Це дозволяє користувачам адаптувати програмне забезпечення відповідно до своїх конкретних потреб.

- Гнучка архітектура: IBM Planning Analytics має гнучку архітектуру, яка дозволяє користувачам легко масштабувати програмне забезпечення відповідно до зростання їхніх потреб. IBM Planning Analytics можна розгорнути на локальному сервері, в хмарі або в гібридному середовищі.

- Сучасний інтерфейс користувача: IBM Planning Analytics має сучасний інтерфейс користувача, який робить програмне забезпечення простим у використанні та навігації. IBM Planning Analytics підтримує візуалізацію даних, що допомагає користувачам краще зрозуміти свої дані.

Недоліки:

- Може бути дорогим: IBM Planning Analytics - це висококласне програмне забезпечення, яке може бути дорогим для невеликих підприємств. Ціна залежить від розміру підприємства, кількості користувачів та функцій, які вибирають користувачі.
- Може вимагати значної інженерної роботи: IBM Planning Analytics має широкий спектр функцій, які можуть вимагати значної інженерної роботи для налаштування та використання. IBM Planning Analytics пропонує широкий спектр навчальних ресурсів, які можуть допомогти користувачам навчитися використовувати програмне забезпечення.

3. Qlik Sense

Qlik Sense - це потужний інструмент для бізнес-аналітики, який пропонує широкий спектр функцій для візуалізації даних, пошуку інформації та аналізу. Інтерфейс програми зображений на рис. 1.3.

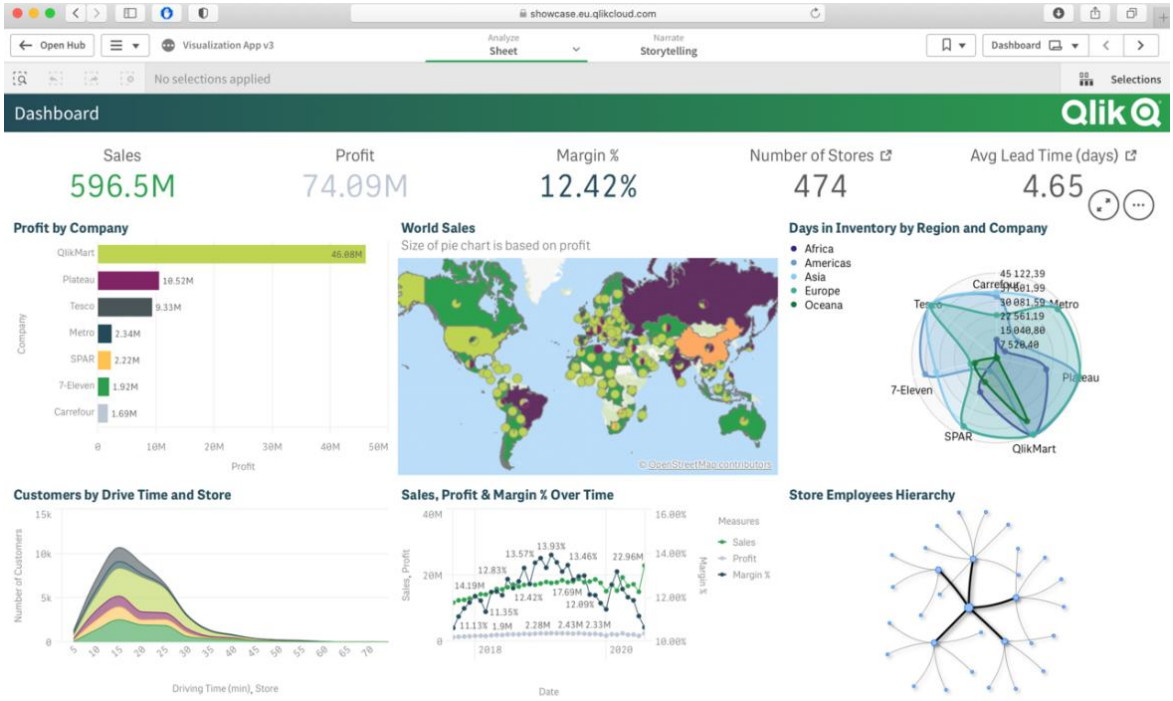


Рис 1.3 Qlik Sense

Переваги:

- Висока якість візуалізації даних: Qlik Sense пропонує широкий спектр інструментів для створення інформативних та привабливих візуалізацій даних. Візуалізації Qlik Sense використовують технологію гіперконектності, яка дозволяє користувачам легко взаємодіяти з даними та отримувати нові insights.

- Гнучкість у використанні: Qlik Sense є гнучким інструментом, який можна використовувати для різних цілей. Qlik Sense можна використовувати для аналізу даних з різних джерел, включаючи бази даних, електронні таблиці та файли. Qlik Sense також пропонує широкий спектр функцій для аналізу даних, включаючи аналіз тенденцій, аналіз чутливості та аналіз сценаріїв.

- Простота використання: Qlik Sense має простий інтерфейс користувача, який робить його доступним для широкого кола користувачів. Qlik Sense не вимагає спеціальних навичок у сфері бізнес-аналітики для

використання. Qlik Sense пропонує широкий спектр навчальних ресурсів, які можуть допомогти користувачам навчитися використовувати програмне забезпечення.

Недоліки:

- Може бути дорогим: Qlik Sense є платним програмним забезпеченням. Ціна залежить від кількості користувачів та функцій, які вибирають користувачі. Ціна Qlik Sense може бути недоступною для невеликих підприємств.
- Може бути складним у налаштуванні: Qlik Sense має широкий спектр функцій, які можуть бути складними для налаштування. Qlik Sense може вимагати значних витрат часу та зусиль для налаштування відповідно до конкретних потреб бізнесу. Qlik Sense пропонує широкий спектр навчальних ресурсів, які можуть допомогти користувачам налаштувати програмне забезпечення.

Проаналізувавши існуючі програмні рішення можна дійти до висновку, що для українського бізнесу наведені програмні рішення не підходять через ряд описаних причин.

Зважаючи на це варто окреслити основні неточності та врахувати їх при розробці системи.

1.4 Виділення основних переваг розроблюваної системи та формальна постановка вимог.

Розроблювана система має назву Spot2D.

Spot2D - це система для штучного інтелекту, яка використовується для прогнозування попиту на основі даних про продажі, дані про попит і поведінку клієнтів. Система використовує машинне навчання для виявлення тенденцій і шаблонів у даних, які можна використовувати для створення точних прогнозів.

Нижче виділимо основні вимоги до розроблюваної системи.

Основні переваги системи Spot2D на д іншими:

- Висока точність прогнозів: Система Spot2D використовує передові алгоритми машинного навчання для створення точних прогнозів попиту. Система може досягти точності до 95%.
- Широкий спектр застосувань: Система Spot2D може використовуватися для прогнозування попиту на широкий спектр товарів і послуг. Система може використовуватися в роздрібній торгівлі, виробництві, логістиці та інших галузях.
- Простота використання: Система Spot2D проста у використанні і не вимагає спеціальних навичок у сфері машинного навчання. Система може бути легко налаштована відповідно до конкретних потреб бізнесу.
- Система може допомогти бізнесу зібрати необхідні дані, якщо вони недоступні.

Основні недоліки системи Spot2D:

- Може бути дорогою: Система Spot2D є платною послугою. Ціна залежить від розміру бізнесу та кількості даних, які використовуються для прогнозування.
- Може вимагати значної кількості даних: Система Spot2D вимагає значної кількості даних для створення точних прогнозів. Бізнес повинен мати доступ до даних про продажі, дані про попит і поведінку клієнтів.
- Висока точність прогнозів: Система Spot2D використовує передові алгоритми машинного навчання для створення точних прогнозів попиту. Система може досягти точності до 95%.
    - Система використовує багатофакторний аналіз, який враховує широкий спектр факторів, що впливають на попит.

- Система постійно вчиться та адаптується до нових даних, що допомагає поліпшити точність прогнозів.

- Широкий спектр застосувань: Система Spot2D може використовуватися для прогнозування попиту на широкий спектр товарів і послуг. Система може використовуватися в роздрібній торгівлі, виробництві, логістиці та інших галузях.
  - Система підтримує різні типи даних, включаючи дані про продажі, дані про попит і поведінку клієнтів.
  - Система може бути легко налаштована відповідно до конкретних потреб бізнесу.

- Простота використання: Система Spot2D проста у використанні і не вимагає спеціальних навичок у сфері машинного навчання.
  - Система має простий інтерфейс користувача, який робить її легкою для вивчення та використання.
  - Система надає широкий спектр навчальних ресурсів, які можуть допомогти користувачам навчитися використовувати систему.

- Може доступна для українського бізнесу. Ціна залежить від розміру бізнесу та кількості даних, які використовуються для прогнозування.

Система Spot2D може бути більш підходящою для українських виробників, ніж Qlik Sense, IBM Planning Analytics та Oracle Fusion Cloud Supply Chain Planning, з кількох причин:

- Висока точність прогнозів: Система Spot2D використовує передові алгоритми машинного навчання для створення точних прогнозів попиту. Ця точність може бути особливо важливою для українських виробників, які працюють у мінливих ринкових умовах.

- Широкий спектр застосувань: Система Spot2D може використовуватися для прогнозування попиту на широкий спектр товарів і послуг. Це робить систему цінним інструментом для виробників будь-якої галузі.
- Простота використання: Система Spot2D проста у використанні і не вимагає спеціальних навичок у сфері машинного навчання. Це робить систему доступною для широкого кола користувачів, включаючи нетехнічних фахівців.
- Доступна ціна: Система Spot2D пропонує різні тарифні плани, які відповідають потребам різних бізнесів. Це робить систему доступною для невеликих та середніх українських виробників.

На відміну від Spot2D, Qlik Sense є інструментом для бізнес-аналітики, який не спеціалізується на прогнозуванні попиту. IBM Planning Analytics та Oracle Fusion Cloud Supply Chain Planning є потужними системами для планування ланцюжка поставок, але вони можуть бути складними у використанні та налаштуванні.

Ось деякі конкретні приклади того, як система Spot2D може допомогти українським виробникам:

- Виробники продуктів харчування можуть використовувати систему Spot2D для прогнозування попиту на сезонні продукти. Це може допомогти їм уникнути дефіциту або надмірних запасів.
- Виробники промислового обладнання можуть використовувати систему Spot2D для прогнозування попиту на обладнання для конкретних галузей. Це може допомогти їм оптимізувати своє виробництво та уникнути перевиробництва або недовиробництва.

- Виробники будівельних матеріалів можуть використовувати систему Spot2D для прогнозування попиту на будівельні матеріали в часі. Це може допомогти їм уникнути затримок у будівництві.

Розроблюване ПЗ має відповідати наступним основним вимогам:

- запуск Python скриптів у веббраузері;
- впровадження та підтрика бібліотек;
- збереження розроблених програм;
- завантаження та зчитування даних з файлів;
- містити базу даних інформації;
- формування звітів, графіків, статистики
- авторизація для створення Python скриптів;
- запуск програми та відтворення її результатів на різних персональних комп’ютерах

Висновки

Аналіз бізнес-процесів є важливою складовою зростання та успіху бізнесу. Виявляючи неефективність, усуваючи вузькі місця та використовуючи можливості для зростання за допомогою аналітичних систем, організації можуть повністю розкрити свій потенціал і отримати конкурентну перевагу на ринку.

Не можна недооцінювати важливість аналізу бізнес-процесів, коли йдеться про досягнення конкурентної переваги та просування прогресу вашої організації. Розкривши весь потенціал своїх аналітичних систем, ви зможете отримати цінну інформацію, визначити сфери, які потребують вдосконалення, і прийняти обґрунтовані рішення, які сприятимуть розвитку вашого бізнесу. Інвестиції в надійні аналітичні інструменти та використання стратегій, що керуються даними, не тільки оптимізують ваші операції, але й підвищать задоволеність клієнтів, збільшать дохід і зменшать витрати. Сила аналітики полягає в її здатності

надавати актуальну інформацію в режимі реального часу, яка дозволяє швидко приймати рішення.

В цьому розділі було доведено важливість застосування аналітичних систем для бізнесу, проаналізовано їх основні типи, окреслено основні вимоги до розроблюваної системи з виявлення її переваг.

РОЗДІЛ 2

ДОСЛІДЖЕННЯ ТЕМИ, АНАЛІЗ ОСНОВНИХ СКЛАДОВИХ СИСТЕМИ, ПОСТАНОВКА ЗАДАЧІ

Обмін аналітичною інформацією від дистриб'юторів - це процес, за допомогою якого виробники отримують від дистриб'юторів інформацію про продажі, запаси, поведінку клієнтів та інші фактори, які можуть вплинути на їхній бізнес. Ця інформація може використовуватися виробниками для прийняття кращих рішень у таких сферах, як:

Прогнозування попиту: Обмін аналітичною інформацією від дистриб'юторів може допомогти виробникам краще зрозуміти попит на їхні продукти та послуги, що може призвести до більш точних прогнозів попиту.

Планування ланцюжка поставок: Обмін аналітичною інформацією від дистриб'юторів може допомогти виробникам оптимізувати свій ланцюжок поставок, що може призвести до зниження витрат і підвищення ефективності.

Маркетинг і продажі: Обмін аналітичною інформацією від дистриб'юторів може допомогти виробникам краще зрозуміти потреби своїх клієнтів і розробляти більш ефективні маркетингові та збутові кампанії.

Роль обміну аналітичною інформацією від дистриб'юторів у бізнес-процесах можна описати наступним чином:

Вдосконалення прийняття рішень: Обмін аналітичною інформацією від дистриб'юторів може допомогти виробникам приймати більш обґрунтовані рішення, що може призвести до підвищення ефективності та прибутковості.

Співпраця та партнерство: Обмін аналітичною інформацією від дистриб'юторів може сприяти співпраці та партнерству між виробниками та дистриб'юторами, що може призвести до створення більш цінних продуктів і послуг для клієнтів.

