МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СТРУКТУРНИЙ ФАХОВИЙ КОЛЕДЖ ЕКОНОМІКИ ТА

ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ПрАТ «ПВНЗ «ЗІЕІТ»

Циклова комісія з інформаційних технологій

ДО ЗАХИСТУ ДОПУЩЕНА
Голова циклової комісії,
спеціаліст в/к
С.О Сабанов_____
(підпис)

«___»_____ 2023 р.

КВАЛІФІКАЦІЙНА ВИПУСКНА РОБОТА

Тема: РОЗРОБКА ХМАРНОГО СХОВИЩА ДАНИХ З ВИКОРИСТАННЯМ АЛГОРИТМІВ КЛАСТЕРИЗАЦІЇ

Виконала ст..гр. - КІ-19/К _____ А.С.Погребняк
(підпис) (ініціали та прізвище)

Науковий керівник

к.т.н., доц. _____ Д.Г.Медведев
(підпис) (ініціали та прізвище)

Запоріжжя

2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СТРУКТУРНИЙ ФАХОВИЙ КОЛЕДЖ ЕКОНОМІКИ ТА
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ПрАТ «ПВНЗ «ЗІЕІТ»
Циклова комісія з інформаційних технологій

ЗАТВЕРДЖЕЮ

Голова циклової
комісії,спеціаліст в/к
С.О. Сабанов_____
(підпис)
« ____» _____ 2023 р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ ВИПУСКНУ РОБОТУ

Студентці гр.КІ-119/К , спеціальності Комп'ютерна інженерія

Погребняк Анастасії Сергіївні

1.Тема: Розробка хмарного сховища даних з використанням алгоритмів кластеризації

Затверджена наказом по інституту № 09.2-16 від 23.03.2023 р.

2. Термін здачі студентом закінченої роботи 14.06.2023 р.

3. Перелік питань, що підлягають розробці:

– розглянути поняття кластерного аналізу;

– провести аналіз існуючих сховищ даних;

– розглянути особливості кластерного аналізу та дослідження їх методів;

– розробити веб-додаток за допомогою вибраного алгоритму кластеризації для зберігання даних в хмарному сховищі.

4. Календарний графік підготовки кваліфікаційної роботи

№ етапу	Зміст	Терміни виконання	Готовність по графіку %, підпис керівника	Підпис керівника про повну готовність етапу, дата
1	Збір практичного матеріалу за темою кваліфікаційної випускної роботи			
2	І атестація І розділ кваліфікаційної випускної роботи			
3	ІІ атестація ІІ розділ кваліфікаційної випускної роботи			
4	ІІІ атестація ІІІ розділ кваліфікаційної випускної роботи			
5	Перевірка кваліфікаційної випускної роботи на оригінальність			
6	Доопрацювання кваліфікаційної випускної роботи, підготовка презентації, отримання відгуку керівника і рецензії			
7	Попередній захист кваліфікаційної випускної роботи			
8	Подача кваліфікаційної випускної роботи на кафедру			
9	Захист кваліфікаційної випускної роботи			

Дата видачі завдання: _____2023 р.

Керівник кваліфікаційної випускної роботи Д.Г.Медведев

_____ _____
(підпис) (прізвище та ініціали)

Завдання отримав до виконання А.С.Погребняк

_____ _____
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Кваліфікаційна випускна робота містить 49 стор., 14 рис., 16 використаних джерел.

Об’єкт дослідження: оптимізація даних в хмарному сховищі.

Предмет дослідження: кластеризація даних за допомогою нейромежевих алгоритмів.

Мета дослідження - розробка кластеризатора даних.

Задачами дослідження є: проаналізувати існуючі методи та алгоритми для розробки кластеризатора; вибрати оптимальний метод роботи з даними.

В роботі виконано аналіз існуючих методів та алгоритмів кластеризації, розроблено сховище даних у вигляді типового файлообмінника. Створено веб-додаток, який реалізовує розроблені алгоритми.

СХОВИЩЕ ДАНИХ, КЛАСТЕРИЗАЦІЯ, АЛГОРИТМИ, HTML5, JAVASCRIPT, PHP, AJAX, MYSQL.

ЗМІСТ

ВСТУП

Наразі у всьому світі незворотною є тенденція використання мережі Інтернет з метою внесення змін у існування людини і людства не тільки технічного та технологічного напрямів, а і використання цих можливостей для більш продуктивного та комфортного життя. За зростаючих постійно можливостей швидкості та об’ємів передачі даних в мережевому середовищі об’єктивно спостерігається необхідність збереження великих обсягів даних користувачів мережі Інтернет, у тому числі і стільникового зв’язку. Локальне збереження інформації на жорсткому диску комп’ютера чи переносному носії інформації стає технічно архаїчним. У світі опановується для збереження великої кількості даних з одночасним швидким до них доступом використання хмарних технологій. Для вирішення проблеми передачі та зберігання даних користувачів в мережі Інтернет використовують кластеризацію, кластер ний аналіз.

Зростання обсягу даних користувачів в мережі Інтернет вимагає шукати нові підходи до їх класифікації. Кластеризація даних дає можливість вирішення цієї задачі, тому наразі є дуже актуальною темою досліджень. До того ж, за великого обсягу інформації для її аналізу виникає необхідність враховувати все більшу кількість параметрів, тому досліджуються, розробляються та застосовуються методи, які націлені саме на класифікацію багатовимірних даних. Ці комп'ютерні технології виконують функцію інтелектуального аналізу і даних набувають все більшої популярності. Наука та пошук не стоять на місці, динамічно з’являються нові ідеї з області ІТ, теорії баз даних і т.інш. Вирішення задачі концентрації інформації в електронних сховищах (data mining) вимагає створення систем її автоматичної обробки та аналізу. Створення будь-якої класифікації передбачає необхідність ідентифікувати дані в масиві, знаходження в них систематизовані закономірності.

Для досягнення цієї мети можна використовувати різні методи, включаючи алгоритми кластеризації та відносно недавно застосовувані методи нейронних мереж, та методи обробки нечітких мереж. Кластеризація може бути застосована для розв'язання різних задач, таких як обробка зображень, класифікація, тематичний аналіз колекцій документів та побудова репрезентативної вибірки.

Однією з переваг нейромережевих методів аналізу є їх здатність до ітераційної обробки даних та величезний потенціал у паралелізмі алгоритмів розв'язання задач. Крім того, штучні нейронні мережі з самого початку орієнтовані на обробку багатовимірних даних. Кластеризація може бути використана для вирішення наступних задач:

1. Обробка зображень.
2. Класифікація.
3. Тематичний аналіз колекцій документів.
4. Побудова репрезентативної вибірки.

Перевага нейромережевих методів аналізу перед традиційними полягає в тому, що методи, які використовують нейронні мережі, поєднують переваги ітераційний і величезний потенціал в паралелізм алгоритмів розв'язання задач. Крім того, штучні нейронні мережі з самого початку орієнтовані на обробку багатовимірних даних.

Мета кваліфікаційної випускної роботи - розробити кластеризатор даних.

Об’єктом досліджень є оптимізація даних в хмарному сховищі.

Предмет дослідження: кластеризація даних за допомогою нейромежевих алгоритмів.

Задачі дослідження:

- розглянути поняття кластерного аналізу;
- провести аналіз існуючих сховищ даних;
- розглянути особливості кластерного аналізу та дослідження їх методів;

- розробити веб-додаток та за допомогою вибраного алгоритму кластеризації проаналізувати існуючі методи та алгоритми для розробки кластеризатора;
- вибрати оптимальний метод роботи з даними.

РОЗДІЛ 1. ПОНЯТТЯ КЛАСТЕРНОГО АНАЛІЗУ

1.1. Вивчення загальних відомостей

Основна мета процедури кластеризації даних або кластерного аналізу полягає в поділі об'єктів на відповідні групи, відомі як кластери. Кожний кластер складається з об'єктів, які поєднуються за певним критерієм, при цьому об'єкти з різних кластерів повинні якомога більше відрізнятися один від одного. У контексті кластеризації, об'єктом може бути будь-який вид даних. Важливо зрозуміти різницю між кластеризацією та класифікацією: коли ми говоримо про кластеризацію, ми не маємо попередньо визначеного списку груп - він формується в процесі роботи алгоритму(рис.1.1).

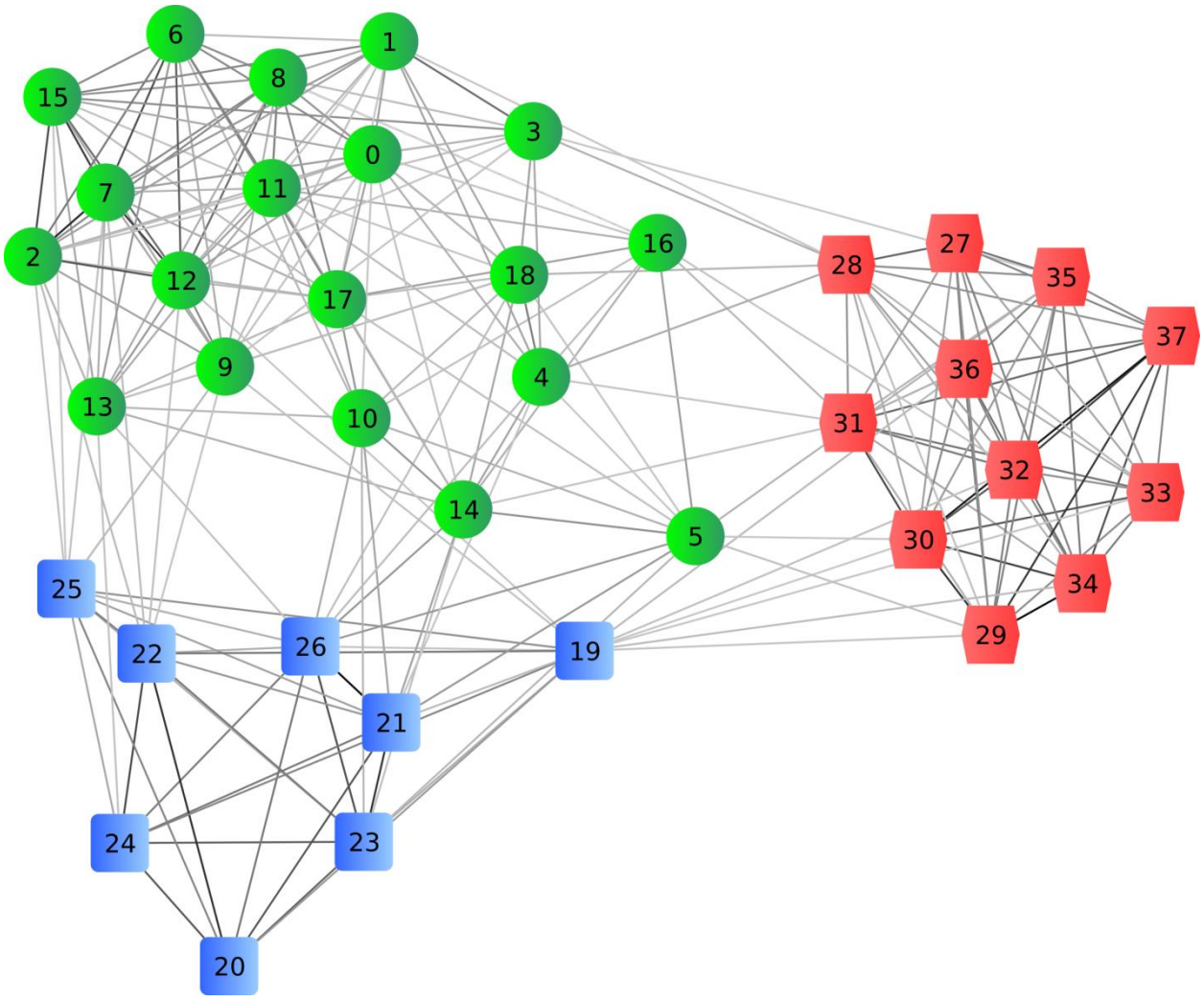


Рис.1.1. Групи кластерів [1,2,3]

Кластеризація має багатий спектр застосувань, що охоплює не лише сферу інформатики, але й інші дисципліни. Це можуть бути такі сфери застосування, як аналіз даних, екстракція та пошук інформації, а також групування та ідентифікація об'єктів.

