НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено:

Завідувач кафедри

_____ Олександр Коваль

«___»_____2020 р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Геометричне моделювання в інформаційних системах»

спеціальності 122 «Комп'ютерні науки та інформаційні технології»

на тему: «Веб-застосування для завантаження та зберігання неструктурованих даних»

Виконала:
студентка IV курсу, групи ТР-62
Тищенко Анастасія Ігорівна _____

Керівник:
к.т.н, доцент
Михайлова Ірина Юріївна _____

Рецензент:
доц., к.т.н., доц.
Побіровський Ю. М. _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студентка _____

Київ – 2020 року

Національний технічний університет України

„Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший рівень

Напрям підготовки 122 Комп’ютерні науки та інформаційні технології

Спеціалізація Геометричне моделювання в інформаційних системах

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ О.В. Коваль
(підпис)
”___”_____2020 р.

ЗАВДАННЯ

на дипломну роботу студенту

Тищенко Анастасії Ігорівни

(прізвище, ім’я, по батькові)

1. Тема роботи “Веб-застосування для завантаження та зберігання неструктурованих даних”

керівник роботи Михайлова Ірина Юріївна, к.т.н.
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”11” травня 2020р. № 1490-с

2. Строк подання студентом роботи 08 червня 2020р.

3. Вихідні дані до роботи операційна система Windows 10, середовище розробки Studio Cache та Web Storm, мова програмування Caché Object Script, фреймворк Angular, мова програмування Type Script, мова розмітки HTML, СУБД InterSystems Caché

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) огляд проблематики обліку неструктурованих даних, обґрунтування вибору засобів реалізації, опис програмної реалізації системи, опис методики роботи користувача з програмною системою.

5. Перелік ілюстраційного матеріалу (з точним зазначенням обов’язкових

креслень) Актуальність теми. Задача програмної реалізації системи. Засоби реалізації. Опис програмної реалізації системи. Методика роботи користувача з програмою системою. Висновки _____
_____

6. Дата видачі завдання ” 5 ” грудня 2020 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Затвердження теми роботи	14.12.19 – 18.12.19	
2.	Вивчення та аналіз задачі	08.02.20 – 12.02.20	
3.	Розробка архітектури та загальної структури системи	15.02.20 – 26.02.20	
4.	Розробка структур окремих підсистем	07.03.20 – 14.03.20	
5.	Підготовка матеріалів	21.03.20 – 04.04.20	
6.	Програмна реалізація системи	11.04.20 – 29.04.20	
7.	Захист програмного продукту	22.04.20 – 27.04.20	
8.	Оформлення пояснювальної записки	03.05.20 – 09.06.20	
9.	Передзахист	08.06.20 – 20.06.20	
10.	Захист	15.06.20 – 22.06.20	

Студент _____ Тищенко А.І.
(підпис) (прізвище та ініціали,)

Керівник роботи _____ Михайлова І.Ю.
(підпис) (прізвище та ініціали,)

АНОТАЦІЯ

Метою роботи є розробка програмного забезпечення завантаження та збереження неструктурованих даних використовуючи сучасні програмні технології. Система автоматизує процес зберігання та обміну документації для користувачів данної системи.

Застосунок буде корисний як для підприємств, так і для звичайних користувачів, що потребують обміну документів.

Обсяг пояснювальної записки – 72 аркуші, містить 33 ілюстрації, 3 таблиці.

Ключові слова: неструктурованні дані, база даних.

ANNOTATION

The aim of the work is to develop software for downloading and storing unstructured data using modern software technologies. The system automates the process of storing and exchanging documentation for users of this system.

The application will be useful for both businesses and ordinary users who need to share documents.

The volume of the explanatory note is 72 pages, contains 33 illustrations, 3 tables.

Keywords: unstructured data, database.

ЗМІСТ

Вступ 6
1. Задача завантаження та зберігання неструктурованих даних 8
2. Системи зберігання неструктурованих даних 10
2.1 Зберігання неструктурованих даних 10
2.2 Програмні системи зберігання неструктурованих даних 11
3. Засоби реалізації веб-застосування для завантаження та зберігання файлів 15
3.1 СУБД Caché 17
3.2 Фреймворк Angular 18
3.3 Мова програмування TypeScript 20
3.4 Мова розмітки HTML 21
3.5 Мова розмітки CSS 22
3.6 Пакетний менеджер NPM 23
3.7 Система контролю версій Git 23
4. Програмна реалізація системи завантаження та зберігання файлів 25
4.1 Архітектура та структура програмної системи зберігання неструктурованих даних 25
4.2 Функціональна декомпозиція програмної системи зберігання неструктурованих даних 26
4.3 Діаграма класів бази даних системи зберігання неструктурованих даних 27
4.4 WEB-клієнт для роботи з RESTful сервісом завантаження та зберігання файлів 28
4.5 Взаємодія клієнта та сервера 34
5. Методика роботи користувача з програмною системою завантаження та зберігання файлів 36
5.1 Системні вимоги для роботи програмної системи завантаження та зберігання файлів 36
5.2 Сценарії роботи з API 37
5.3 Сценарії роботи користувача з системою завантаження та зберігання файлів 41
Висновки 46
Список використаної літератури 47
Додаток 1 49
Додаток 2 51
Додаток 3 62

Додаток 4 68

ВСТУП

Зважаючи на посилення зв’язків між різними державами та різними державними органами набувають все більшого поширення бази даних з інформацією та документами, до яких мають мати доступ усі члени організації. Так само і на більш низькому рівні, організації та фізичні особи використовують різноманітні бази даних з документами для виконання повсякденних завдань. Наразі існує багато різних варіантів оформлення таких документів у базу даних, якою зручно користуватися та шукати необхідні документи, «з коробки», які можуть використовуватися користувачами. Однак більшість таких системи – пропрієтарні, вимагають оплачуваних налаштувань під конкретного клієнта і мають надлишок функцій для задоволення простих потреб. Таким чином актуальним є питання розробки простого у використанні веб-реєстру з базовими функціями, доступного безкоштовно для широкого загалу.

Для вирішення зазначеної проблеми було вирішено створити веб-застосування для завантаження і зберігання даних [1, 2]. Для його створення було запропоновано використати об’єктно-орієнтовану технологію постреляційної мультимодельної СУБД InterSystems Caché. Ця СУБД дозволяє зберігати неструктуровані данні, використовувати об’єктно-орієнтований підхід до проектування, створювати класи з властивостями різних типів даних та об’єкти, до яких за необхідності можна отримати реляційний та ієрархічний доступ [3]. Для написання бізнес-логіки клієнтської сторони запропоновано використовувати мову TypeScript, для відображення інтерфейсу користувача – фреймворк Angular 9.

Програмна система дозволятиме завантажувати файли будь-якого типу в БД, визначати мета-інформацію завантажених файлів, зберігати їх в зашифрованому вигляді в БД, виконувати пошук файлів, що зберігаються в БД, виходячи з мета-інформації, наприклад, імені файлу, його типу, дати створення тощо, а також вивантажувати файли на комп’ютер користувача.

Перевагою даної системи буде те, що користувач отримає доступ до єдиного універсального сховища, для роботи з яким йому не потрібно буде мати ніяких додаткових знань щодо розподілу даних в залежності від їх формату, місця знаходження тощо. Все, що від нього вимагатиметься, – це ввести адресу WEB-ресурсу та завантажити або знайти вже існуючі файли в базі даних.

Завдяки даному застосуванню користувач при роботі з даними буде володіти широким набором інструментів для завантаження, зберігання, фільтрації та сортування даних зі зручним веб-інтерфейсом.

Особливу увагу приділено можливості розширення функціоналу даної системи сторонніми розробникам, для цього були використані технології, що стандартизують підхід до написання коду.

1. ЗАДАЧА ЗАВАНТАЖЕННЯ ТА ЗБЕРІГАННЯ НЕСТРУКТУРОВАНИХ ДАНИХ

Метою дипломної роботи є створення веб-застосування для завантаження та зберігання неструктурованих даних.

Для досягнення поставленої мети необхідно вирішити наступні завдання:

— проаналізувати існуючі програмні системи для збереження неструктурованих даних;

— обрати засоби реалізації програмної системи для завантаження та зберігання неструктурованих даних;

— розробити архітектуру та структуру веб-застосування для завантаження та зберігання неструктурованих даних;

— розробити веб-застосування для завантаження та зберігання неструктурованих даних.

Призначення програмного продукту має полягати в:

— зборі, накопиченні, зберіганні неструктурованих даних (файлів);

— додаванні до файлу унікального опису та додаткової інформації;

— пошуку файлів в системі;

— можливості додавати нові файли до системи;

— забезпеченні швидкої взаємодії користувача з системою.

Програмний продукт повинен мати такі функції:

— додавання неструктурованих даних (файлів);

— додавання тегів та опису до кожного файлу при завантаженні;

— фільтрації даних за назвою, тегами, типом файлу та датою завантаження;

— вивантаження файлів з системи на комп’ютер користувача;

— видалення одного або декількох файлів із системи.

Вхідними даними для веб-застосування є файл, який потрібно

завантажити в систему і його опис.

Вихідними даними для веб-застосування є результати фільтрації та пошуку файлів за критеріями.

Дане веб-застосування призначене для користувачів та організацій, які хочуть мати можливість зберігати свої файли в зашифрованому вигляді та мати до них зручний доступ.

2. СИСТЕМИ ЗБЕРІГАННЯ НЕСТРУКТУРОВАНИХ ДАНИХ

2.1 Зберігання неструктурованих даних

Інформаційний ресурс (ІР) – систематизована інформація чи знання, які мають цінність у певній предметній області та можуть бути використані людиною у своїй діяльності для досягнення певної мети [4]. Ці типи ІР, які використовуються організаціями чи державними структурами, зберігаються в реєстрах. Реєстр – це інформаційно-телекомунікаційна система, призначена для реєстрації, обліку, накопичення, обробки та зберігання інформації про зміст, місцезнаходження, умови доступу до електронних інформаційних ресурсів та задоволення потреб юридичних та фізичних осіб в інформаційних послугах.

На державному рівні існують різні реєстри, в яких зберігається інформація з різними типами доступу до неї. Так само організації та особи використовують реєстри для виконання щоденних завдань.

Для реалізації цього завдання було визначено функціональність такого виду реєстрів:

— завантаження ІР в базу даних;
— зберігання метаінформації про завантажений ІР (ім'я, розширення, розмір, теги, опис файлу, дата завантаження);
— видалення ІР із бази даних;
— видалення всіх ІР із бази даних;
— пошук ІР на основі назви документа, розширення, тегів, дати завантаження;
— відображення даних про ІР користувачеві;
— відображення списку ІР, завантажених у базу даних;
— вивантаження обраного ІР на комп’ютер користувача для подальшої роботи з ним.

2.2 Програмні системи зберігання неструктурованих даних

У процесі пошуку інформації та аналізу існуючих рішень було виявлено, що на даний момент системи для завантаження, зберігання та фільтрації неструктурованих даних у вигляді файлів вже існують, але більшість з них знаходяться в обмеженому доступі й коштують достатньо дорого.

Серед популярних систем файлових реєстрів можна виділити наступні.

«FossDoc» — система електронного документообігу, яка була розроблена в Україні. Система FossDoc призначена для створення електронного архіву документів, організації корпоративного документообігу та автоматизації бізнес-процесів. FossDoc побудована на основі класичної клієнт-серверної архітектури. При наявності необхідних паролів користувач в будь-який момент може отримати доступ до необхідних документів, а легкість пошуку допоможе зробити це швидко. Крім того, створення, зміна, розсилка документації та звітів за допомогою СЕД (система електронного документообігу) заощаджує час користувача. Інтерфейс програми поданий на рисунку 2.1.

«FlyDoc» — це зручний модуль, який зазвичай використовується разом з бухгалтерськими та офісними програмами 1С. Він синхронізує весь документообіг компанії з використанням платформи 1С, автоматично накладає цифровий підпис, обмінюється необхідними документами з контрагентами та подає регулярну звітність до певних контролюючих органів. FlyDoc співпрацює з платформами, стандартизованими як ПТАХ (захищений засіб обміну документами і фінансовою інформацією для вітчизняного бізнесу). Ця система також має функціонал графічного відображення підписів і печаток. Інтерфейс програмного модуля зображений на рисунку 2.2.

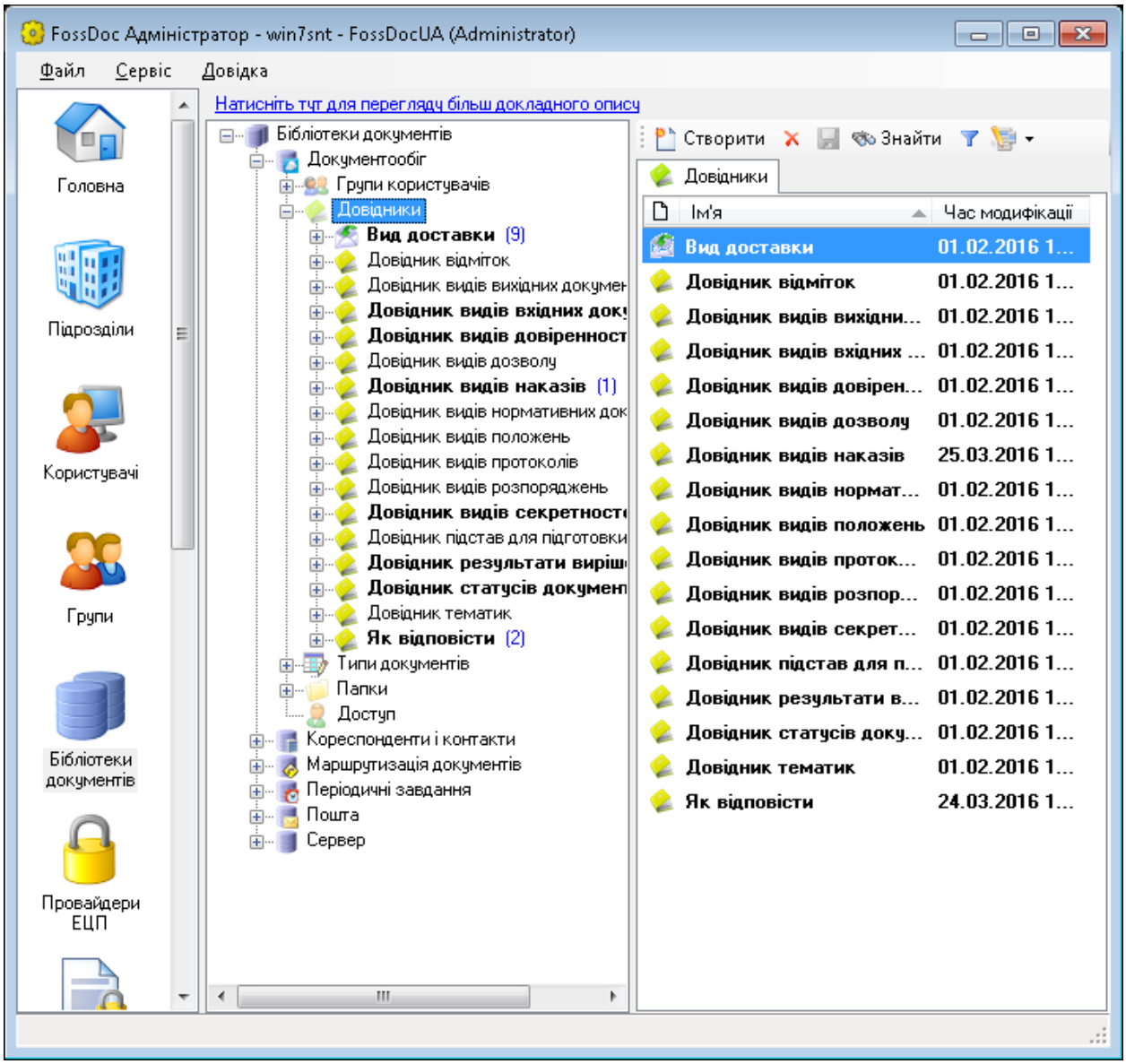


Рисунок 2.1 — Головний екран FossDoc

«Directum» — система управління цифровими процесами і документами з елементами штучного інтелекту. В основі цієї системи лежать технології машинного навчання і комп'ютерного зору. Робота Directum покоління RX можлива не тільки на базі СУБД Microsoft SQL Server, підтримуються також PostgreSQL і комерційна версія Postgres Pro для великих компаній. В якості операційної системи можна використовувати Linux, Mac, Windows. При розробці використовується мова програмування C#. Інтерфейс Directum зображено на рисунку 2.3.