Інноваційна діяльність: Обмін аналітичною інформацією від дистриб'юторів може допомогти виробникам розвивати інноваційні продукти і послуги, що може призвести до підвищення конкурентоспроможності.

Теоретичні аспекти обміну аналітичною інформацією від дистриб'юторів можна поділити на три основні групи:

Теорія прийняття рішень: Теорія прийняття рішень вивчає, як люди приймають рішення. Обмін аналітичною інформацією від дистриб'юторів може допомогти виробникам приймати більш обґрунтовані рішення, що є основою для ефективного управління бізнесом.

Теорія організаційної поведінки: Теорія організаційної поведінки вивчає поведінку людей в організаціях. Обмін аналітичною інформацією від дистриб'юторів може сприяти співпраці та партнерству між виробниками та дистриб'юторами, що є важливим для ефективної роботи ланцюжка поставок.

Теорія інновацій: Теорія інновацій вивчає процеси розробки та впровадження нових продуктів і послуг. Обмін аналітичною інформацією від дистриб'юторів може допомогти виробникам розвивати інноваційні продукти і послуги, що є ключем до успіху в конкурентному середовищі.

2.1 Розгляд теоретичних аспектів обміну аналітичною інформацією від дистриб'юторів та його ролі в бізнес-процесах

Коли йдеться про ведення успішного бізнесу, розуміння та вдосконалення ваших внутрішніх процесів має вирішальне значення. Аналіз бізнес-процесів (BPA) відіграє ключову роль у виявленні областей неефективності, вузьких місць і втрачених можливостей у вашій діяльності. Досліджуючи та оптимізуючи ці процеси, компанії можуть повністю розкрити свій потенціал і отримати конкурентну перевагу на ринку.

1.1 Чому BPA важливий для розвитку бізнесу

Ефективний аналіз бізнес-процесів є фундаментальним кроком до досягнення зростання та прогресу. Це дозволяє організаціям отримати повне

розуміння своєї діяльності, дозволяючи їм приймати обґрунтовані рішення та впроваджувати стратегічні вдосконалення. Аналізуючи різні бізнес-процеси, такі як управління ланцюгом постачання, управління взаємовідносинами з клієнтами та фінансові операції, компанії можуть визначити області для оптимізації та оптимізувати свої робочі процеси.

1.2 Виявлення неефективності та вузьких місць

Однією з основних цілей аналізу бізнес-процесів є виявлення неефективності та вузьких місць, які перешкоджають продуктивності. Вивчаючи кожен крок процесу, організації можуть точно визначити надмірності, ручні обхідні шляхи та інші фактори, які сприяють затримкам і помилкам. Цей рівень контролю дозволяє підприємствам розробляти оптимізовані процеси, які усувають відходи, зменшують витрати та покращують загальну ефективність.

1.3 Використання можливостей зростання

Аналіз бізнес-процесів також допомагає компаніям виявити можливості для зростання, які в іншому випадку могли б залишитися непоміченими. Аналізуючи дані та визначаючи закономірності, організації можуть визначати ринкові тенденції, уподобання клієнтів і нові вимоги. Озброївшись цією інформацією, підприємства можуть завчасно коригувати свої стратегії та використовувати ці можливості, максимізуючи свій потенціал успіху.

1.4 Роль технології в аналізі бізнес-процесів

Удосконалення технологій революціонізувало спосіб аналізу бізнес-процесів. Інструменти автоматизації, програмне забезпечення для аналізу даних та інші технологічні рішення відіграють вирішальну роль у оптимізації процесу аналізу та зборі цінної інформації. Компанії можуть використовувати ці інструменти для отримання значущих даних, візуалізації тенденцій і створення дієвих рекомендацій для оптимізації процесів.

Обмін аналітичною інформацією від дистриб'юторів є важливою частиною бізнес-процесів, оскільки надходження якісної та актуальної інформації від дистриб'юторів може суттєво впливати на прийняття стратегічних рішень компанією. Розглянемо деякі теоретичні аспекти та роль обміну аналітичною інформацією від дистриб'юторів у бізнес-процесах:

- Основи Обміну Аналітичною Інформацією:

Обмін аналітичною інформацією від дистриб'юторів передбачає передачу даних, які можуть включати в себе інформацію про продажі, стан запасів, поведінку покупців, маркетингові зусилля та інше. Ці дані можуть бути використані для аналізу тенденцій, прогнозування попиту та оптимізації бізнес-процесів.

- Роль в Управлінні Ланцюгом Постачання:

Інформація від дистриб'юторів є важливою для управління ланцюгом постачання. Знання про стан запасів, час поставки та інші параметри допомагають виробникам оптимізувати запаси, уникати затримок та покращувати ефективність постачання.

- Аналітика та Прийняття Рішень:

Аналіз аналітичної інформації від дистриб'юторів дозволяє виробникам зрозуміти ефективність продажів, виявляти тенденції, оцінювати результати маркетингових кампаній та приймати інформовані рішення для подальшого розвитку бізнесу.

- Планування Продукції та Замовлення:

Інформація від дистриб'юторів допомагає виробникам планувати виробництво відповідно до реального попиту. Вона також впливає на процес прийняття замовлень та оптимізацію запасів готової продукції.

- Взаємодія та Партнерство:

Обмін аналітичною інформацією між виробниками та дистриб'юторами підсилює взаємодію та партнерські відносини. Виробник може надавати дистриб'юторам інструменти для аналізу даних, що сприяє спільному розвитку стратегій та досягненню спільних цілей.

- Підвищення Ефективності Маркетингу:

Знання про ефективність маркетингових кампаній дозволяє виробникам та дистриб'юторам визначати найбільш прибуткові канали реклами та вдосконалювати стратегії маркетингу.

- Адаптація до Змін Ринкових Умов:

Аналіз інформації від дистриб'юторів дозволяє виробникам швидше реагувати на зміни на ринку, адаптувати стратегії та заходи до реальних умов.

2.2 Основи оптимізації збору даних та їх вплив на аналітичні системи.

Оптимізація збору даних має вирішальне значення для виробників з кількох причин. По-перше, оптимізація даних дозволяє виробникам підвищити загальну ефективність і продуктивність. Збираючи та аналізуючи відповідні дані, виробники можуть виявити вузькі місця у своїх процесах, оптимізувати робочі процеси та усунути непотрібні кроки. Це в кінцевому підсумку призводить до економії витрат і підвищення операційної ефективності.

Крім того, аналітичні системи відіграють вирішальну роль в аналітиці бізнес-процесів. Ці системи надають цінну інформацію та допомагають виробникам приймати обґрунтовані рішення. Аналізуючи дані, зібрані з різних джерел, виробники можуть виявити тенденції, закономірності та потенційні ризики. Це дозволяє їм впроваджувати проактивні заходи та приймати рішення на основі даних для оптимізації своїх бізнес-процесів.

Коли йдеться про оптимізацію інформації, виробники повинні враховувати різні фактори. По-перше, вони повинні забезпечити точність і чистоту даних. Перевірка та очищення даних гарантує, що для прийняття рішень

використовується лише надійна та точна інформація. Крім того, виробники повинні надавати пріоритет безпеці та конфіденційності даних. Впровадження надійних заходів безпеки допомагає захистити конфіденційну інформацію від несанкціонованого доступу та потенційних порушень.

Виробники також повинні зосередитися на інтеграції та інтероперабельності даних. Інтеграція даних з різних джерел дає змогу отримати цілісне уявлення про операції та сприяє кращому аналізу даних. Інтероперабельність забезпечує безперебійний обмін даними між системами, що дає змогу отримувати інформацію в режимі реального часу та оптимізувати процеси.

Що стосується методів обробки, виробники повинні застосовувати гнучкі та масштабовані підходи. Використовуючи такі технології, як хмарні обчислення та автоматизація, виробники можуть ефективно обробляти великі обсяги даних та адаптуватися до мінливих потреб бізнесу. Впровадження систем управління даними забезпечує їхню якість, узгодженість і відповідність нормативним вимогам.

Оптимізація збору даних - це процес покращення якості та кількості даних, які збираються для аналітичних систем. Цей процес включає в себе такі завдання, як:

- Визначення даних, які потрібно збирати: Першим кроком у процесі оптимізації збору даних є визначення даних, які потрібно збирати. Це можна зробити, визначивши, які дані необхідні для підтримки аналітичних запитів і завдань.

Визначення даних, які потрібно збирати

- Оптимізація процесу збору даних: Після того, як визначено, які дані потрібно збирати, наступним кроком є оптимізація процесу збору

даних. Це можна зробити, покращивши ефективність процесу збору даних, зменшивши витрати на збір даних та підвищивши точність даних.

Оптимізація процесу збору даних

- Забезпечення якості даних: Останнім кроком у процесі оптимізації збору даних є забезпечення якості даних. Це можна зробити, запровадивши процеси для очищення даних, усунення аномалій та виправлення помилок.

Вплив оптимізації збору даних на аналітичні системи

Оптимізація збору даних може мати значний вплив на аналітичні системи. Ось деякі з переваг оптимізації збору даних:

- Поліпшення якості аналітичних результатів: Оптимізація збору даних може привести до поліпшення якості аналітичних результатів. Це відбувається тому, що аналітичні системи працюють краще з високоякісними даними.

Покращення якості аналітичних результатів

- Зменшення витрат на аналіз: Оптимізація збору даних може призвести до зменшення витрат на аналіз. Це відбувається тому, що аналітичні системи можуть обробляти дані швидше і ефективніше з високоякісними даними.

Зменшення витрат на аналіз

- Підвищення ефективності прийняття рішень: Оптимізація збору даних може призвести до підвищення ефективності прийняття рішень. Це відбувається тому, що аналітичні системи можуть надавати більш точну і актуальну інформацію для прийняття рішень.

Підвищення ефективності прийняття рішень

Оптимізація збору даних є важливим кроком для будь-якої організації, яка хоче отримати максимальну віддачу від своїх аналітичних систем.

Оптимізація збору даних впливає на ефективність та точність аналітичних систем. Нижче подані основи оптимізації збору даних та їх вплив на аналітичні системи:

Чіткість та Спрощення:

Оптимізація починається з чіткого визначення мети збору даних. Важливо обрати лише ті дані, які дійсно необхідні для досягнення поставленої мети, уникати збору зайвої інформації.

Структурованість та Консистентність:

Збір даних повинен бути структурованим та консистентним. Використання стандартів та об'єднання даних у єдиний формат полегшує їх обробку та аналіз.

Автоматизація Збору Даних:

Автоматизація процесу збору даних дозволяє отримувати дані в реальному часі та зменшує ймовірність помилок. Використання інструментів автоматизації сприяє ефективнішому збору та обробці даних.

Валідація та Перевірка Даних:

Важливо включити механізми валідації даних для виявлення та виправлення помилок на етапі збору. Це сприяє підвищенню якості даних та надійності аналітичних результатів.

- Захист приватності та відповідність:

Забезпечення відповідності з правилами та стандартами щодо захисту особистої інформації важливо для збереження довіри клієнтів та виконання законодавчих вимог.

- Оптимізація запитів:

У випадку баз даних оптимізація запитів може включати індексацію, використання кешування та інші методи для прискорення витратних операцій збору даних.

- Моніторинг та аналіз продуктивності:

Регулярний моніторинг продуктивності системи збору даних дозволяє вчасно виявляти можливі проблеми та вдосконалювати процеси для оптимізації.

- Гнучкість та розширюваність:

Забезпечення гнучкості та можливості розширення системи збору даних дозволяє легше адаптуватися до змін в бізнес-вимогах та обсязі даних.

- Крос-функціональна співпраця:

Збір даних повинен бути інтегрованим з різними департаментами та системами компанії, що сприяє вивченню повного спектру інформації та розширює можливості аналізу.

- Навчання моделей та машинне навчання:

Використання методів машинного навчання для аналізу та передбачення може значно покращити результати. Важливо постійно навчати моделі та вдосконалювати їх з часом.

Оптимізований процес збору даних є ключовим для створення надійних та ефективних аналітичних систем, які допомагають виробникам приймати інформовані стратегічні рішення.

2.3 Виявлення функціональних та нефункціональних вимог до аналітичної системи

На основі описаного вище якісних та кількісних характеристик аналітичних систем можемо визначити основін функціональні та нефункціональні вимоги.

Функціональні вимоги — це конкретні функції та сервіси, які має надавати система. Ось можливий список функціональних вимог для системи Spot2D (примітка: вони вигадані, оскільки не має інформації про конкретну систему):

- **Реєстрація та Авторизація:** можливість реєстрації нових користувачів. Також важливо поділити користувачів за ролями, наприклад: адміністратор, менеджер дистрибютора, менеджер виробника, користувач, автозавантаження та ін.)
- **Вказання географічних координат для кожноого клієнта.** Вказування місця розташування.
- **Розшарування точок доставки на клієнта та торгову точку**
- **Редагування та Видалення Точок:** Можливість користувача редагувати деталі точок, створювати клоновані ТТ для розподілення продажів, перенесення продажів з однієї на іншу ТТ.
- **Здатність видаляти непотрібні точки.**
- **Категоризація точок:** Можливість додавати категорії до точок (наприклад, "Ресторани", "Магазини", "Події"). Фільтрація точок за категоріями. Додавання додаткових параметрів, таких як «Тип ТТ», «Мережа», «Тип кухні», вид торгівлі (оптова, рознична, Horeca)
- **Пошук Точок:** Віднесення ТТ до певного регіону
- **База дистриб`юторів:** присвоєння унікального id в системі для кожного окремого дистриб`ютора, внесення відповідальних співробітників у базу та надання їм доступу до системи
- **Події та Нагадування.** Можливість додавати нагадування пов'язані з неточністю, помилками чи корректністю завантаження данних.
- **Імпорт даних:** Система Spot2D повинна мати можливість імпортувати дані з різних джерел, включаючи бази даних, електронні таблиці та файли. Система повинна підтримувати різні формати даних, щоб унеможливити необхідність конвертації даних.