Застосування нейронних мереж для класифікації у кластеризації є одним з найважливіших. Завдання класифікації - це порівняння зразків до одного з декількох множин, які не перетинаються. Прикладом таких задач можуть бути багатофакторні блоки даних, наприклад, результати медичних оглядів людини (хворої чи здорової), з яких необхідно для прийняття рішення виділити напрямок класифікації та аналізу - результат захворювання, вирішення проблем аналізу та співставлення великих обсягів інформації у криміналістиці.

На сучасному рівні дослідження та розвитку мережевих технологій нейронні мережі представляються як найбільш ефективні способи класифікації, тому що дозволяють аналізувати, систематизувати та класифікувати величезні обсяги залежних елементів..

Виділяють дві основні класифікації алгоритмів кластеризації - ієрархічні і плоскі та чіткі і нечіткі.

Існують ієрархічні алгоритми (таксономічні), що дозволяють створювати не просто вибірки, а системи вибірок непересічних кластерів. Основою служить вся вибірка, від якої відростають більш деталізовані кластери. З іншого боку, плоскі алгоритми конструюють єдине розбиття об'єктів на кластери.

Згідно з класифікацією, чіткі (непересічні) алгоритми формуються з урахуванням конкретного завдання - кожен об'єкт вибірки відповідає номеру лише одного кластера. Нечіткі (пересічні) алгоритми присвоюють кожному

об'єкту не номер одного кластера, а набір числових значень, що визначають ступінь відносин об'єкта до кластерів, з відповідною ймовірністю.

Серед ієрархічних алгоритмів кластеризації визначаються два основних типи: знижувальні, які працюють за принципом "зверху-вниз", та зростаючі алгоритми. У знижувальних алгоритмах усі об'єкти спочатку збираються в один кластер, який потім дробиться на більш мелкі. У зростаючих алгоритмах ситуація обертається: окремі кластери об'єднуються з ростом вибірки. Зростаючі алгоритми більш поширені, оскільки вони використовують системний підхід до кожного кластерного об'єкту та їх зв'язків. Результати таких алгоритмів зазвичай представляють у вигляді дерева - дендрограми [3].

Створення алгоритму вимагає визначення відстаней між кластерами. Найчастіше використовуються методи одиночного або повного зв'язку [1, 3]. Одним з недоліків ієрархічних алгоритмів є складність, що виникає через квадратичну помилку.

Загалом, метою кластеризації є обґрунтоване і оптимальне розбиття об'єктів на групи. Оптимальність розбиття визначається як потреба мінімізувати середньоквадратичну помилку розбиття. "Центр мас" кластера j (точка із середніми значеннями характеристик для даного кластера) відіграє важливу роль у цьому процесі.

$$e^2(X,L) = \sum_{j=1}^{K} \sum_{i=1}^{n_j} \| x_i^{(j)} - c_j \|^2$$

Алгоритми з квадратичною помилкою належать до плоского типу алгоритмів [3]. Найбільш поширеним з них є метод k-середніх. Він вимагає визначення кількості найбільш віддалених один від одного кластерів та побудову алгоритму поетапно. Робота алгоритму включає такі етапи:

-вибір k точок, які служатимуть початковими "центрами мас" кластерів;

-присвоєння кожного об'єкту кластеру з найближчим "центром мас";

-перерахунок "центрів мас" кластерів відповідно до їх поточного складу [3].

В якості критерію сигналу зупинки роботи алгоритму зазвичай вибирають мінімальне значення середньої квадратичної помилки. Аналогічно можливо зупиняти роботу алгоритму, якщо на наступному кроці не було об'єктів, що перемістилися з кластера в кластер.

Недоліком даного алгоритму вважається доля суб’єктивності при визначенні кількості кластерів для розбиття.

Дійсно, одним з найбільш популярних алгоритмів нечіткої кластеризації є алгоритм c-середніх (c-means). Цей алгоритм є модифікацією методу k-середніх. Основні кроки роботи алгоритму [3] включають:

1. Вибір початкового нечіткого розбиття n об'єктів на k кластерів, шляхом вибору матриці приналежності U розміру n x k.

2. Пошук критерію нечіткої помилки за допомогою матриці U. Відмітимо, що ck є "центром мас" нечіткого кластера k.

$$E^2(X,U) = \sum_{i=1}^{N} \sum_{k=1}^{K} U_{ik} \| x_i^{(k)} - c_k \|^2$$

3. Перегрупування об'єктів з метою зменшення значення критерію нечіткої помилки та придання йому відповідності заченню нечіткої помилки.

4. Повторення циклу до тих пір, поки зміни матриці U не стануть незначними.

Цей алгоритм не може бути застосований, якщо заздалегідь невідомо число кластерів, або якщо потрібно однозначно віднести кожен об'єкт до одного кластера.

Суть алгоритмів на основі графів полягає в тому, що вибірка об'єктів представляється у вигляді графа G = (V, E), вершини якого відповідають об'єктам, а ребра мають вагу, рівну "відстані" між об'єктами. Перевагами графових алгоритмів кластеризації є наочність, відносна простота реалізації і можливість внесення різних удосконалень, заснованих на геометричних міркуваннях. Основні алгоритми включають алгоритм виділення зв'язкових компонент, алгоритм побудови мінімального покриваючого дерева і алгоритм пошарової кластеризації [9].

Дійсно, алгоритм виділення зв'язкових компонент в графі є важливою методикою у кластерному аналізі. Він включає визначення вхідного параметра R і видалення всіх ребер в графі, для яких "відстані" перевищують R. Таким чином, лише найближчі пари об'єктів залишаються з'єднаними.

Головне завдання алгоритму - вибрати таке значення R, що лежить у діапазоні всіх "відстаней", при якому граф "розвалиться" на кілька зв'язкових компонент. Кожна з цих компонент відповідає окремому кластеру [3].

Щодо вибору параметра R, зазвичай будується гістограма розподілів попарних відстаней. У випадках, коли дані мають ясну кластерну структуру, гістограма має два піка: один відповідає середнім відстаням в межах кластерів, а інший - відстаням між кластерами. Параметр R вибирається з області мінімуму між цими піками. Однак, керування кількістю кластерів за допомогою порогової відстані може бути викликом [3, 9].

Алгоритм формування мінімального охоплюючого дерева спершу створює на графі дерево з найменшою сумарною вагою ребер, а далі поступово вилучає ребра з найвищим ваговим значенням. На рис. 1.4 ілюструється мінімальне охоплююче дерево, створене для дев'яти елементів.

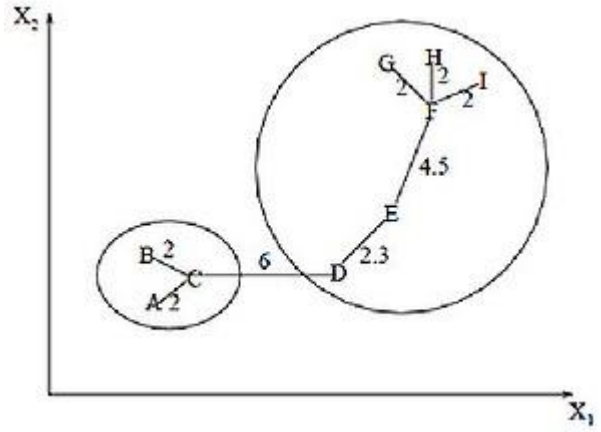


Рис. 1.4. Дерево, отримане для дев'яти об'єктів дерево [3]

Усуваючи зв'язок з маркуванням CD, що має довжину в 6 одиниць (ребро з найбільшою дистанцією), ми формуємо два кластери: {A, B, C} та {D, E, F, G, H, I}. Другий з них може бути подальше поділений на два окремі кластери шляхом вилучення ребра EF з довжиною 4,5 одиниць.

За умов використання ієрархічних алгоритмів постають задачі об'єднання між собою кластерів та розрахунку «відстані» між ними. Застосовуються певні підходи, наведемо основні п’ять найбільше використовуваних з них.

1. На початку - одиночний зв'язок використовується для встановлення найменших відстаней, які визначаються як відстань між двома найближчими елементами (або найближчими сусідами) від різних кластерів. Згодом, отримані кластери об'єднуються в ланцюгові структури.

2. Застосування повного зв'язку вимагає встановлення найбільших відстаней між об'єктами. Тут відстань між кластерами визначається як максимальна відстань між будь-якими двома об'єктами від різних кластерів. Цей підхід добре працює, коли об'єкти належать до віддільних груп. Але

якщо кластери мають подовжену форму, або їх природний тип є "ланцюговий", цей метод не підходить.

3. В методі визначення невзважених попарних середніх, відстань між двома різними кластерами вираховується як середнє значення відстаней між усіма парами об'єктів у цих кластерах. Цей метод є ефективним, коли об'єкти створюють різні групи, але він також добре працює і для протяжних кластерів ланцюгового типу.

4. Метод виваженого попарного середнього практично не відрізняється від методу невиваженого попарного середнього. Однак, у цьому методі вага кластера (тобто кількість об'єктів у кластері) береться до уваги при виконанні обчислень. Тому цей метод має застосовуватись, коли очікується, що розміри кластерів будуть нерівними.

5. В зваженому центроїдному методі, як і в попередньому, використовуються ваги для врахування різниці в розмірах кластерів. Якщо існують або очікується значні відмінності в розмірах кластерів, цей метод є більш підходящим ніж попередній [3].

Щоб розпізнати "схожість" між об'єктами, спочатку треба сформувати вектор особливостей для кожного об'єкта. Зазвичай це набір чисельних даних, таких як ріст та вага людини. Проте, існують алгоритми, що можуть обробляти якісні характеристики.

Коли ми визначаємо вектор характеристик, ми можемо здійснити нормалізацію, таким чином всі елементи вноситимуть однаковий вклад в розрахунок "відстані". Під час нормалізації, всі значення відповідають певному діапазону, наприклад, [-1, -1] або [0, 1].

Остаточно, для кожної пари об'єктів ми вимірюємо "відстань" між ними, яка представляє ступінь схожості. Існує багато метрик для цього, ось декілька основних:

1. Евклідова відстань.

Ця відстань - найпоширеніша функція для визначення відстані. Вона відображає геометричну відстань в багатовимірному просторі:

$$\rho(x, x') = \sqrt{\sum_{i}^{n} (x_i - x_i')^2}$$

Квадрат Евклідової відстані.

Використовується для надання більшої ваги об'єктам, які значно віддалені один від одного.

$$\rho(x, x') = \sum_{i}^{n} (x_i - x_i')^2$$

2. Манхеттенська відстань (відстань в міських кварталах).

Ця відстань представляє середню різницю по координатам. В більшості випадків, цей показник відстані дає подібні результати до Евклідової відстані.

$$\rho(x, x') = \sum_{i}^{n} | x_i - x_i' |$$

3. Відстань Чебишева.

Ця міра відстані може бути корисною, коли потрібно відрізнити два об'єкти як "різні", якщо вони відрізняються хоча б по одній з координат.

$$\rho(x, x') = \max(| x_i - x_i' |)$$

4. Ступенева відстань.

