Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет _____Комп'ютерних наук_____
(повна назва)

Кафедра _____Програмної інженерії_____
(повна назва)

АТЕСТАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти – другий (магістерський)

Дослідження методів аналізу даних в data set для прийняття рішень з урахуванням вподобань
(тема)

Виконав: студент 2 курсу, групи ІПЗм-18-4
спеціальності 121- Інженерія програмного забезпечення
(код і повна назва спеціальності)
Освітньо-наукової програми
Інженерія програмного забезпечення

Хілюк Є.В.

(прізвище, ініціали)
Керівник _к.т.н., проф. Дудар З.В._____
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри, проф. _____ З.В.Дудар

2020 р.

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Факультет_____Комп'ютерних наук_____

Кафедра_____Програмної інженерії_____

Рівень вищої освіти_____другий (магістерський)_____

Спеціальність _____121- Інженерія програмного забезпечення_____
Тип програми _____освітньо-наукова програма_____
Освітня програма _____Інженерія програмного забезпечення_____

ЗАТВЕРДЖУЮ:
Зав. кафедри_____
(підпис)
«____»_____ 20 ___ р.

ЗАВДАННЯ

НА АТЕСТАЦІЙНУ РОБОТУ

студентові _____Хілюку Єгору Володимировичу_____
(прізвище, ім'я, по батькові)

1. Тема роботи _____Дослідження методів аналізу даних в data set для прийняття рішень з урахуванням вподобань_____
затверджена наказом по університету від "___" _____20 ___ р № _____
заповнюється вручну після отримання наказу
2. Термін подання студентом роботи до екзаменаційної комісії _____2020 р.
3. Вихідні дані до роботи *методи аналізу даних, пояснювальна записка, програмна система для обробки великих обсягів даних. Використовувати ОС Windows, середовище об'єктно-орієнтованого проектування*
4. Перелік питань, що потрібно опрацювати в роботі *мета роботи, аналіз проблемної галузі і постановка задачі, огляд методів аналізу великих обсягів даних, реалізація програмної системи*_____
_____

5. Консультанти розділів роботи

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Спецчастина	проф. к. т. н. Дудар З.В.		

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка *
1.	Аналіз предметної галузі	27 січня 2020 р.	
2.	Огляд існуючих методів	10 лютого 2020 р.	
3.	Підготовка пояснювальної записки	23 березня 2020 р.	
4.	Спецчастина	06 квітня 2020 р.	
5.	Підготовка презентації та доповіді	14 квітня 2020 р.	
6.	Попередній захист	20 квітня 2020 р.	
7.	Нормоконтроль, рецензування	05 травня 2020 р.	
8.	Занесення диплома в електронний архів	05 травня 2020 р.	
9.	Допуск до захисту у зав. кафедри	05 травня 2020 р.	
* заповнюється вручну після виконання чергового пункту			

Дата видачі завдання - - *2020р*.

Студент гр.ІПЗм-18-4_____ Хілюк Є.В.

Керівник роботи _____ к.т.н., проф. Дудар З.В.

РЕФЕРАТ

Звіт з науково-дослідної практики магістрів: 66 с., 30 рис., 26 джерела.

ПРОГРАМНА СИСТЕМА, ПРОГРАМНИЙ ПРОДУКТ, BIG DATA, DATA SET, АНАЛІЗ ВЕЛИКИХ ОБСЯГІВ ДАНИХ

Об’єкт – методи аналізу великих обсягів даних, програми обробки даних, інструменти аналізу у big data.

Мета роботи – дослідити основні методи, технології та алгоритми для аналізу великих обсягів даних з метою ефективного застосування в системах обробки даних, розробити програму на основі проведених досліджень.

В роботі проведений аналіз предметної галузі, досліджені основні методи та фреймворки аналізу великих обсягів даних, проведений аналіз актуальних технологій, розроблена програма із використанням знань, набутих під час дослідження.

ABSTRACT

Master final work contains: 66 p., 30 pic., 26 sources.

SOFTWARE SYSTEM, SOFTWARE PRODUCT, BIG DATA, DATA SET, ANALYSIS OF LARGE AMOUNTS OF DATA

The object of the development are big data analysis methods, data processing programs, big data analysis tools.

The aim of the task is to investigate the basic methods, technologies and algorithms for the analysis of large volumes of data for the purpose of effective application in data processing systems, to develop a program on the basis of the conducted researches..

As a result, the analysis of the subject area was carried out, the basic methods and frameworks for the analysis of large volumes of data were investigated, the analysis of the current technologies was conducted, a program with the use of knowledge gained during the research was developed.

ЗМІСТ

ВСТУП

Функціонувати в нормальному суспільстві стає все важче і важче без наявності якихось великих даних у деяких сферах нашого життя. Багато змін, пов'язаних із зростанням обсягів даних, є настільки тривіальними, що ми їх ледве помічаємо. Не буде перебільшенням сказати, що термін "Big Data" є скрізь сьогодні.

Аналіз великих даних[1] - це процес вивчення великих наборів даних, задля подальшого оптимального та зрозумілого використання у окремих областях знань. Сама по собі, галузь аналітики та обробки даних дуже величезна і зростає з кожним днем.

Аналітика "Big Data" - це справді революція в галузі інформаційних технологій. Використання аналітики даних компаніями посилюється з кожним роком. Основна увага компаній - на клієнтах. Дані компанії можуть знаходитися в абсолютно різних секторах, таких як соціальні мережі, геолокаційні сервіси, з яких виокремлюються деякі прикладні задачі (проектування транспорту, аналіз суспільної думки, виявлення та координація надзвичайних ситуацій), медичні сервіси, які можуть запобігати розвитку хвороб на ранніх стадіях, тощо.

Інструменти та методи аналізу великих даних користуються великим попитом завдяки використанню "Big Data" у бізнесі. Організації можуть знайти нові можливості та отримати нову інформацію для ефективного ведення свого бізнесу. Ці інструменти допомагають надавати змістовну інформацію для прийняття кращих бізнес-рішень.

Компанії можуть вдосконалювати свої стратегії, пам'ятаючи про орієнтацію на споживача. Аналітика великих даних ефективно допомагає операціям стати більш швидкими, оптимальними та максимально корисними для кінцевого користувача. Це допомагає покращити прибуток компанії в найбільш короткі терміни.

Засоби аналізу великих даних, такі як Hadoop, допомагають знизити витрати на зберігання. Це ще більше підвищує ефективність бізнесу. За допомогою новітніх інструментів аналітики, аналіз даних стає простішим і швидшим. Це, у свою чергу, призводить до швидшого прийняття рішень, економлячи час та енергію.

Завдяки величезному інтересу та інвестиціям у технології, “Big Data” користуються величезним попитом професіонали, що володіють навичками аналізу великих даних.[1] Організації сплачують великі гроші та соціальні пакети для кваліфікованих фахівців. IT-професіонали, такі як інженери та адміністратори даних, можуть вивчити інструменти аналітики та обробки даних для перспективної кар'єри.

Важливість аналітики великих даних призводить до інтенсивної конкуренції та збільшення попиту на фахівців у цій сфері. Обробка та аналітика даних - це сфера з величезним потенціалом. Аналіз та обробка даних допомагає аналізувати ланцюжок вартості бізнесу та отримувати уявлення про його подальший розвиток. Використання аналітики може підвищити галузеві знання окремих робітників підприємств. Експерти з аналізу даних надають організаціям можливість дізнатися про додаткові можливості для бізнесу.

Відомо, що останнім часом все більша кількість компаній шукає спеціалістів, які здатні створити продукт, який зможе трансформувати їх згруповані у файли дані, в деяку візуально зрозумілу та оптимально відображену таблицю з подальшою можливістю її аналітики. Це викликано тим, що з кожним роком інформації стає все більше, а структурувати її дедалі важче. Все більше число людей воліють обирати ті чи інші обхідні шляхи для аналізу великих об’ємів даних. В таких ситуаціях було б дуже доцільно скористуватися відповідною програмою для обробки згрупованого якимись правилами тексту у відповідну структуровану інформацію.

В якості основної технології при розробці програмного забезпечення для обробки та аналітики великих обсягів даних необхідно користуватися одразу декількома технологіями та методами, доцільно поєднуючи їх між собою. Є велика кількість інструментів для роботи з даними. Кожен з них має свої недоліки і

переваги. Необхідно досягти гнучкості, швидкості та високої продуктивності обробки даних, адже вони можуть бути дуже великих розмірів та різних структур.

Все вищесказане визначило актуальність теми роботи - використання найбільш придатного методу в задачах обробки інформації.

Отже, тема та засоби для виконання роботи є актуальними. Метою роботи є проектування та створення програмної системи, яка дозволяє обробити та аналізувати текстову інформацію у зрозумілий та структурований вигляд.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Історія та розвиток Big Data

Деякі спеціалісти з управління даними (Big Data) описують великі дані як "величезні, непосильні та неконтрольовані обсяги інформації". У 1663 році Джон Граун також розбирався з "величезною кількістю інформації", вивчаючи бубонну чуму, яка в той час розорювала Європу. Грант використовував статистику і відомий тим, що він перший, хто застосував аналіз статистичних даних. На початку 1800-х років область статистики розширилася, включаючи збір та аналіз даних.[3]

Еволюція Big Data включає ряд попередніх кроків для її заснування. "Big Data" - відносний термін залежно від того, хто обговорює. Великі дані для Amazon або Google дуже відрізняються від великих даних для середньої страхової організації, але не менш «великі» у їх свідомості.

Сучасна концепція Big Data передбачала розробку комп'ютерів, смартфонів, а також «інтернет речей» обладнання для надання даних. Кредитні картки також відігравали певну роль, надаючи все більший обсяг даних, і, безумовно, соціальні медіа змінили характер обсягів даних на нові. Еволюція сучасних технологій переплітається з еволюцією Big Data.

Дані стали проблемою для бюро перепису населення США у 1880 році. Вони підрахували, що обробка та аналіз даних, зібраних під час перепису 1880 р., потребуватиме вісім років, і передбачили, що дані, які будуть переписувати у 1890 р., потребуватимуть більше 10 років. На щастя, у 1881 році молодий чоловік, який працював у бюро, на ім’я Герман Холлеріт, створив машину для розміщення даних Холлеріта.[1] Його винахід базувався на перфокартах, призначених для управління візерунками, витканими механічними ткацькими верстатами. Його машина складання таблиць скоротила десять років праці до трьох місяців.

У 1927 р. Фріц Флеймер, австрійсько-німецький інженер, розробив засіб магнітного зберігання інформації на магнітофонній стрічці. Флеймер розробив

метод приклеювання металевих смужок до сигаретних паперів і вирішив, що він може використовувати цю техніку для створення магнітної смужки.

Під час Другої світової війни (точніше 1943 р.) англійці винайшли машину, яка сканувала повідомлення, перехоплені у німців. Машина отримала назву Colossus, і сканувала 5000 символів в секунду, скорочуючи навантаження з тижнів на просто години. Colossus був першим процесором даних. Через два роки, в 1945 році, Джон Фон Нойман опублікував документ про електронну дискретну змінну автоматичну машину (EDVAC), перший «документально обговорений» пристрій щодо зберігання програм, і заклав основу комп'ютерної архітектури сьогодні.

Кажуть, що ці поєднані події спонукали до "формального" створення АНБ США (Агенції національної безпеки) президентом Труменом у 1952 році. Персоналу АНБ було доручено розшифрувати повідомлення, перехоплені під час холодної війни. Комп'ютери цього часу еволюціонували до того, що вони могли збирати та обробляти дані, працюючи незалежно та автоматично.

У 1990-х роках спостерігався неймовірний ріст популярності інтернету, а персональні комп'ютери стали потужнішими та гнучкішими. Зростання поширення інтернету базувалося на зусиллях Тіма Бернерса-Лі.

У 2005 році Big Data, яка використовувалася без імені, була названа Роджером Мугаласом. Він мав на увазі великий набір даних, якими на той час було майже неможливо керувати та обробляти, використовуючи наявні традиційні інструменти бізнес-аналітики. Крім того, Hadoop, який може обробляти Big Data, був створений у 2005 році.[1] Hadoop базувався на основі програмного забезпечення під відкритим кодом під назвою Nutch і був об'єднаний з Google MapReduce. Hadoop - це програмне забезпечення з відкритим кодом, яке може обробляти структуровані та неструктуровані дані майже з усіх цифрових джерел.

Наразі магнітне зберігання є одним з найменш дорогих методів зберігання даних. Концепція смугастих магнітних ліній Фріца Флейнера 1927 р. була адаптована до різноманітних форматів, починаючи від магнітної стрічки, магнітних барабанів, дискет та жорстких дисків. Магнітне зберігання описує будь-яке

зберігання даних на основі намагніченого середовища. Він використовує дві магнітні полярності, північну та південну.

Хмарне зберігання даних стало досить популярним в останні роки. Перша справжня хмара з’явилася в 1983 році, коли CompuServe запропонував своїм клієнтам 128 КБ простору для особистого та приватного зберігання. У 1999 році Salesforce запропонував із свого веб-сайту програмне забезпечення як послугу (SaaS). Технічні вдосконалення в інтернеті в поєднанні з падінням витрат на зберігання даних зробили більш економічно вигідним для бізнесу та приватних осіб використання хмарного серededовища для цілей зберігання даних. Це економить організаціям на закупівлі, обслуговуванні та, врешті-решт, заміні їх комп'ютерної системи. Хмара забезпечує майже нескінченну кількість масштабованості та є доступною в будь-якому місці та в будь-який час.

У 2017 році було опитано 2800 досвідчених фахівців, які працювали з Business Intelligence, і вони прогнозували, що виявлення даних та візуалізація стануть важливою тенденцією. Візуалізація даних - це форма візуального спілкування. Вона описує інформацію, яка була переведена у схематичний формат, та включає зміни, змінні та коливання. Людський мозок може дуже ефективно обробляти зорові структури.

Безумовно, коротка історія великих даних не така коротка, як здається. Великі дані лише продовжуватимуть зростати, а разом з цим будуть розвиватися нові технології для кращого збору, зберігання та аналізу даних.