- **Підготовка даних:** Система Spot2D повинна мати можливість очищати та готувати дані для використання в процесі прогнозування. Система повинна мати можливість видаляти аномалії, заповнювати прогалини та обробляти неповні дані.
- **Прогнозування попиту:** Система Spot2D повинна використовувати передові алгоритми машинного навчання для створення точних прогнозів попиту. Система повинна підтримувати різні моделі прогнозування, щоб користувачі могли вибрати модель, яка найкраще відповідає їхнім потребам.
- **Візуалізація прогнозів:** Система Spot2D повинна мати можливість візуалізувати прогнози попиту для кращого розуміння. Система повинна підтримувати різні типи візуалізацій, щоб користувачі могли вибрати візуалізацію, яка найкраще відповідає їхнім потребам.
- **Аналіз прогнозів:** Система Spot2D повинна мати можливість аналізувати прогнози попиту для виявлення тенденцій і шаблонів. Система повинна підтримувати різні інструменти аналізу, щоб користувачі могли отримати глибинне розуміння своїх прогнозів.
- **Зіставлення кодів продуктів виробника та дистриб`ютора:** система повинна надавати можливість привязувати коди виробника
- **Ведення переліку територіальних та регіональних менеджерів виробника:** привязка дистриб`юторів до менеджерів
- **Перегляд та оновлення списку продукції виробника:** виробник повинен мати можливість оновлювати асортимент, вносити корективи в назви, основні параметри; додавати нові продукти, встановлювати їх активність та видаляти застарілі позиції
- **Простий та доступний спосіб передачі даних:** використання нересурснозатратного способу передачі даних, який зможе налаштувати системний адміністратор клієнта

Нефункціональні вимоги — це характеристики системи, які не визначають її функціональність, але визначають, яким чином ця функціональність повинна бути реалізована. Ось приклади нефункціональних вимог для системи Spot2D:

- Видимість та Зручність Використання: інтерфейс користувача повинен бути інтуїтивно зрозумілим та зручним для використання. Метрика: середній час, який витрачає користувач на ознайомлення з системою та виконання базових дій.
- Продуктивність та Відгуки: система повинна швидко реагувати на дії користувача та надавати оперативні результати. Метрика: середній час відгуку системи на запити користувачів.
- Надійність та Доступність: система повинна бути доступною 99.9% часу та має забезпечувати відновлення після можливих збоїв. Метрика: відсоток часу доступності системи за певний період.
- Безпека та Конфіденційність: забезпечення заходів безпеки для захисту конфіденційності даних користувачів. Метрика: кількість успішно виявлених атак або невдалі спроби доступу.
- Масштабованість: система повинна бути масштабованою для ефективної роботи з ростом кількості користувачів та обсягу даних. Метрика: Продуктивність системи при збільшенні завантаження.
- Сумісність та Розширюваність: система повинна бути сумісною з різними операційними системами та браузерами, а також легко розширюватися новими функціями. Метрика: Частота випуску нових версій та рівень сумісності з різними середовищами.
- Витрати та Ресурси: система повинна ефективно використовувати ресурси, такі як пам'ять та обчислювальна потужність. Метрика: Витрати ресурсів на операцію або запит.
- Технічна Підтримка: Забезпечення технічної підтримки для вирішення проблем користувачів та виправлення помилок. Метрика: Середній час від виявлення проблеми до її виправлення.

- Екологічність: мінімізація екологічного впливу, наприклад, шляхом ефективного використання ресурсів.

Висновок

Підсумовуючи, не можна недооцінювати важливість аналізу бізнес-процесів, коли йдеться про досягнення конкурентної переваги та просування прогресу бізнесу. Розкривши весь потенціал аналітичних систем, можливості взаємодії з нею, налагодженого ланцюга передачі інформації, можна отримати цінну інформацію, визначити сфери, які потребують вдосконалення, і прийняти обґрунтовані рішення, які сприятимуть розвитку як систем, так і бізнесу в цілому.

Інвестиції в надійні аналітичні інструменти та використання стратегій, що керуються даними, не тільки оптимізують всі операції, але й підвищують задоволеність клієнтів, збільшують дохід і зменшують витрати. Сила аналітики полягає в її здатності надавати актуальну інформацію в режимі реального часу, яка дозволяє швидко приймати рішення.

В цьому розділі я розглянула теоретичну частину основних аспектів **о**бміну аналітичною інформацією від дистриб'юторів та його ролі в бізнес-процесах. Також було виявлено основні напрямки оптимізації збору даних та їх вплив на аналітичні системи. Було опрацьовано та зазначено перелік функціональних та нефункціональних вимог, з урахуванням недоліків систем-аналогів.

РОЗДІЛ 3

ПІДБІР ТА РОЗРОБКА АРХІТЕКТУРНИХ РІШЕНЬ ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1 Опис методології дослідження та вибір методів оптимізації збору даних.

Методології дослідження - це набори принципів, процедур та методів, які використовуються для збирання та аналізу інформації. Вони можуть бути використані для дослідження будь-якої теми, включаючи створення програмного продукту.

При створенні програмного продукту методології дослідження використовуються для збирання інформації про потреби користувачів, вимоги до продукту та фактори навколишнього середовища. Ця інформація використовується для розробки вимог до продукту, які є основою для його створення.

Роль методологій дослідження при створенні програмного продукту полягає в наступному:

- Забезпечення системного підходу до проектування. Методології дослідження допомагають дослідити всі аспекти продукту, включаючи його цілі, вимоги, обмеження та фактори середовища. Це допомагає забезпечити, що продукт буде розроблений таким чином, щоб він задовольняв всі потреби користувачів і відповідав вимогам бізнесу.
- Збір повної та всебічної інформації. Методології дослідження допомагають зібрати повні та всебічні дані про продукт. Це допомагає забезпечити, що продукт буде розроблений на основі чіткої і точної інформації.
- Забезпечення об'єктивності і неупередженості. Методології дослідження допомагають забезпечити об'єктивність і неупередженість у

процесі проектування. Це допомагає гарантувати, що продукт буде розроблений на основі реальних потреб і вимог.

Конкретні переваги використання методологій дослідження при створенні програмного продукту включають:

- Зменшення ризику невдачі проекту. Методології дослідження допомагають виявити потенційні проблеми на ранніх етапах проекту. Це може допомогти зменшити ризик невдачі проекту і заощадити час і гроші.
- Покращення якості продукту. Методології дослідження допомагають забезпечити, що продукт буде розроблений відповідно до вимог і обмежень. Це може допомогти покращити якість продукту і його здатність задовольняти потреби користувачів.
- Збільшення продуктивності. Методології дослідження можуть допомогти організувати процес проектування і забезпечити ефективне використання ресурсів. Це може збільшити продуктивності і скоротити час розробки продукту.

Вибір методів дослідження, які будуть використані при створенні програмного продукту, залежить від конкретних потреб проекту. Однак, використання методологій дослідження є важливою частиною процесу створення програмного продукту, оскільки вони можуть допомогти забезпечити успіх проекту.

Використання методологій дослідження при створенні програмного продукту є важливим фактором, який може допомогти забезпечити успіх проекту.

В своєму проекті я буду застосовувати наступні методології для виявлення основних вимог до програмного продукту.

3.1.1 Аналіз потреб до ПЗ

Методологія: Ретельне вивчення бізнес-потреб та вимог до збору та обробки аналітичної інформації. Визначення ключових параметрів, які слід враховувати при зборі даних.

При аналізі потреб мною було виявлено наступні вимоги у вигляді специфікації:

Мета Документа

Цей документ призначений для опису вимог до розробки інформаційної системи обміну аналітичною інформацією від дистриб'юторів.

Контекст Проекту

Інформаційна система призначена для отримання, обробки та аналізу даних від дистриб'юторів для підтримки бізнес-процесів.

Основні бізнес-потреби та вимоги до збору та обробки аналітичної інформації від дистрибюторів включають наступне:

- Забезпечення розуміння поточної ситуації. Аналітична інформація від дистрибюторів може допомогти компаніям зрозуміти поточну ситуацію на ринку, включаючи такі фактори, як обсяги продажів, ціни, відсоток повторних покупок тощо. Це може допомогти компаніям приймати більш обґрунтовані рішення про бізнес-стратегію та операції.
- Ідентифікація тенденцій та можливостей. Аналітична інформація від дистрибюторів може допомогти компаніям ідентифікувати тенденції на ринку та можливі можливості для зростання. Це може допомогти компаніям розробляти нові продукти та послуги, а також впроваджувати нові маркетингові стратегії.
- Покращення обслуговування клієнтів. Аналітична інформація від дистрибюторів може допомогти компаніям покращити обслуговування клієнтів, розуміючи їхні потреби та поведінку. Це може допомогти компаніям розробляти персоналізовані пропозиції та програми лояльності.

Конкретні вимоги до збору та обробки аналітичної інформації від дистрибюторів можуть включати наступне:

- Точність та повнота інформації. Аналітична інформація повинна бути точною та повною, щоб вона була корисною для прийняття рішень.
- Своєчасність інформації. Аналітична інформація повинна бути своєчасною, щоб компанії могли швидко реагувати на зміни на ринку.
- Легкість доступу до інформації. Аналітична інформація повинна бути легкодоступною для користувачів, які її потребують.

Для задоволення цих потреб компанії можуть використовувати різні методи збору та обробки аналітичної інформації від дистрибюторів. Основним методом в даній ситуації є аналіз електронних даних. Цей метод включає аналіз даних, які генеруються дистриб'юторами в процесі своїх операцій, таких як дані про продажі, запаси, витрати, повернення, переміщення, списання.

Всі дані повинні формуватись у таблиці. Основними файлами з таблицями є: відвантаження, залишки, довідник клієнтів, довідник продуктів, довідник торгових агентів. В структурі даних кожного з файлів повинна міститись наступна інформація:

Файл-довідник продуктів має містити наступні заголовки з інформацією:

Таблиця 3.1 Файл-довідник продуктів

Назва заголовка	Формат	Коментар
id дистриб'ютора	I	Унікальний номер дистриб’ютора в інформаційній системі

	nteger	
Код продукту дистриб'ютора	String (128)	Код продукту в обліковій системі дистриб’ютора
Назва продукту дистриб'ютора	String (128)	Назва продукту в обліковій системі дистриб’ютора
Код продукту виробника	String (128)	Код продукту виробника

Файл-довідник клієнтів має містити наступні заголовки з інформацією:

Таблиця 3.2 Опис файлу-довідника клієнтів

Назва заголовка	Формат	Коментар
id дистриб'ютора	Integer	Унікальний номер дистриб’ютора в інформаційній системі

Код клієнта ERP	String (128)	Код клієнта у внутрішній системі дистриб’ютора
Назва клієнта	String (128)	Назва клієнта у внутрішній системі дистриб’ютора
Адреса клієнта	String (128)	Адреса клієнта у внутрішній системі дистриб’ютора

Файл довідник торгових агентів повинен мати наступну структуру:

Таблиця 3.3 Опис файлу довідника торгових агентів

Назва заголовка	Формат	Коментар
id дистриб'ютора	Integer	Унікальний номер дистриб’ютора в інформаційній системі
Код Та	String (128)	Код ТА у внутрішній системі дистриб’ютора

Ім`я ТА	String (128)	ПІБ торгового агента

Файл Відвантаження має містити наступні заголовки з даними:

Таблиця 3.4 Опис файлу Відвантажень

Назва заголовка	Формат	Коментар
id дистриб'ютора	Integer	Унікальний номер дистриб’ютора в інформаційній системі
Дата	Date	Фактична дата відвантаження в одному з наступних форматів: dd.mm.yy, dd-mm-yy, dd.m.yy, dd-mm-yyyy, dd.mm.yyyy, dd-m-yyyy, yyyy.mm.dd, yyyy-mm-dd, yyyy-m-dd, yyyy-m-d, yyyy-mm-d
Код клієнта ERP	String (128)	Код клієнта у внутрішній системі

		дистриб’ютора
Код продукту дистриб'ютора	String (128)	Код продукту в обліковій системі дистриб’ютора
Кількість	Float	К-сть відвантажень
Сума відвантаження з ПДВ	Float	Сума відвантаження з ПДВ
Номер накладної	String (128)	Номер накладної в ОС дистриб`ютора

Файл Залишки повинен містити наступну інформацію:

Таблиця 3.5 Опис файлу Залишків

Назва заголовка	Формат	Коментар
id дистриб'ютора	Integer	Унікальний номер дистриб’ютора в інформаційній системі

Дата	Date	Фактична дата залишку в одному з наступних форматів: dd.mm.yy, dd-mm-yy, dd.m.yy, dd-mm-yyyy, dd.mm.yyyy, dd-m-yyyy, yyyy.mm.dd, yyyy-mm-dd, yyyy-m-dd, yyyy-m-d, yyyy-mm-d
Код продукту дистриб'ютора	String (128)	Код продукту в обліковій системі дистриб’ютора
Кількість	Float	К-сть залишків за конкретну дату

Функціональні, нефункціональні вимоги

Функціональні та нефункціональні вимоги було виявлено в попередньому розділі.

Інтерфейси

Користувацький інтерфейс: система повинна мати інтуїтивно зрозумілий та дружній інтерфейс для користувачів з можливістю налаштування відображення аналітичної інформації.

Інтерфейс інформаційної системи для обробки аналітичної інформації від дистрибюторів повинен відповідати наступним вимогам:

- Легкість використання. Інтерфейс повинен бути простим у використанні та зрозумілим для користувачів з різним рівнем підготовки.
- Інформативність. Інтерфейс повинен надавати користувачам всю необхідну інформацію, щоб вони могли швидко та легко знайти потрібні дані.
- Надійність. Інтерфейс повинен бути надійним і не повинен піддаватися збоїв.
- Безпека. Інтерфейс повинен бути захищений від несанкціонованого доступу.