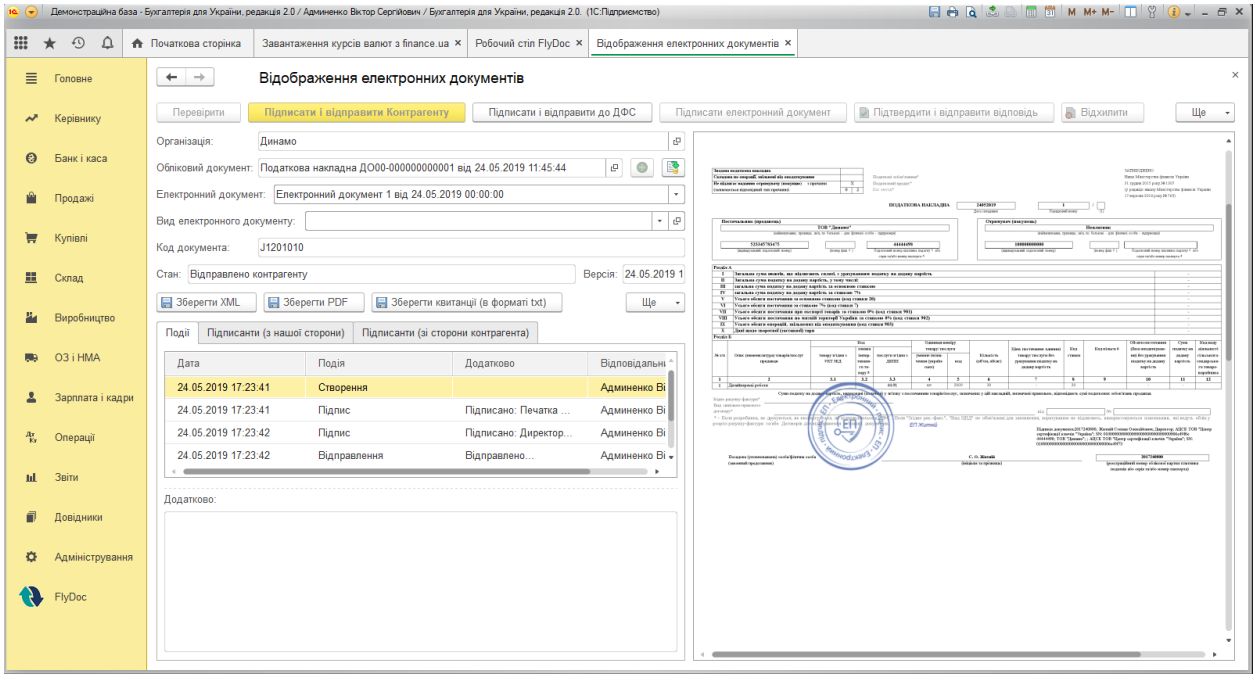


Рисунок 2.2 — Головний екран FlyDoc

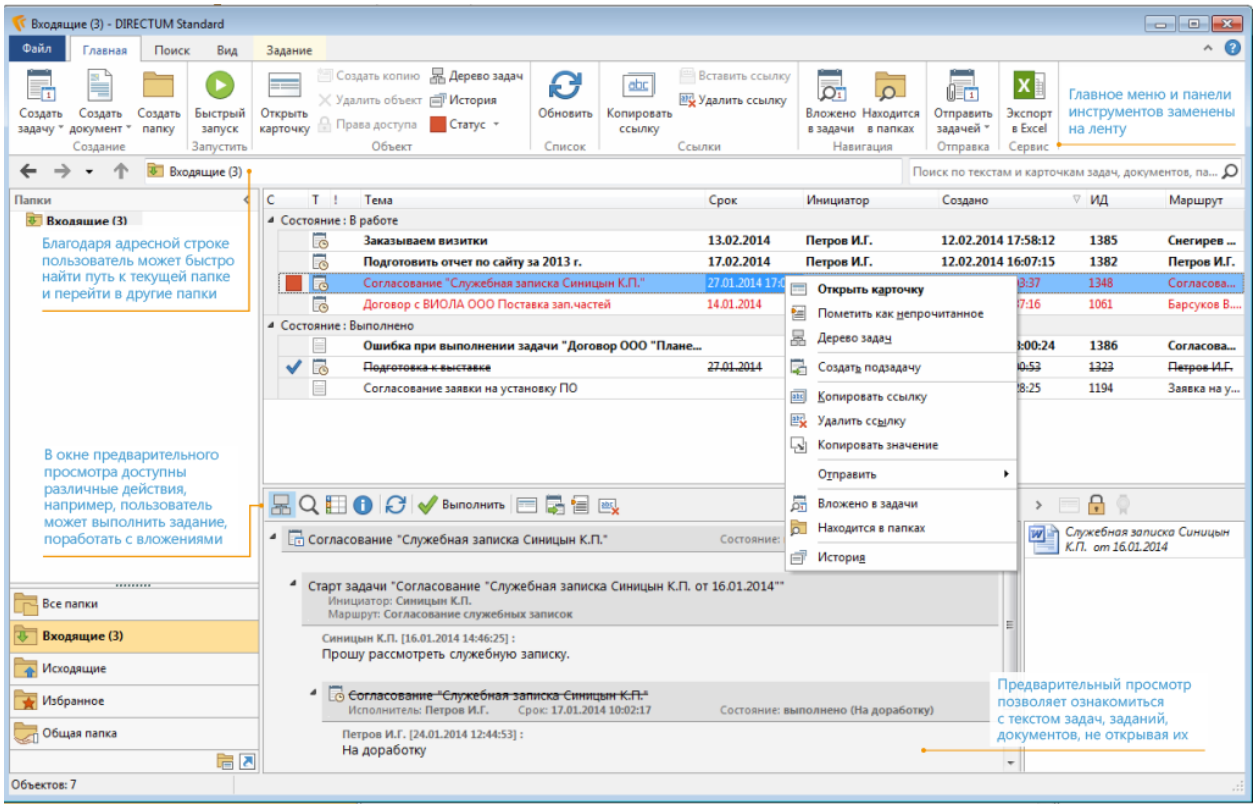


Рисунок 2.3 — Головний екран Directum

Більшість з представлених систем потребують попереднього налаштування перед розгортанням. Ці системи мають низьку швидкість

індексації і знаходяться в обмеженому доступі, тому неможливо повністю оцінити їх можливості. Це є їх головними недоліками.

Також до недоліків можна віднести важкий для розуміння інтерфейс користувача. Він вимагає від користувача впевнених знань в даній тематиці та певний досвід роботи з комп'ютером та програмними системами.

Головною перевагою даних систем є те, що вони надають великий набір інструментів для роботи з документами, що задовольняють найбільш вимогливого користувача, та мають пряму інтеграцію з найпопулярнішими базами даних.

3. ЗАСОБИ РЕАЛІЗАЦІЇ ВЕБ-ЗАСТОСУВАННЯ ДЛЯ ЗАВАНТАЖЕННЯ ТА ЗБЕРІГАННЯ ФАЙЛІВ

Завдання розробки програмного продукту є комплексним завданням, тому потрібно вибрати такі інструменти які б полегшували роботу програміста, надавали усі необхідні інструменти для виконання завдання та отримання результатів, що повністю відповідають потребам користувача.

Вибір клієнт-серверної архітектури для реалізації даного проекту пояснюється тим, що така архітектура дозволяє організовувати систему, яка матиме централізоване сховище інформації, що необхідно для організації системи збереження. Клієнтам пропонується функціонал для користування системою. Веб-портал був обраний клієнтом, оскільки він дозволяє взаємодіяти із системою через будь-який пристрій та віддалений об’єкт, коли сервер і клієнт знаходяться в одній мережі. Було вирішено розробити автономний сервер, оскільки він буде розгортати застосування у внутрішній системі, тобто для доступу до нього не потрібно використовувати підключення до Інтернету, а потрібно лише підключити комп'ютер в мережу, в якій знаходиться БД з файлами. Цей спосіб підвищить надійність системи.

Для зберігання файлів та роботи з ним було вирішено використовувати об’єктну модель мультимодельної постреляційної СУБД InterSystems Caché. Причиною вибору саме цієї СУБД є не тільки надання нею можливості зберігати файли і їх метаданні, а й надання можливості викликати методи, пов’язані з об’єктами [14].

Для розробки інтерфейсу програмного продукту було вирішено використати фреймворк Angular та його модулі, адже такий підхід дозволяє розробити гнучкий програмний продукт, реалізацію якого можна змінювати без зміни функціональності застосунку.

При створенні програмного продукту було використано засоби реалізації, що зображені на рисунку 3.1.

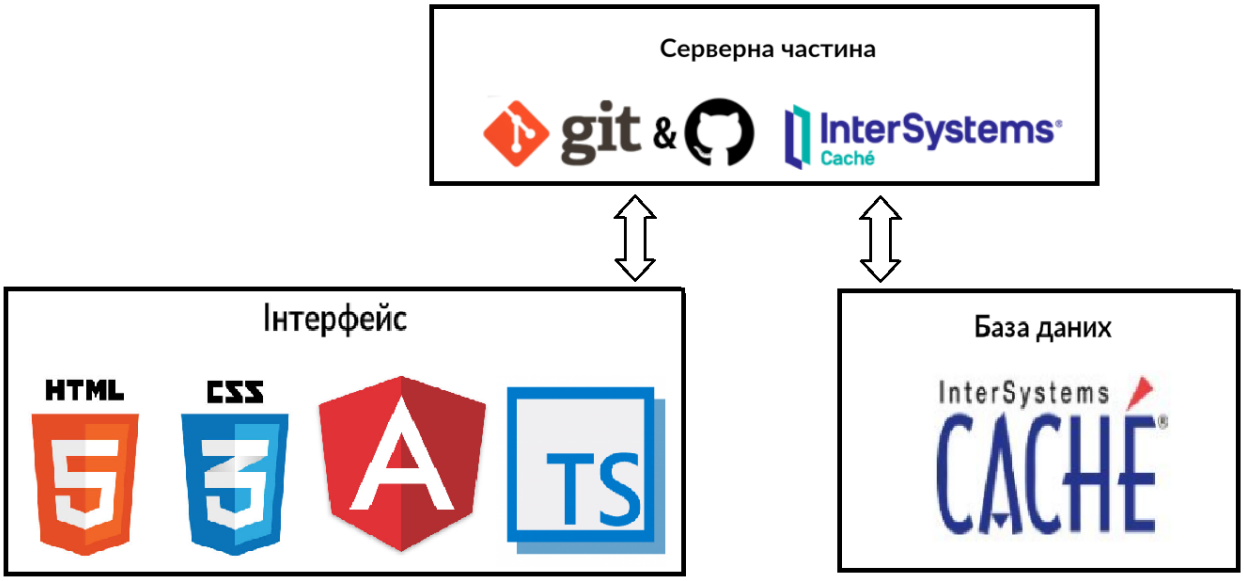


Рисунок 3.1 — Засоби реалізації програмного забезпечення

Як зображено на рисунку 3.1, при створенні програмного забезпечення були використані такі засоби реалізації:

— середовища розробки WebStorm та Studio Caché;

— мову програмування Caché ObjectScript для написання серверної частини;

— фреймворк Angular, для організації архітектури та реалізації взаємодії інтерфейсу з серверною частиною;

— мову програмування TypeScript, для реалізації бізнес логіки інтерфейсу;

— мову розмітки HTML, для реалізації веб-інтерфейсу;

— мову розмітки CSS для розробки графічного інтерфейсу користувача;

— Git для версіювання розробленої системи;

— базу даних СУБД Caché для збереження неструктурованих даних.

Вибір даних технологій пояснюється тим, що застосування не повинно залежати від різних видів пристроїв, з яких будуть здійснювати вхід, та пристрою, на якому буде розгорнута серверна частина програмного

забезпечення.

Вибір зазначених засобів для розробки веб-клієнта, окрім незалежності від різних платформ, обумовлений тим, що здебільшого на державних підприємствах використовуються комп’ютери старого покоління, а отже використання на них програм, що використовують більше комп’ютерної потужності, ніж браузер — неможливо.

3.1 СУБД Caché

InterSystems Caché – сучасна система управління базами даних, яка дозволяє досягти високого рівня продуктивності і масштабованості, який був недоступний в рамках реляційної технології.

Caché – мультимодельна система керування базами даних, яка була спроектована та розроблена компанією InterSystems. Дані в базах даних зберігаються у виді багатомірних масивів. Caché надає програмістам три режими доступу до даних: об'єктний, реляційний чи шляхом прямого доступу до багатомірних структур даних.

Дуже великою проблемою об’єктно-орієнтованих баз даних є значна перевага проектів, що змушені використовувати реляційні бази даних, оскільки при переході на об’єктну технологію, розробники будуть змушені знову починати розробку з самого початку. Також, об’єктна технологія не містить в собі мови запитів, яка буде достатньо розвинутою та стандартизованою. В СУБД Caché реалізовані всі необхідні особливості для об’єктно-орієнтованої технології. Також, вона має полегшити користувачам та розробникам перехід із реляційної технології завдяки особливості Caché – зберігання даних, які не залежать від способу їх представлення.

Архітектура СУБД Caché складається із наступних основних частин, що зображені на рисунку 3.2.

1. Caché SQL – представлення даних у вигляді реляційних таблиць.
2. Caché Direct – забезпечення прямого доступу до ієрархічних даних.

3. Caché Objects – представлення багатомірних структур даних у вигляді об’єктів, у яких інкапсульовані як дані, так і методи їх обробки.