Використовується, коли потрібно збільшити або зменшити вагу, що відноситься до виміру, за яким відповідні об'єкти значно відрізняються.

$$\rho(x, x') = \sqrt[r]{\sum_{i}^{n} (x_i - x_i')^p}$$

1.2. Вивчення поняття «хмарні сховища» даних

Хмарне сховище даних – онлайн-сховище, в якому дані зберігаються на серверах, що можуть знаходитись в різних частинах планети та об'єднуватися в єдину мережу. В моделі зберігання даних на власних серверах (або локально), зберігати дані за допомогою хмарних технологій значно практичніше. Завдяки цьому, тимчасові файли можна зберігати на окремо виділеному хмарному сховищі, таким чином економити місце на власному пристрої. Є можливість збереження і обробки в залежності від призначення або того обо іншого хмарного сховища в залежності від потреби користувачів. Наприклад, GoogleDrive має функцію онлайн редагування таблиць та документів, відтворення аудіо чи відео файлів.

Розглянемо та здійснимо аналіз сервісів для зберігання даних у хмарі .

Google Drive – це одне з найбільш відомих хмарних сховищ, яке надає користувачам можливість зберігати свої дані на серверах у хмарі та обмінюватися ними з іншими користувачами в мережі. Google Drive підтримує зберігання різноманітних типів файлів, включаючи документи, фотографії, музику, відео та інше, але не є досить оптимізованим під специфічні потреби користувача. Важливою складовою є швидкість обробки запитів, яку Google Drive не завжди може гарантувати. Недостатньо реалізовано сортування файлів за частотою їх використання. Кожен користувач має доступ до 15 ГБ безкоштовного місця.

Dropbox – це ще одне хмарне сховище, яке дозволяє зберігати дані на хмарних серверах і обмінюватися ними з іншими користувачами в інтернеті. Робота Dropbox базується на принципі синхронізації даних. Однією з головних переваг Dropbox є простота та інтуїтивність. Користувачі можуть легко ділитися папками, файлами або синхронізувати їх з потрібним пристроєм. Відмінність від інших сервісів полягає в тому, що Dropbox не

копіює цілком редаговані файли на сервер - лише змінену частину, яка попередньо стискається. Це значно покращує продуктивність роботи з Dropbox порівняно з аналогами.

Mega – (MEGAEncryptedGlobalAccess) – хмарний сервіс обміну файлами від Кіма Доткома (Kim Dotcom), засновника популярного Megaupload. Головна перевага Mega полягає в тому, що всі дані шифруються в браузері за допомогою алгоритму AES; користувачі мають можливість передавати файли одне одному в зашифрованому вигляді, при цьому всі дані зберігаються в хмарі. Ключі доступу до файлів не публікуються у відкритому доступі, а поширюються за принципом Friend-to-Friend. Але це стосується безпеки, а не оптимізації роботи з хмарним сховищем та економії часу.

1.3.Обгрунтування шляху вирішення задачі.

На сьогодні існує багато хмарних сховищ, але жоден з них не має такої цілі, як оптимізація даних користувача. Тому задачею кваліфікаційної роботи є розробити хмарне сховище з можливістю оптимізувати дані на сервері.

Файлообмінник з інтеграцією нечіткого кластеризатора даних, на основі характеристик: розмір файлу, дата створення, дата останнього відкриття, частота звернення.

Нечітка кластеризація відносить дані до конкретної групи, об’єктів і не показує ступінь схожості.

Чітка кластеризація відносить до деякого кластеру з певним числом схожості. Наприклад об’єкт А за нечіткою кластеризацією відноситься до кластеру К1 за схожістю 0,9, до К2 за схожістю 0,09. Кластерів може бути багато, та не завжди відомо їх кількість. Чітка кластеризація просто відносить до кластеру К1.

Для рішення даної задачі використовуємо чітку кластеризацію.

Задачі дослідження:

– розглянути поняття кластерного аналізу;

– провести аналіз існуючих сховищ даних;

– розглянути особливості кластерного аналізу та дослідження їх методів;

– розробитивеб-додаток та за допомогою вибраного алгоритму кластеризації оптимально зберігати дані в хмарному сховищі.

У першому розділі було розглянуто поняття «кластеризація», її вплив на сучасне життя, взаємодія з професіями та наукою. Розглянуто основні методи та алгоритми кластеризації. Визначили поняття схожості об’єктів, нормалізацію відстаней. Проведення аналізу існуючих хмарних сховищ та визначили їх переваги та недоліки дозволило виконати постановку задачі дослідження.

РОЗДІЛ 2. ПРОЕКТУВАННЯ ХМАРНОГО СХОВИЩА

2.1. Вибір даних для аналізу

Кластерний аналіз проводиться над даними з певними характеристиками (розмір файлу, дата створення, дата останнього відкриття, частота звернення, формат: відео, аудіо, текстовий документ та інші) та з певною кількістю набором вхідних даних.

Основні вимоги до даних - повнота і однорідність. Однорідність означає, що всі кластеризуємі об'єкти повинні бути однакового типу зі схожим набором властивостей. Повнота вимагає невичерпності набору даних для нормалізації результатів кластерного аналізу.

Завантажуємо файли та сортуємо їх за допомогою методів аналізу, використовуючи характеристики файлів. Це дозволяє отримати оптимізований список файлів, що знаходяться на файлообміннику. Ведемо статистику завантаження файлів, їх відкриття та інших маніпуляцій для визначення списку.

Дані повинні бути розташовані в зручному порядку для спрощення їх використання в майбутньому. Файли, які використовувалися рідше, розташовуються в кінці списку та мають невелику групу об'єктів. Використання кластеризації допоможе оптимізувати розташування файлів на файлообміннику і забезпечити більш ефективне використання ресурсів.

Для оптимального доступу до файлів проводимо кластеризацію на

основі вхідних даних, та за допомогою визначених характеристик таких як: частота завантаження файлів, час завантаження.

Оголошуємо функції: fuzzify, defuzzify, fuzzyClustering, saveFile

```
function fuzzify($value, $min, $max) {
    $low = max(min(1, ($max - $value) / ($max - $min)), 0);
    $high = 1 - $low;
    return ['low' => $low, 'high' => $high];
}
```

Функція fuzzify приймає значення, мінімальне та максимальне значення і виконує розмиття (fuzzification) для вхідного значення. Результатом є асоціативний масив зі значеннями "low" (низьке) та "high" (високе).

```
function defuzzify($rules) {
        $numerator = 0;
        $denominator = 0;
    foreach ($rules as $cluster => $value) {
        $numerator += $cluster * $value;
        $denominator += $value;
                }
        if ($denominator == 0) {
                return 0;
                }
    return $numerator / $denominator;
                }
```

Функція defuzzify приймає масив правил, який містить кластери та значення, і виконує розмиття (defuzzification) для вихідного значення. Результатом є числове значення.

```
function fuzzyClustering($file_size, $file_created,
$file_last_opened, $access_frequency) {
    $size_fuzzy = fuzzify($file_size, 0, 10 * 1024 * 1024);
    $created_fuzzy = fuzzify((time() - strtotime($file_created))
/ (60 * 60 * 24), 0, 365);
    $last_opened_fuzzy = fuzzify((time() -
strtotime($file_last_opened)) / (60 * 60 * 24), 0, 365);
    $access_frequency_fuzzy = fuzzify($access_frequency, 0, 10);

    $rules = [
        1 => min($size_fuzzy['high'], $created_fuzzy['low'],
$last_opened_fuzzy['low'], $access_frequency_fuzzy['high']),
        2 => min($size_fuzzy['high'], $created_fuzzy['low'],
$last_opened_fuzzy['high'], $access_frequency_fuzzy['high']),
        3 => min($size_fuzzy['low'], $created_fuzzy['low'],
$last_opened_fuzzy['high'], $access_frequency_fuzzy['high']),
        4 => min($size_fuzzy['high'], $created_fuzzy['high'],
$last_opened_fuzzy['low'], $access_frequency_fuzzy['low']),
        5 => min($size_fuzzy['low'], $created_fuzzy['high'],
$last_opened_fuzzy['low'], $access_frequency_fuzzy['low'])
        ];

        return defuzzify($rules);
    }
```

Функція fuzzyClustering приймає параметри файлу (розмір, дату створення, останнього відкриття та частоту доступу) і виконує розмиття для кожного з них. Потім вона застосовує правила для визначення кластера, який відповідає файлу. Результатом є числове значення кластера.

```
    function saveFile($file_path, $file) {
        if (move_uploaded_file($file['tmp_name'], $file_path)) {
            return $file_path;
```

```
        } else {
            return false;
        }
    }
```

Функція saveFile зберігає завантажений файл на локальний сервер і повертає шлях до файлу.

```
    function saveFileToDropbox($file_path, $file) {
        $app = new DropboxApp("client_id", "client_secret",
"access_token");
        $dropbox = new Dropbox($app);

        $file = fopen($file['tmp_name'], 'rb');
        $dropboxFile = $dropbox->upload($file, "/Fuzzy_Cluster/"
. $file_path, ['autorename' => true]);

        return $dropboxFile->getPathDisplay();
    }

    if ($_SERVER['REQUEST_METHOD'] === 'POST') {
        if (isset($_FILES['file'])) {
            $file = $_FILES['file'];
            $file_path = 'uploads/' . $file['name'];

            if ($_POST['storage'] === 'local') {
                $file_path = saveFile($file_path, $file);
            } else if ($_POST['storage'] === 'dropbox') {
                $file_path = saveFileToDropbox($file_path,
$file);
            }

            if ($file_path) {
                $file_size = $file['size'];
                $file_created = date("Y-m-d H:i:s");
                $file_last_opened = $file_created;
                $access_frequency = 0;
```

```
                $cluster = fuzzyClustering($file_size,
$file_created, $file_last_opened, $access_frequency);

                $db = new SQLite3('files.db');
                $db->exec("INSERT INTO files (name, path, size,
created, last_opened, access_frequency, cluster)
                           VALUES ('$file[name]', '$file_path',
$file_size, '$file_created', '$file_last_opened',
$access_frequency, $cluster)");

                echo json_encode(['message' => 'Файл успішно
завантажено!', 'success' => true]);
            } else {
                echo json_encode(['message' => 'Не вдалося
завантажити файл.', 'success' => false]);
            }
        }
```

Функція saveFileToDropbox зберігає завантажений файл на Dropbox за допомогою API Dropbox. Повертається шлях до файлу у Dropbox.

Умовний оператор перевіряє, чи отримано дані через метод POST.

Якщо отримано файли через параметр 'file', то виконуються наступні дії:

1. Створюється шлях для збереження файлу.

Якщо параметр 'storage' має значення 'local', файл зберігається на локальному сервері за допомогою функції saveFile.

Якщо параметр 'storage' має значення 'dropbox', файл зберігається на Dropbox за допомогою функції saveFileToDropbox.

Після завантаження групи об’єктів, можемо проводити сортування файлів за популярністю, нові об’єкти, старі об’єкти.