1.2 Огляд існуючих програм з обробки великих даних

Як було зазначено вище обрана тема є актуальною на сьогодні і на близьке майбутнє. Щоб мати більше уявлення про роботу систем з обробки та аналізу великих даних було виділено декілька програмних продуктів для розгляду.

OpenRefine (раніше Google Refine) є потужним інструментом для роботи з брудними даними: їх очищення; перетворення з одного формату в інший; розширення його веб-сервісами та зовнішніми даними.

OpenRefine завжди зберігає ваші дані приватними на власному комп'ютері, доки ви не захочете ділитися ними з іншими[2] (це працює за допомогою невеликого сервера, і ви використовуєте веб-браузер для взаємодії з ним).

OpenRefine надає можливість :

- швидко отримати уявлення про великі масиви даних, що ви маєте;

- переробляти дані в потрібний формат, робити базові розрахунки;

- фільтрувати і об'єднувати дані;

- знаходити помилки і несподіванки - наприклад, занадто великі цифри, слова замість чисел, порожні значення;

- автоматично знаходити потенційні помилки і невідповідності в назвах, дозволяючи приводити записи до єдиного вигляду (кластеризація текстових записів);

- створювати «макроси», автоматизуючи обробку даних;

- велика перевага OR - його простота. Програма не складніше, ніж, наприклад, excel.

Приклад інтерфейсу OpenRefine представлено на рисунку 1.1.

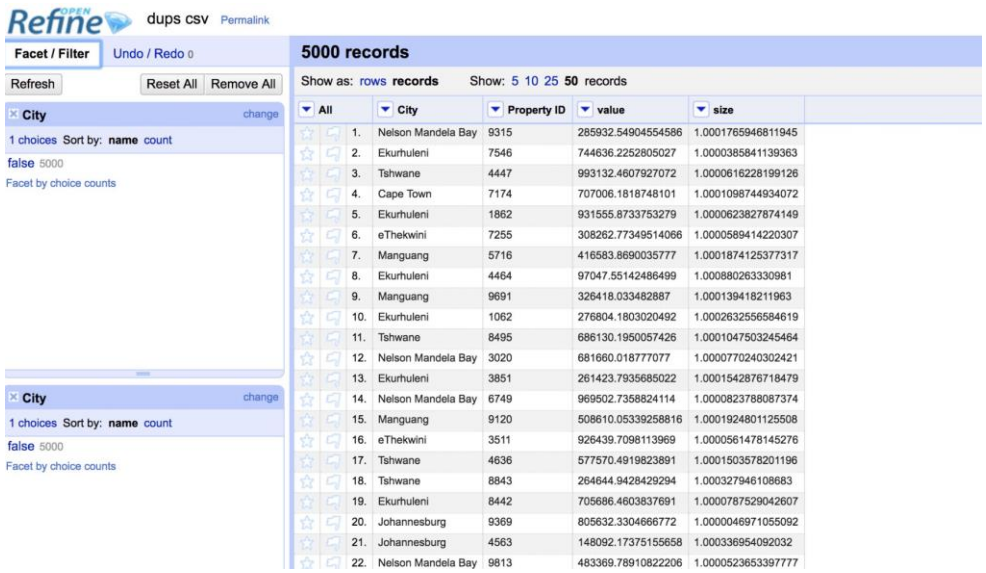

Рисунок 1.1 – Робота додатку OpenRefine

З основних недоліків можна виділити недолік кастомізації вхідних даних та подальшого використання за власними правилами для таблиць.

Orange - це візуалізація даних, машинне навчання та інструментарій вибору даних з відкритим кодом. У ньому представлено візуальне програмування, призначене для дослідницького аналізу даних та інтерактивної візуалізації даних.

Orange - це програмний пакет з відкритим кодом, випущений в рамках GPL. Версії до 3.0 включають основні компоненти в C++ із обгортками в Python, доступні на GitHub[2]. Починаючи з версії 3.0, Orange використовує поширені бібліотеки з відкритим кодом Python для наукових обчислень, такі як numpy, scipy та scikit-learn, в той час як його графічний інтерфейс користувача працює в рамках платформи Qt між платформами.

Orange має наступну функціональність:

- віджети для введення даних, фільтрації даних, вибірки, імпутації, маніпулювання та вибору функції;

- віджети для загальної візуалізації (графічна скринька, гістограми, графік розсіювання) та багатоваріантну візуалізацію (мозаїчне відображення);

- перехресна валідація, процедури на основі вибірки, оцінка надійності та оцінка методів прогнозування;

- непідконтрольні алгоритми навчання кластеризації (k-засоби, ієрархічна кластеризація) та методи прогнозування даних (багатовимірне масштабування, аналіз основних компонентів, аналіз кореспонденції).

Приклад візуалізації даних за допомогою Orange представлено на рисунку 1.2.

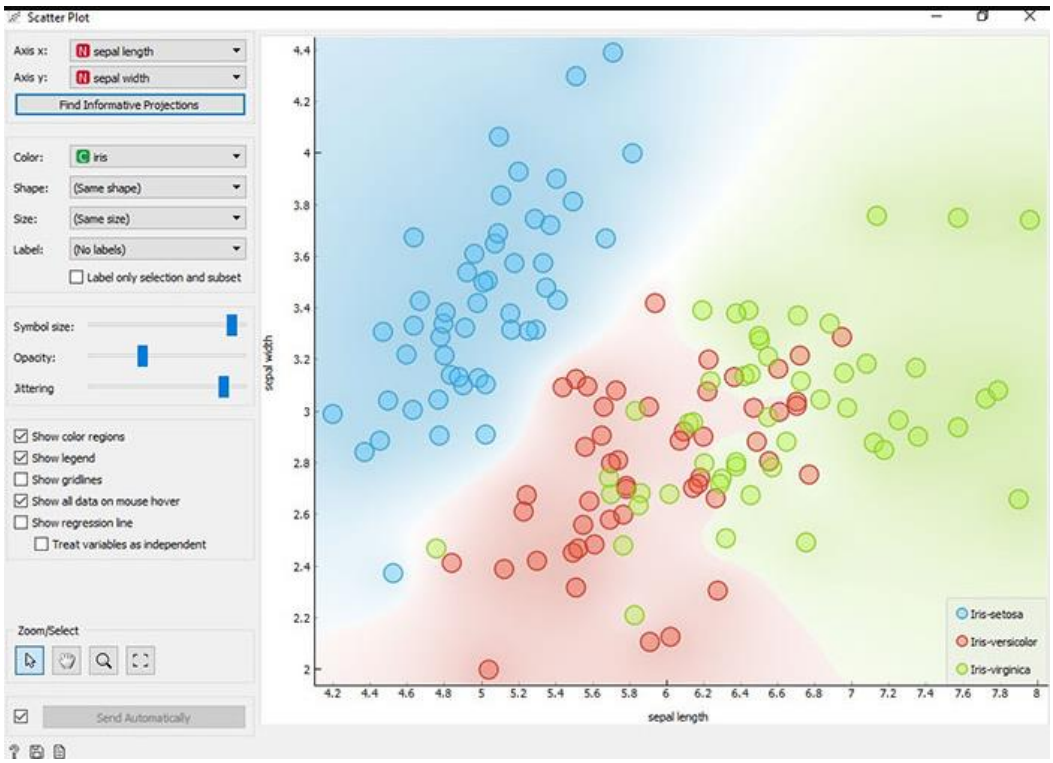

Рисунок 1.2 – візуалізація даних за допомогою Orange

Головним недоліком додатку є те, що програма підтримує цілий спектр віджетів, які потрібно завжди доставляти на вашу робочу машину (замість використання єдиної системи), а також відсутність розкладу заливок даних.

Weka — це набір засобів візуалізації та алгоритмів для аналізу даних і вирішення задач прогнозування, разом з графічною оболонкою для доступу до них.

Weka надає користувачам наступний перелік можливостей:

- уможливлює імпорт даних з бази даних, текстових файлів у форматі CSV, а також попереднє опрацювання цих даних за допомогою різноманітних алгоритмів (фільтрів). Ці фільтри використовуються для трансформування даних, а також для видалення певних атрибутів;

- надає можливість застосувати алгоритми класифікації та регресійного аналізу до обраного набору даних, візуалізувати та оцінити результати, відобразити ROC криві тощо;

- надає доступ до методів, які дозволяють оцінити взаємозв'язки між атрибутами;

- містить різноманітні методи кластеризації, наприклад метод кластеризації методом k-середніх, EM-алгоритм тощо;

- дозволяє ідентифікувати атрибути, які найбільш впливають на якість прогнозування;

- відображає точкові діаграми.

Приклад програмного інтерфейсу Weka наведено на рисунку 1.3.

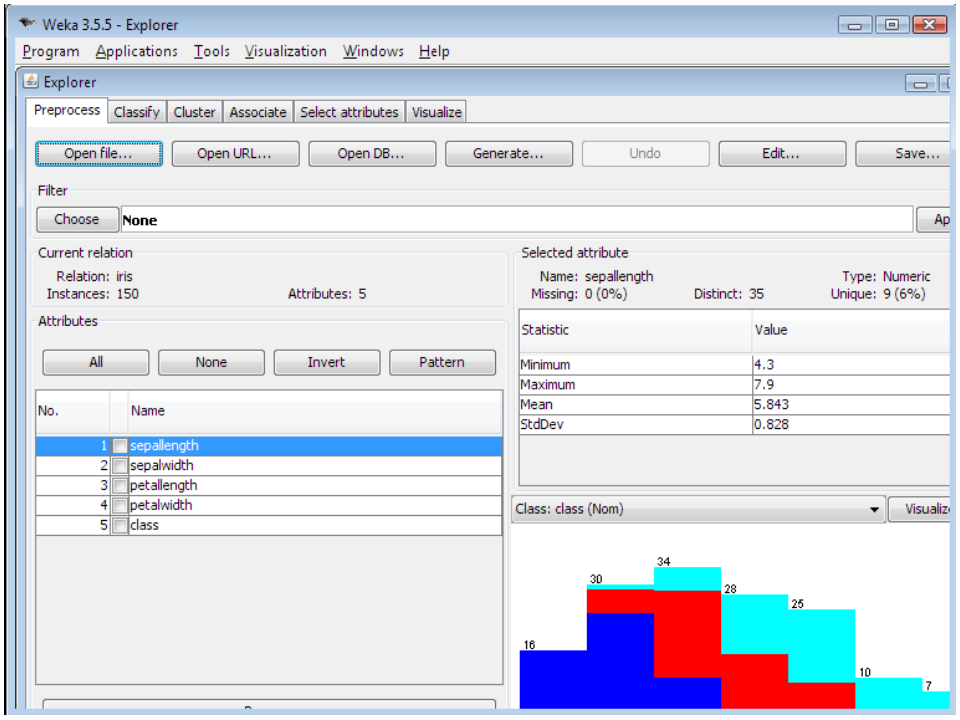

Рисунок 1.3 – Програмний інтерфейс Weka

З головних мінусів – досить примітивний візуальний інтерфейс, неможливість виставлення періодичності заливок, а також відсутність кастомізації вхідних даних.

Жодна із сучасних програмних реалізацій не є ідеальною, у них є певні недоліки. До основних можна віднести відсутність можливості користувачу налаштовувати форматування вхідних даних, а також відсутність якого-небудь планувальника обробки даних за часом. Також багато з доступних на даний момент рішень прив’язані до даних конкретних форматів, що сильно звужує коло методів обробки інформації.

1.3 Фреймфорки для обробки великих даних

Жодна сучасна система, націлена на аналіз та візуалізацію великих обсягів даних, не обходиться без технологій обробки тих самих даних.

В даний час існує близько декілька десятків різновидів фреймворків з обробки великих обсягів даних. Одним з найбільш широко використовуваних варіантів є класичний, а також один з найкращих в наш час, фреймворк, який вже став синонімом Big Data – Hadoop[5].

Hadoop чудово підходить для надійних, масштабованих, розподілених обчислень. Однак, він також може використовуватися для зберігання файлів загального призначення. Він може зберігати та обробляти петабайти даних. Це рішення складається з трьох основних компонентів:

- файлова система HDFS, відповідальна за зберігання даних у кластері Hadoop;

- система MapReduce, призначена для обробки великих обсягів даних у кластері;

- YARN, ядро, яке обробляє управління ресурсами.

Hadoop може зберігати та обробляти багато петабайтів інформації, тоді як для найшвидших процесів у Hadoop потрібно лише кілька секунд. Він також забороняє будь-які редагування даних, які вже зберігаються в системі HDFS під час обробки.

До основних недоліків Hadoop можна віднести проблеми із великою кількістю невеликих файлів. Hadoop підходить для невеликої кількості великих файлів, але якщо мова йде про програму, яка займається великою кількістю невеликих файлів, Hadoop тут не працює. Невеликий файл - це не що інше, як файл, який значно менший за розмір блоку Hadoop, який за замовчуванням може бути або 128 МБ, або 256 Мб. Ці великі кількості невеликих файлів перевантажують Namenode, оскільки він зберігає простір імен для системи та ускладнює роботу Hadoop.

Наступним розглянемо фреймворк Apache Spark. Це фреймворк з відкритим кодом, створений як більш досконале рішення, порівняно з Apache Hadoop. Початковий фреймворк був чітко побудований для роботи з Big Data. Основна відмінність цих двох рішень - це модель пошуку даних.

Hadoop зберігає дані на жорсткому диску разом з кожним кроком алгоритму MapReduce. У той час як Spark здійснює всі операції, використовуючи пам'ять з випадковим доступом. Завдяки цьому Spark демонструє швидку продуктивність і дозволяє обробляти масивні потоки даних. Функціональні стовпи та основні особливості Spark - це висока продуктивність та безвідмовна безпека[3].

Він має п'ять компонентів: ядро та чотири бібліотеки, що оптимізують взаємодію з Big Data. Spark SQL[6] - одна з чотирьох виділених базових бібліотек, яка використовується для структурованої обробки даних.

До основних недоліків можна віднести процес оптимізації коду та відсутність власної файлової системи.