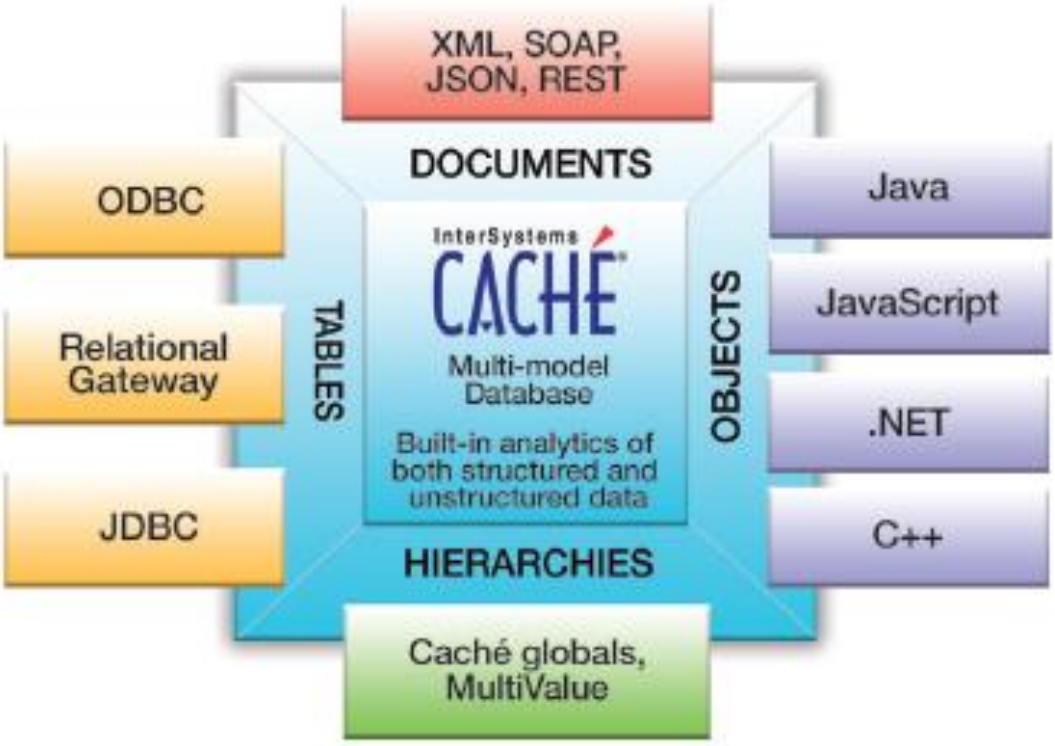


Рисунок 3.2 — Архітектура СУБД Caché

3.2 Фреймворк Angular

Angular – фреймворк для розробки односторінкових застосувань від Google. В наш час, Angular постійно перебуває в процесі розробки, проте вже має значний попит у розробників застосувань, оскільки він забезпечує необхідну інфраструктуру для web застосунку та унікальні рішення, які не вирішують інші фреймворки.

Серед його можливостей можна назвати наступні.

1. Зв’язування.

2. Відкритий сирцевий код (розповсюджується за MIT ліцензією).

3. Шаблони (прототипування).

4. Впровадження залежностей.

5. Клієнтська маршрутизація.

Angular 2 – це найновіший і найбільший фреймворк, який не тільки

допомагає розробляти односторінкові програми, але також використовує багато інших сучасних технологій (таких як TypeScript і ECMAScript6) для забезпечення всієї інфраструктури. Angular 2 також інтегрує багато невеликих бібліотек, аби вирішити недосконалості першої версії.

Web-застосування Angular 2 складається з компонентів, директив та сервісів, що показано на рисунку 3.3. Компоненти керують певною частиною DOM-елементів та містять в собі логіку компонентів відображення й поведінки. Директиви містять в собі лише поведінку, на відміну від компонентів. Сервіси були створені для вставки екземпляра класу в компонентах та директивах застосування. Тому можна сказати, що сервіси реалізують шаблон проектування «Singletone».

Метадані використовуються для позначення класу компонентів, директив або сервісів Angular. Це відбувається за допомогою функції-декоратора, яка розташовується перед класом з необхідними параметрами.

Рисунок 3.3 — Архітектура застосування Angular 2

Зручну ієрархію елементів застосування можуть реалізовувати компоненти, використовуючи інші компоненти як дочірні (рисунок 3.4). Зв'язування даних працює між батьківськими та дочірніми компонентами.

Фреймворк Angular є гнучким та зручним у розробці. Він має безліч можливостей та варіантів для створення новітніх односторінкових веб-застосувань.

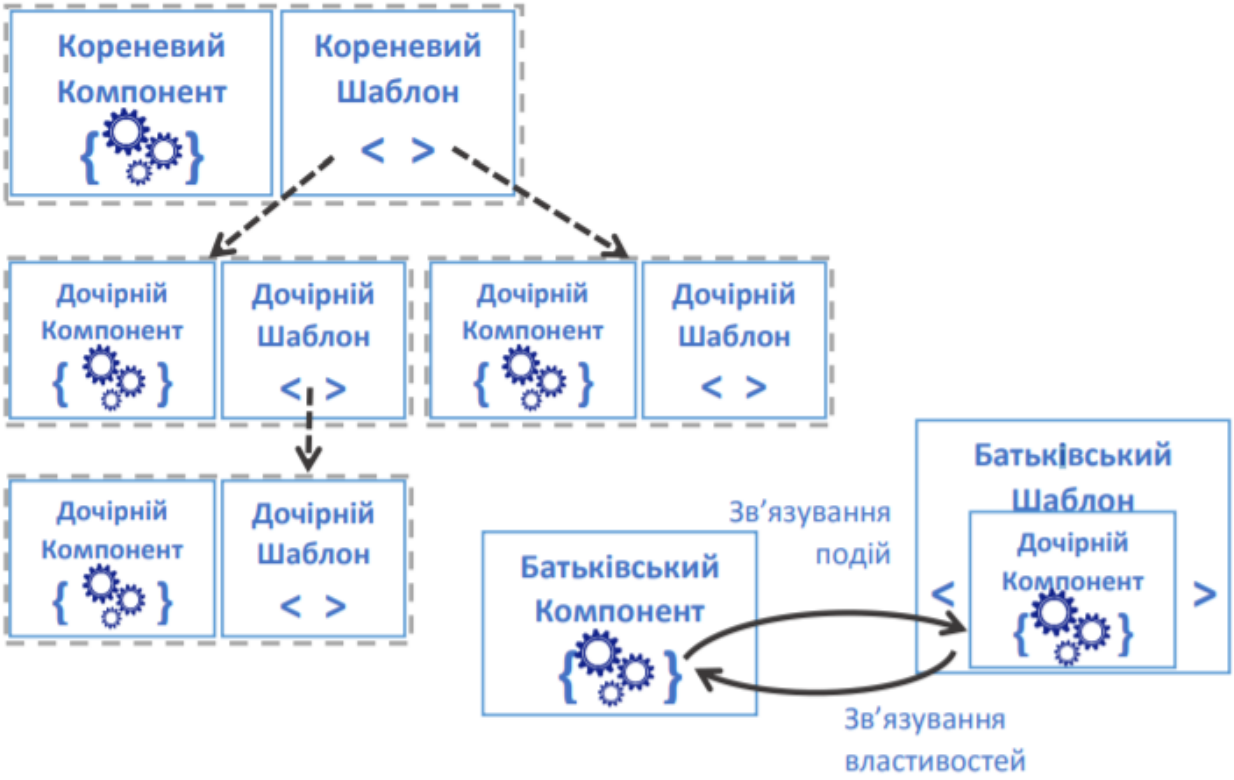


Рисунок 3.4 — Ієрархія та зв'язок компонентів Angular

3.3 Мова програмування TypeScript

Angular рекомендує використання модульного підходу, а саме модулів ES2015, які підтримуються у TypeScript, хоча це не є обов'язковим. Якщо необхідно, використання TypeScript легко замінюється мовою JavaScript, бо код TypeScript завжди компілюється у JavaScript [9]. Використання модулів та TypeScript спрощує розробникам структуризацію, написання та підтримку коду тому відмовлятися від цього не бажано.

Модулі схожі з просторами імен в таких мовах програмування, як C # і C++. Вони мають внутрішній і зовнішній код, тобто код, який може використовуватися іншими модулями. Для подання зовнішнього коду експортованого модуля використовується модифікатор "експорт", наприклад:

```
export class AppComponent { }
```

Для того, щоб підключити частину іншого модулю треба використовувати наступний синтаксис:

```
import { AppComponent } from './app.component';
```

3.4 Мова розмітки HTML

Майже всі web-сторінки, які ми бачимо в браузері, написані саме на мові HTML. Сторінки з кодом HTML оброблюються транслятором браузера та відображаються на екрані у звичайному вигляді для користувача [11].

Разом із каскадними таблицями стилів та вбудованими скриптами, HTML є основною технологією для побудови web-сторінок.

Мова HTML забезпечує засоби для:

- структурування документа з позначенням складу структури тексту: заголовки, абзаци, списки, таблиці, цитати та інше;
- гіперпосилання, що забезпечують отримання потрібної інформації;
- створення форм з якими можна взаємодіяти;
- додавання зображень, аудіо, відео.

У розмітці в HTML розрізняють основні компоненти: елементи з їх атрибутами, базові типи даних, символи. Ці елементи представляють собою основні компоненти для розмітки HTML. Кожен з елементів містить дві важливі властивості: атрибути та їх зміст. Щоб HTML-документ був визнаний валідним, існують стандартні вказівки щодо властивостей елемента, які треба обов’язково виконувати.

Семантично вірний HTML може значно покращити доступ до веб-документів.

Для редагування коду можна використовувати різні текстові редактори, щоб переглянути розмітку HTML документа.

3.5 Мова розмітки CSS

Каскадні таблиці стилів CSS — спеціально розроблена мова, яка використовується для структурування сторінок, написаних мовами розмітки даних, такими як HTML.

CSS використовують для візуального відтворення сторінок, написаних мовами розмітки даних HTML та XHTML. Вона також може бути застосована до різних видів XML-документів. Зовнішній вигляд HTML-документів визначають саме каскадні таблиці стилів.

CSS застосовується розробниками та користувачами веб-сторінок, для визначення кольору, шрифту та інших візуальних аспектів. Головною перевагою каскадних стилів є розділення змісту сторінки (або контент, наповнення, зазвичай HTML, XML або подібна мова розмітки) від вигляду документа (що описується в CSS) [12].

Саме цей поділ покращує відображення та візуальний вигляд контенту. Це забезпечує більше можливостей у керуванні відображенням контенту в різних браузерах, структурування та спрощення вмісту, видалення дублікатів.

Залежно від використаного CSS вигляд одного і того ж документа з розміткою може виглядати по різному.

Основними перевагами CSS є:

— стилі усього сайту або його частин містяться в одному документі з розширенням .css, що дає можливість швидко змінювати дизайн та відображення сторінок;

— сторінки займають менше в об’ємі та виглядають структурованими;

— браузери можуть кешувати (запам’ятовувати) інформацію про стилі і додавати її для нових сторінок, що прискорює завантаження сторінок і зменшує розмір сторінки;

— одна таблиця стилів може використовуватися для відображення безлічі документів;

— надає сторінці структурований вигляд;

— використовується для різних носіїв інформації (екран, друк, і т. д.).

Для прискорення розробки дизайну було використано вже готовий набір стилів Bootstrap 3 [13].

3.6 Пакетний менеджер NPM

NPM (Node Package Manager) – пакетний менеджер для мови програмування JavaScript. NPM являє собою online склад для збереження та відображення проектів з відкритим вихідним кодом на Node.js. Також, NPM виконує функції утиліти для командного рядка аби взаємодіяти зі сховищем. Функції утиліти допомагають встановлювати пакети та керувати актуальними версіями і налаштовувати залежності. Безліч бібліотек Node.js та застосунків публікуються за допомогою npm кожного дня. Кожен знайдений пакет може бути встановлений за допомогою спеціальної команди для командного рядка.

Головною метою NPM є автоматизоване управління залежностями та пакетами. Всі залежності проекту вказуються в файлі package.json. Перед початком роботи над проектом, користувач має виконати тільки одну команду npm install і всі залежності встановляться у його проекті. Також можна вказувати актуальні версії як проекту, так і пакетів.

3.7 Система контролю версій Git

Git – це система контролю версій, яка дає можливість зберігати певну кількість версій файлу, певного проекту чи деяких його складових. Розробники програми можуть бачити зміни, які були додані до системи розробником проекту під час роботи над ним. Git використовується для додавання та майбутнього використання коду методом створення відбитків, так званих коммітів, у базі. Git визнаний найбільш надійною та високопродуктивною системою контролю версій, що забезпечує гнучкі засоби нелінійної розробки, які ґрунтуються на відгалуженні та злитті гілок. Аби

зберегти цілісність та стійкість історії до змін до минулих дат використовуються криптографічні методи. Ще є можливість прив'язати цифрові підписи авторів до тегів і коммітів.

Значна кількість дій виконується в локальній файловій системі без інтернет підключення. Вся історія змін зберігається локально і при необхідності вивантажується у віддалений репозиторій.

Сайт github.com водночас є веб-сервісом для хостингу завантажених проектів, так і соціальною мережею розробників за певними інтересами. Користувачі додають свої проекти та презентують їх за допомогою цього сервісу. У кожного користувача є власний створений окремий репозиторій, де він може публікувати свої роботи. GitHub на даний момент – найпопулярніший сервіс контролю версій у світі.

4. ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ ЗАВАНТАЖЕННЯ ТА ЗБЕРІГАННЯ ФАЙЛІВ

Для того, щоб автоматизувати процес зберігання неструктурованих даних потрібна система, яка надасть функціонал для взаємодії користувача з сервером.

4.1 Архітектура та структура програмної системи зберігання неструктурованих даних

Розроблена система збереження та роботи з файлами складається з декількох підсистем (модулів) та бази даних.

Дана система являється клієнт-серверною, для її використання потрібен доступ до мережі Internet та браузер.

Для створення програмної системи пошуку та зберігання неструктурованих документів було написано кілька програмних модулів різними мовами програмування (рисунок 4.1).

Пошук та зберігання у великій кількості неструктурованих документів — це достатньо складна та трудомістка задача, тому програмне забезпечення, що вирішує дану задачу, повинно ефективно та швидко справлятися з нею.

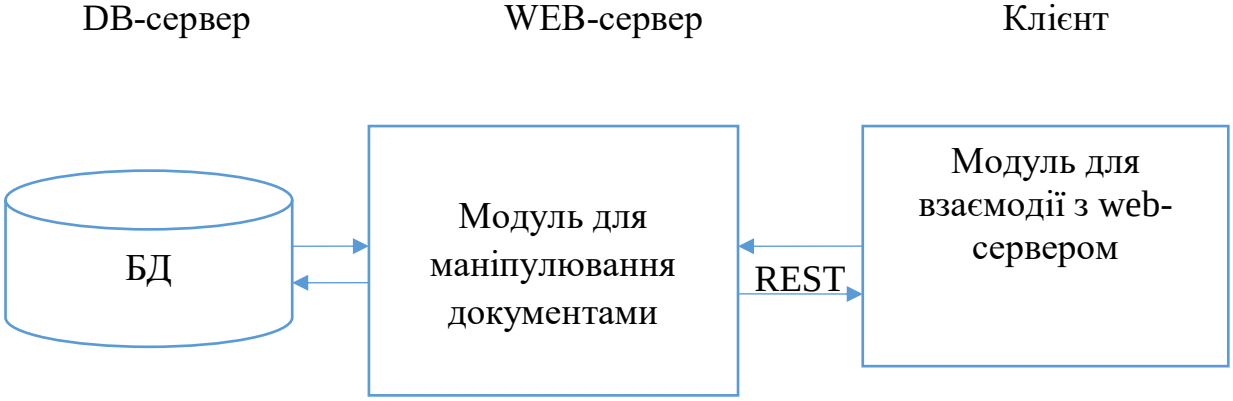


Рисунок 4.1 — Архітектура системи зберігання та пошуку неструктурованих даних

Модуль маніпулювання документами призначений для збереження, повернення та видалення документів ті їх опису з БД. Він взаємодіє з базою даних. Вхід до цього блоку – це файл та його метадані, такі як опис, теги, дата завантаження, розширення та назва файлу.