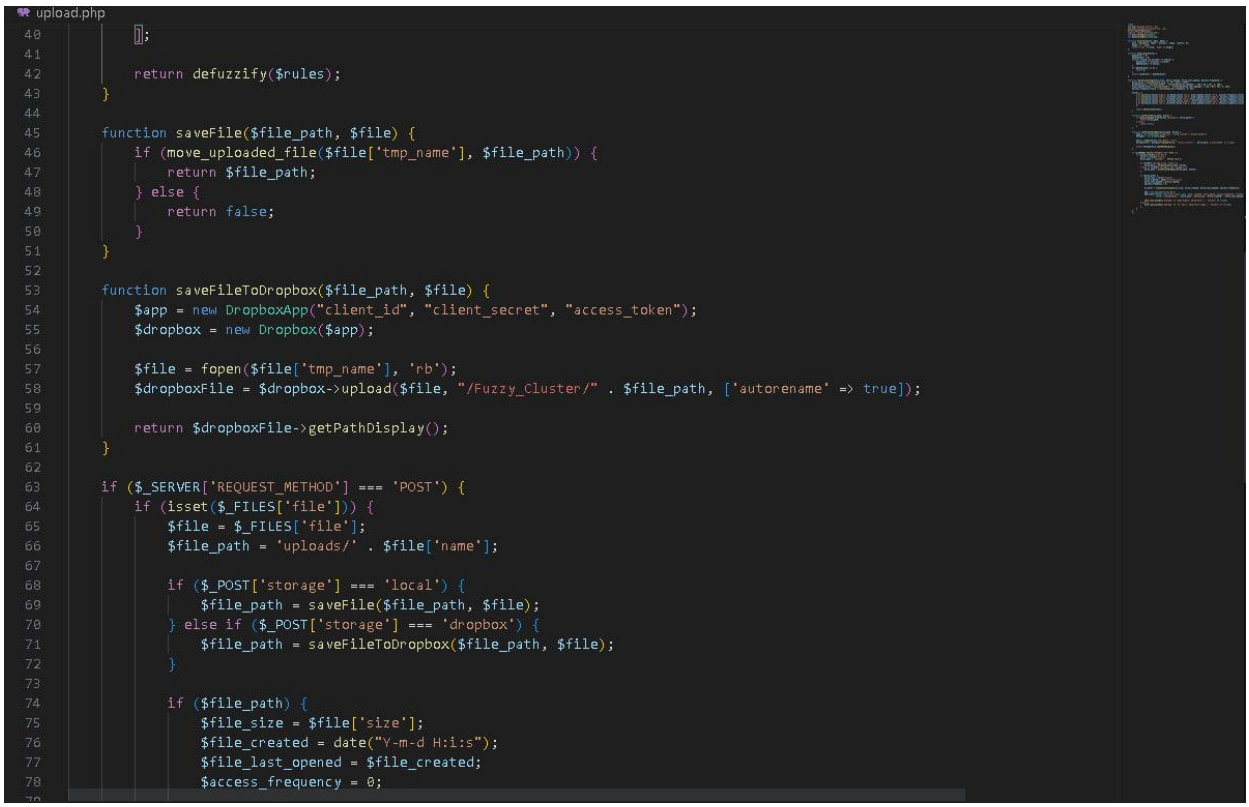
```
public function files(){
   $me=$this->checkMe();
   if($_SESSION["sort"] !='') $sort = $_SESSION["sort"];
   else $sort = 1;
```

```
   if($sort == 1)
      $query = "SELECT * FROM `".$me['id']."_files` ORDER BY
count DESC";
   elseif($sort == 2)
      $query = "SELECT * FROM `".$me['id']."_files` ORDER BY id
DESC";
   elseif($sort == 3)
      $query = "SELECT * FROM `".$me['id']."_files` ORDER BY id
ASC";
   return $this->query($query);
}
```

Отже тепер є готова форма завантаження файлів (рис 2.1 ; рис 2.2 ; рис 2.3).

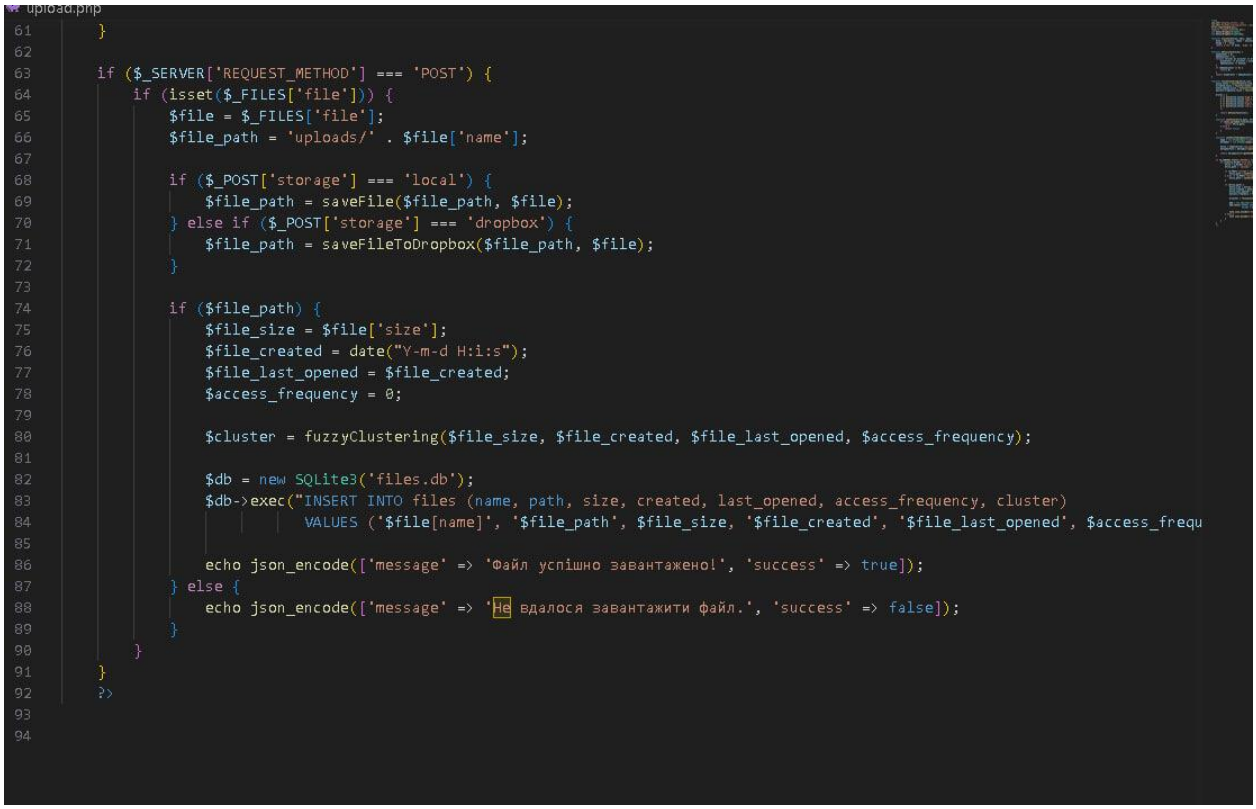
```php
<?php
ini_set('display_errors', 1);
ini_set('display_startup_errors', 1);
error_reporting(E_ALL);
require 'vendor/autoload.php';
use Kunnu\Dropbox\Dropbox;
use Kunnu\Dropbox\DropboxApp;

function fuzzify($value, $min, $max) {
    $low = max(min(1, ($max - $value) / ($max - $min)), 0);
    $high = 1 - $low;
    return ['low' => $low, 'high' => $high];
}

function defuzzify($rules) {
    $numerator = 0;
    $denominator = 0;
    foreach ($rules as $cluster => $value) {
        $numerator += $cluster * $value;
        $denominator += $value;
    }
    if ($denominator == 0) {
        return 0;
    }
    return $numerator / $denominator;
}

function fuzzyClustering($file_size, $file_created, $file_last_opened, $access_frequency) {
    $size_fuzzy = fuzzify($file_size, 0, 10 * 1024 * 1024);
    $created_fuzzy = fuzzify((time() - strtotime($file_created)) / (60 * 60 * 24), 0, 365);
    $last_opened_fuzzy = fuzzify((time() - strtotime($file_last_opened)) / (60 * 60 * 24), 0, 365);
    $access_frequency_fuzzy = fuzzify($access_frequency, 0, 10);

    $rules = [
        1 => min($size_fuzzy['high'], $created_fuzzy['low'], $last_opened_fuzzy['low'], $access_frequency_fuzzy['high']),
        2 => min($size_fuzzy['high'], $created_fuzzy['low'], $last_opened_fuzzy['high'], $access_frequency_fuzzy['high']),
        3 => min($size_fuzzy['low'], $created_fuzzy['low'], $last_opened_fuzzy['high'], $access_frequency_fuzzy['high']),
        4 => min($size_fuzzy['high'], $created_fuzzy['high'], $last_opened_fuzzy['low'], $access_frequency_fuzzy['low']),
        5 => min($size_fuzzy['low'], $created_fuzzy['high'], $last_opened_fuzzy['low'], $access_frequency_fuzzy['low'])
```

Рис.2.1 Загальний вигляд системної частини коду яка відповідає за кластеризацію та заватаження фалів до сховища



```php
        ];

        return defuzzify($rules);
    }

    function saveFile($file_path, $file) {
        if (move_uploaded_file($file['tmp_name'], $file_path)) {
            return $file_path;
        } else {
            return false;
        }
    }

    function saveFileToDropbox($file_path, $file) {
        $app = new DropboxApp("client_id", "client_secret", "access_token");
        $dropbox = new Dropbox($app);

        $file = fopen($file['tmp_name'], 'rb');
        $dropboxFile = $dropbox->upload($file, "/Fuzzy_Cluster/" . $file_path, ['autorename' => true]);

        return $dropboxFile->getPathDisplay();
    }

    if ($_SERVER['REQUEST_METHOD'] === 'POST') {
        if (isset($_FILES['file'])) {
            $file = $_FILES['file'];
            $file_path = 'uploads/' . $file['name'];

            if ($_POST['storage'] === 'local') {
                $file_path = saveFile($file_path, $file);
            } else if ($_POST['storage'] === 'dropbox') {
                $file_path = saveFileToDropbox($file_path, $file);
            }

            if ($file_path) {
                $file_size = $file['size'];
                $file_created = date("Y-m-d H:i:s");
                $file_last_opened = $file_created;
                $access_frequency = 0;
```

Рис.2.2 Продовження першої частини коду



```php
    }

    if ($_SERVER['REQUEST_METHOD'] === 'POST') {
        if (isset($_FILES['file'])) {
            $file = $_FILES['file'];
            $file_path = 'uploads/' . $file['name'];

            if ($_POST['storage'] === 'local') {
                $file_path = saveFile($file_path, $file);
            } else if ($_POST['storage'] === 'dropbox') {
                $file_path = saveFileToDropbox($file_path, $file);
            }

            if ($file_path) {
                $file_size = $file['size'];
                $file_created = date("Y-m-d H:i:s");
                $file_last_opened = $file_created;
                $access_frequency = 0;

                $cluster = fuzzyClustering($file_size, $file_created, $file_last_opened, $access_frequency);

                $db = new SQLite3('files.db');
                $db->exec("INSERT INTO files (name, path, size, created, last_opened, access_frequency, cluster)
                           VALUES ('$file[name]', '$file_path', $file_size, '$file_created', '$file_last_opened', $access_frequ

                echo json_encode(['message' => 'Файл успішно завантажено!', 'success' => true]);
            } else {
                echo json_encode(['message' => 'Не вдалося завантажити файл.', 'success' => false]);
            }
        }
    }
    ?>
```

Рис.2.3 Продовження другої частини коду

Використана література [ ] щодо реалізації коду нечіткої кластеризації та завантаження на сервер або в середовище API dropbox/

Список вхідних файлів формується на основі таких даних:

1. $file_size: Розмір файлу. Це числове значення, яке представляє розмір завантаженого файлу у байтах.

2. $file_created: Дата створення файлу. Це рядок, який містить дату та час створення файлу у форматі "Y-m-d H:i:s" (рік-місяць-день година:хвилина:секунда).

3. $file_last_opened: Дата останнього відкриття файлу. Це також рядок, який містить дату та час останнього відкриття файлу у тому ж форматі.

4. $access_frequency: Частота доступу до файлу. Це числове значення, яке вказує, як часто файл був відкритий або використовувався.

Ці дані використовуються в функції fuzzyClustering для виконання розмиття та визначення кластера, який відповідає файлу. Функція fuzzyClustering виконує розмиття (fuzzification) для кожного з параметрів файлу та застосовує правила, які визначають значення кластера файлу на основі цих розмитих значень.

2.3. Вибір середовища програмування

Microsoft Visual Studio Code (VS Code) — це безкоштовний редактор коду, розроблений компанією Microsoft. Він забезпечує широкі можливості для редагування, налагодження та розширення різних мов програмування. VS Code володіє простим та інтуїтивно зрозумілим інтерфейсом, що робить його легким у використанні навіть для новачків.

Основні особливості Microsoft Visual Studio Code включають:

1. Підсвічування синтаксису для багатьох мов програмування, включаючи HTML, CSS, JavaScript, PHP та багато інших.

2. Автодоповнення коду, що спрощує процес написання коду швидше і без помилок.