Конкретні вимоги до інтерфейсу включають наступне:

- Використання візуальних елементів. Візуалізація даних може допомогти користувачам краще зрозуміти інформацію.
- Фільтрація та сортування даних. Користувачі повинні мати можливість фільтрувати та сортувати дані за різними параметрами, щоб вони могли швидко знайти потрібну інформацію.
- Можливість генерування звітів. Користувачі повинні мати можливість генерувати звіти, щоб вони могли ділитися інформацією з іншими.

При проектуванні інтерфейсу необхідно враховувати потреби різних типів користувачів, включаючи:

- Адміністраторів. Адміністратори повинні мати можливість налаштовувати систему та управляти нею.
- Аналітиків. Аналітики повинні мати можливість аналізувати дані та отримувати висновки.

- Бізнес-користувачів. Бізнес-користувачі повинні мати можливість приймати рішення на основі аналітичної інформації.

Важливо, щоб інтерфейс був адаптивним і міг змінюватися відповідно до потреб користувачів.

4.2 Інтеграція: система повинна бути інтегрована з існуючими бізнес-процесами та системами.

Технічні Вимоги

Платформа: система повинна бути розроблена для використання на платформах, таких як Windows, Linux та macOS.

Мови Програмування: розробка повинна використовувати мови програмування, такі як Java, Python, JavaScript.

Визначення джерел даних:

Визначення джерел даних, які використовують дистриб'ютори. Врахування різноманіття форматів та типів даних.

Дистриб'ютори використовують різноманітні джерела даних для забезпечення обміну аналітичною інформацією. Ось деякі типові джерела даних, їх формати та типи даних:

Електронні замовлення - інформація про товари, кількість, ціни, дати постачань.

Формати: Електронні таблиці (наприклад, Excel), XML, JSON.

Інвентаризаційні дані - запаси, стан товарів, вартість товарів.

Формати: CSV, бази даних (SQL, NoSQL).

Дані продажів - кількість проданих товарів, прибуток, розподіл продажів за період.

Формати: XML, JSON, CSV.

Дані про клієнтів - персональні дані клієнтів, історія покупок, промокоди.

Формати: CSV, бази даних.

Фінансові дані - витрати, прибуток, податки, фінансові звіти.

Формати: PDF, CSV, XML.

Отже, зрівнявши всі джерела та структури даних, які будуть потрапляти в систему оптимальним варіантом буде формат – CSV.

Дані саме в цьому форматі найчастіше використовуються дистриб`юторами.

Він є простим для табличних даних, легко оброблюється та експортується, а отже, підходить найбільше.

3.1.2 Проектування оптимальної архітектури

Методологія: Розробка оптимальної архітектури системи збору даних, враховуючи масштабованість, надійність та ефективність.

Оптимальна архітектура для системи збору даних від дистриб'юторів повинна враховувати масштабованість, надійність та ефективність. Важливо врахувати різноманітність джерел даних, обсяги інформації, швидкодію та інші аспекти. Тому розглянемо ключові складові оптимальної архітектури для інформаційної системи обробки аналітичної інформації від дистриб`юторів:

- Мікросервісна архітектура, де окремі компоненти відповідають за конкретні функції. Це полегшує масштабування та розвиток системи.
- Інтеграція з системами обміну даними, яка забезпечить можливість інтеграції з різноманітними джерелами даних, такими як системи обміну даними (EDI), API, бази даних тощо.
- Потокова обробка даних застосовуватиметься для обробки великих обсягів даних в режимі реального часу. Apache Kafka, Apache Flink або AWS Kinesis можуть бути корисними в цьому випадку.
- Інтелектуальний аналіз даних для впровадження алгоритмів машинного навчання та інтелектуального аналізу даних для отримання цінних інсайтів та покращення прогнозування.

- Кешування та оптимізація баз даних, як механізми кешування для прискорення доступу до часто використовуваних даних. Вони оптимізують структури баз даних та запити для забезпечення ефективності.
- Використання хмарних ресурсів прихмарних обчисленнях для масштабування ресурсів в залежності від обсягів та потреб системи.
- Моніторинг та логування важливо використати для відстеження продуктивності, виявлення помилок та відновлення після них.
- Безпека та контроль доступу забезпечить надійний рівень безпеки, використовуючи механізми шифрування, аутентифікації та авторизації.
- Резервне копіювання та відновлення заьезпечить безпеку даних в разі втрати або виходу з ладу системи.
- Гнучкість та розширюваність необхідно врахувати задля можливості майбутнього розширення та змін.

Програмні модулі, описаних вище складових архітектури системи будуть наведені в додатку до дипломної роботи.

Вибір технологій:

Вибір технологій та інструментів, які найкраще відповідають потребам системи. Розгляд хмарних рішень, баз даних, інструментів збору даних тощо.

Технології та інструменти, які найкраще відповідають потребам системи збору та обробки аналітичної інформації від дистрибюторів, повинні відповідати наступним вимогам:

Можливість обробки великих обсягів даних.

Системи повинні бути здатні обробляти великі обсяги даних, які генеруються дистриб'юторами в процесі своїх операцій. Для обробки великих обсягів даних у системі збору та обробки аналітичної інформації від дистриб'юторів важливо використовувати технології та підходи, які забезпечують ефективність та масштабованість. Серед них розподілена архітектура, системи

обробки пакетів, кешування та оптимізація запитів, моніторинг та аналіз продуктивності. Застосування цих підходів сприятиме ефективній обробці та аналізу великих обсягів даних у системі збору аналітичної інформації.

Можливість аналізу різноманітних даних. Системи повинні бути здатні аналізувати різноманітні типи даних, включаючи структуровані, неструктуровані та напівструктуровані дані.

Можливість візуалізації даних. Системи повинні забезпечувати візуалізацію даних, щоб користувачі могли краще зрозуміти інформацію. Візуалізація даних є важливою частиною системи збору та обробки аналітичної інформації від дистрибюторів. Вона допомагає користувачам краще зрозуміти дані і зробити більш обґрунтовані рішення.

Візуалізація даних може використовуватися для представлення даних у різних формах, таких як графіки, діаграми, карти та інфографіка. Ці форми представлення даних можуть допомогти користувачам:

Зрозуміти складні дані. Візуалізація даних може допомогти користувачам зрозуміти дані, які можуть бути складними або важко зрозуміти. Наприклад, графіки можуть використовуватися для представлення даних про продажі, щоб показати тенденції та порівняння.

Побачити зв'язки між даними. Візуалізація даних може допомогти користувачам побачити зв'язки між даними, які можуть бути невідомими. Наприклад, діаграми можуть використовуватися для представлення даних про продажі та маркетинг, щоб показати, як маркетингові кампанії впливають на продажі.

Зробити більш обґрунтовані рішення. Візуалізація даних може допомогти користувачам зробити більш обґрунтовані рішення, надаючи їм кращий огляд даних. Наприклад, карти можуть використовуватися для представлення даних про продажі, щоб показати, де знаходяться найкращі ринки для компанії.

У системі збору та обробки аналітичної інформації від дистрибюторів візуалізація даних може використовуватися для різних цілей, таких як:

Аналіз тенденцій на ринку. Візуалізація даних може використовуватися для аналізу даних про продажі, щоб визначити тенденції на ринку. Це може допомогти компаніям приймати рішення про такі фактори, як ціноутворення, маркетинг та розробка нових продуктів.

Ідентифікація можливостей для зростання. Візуалізація даних може використовуватися для аналізу даних про продажі, щоб ідентифікувати можливості для зростання. Це може допомогти компаніям визначити нові ринки, на які вони можуть вийти, або нові продукти, які вони можуть розробити.

Покращення обслуговування клієнтів. Візуалізація даних може використовуватися для аналізу даних про продажі, щоб зрозуміти потреби і поведінку клієнтів. Це може допомогти компаніям розробляти персоналізовані пропозиції та програми лояльності.

Важливо, щоб візуалізація даних була розроблена з урахуванням потреб користувачів. Вона повинна бути зрозумілою та інформативною, щоб користувачі могли легко зрозуміти інформацію, яка представлена.

Можливість інтеграції з іншими системами. Системи повинні бути здатні інтегруватися з іншими системами, такими як системи управління відносинами з клієнтами (CRM), системи управління ланцюгами поставок (SCM) та системи управління бізнес-процесами (BPM).

3.2 Розробка концепції нової методики та обґрунтування її вибору.

Розробка концепції нової методики є важливою частиною процесу проектування інформаційної системи. Вона дозволяє забезпечити ефективне виконання проекту та досягти поставлених цілей.

За допомогою розробка концепції нової методики при проектуванні інформаційної системи ми зможемо:

- Визначити основні цілі та завдання проекту. Концепція методики визначає, що необхідно досягти в результаті реалізації проекту та які завдання необхідно виконати для цього.
- Обрати оптимальний підхід до реалізації проекту. Концепція методики визначає, які технології, інструменти та методи будуть використовуватися для реалізації проекту.
- Оцінити витрати та ризики проекту. Концепція методики дозволяє визначити, скільки коштів буде потрібно для реалізації проекту та які ризики можуть виникнути в процесі реалізації.

3.3 Розробка та опис моделі даних інформаційної системи обміну аналітичної інформації від дистриб`юторів

Модель даних в інформаційній системі аналітичної інформації дистриб'юторів визначає структуру та взаємозв'язки між різними об'єктами даних, які використовуються для збереження та обробки інформації.

Процес збереження та обробки даних, відображення їх у звітах та діаграмах передбачає попереднє завантаження інформації до системи.

Тому нижче подана модель даних у вигляді взаємозв`язків між основними компонентами:

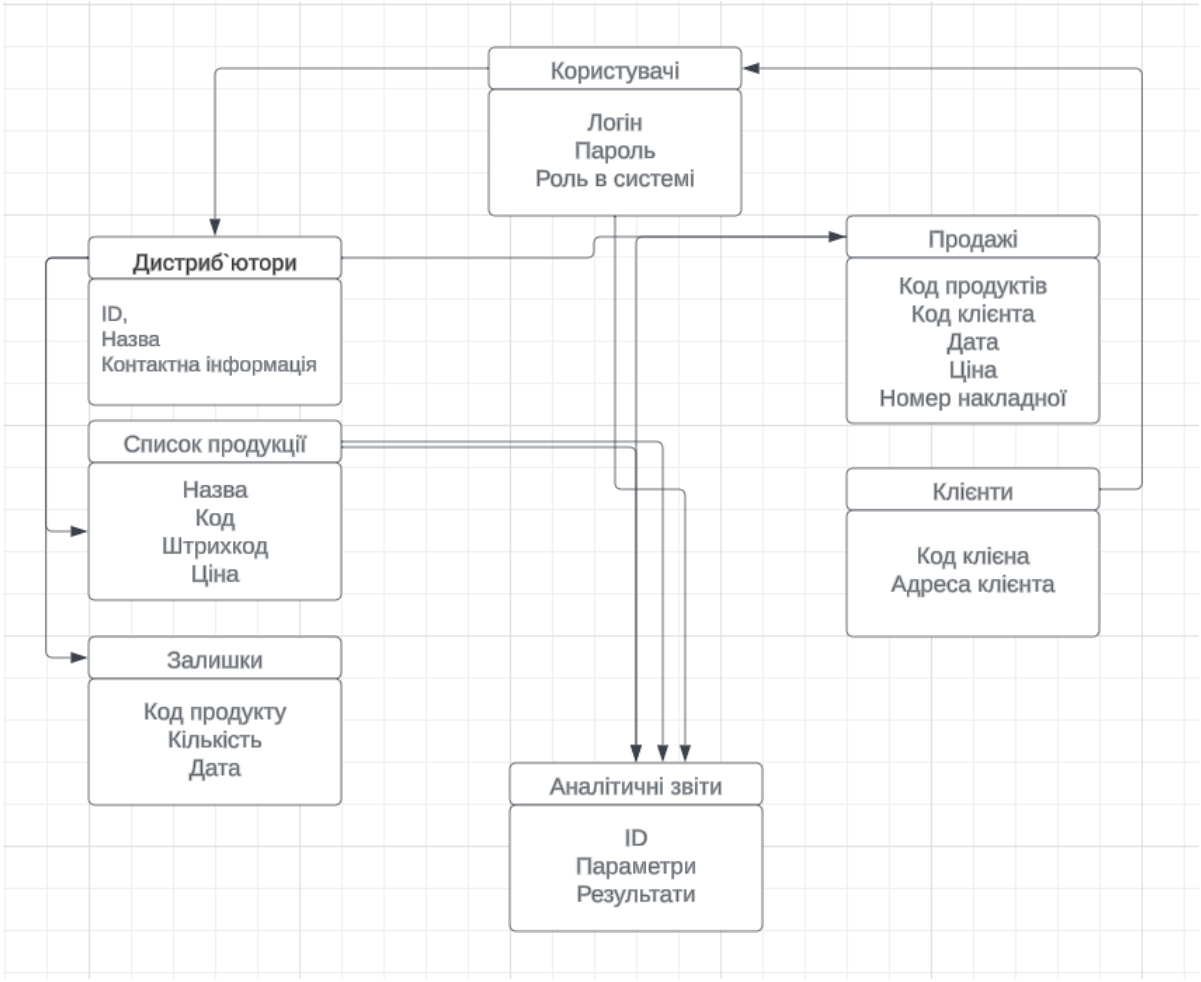


Рис. 3.1 Модель даних взаємозв`язків між основними компонентами

- Дистриб'ютори:

Об'єкти, які представляють інформацію про кожного дистриб'ютора. Включають атрибути, такі як ідентифікатор, назва, адреса, контактна інформація тощо.

- Список продукції:

Інформація про товари, які продає кожен дистриб'ютор. Це може включати атрибути, такі як ідентифікатор товару, назва, категорія, виробник, ціни тощо.

- Залишки:

Інформація про залишки товарів виробника на складі дистриб'ютора.

- Аналітичні Звіти:

Зберігання результатів аналітичних обчислень та звітів, які надають інформацію про продажі, ефективність, тенденції та інші ключові аспекти діяльності дистриб'ютора.

- Продажі:

Дані про продажі товарів та доходи дистриб'ютора. Може містити інформацію про обсяги продажів, прибуток, податки тощо.