Модуль для взаємодії з веб-сервером призначений для коректної обробки отриманих даних з сервера та відображення їх для користувача.

Також в модулі маніпулювання документами повинні опрацьовуватись усі помилки, що виникали під час роботи інших модулів.

4.2 Функціональна декомпозиція програмної системи зберігання неструктурованих даних

Опис функціональності і поведінки розробленої системи представлено на діаграмі прецедентів (рисунок 4.2).

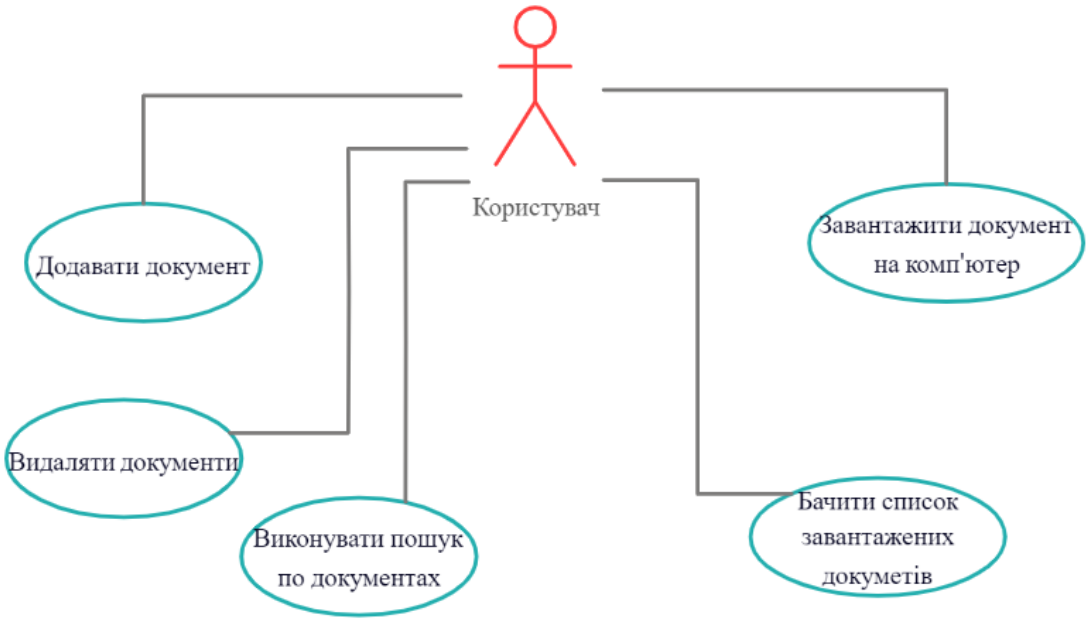


Рисунок 4.2 — Діаграма прецедентів системи

На рисунку 4.2 можна побачити, що у системи є лише один актор — це користувач даної системи. За допомогою інтерфейсу, користувач має можливість додати до бази даних документи з описом, видалити декілька

документів одночасно, виконати пошук по документах та отримати результати пошуку.

4.3 Діаграма класів бази даних системи зберігання неструктурованих даних

Для роботи з базою даних (БД) файлів було використано СУБД Caché. Файл для збереження в БД описується за допомогою класу, представленого на рисунку 4.3.

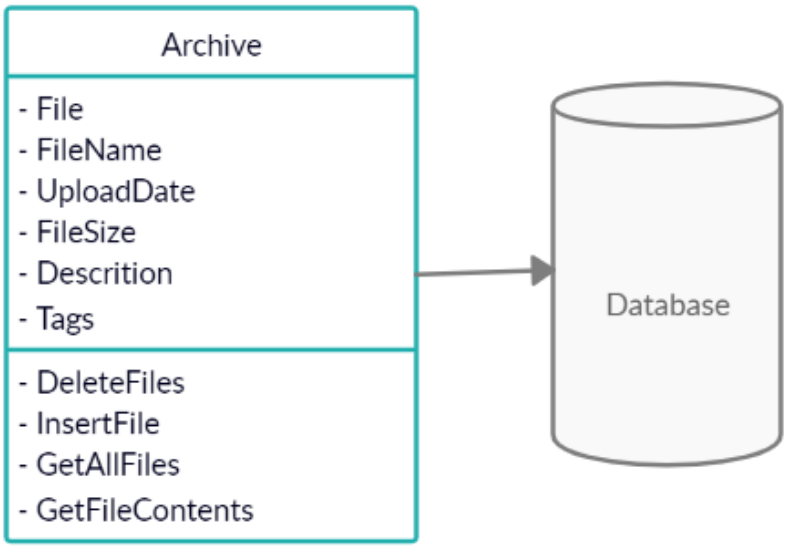


Рисунок 4.3 — Клас Archive

Клас Archive (рисунок 4.4) включає в себе наступні поля:

— FileName – рядок, який містить унікальне ім’я документа, є обов’язковим для заповнення;

— File – бінарний потік, який містить сам файл, є обов’язковим для заповнення;

— UploadDate – дата, яка містить дату завантаження документа до системи, є обов’язковим для заповнення;

— FileSize – число, яке містить розмір документа, є обов’язковим для заповнення;

— Description – рядок, який містить короткий опис документа;

— Tags – теги для документа.

```
Class Archive.FileDesc Extends %Persistent
{

Property File As %Stream.GlobalBinary [ Required ];

Property FileName As %String [ Required ];

Property UploadDate As %TimeStamp [ Required ];

Property FileSize As %Float [ Required ];

Property Descrition As %String;

Property Tags As array Of %String;
```

Рисунок 4.4 — Визначення класу Archive

4.4 WEB-клієнт для роботи з RESTful сервісом завантаження та зберігання файлів

Web-клієнт було написано за допомогою засобів Angular 9 Framework. Він дає можливість користувачу взаємодіяти з розробленими серверними модулями програмної системи пошуку та зберігання неструктурованих документів.

Файли відображення (View) приймають моделі, які будуть на них відображатися. Прийом моделей на сторінці виконується безпосередньо під час створення сторінки на сервері або під час виконання ajax запиту до серверу.

Оскільки програмний продукт реалізований за допомогою Angular Framework, це означає, що йому не потрібно оновлювати сторінки під час кожного запиту на сервер. Сторінки динамічно оновлюють дані. Angular Framework має структуру проекту папок , як показано на рисунку 4.5.

Структура проекту складається з наступних модулей: src, node_modules, e2e.

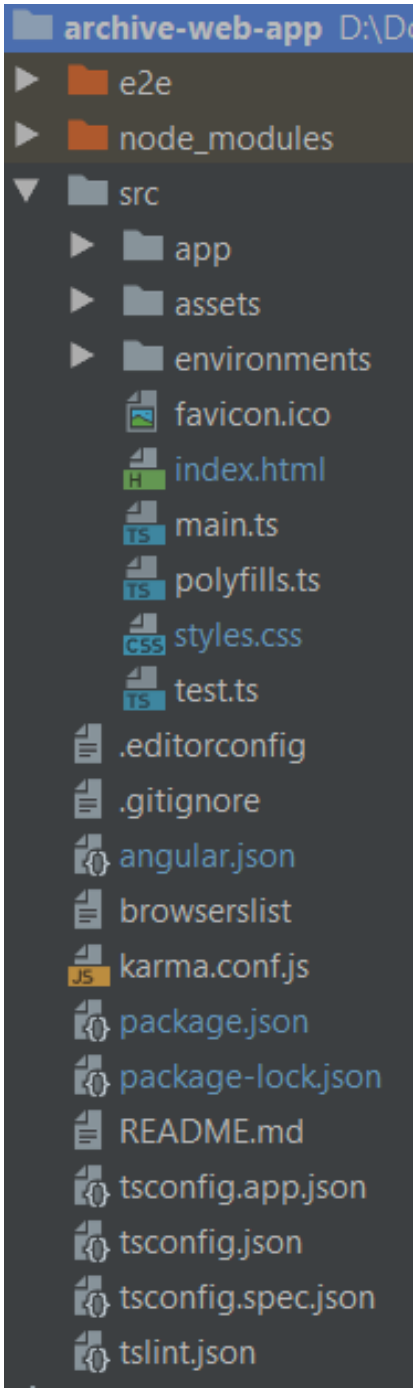


Рисунок 4.5 — Структура проекту

Модуль src поділяється на app, assets та enviroments.

Файли, що відповідають за роботу серверної частини:

— «package.json» – файл пакетного менеджеру npm, який містить перелік та версії залежностей які необхідні для роботи застосування, а також деякі команди для збірки та запуску сервера;

— «tsconfig.json» – конфігурація компілятора TypeScript;

— «typings.json» – ідентифікує TypeScript визначення;

— «systemjs.config.js» – конфігурація завантажувача модулів SystemJS.

Файли, що відповідають за роботу клієнтської частини:

— «index.html» – головна сторінка веб-клієнта, яка містить лише основні теги та скрипти, які асинхронно завантажують потрібні ресурси для повного відображення сторінки;

— «main.ts» – містить код, який завантажує інфраструктуру фреймворка Angular, сервіси та запускає застосування з головним компонентом «AppComponent». За це відповідає функція «bootstrap»;

— «styles.css» – містить загальні спільні стилі, необхідні для роботи застосування.

Кожен компонент має «*.ts» файл який містить його TypeScript код, та може мати «*.html» та «*.css» файли, що містять HTML шаблон та CSS стилі компонента відповідно. За рекомендованими нормами кодування Angular 2 компоненти мають містити у назві слово «component».

4.4.1 Архітектура web-клієнта

При розробці графічного інтерфейсу було використано шаблон проектування MVVM (рисунок 4.6).

MVVM – більш сучасна версія MVC, яка більше підходить під сучасні особливості розробки графічного інтерфейсу. В MVVM контролер відповідає за оновлення представлення, та має назву модель представлення (ViewModel).

1. Модель (Model) – представляє данні й бізнес-логіку предметної області.

2. Модель представлення (ViewModel) – абстрактне відображення представлення.

3. Представлення (View) – відображає модель представлення.

Клієнтське застосування складається з кореневого компоненту «AppComponent», що містить налаштування клієнтської маршрутизації, та є контейнером для компонентів, що задають поточне представлення, тобто вигляд застосування.

«FilesListComponent» - компонент-представлення за замовчуванням. Містить кнопку виклика до компонента представлення «UploadFilePopUpComponent», фільтри за якими можна відсортувати список та список завантажених документів. Також містить фільтри за назвою, датою, тегами та розширенням файлу.

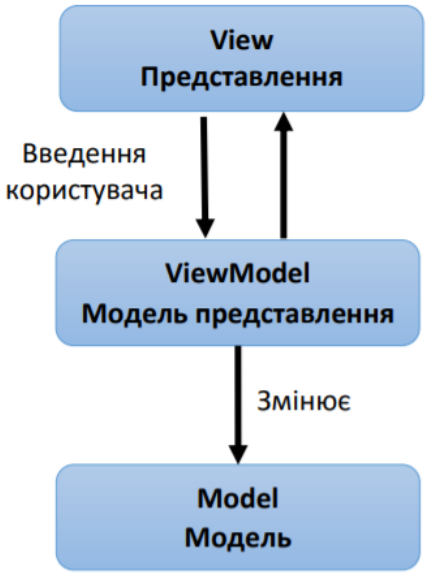


Рисунок 4.6 — Шаблон проектування MVVM

«UploadFilePopUpComponent» - компонент-представлення, що відображає спливаюче вікно, де можна завантажити файл, вказати його опис, додати потрібні теги, та кнопку відправки або відміни дії та повернення до головного екрану застосунку «FilesListComponent».

При завантаженні сторінки «FilesListComponent» автоматично відправляє запит на сервер для отримання усього списку документів. Для взаємодії з сервером і відправлення запитів по протоколу http застосовується клас HttpClient. Цей клас визначає ряд методів для відправки різного роду запитів: GET, POST, PUT, DELETE. Для цього в файлі модуля AppModule повинен бути імпортований клас HttpClientModule з пакету "@ angular / common / http" (рисунок 4.7).

Для відправки усіх запитів на сервер було визначено окремий сервіс FilesService.

Рисунок 4.7 — модуль AppModule

4.4.2 Відправка Get-запитів

Для відправлення запиту на отримання усіх документів в сервісі FilesService використовується метод getFiles.

Для виконання get-запиту у об'єкта HttpClient викликається метод get(), в який передається адреса запиту – json-файл з даними. Метод getFiles відправляє запит на сервер за роутом `archive/files` та отримує список усіх завантажених файлів до системи.

Для завантаження файлу з бази даних було реалізовано метод downloadFile який як аргумент приймає унікальний ідентифікатор файлу.

У відповідь отримуємо файл для завантаження.

4.4.3 Відправка Post-запитів

Для відправлення на сервер завантаженого файлу з додатковою інформацією про нього, такою як набір тегів та опис, було реалізовано два методі uploadFileOptions та uploadFile. В метод uploadFileOptions, в тіло запиту додається вся інформація про завантажений файл, але без самого тіла файлу. Ці всі данні відправляються в об'єкті body. У відповідь на цей запит отримується ідентифікатор, по якому далі завантажуэться сам файл до системи.

Для відправки застосовується метод http.post(), в який передається адреса сервера і відправляється об'єкт.

Для завантаження самого файлу було написано метод uploadFile, за допомогою якого відправляється саме тіло файла за певним ідентифікатором до бази даних. Цей метод викликається після uploadFileOptions.

4.4.4 Відправка Delete-запитів

Даний запит направлений на видалення певного файлу з системи за id. У запиті передається список усіх ідентифікаторів за якими потрібно видалеити файли. Тип запитів не приймає тіло, тому єдиний варіант - це тільки вписати дані в URI.

В метод deleteFileById передається список ідентифікаторів з компоненту FilesListComponent звідки і викликається цей метод. Для цього застосовується метод http.delet(), в який передається адреса сервера, список ідентифікаторів і відправляється на сервер.