У випадку Apache Spark вам потрібно оптимізувати код вручну, оскільки в ньому немає жодного процесу автоматичної оптимізації коду. Це перетворюється на недолік, коли всі інші технології та платформи рухаються до автоматизації.

Apache Spark не має власної системи управління файлами. Він залежить від деяких інших платформ, таких як Hadoop або інших хмарних платформ.

І останнім розглянемо такий фреймворк, як Apache Flink. Це система потокові передачі даних, спрямована на забезпечення можливостей для розподілених обчислень по потоках даних. Трактуючи пакетні процеси як особливий випадок потокової передачі даних, Flink є ефективним як пакетною, так обробкою в режимі реального часу.

Flink добре підходить для розробки програм на основі подій. Ви можете ввести в нього контрольні точки, щоб зберегти прогрес у разі відмови під час обробки. Flink також має зв'язок із популярним засобом візуалізації даних Zeppelin.[7]

Alibaba використовував Flink для спостереження за поведінкою споживачів та пошуком рейтингу в день синглів. Як результат, продажі зросли на 30%.

Фінансовий гігант ING використовував Flink для створення програм виявлення шахрайства та сповіщення користувачів[8]. Більше того, Flink також має алгоритми машинного навчання.

Серед основних недоліків цього фреймворку можно виділити відсутність зручного API для роботи з ним, а також певні обмеження щодо управління пам’яттю та відмовостійкість.

1.4 Hadoop MapReduce або Apache Spark

MapReduce - модель програмування та пов'язана з нею реалізація для обробки та генерації великих наборів даних з паралельним розподіленим алгоритмом на кластері[8].

Алгоритм MapReduce корисний для обробки величезної кількості даних паралельно, надійним та ефективним способом у кластерних середовищах.

Він розділяє вхідні завдання на більш дрібні та керовані підзадачі для їх виконання паралельно.

Алгоритм MapReduce працює, розбиваючи процес на 3 фази (рис. 1.4.):

- Map фаза;
- Sort & Shuffle фаза;
- Reduce фаза.

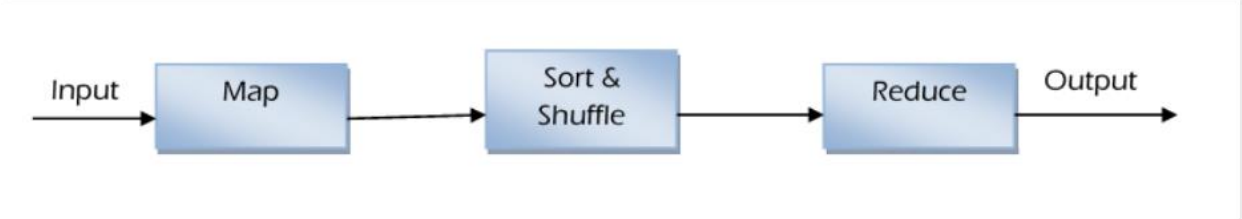

Рисунок 1.4 – фази MapReduce

У MapReduce для кожної фази є пара ключ-значення як для входу, так і виходу.

MapReduce завжди очікуватиме введення у вигляді пар ключ & значення від шарів HDFS (якщо розглядати Hadoop, рис. 1.5.).

Як тільки обробка MapReduce завершиться, вона знову видасть результат поверх HDFS у вигляді пари (Key, Value).

Phase	Input	Output
Mapper	(K,V)	(K,V)
Shuffle & Sort	(K,V)	(K, list(V))
Reducer	(K,list(V))	(K,V)

Рисунок 1.5 – введення та виведення фаз MapReduce

Функція map - перший крок в алгоритмі MapReduce. Вона приймає вхідні значення та розділяє їх на менші підзадачі, а потім паралельно виконує необхідні обчислення для кожної підзадачі[8].

Фаза map виконує наступні два кроки:

- розщеплення - приймає вхідний набір даних і ділить на менші набори підданих;

- картографування - приймає менші набори підданих як вхідні дані та виконує необхідні дії або обчислення для кожної підмножини даних.

Вихід функції Map - це набір пар ключів і значень як <Ключ, Значення>.

Приклад роботи map функції можна побачити на рис. 1.6.

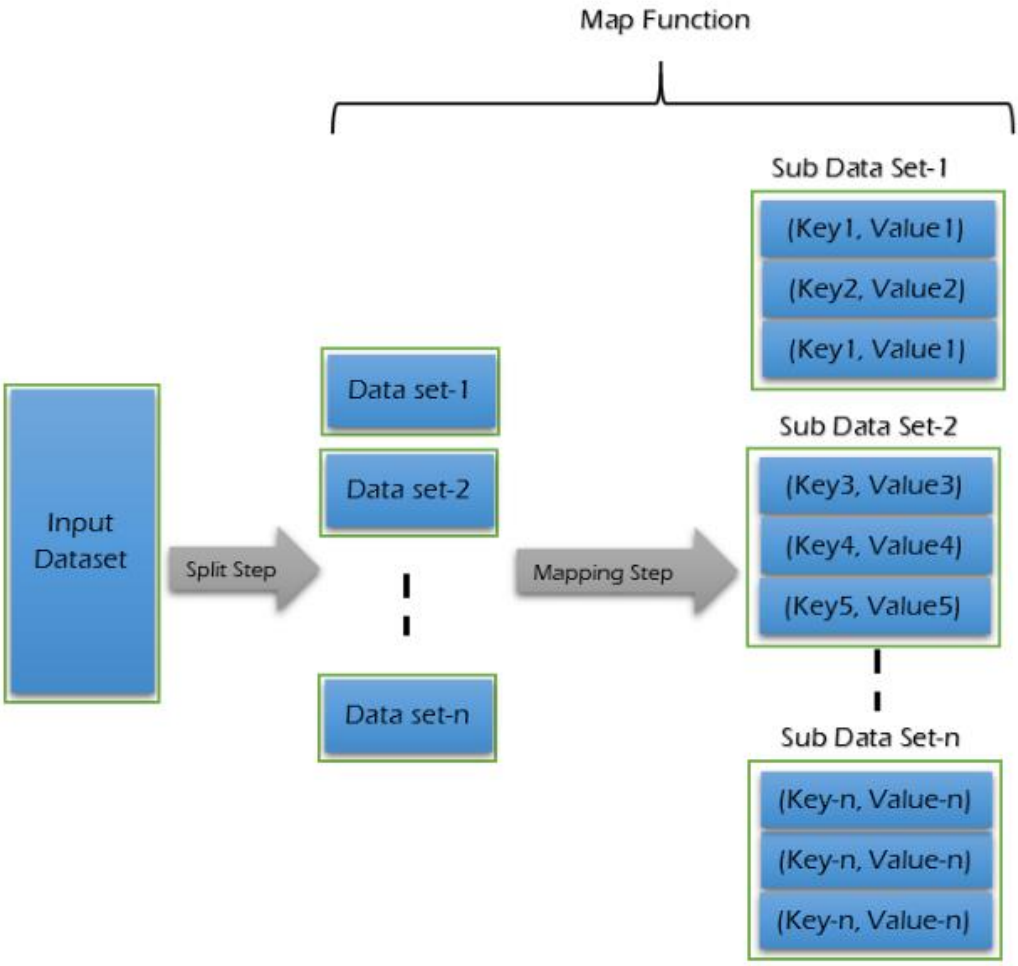

Рисунок 1.6 – приклад роботи map функції

Sort & Shuffle - це другий крок в алгоритмі MapReduce.

Вихід Mapper буде прийнятий як вхід для сортування та переміщення.

Переміщення - це групування даних з різних вузлів на основі ключа. Це логічна фаза.

Він виконує наступні два кроки:

- об'єднання - поєднує всі пари ключових значень, які мають однакові ключі та повертає <Ключ, Список <Значення>>;

- сортування - відбирає результат з об'єднання та сортує всі пари ключових значень за допомогою ключів. Цей крок також повертає <Ключ, Список <Значення>> вихід, але з відсортованими парами ключ-значення[9].

Нарешті, функція shuffle повертає список <Ключ, Список <Значення>> відсортованих пар до фази reducer[7].

Приклад фази Sort & Shuffle можна побачити на рис. 1.7.

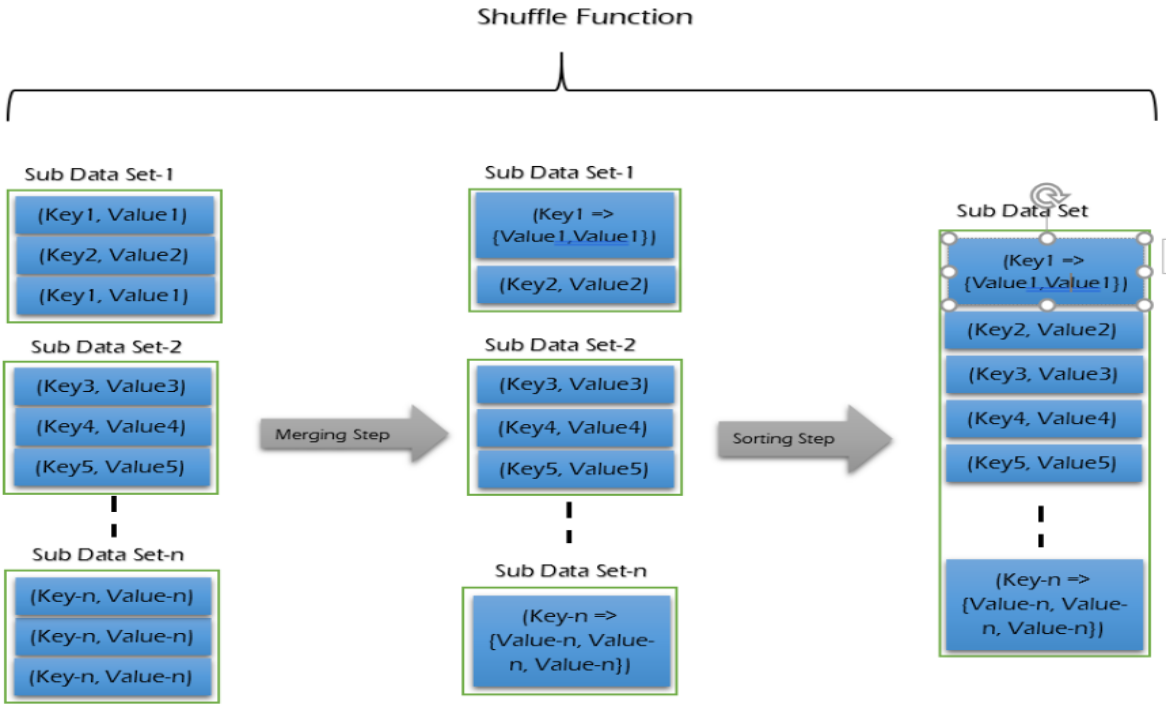

Рисунок 1.7 – приклад роботи shuffle функції

Фаза reduce є завершальним кроком в алгоритмі MapReduce.

Вона бере список <Ключ, список <Значення>> відсортованих пар з функції Shuffle та виконує операцію reduce.

Після завершення фази reduce кластер збирає дані для формування відповідного результату і відправляє їх назад на сервер Hadoop.

Приклад роботи фази reduce можна побачити на рис. 1.8.

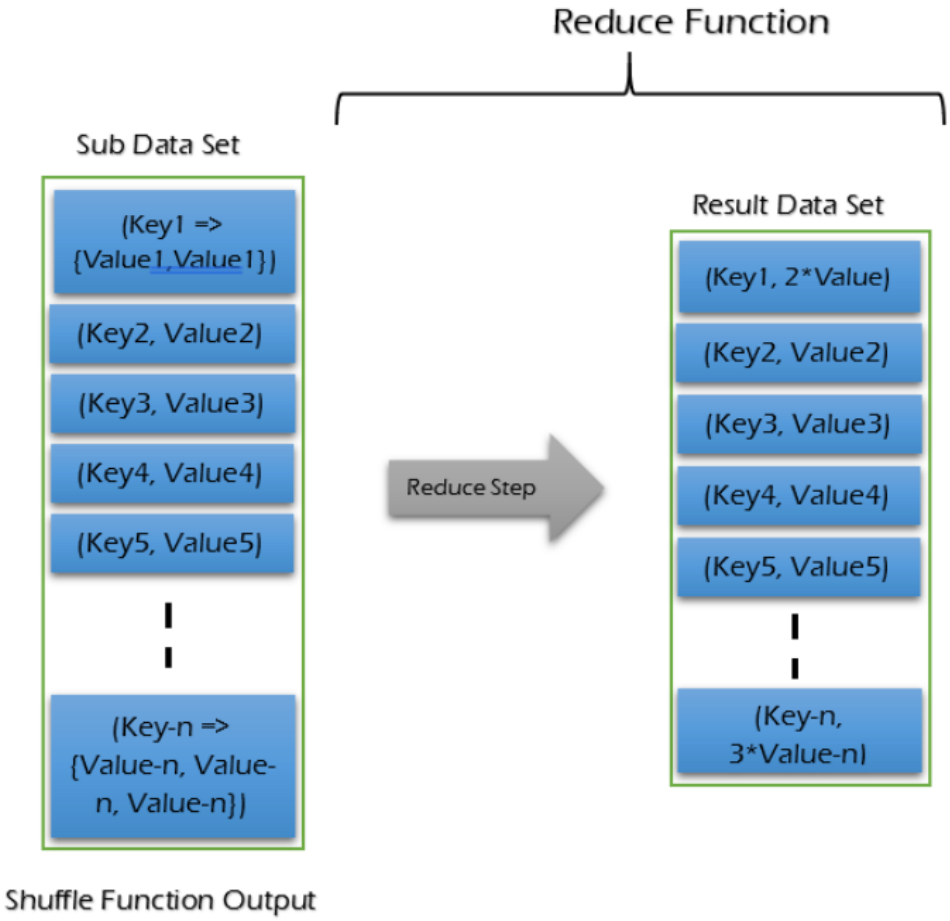

Рисунок 1.8 – приклад роботи reduce функції

У Hadoop, з паралельним та розподіленим алгоритмом, MapReduce обробляє великі набори даних. Існують завдання, які нам потрібно виконати: Map і Reduce, MapReduce вимагає багато часу для виконання цих завдань, тим самим збільшуючи затримку. Дані розподіляються та обробляються через кластер в MapReduce, що збільшує час і зменшує швидкість обробки.