3. Інтегрована система керування версіями (наприклад, Git), що дозволяє зручно відстежувати зміни в проекті та співпрацювати з іншими розробниками.

4. Вбудована консоль для виконання команд та перегляду результатів.

5. Можливість налагодження коду, що дозволяє знайти та виправити помилки в процесі розробки.

6. Розширюваність через велику кількість додатків та розширень, які додають нові функції та можливості до редактора.

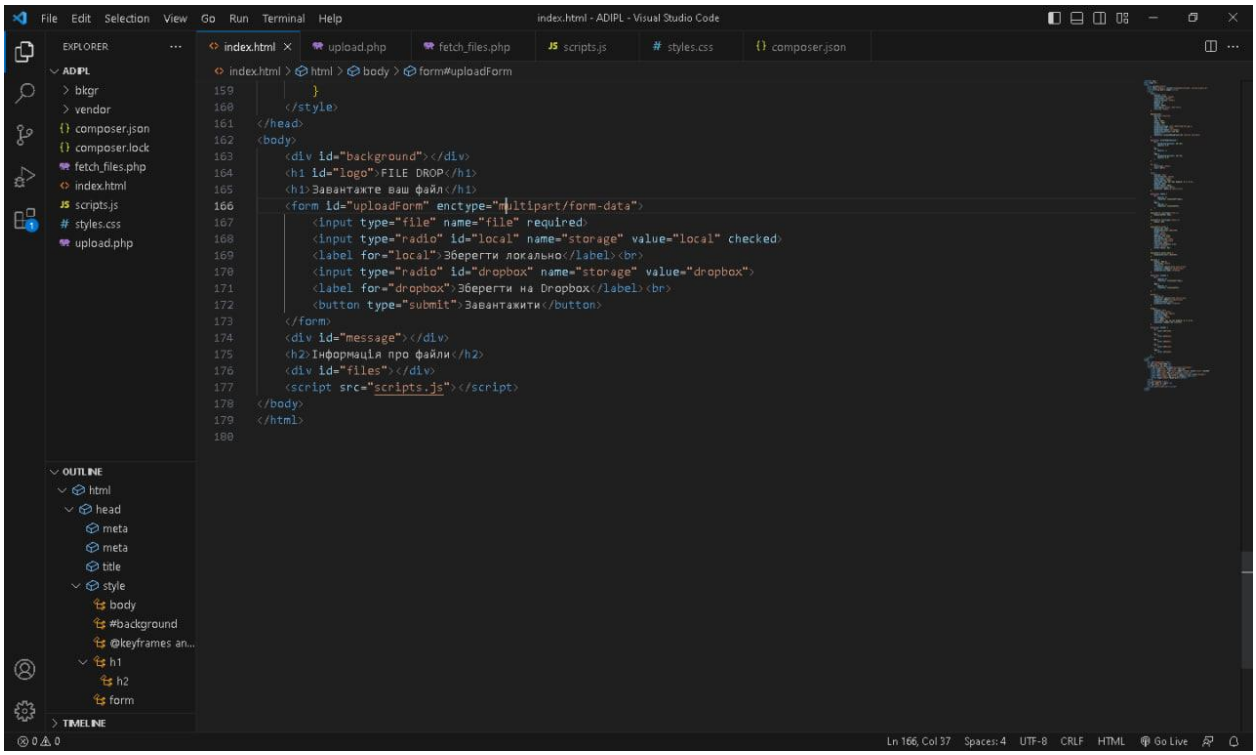


Рис.2.4 Інтерфейс середовища розробки Visual Code

Давайте розглянемо мову програмування під назвою JavaScript. JavaScript (або скорочено JS) є мовою, що демонструє динамічність та орієнтованість на об'єкти. Вона випливає зі стандарту ECMAScript і активно застосовується в браузері, забезпечуючи здатність коду взаємодіяти з користувачем, управляти браузером, обмінюватися даними з сервером асинхронно, змінювати веб-сторінку структурно та візуально.

JavaScript також використовується в програмуванні на стороні сервера, аналогічно до таких мов як Java та C#. Її використання охоплює галузі розробки ігор, створення додатків для робочого столу та мобільних пристроїв, написання сценаріїв для програмного забезпечення, а також використання всередині PDF-документів та ін.

JavaScript розглядається як прототипна (це є підгрупою об'єктно-орієнтованих) та скриптова мова програмування з динамічною типізацією. Вона не лише включає в себе прототипну парадигму, але й частково підтримує інші парадигми програмування, такі як імперативну та функціональну. Серед її особливостей варто відзначити динамічну та слабку типізацію, автоматизоване управління пам'яттю, прототипне наслідування та функції як об'єкти першого класу.

JavaScript демонструє ряд особливостей, що притаманні об'єктно-орієнтованим мовам, але, виходячи з концепції прототипів, вона обробляє об'єкти не так, як це виконується в традиційних ООП мовах. Крім того, в JavaScript властиві елементи функціональних мов - функції як об'єкти першого класу, об'єкти як списки, каррінг, анонімні функції, замикання, що надає мові додаткової варіативності. Отже, JavaScript вважається мовою з великою гнучкістю та мультіпарадигмальним характером, що дає розробникам можливість використовувати різні стратегії для вирішення широкого спектра задач.

JavaScript пропонує синтаксис, що нагадує C, але порівняно з ним, має кілька унікальних особливостей:

1. Використання об'єктів з можливістю інтроспекції і динамічної зміни типу через прототипи.
2. Представлення функцій як об'єктів першого класу.
3. Обробка виключень.
4. Автоматичне приведення типів.
5. Автоматичне звільнення пам'яті.
6. Використання анонімних функцій.

JavaScript містить набір вбудованих об'єктів: Global, Object, Error, Function, Array, String, Boolean, Number, Math, Date, RegExp. Вона також пропонує ряд вбудованих операцій, що не обов'язково функцій або методів, а також вбудованих операторів для управління логікою виконання програми. Синтаксис JavaScript схожий на Java (та наслідує його від C), але був спрощений для полегшення навчання. Так, змінна визначається без вказівки типу, властивості не мають визначених типів, а декларація функції може йти після її використання в коді.

HTML5 представляє собою наступну еволюцію мови HTML. Її специфікації не обмежуються лише маркерами, але включають у себе цілий набір веб-технологій. Разом вони формують відкриту Веб-платформу, яка створює програмне середовище для крос-платформових додатків. Ці додатки мають здатність взаємодіяти з апаратними ресурсами, а також мають інструменти для роботи з відео, графікою, анімацією і можуть надавати розширені мережеві функції.

З нової версії HTML пропонується вилучити приблизно 15 тегів. При розробці нових тегів було проаналізовано велику кількість популярних сайтів, щоб визначити основні елементи, які є загальними для всіх веб-сторінок.

Завдяки розмітці областей на сторінці за допомогою спеціальних елементів, HTML5 може допомогти користувачам навігуватися легше. Наприклад, вони можуть легко пропустити розділ навігації або швидко перейти від одної статті до іншої без потреби в створенні відповідних посилань авторами. Автори, у свою чергу, отримують перевагу заміни численних div-ів одним з декількох відповідних елементів, що призводить до чистішого і зрозумілішого початкового коду.

Відео та аудіо елементи легко інтегруються в HTML5. Більшість API є загальними для цих елементів, із деякими відмінностями в візуальних та не-візуальних медіа. Усі сучасні браузери (з винятком Internet Explorer) вже

впровадили підтримку цих елементів. Найлегший спосіб вбудувати відео - це використати тег video і дозволити браузеру відобразити інтерфейс за замовчуванням.

Переглянемо CSS мову. Каскадні таблиці стилів, відомі англійською як Cascading Style Sheets, є специфічною мовою, розробленою для деталізації веб-сторінок, які створюються за допомогою мов розмітки даних.

CSS використовується творцями та користувачами веб-сторінок з метою визначення кольорової гами, шрифтів, компонування та інших візуальних характеристик сторінки.

Однією з ключових переваг CSS є можливість відокремлення вмісту сторінки (або контенту, створеного за допомогою HTML, XML або аналогічних мов розмітки) від візуального представлення документу. Це відокремлення поліпшує сприйняття та доступність вмісту, надає більше гнучкості та контролю над представленням контенту в різних умовах, дозволяє структурувати вміст більш ефективно та уникнути повторення. Завдяки CSS можна адаптувати вміст до різних умов відображення.

MySQL відноситься до числа найбільш відомих та широко використовуваних систем управління базами даних (СУБД) в мережі інтернет. Хоча вона не розроблена для роботи з величезними обсягами даних, MySQL ідеально підходить для використання на веб-сайтах, починаючи від маленьких і закінчуючи досить об'ємними.

База даних включає в себе такі головні таблиці як: users,де зберігаються усі користувачі;в таблиці files записується всі дії, що виконувались над файлами;folders та load, де зберігається посилання з сервера на файл(Рис.2.5).

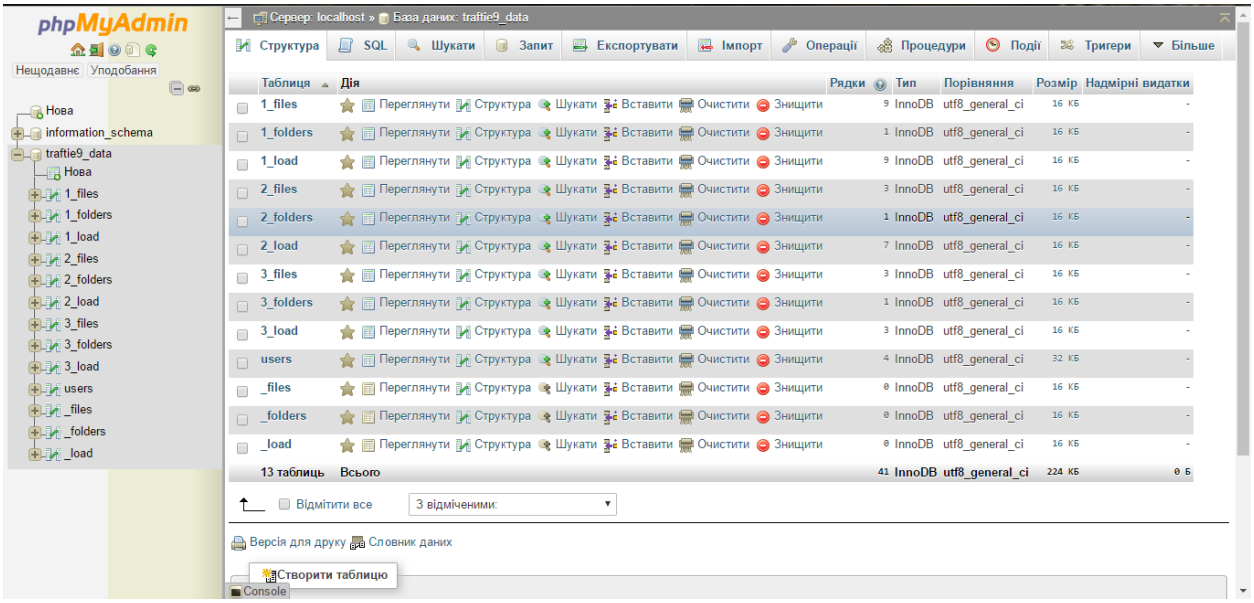


Рис.2.5 Таблиці бази даних [9]

2.3. Розробка інтерфейсу проекту

Програмне забезпечення має бути простим і зрозумілим, показуючи користувачеві, що розробники працювали над зручністю використання, а не лише над привабливим зовнішнім виглядом продукту. Зручність та результативність є найважливішими факторами для користувачів, адже вони бачать лише інтерфейс, який стає кінцевим продуктом для них.