4.4.5 Реалізація API

Набір методів для взаємодії з базою даних та клієнтським застосунком було написано на мові ObjectScript. Набір методів та роутів реалізовано в класі Archive.Broker. Цей клас включає в себе перелік можливих роутів, зображений на рисунку 4.8

Для отримання усіх файлів було реалізовано метод GetAllFiles, у якому виконується SQL запит до бази даних для отримання усіх завантажених раніше файлів.

```
<Routes>
        <Route Url="/files" Method="GET" Call="GetAllFiles"/>
        <Route Url="/file/:id/contents" Method="GET" Call="GetFileContents"/>
        <Route Url="/file/:id" Method="GET" Call="GetFileInfo"/>

        <Route Url="/file/:id" Method="DELETE" Call="DeleteFile"/>
        <Route Url="/files/:ids" Method="DELETE" Call="DeleteFiles"/>

        <Route Url="/file" Method="POST" Call="InsertFile"/>
        <Route Url="/file/:id" Method="POST" Call="InsertFileContents"/>
</Routes>
```

Рисунок 4.8 — Роути класу Archive.Broker

Відповідь на цей запит відправляється у вигляді масиву на клієнтський застосунок з інформацією про усі завантаженні файли, з такими полями як опис файлу, його назва, список тегів, розмір файлу, дата завантаження. Клієнтський застосунок обробляє ці данні та відображає користувачу.

Для видалення усіх обраних на клієнті файлів використовується функція DeleteFiles. В ній виконується прохід по масиву ідентифікаторів файлів і видаляється кожен об’єкт. Якщо ідентифікатор не знайдений, то метод поверне повідомлення про помилку.

Для завантаження інформації про файл було розроблено метод InsertFile, який у відповідь повертає ідентифікатор з бази даних по якому були записані параметри файлу.

Після запиту InsertFile на сервер приходить запит з самим файлом, реалізація такого методу знаходиться під назвою InsertFileContents. Цей метод за переданим ідентифікаторів завантажує в базу файл та при необхідності повідомляє, якщо цього ідентифікаторів не існує у системі.

Для завантаження файка з бази даних було реалізовано метод GetFileInfo, який повертає на клієнт всю інформацію про файл. Для отримання тіла файлу використовується метод GetFileContents.

4.5 Взаємодія клієнта та сервера

Передача даних між клієнтом та сервером відбувається за допомогою REST API технології.

REST API — це ідеологія побудови API, яка розшифровується як Representational State Transfer API. Вона ґрунтується на принципах:

— клієнт-серверної архітектури;

— єдиного інтерфейсу.

Приклад діаграми взаємодії між клієнтом та сервером для запису файлу і всієї супутньої інформації в базу даних показана на рисунку 4.9.

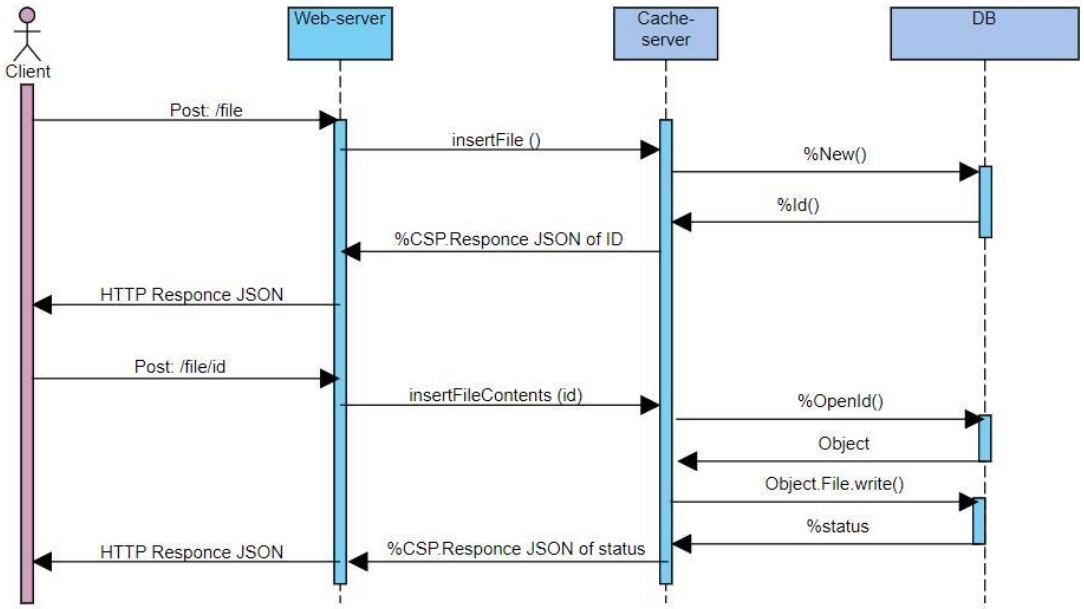


Рисунок 4.9 — Діаграма взаємодії клієнта з сервером для запису файлу в БД

Після кожного запиту повертається відповідь і статус запиту або відповідь від сервера у вигляді даних JSON. Варіанти статусів можуть бути такі:

— 1xx Інформаційні;
— 2xx Успішні операції;
— 3xx Перенаправлення;
— 4xx Клієнтська помилка;
— 5xx Серверна помилка.

5. МЕТОДИКА РОБОТИ КОРИСТУВАЧА З ПРОГРАМНОЮ СИСТЕМОЮ ЗАВАНТАЖЕННЯ ТА ЗБЕРІГАННЯ ФАЙЛІВ

Для роботи системи необхідно розгорнути сервер та знати його адресу. Єдине програмне забезпечення необхідне для підключення до системи — браузер.

5.1 Системні вимоги для роботи програмної системи завантаження та зберігання файлів

Для роботи користувача з розробленою програмною системою йому потрібні мінімальні потужності апаратного забезпечення (вимоги наведені в таблиці 5.1).

Таблиця 5.1. Вимоги до апаратного забезпечення клієнта

Пристрій	Характеристика
Процесор	Intel ® Core ™ 2 / 2 Duo / Pentium ® / Celeron ® / Xeon™ / i3 / i5 / i7 чи AMD 6 / Turion ™ / Athlon ™ / Duron ™ / Sempron ™ з тактовою частотою не нижче 1.5 GHz.
Оперативна пам’ять (RAM − Random Access Memory)	Рекомендовано не менше 1 RAM
Швидкість з’єднання з інтернет	Рекомендовано не менше 128 кб/сек

Підтримувані апаратні архітектури:

- 32-розрядна (x86);
- 64-розрядна (x64).

Нижче розглянемо вимоги до апаратного та програмного забезпечення комп’ютера, який буде виконувати функції сервера, на якому буде розгорнуто програмний застосунок (вимоги наведено в таблиці 5.2 та в таблиці 5.3 відповідно).

Таблиця 5.2. Вимоги до апаратного забезпечення сервера

Пристрій	Характеристика
Процесор	Intel ® Core ™ / i3 / i5 / i7 чи AMD K5 ™ / K6 ™/ K7 ™ з тактовою частотою не нижче 3.0 GHz.
Оперативна пам’ять (RAM − Random Access Memory)	Рекомендовано не менше 4 RAM
Швидкість з’єднання з інтернет	Рекомендовано не менше 512 кб/сек
Вільне місце на жорсткому диску	Рекомендовано 50 гб

Таблиця 5.3. Вимоги до програмного забезпечення сервера

Розділ	Назва
Операційна система	Windows Vista, 7, 8, 8.1, 10, Mac OS 10.10, 10.11, 10.12, Linux
Передустановлені програми та компоненти	СУБД Caché

5.2 Сценарії роботи з API

В ході виконання дипломної роботи було розроблене API за допомогою якого можна обмінюватися запитами з сервером. Для того, щоб перевірити роботу сервера достатньо скористатися будь-яким клієнтським менеджером

для відправки REST запитів.

Як вже зазначалося на рисунку 4.11 було запрограмовано такі маршрути:

— /files метод - GET за допомогою якого можна отримати увесь список завантажених файлів (рисунок 5.1);

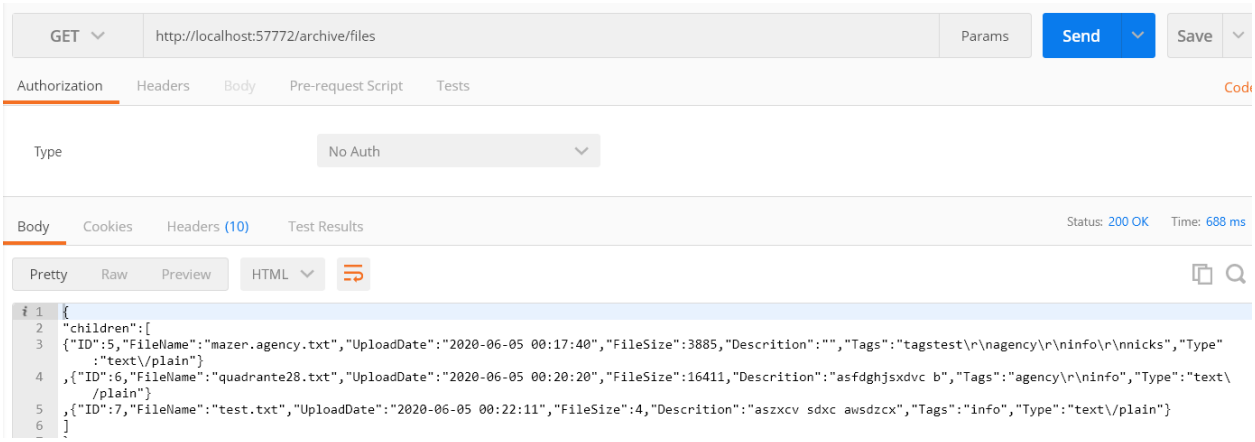


Рисунок 5.1 — Робота маршруту /files метод – GET

— /file/:id/contents метод – GET за допомогою якого можна отримати вміст обранного файла за індифікатором (рисунок 5.2). Метод повертає у відповітдь об’єкт з назвою, розширенням, тегами, описом та датою завантаження файла;

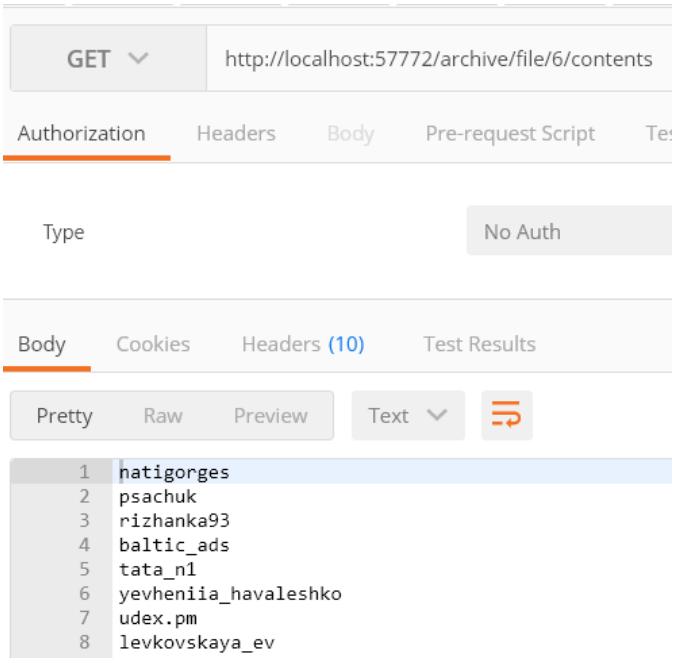


Рисунок 5.2 — Робота маршруту /file/:id/contents метод – GET

— /file/:id метод – GET за допомогою цього методу можна отримати сам файл для завантаження на власний комп’ютер (рисунок 5.3);

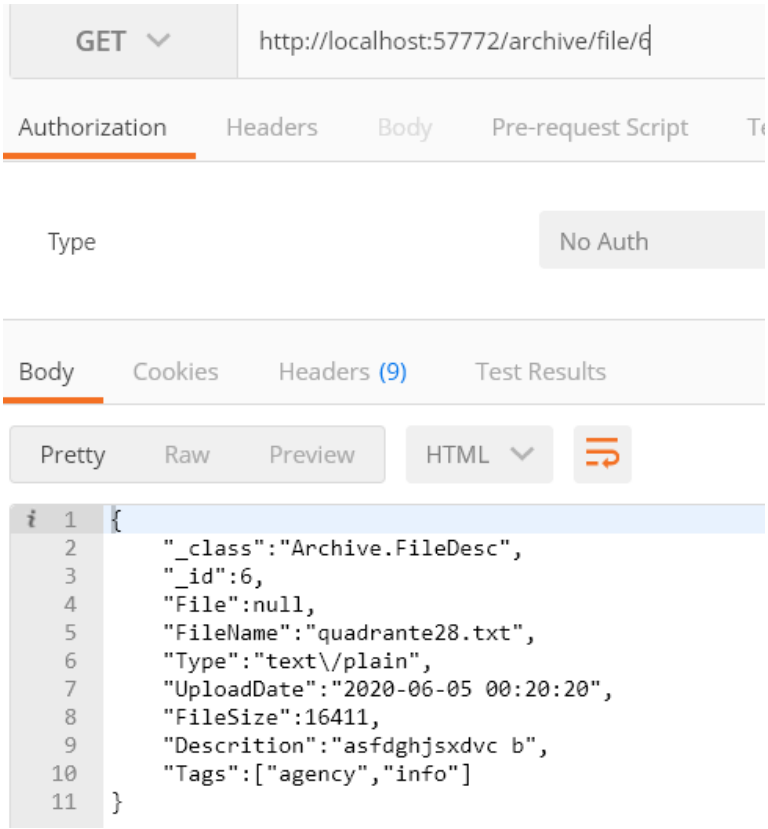


Рисунок 5.3 — Робота маршруту /file/:id/ метод – GET

— /file/:id метод – DELETE метод забезпечує видалення одного файлу файлу з бази даних (рисунок 5.4);

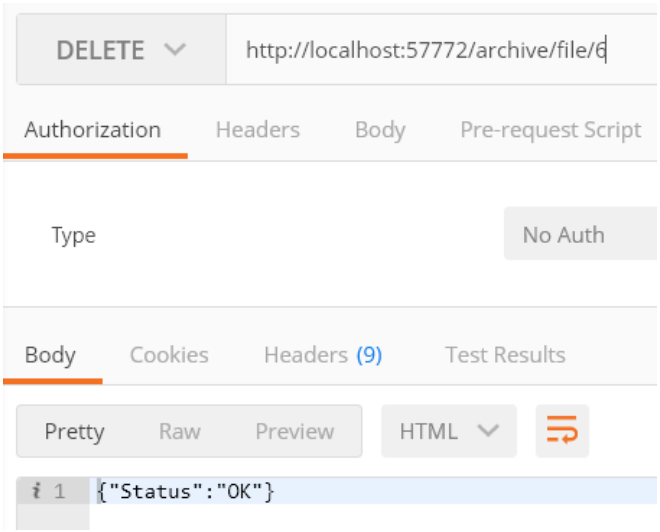


Рисунок 5.4 — Робота маршруту /file/:id метод – DELETE

— /files/:ids метод – DELETE метод забезпечує видалення декількох файлів з бази даних одночасно (рисунок 5.5);