Як рішення цього обмеження, Hadoop Spark подолала цю проблему шляхом обробки даних в пам'яті. Обробка пам'яті відбувається швидше, оскільки не витрачається час на переміщення даних/процесів на диск і з нього. Spark в 100 разів швидша за MapReduce, оскільки вона обробляє все в пам'яті. Однак обсяг оброблюваних даних також відрізняється: Hadoop MapReduce здатний працювати з набагато більшими наборами даних, ніж Spark.

Hadoop MapReduce дозволяє паралельно обробляти величезну кількість даних. Він розбиває великий фрагмент на більш дрібні, які обробляються окремо

на різних вузлах даних, і автоматично збирає результати по декількох вузлах, щоб повернути один результат. Якщо отриманий набір даних перевищує доступну оперативну пам’ять, Hadoop MapReduce може перевершити Spark.

Hadoop MapReduce є хорошим рішенням, якщо швидкість обробки не є критичною. Наприклад, якщо обробку даних можна проводити протягом нічних годин, є сенс розглянути можливість використання Hadoop MapReduc[10].

Переваги Apache Spark:

- швидка обробка даних. Обробка в пам'яті робить Spark швидше, ніж Hadoop MapReduce - до 100 разів для даних в оперативній пам'яті і до 10 разів для даних у дисках;

- ітеративна обробка. Якщо завдання полягає в обробці даних знову і знову - Spark перемагає Hadoop MapReduce. Еластичні розподілені набори даних Spark (RDD) дозволяють виконувати кілька операцій в пам'яті, тоді як Hadoop MapReduce повинен записувати проміжні результати на диск[11];

- обробка в режимі реального часу. Якщо бізнесу потрібна негайна інформація, тоді він повинен вибрати Spark та його обробку в пам'яті;

- обробка графіків. Обчислювальна модель Spark хороша для ітеративних обчислень, типових для обробки графіків. І Apache Spark має GraphX - API для обчислення графіків;

- приєднання до наборів даних. Завдяки своїй швидкості, Spark може створювати всі комбінації швидше, хоча Hadoop може бути кращим, якщо потрібно об'єднати дуже великі набори даних, які потребують багато перетасувань та сортування.

1.5 Постановка задачі

Метою роботи є дослідження засобів і технологій обробки та аналізу великих обсягів даних. Результатом дослідження повинна стати розробка веб-додатку з

використанням технологій обробки та аналізу великих даних та застосування знань, здобутих під час дослідницької роботи.

Для забезпечення роботи веб-додатку необхідна наявність у користувача пристрою з доступом до мережі Інтернет та щонайменше наступними системними вимогами: Internet Explorer 11, Google Chrome 49, Safari 10, Firefox 52.

Для максимально ефективної обробки текстової інформації та успішної ідентифікації та візуалізації даних в системі необхідно реалізувати:

- підходи та методи обробки текстових файлів різних форматів;
- систему із можливістю аналізу і візуалізації даних за вказаними параметрами;
- методи конвертування даних для підвищення швидкості обробки інформації.

Функціонал додатку повинен бути наступний:

– авторизація користувача в системі;

– можливість налаштування параметрів вхідних даних;

– отримання візуальної схеми отриманих даних;

– можливість зберігати конвертовані дані та експортувати їх;

– можливість задавати періодичність заливок даних.

Відповідно до аналізу предметної галузі, задоволення потреб кінцевих користувачів систем аналізу даних є пріоритетним напрямом діяльності, а системи обробки великих масивів даних набирають все більшу популярність, отже аналіз та розробка систем обробки та аналізу великих даних та визначення його вартості і місця продажу є актуальною.

2 ОПИС ДОСЛІДЖЕННЯ ТА ПРИЙНЯТИХ ПРОГРАМНИХ РІШЕНЬ

2.1 Аналіз підходів зі зменшення розмірності великих даних

Коли мова заходить про роботу з дуже великими обсягами даних, найбільшою проблемою є знаходження та виявлення причин подібності деяких частин даних, що дозволить у подальшому зменшити розмірність з мінімальними втратами інформації.

Існує дуже багато алгоритмів кластеризації даних, але якщо розглядати роботу із великими об’ємами, тоді можна виділити п’ять основних:

– K-means;
– Gaussian mixture;
– Power iteration clustering (PIC);
– Latent iteration clustering (LDA);
– Streaming k-means.

Також ці методи допомагають у візуалізації великих обсягів даних, що потребується для заданої роботи.

Серед ознак того, що нам підходить той чи інший алгоритм кластеризаціїї треба виділити три основних критерії:

- алгоритм не повинен вимагати великих обчислень через те, що у більшості випадків буде використовуватися дуже великі обсяги даних[12];
- алгоритм у своїй основі повинен розглядати найширші варіації розподілення даних;
- задля досягнення більш продуктивної системи на великих обсягах даних, ми можемо пожертвувати деякої продуктивністю на дуже малих розподілених даних.

Розглянувши усі наведені вище алгоритми, було вирішено використати один з двох – PIC або LDA, що будуть розглянуті у подальшому.

2.2 Principal Component Analysis (PCA)

PCA - це безконтрольний метод машинного навчання, який використовується для зменшення розмірності[14]. Основна ідея алгоритму аналізу основних компонентів (PCA) полягає в зменшенні розмірності набору даних, що складається з безлічі змінних, співвіднесених одна з одною, сильно або не дуже, зберігаючи при цьому варіації, наявні в наборі даних, до максимальної міри.

Це робиться шляхом перетворення змінних в новий набір змінних, який є комбінацією змінних або атрибутів з початкового набору. Ця комбінація атрибутів відома як "Основні компоненти", а компонент, який має максимальну відхилену дисперсію, називається домінуючим головним компонентом. Порядок збереження дисперсії зменшується, коли ми рухаємось вниз у порядку, тобто PC1> PC2> PC3> ... тощо.

Нижче наведено зразок трансформації 2D малюнку у 1D (рис. 2.1.).

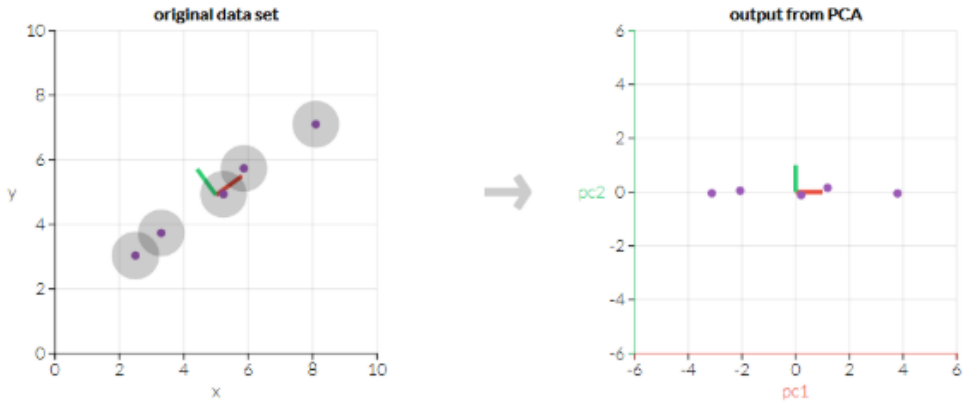

Рисунок 2.1 – трансформація 2D у 1D

Також на рисунку 2.2 наведено приклад трансформації 3D зображення у 2D/1D

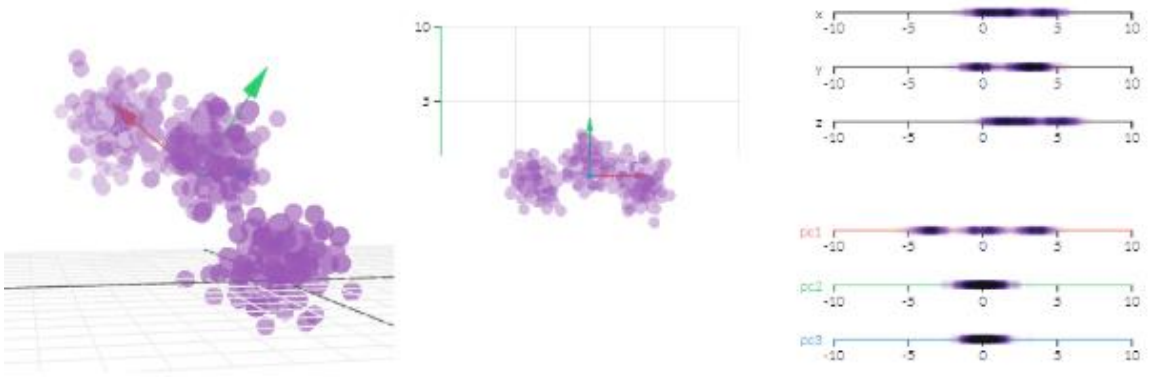

Рисунок 2.2 – трансформація 3D у 2D/1D (PC1 > PC2 > PC3)

Після того, як ми перетворимо дані в основні компоненти, ми можемо вирішити відмовитись від змінних, які не мають різниці. Це дає можливість зменшити розміри і зосередити увагу на тих, що мають більшу дисперсію[14].

На практиці PCA використовується здебільшого у двох випадках:

- зменшення розмірності: інформація, що поширюється на велику кількість стовпців, перетворюється на основні компоненти (ОК), так що перші кілька ОК можуть пояснити значну частину загальної інформації (дисперсії). Ці ОК можуть використовуватися як пояснювальні змінні в моделях машинного навчання;
- візуалізація класів: візуалізувати поділ класів (або кластерів) важко для даних з більш ніж 3 вимірами (функціями). З першими двома ОК, як правило, можна побачити чітке розмежування.

Першим кроком необхідно стандартизувати кожен стовпець. Це означає, що необхідно перетворити категоричну змінну у фіктивні числові змінні, оскільки PCA працює тільки на числових значеннях.

Другим кроком йде обчислення матриці коваріації.

Коваріація описується як дві змінні пов'язані одна з одною, тобто якщо дві змінні рухаються в одному напрямку відносно один одного чи ні. Коли коваріація є позитивною, це означає, що якщо одна змінна збільшується, збільшується і інша[15]. І навпаки.

Коваріація двох змінних, де X та Y – це дві різні змінні, а n – кількість комбінацій, показана на рисунку 2.3.

$$Cov(X,Y) = \frac{\Sigma(X_i - \bar{X})\,(Y_i - \bar{Y})}{n}$$

Рисунок 2.3 – коваріація X та Y

Коваріаційна матриця обчислює коваріацію всіх можливих комбінацій стовпців. В результаті вона стає квадратною матрицею з однаковою кількістю рядків і стовпців[12]. Дія матриці на загальний вектор можна розглядати як поєднання розтягування та обертання (рис.2.4.).

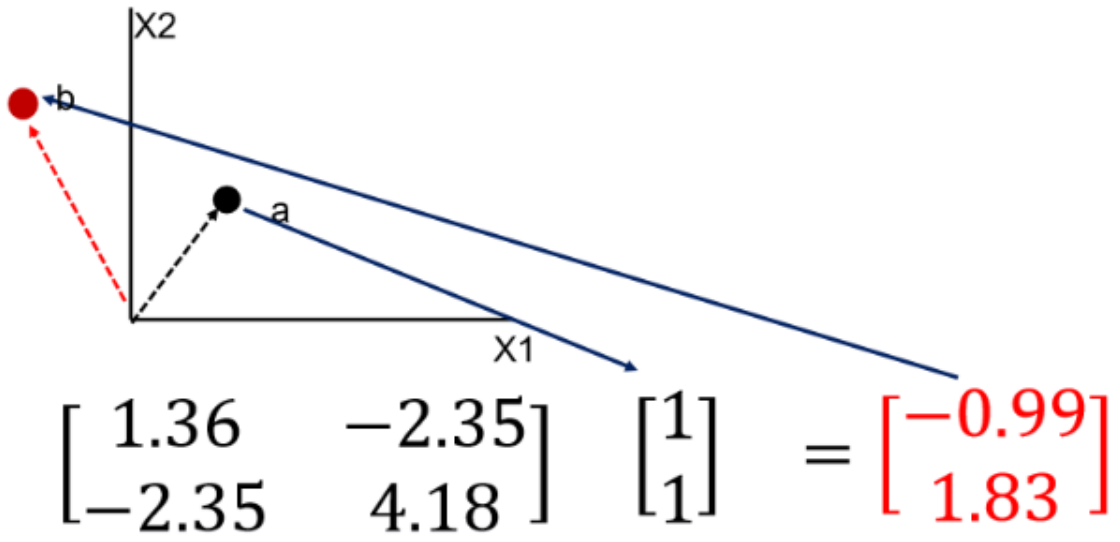

Рисунок 2.4 – коваріація матриці з поєднанням та розтягуванням

Для даної матриці існує спеціальний напрямок, уздовж якого ефект лише розтягування (без обертання), ці спеціальні напрями називаються Ейгеновими напрямками (рис. 2.5.).