- Клієнти:

Інформація про клієнтів та партнерів дистриб'ютора. Включає контактні дані, історію взаємодії та інші характеристики.

- Користувачі

Дані про користувачів системи. Кожен користувач має свою роль, яка надає доступ до певного функціоналу. Основні ролі: менеджер виробника, менеджер дистриб`ютора, адміністратор.

Також взаємозв`язки основних модулів можливості показують використання для документування архітектури системи. Вона може бути використана для пояснення того, як система працює, і для комунікації з іншими членами команди, які беруть участь у розробці системи.

Діаграма класів, наведена нижче на рис 3.3, грає ключову роль у розробці архітектури інформаційної системи, оскільки вона надає візуальне подання структури класів, їх взаємозв'язків та ролей у системі:

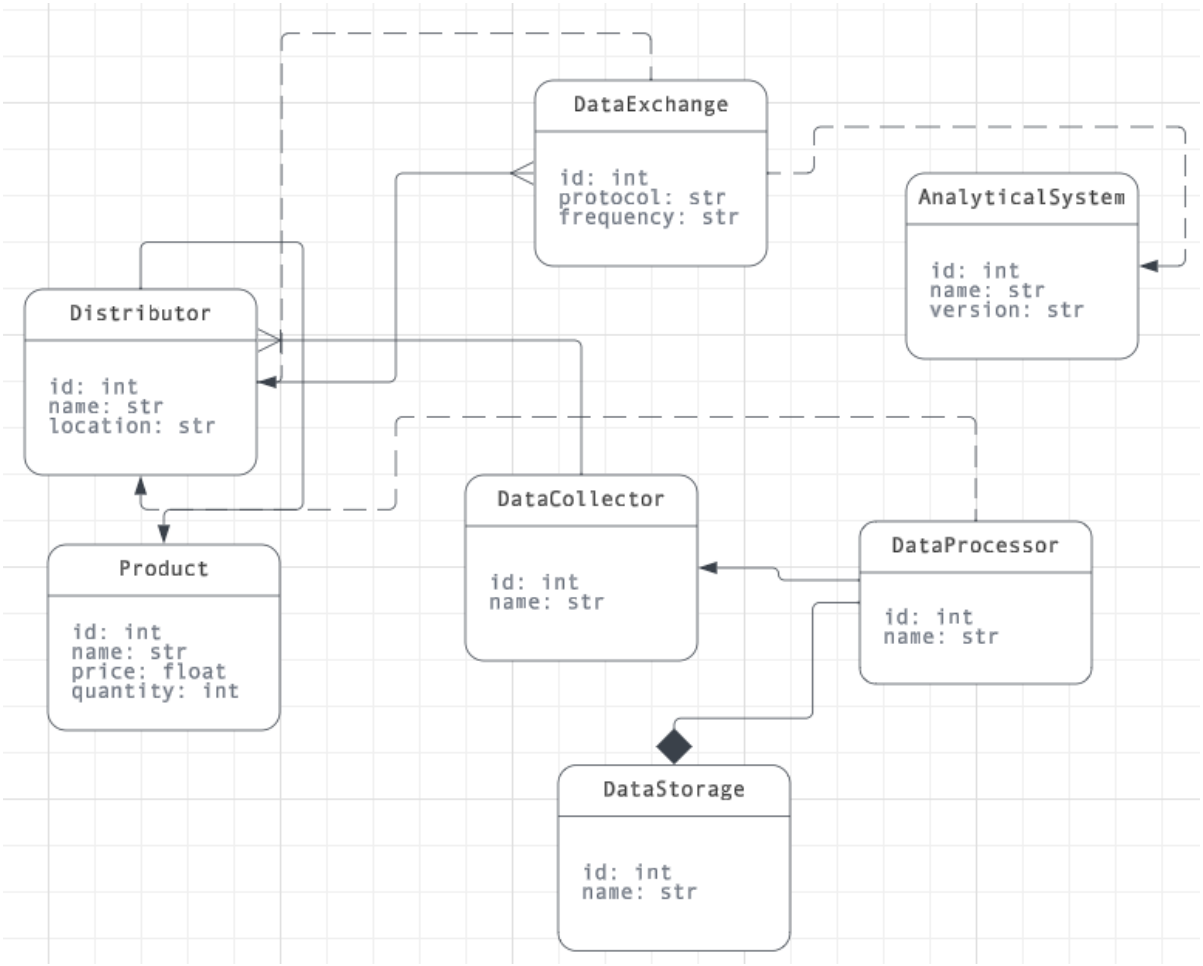


Рис.3.3 Діаграма класів

Інформаційна система збору аналітичної інформації грунтується на таких принципах, як отримання, обробка та відтворення інформації в реальному часі. Це забезпечує швидке та своєчасне відтворення інформації та здійснення аналітики.

В даному випадку система орієнтована на реалізацію її основних функцій збереження, обробку та відтворення.

На рисунку 3.2 подано схему процесу інформаційної системи обміну аналітичної інформації від дистриб`юторів:

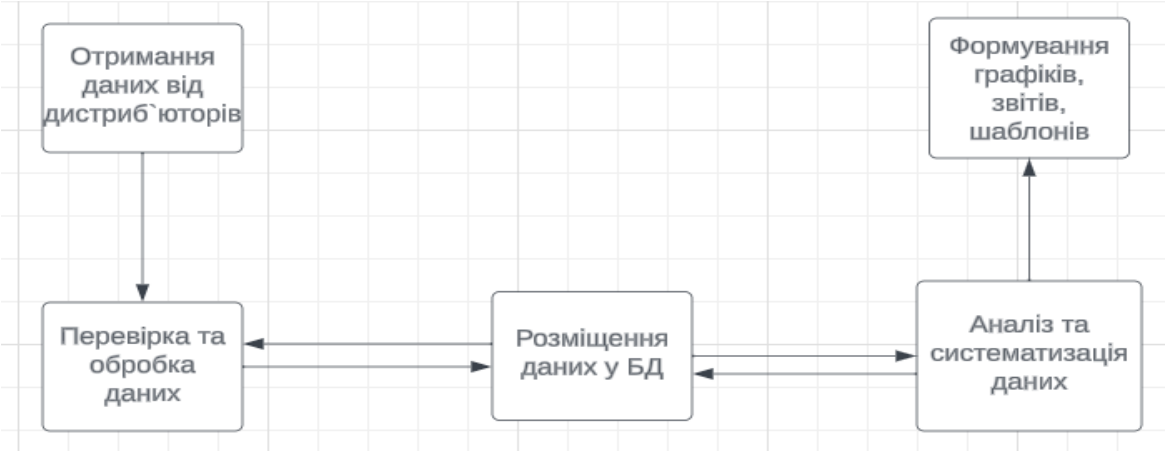


Рис. 3.4 Схема процесу інформаційної системи обміну аналітичної інформації від дистриб`юторів

Діаграма розгортання в UML (Unified Modeling Language) відображає фізичне розташування компонентів системи та їх взаємозв'язки. Ось опис діаграми розгортання для інформаційної системи аналітичної інформації для дистриб'юторів:

Клієнтська Сторона:

Веб-Браузери Клієнтів: клієнти мають доступ до системи через веб-браузери.

Мобільні Додатки: можливість використання мобільних додатків для доступу до системи.

Серверна Сторона:

Web Server: сервер для обробки запитів від клієнтів, виконання бізнес-логіки та відправлення відповідей.

База Даних Server: сервер бази даних для зберігання та управління даними.

Інфраструктурні Компоненти:

Хмарні Сервіси: використання хмарних сервісів для забезпечення масштабованості та доступності.

Балансувальник навантаження: компонент для розподілу навантаження між серверами для підтримки високої доступності.

Безпека та Аутентифікація:

Firewall: захист від несанкціонованого доступу до системи.

Система Аутентифікації та Авторизації: компонент для контролю доступу до різних ресурсів системи.

Ця діаграма розгортання вказує на фізичну структуру системи та взаємодії між її компонентами на рівні інфраструктури та мережі:

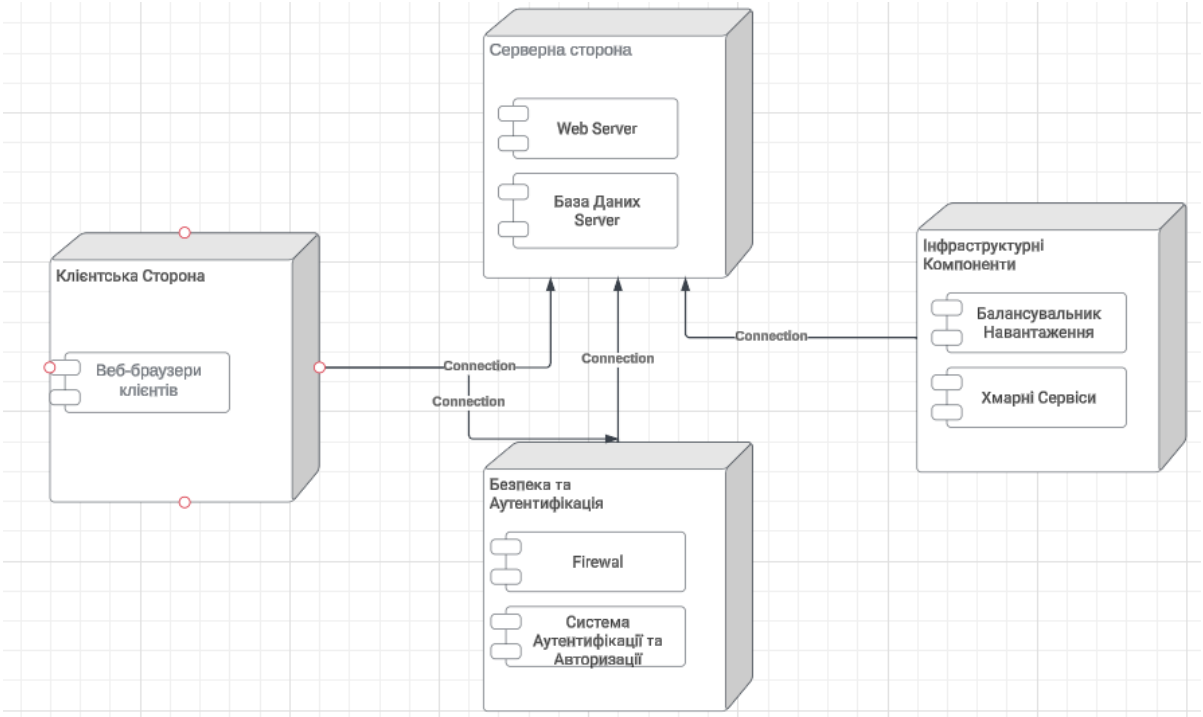


Рис. 3.5 Діаграма розгортання

3.4 Розгляд архітектурних рішень для покращення процесу збору даних в інформаційній системі.

Вибір архітектурного рішення для створення інформаційної системи обміну аналітичною інформацією від дистриб'юторів залежить від конкретних вимог та характеристик вашого бізнесу та інформаційної системи. Однак деякі архітектурні підходи можуть бути особливо корисними. Розглянемо кілька можливих варіантів:

Microservices Architecture:

Переваги:

- Гнучкість: Дозволяє вам розробляти та масштабувати різні компоненти системи незалежно. Легкість вдосконалення: Внесення змін в один сервіс не впливає на інші.
- Розгортання: Реалізація аналітичних сервісів, таких як збір даних, обробка та звітність, у вигляді мікросервісів.

Event-Driven Architecture:

Переваги:

- Реактивність: Система може реагувати на події негайно, що особливо важливо для реального часу аналітики.
- Гнучкість: Легко додавати нові джерела даних або аналітичні компоненти через обмін подіями.

Розгортання:

- Використання подій для сповіщення про нові дані, аналіз подій та відправлення звітів.

Data Lake Architecture:

Переваги:

- Зберігання великих обсягів даних: Дозволяє зберігати різноманітні дані без обмежень.
- Легкий доступ до даних: Дані можуть бути доступні для аналізу в реальному часі.

Розгортання:

- Створення центрального сховища даних, яке об'єднує дані від різних дистриб'юторів.

Real-Time Processing:

Переваги:

- Негайний відгук: Дозволяє аналізувати та реагувати на дані в реальному часі.
- Реактивність: Система може автоматично виконувати аналіз та надсилати звіти при зміні даних.

Розгортання:

- Використання потокових аналітичних сервісів для негайного аналізу даних.

Проаналізувавши основні варіанти архітектурних рішень для системи розглянемо деякі з них більш детально.

Мікрослужби – це незалежні розгортувані служби, котрі змодельовані навколо певного бізнес-доменую Існує певний ряд варіантів архітектури для вирішення завдань, з котрими можна зіткнутись. Архітектура базується на основі багаточисельних мікрослужб які між собою сполучні.

Розглянемо архітектурний тип орієнтований на служби які нейтральні до технологій та ключ яких їх незалежне розгортання.

З технічної точки зору мікро служби показують назовні можливості бізнеса, які вони приховують через одну чи декілька кінцевих точок в мережі. Взаємодія один з одним через мережу робить мікро служби окремим різновидом розподільної системи. Приховання зберігання та вивільнення даних, через окремі інтерфейси. Бази даних повинні бути призовані в границях окресленої служби.

Незажна розготуваність

Незалежна розгортуваність – це ідея, призначеня для змін які вносяться в мікро службу і розгортуються в середовищі без необхідності використання других служб. Головним фактором для забезпечення гарантій незалежної розготуваності є можливість змінювати одну службу при цьому не змінювати щось ще. Іншими

словами це значить що потрібні ясні і чіткі котракти між службамию Один із недоліків є проблема використовувати декілька баз даних.

Моделюються навколо бізнес домену

Внемення змін є через контур є вартісним процесом. Коли потрібно змінювати дві служби і виконати налаштування вказаних двох змін, то це займе більше часу чи внести однакові зміни всередині однієї служби. З цього виходить що трансграничні зміні між службами треба уникати або використовувати максимально рідко.