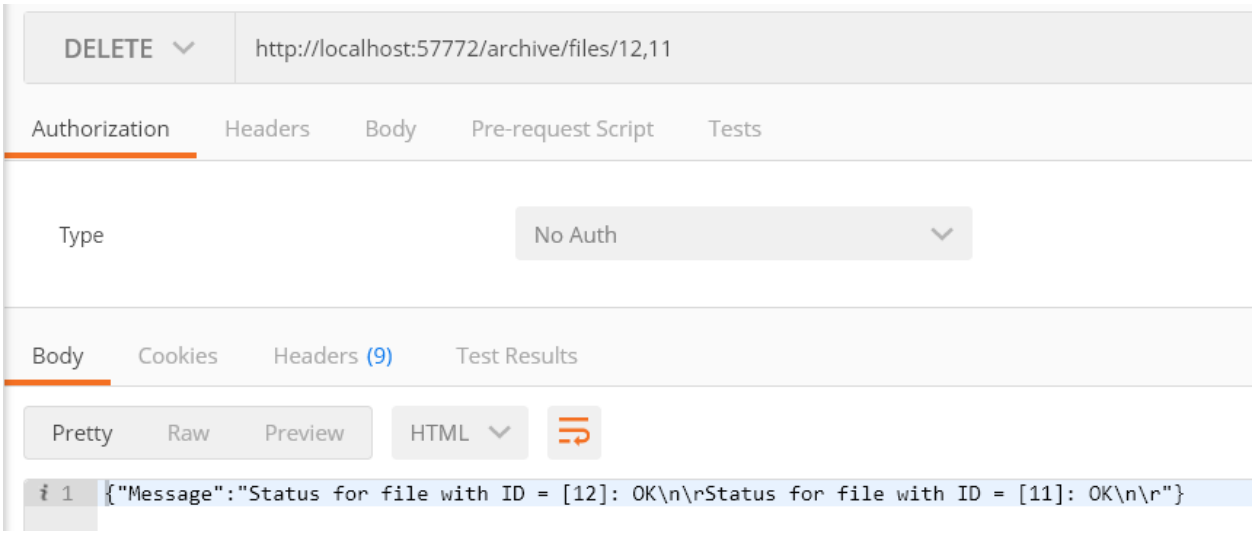


Рисунок 5.5 — Робота маршруту /files/:ids метод – DELETE

— /file метод- POST за допомогою цього метода можна відправити данні про завантажений файл, та у відповідь отримати ідентифікатор нового файлу (рисунок 5.6);

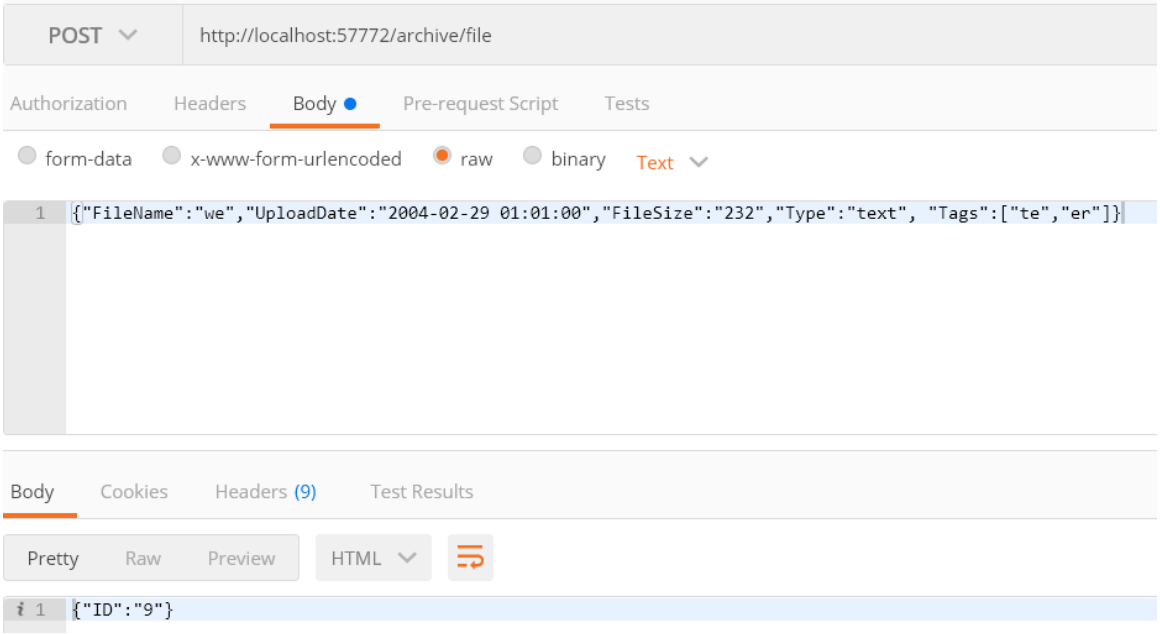


Рисунок 5.6 — Робота маршруту /files/:ids метод – POST

— /file/:id метод-POST отриманий ідентифікатор з попереднього методу використовується при відправленні самого файлу до бази даних (рисунок 5.7).

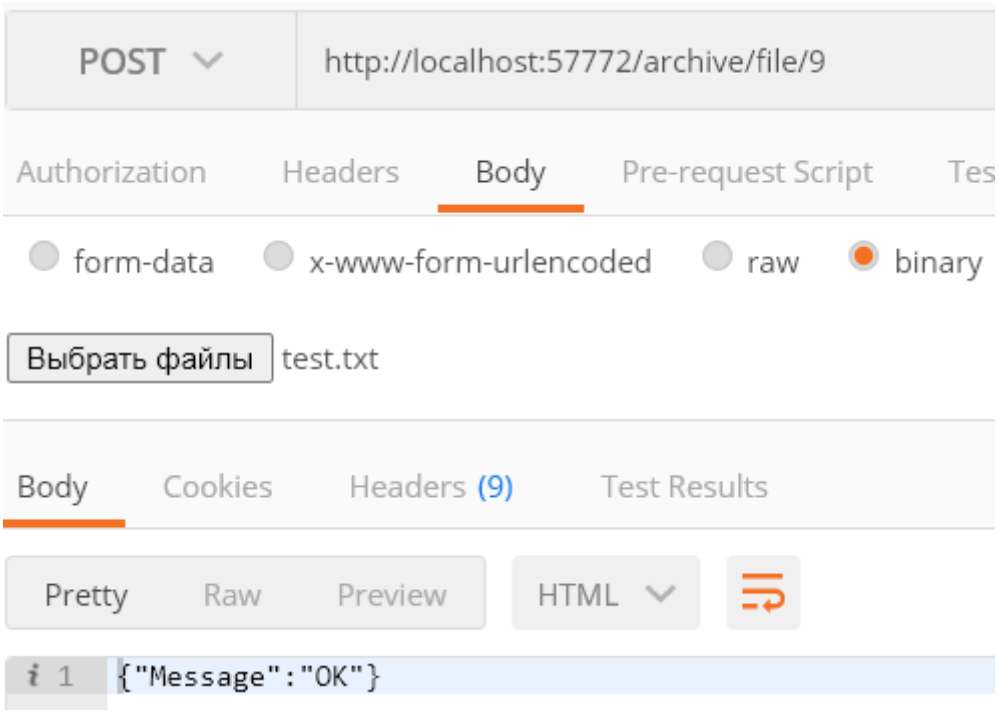


Рисунок 5.7 — Робота маршруту /file/:id метод- POST

5.3 Сценарії роботи користувача з системою завантаження та зберігання файлів

Перше, що бачить користувач при потраплянні на ресурс – це головну сторінку застосування (рисунок 5.7).

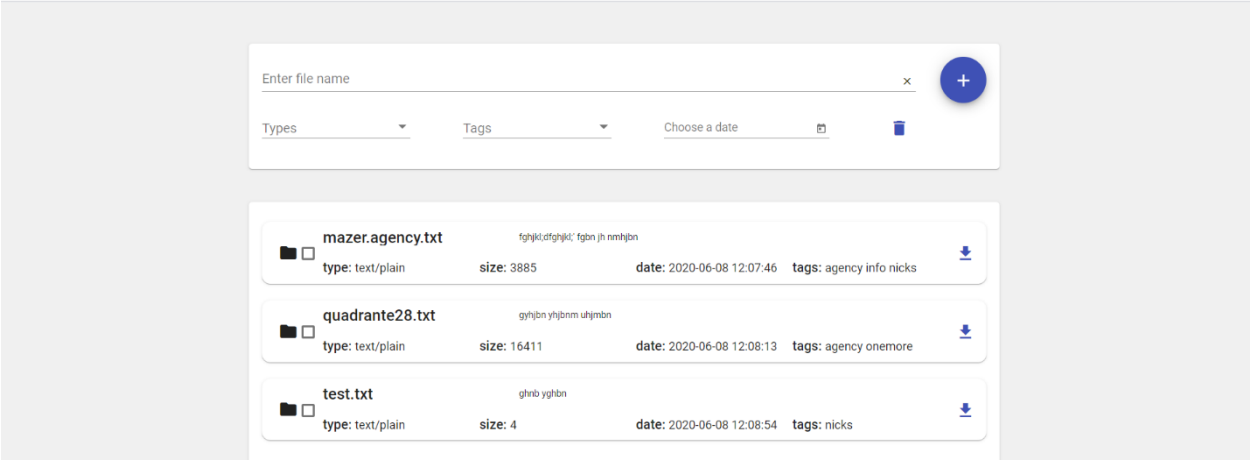


Рисунок 5.7 — Головна сторінка

На головній сторінці застосування знаходиться таблиця, в якій користувач може переглядати завантаженні раніше документи, завантажувати нові документи, а також видаляти непотрібні документи. В цій таблиці користувач може побачити усю інформацію про даний документ — тип файлу, його опис, дату створення, назву файлу, розмір та унікальні теги файлу.

Для того, щоб завантажити новий документ, користувач має натиснути кнопку “+”, після чого з’явиться вікно (рисунок 5.8), де користувач має обрати новий файл на диску, вказати теги та опис файлу за бажанням, та натиснути кнопку “Ok”. Після завантаження документу вікно закриється і оновиться список файлів, де вже буде знаходиться завантажений файл. Якщо користувач передумав завантажувати файл, тоді потрібно закрити вікно, натиснувши на кнопку “Cancel”.

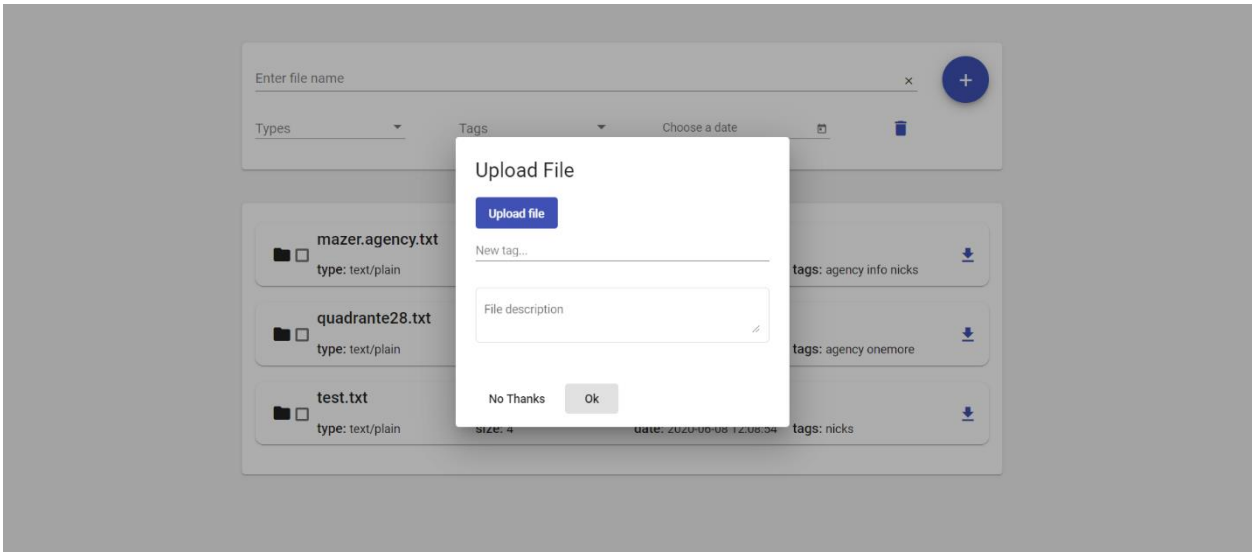


Рисунок 5.8 — Меню додавання файла в систему

Особливістю поля вводу для тегів є те, що теги можна вибрати з уже існуючого списку, а можна створити новий, написавши її назву у те ж саме поле.

На головній сторінці проектувальника знаходиться список усіх документів, на яких показані дати створення, теги та інформація про документ, включаючи назву документа. У кожного документа є чекбокс, вибравши його

документ може бути видалений за допомогою спеціальної кнопки (рисунок 5.9). Також, кожен файл має спеціальну кнопку для збереження документа на компьютер. Для цього потрібно натиснути на кнопку з зображеням завантаження.

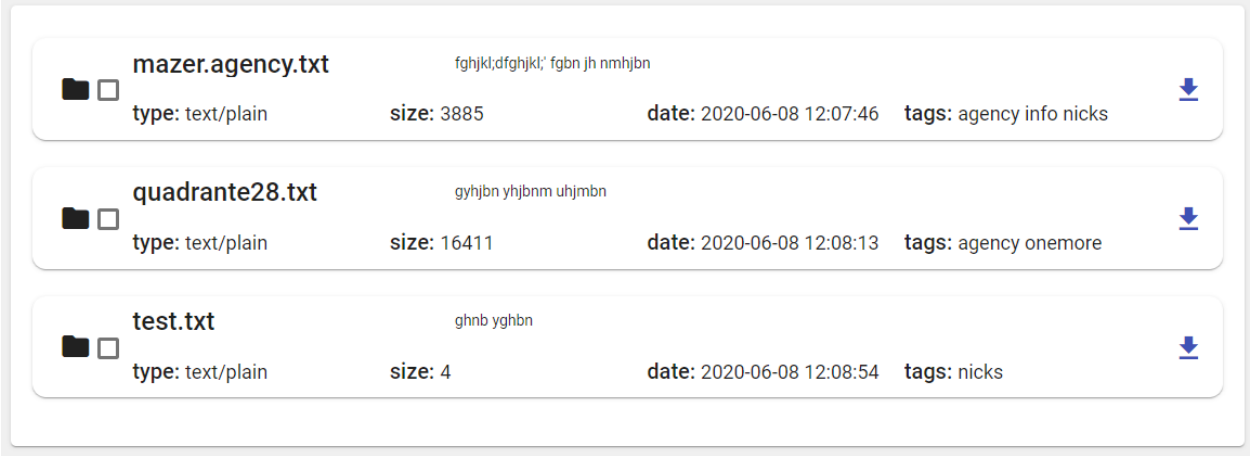


Рисунок 5.9 — Список файлів

У користувача є функціонал фільтрації файлів за назвою, тегами, типом та датою завантаження. При виборі потрібних фільтрів, список файлів буде автоматично відфільтровано. Меню фільтрації на рисунку 5.10