Розробники інтерфейсів повинні зосередитися на врахуванні потреб користувачів, спочатку враховуючи загальні вимоги до інтерфейсу, а потім вивчаючи різні потреби, залежно від конкретної задачі. Важливо мати розуміння своїх користувачів, що може бути досягнуто через консультації з експертами у різних галузях, оскільки проектування інтерфейсів часто потребує експертних знань з психології, ергономіки та соціології. Інтерфейси користувача мають враховувати психологічні та поведінкові закономірності людини. Розробники також повинні знайти оптимальний зовнішній вигляд, наповнення та функції інтерфейсу, що задовольнятимуть широкий спектр вимог і завдань користувача.

Використання сучасних технологій розроблення програмного забезпечення вимагає відповідного розвитку методів та засобів проектування і розроблення інтерфейсів користувача, оскільки проблема неадекватного сприйняття візуальної інформації користувачем може призвести до катастроф з людськими жертвами, екологічними катаклізмами або значних економічних втрат.

Інтерфейс - це не лише вікна, кнопки, піктограми, меню та курсор миші. Необхідність проектування інтерфейсу вже на ранніх стадіях розробки програмного забезпечення іноді недооцінюється розробниками програмних продуктів. Але незважаючи на складність задачі, виконуваної програмним засобом, складові цієї задачі повинні залишатись простими.

Інтерфейс користувача (ІК) - це сукупність засобів, за допомогою яких користувач взаємодіє з різними пристроями (комп'ютером або побутовою технікою) або іншим складним інструментарієм (системою). Інтерфейс користувача - це різновид інтерфейсів, де з одного боку є людина, а з іншого - машина (пристрій, програмне забезпечення).

Головна сторінка повинна мати інтуїтивно зрозумілий інтерфейс, що складається з загальноприйнятих елементів (рис. 2.6):

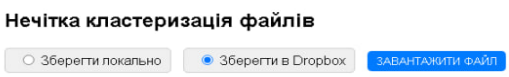


Рис. 2.6. Загальний інтерфейс програми [9]

1. Відкриваємо текстовий редактор і створюємо новий файл HTML.

Встановлюємо кодування UTF-8 за допомогою рядка <meta charset="UTF- 8">. Це дозволяє коректно відображати українські символи.

Додаємо метатег <meta name="viewport" content="width=device-width, initial-scale=1.0">, яка встановлює правильний масштаб та розмір екрану для пристроїв з різними розмірами екрану (рис.2.7)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>

</body>
</html>
```

Рис.2.7. Створення шаблону HTML сторінки [9]

2. Встановлюємо заголовок сторінки <title>Нечітка Кластеризація</title ,та додаємо стилі CSS між тегами <style>...</style> у блок <head>…</head>. Ці стилі відповідають за вигляд сторінки. Вони включають розміщення елементів на екрані, властивості фону, анімації та інші дизайнерські параметри:

```
<style>
        body {
            display: flex;
            flex-direction: column;
            align-items: center;
            justify-content: center;
            padding: 0;
            margin: 0;
            height: 100vh;
            font-family: Arial, sans-serif;
            overflow: hidden;
        }

        #background {
            position: absolute;
```

```
            top: 0;
            left: 0;
            width: 100%;
            height: 100%;
            z-index: -1;
            background-image: url('/ADIPL/bkgr/01.jpg');
            background-size: cover;
            background-repeat: no-repeat;
            background-position: 50% 50%;
            opacity: 1;
            animation: animatedBackground 20s infinite alternate;
        }

        @keyframes animatedBackground {
            0% {
                background-position: 40% 40%;
                opacity: 0.5;
            }
            50% {
                opacity: 1;
            }
            100% {
                background-position: 70% 70%;
                opacity: 0.5;
            }
        }

        h1, h2 {
            text-align: center;
            color: #333;
        }

        form {
            display: flex;
            flex-direction: column;
            align-items: center;
            margin-bottom: 20px;
            box-shadow: 0px 4px 15px rgba(0, 0, 0, 0.1);
            padding: 20px;
            border-radius: 10px;
            background-color: #fff;
            animation: fadeIn 1s ease-in-out;
        }

        @keyframes fadeIn {
            0% {
                opacity: 0;
                transform: translateY(-20px);
            }
            100% {
```

```
            opacity: 1;
            transform: translateY(0);
        }
    }

    #uploadForm input[type='file'] {
        margin-bottom: 10px;
    }

    #uploadForm input[type='radio'] {
        margin: 5px;
    }

    #uploadForm button {
        margin-top: 10px;
        background-color: #4caf50;
        border: none;
        color: white;
        padding: 15px 32px;
        text-align: center;
        text-decoration: none;
        display: inline-block;
        font-size: 16px;
        transition-duration: 0.4s;
        cursor: pointer;
        border-radius: 5px;
    }

    #uploadForm button:hover {
        background-color: #45a049;
    }

    #message {
        margin: 20px 0;
        text-align: center;
        opacity: 0;
        transition: opacity 0.5s ease-in-out;
        animation: slideIn 1s ease-in-out;
        animation-fill-mode: forwards;
    }

    @keyframes slideIn {
        0% {
            opacity: 0;
            transform: translateY(-20px);
        }
        100% {
            opacity: 1;
            transform: translateY(0);
        }
```

```
        }

        #files {
            opacity: 0;
            transition: opacity 0.5s ease-in-out;
            animation: fadeIn 1s ease-in-out;
            animation-delay: 0.5s;
            animation-fill-mode: forwards;
        }

        #logo {
            display: flex;
            align-items: center;
            justify-content: center;
            font-size: 48px;
            font-weight: bold;
            color: #000;
            text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.5);
            animation: shimmer 10s infinite;
        }

        @keyframes shimmer {
            0% {
                color: #7fe7d9;
            }
            25% {
                color: #78e7e2;
            }
            50% {
                color: #6bb9cc;
            }
            75% {
                color: #63c2e7;
            }
            100% {
                color: #75d2e9;
            }
        }
    </style>
```

3. Починаємо основне тіло сторінки за допомогою тегів <body>...</body>. Всі наступні елементи будуть розміщені всередині цих тегів, та створюємо елемент , який відповідає за фонове зображення сторінки. Встановлюємо його розміри та параметри відображення за допомогою стилів CSS.

```
<body>
    <div id="background"></div>
    <h1 id="logo">FILE DROP</h1>
</body>
```

4. Наступним кроком додаємо заголовок першого рівня <h1 id="logo">FILE DROP</h1>, який відображає назву сторінки. У даному випадку назва "FILE DROP" може бути замінена на бажану назву.Додаємо заголовок першого рівня <h1>Завантажте ваш файл</h1>, який відображає інструкцію для користувачів.

```
<body>
    <div id="background"></div>
    <h1 id="logo">FILE DROP</h1>
    <h1>Завантажте ваш файл</h1>
</body>
```

5. Створюємо форму за допомогою тегу <form id="uploadForm" enctype="multipart/form-data">. Ця форма дозволяє користувачеві вибрати та завантажити файл. Далі становлюємо тип кодування форми enctype="multipart/form-data", щоб дозволити передачу файлів. Вставляємо елемент <input type="file" name="file" required>, який дозволяє користувачеві вибрати файл для завантаження. Потім встановлюємо атрибут required, щоб забезпечити обов'язкове заповнення поля, тепер додаємо два радіокнопки <input type="radio" id="local" name="storage" value="local" checked> та <input type="radio" id="dropbox" name="storage" value="dropbox">, які дозволяють користувачеві обрати місце збереження файлу, також на всякий випадок становлюємо атрибут checked для першої радіокнопки, щоб вона була вибрана за замовчуванням. Тепер додаємо кнопку <button type="submit">Завантажити</button>, яка надсилає форму для обробки завантаженого файлу.

```
<body>
    <div id="background"></div>
    <h1 id="logo">FILE DROP</h1>
    <h1>Завантажте ваш файл</h1>
    <form id="uploadForm" enctype="multipart/form-data">
        <input type="file" name="file" required>
```

```
        <input type="radio" id="local" name="storage" value="local" checked>
        <label for="local">Зберегти локально</label><br>
        <input type="radio" id="dropbox" name="storage" value="dropbox">
        <label for="dropbox">Зберегти на Dropbox</label><br>
        <button type="submit">Завантажити</button>
    </form>
```

6. Наступним кроком додаємо елемент , який буде відображати повідомлення користувачеві після завантаження файлу, і також додаємо заголовок другого рівня <h2>Інформація про файли</h2>, який відображає заголовок для розділу з інформацією про файли, сюди ж додаємо елемент , який буде містити інформацію про завантажені файли.

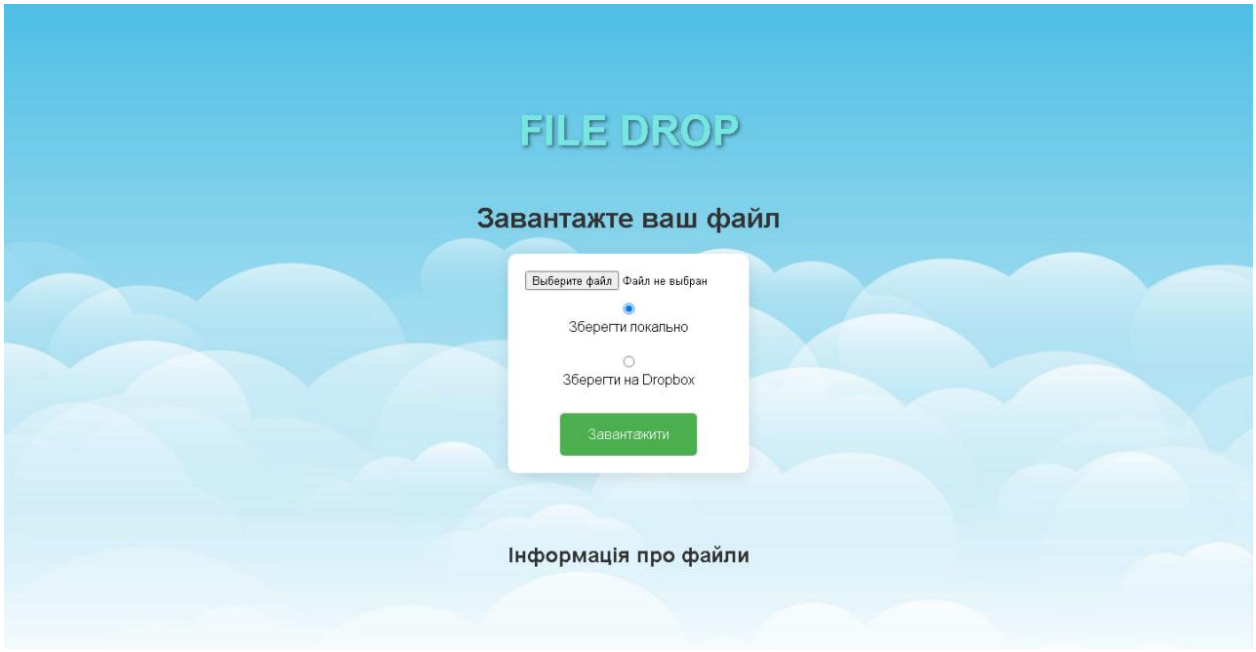
```
    <div id="background"></div>
    <h1 id="logo">FILE DROP</h1>
    <h1>Завантажте ваш файл</h1>
    <form id="uploadForm" enctype="multipart/form-data">
        <input type="file" name="file" required>
        <input type="radio" id="local" name="storage" value="local" checked>
        <label for="local">Зберегти локально</label><br>
        <input type="radio" id="dropbox" name="storage" value="dropbox">
        <label for="dropbox">Зберегти на Dropbox</label><br>
        <button type="submit">Завантажити</button>
    </form>
    <div id="message"></div>
    <h2>Інформація про файли</h2>
    <div id="files"></div>
```

7. Підключаємо зовнішній файл JavaScript за допомогою тегу <script src="scripts.js"></script>. Цей файл може містити скрипти для обробки подій