Розглянемо на прикладі компанію Music Corp яка є мультиорганізацією котрі залишається в бізнесі не дивлячись на те що в даний момент вона продає компакт-диски.

Пропоную перенести дану компанію в наше 21 століття і оцінити існуючу системну архітектуру. На рисунку зображено просту трьох шарову архітектуру. Зображено веб-інтерфейс користувача, бізнес-логіка в формі монолітного бекенду і збереження даних в базі даних. Ці шари належать різним операційним системам.

Додамо функцію при якій замовники можуть вказувати улюблений жанр музики. Потрібно змінити користувацький інтерфейс (UI), показуючи в ньому жанри на вибір, щоб бекенд забезпечував появу жанру в UI і зміну значення і щоб база даних приймала зміни які керуються кожною системою на рисунку 1.2 і розгорнути і коректному порядку. Вкінці вся архітектура розгорнута навколо декількох цілей. Трьохрівнева архітектура розповсюджена тому що вона універсальна і вона популярна.

Закон Конвея

В квітні 1968 року Мелвін Конвей написав статтю під назвою «Як винаходять комітети?» в журналі Datamation. Конвей пропонує, щоб організації (і команди) будували системи, які були б копіями їх комунікаційної структури. Він використовує найширше визначення систем — від комп'ютерів до громадського

транспорту — і підкреслює, що його пропозиція застосовна до всіх і на всіх рівнях.

Його приклад показує його вік: «Контрактна дослідницька організація мала вісім осіб, які повинні були створити компілятор COBOL і ALGOL. Після деяких початкових оцінок складності та часу, п'ять людей були призначені на роботу COBOL і троє на роботу ALGOL. Отриманий компілятор COBOL працював у п'ять етапів, компілятор ALGOL — у три».

Одним словом, невеликі команди з невеликими комунікаційними мережами. Фактично, побудуйте команду навколо найменшої комунікаційної мережі, необхідної для виконання роботи.

«Необхідно знайти способи винагородити менеджерів з дизайну за те, що вони зберігають свої організації ощадливими та гнучкими». Він підкріплює це ідеями, які ми обговорювали в останніх кількох публікаціях.

По-перше, велика команда не обов'язково виконає роботу швидше, але менша команда дасть нам кращу систему. Конвей стверджує, що, хоча більшій кількості людей на роботі може бути «достатньо для чищення картоплі», це не працює для будівельних систем. Це тільки я, чи це звучить як попередник закону Брукса (який з'явився через 7 років)?

По-друге, Конвей прямо посилається на закон Паркінсона. Цей закон говорив, що менеджери хочуть більші команди, але закон Конвея передбачає, що нам краще мати маленькі. «Ймовірно, найбільшим спільним фактором, що стоїть за багатьма погано спроектованими системами, що існують зараз, була наявність проектної організації, яка потребує роботи».

Трьохшарова архітектура – гарний приклад цього закону в дії. Пояснюється це тим чому ж архітектура так розповсюджена, адже тепер об`єднання поліквaліфікованих людей в групи з ціллю скоротити передачу між ними.

Одна з цілей котрі випробовують особливі труднощі, складається в тому що мікрослужба не повинна використовувати бази спільно, коли служба хоче звернутись до даних котрі знаходяться в другій службі, то повинна запросити дані необхідні у першій службі. Таким чином можна відокремити що є спільним що прихованим.

Переваги мікрослужб багаточисельні і різнообразні. Незалежна поведінка розгортуваності відкриває нові моделі збільшенню масштабу і можливостей систем дозволяючи комбінації та спільного використання. Оскільки служби можуть працювати спільно та паралельно ви можете притягувати до вирішенню задач багатьох розробників, які можуть працювати без трудощів окремо, оскільки вони можуть сконцентруватись на своїй частина. Такий стиль програмування допоможе розгортанню платформи чи бази даних в правильній пропорції часу для виконання. Окрім цього архітектури на основі мікрослужб, мають можливість гнучкості для вирішення деяких задач в майбутньому. Однак слід зауважити, що ніяка перевага не безкоштовна. Існує багато підходів вирішення про них буде вказано пізніше.

Орієнтування на служби ввійшла в моду тому що компютери стали дешевше і їх стало більше. Замість цього розгортуваність системи на одному велитенському «mainframe» краще ніє на дешевих машинах. Архітекрутура орієнтована на служби була поштовхом для напрацювання для винаходу і пропрацбвання підходу по розробці додатків. Одні з головних труднощів в цьому всьому буда комунікація між комп`ютерами по мережі, оскільки вона є не миттєвою.

Користувацькі інтерфейси

Дуже часто користувачі зосереджують свою роботу по включенні мікросервісу на боці сервера, залишаючи користувацький інтерфейс як один монолітний блок. Якщо основне завдання швидке розгортання нових функцій то

вибір одного монолітного блоку буде помилкой, оскільки краще буде їх розбити на користувацькі інтерефейси.

Горочи про розмір якою повинна бути мікрослужба, напевно приставка мікро говорить за себе, але ж як виміряти розмір, довжиною кода. Взагалі це не має суті так як можна писати на Java і це займатиме 25 рядків коду чи Clojure в 12 строк, яке це не означає зо Clojure краще ніж Java, просто одна мова більше виразніша в написанні чим інша.

Найкраща думка в цьому направлені була сказана Крісом Річардсоном, котрий сказав що ціль мікрослужб – мати мінімально можливий розмір.

Коротко потрібно визначити що ж таке моноліт. Моноліт це одиниця розгортуваності, коли всі функції розгортуються разом. Існує три типи монолітних систем: однопроцесорна система, розподілений моноліт тв сторонні чорно-ящикові системи.

Однопроцесній моноліт найбільш розповсюджений коли весь код розгортуваності як один процес. Маючи декілька варіантів одного процесу по причині масштабування, але весь код упакований в один процес. Однопроцесорні системи можуть бути простими розподіленими системами для читання даних із бази даних і збереження їх.

Модульний моноліт є як варіант однопроцессорного моноліту. Який складається з окремих модулів які працюють незалежно але для розгортання повинні обєднатись.

Для більшості організацій модульний моноліт – чудовий вараінт. Якщо завдання повністю сформовано і є можливість паралельної роботи. Але є деякі нюанси при використанні доаного модульнного моноліту, те що стикаємось з тенденцієб декомпозиції яка визначається на равні коду.

Розподілений моноліт це система в котрій є багаточисельні служби. Але по якійсть причині система розгортажться одночасно. Розподілений моноліт виникаж

в середовищі де є недостатня приділялось уваги звязвності бізнес функціональності та приховування інформації.

Сторонні чорно-ящикові системи

Моноліти котрі можливо розкласти в гриницях міграції можна також розглядати сторонній контекст, в який входять платіжні системи, CRM – системи і HR- системи. Спільним є те що аплікація розроблена іншими людьми і вони не мають змоги його змінювати.

Моноліт однопроцесорний чи розподілений часто буває схильний до небезпеки сполученості. По мірі того як працюють все більше і більше людей вони починають заважити один одному на шляху. Різні розробники можуть змінювати один і той же фрагмент коду, але це призведе до конкуренції за доставку, яка виникне при плутанині хто чи володіє і які рішення приймає. Наявність моноліту не говорить що ви обов`язково зустрінете таку плутанину, але архітектура мікрослужб дає саме конкретні границі навколо яких можуть бути проведені лінії володіння, що дозволить легше вирішити проблему.

Безсерверний режим це парадигма для розгортання програм і послуг, що базуються на архітектурі яка керується подіями, вбудованій масштабності використовуючи підхід, орієнтований на застосуванню повністю абстрагуючи фізичне управління нею від розробників додатків. Усі безсерверні платформи повинні підготувати сервер до інфраструктури для завершення її реалізації.

Працює даний підхід так, коли інтерфейс надсилає HTTP-запит, безсерверна платформа перевіряє серверну програму що доступна для обслуговування даного запиту. Якщо серверна програма недоступна, тоді платформа розгортає його, що обслуговуватиме запит. Якщо таких запитів декілька то створюються декілька екземплярів серверної програми і яещо вхідні запити падають. Безсерверна програма зменшить масштаб до нуля.

Безсерверна концепція має багато переваг котрі забезпечують оптимальне використання, коли немає робочого навантаження на безсерверну платформу, тоді

зменшує мого до нуля. Це означає у таких випадках платформа блокуєьбся і дозволяє її використовувати для інших програм які цього потребують. За своєю суттю масштабована архітектура необхадна бути масштабованою за замовчуванням. Розробники безсерверних программ не пповинні визначати наскільки його потрібно масштабувати, адже додаток може бути максимально масштабовано по певного ліміту платформи, а якщо немаж навантаження то знижено до нуля.

Планування міграції від моноліту до мікросервісу

Мікрослужби не є самоціллю. Рішення встановити архітектура мікрослужб повинно бути прийнято на основі раціонального рішення. Одним з головним факторів на зміну повинно бути те ще ви не можете отримати тих результатів при наявній системі. Включаючи мікрослужби йде намагання отримати таким способом стан при якому значно зміниться те над чим ви зосередились і розтавити пріоритети зусиль.

Проблеми з відсутністю чіткого бачення відносно того чому використовувати мікрослужби, мають різний характер. Інколи потрібно багато інвестицій для збільшення числа людей чи кількості фінансів. Це усладнюється тим що потрібен певний час щоб бачити результат.

Є три ключових питання:

- Що ви надієтесь достигнути?
- Подумали чи ви про альтернативах використання мікрослужб.
- Як ви дізнаєтесь чи працює транзит.

Безсерверні обчислення та FaaS Спочатку термін «безсерверні обчислення» інтерпретувався інакше, ніж зараз. Передбачувана під ним технологія називалася сервер як послуга (Backend as a Service, BaaS) і призначалася для додатків, які частково або повністю залежать від сторонніх послуг з надання серверної логіки. Пізніше ця технологія стала називатися функція як послуга (Function as a Service,

FaaS), оскільки провайдери послуг безсерверних обчислень інтерпретують додатки як функції, викликаючи їх тільки за запиту.

Як було вказано вище, провайдери послуг безсерверних обчислень пропонують ізольовані контейнери для аплікацій. Контейнер керується подіями, тому активується тільки після окремої події.

Події – це конкретні зовнішні чинники котрі впливають, подібні до фізичних вимикачів. Візьмемо як приклад освітлення в будинку: події, що включають його, можуть відрізнятися. Світло може бути включене людиною, звичайним вимикачем, Але контейнери не обмежуються прийомом певних подій і викликом у них функцій; вони також дозволяють вашим функціям самим створювати події. У технічному сенсі в безсерверних обчисленнях контейнери функцій є і приймачами, і джерелами подій. Провайдери пропонують різні події, здатні запускати ваші функції. Список подій залежить від провайдера та реалізації, але часто їх роль відіграють HTTP-запити, вивантаження файлів у сховище, оновлення бази даних, події інтернету речей (Internet of Things, IoT) та безліч інших.

Ландшафт безсерверних обчислень має багато рухомих частин, а якщо враховувати обмеженість ресурсів, самий розповсюджений рішенням є створення свого API і використанням популярного фреймворку Node.js і налаштування бази даних.

Код типового API можна розділити на декілька рівнів: представлення, бізнес-логіка і дані.

Трирівнева архітектура – це шаблон архітектури клієнт-серверного програмного забезпечення, в якому користувальницький інтерфейс (подання), логіка роботи («бізнес-правила»), зберігання даних та доступ до них розробляються та підтримуються як незалежні модулі, найчастіше на окремих платформах.

Висновок

В цьому розділі я описала методології дослідження та вибір методів оптимізації збору даних.

Також було описано концепцію нової методики.

Розробка концепції нової методики при проектуванні інформаційної системи — це ключовий етап, на якому визначаються основні принципи, стратегії та підходи, які будуть використовуватися під час розробки та експлуатації системи.

Також були розглянуті архітектурні рішення для покращення процесу збору даних в інформаційній системі. Виявлені та описані їх переваги, обгрунтовано вибір.

РОЗДІЛ 4

РОЗРОБКА ТА РЕАЛІЗАЦІЯ СИСТЕМИ

4.1 Проектування та розробка нової архітектури системи обміну аналітичною інформацією.

При проектуванні системи крім програмного забезпечення важливо звернути увагу на інформаційне та технічне забезпечення.

Інформаційне забезпечення інформаційної системи обміну аналітичною інформацією для дистриб'юторів містить широкий спектр елементів та компонентів, які забезпечують збір, обробку, зберігання та надання доступу до аналітичних даних.

Інформаційне забезпечення системи обміну аналітичною інформацією дистрибюторів включає в себе наступні елементи:

- Дані. Дані є основою інформаційної системи. При розробці системи будуть використовуватись структуровані.
- Програмне забезпечення. Програмне забезпечення використовується для зберігання, обробки та аналізу даних.
- Апаратне забезпечення. Апаратне забезпечення забезпечує функціонування програмного забезпечення.

Програмне забезпечення в інформаційній системі обміну аналітичною інформацією дистрибюторів розділене на два основних типи:

- Платформа зберігання даних. Платформа зберігання даних використовується для зберігання даних з різних джерел.
- Система аналізу даних. Система аналізу даних використовується для обробки та аналізу даних.

Апаратне забезпечення в інформаційній системі обміну аналітичною інформацією дистрибюторів включає в себе наступні компоненти:

- Сервери. Сервери використовуються для зберігання даних і програмного забезпечення.
- Веб-сервери. Веб-сервери використовуються для надання доступу до системи користувачам.

Інформаційне забезпечення інформаційної системи обміну аналітичною інформацією дистрибюторів відповідає наступним вимогам:

Можливість зберігання великих обсягів даних різноманітних типів. Дистриб'ютори генерують великі обсяги даних у процесі своїх операцій. Система повинна бути здатна зберігати ці дані, щоб їх можна було аналізувати.

Можливість швидкого доступу до даних. Користувачі повинні мати можливість отримувати доступ до даних в режимі реального часу, щоб приймати обґрунтовані рішення.