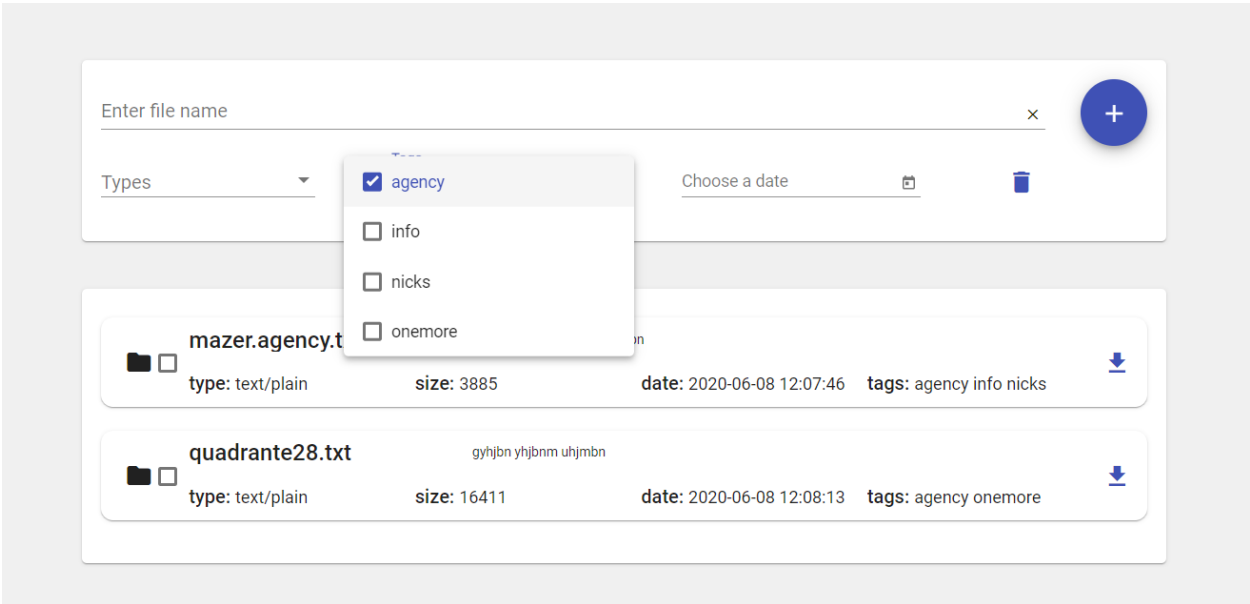


Рисунок 5.10 — Меню фільтрації файлів за тегами

Аналогічно, для пошуку за датою завантаження, тегами чи ім'ям файлу користувачеві необхідно ввести ключові слова або значення у поля форми на головному екрані (рисунок 5.7), і список автоматично фільтрується залежно від параметрів пошуку, введених користувачем (рисунок 5.11).

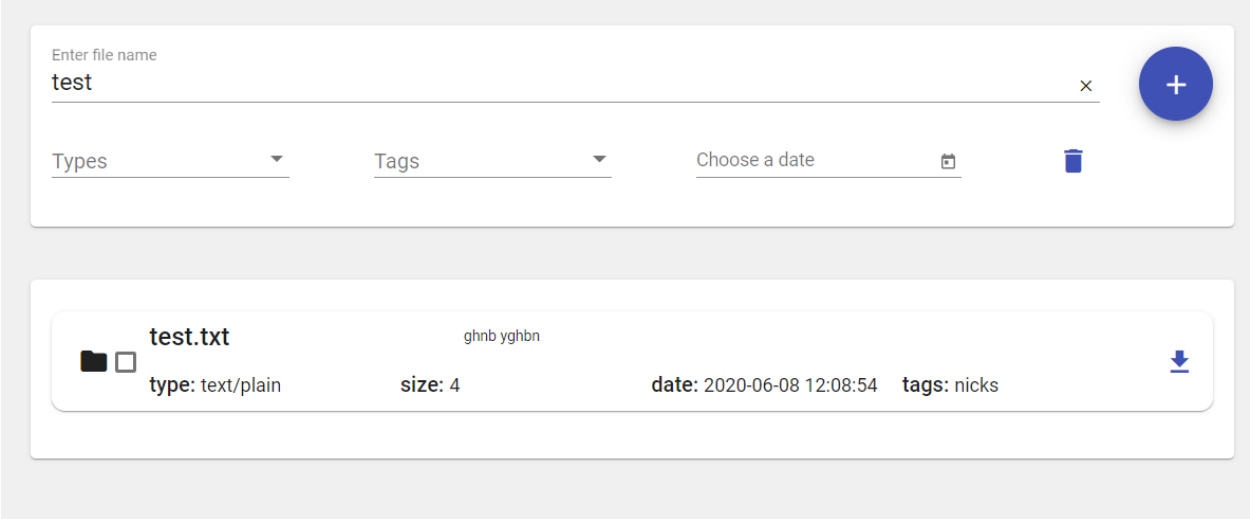


Рисунок 5.11 — Відфільтрований список файлів за назвою файла

Для видалення файлів із системи потрібно вибрати у полі з чекбоксом файли, а потім натиснути на кнопку з іконкою смітник рисунок 5.12. Файли будуть автоматично видалені із системи.

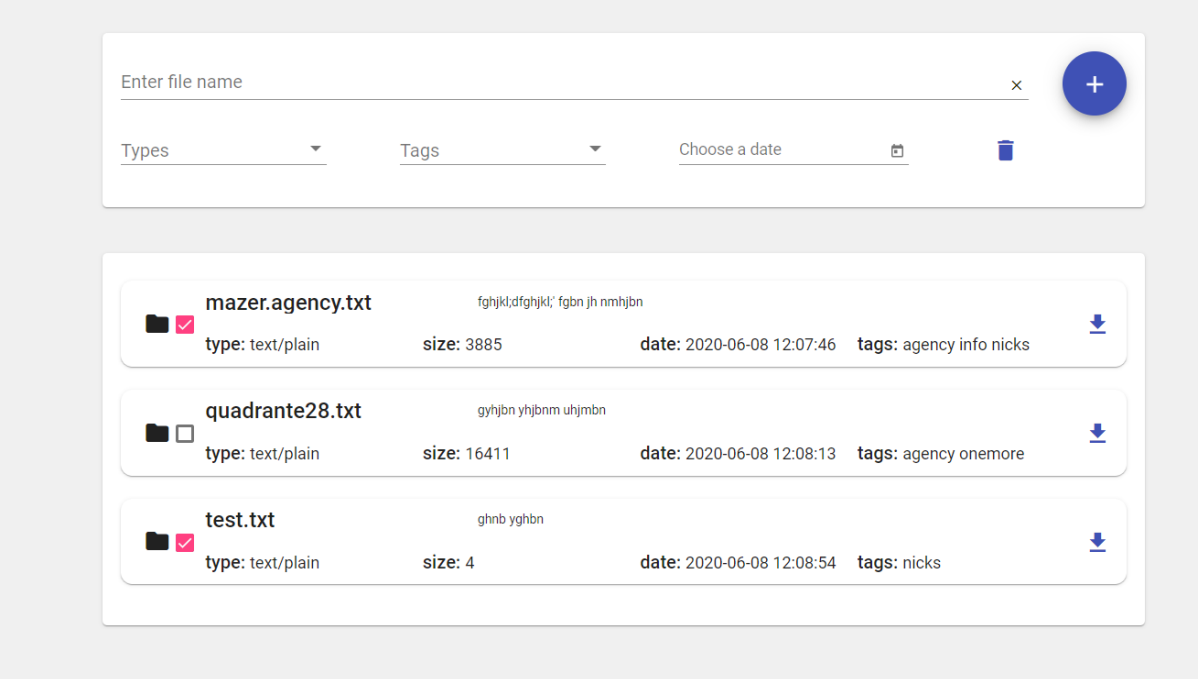


Рисунок 5.12 — Відмічені файли для видалення з системи

Таким чином, було створено веб-застосування, якн дозволяє користувачеві завантажувати, шукати, зберігати та видаляти інформаційні ресурси із бази даних. Перевага цього програмного забезпечення полягає в тому, що користувач має єдине універсальне сховище, для роботи з яким йому не потрібні додаткові знання про розподіл даних залежно від їх формату, все що від нього потрібно – це ввести адресу ресурсу та виконати потрібні дії.

ВИСНОВКИ

В процесі виконання роботи було виконано наступне.

1. Було проаналізовано існуючі програмні системи для збереження неструктурованих даних, виділено їх переваги і недоліки і сформульовано завдання, які має реалізовувати програмне забезпечення.

2. Було обрано засоби реалізації програмної системи – мультимодельна СУБД Caché для збереження документів та мета-даних, RESTful підхід до клієнт-серверної архітектури, Angular та TypeScript для реалізації клієнтського застосування.

3. Було розроблено архітектуру та структуру клієнт-серверного застосування, яка дозволяє завантажувати та зберігати різні види документів.

4. Було розроблено програмне забезпечення, яке дозволяє завантажувати документи та їх мета-дані в БД, шукати та фільтрувати документи на основі їх мета-даних, видаляти документи з БД та зберігати документи на комп’ютер користувача.

Програмне забезпечення може бути впроваджено в компаніях з великим оборотом документів, що надають інформаційні послуги та в наукових лабораторіях, що мають зберігати та опрацьовувати велику кількість інформації.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Михайлова І. Ю. Веб-застосування для завантаження та зберігання неструктурованих даних / І. Ю. Михайлова, А. І. Тищенко // ІНФОРМАТИКА, МАТЕМАТИКА, АВТОМАТИКА ІМА :: 2020 : мат. міжнар. наук.-техн. конф., 20-24 квітня 2020 р. / М-во освіти і науки України, Сумський державний університет. – Суми : Сумський державний університет, 2020. – С. 152.

2. Tyschenko A. I. Web registry of electronic information resources / A. I. Tyschenko, I. Y. Mykhailova // Сучасні аспекти розробки програмного забезпечення : збір. стат. VII наук.-прак. дист. конф., 01 червня 2020 р. / М-во освіти і науки України, КПІ ім. Ігоря Сікорського. – К. : КПІ ім. Ігоря Сікорського , 2020. – С. 56-59.

3. Гайджаржи В. І. Об’єктно-реляційна СУБД Caché. Багатовимірний сервер даних і способи реалізації бізнес логіки засобами вбудованої мови Caché ObjectScript. Навч. посібн. / В. І. Гайджаржи, І. Ю. Михайлова. — К.: Освіта України, 2015. — 312 с.

4. Про схвалення Стратегії розвитку інформаційного суспільства в Україні від 15 травня 2013 р. № 386-р [Electronic resource]. - Access mode: https://zakon.rada.gov.ua/laws/show/386-2013-%D1%80/ed20130515#n21

5. The Multi-Model Database: Enabling Business Transformational Applications [Electronic resource]. - Access mode: https://www.intersystems.com/resources/detail/multi-model-database-enabling-business-transformational-applications/

6. Что такое TypeScript: взаимодействие с JavaScript и Angular, описание interface и других функций [Електронний ресурс]. – Режим доступу: https://webformyself.com/chto-takoe-typescript-staticheskaya-tipizaciya-dlya-interneta/

7. Angular [Електронний ресурс]. – Режим доступу: https://angular.io/docs

8. Когаловский М. Р. Энциклопедия технологий баз данных / М. Р. Когаловский. — М.: Финансы и статистика, 2002. — 800 с.

9. Typescript [Electronic resource]. - Access mode: https://www.typescriptlang.org/

10. Кисленко Н.П. HTML. Самое необходимое. / Н.П. Кисленко — СПб.: БХВ-Петербург, 2008. — 352 с.

11. Пауэлл Т.А. Полное руководство по HTML / Т.А. Пауэлл. — М.: Мир, 2001. — 912 с.

12. Справочник CSS [Електронний ресурс] — Режим доступу: http://htmlbook.ru/css Дата доступу: 01.06.2017

13. Офіційний сайт Bootstrap 3 [Електронний ресурс] — Режим доступу: http://bootstrap-3.ru/index.php Дата доступу: 01.06.2017

14. The Multi-Model Database: Enabling Business Transformational Applications [Електронний ресурс]. - Режим доступу: https://www.intersystems.com/resources/detail/multi-model-database-enabling-business-transformational-applications/

Додаток 1

Веб-застосування для завантаження та зберігання неструктурованих даних

Специфікація

УКР.НТУУ “КПІ”.ТР32269_17Б

Аркушів 2

2020

Позначення	Найменування	Примітки
Документація		
УКР.НТУУ“КПІ”. ТР32269_17Б 81-1	Записка.docx	Пояснювальна записка
Компоненти		
УКР.НТУУ“КПІ”. ТР32269_17Б 12-1	Arhive.xml	Текст програмного модуля маніпулювання документами
УКР.НТУУ“КПІ”. ТР32269_17Б 13-1	Arhive.docx	Опис програмного модуля маніпулювання документами
УКР.НТУУ“КПІ”. ТР32269_17Б 12-2	package.json	Перелік залежностей проекту
УКР.НТУУ“КПІ”. ТР32269_17Б 12-3	tsconfig.json	Налаштування збірки проекту
УКР.НТУУ“КПІ”. ТР32269_17Б 12-5	app.component.ts	Логіка роботи головної сторінки
УКР.НТУУ“КПІ”. ТР32269_17Б 12-6	app.component.html	Текст головної сторінки
УКР.НТУУ“КПІ”. ТР32269_17Б 12-7	files-list.component.ts	Логіка відображення та взаємодії зі списком файлів
УКР.НТУУ“КПІ”. ТР32269_17Б 12-8	files-list.component.html	Текст модуля зі списком файлів
УКР.НТУУ“КПІ”. ТР32269_17Б 12-9	files-list.component.css	Текст зовнішнього вигляду списку файлів
УКР.НТУУ“КПІ”. ТР32269_17Б 12-10	upload-file-pop-up.component.ts	Логіка роботи вікна для завантаження файлів

Додаток 2

Веб-застосування для завантаження та зберігання неструктурованих даних

Текст програмного модуля

УКР.НТУУ “КПІ”.ТР32269_17Б 12-1

Аркушів 11

2020

- 2 -

Back-end

«Archive.FileDesc.cls»:

```
Class Archive.FileDesc Extends %Persistent
{

Property File As %Stream.GlobalBinary;

Property FileName As %String [ Required ];

Property Type As %String [ Required ];

Property UploadDate As %TimeStamp [ Required ];

Property FileSize As %Float [ Required ];

Property Descrition As %String;

Property Tags As list Of %String;

Storage Default
{
<Data name="FileDescDefaultData">
<Value name="1">
<Value>%%CLASSNAME</Value>
</Value>
<Value name="2">
<Value>File</Value>
</Value>
```

```
<Value name="3">
<Value>FileName</Value>
</Value>
<Value name="4">
<Value>Type</Value>
</Value>
<Value name="5">
<Value>UploadDate</Value>
</Value>
<Value name="6">
<Value>FileSize</Value>
</Value>
<Value name="7">
<Value>Descrition</Value>
</Value>
<Value name="8">
<Value>Tags</Value>
</Value>
</Data>
<DataLocation>^Archive.FileDescD</DataLocation>
<DefaultData>FileDescDefaultData</DefaultData>
<IdLocation>^Archive.FileDescD</IdLocation>
<IndexLocation>^Archive.FileDescI</IndexLocation>
<StreamLocation>^Archive.FileDescS</StreamLocation>
<Type>%Library.CacheStorage</Type>
}
}
```