```
    <script src="scripts.js"></script>
</body>
</html>
```

Таким чином, обрано характеристики для кластерного аналізу такі як: частота завантаження, час завантаження. Визначили структуру бази даних та середовище програмування. Було обрано мову програмування PHPяк основну мову реалізації даної задачі.

Розробили анімований інтерфейс так , щоб користувачеві було зручно і приємно користуватися функціями web додатку .



РОЗДІЛ 3. РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ПРОГРАМИ

3.1. Реалізація кластеризації

Головне завдання кластерного аналізу є оптимізація даних, успішне розміщення на серверній частині або в інтерфейсі.

Оптимізація виконується за різними методами, такими як:

1. Популярніть (частота запитів);
2. Дата та час завантаження.

Згідно другого пункту, розподіл на нові або старі версії документів. На стороні серверу проходять такі запити в базу даних:

```
      public function files(){
        $me=$this->checkMe();
        if($_SESSION["sort"] !='') $sort = $_SESSION["sort"];
        else $sort = 1;
        if($sort == 1)
           $query = "SELECT * FROM `".$me['id']."_files` ORDER BY count DESC";
        elseif($sort == 2)
           $query = "SELECT * FROM `".$me['id']."_files` ORDER BY id DESC";
        elseif($sort == 3)
           $query = "SELECT * FROM `".$me['id']."_files` ORDER BY id ASC";
        return $this->query($query);
     }


     public function files_f($id){
        $me=$this->checkMe();
        if($_SESSION["sort"] !='') $sort = $_SESSION["sort"];
        else $sort = 1;
        if($sort == 1)
           $query = "SELECT * FROM `".$me['id']."_files` WHERE `folder`='".$id."'
ORDER BY count DESC";
        elseif($sort == 2)
           $query = "SELECT * FROM `".$me['id']."_files` WHERE `folder`='".$id."'
ORDER BY id DESC";
        elseif($sort == 3)
           $query = "SELECT * FROM `".$me['id']."_files` WHERE `folder`='".$id."'
ORDER BY id ASC";
        return $this->query($query);
     }
```

Перша функція files() розподіляє не згруповані дані. Функція files_f() працює вже з згрупованими даними

Також можна розподіляти каталоги:

```
public function folders(){
   $me=$this->checkMe();
      $query = "SELECT * FROM `".$me['id']."_folders` ORDER BY id DESC";
   return $this->query($query);
}
```

3.2. Реалізація хмарного сховища

Робота будь-якого хмарного сховища традиційно розпочинається з пункту авторизації або реєстрації користувача, але у нашому випадку цей пункт не враховується . Почнемо з головної сторінки , це результат нашої роботи зі стилями так HTML :

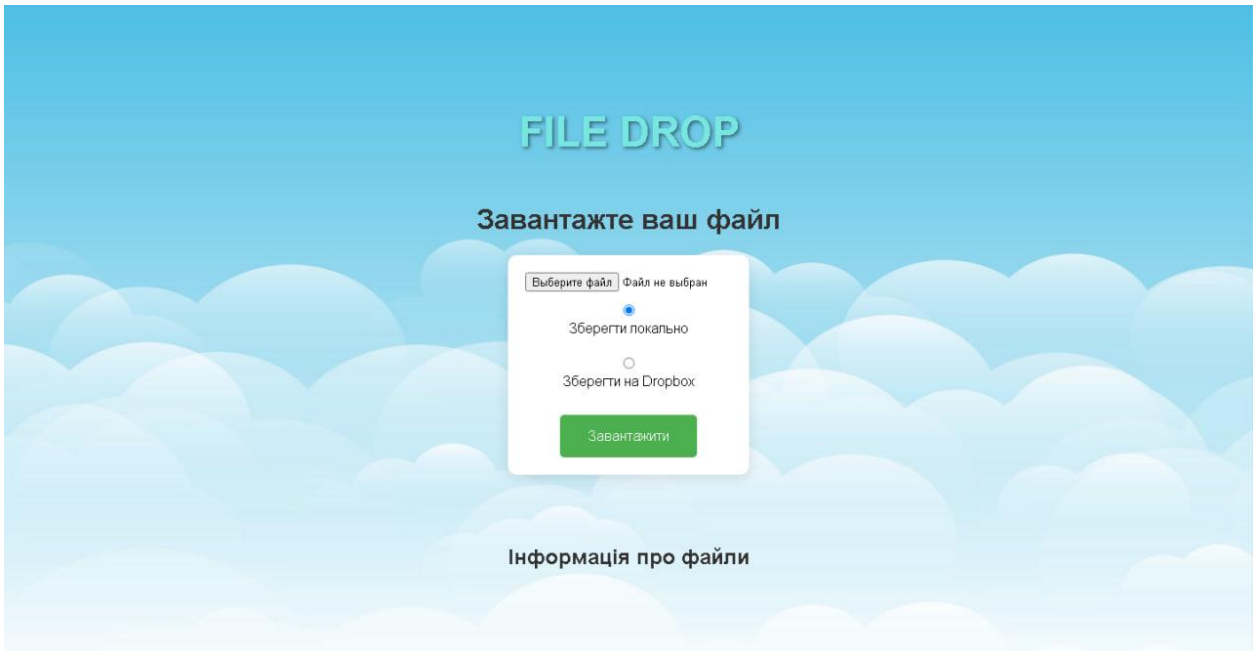


Рис.3.2 Головна сторінка [ ]

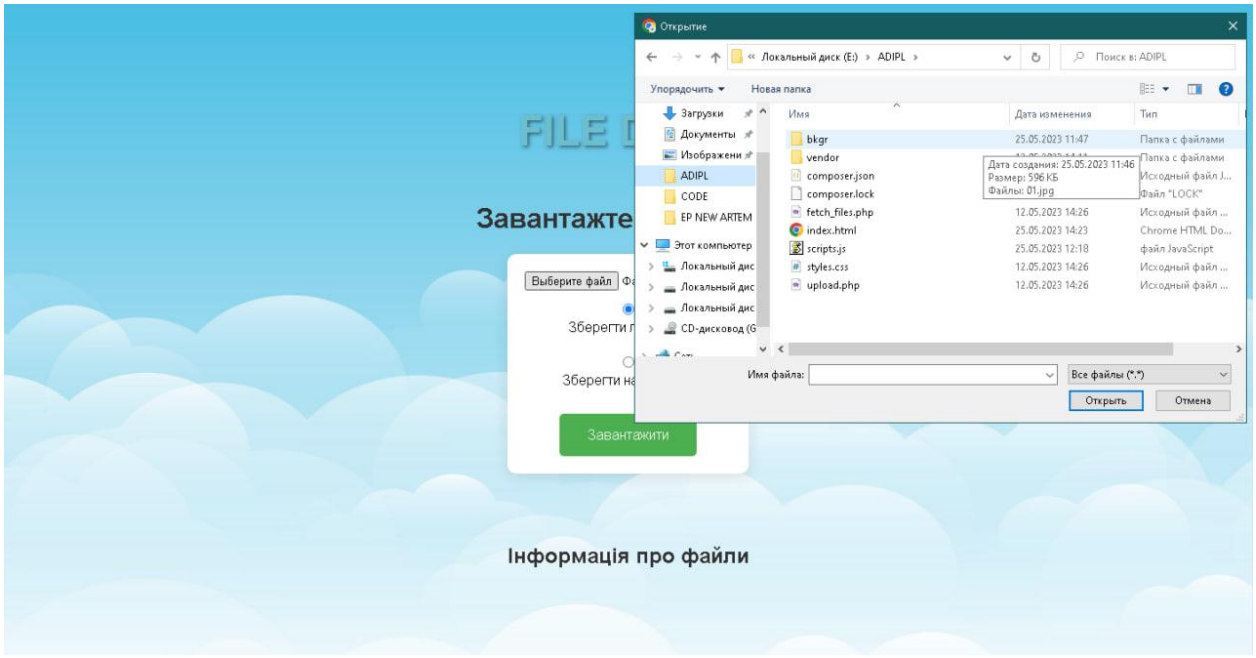


Рис.3.3 Діалогове вікно вибору файлів [ ]

Відповідно код реалізації двох методів з покажчиком завантаження:

```
function loads(){
      var load = $('#file-load');
      var dropzone2 = document.getElementById("dropZone");
      load.text("Загрузка");
      load.css("display","block");
      load.css("opacity","1");
      var files = document.getElementById("ll").files;
      var formData = new FormData(),
      xhr = new XMLHttpRequest(),
      x;
      xhr.addEventListener('progress', uploadProgress2, false);
      for(x = 0; x < files.length; x++) {
         formData.append('file[]', files[x]);
      }
      xhr.onload = function() {
            if (event.target.status == 200) {
               load.text('Готово');
            } else {
               load.text('Ошибка!');
            }
            setTimeout(function() {
               load.css("opacity","0");
               setTimeout(function() {
                  load.css("display","none");
               }, 800);
```

```
            }, 3000);
            if(dropzone2.classList.contains('files')){
               show(0);
            }
            else show(4);

      };
      xhr.open('post', '/load');
      xhr.send(formData);
   }
   function uploadProgress2(event) {
      var percent = parseInt(event.loaded / event.total * 100);
      load.text('Загрузка: ' + percent + '%');
   }
```

Відповідно параметр DropZone визначає поле де самє діє функція DragandDrop. Також наявний параметр завантаження файлів.

```
   $(document).ready(function() {
   var dropzone2 = document.getElementById("dropZone");
   var dropZone = $('#dropZone');
   var load = $('#file-load');
   dropzone2.ondrop = function(e) {
      e.preventDefault();
      dropZone.removeClass('hover');
      load.text("Загрузка");
      load.css("display","block");
      load.css("opacity","1");
      upload(e.dataTransfer.files);
    };
   var upload = function(files) {
      var formData = new FormData(),
      xhr = new XMLHttpRequest(),
      x;
      xhr.upload.addEventListener('progress', uploadProgress,
false);
      for(x = 0; x < files.length; x++) {
         formData.append('file[]', files[x]);
      }
      xhr.onload = function() {
         if (event.target.status == 200) {
            load.text('Готово');
         } else {
            load.text('Ошибка!');
         }
         setTimeout(function() {
            load.css("opacity","0");
            setTimeout(function() {
               load.css("display","none");
            }, 800);
         }, 3000);
```

```
            if(dropzone2.classList.contains('files')){
               show(0);
            }
            else show(4);