Можливість аналізу даних. Система повинна бути здатна аналізувати дані, щоб виявляти тенденції, ідентифікувати можливості для зростання та покращувати обслуговування клієнтів.

Можливість забезпечення безпеки даних. Дані повинні бути захищені від несанкціонованого доступу, використання або модифікації.

Розробка інформаційного забезпечення інформаційної системи обміну аналітичною інформацією дистриб`юторів є важливою частиною процесу проектування системи. Воно дозволяє забезпечити ефективне зберігання, обробку та аналіз даних, які необхідні для успішного ведення бізнесу дистриб'юторами.

При проектуванні інформаційного забезпечення було враховано наступні фактори:

- Типи даних, які будуть зберігатися та оброблятися. Необхідно визначити, які типи даних будуть зберігатися та оброблятися в системі.
- Обсяг даних, які будуть зберігатися та оброблятися. Необхідно визначити, який обсяг даних буде зберігатися та оброблятися в системі.
- Частота доступу до даних. Необхідно визначити, наскільки часто буде доступ до даних.
- Безпека даних. Необхідно розробити заходи безпеки для захисту даних від несанкціонованого доступу, використання або модифікації.

Створення аналітичної системи обліку даних для виробника від дистриб'ютора включає в себе розробку архітектури, яка відповідає специфічним вимогам та потребам бізнес-процесів. Нижче описані ключові компоненти архітектури для такої системи:

Компонент для збору та інтеграції даних: розробка інтерфейсів для збору даних від різних джерел, таких як виробник та дистриб'ютор.

Розробка інтерфейсів для збору даних від різних джерел включає в себе створення механізмів, які дозволяють ефективно та надійно отримувати, обробляти та інтегрувати дані в аналітичну систему.

Для цього раніше я проаналізувала та обрала оптимальний спосіб представлення інформації в форматі CSV.

Необхідним процесом при проектуванні є створення API виробника - це широкий процес, і структура API може значно відрізнятися в залежності від специфікацій та потреб вашого проекту. Нижче наведено лістинг веб-сервера на Python, який використовує бібліотеку Flask для створення простого API.

```
from flask import Flask, request, jsonify

app = Flask(__name__)
```

```
    # Приклад даних про продукти (замініть це реальними даними)
    products_data = [
        {"id": 1, "name": "Product A", "price": 50.0},
        {"id": 2, "name": "Product B", "price": 75.0},
        {"id": 3, "name": "Product C", "price": 100.0},
    ]
    # API endpoint для отримання всіх продуктів
    @app.route('/api/products', methods=['GET'])
    def get_all_products():
        return jsonify(products_data)

    # API endpoint для отримання конкретного продукту за ID
    @app.route('/api/product/<int:product_id>', methods=['GET'])
    def get_product(product_id):
        product = next((p for p in products_data if p['id'] == product_id), None)
        if product:
            return jsonify(product)
        else:
            return jsonify({"message": "Product not found"}), 404
```

```
if __name__ == '__main__':
    app.run(debug=True)
```

Також необхідно відмітити важливість розробки інтеграційних модулів. Процес створення програмного забезпечення, яке дозволяє різним інформаційним системам взаємодіяти між собою. Інтеграційні модулі можуть використовуватися для різних цілей. В даному випадку вони застосовуються для ефективної взаємодії з API виробника та інших джерел даних, реалізації механізмів обробки помилок та забезпечення стійкості в разі недоступності джерела.

```
import requests


def fetch_data_from_manufacturer_api(api_url, api_key):
    headers = {
        'Authorization': f'Bearer {api_key}',
        'Content-Type': 'application/json',
    }

    try:
        response = requests.get(api_url, headers=headers)
        response.raise_for_status()  # Перевірка на наявність помилок
        data = response.json()
```

```
            return data
        except requests.exceptions.RequestException as e:
            print(f"Error fetching data from manufacturer
API: {e}")
            return None

    # Приклад використання:
    manufacturer_api_url = 'https://manufacturer-
api.example.com/data'
    manufacturer_api_key = 'your_api_key'
    manufacturer_data =
fetch_data_from_manufacturer_api(manufacturer_api_url,
manufacturer_api_key)
```

Також одним з основних компонентів, на які варто звернути увагу є використання стандартів інтеграції для ефективного обміну даними.

В даному випадку краще обрати SOAP, тому що він краще підходить для обміну великими обсягами даних.

Для використання стандарту інтеграції SOAP вам знадобиться використовувати спеціалізовані бібліотеки для обробки SOAP-запитів та відповідей. У Python для цього можна використовувати бібліотеку zeep. Нижче наведений код, який використовує zeep для взаємодії з SOAP-сервісом.

Спочатку потрібно встановити бібліотеку zeep, якщо вона не встановлена:

```
pip install zeep
```

Нижче наведений лістинг шаблон для використання бібліотеки zeep у Python для SOAP-запитів:

```
from zeep import Client
```

```
    wsdl_url = 'https://your-distributor-info-
system.com/soap?wsdl'


    # Створення клієнта для взаємодії з SOAP-сервісом
    client = Client(wsdl_url)


    soap_method = 'GetAnalyticalInformation'
    soap_params = {'param1': 'value1', 'param2': 'value2'}


    # Виклик SOAP-методу з використанням параметрів
    response = client.service[soap_method](**soap_params)


    # Обробка результату SOAP-запиту
    if response:
        print("SOAP Response:")
        print(response)
    else:
        print("SOAP Request Failed")
```

Механізми обробки та трансформації даних

Реалізація модулів для обробки та трансформації даних застосовується перед їх збереженням у хранилище. Використання ETL (Extract, Transform, Load) процесів сприяє ефективному управлінню потоками даних.

Нижче реалізовано ETL-процес для обробки та трансформації даних у форматі CSV перед їх збереженням у хранилище.

Модуль для вилучення даних (Extract):

```
# extract_module.py
import csv
def extract_data_from_csv(source_file):
    data = []
    with open(source_file, 'r', newline='') as file:
        reader = csv.DictReader(file)
        for row in reader:
            data.append(row)
    return data
```

Модуль для трансформації даних (Transform):

```
# transform_module.py
def transform_data(data, new_attribute_value):
    transformed_data = []

    for product in data:
        # Додаємо новий атрибут
        product['new_attribute'] = new_attribute_value

        transformed_data.append(product)

    return transformed_data
```

Модуль для завантаження даних (Load):

```
# load_module.py
import csv

def load_data_to_csv(data, destination_file):
    fieldnames = data[0].keys()

    with open(destination_file, 'w', newline='') as
file:
        writer = csv.DictWriter(file,
fieldnames=fieldnames)

        # Записуємо заголовок
        writer.writeheader()

        # Записуємо трансформовані дані
        writer.writerows(data)
```

Головний скрипт ETL:

```
# main_etl_script.py
from extract_module import extract_data_from_csv
from transform_module import transform_data
from load_module import load_data_to_csv

# Параметри ETL-процесу
source_file = 'source_data.csv'
```

```
    new_attribute_value = 'example_value'
    destination_file = 'transformed_data.csv'

    # Вилучення даних
    source_data = extract_data_from_csv(source_file)

    # Трансформація даних
    transformed_data = transform_data(source_data,
new_attribute_value)

    # Завантаження даних
    load_data_to_csv(transformed_data, destination_file)

    print("ETL process completed successfully.")
```

Лістинг коду всіх описаних нижче модулів системи буде представлено у Додатку А.

Хранилище даних

Створення хранилища даних, яке дозволяє зберігати великі обсяги даних та забезпечує швидкий доступ до них. Використання технологій, таких як реляційні або NoSQL бази даних, залежно від конкретних потреб.

Модуль аналітики та візуалізації

Розробка інструментів для аналізу даних та візуалізації результатів. Використання бізнес-інтелект платформ або власних інструментів для створення зручного інтерфейсу для користувачів.

Система безпеки та автентифікації

Впровадження механізмів безпеки для захисту конфіденційності та цілісності даних. Реалізація системи автентифікації та авторизації для контролю доступу до різних рівнів інформації.

Модуль моніторингу та звітності

Введення засобів моніторингу для виявлення можливих проблем та удосконалення продуктивності системи. Розробка модулів звітності, які дозволяють стежити за різними параметрами та представляти статистику користувачам.

Інтеграція з існуючими системами

Взаємодія з існуючими системами обліку та управління для забезпечення синхронізації даних та уникнення дублювання інформації.

Резервне копіювання та відновлення

Забезпечення системи регулярним резервним копіюванням та механізмами відновлення для уникнення втрати даних в разі непередбачуваних ситуацій.

Ці компоненти дозволяють створити архітектуру, яка відповідає вимогам аналітичної системи обліку даних для виробника від дистриб'ютора, забезпечуючи ефективність, надійність та зручний інтерфейс для користувачів.

4.2 Проведення тестів для оцінки ефективності та функціональності нової системи.

Написання тестів для інтеграційного модулю є важливою частиною розробки, оскільки вони дають гарантії, що система працює належним чином і відповідає вимогам користувачів. Тестування було проведено для покращення якості системи, зменшення ризиків та збільшення продуктивності.

Для проведення тестування основних модулів системи я обрала функціональне тестування.

Нижче описаний тест взаємодії з API виробника. Для написання тестів використала бібліотеку unittest у Python.

```
import unittest

from unittest.mock import patch

from your_integration_module import
fetch_data_from_manufacturer_api


class TestIntegrationModule(unittest.TestCase):


    @patch('requests.get')

    def
test_fetch_data_from_manufacturer_api_success(self,
mock_get):

        # Моделюємо успішну відповідь від API
виробника

        mock_get.return_value.status_code = 200

        mock_get.return_value.json.return_value =
[{'id': 1, 'name': 'Product A'}]


        # Викликаємо функцію і перевіряємо результат

        result =
fetch_data_from_manufacturer_api('http://example.com/api',
'your_api_key')
```

```
            self.assertEqual(result, [{'id': 1, 'name':
'Product A'}])


        @patch('requests.get')
        def
test_fetch_data_from_manufacturer_api_failure(self,
mock_get):
            # Моделюємо невдалий запит до API виробника
            mock_get.return_value.status_code = 404


            # Викликаємо функцію і перевіряємо результат
            result =
fetch_data_from_manufacturer_api('http://example.com/api',
'your_api_key')


            self.assertIsNone(result)


    if __name__ == '__main__':
        unittest.main()
```

Задача написання тестів для модулів обробки та трансформації даних важлива для переконання в тому, що ваші функції працюють коректно. Нижче наведені тести для описаних модулів обробки та трансформації даних:

Тести для extract_module.py:

```
import unittest
```

```
    from extract_module import extract_data_from_csv

    class TestExtractModule(unittest.TestCase):

        def test_extract_data_from_csv(self):
            # Створюємо тимчасовий CSV-файл для тесту
            temp_csv_file = 'test_data.csv'
            with open(temp_csv_file, 'w', newline='') as
file:

file.write('id,name,price\n1,ProductA,10.0\n2,ProductB,20.0
\n')

            # Тестуємо вилучення даних
            extracted_data =
extract_data_from_csv(temp_csv_file)

            # Перевіряємо, чи дані вилучено коректно
            self.assertEqual(extracted_data, [
                {'id': '1', 'name': 'ProductA', 'price':
'10.0'},
                {'id': '2', 'name': 'ProductB', 'price':
'20.0'},
            ])
```

```
if __name__ == '__main__':
    unittest.main()
```

Тести для transform_module.py:

```
import unittest
from transform_module import transform_data

class TestTransformModule(unittest.TestCase):

    def test_transform_data(self):
        # Тестові дані для трансформації
        input_data = [
            {'id': '1', 'name': 'ProductA', 'price': '10.0'},
            {'id': '2', 'name': 'ProductB', 'price': '20.0'},
        ]

        # Тестуємо трансформацію даних
        transformed_data = transform_data(input_data, new_attribute_value='example_value')

        # Перевіряємо, чи дані трансформовані коректно
        self.assertEqual(transformed_data, [
```

```
                {'id': '1', 'name': 'ProductA', 'price':
'10.0', 'new_attribute': 'example_value'},
                {'id': '2', 'name': 'ProductB', 'price':
'20.0', 'new_attribute': 'example_value'},
            ])

    if __name__ == '__main__':
        unittest.main()
```

Тести для load_module.py:

```
    import unittest
    import csv
    from io import StringIO
    from load_module import load_data_to_csv

    class TestLoadModule(unittest.TestCase):

        def test_load_data_to_csv(self):
            # Тестові дані для завантаження
            input_data = [
                {'id': '1', 'name': 'ProductA', 'price':
'10.0', 'new_attribute': 'example_value'},
                {'id': '2', 'name': 'ProductB', 'price':
'20.0', 'new_attribute': 'example_value'},
            ]
```

```
            # Створюємо тимчасовий CSV-файл для тесту
            temp_csv_file = 'test_output.csv'

            # Тестуємо завантаження даних
            load_data_to_csv(input_data, temp_csv_file)

            # Читаємо вміст тимчасового CSV-файлу
            with open(temp_csv_file, 'r', newline='') as
file:
                loaded_data = list(csv.DictReader(file))

            # Перевіряємо, чи дані були завантажені
коректно
            self.assertEqual(loaded_data, input_data)

    if __name__ == '__main__':
        unittest.main()
```

Висновок

В процесі написання розділу було виконано проектування та розробку нової архітектури системи обміну аналітичною інформацією.

Наведений детальний опис кожного модулю системи. Також був створений лістиг коду для кожного компоненту.

В результаті було проведено тести для оцінки ефективності та функціональності нової системи.

ВИСНОВКИ

У ході дослідження, проведеного в рамках даної дипломної роботи, було розроблено методику оптимізації збору даних в інформаційній системі обміну аналітичної інформації від дистриб'юторів.

Методика оптимізації може бути адаптована до конкретних потреб.

Впровадження оптимізації в інформаційній системі обміну аналітичної інформації від дистриб'юторів може призвести до наступних переваг:

Зменшення витрат. Оптимізація може допомогти скоротити витрати на збір даних.