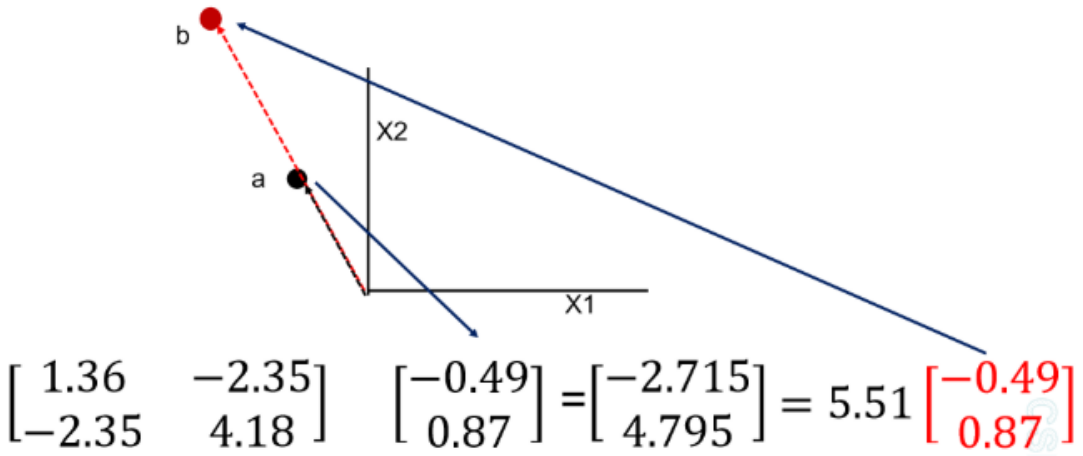

Рисунок 2.5 – коваріантна матриця тільки з розтягуванням

Останнім же кроком є отримання основних характеристик компонента. Взявши крапковий добуток власного вектора та стандартизованих стовпців, ми можемо отримати основні компоненти.

2.3 Linear Discriminant Analysis (LDA)

LDA - це контрольований метод машинного навчання, який використовується для розділення двох груп/класів. Основна ідея лінійного дискримінантного аналізу (LDA) полягає в тому, щоб максимально розділитись між двома групами, щоб ми могли прийняти найкраще рішення про їх класифікацію. LDA схожий на PCA, який допомагає зменшити розмірність, але він фокусується на максимальній відокремленості відомих категорій шляхом створення нової лінійної осі та проектування точок даних на цій осі[16].

LDA не працює над пошуком головного компонента, він в основному розглядає, який тип точки/підпростору дає більше дискримінації для розділення даних.

Порівння PCA та LDA розподілення наведено на рисунку 2.6.

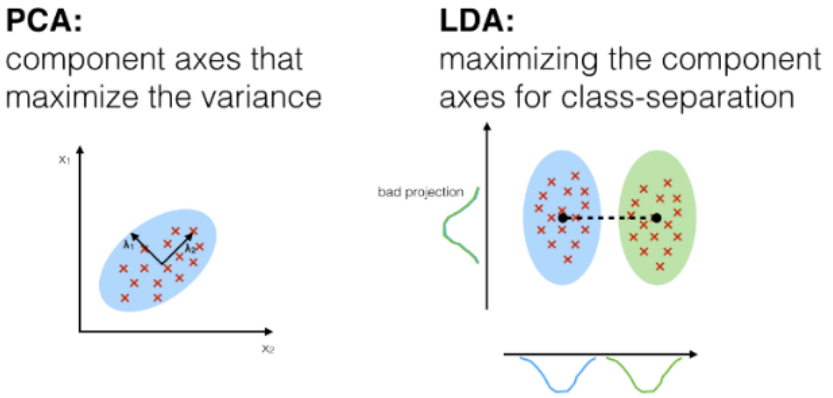

Рисунок 2.6 – PCA vs LDA

Завдання LDA - знайти лінію, яка максимально розмежовує клас.

Середній вектор використовується для пошуку середнього значення точок даних кожного класу (μ), x – середня змінна, а T - період (рис. 2.7.).

$$\mu_i = \frac{1}{N_i}\sum_{x\in\omega_i} x \quad and \quad \tilde{\mu}_i = \frac{1}{N_i}\sum_{y\in\omega_i} y = \frac{1}{N_i}\sum_{x\in\omega_i} w^T x$$
$$= w^T \frac{1}{N_i}\sum_{x\in\omega_i} x = w^T \mu_i$$

Рисунок 2.7 – середній вектор кожного класу x та y

Мета алгоритму розподілення - знайти найкращий набір w, який дає максимальне розмежування. На рисунку 2.8 можна побачити L1 об’єктивну функцію.

$$J(w) = \left|\tilde{\mu}_1 - \tilde{\mu}_2\right| = \left|w^T\mu_1 - w^T\mu_2\right| = \left|w^T\left(\mu_1 - \mu_2\right)\right|$$

Рисунок 2.8 – L1 об’єктивна функція

Однак відстань між цими засобами не є дуже хорошим показником, оскільки воно не враховує стандартне відхилення в межах класів.

Дані, де інваріантність в межах класу мінімальна, а мінливість серед інших класів є максимальною, вважаються хорошими. Рішення, запропоноване Фішером, полягає в максимізації функції, яка представляє різницю між засобами, нормованими мірою внутрішньокласової змінності, або так званого розсіювання. Для кожного класу ми визначаємо розкид, еквівалент дисперсії (рис. 2.9.)[16].

$$\tilde{s}_i^2 = \sum_{y \in \omega_i} \left(y - \tilde{\mu}_i \right)^2$$

Рисунок 2.9 – функція розкиду

Si^2 вимірює мінливість у класі ω і після проектування його на Y-простір.

$S1^2 + S2^2$ вимірює мінливість у двох класах, що знаходяться після проектування, тому її називають внутрішньокласовим розсіюванням проектованих зразків.

Отже, лінійний дискримінант Фішера визначається як лінійна функція, яка максимізує функцію критерію. На рисунку 2.9 наведена L2 об’єктивна функція.

$$J(w) = \frac{\left| \tilde{\mu}_1 - \tilde{\mu}_2 \right|^2}{\tilde{s}_1^2 + \tilde{s}_2^2}$$

Рисунок 2.8 – L2 об’єктивна функція

Отже, порівнюючи ці два методу зменшення розмірності великих обсягів даних, можна зробити висновок, що для нашої системи більш зручним та комфортним є алгоритм PCA, так як він має більше можливостей для візуалізації отриманих даних та досить гнучку структуру.

3 ПРОЕКТУВАННЯ ПРОГРАМНОЇ СИСТЕМИ

3.1 UML проектування програмної системи

Use case діаграма відображає взаємодію користувача з системою та функції системи, доступні для того чи іншого актора[20]. В розробленій системі є один актор, котрий може авторизуватися в системі щоб відкрити доступ до додатку. Усі функції додатку доступні лише при наявності стабільного підключення до мережі Інтернет. Основними функціями є: можливість налаштування параметрів вхідних даних, можливість задавати періодичність заливок даних, отримання візуальної схеми отриманих даних, перегляд історії безпосередньо сама заливка даних. Діаграма зображена на рисунку 3.1.

Для побудови UML діаграм було використано онлайн ресурс Draw.io.

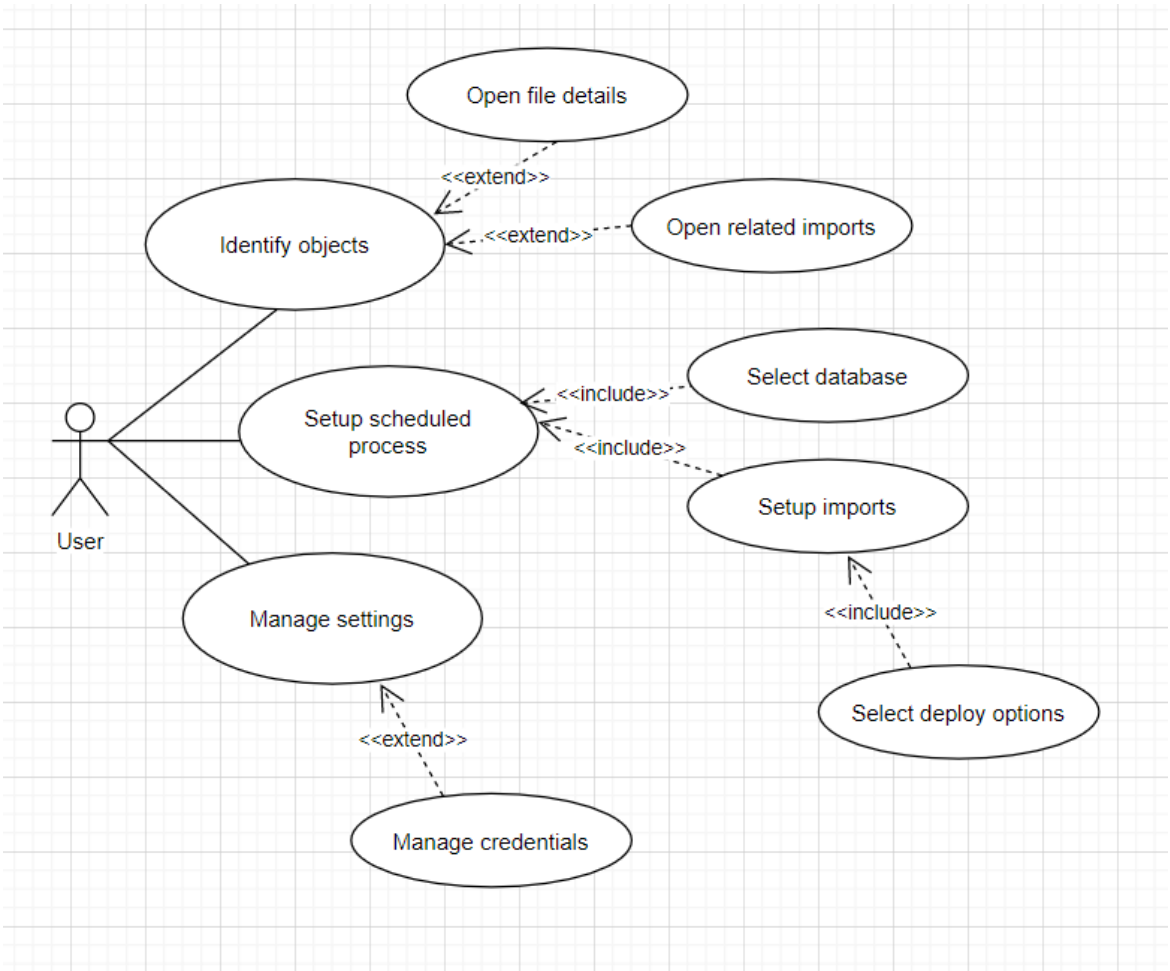

Рисунок 3.1 – Use case діаграма

На Use Case діаграмі зображено весь основний функціонал веб-додатку та вказано, які функції розширюють інші. Наприклад, як видно на діаграмі, користувач може відкрити деталі файлу або подивитися зв’язані з ним імпорти після його ідентифікації.

Користувач також може налаштовувати процес періодичності заливок. В цьому випадку, усі дані будуть заливатися в зручний для користувача час.

Діаграма послідовностей відображає те, що відбувається поза очима користувача, коли додаток намагається розпізнати об’єкт[21]. Відразу після автентифікації, користувач бачить інтерфейс програми, де має можливість налаштувати необхідні маски файлів, параметри валідації, а також задати періодичність заливок файлів на сервер. Модель запускає Data Ingestion Service, котрий використовує методи розпізнавання і валідації файлів. На даному етапі працюють функції розпізнавання і валідації файлів. Діаграму наведено на рисунку 3.2.

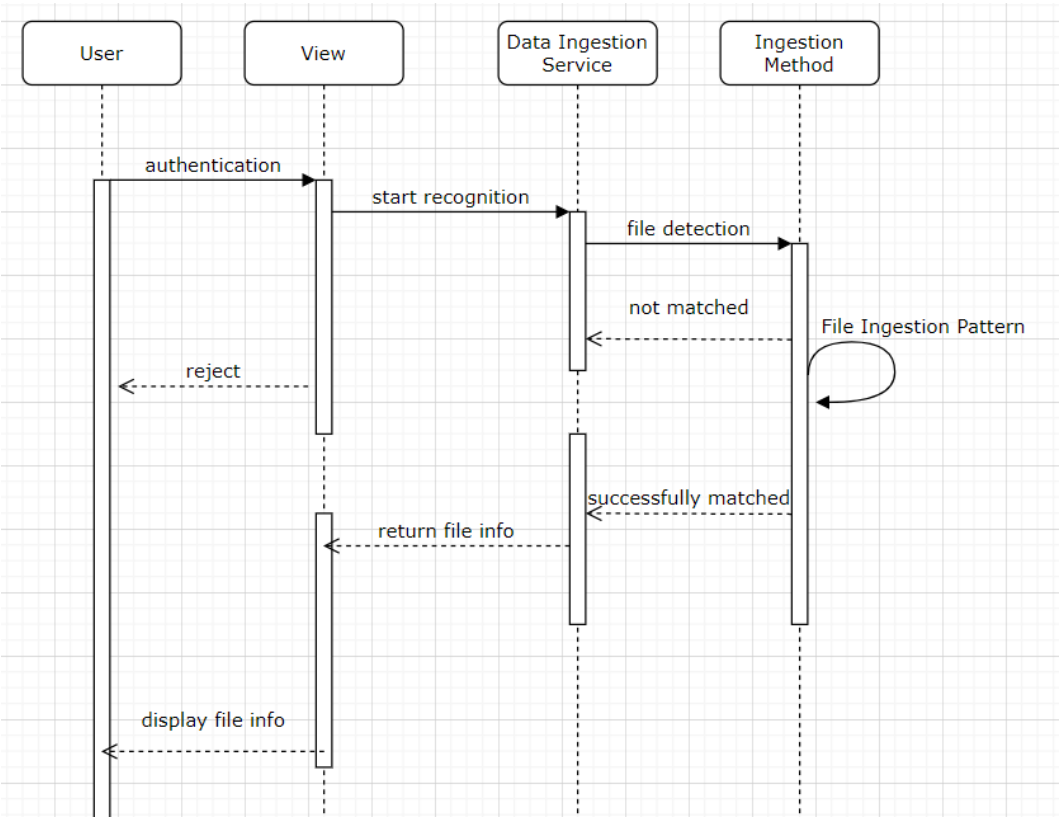

Рисунок 3.2 – Діаграма послідовності

Якщо файл розпізнався, то дані про нього передаються до Data Ingestion Service, котрий формує набір інформації про цей файл та відображує її в інтерфейсі програми.

Наступним кроком було створення діаграми класів (див. рис. 3.3). Ця діаграма відображає основні моделі системи та залежності між ними.

Більшість службових класів та класів роботи з базою даних не були включені до діаграми, бо вони мають службове значення та не допомагають зрозуміти систему на високому рівні.