         };
         xhr.open('post', '/load');
         xhr.send(formData);
      };
      dropzone2.ondragover = function() {
         dropZone.addClass('hover');
         return false;
      };

      dropzone2.ondragleave = function() {
         dropZone.removeClass('hover');
         return false;
      };
      function uploadProgress(event) {
         var percent = parseInt(event.loaded / event.total * 100);
         load.text('Загрузка: ' + percent + '%');
      }
});
```

Відповідно у базу даних записується посилання на файл, який зберігається на сервері у вигляді файлу розширення file(рис.3.6).

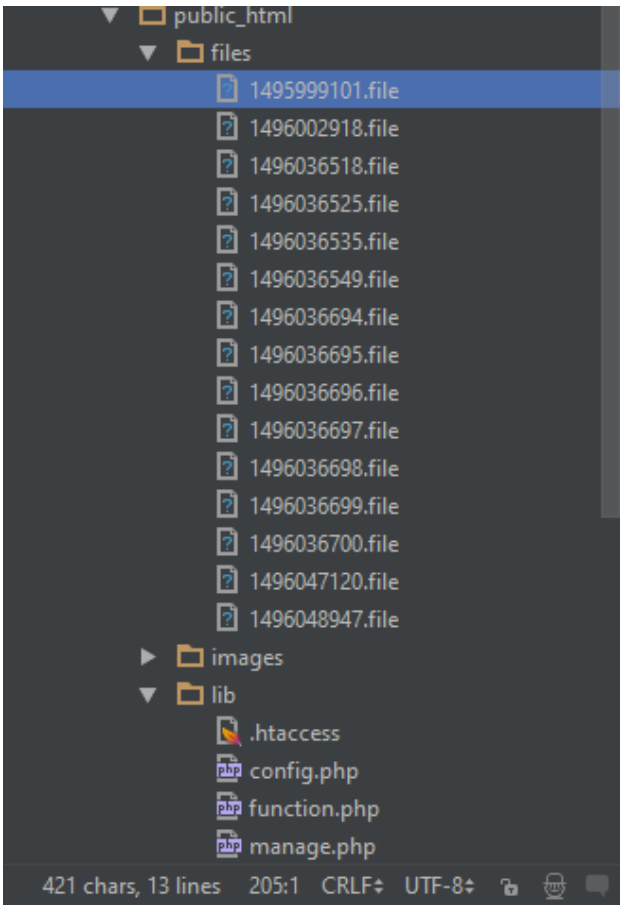


Рис.3.6 Файли на сервері [ ]

Також слід зазначити окремий розділ «Статистика», де відзначається період та трафік завантаження; об'єм сховища та нові файли(рис.3.8).

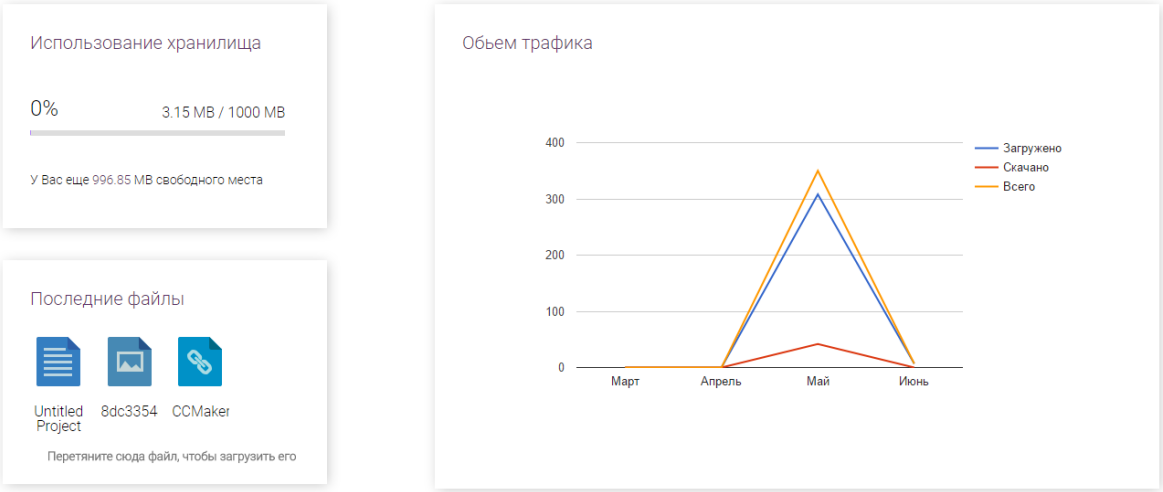


Рис.3.8. Сторінка статистики [ ]

Розглянемо реалізацію за допомогою GoogleChartsAPI.

```
     google.charts.load('current', {'packages':['corechart']});

    google.charts.setOnLoadCallback(drawChart);

    function drawChart() { var data = new google.visualization.DataTable();
    data.addColumn('string', 'Месяц');
    data.addColumn('number', 'Загружено');
    data.addColumn('number', 'Скачано');
    data.addColumn('number', 'Всего');
data.addRows([
<?
      $month=array("Январь","Февраль","Март","Апрель","Май","Июнь","Июль","Август
","Сентябрь","Октябрь","Ноябрь","Декабрь");
      $m=date("n");
      $y=date("Y");
      for($i=3; $i>0;$i--){
         if($m-$i>0){
            $count=$function->traffic($m-$i,$y);
            echo("['".$month[$m - $i -1]."', ".$count[0].", ".$count[1].",
".($count[0]+$count[1])."],\n\r");
         }
         else{
            $count=$function->traffic(12+$m-$i,$y-1);
            echo("['".$month[12+$m - $i -1]."', ".$count[0].", ".$count[1].",
".($count[0]+$count[1])."],\n\r");

         }
      }
            $count=$function->traffic($m,$y);
            echo("['".$month[$m-1]."', ".$count[0].", ".$count[1].",
".($count[0]+$count[1])."],\n\r");
   ?>
      ]);
      var options = {};
      var chart = new
google.visualization.LineChart(document.getElementById('curve_chart'));

      chart.draw(data, options);
    }
```

Кількість вільного простору на сервері:

```
h1>Использование хранилища</h1>
<h2><?echo(round($me['space']/1000));?>%<span><?echo($me['space']/100);?>MB /
1000 MB</span></h2>
<div class="status-bar" id="dload"></div>
<? if($me['space']/100<950){
```

```
   echo("<h3>У Вас еще <span>".(1000 - $me['space']/100)."</span>MB свободного
места</h3>");
}
else{
   echo("<h3>У Вас осталось мало свободного места</h3>");

}?>
```

3.3. Розробка інструкції користувача

Завантажити необхідні файли можна два методами. Перший – перетягування документа у визначену область. Такий метод називається DragandDrop. Другий метод – натискання на відповідний елемент інтерфейсу.

Потім як завантажили необхідні файли, можемо оптимізувати їх за допомогою реалізованих методів в сховищі даних: по популярності, новій файли, старі файли

Користувач може групувати файли в теки. Для цього потрібно визвати контекстне меню натискаючи праву клавішу миші і у списку вибрати «в новую папку»

Також користувач може видалити файл, чи завантажити його до себе на диск. В пункті «Информация» можна отримати детальний опис цього файлу, зокрема його розмір, дату завантаження(Рис.3.13).

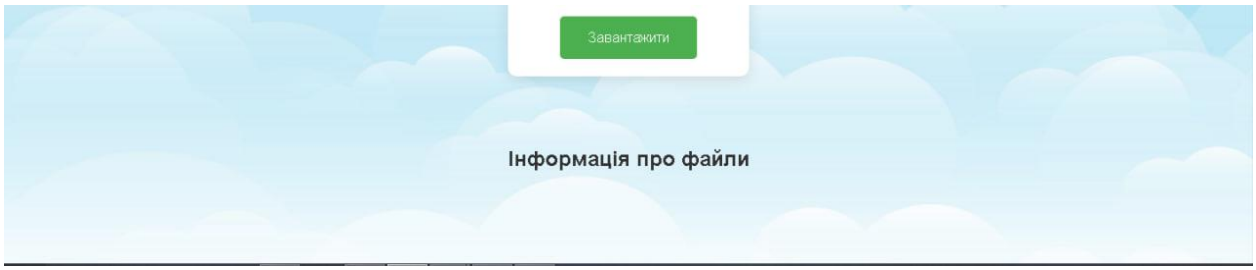
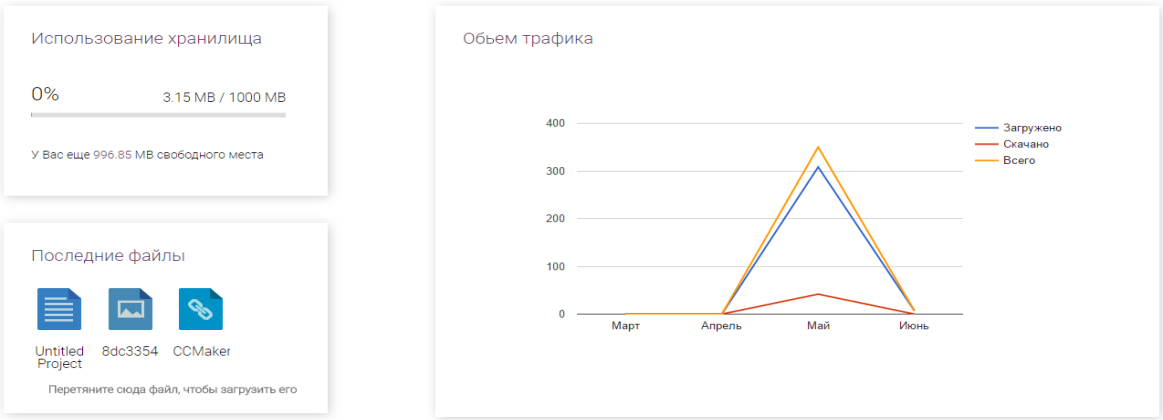


Рис.3.13 Інформація про файли [ ]

Також доступний розділ «Статистика», де користувач може переглядати трафік, останні свої завантаження та вільне місце у сховищі(рис.3.15).

Рис.3.15 Сторінка статистики



Таким чином, в роботі **б**ула розроблена кластеризація в хмарному сховищі. Визначено характеристики за якими проводиться оптимізація. Розроблене хмарне сховище даних. Розроблений основний функціонал сховища. Реалізований онлайн перегляд файлів.

На основі кількох характеристик була проведена оптимізація даних. Ініціалізація оптимізації була представлена за трьома різними методами: по популярності, нові, старі дані. Всі дані зберігається на сервері та є надійно захищені.

Дане програмне забезпечення було успішно протестоване.

ВИСНОВКИ

В результаті виконаної роботи було досягнуто поставленої мети. Розроблено хмарне сховище даних, на основі аналізу програмних аналогів. Засвоєні методи та алгоритми кластеризації, методи оптимізації.

Розроблений зручний інтерфейс користувача та весь необхідний функціонал для роботи хмарного сховища. Були використані технології зберігання файлів на сервері, завдяки чому вони є надійно захищеними.

Ініціалізовані все представлені методи оптимізації дискового простору, тому користуватись даним сховищем досить зручно.

При значній кількості неоптимізованих даних дуже не зручно шукати потрібний документ, після проведення оптимізації одразу ж видно як групуються дані та їх зручне розташування.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Классификация и кластер. Под ред. Дж. Вэн Райзина. М.: Мир, 1980. 390 с.
2. Гуд И. Дж. Ботриология ботриологии / Классификация и кластер. - М.: Мир, 1980.
3. Дюран Н., Оделл П. Кластерный анализ. - М.: Статистика, 1977.
4. Шурыгин А. М. Распределение межточечных расстояний и разностей// Программно-алгоритмическое обеспечение прикладного многомерного статистического анализа. - М., 1983.
5. Лахатин А. С. Языки программирования. Учеб. пособие / А.С. Лахатин, Л.Ю. Искакова. – Екатеринбург, 2013. – 548 с.
6. Петюшкин А. В. HTML. Экспресс-курс. / А. В. Петюшкин. – СПб.: БХВ-Петербург, 2004. – 258 с.
7. Кингсли-Хью Э. JavaScript 1.5. Учебный курс. / Э. Кингсли-Хью, К. Кингсли-Хью. – СПб.: Питер, 2001. – 272 с.
8. Стивен Хольцнер . PHP в примерах. / Стивен Хольцнер . М.: 000 «Бином-Пресс», 2007 г. Пер. с англ. 352 с
9. Томсон Лаура. Разработка Web-приложений на PHP и MySQL: Пер. с англ. /Лаура Томсон, Люк Веллинг. — 2-е изд., испр. — СПб: ООО «ДиаСофтЮП», 2003. — 672 с.
10. Флэнаган Д. JavaScript. Подробное руководство. / Д. Флэнаган. – Символ-Плюс, 2012. – 1081 с.
11. Стефанов С. JavaScript. Шаблоны. / С. Стефанов. – Символ-Плюс, 2015. – 272 с.
12. Мейер Э. CSS. Каскадные таблицы стилей. Подробное руководство. / Э. Мейер. – Символ-Плюс, 2008. – 576 с.
13. Хольцнер С. HTML5 за 10 минут / С. Хольцнер. – М.: Вильямс, 2011. – 237 с.

14. Грофф, Джеймс; Вайнберг, Пол SQL: полное руководство; Киев: BHV, 2005. - 608 с.
15. Аткинсон, Леон MySQL. Библиотека профессионала; М.: Вильямс, 2013. – 624 с.
16. Пауэлл, Томас; Шнайдер, Фриц Полный справочник по JavaScript; М.: Вильямс; Издание 2-е, 2007. - 960 с.