Покращення ефективності. Оптимізація може допомогти скоротити час і зусилля, необхідні для збору даних.

Покращення якості даних. Оптимізація може допомогти зменшити кількість помилок у даних.

Результати дослідження, проведеного в рамках даної дипломної роботи, можуть бути використані для розробки та впровадження методів оптимізації збору даних від дистриб'юторів в інших організаціях.

Для отримання максимальних переваг від оптимізації збору даних від дистриб'юторів рекомендується:

Залучити до процесу розробки та впровадження оптимізації всіх зацікавлених сторін, таких як керівники організації, менеджери інформаційних систем, дистриб'ютори та інші.

Планувати впровадження оптимізації на тривалий термін, щоб встигнути провести всі необхідні зміни в інформаційній системі та бізнес-процесах.

Оцінювати ефективність оптимізації на регулярній основі, щоб виявити можливі проблеми і внести необхідні корективи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. "The Data Warehouse Toolkit" by Ralph Kimball and Margy Ross
2. "Data Science for Business" by Foster Provost and Tom Fawcett
3. "Designing Data-Intensive Applications" by Martin Kleppmann
4. Journal of Data and Information Quality
5. "Data Integration Blueprint and Modeling" by Anthony David Giordano:
6. "Data Engineering Teams" by Mike Barlow
7. ISO 8000
8. ISO 27001
9. ISO 22745
10. Додонов А.Г., Ланде Д.В., Путятін В.Г. Комп'ютерні мережі та аналітичні дослідження. – К.: ІПРІ НАН України, 2014. – 486 с. ISBN 978-966-02-7422-8
11. Ланде Д.В. Новітні підходи й технології інформаційно-аналітичної підтримки прийняття рішень. // Національна безпека: український вимір: щокв. наук. зб. / Рада нац. безпеки і оборони України, Ін-т пробл. нац. безпеки; - К., 2008. - Вип. 1-2 (20-21). - С. 87-105.
12. O'Brien, James A. ve George M. Marakas (2007); Enterprise Information Systems, New York: The McGraw-Hill.
13. Ілля Волков, Ілля Галахов. Архітектура сучасної інформаційно-аналітичної системи, Директор ІС, 2002, № 3.

14. Григорова А. А., Чорний С. Г. Формування сучасної інформаційно-аналітичної системи для підтримки прийняття рішень. ААЕКС, № 2(12), 2003

15. Положення про дипломні роботи (проекти) випускників Національного Авіаційного Університету Київ 2017. -72 с.

ДОДАТКИ

Додаток А. Лістинг коду

1. Створення веб-серверу за допомогою Flask з використанням SQLAlchemy для взаємодії з базою даних

```
from flask import Flask, request, jsonify
from flask_sqlalchemy import SQLAlchemy

app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///data.db'
db = SQLAlchemy(app)

class DistributorData(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    distributor_name = db.Column(db.String(50))
    sales_data = db.Column(db.Float)
    inventory_data = db.Column(db.Integer)
    # Додайте інші поля за необхідності

# Маршрут для додавання даних від дистриб'ютора
@app.route('/add_data', methods=['POST'])
def add_data():
```

```
    data = request.get_json()

    new_data = DistributorData(
        distributor_name=data['distributor_name'],
        sales_data=data['sales_data'],
        inventory_data=data['inventory_data']
    )

    db.session.add(new_data)
    db.session.commit()

    return jsonify({'message': 'Data added successfully'}),
201

# Маршрут для отримання всіх даних від дистриб'юторів
@app.route('/get_all_data', methods=['GET'])
def get_all_data():
    data = DistributorData.query.all()

    data_list = []
    for item in data:
        data_list.append({
            'distributor_name': item.distributor_name,
            'sales_data': item.sales_data,
            'inventory_data': item.inventory_data
        })

    return jsonify({'data': data_list}), 200

if __name__ == '__main__':
```

```
    db.create_all()
    app.run(debug=True)
```

2. Вихідний код, який додає базову аутентифікацію та авторизацію за допомогою токенів та валідацію даних

```
from flask import Flask, request, jsonify
from flask_sqlalchemy import SQLAlchemy
from flask_bcrypt import Bcrypt
import jwt
from datetime import datetime, timedelta
from functools import wraps

app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///data.db'
app.config['SECRET_KEY'] = 'your_secret_key'  # Важливо зберігати секретний ключ в безпечному місці
db = SQLAlchemy(app)
bcrypt = Bcrypt(app)

# Модель користувача
class User(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(50), unique=True, nullable=False)
    password = db.Column(db.String(100), nullable=False)

# Маршрут для реєстрації нового користувача
@app.route('/register', methods=['POST'])
def register():
```

```
    data = request.get_json()

    hashed_password =
bcrypt.generate_password_hash(data['password']).decode('utf
-8')

    new_user = User(
        username=data['username'],
        password=hashed_password
    )

    db.session.add(new_user)
    db.session.commit()

    return jsonify({'message': 'User registered
successfully'}), 201

# Маршрут для входу користувача та отримання токена
@app.route('/login', methods=['POST'])
def login():
    data = request.get_json()

    user =
User.query.filter_by(username=data['username']).first()

    if user and bcrypt.check_password_hash(user.password,
data['password']):
        token = jwt.encode({'user_id': user.id, 'exp':
datetime.utcnow() + timedelta(minutes=30)},
app.config['SECRET_KEY'])
        return jsonify({'token': token.decode('utf-8')})
```

```
    else:
        return jsonify({'message': 'Invalid credentials'}),
401

# Декоратор для перевірки аутентифікації користувача
def token_required(f):
    @wraps(f)
    def decorated(*args, **kwargs):
        token = request.headers.get('Authorization')

        if not token:
            return jsonify({'message': 'Token is
missing'}), 401

        try:
            data = jwt.decode(token,
app.config['SECRET_KEY'])
            current_user = User.query.get(data['user_id'])
        except:
            return jsonify({'message': 'Token is
invalid'}), 401

        return f(current_user, *args, **kwargs)

    return decorated

# Маршрут для додавання даних від дистриб'ютора (захищений
токеном)
@app.route('/add_data', methods=['POST'])
@token_required
def add_data(current_user):
```

```
    if current_user:
        data = request.get_json()

        new_data = DistributorData(
            distributor_name=current_user.username,  # Використовуємо ім'я користувача замість переданого з JSON
            sales_data=data['sales_data'],
            inventory_data=data['inventory_data']

        )

        db.session.add(new_data)
        db.session.commit()

        return jsonify({'message': 'Data added successfully'}), 201
    else:
        return jsonify({'message': 'User not authorized'}), 401
```

3. Модулю аналітики та візуалізації інформаційній системі обміну аналітичної інформації від дистриб`юторів

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

class AnalyticsModule:
    def __init__(self, data):
```

```
        self.data = data

    def perform_analysis(self):

        # обчислення ключових показників, групування даних.
        analysis_result =
self.data.groupby('Distributor')['Sales'].sum()

        return analysis_result

class VisualizationModule:
    def __init__(self, analysis_result):
        self.analysis_result = analysis_result

    def generate_visualization(self):
        plt.figure(figsize=(10, 6))
        sns.barplot(x=self.analysis_result.index,
y=self.analysis_result.values)
        plt.title('Sales by Distributor')
        plt.xlabel('Distributor')
        plt.ylabel('Sales')
        plt.show()

# Припустимо, у вас є DataFrame `df` з даними:
```

```
# df = pd.read_csv('your_data.csv')

# Ініціалізуємо та використовуємо модуль аналітики
analytics_module = AnalyticsModule(data=df)
analysis_result = analytics_module.perform_analysis()

# Ініціалізуємо та використовуємо модуль візуалізації
visualization_module = VisualizationModule(analysis_result)
visualization_module.generate_visualization()
```

4. Система безпеки та автентифікації в інформаційній системі обміну аналітичної інформації від дистриб`юторів

```
class User:
    def __init__(self, username, password):
        self.username = username
        self.password = password

class AuthenticationService:
    def __init__(self):
        self.users = []  # Замість цього слід використовувати базу даних або інший механізм зберігання користувачів

    def register_user(self, username, password):
```

```
        # Перевірка, чи користувача з таким ім'ям користувача ще не існує
        if not any(user.username == username for user in self.users):
            new_user = User(username, password)
            self.users.append(new_user)
            return new_user
        else:
            raise ValueError("Користувач з таким ім'ям користувача вже існує.")

    def authenticate_user(self, username, password):
        # Перевірка автентифікації
        user = next((user for user in self.users if user.username == username and user.password == password), None)
        if user:
            return user
        else:
            raise ValueError("Невірне ім'я користувача або пароль.")

class SecuritySystem:
    def __init__(self):
        self.auth_service = AuthenticationService()
```

```
    def secure_operation(self, username, password,
operation_data):
        # Аутентифікація користувача
        authenticated_user =
self.auth_service.authenticate_user(username, password)

        # Додаткові дії після успішної аутентифікації
        print(f"Користувач {authenticated_user.username}
успішно аутентифікований.")

if __name__ == "__main__":
    security_system = SecuritySystem()

    # Реєстрація нового користувача
    new_user =
security_system.auth_service.register_user("john_doe",
"password123")
    # Аутентифікація та виконання безпечної операції
    security_system.secure_operation("john_doe",
"password123", operation_data="some_data")
```

5. Модуль моніторингу та звітності в інформаційній системі обміну аналітичної інформації від дистриб`юторів

```
import time
from datetime import datetime
```

```
class MonitoringModule:
    def __init__(self):
        self.metrics = {}  # Словник для зберігання метрик

    def collect_metric(self, metric_name, value):
        # Збір метрики
        timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
        if metric_name not in self.metrics:
            self.metrics[metric_name] = []
        self.metrics[metric_name].append({"timestamp": timestamp, "value": value})

    def generate_report(self, start_time, end_time):
        # Генерація звіту на основі зібраних метрик у вказаний період
        report = {"start_time": start_time, "end_time": end_time, "metrics": self.metrics}
        return report

class ReportingModule:
    def __init__(self, monitoring_module):
        self.monitoring_module = monitoring_module
```

```
    def generate_daily_report(self):
        # Здійснює генерацію щоденного звіту
        today = datetime.now().strftime("%Y-%m-%d")
        start_time = f"{today} 00:00:00"
        end_time = f"{today} 23:59:59"
        daily_report =
self.monitoring_module.generate_report(start_time,
end_time)
        return daily_report

if __name__ == "__main__":
    monitoring_module = MonitoringModule()

    # Симуляція збору метрик
    monitoring_module.collect_metric("user_registration",
10)
    monitoring_module.collect_metric("user_login", 200)

    reporting_module = ReportingModule(monitoring_module)

    # Генерація щоденного звіту
    daily_report = reporting_module.generate_daily_report()
    print("Daily Report:")
```

```
    print(daily_report)
```

6. Модуль інтеграції з існуючими системами

```
import requests


class IntegrationModule:
    def __init__(self, api_key):
        self.api_key = api_key
        self.base_url = "https://api.example.com"  #


    def fetch_data_from_external_system(self, endpoint, params=None):
        # Здійснює HTTP-запит для отримання даних з іншої системи
        headers = {"Authorization": f"Bearer {self.api_key}"}
        response = requests.get(f"{self.base_url}/{endpoint}", params=params, headers=headers)


        if response.status_code == 200:
            return response.json()
        else:
            raise ValueError(f"Failed to fetch data. Status code: {response.status_code}, Response: {response.text}")
```

```
    def send_data_to_external_system(self, endpoint, data):
        # Здійснює HTTP-запит для відправлення даних до іншої системи
        headers = {"Authorization": f"Bearer {self.api_key}", "Content-Type": "application/json"}
        response = requests.post(f"{self.base_url}/{endpoint}", json=data, headers=headers)

        if response.status_code == 200:
            return response.json()
        else:
            raise ValueError(f"Failed to send data. Status code: {response.status_code}, Response: {response.text}")

if __name__ == "__main__":
    api_key = "your_api_key"  # Замініть це на реальний API ключ

    integration_module = IntegrationModule(api_key)

    # отримання даних з іншої системи
    external_data = integration_module.fetch_data_from_external_system("external_data_endpoint", params={"param1": "value1"})
```

```
    # відправлення даних до іншої системи
    data_to_send = {"key1": "value1", "key2": "value2"}
    response =
integration_module.send_data_to_external_system("send_data_
endpoint", data_to_send)

    print("Received data from external system:",
external_data)
    print("Response from sending data:", response)
```

7. Модуль резервного копіювання та відновлення

```
import shutil
import os
import zipfile
from datetime import datetime

class BackupModule:
    def __init__(self, backup_folder):
        self.backup_folder = backup_folder

    def create_backup(self, data_folder):
        # Створення резервної копії
        backup_timestamp =
datetime.now().strftime("%Y%m%d_%H%M%S")
```

```
        backup_filename = f"backup_{backup_timestamp}.zip"
        backup_path = os.path.join(self.backup_folder,
backup_filename)

        with zipfile.ZipFile(backup_path, 'w') as
backup_zip:
            for root, _, files in os.walk(data_folder):
                for file in files:
                    file_path = os.path.join(root, file)
                    arcname = os.path.relpath(file_path,
data_folder)
                    backup_zip.write(file_path,
arcname=arcname)

        print(f"Backup created: {backup_path}")
        return backup_path

    def restore_backup(self, backup_path, restore_folder):
        # Відновлення даних з резервної копії
        with zipfile.ZipFile(backup_path, 'r') as
backup_zip:
            backup_zip.extractall(restore_folder)

        print(f"Data restored to: {restore_folder}")
```

```
if __name__ == "__main__":
    backup_folder = "path/to/backup/folder"
    data_folder = "path/to/data/folder"

    backup_module = BackupModule(backup_folder)

    # Створення резервної копії
    backup_path = backup_module.create_backup(data_folder)

    # Відновлення даних з резервної копії
    restore_folder = "path/to/restore/folder
    backup_module.restore_backup(backup_path,
restore_folder)
```