Діаграма включає в себе клас ImportService, котрий дозволяє виконувати основні операції над імпортами, такі як їх додавання, редагування та видалення. Ще однією важливою складовою є сервіс для розпізнавання та обробки файлів.

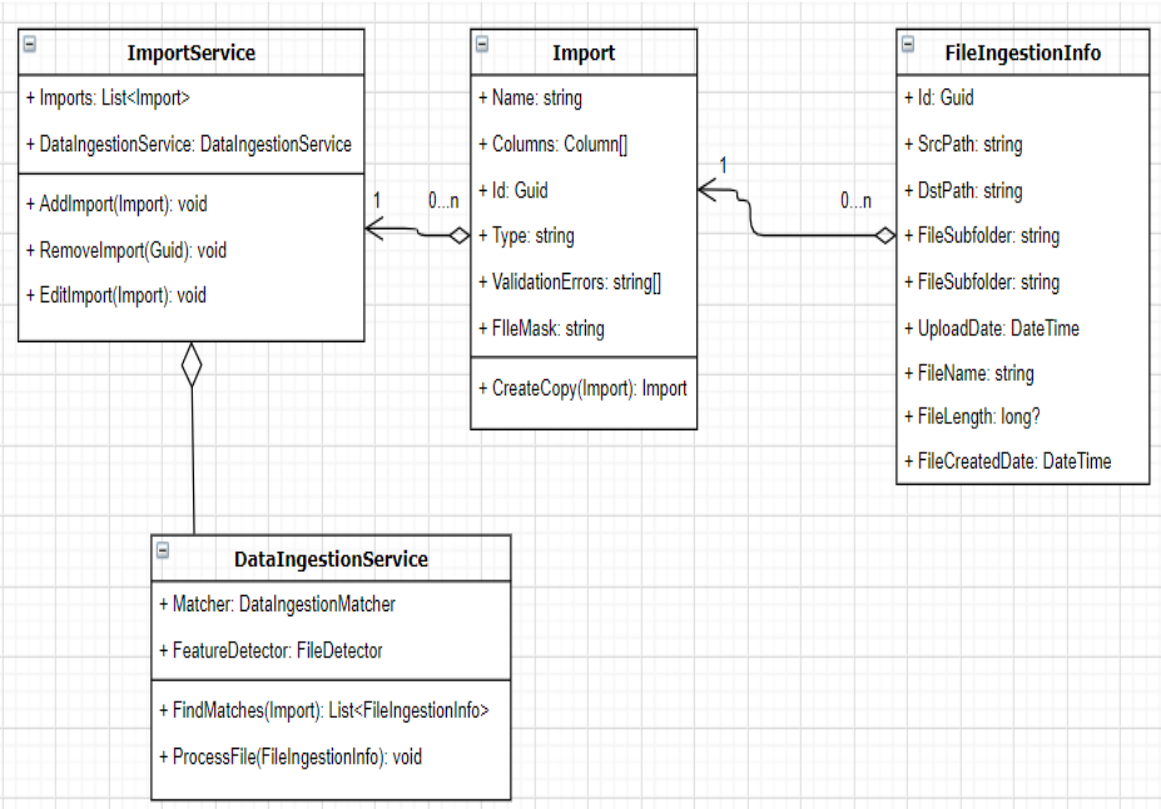

Рисунок 3.3 – Діаграма класів

Клас Import дозволяє створювати копію імпортів, а також використовується для знаходження файлів за заданими масками. Цей клас відповідає безпосередньо

за той об’єкт, котрий буде розпізнаватися та включає в себе усю інформацію, котра необхідна для валідатора та сторонніх сервісів.

3.2 Архітектра веб-додатку

Для реалізації сервісу вирішено використовувати клієнт-серверну архітектуру. Ця архітектура є одним із архітектурних шаблонів програмного забезпечення та є домінуючою концепцією у створенні розподілених мережевих додатків і передбачає взаємодію та обмін даними між ними. На рисунку 3.4 наведене схематичне зображення архітектури системи із вказаними технологіями, що були застосовані на кожному з рівнів.

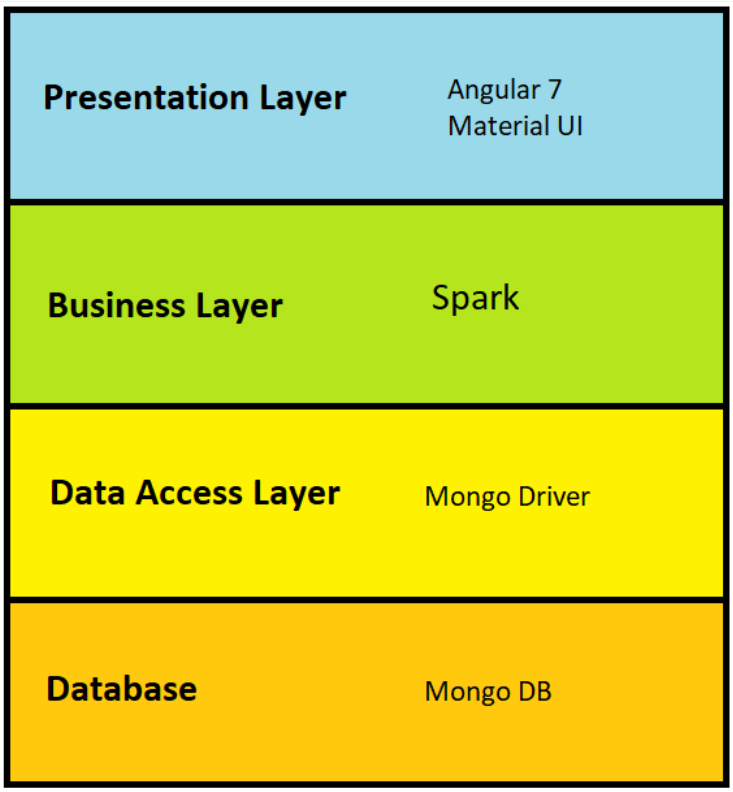

Рисунок 3.4 – Архітектура системи

Presentation layer - це той рівень, з яким безпосередньо взаємодіє користувач. Цей рівень включає компоненти для користувача інтерфейсу, механізм отримання даних від користувача. На даному рівні розташовані всі ті компоненти, які складають призначений для користувача інтерфейс (стилі, статичні сторінки html, javascript), а також контролери, об'єкти контексту запиту. Основою цього рівня лежить проект, створений із застосуванням фреймворку Angular та дизайну Material UI.

Business layer (рівень бізнес-логіки) містить набір компонентів, які відповідають за обробку отриманих від рівня уявлень даних, реалізує всю необхідну логіку додатка, все обчислення, взаємодіє з базою даних і передає результат обробки. Саме на цьому рівні відбувається розпізнавання та обробка файлів[22].

Data Access layer (рівень доступу до даних) зберігає моделі, що описують використовувані сутності, також тут розміщуються специфічні класи для роботи з різними технологіями доступу до даних. Тут також зберігаються репозиторії, через які рівень бізнес-логіки взаємодіє з базою даних.

4 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

4.1 Вибір засобів розробки

Під час планування та моделювання програмного продукту було визначено, що проект має складатися з двох частин: сервер, на якому будуть зберігатися дані, та інтерфейс веб-додатку, розроблений із застосуванням Angular, що взаємодіє із сервером за допомогою Web API 2 - контроллерів.

Для розробки серверу використовується технологія створення Web-застосувань мовою C#[22] з використанням ASP MVC 5 фреймворку. C# – об'єктно-орієнтована мова програмування, розроблена в 1998-2001 роках групою інженерів під керівництвом Андерса Хейлсберга в компанії Microsoft як мова розробки додатків для платформи Microsoft .NET Framework[20] і згодом була стандартизована як ECMA-334 і ISO / IEC 23270.

Для розробки використовується середовище Visual Studio Professional 2019. Visual Studio - продукт компанії Microsoft, що включає інтегроване середовище розробки програмного забезпечення і ряд інших інструментальних засобів. Дані продукти дозволяють розробляти як консольні додатки, так і додатки з графічним інтерфейсом, в тому числі з підтримкою технології Windows Forms, а також веб-сайти, веб-додатки, веб-служби як в рідному, так і в керованому кодах для всіх платформ, підтримуваних Windows, Windows Mobile, Windows CE, .NET Framework, Xbox, Windows Phone .NET Compact Framework і Silverlight.

Visual Studio включає в себе редактор вихідного коду з підтримкою технології IntelliSense і можливістю найпростішого рефакторінга коду. Вбудований відладчик може працювати як відладчик рівня вихідного коду, так і відладчик машинного рівня. Решта вбудованих інструментів включають в себе редактор форм для спрощення створення графічного інтерфейсу додатку, веб-редактор, дизайнер класів і дизайнер схеми бази даних. Visual Studio дозволяє створювати і підключати сторонні додатки (плагіни) для розширення функціональності практично на кожному рівні, включаючи додавання підтримки систем контролю версій

вихідного коду (як, наприклад, Subversion і Visual SourceSafe), додавання нових наборів інструментів (наприклад, для редагування і візуального проектування коду на предметно-орієнтованих мовах програмування) або інструментів для інших аспектів процесу розробки програмного забезпечення (наприклад, клієнт Team Explorer для роботи з Team Foundation Server).

Для виклику методів сервера використовуються REST контролери з фреймворку ASP.NET MVC[21] , що дозволяє отримувати та відправляти різноформатні дані по протоколу HTTP.

На сьогоднішній день прийнято використовувати REST – (скор. від англ. RepresentationalStateTransfer – «передача репрезентативного стану») – метод взаємодії компонентів розподіленого додатка в мережі Інтернет, при якому виклик віддаленої процедури являє собою звичайний HTTP-запит, а необхідні дані передаються як параметри запиту.

У свою чергу HTTP – протокол передачі даних, що використовується в комп'ютерних мережах. Назва скорочена від Hyper Text Transfer Protocol, протокол передачі гіпер-текстових документів.

HTTP – протокол прикладного рівня, схожими на нього є FTP і SMTP. Обмін повідомленнями йде за звичайною схемою «запит-відповідь»[24].

Для збереження даних програмного продукту була використана СУБД MongoDB – це база даних документів з досить серйозною масштабованістю та гнучкістю, що потрібно для запитів та індексування, які вам потрібні.

Основна причина, чому MongoDB є фаворитом розробників і адміністраторів віртуалізації, так це простота у використанні. MongoDB поставляється з відмінними інструментами, які заощадять багато часу в цих областях - інструменти, такі як Robo3T, Studio 3T та BI інструменти.

Модель документа MongoDB дуже проста та зрозуміла для розробників, вивчення та використання, але все ж надає всі можливості, необхідні для задоволення найскладніших вимог у будь-якому масштабі[23].

Доступ і маніпуляція даними в базі здійснюється засобами ORM Mongo Driver, метою якого є звільнення розробника від значних типових завдань із

програмування взаємодії з базою даних. Розробник може використовувати Mongo Driver як при розробці з нуля, так і для вже існуючої бази даних.

Mongo Driver піклується про зв'язок класів з таблицями бази даних (і типів даних мови програмування із типами даних Mongo), і надає засоби автоматичної побудови NoSql запитів й зчитування/запису даних, і може значно зменшити час розробки, який зазвичай витрачається на ручне написання типового коду. Mongo Driver генерує виклики і звільняє розробника від ручної обробки результуючого набору даних, конвертації об'єктів і забезпечення сумісності із різними базами даних[24].

Сервер реалізований за допомогою наступних технологій:

- .NET Framework 4.5.1;
- ASP.NET MVC 5;
- Web API 2.

Сервіс представляє собою single page application – додаток, що містить лише одну HTML сторінку, а подальша взаємодія виконується за допомогою динамічно завантажуваних HTML, CSS та JavaScript із використанням Angular. При цьому роутинг конфігурується клієнтом додатку, який отримує та відправляє запити до сервера.

Angular представляє фреймворк від компанії Google для створення клієнтських додатків. Перш за все він націлений на розробку SPA-рішень (Single Page Application), тобто односторінкових додатків. В цьому плані Angular є спадкоємцем іншого фреймворка AngularJS. У той же час Angular це не нова версія AngularJS, а принципової новий фреймворк.

Angular надає таку функціональність, як двостороннє зв'язування, що дозволяє динамічно змінювати дані в одному місці інтерфейсу при зміні даних моделі в іншому, шаблони, маршрутизація і так далі. Однією з ключових особливостей Angular є те, що він використовує в якості мови програмування TypeScript.

Окрім цього, для створення інтерфейса [22] був використаний сучаний фреймворк Material Design.

Material Design - це мова дизайну для веб- і мобільних додатків, який був розроблений Google в 2014 році. Material Design спрощує розробникам налаштування UI, зберігаючи при цьому зручний інтерфейс додатків. Material Design надає добре організований формат і гнучкість.

Angular Material складається з набору попередньо встановлених компонентів Angular. На відміну від Bootstrap, який надає компоненти, які можна використовувати будь-яким способом, Angular Material прагне забезпечити розширений і послідовний інтерфейс. У той же час він дає можливість контролювати, як поводяться різні компоненти.

Фронт-енд додатку в повній мірі може функціонувати на таких браузерах: Google Chrome, Internet Explorer, Opera, Mozilla Firefox та інших браузерах, версії яких підтримують JavaScript, CSS3 та HTML5.

Керування життєвим циклом об’єктів на сервері здійснюється за допомогою Інверсії управління.

Якщо описувати модель взаємодії усіх компонентів системи, то можна зрозуміти, що головною ланкою є сервер з REST-сервісом.

4.2 Опис програмної системи

Як вже було неодноразово сказано, програмна система складається з двох компонентів: серверу та веб-клієнта.

На сервері реалізований REST-сервіс, за допомогою якого веб-клієнт використовує функції системи. Методи, що вертають дані, вертають їх у форматі JSON.

Під час розробки програмної системи однією з цілей було створити максимально простий та зрозумілий інтерфейс веб-клієнта, щоб у користувача не виникало труднощів із завантаженням зображень та пошуком товарів.

Веб-клієнт повністю відповідає принципам юзабіліті з точки зору інтерфейсу. Якщо користувач ще не авторизувався у системі, його буде перенаправлено до сторінки автентифікації (див. рис. 4.1).

Рисунок 4.1 – Сторінка автентифікації

Ця сторінка є формою для вводу даних. Користувач обов’язково повинен ввести свій логін та пароль від акаунту. Тут присутня валідація, що не дозволить користувачеві вести некоректні дані. Валідація присутня як на сервері, так і на клієнті. У випадку введення невірних даних користувача буде повідомлено за допомогою спливаючого вікна та підсвічення невірно заповнених полів. Якщо користувач ще не має аккаунту в системі, його необхідно зареєструвати.

Після авторизації користувач потрапляє на головну сторінку (див. рис. 4.2) зі списком налаштувань для заливок.

Цей список надає основні поля для налаштування, а саме:

- директорію, яку потрібно перевіряти;
- ім’я періодичної заливки;
- поля для налаштування періодичності;

- список імпортів з необхідними параметрами;
- цільову базу даних.

Користувач може з легкістю налаштувати декілька видів заливок для своїх імпортів та виставити зручний для нього час та періодичність.

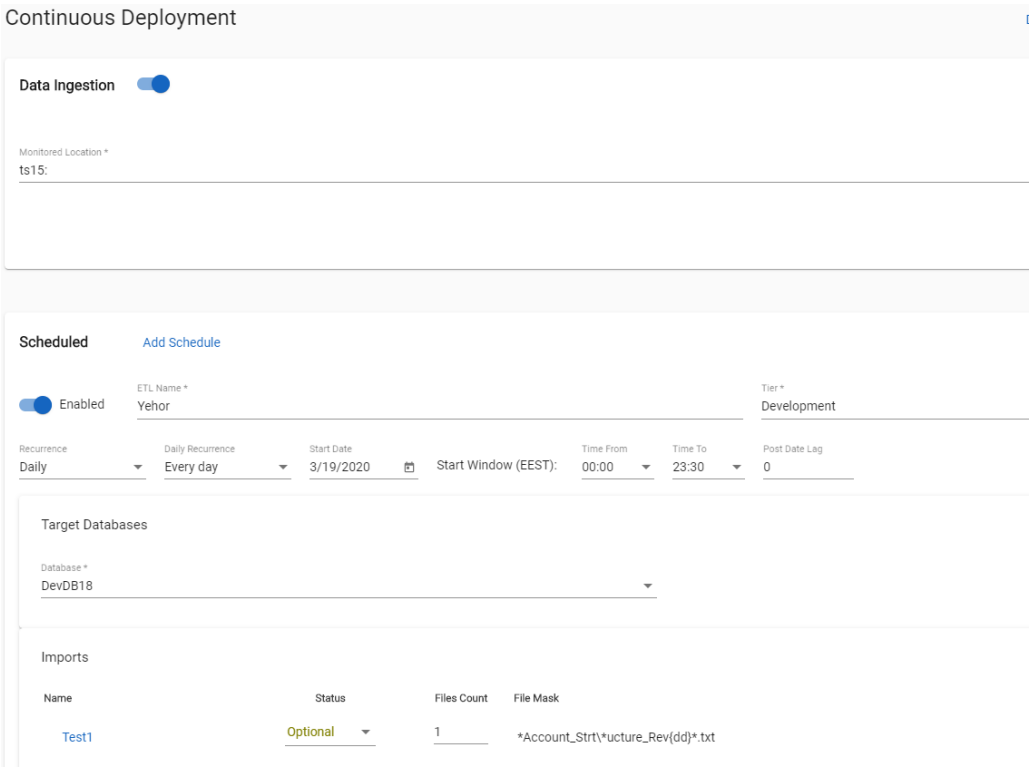

Рисунок 4.2 – Головна сторінка

Головна сторінка дає користувачу доступ до всіх основних функцій системи, а саме до налаштування необхідних йому імпортів.

При натисненні на кнопку “Imports” користувачу відкриється сторінка із списком всіх його імпортів (див. рис. 4.3). На цій сторінці користувач може створити новий імпорт або відредагувати існуючі.

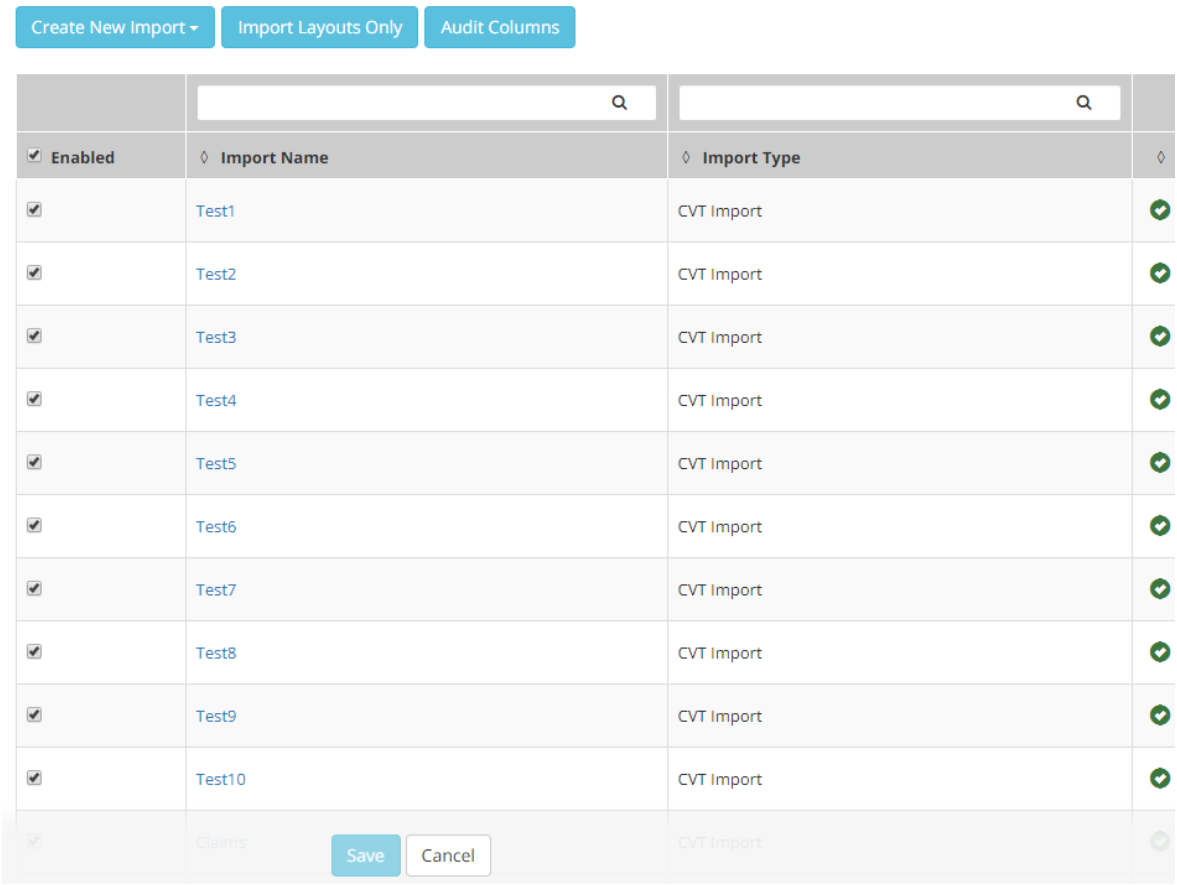

Рисунок 4.3 – Сторінка зі списком імпортів

Після того, як користувач натисне кнопку створення нового імпорту або ж редагування існуючого – відкриється сторінка з деталями імпорту (див. рис. 4.4). На цій сторінці користувач зможе задати необхідні параметри імпорту, такі як:

- ім’я імпорту;
- список масок файлів;
- очікуєму кількість файлів;
- ряд налаштувань щодо структури файлів, такі як строка заголовку або закінчення;
- ряд налаштувань щодо валідації файлів.

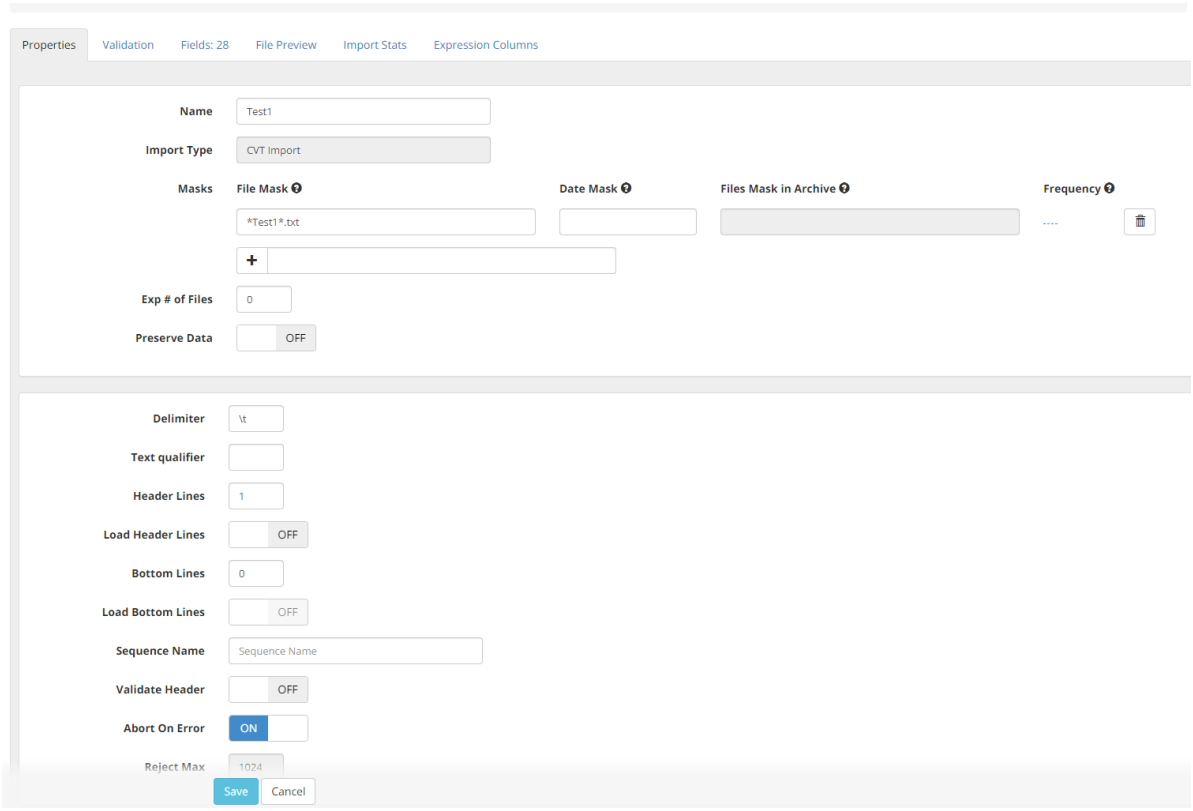

Рисунок 4.4 – Сторінка деталей конкретного імпорту

Після збереження необхідних налаштувань імпортів, користувач зможе завантажувати необхідні файли у директорію, яка мониториться сервісом (попередньо налаштована директорія) та спостерігати за процесом на екрані Ingestion монітора (див. рис. 4.5).

На цьому екрані виводяться дані щодо усіх файлів, які біли знайдені, а саме:

- ім’я файлу;
- шлях до файлу;
- дата завантаження;
- розмір файлу;
- його статус (що з ним відбувається у данний момент);
- списко імпортів, яким він підійшов;
- тип імпортів;
- результати валідації.

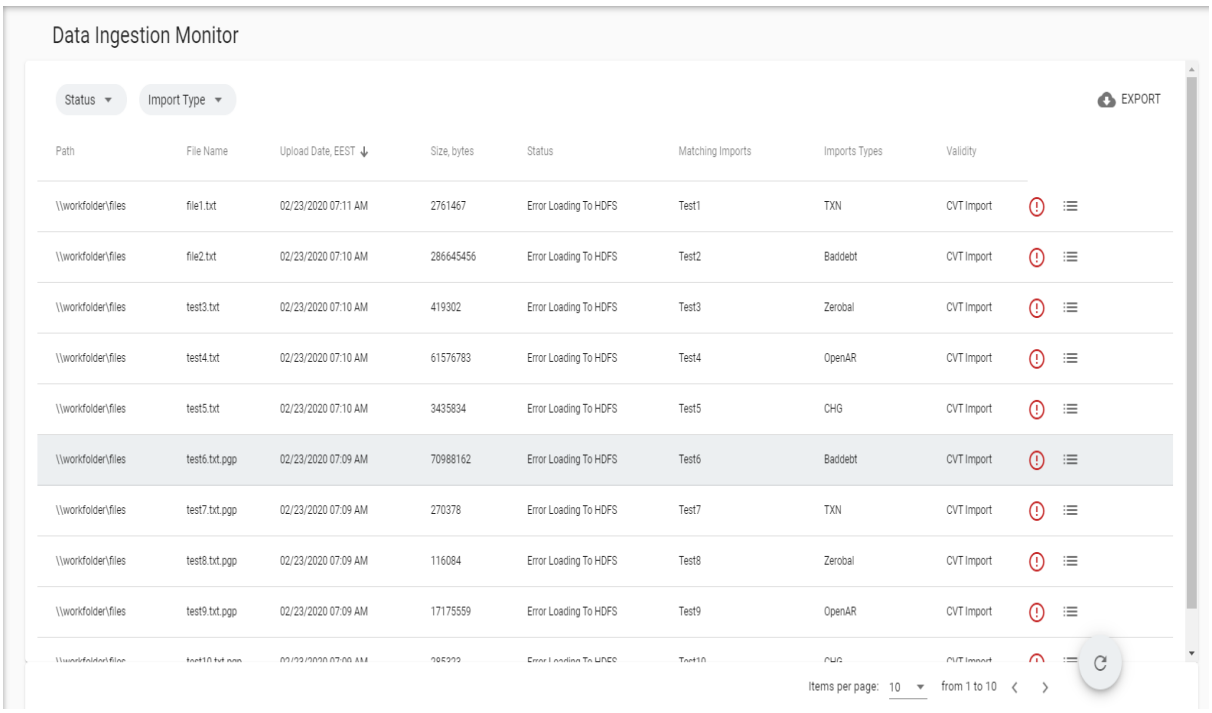

Рисунок 4.5 – “Ingestion Monitor” сторінка

Після того, як файли будуть успішно завантажені, користувачу необхідно буде подивитися на інформації по цих файлах. Вона буде надана у виді таблиць з даними. Користувач може перейти на сторінку з даними (див. рис. 4.6).

В залежності від типу імпорту та розширення отриманих файлів тип таблиць буде відрізнятися. На цій сторінці видно:

- ім’я створеної таблиці;
- валідаційну колонку;
- тип створеної таблиці;
- тип стійкості;
- використовувалась паралельна обробка чи ні;

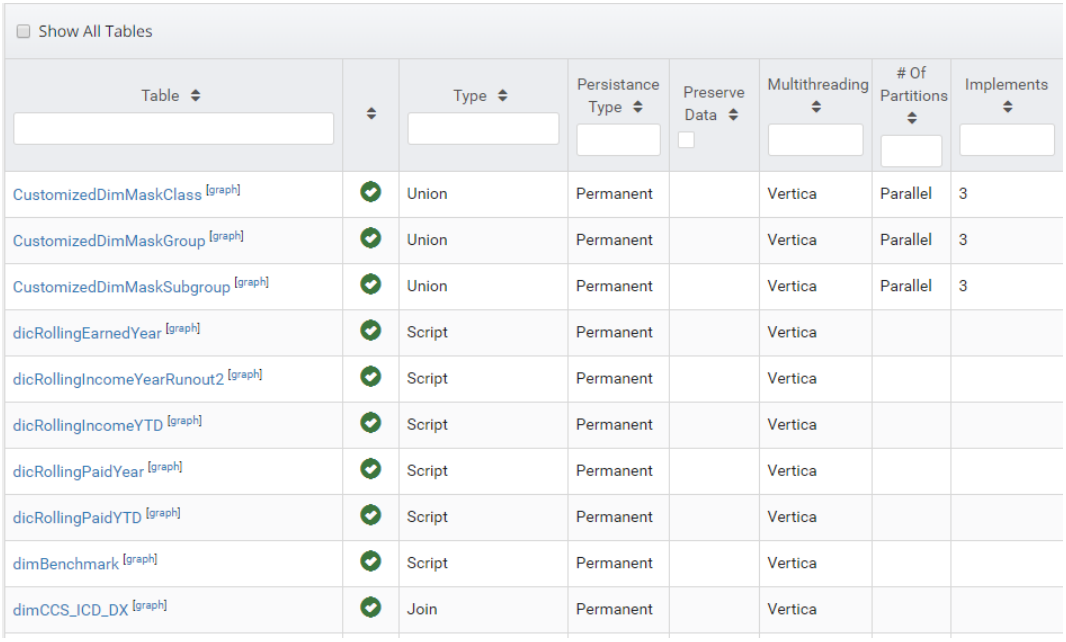

Рисунок 4.6 – Сторінка зі списком таблиць

Щоб подивитися результати, користувачу необхідно перейти у таблицю – там він зможе побачити сторінку з деталями заданої таблиці (див. рис. 4.7.).

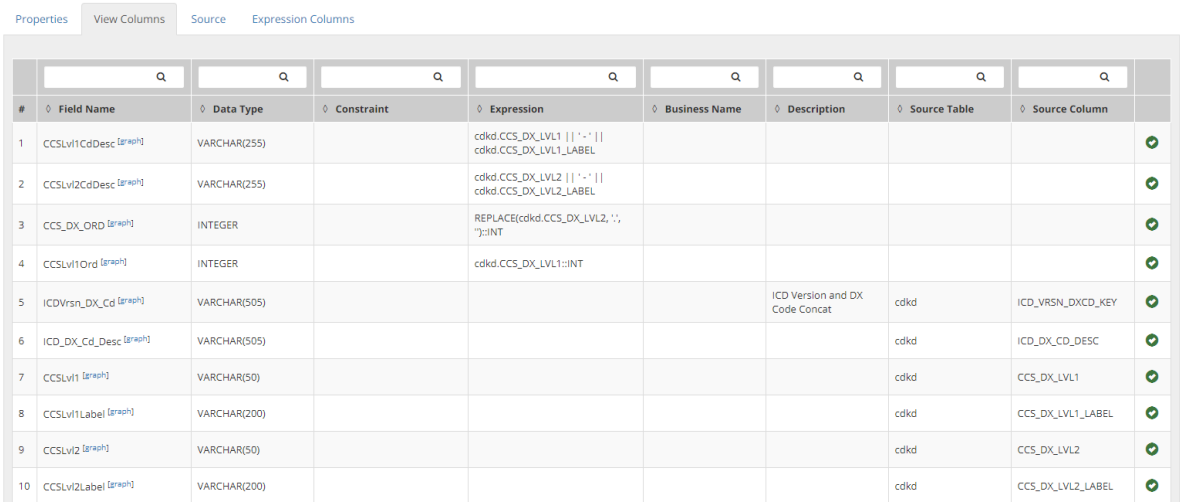

Рисунок 4.7 – Сторінка з деталями таблиці

Окрім наведених функцій користувач також може очистити список старих завантажених файлів, вибравши відповідний пункт у меню, що відкривається при натисканні на значок користувача на верхній панелі (див. рис. 4.10).

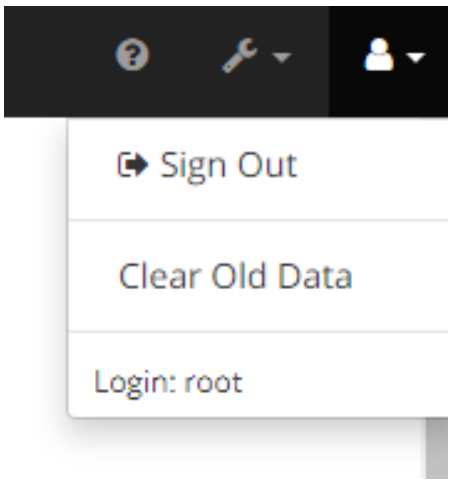

Рисунок 4.10 – Меню користувача

В цьому меню також знаходиться кнопка виходу з поточного акаунту.

Слід також зазначити, що завантаження веб-сторінок є повністю автоматичним завдяки двосторонньому зв’язуванню об’єктів в Angular. Це дозволить користувачу завжди бачити актуальну інформацію на своєму моніторі та стежити за станом списку товарів без потреби постійно перевантажувати сторінку в браузері.

ВИСНОВКИ

В процесі роботи був проведений аналіз предметної галузі програм для аналізу та обробки великих обсягів даних. Були досліджені основні фреймворки та технології, що застосовуються у сучасних системах аналізу та обробки даних, а саме:

– фреймворки для обробки та аналізу великих даних (Apache Spark, Hadoop, Flink);

– алгоритми кластеризації даних;

Це дослідження відіграло важливо роль у виборі ефективних методів обробки та аналізу великих обсягів даних, а також побудови розширюваного програмного продукту із можливістю оброблювати та аналізувати файли великих обсягів.

У результаті виконання роботи був розроблений програмний продукт, що дозволяє на основі вхідних файлів ідентифікувати, обробити та візуалізувати вхідні дані користувача у найкоротший термін. Система базується на клієнт-серверній архітектурі, тому складається з двох окремих частин: сервера та веб-клієнта.

Сервер було розроблено з використанням мови програмування C# у середовищі Microsoft Visual Studio 2019 Professional, клієнтська частина була написана переважно на мові TypeScript із застосуванням фреймворку Angular. Сервер розроблявся з використанням фреймворку ASP.NET MVC 5. Для бази даних була застосована СКБД MongoDB.

В результаті виконання роботи було виконано наступні задачі:

- розроблено сервер з сервісами, які надають змогу працювати одночасно різним типам клієнтів з системою;

- розроблено веб-клієнт з достатнім функціоналом та зручним інтерфейсом.

В результаті розробки поставлену задачу було цілком виконано. Програмний продукт має зрозумілий інтерфейс для користувачів, не викликає труднощів з відправкою даних на сервер.

Під час виконання роботи були набуті навички з застосування фреймворків глибинного навчання і технологій обробки та аналізу великих обсягів даних. Також були поглиблені знання стосовно проектування баз даних, вивчено і втілено багато нового стосовно розробки клієнтських засобів на мові Typescript.

Програмний продукт навіть після реалізації усіх поставлених задач, для комерційного запуску має дороблюватися. Перш за все, необхідно доробити додатковий функціонал та створити комерційну складову проекту.

У подальшому планується розширити функціональну складову розробленої системи шляхом додавання наступних можливостей:

- налаштувати механізм валідаціїї файлів на основі подальших вподобань користувача;

- ввести можливість будувати многомірні куби на основі отриманих даних;

- розробити утиліту для додаткової обробки специфічних даних.

Другорядним завданням стоїть реалізація клієнтів для різних мобільних платформ (iOS, OS Android, Windows Phone), це дозволить покрити більшість сучасних смартфонів та планшетів. Також, необхідно доробити локалізацію.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Big Data Timeline - Series of Big Data Evolution [Електронний ресурс]. – Режим доступу : https://www.dezyre.com/article/big-data-timeline-series-of-big-data-evolution/160
2. Big data preprocessing: methods and prospects [Електронний ресурс]:. – Режим доступу : https://bdataanalytics.biomedcentral.com/articles/10.1186/s41044-016-0014-0
3. Дударь З.В., Иванилов А.А. / Применение элементов теории баз данных к разработке средств декомпозиции предикатов // Системи управління, навігації та зв'язку: Зб. наук. праць. – К.: Центральний НДІ навігації і управління, 2007. – № 3. – С. 52-58..
4. Apache Hadoop [Електронний ресурс]:. – Режим доступу : https://hadoop.apache.org/
5. How do Hadoop and Spark Stack Up? [Електронний ресурс]. – Режим доступу : https://logz.io/blog/hadoop-vs-spark/
6. Features of Apache Spark – Learn the benefits of using Spark [Електронний ресурс]:. – Режим доступу: https://data-flair.training/blogs/apache-spark-features/
7. Hadoop vs Spark [Електронний ресурс]. – Режим доступу : https://www.educba.com/hadoop-vs-spark/
8. Basics of Map Reduce Algorithm. [Електронний ресурс]. – Режим доступу: https://www.thegeekstuff.com/2014/05/map-reduce-algorithm/
9. MapReduce - Algorithm [Електронний ресурс].– Режим доступу : https://www.tutorialscampus.com/tutorials/map-reduce/algorithm.htm
10. Big Data Processing 101: The What, Why, and How [Електронний ресурс]. – Режим доступу : https://www.dataversity.net/big-data-processing-101/
11. Distortion Invariant Object Recognition [Електронний ресурс] – Режим доступу http://ieeexplore.ieee.org/document/210173/

12. Kuzochkina A., Shirokopetleva M., Dudar Z. / Analyzing and Comparison of NoSQL DBMS // International Scientific and Practical Conference «Problems of Infocommunications. Science and Technology» (PIC S&T`2018), October 9-12, 2018. - Str. 560-565
13. Image Analysis With Convolutional Neural Networks [Електронний ресурс] – Режим доступу http://ieeexplore.ieee.org/document/554195/
14. An Overview of Principal Component Analysis [Електронний ресурс] – Режим доступу https://www.scirp.org/pdf/JSIP_2013101711003963.pdf
15. PCA vs LDA vs T-SNE — Let's Understand the difference between them [Електронний ресурс] – Режим доступу https://medium.com/analytics-vidhya/pca-vs-lda-vs-t-sne-lets-understand-the-difference-between-them-22fa6b9be9d0
16. The analysis of the existing approaches to object recognition [Електронний ресурс] – Режим доступу http://developers-club.com/posts/238129/
17. Аналіз існуючих підходів до розпізнавання об'єктів [Електронний ресурс] – Режим доступу http://it-ua.info/news/2014/09/25/analz-snuyuchih-pdhodv-do-rozpznavannya.html
18. Ojala, T., Pietikainen, M., Maenpaa, T.: Multiresolution gray-scale and rotation invariant texture classification with local binary patterns, IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume 24, Issue 7, 971–987 (2002)
19. Фаулер, М. UML. Основы [Текст] : пер. с англ. А.: Петухов/ М. Фаулер, К. Скотт. - СПб.: Символ, 2017. - 184 с.
20. A. Rabotiahov, O. Kobylin, V. Lyashenko. / Bionic image segmentation of cytology samples method // Telecommunications and Computer Engineering (TCSET), pp. 665–670, 2018
21. A. Rabotiahov, O. Kobylin, V. Lyashenko. / Bionic image segmentation of cytology samples method // Telecommunications and Computer Engineering (TCSET), pp. 665–670, 2018
22. Мюлер Джордж Р. Проектирование баз данных и UML – Москва: Лори, 2013. – 432 с.

23. Ponomarenko O.A.,Shirokopetleva M.S., Dudar Z. / Analysis of time series forecasting methods // SCIENCE, RESEARCH, DEVELOPMENT #11. TECHNICS AND TECHNOLOGY. Rotterdam (The Netherlands) 29.11.2018 - 30.11.2018 (30.11.2018) - 2018. Str.59-61
24. Пугачев, С. Разработка приложений для Windows на языке C# [Текст] / С.В. Пугачов, А.М. Шеріев, К.А. Кичинський. – СПб.: БХВ-Петербург, 2013. – 416 с.
25. Рихтер, Дж. CLR via C# Программирование на платформе Microsoft .NET Framework 4.0 на языке C# [Текст] / Дж. Рихтер, пер. с англ. Радченко И., Рузмайкина И. – СПб. Питер, 2013. – 428 с..
26. Шилдт, Г. C#4.0: Полное руководство [Текст] / Г. Шилдт. ; пер. с англ. И. Берштейн. – М.: ООО «И.Д. Вильямс», 2011. – 356 с.