«Archive.Broker.cls»:

```
Class Archive.Broker Extends %CSP.REST
{

XData UrlMap
{
<Routes>
        <Route Url="/files" Method="GET" Call="GetAllFiles"/>
        <Route Url="/file/:id/contents" Method="GET"
Call="GetFileContents"/>
        <Route Url="/file/:id" Method="GET" Call="GetFileInfo"/>

        <Route Url="/file/:id" Method="DELETE"
Call="DeleteFile"/>
        <Route Url="/files/:ids" Method="DELETE"
Call="DeleteFiles"/>

        <Route Url="/file" Method="POST" Call="InsertFile"/>
        <Route Url="/file/:id" Method="POST"
Call="InsertFileContents"/>
</Routes>
}

ClassMethod DeleteFile(id As %String) As %Status
{
  Set result={}
  Set sc=0
```

```
 if id'="", ##class(Archive.FileDesc).%ExistsId(id) {
   Set sc=##class(Archive.FileDesc).%DeleteId(id)
   do
result.%Set("Status",$s($$$ISERR(sc):$system.Status.GetOneErrorText(sc),
1:"OK"))
  }
  else  {
   do result.%Set("Status","File with ID = ["_id_"] does not exist!")
  }

  write result.%ToJSON()
  quit sc
 }

 ClassMethod DeleteFiles(ids As %String) As %Status
 {
  Set result={}
  Set sc=0
  Set message = ""

  if ids'="" {
   set list = $listfromstring(ids,",")
   if $listvalid(list) {
        SET ptr=0
   WHILE $LISTNEXT(list, ptr, id) {
       if ##class(Archive.FileDesc).%ExistsId(id) {
                Set sc=##class(Archive.FileDesc).%DeleteId(id)
```

```
        set message = message_"Status for file with ID = ["_id_"]:
"_$s($$$ISERR(sc):$system.Status.GetOneErrorText(sc),1:"OK")_$char(10
)_$char(13)
                        }
                        else  {
                        do result.%Set("Status","File with ID = ["_id_"] does not
exist!")
                        }
            }
            do result.%Set("Message",message)
            Set sc=$$$OK
            } else {
                  do result.%Set("Status","You haven't specified IDs with ,
delimiter")
            }
           }
           else {
            do result.%Set("Status","You haven't specified IDs")
           }

           write result.%ToJSON()
           quit sc
         }

         ClassMethod InsertFile() As %Status
         {
          Set result={}
```

```
        do
##class(%ZEN.Auxiliary.jsonProvider).%ConvertJSONToObject(%request.
Content, "Archive.FileDesc", .f)
         Set sc = f.%Save()

         if ($$$ISERR(sc)) {
            do result.%Set("Message",$system.Status.GetOneErrorText(sc))

            Set sc=$$$OK
         } else {
          do result.%Set("ID",f.%Id())
         }
         write result.%ToJSON()
         Quit sc
        }

        ClassMethod InsertFileContents(id As %String) As %Status
        {
         Set result={}
         Set st=0

         #dim tJsonObj As %CSP.BinaryStream


         if id'="", ##class(Archive.FileDesc).%ExistsId(id) {
           set f = ##class(Archive.FileDesc).%OpenId(id)

           Set st=%request.Content.Rewind()
```

```
        If $$$ISOK(st) {
            Set tJsonObj=%request.Content
          set st = f.File.CopyFrom(tJsonObj)

            If $$$ISOK(st) {
              set st = f.%Save()
              If $$$ISOK(st) {
                do result.%Set("Message","OK")
              } else {
                 do result.%Set("Message
3",$system.Status.GetOneErrorText(st))
                  }
            } else {
              do result.%Set("Message
3",$system.Status.GetOneErrorText(st))
            }
          } else {
            do result.%Set("Message
4",$system.Status.GetOneErrorText(st))
          }
        }
        else  {
          do result.%Set("Status","File with ID = ["_id_"] does not exist!")
        }

        write result.%ToJSON()
        Quit st
      }
```

```
ClassMethod GetAllFiles() As %Status
{
   Set result={}
   set st = 0
   try {
         set sql = "SELECT ID, FileName, UploadDate, FileSize,
Descrition, Tags, Type FROM Archive.FileDesc ORDER BY UploadDate"
         do
##class(%ZEN.Auxiliary.jsonSQLProvider).%WriteJSONFromSQL(,sql)
         set st = $$$OK
   }
   catch (ex){
         set st = ex.AsStatus()
         do result.%Set("Status","")
         write result.%ToJSON()
   }
   quit st
}

ClassMethod GetFileContents(id As %String) As %Status
{
  Set result={}
  Set st=0

  if id'="", ##class(Archive.FileDesc).%ExistsId(id) {
    set f = ##class(Archive.FileDesc).%OpenId(id)
    set %response.ContentType = "application/octet-stream"
```

```
        set %response.ContentLength = f.File.Size


          /*set %response.CharSet = "ANSI"
          set %response.NoCharSetConvert = 1*/

          set fs = f.File.Size
          set ^a(id, 1)=fs
          set st = f.File.OutputToDevice(.fs)
          set ^a(id, 2)=fs
          If $$$ISERR(st) {
             do result.%Set("Message
3",$system.Status.GetOneErrorText(st))
             write result.%ToJSON()
           }
         }
         else  {
          do result.%Set("Status","File with ID = ["_id_"] does not exist!")
          write result.%ToJSON()
         }

         quit st
        }

        ClassMethod GetFileInfo(id As %String) As %Status
        {
         Set result={}
         Set st=0
```

```
        if id'="", ##class(Archive.FileDesc).%ExistsId(id) {
          set f = ##class(Archive.FileDesc).%OpenId(id)
          set st = ##class(%ZEN.Auxiliary.jsonProvider).%ObjectToJSON(f)
        }
        else  {
          do result.%Set("Status","File with ID = ["_id_"] does not exist!")
          write result.%ToJSON()
        }
        quit st
      }

}
```

Додаток З

Веб-застосування для завантаження та зберігання неструктурованих даних

Опис програмного модуля

УКР.НТУУ “КПІ”.ТР32269_17Б 13-1

Аркушів 6

2020

АНОТАЦІЯ

Розроблене програмне забезпечення призначене для завантаження та зберігання неструктурованих даних.

Результатом виконання є клієнт-серверний застосунок, що реалізує зберігання та завантаження даних до системи. Перевагою даної системи є те, що користувач отримає доступ до єдиного універсального сховища, для роботи з яким йому потрібно ввести адресу WEB-ресурсу та завантажити або знайти вже існуючі файли в базі даних.

Мова програмної реалізації — Caché ObjectScript, база даних Cache та фреймворк Angular. Основне середовище розробки — WebStorm та Studio Cache.

ЗМІСТ

1. Загальні відомості……4
2. Технічні засоби……5
3. Вхідні та вихідні данні……6

- 4 -

1. ЗАГАЛЬНІ ВІДОМОСТІ

Програмна система є частиною програмного забезпечення для зберігання та завантаження неструктурованих даних. Після інсталяції системи на сервер згаданий модуль буде забезпечувати роботу функцій, які в ньому описані.

Для роботи модуля потрібно, мати встановлену СКБД Caché із розгорнутими в ній всіма іншими модулями системи.

- 5 -

2. ТЕХНІЧНІ ЗАСОБИ

Найменування програмного модуля — “Archive”. Мова програмної реалізації — Caché ObjectScript. Середовище розробки — Studio Cache.

Даний модуль являє собою серверну частину клієнт-серверного застосунку, який необхідний для роботи з базою даних.

Для використання програмного забезпечення необхідно мати:

— комп’ютер з довільною ОС;

— база даних Cache.

- 6 -

3. ВХІДНІ ТА ВИХІДНІ ДАНІ

Вхідними даними для роботи програмної системи є дані про документи, котрі були завантажені користувачами (рисунок 3.1).

Вихідними даними є:

— ім’я файлу;

— тип файлу;

— розмір файлу;

— теги файлу;

— дата завантаження файлу;

— опис файлу.

Вихідними даними результату роботи є статуси виконання запитів до бази даних.

Додаток 4

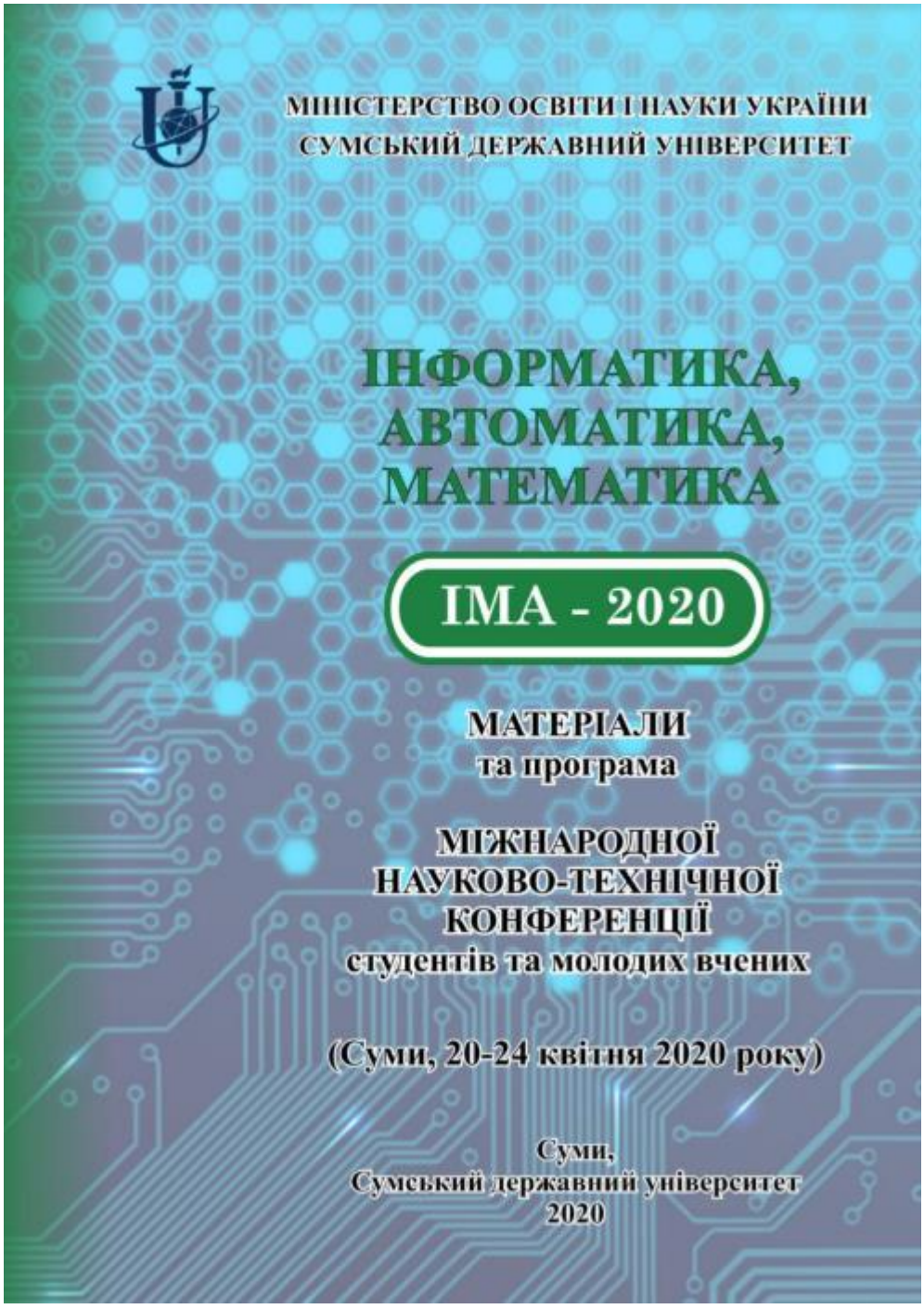
Веб-застосування для завантаження та зберігання неструктурованих даних

Апробації

УКР.НТУУ “КПІ”.ТР32269_17Б

Аркушів 5

2020


МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАТИКА, АВТОМАТИКА, МАТЕМАТИКА
ІМА - 2020
МАТЕРІАЛИ
та програма
МІЖНАРОДНОЇ НАУКОВО-ТЕХНІЧНОЇ КОНФЕРЕНЦІЇ
студентів та молодих вчених
(Суми, 20-24 квітня 2020 року)
Суми,
Сумський державний університет
2020

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

ІНФОРМАТИКА, МАТЕМАТИКА, АВТОМАТИКА

ІМА :: 2020

МАТЕРІАЛИ
та програма

МІЖНАРОДНОЇ
НАУКОВО-ТЕХНІЧНОЇ КОНФЕРЕНЦІЇ
студентів та молодих вчених

(Суми, 20–24 квітня 2020 року)

Суми
Сумський державний університет
2020

Автори: доц. Михайлова І.Ю.,
студ. **Тищенко А.І.**

52. Віртуальна модель корпусу «Н» Сумського державного університету

Автори: асп. **Кухарчук О.Р.**,
ст. викл. Нагорний В.В.

53. Web-додаток підтримки продажу аксесуарів для мобільних пристроїв

Автори: асп. **Противень Д.А.**,
ст. викл. Нагорний В.В.

54. Алгоритм моделювання знаків музичної нотації в поліграфічно-орієнтованому редакторі

Автори: студ. **Слющинський В.Я.**,
студ. **Сабуров О.В.**

55. Система забезпечення документообігу для дипломного проектування із використання хмарних сервісів

Автори: студ. **Александренко Т.В.**,
доц. Парфененко Ю.В.

56. Аналіз інтелектуальних методів обробки природної мови

Автори: студ. **Вербицька А.А.**,
асп. Бичко Д.В.,
доц. Парфененко Ю.В.

57. Мобільний додаток-довідник для секції інформаційних технологій проектування Сумського державного університету

Автори: студ. **Карпенко Д.В.**,
доц. Парфененко Ю.В.

Веб-застосування для завантаження та зберігання неструктурованих даних

Михайлова І.Ю., *доцент*; Тищенко А.І., *студентка*
НТУУ «КПІ» ім. Ігоря Сікорського, м. Київ, Україна

У зв'язку зі швидким збільшенням обсягу неструктурованих даних виникає необхідність їх обробки та оптимального збереження. Популярні реляційні системи управління базами даних (СУБД) мають певні обмеження, які не дозволяють швидко і зручно опрацьовувати такого роду дані. Таким чином, актуальною є проблема вибору СУБД, яка дозволить працювати з неструктурованими даними, та написання програмної системи, за допомогою якої можна зберігати файли різного типу у базі даних (БД) та знаходити необхідну інформацію, базуючись на певній ключовій інформації про збережені файли.

Для вирішення зазначеної проблеми та створення зазначеної програмної системи було запропоновано використати об'єктно-орієнтовану технологію мультимодельної СУБД InterSystems Caché. Для написання бізнес-логіки клієнтської сторони вирішено використовувати мову TypeScript, для відображення інтерфейсу користувача – Angular 9.

Програмна система дозволяє завантажувати файли будь-якого типу в БД, визначати мета-інформацію завантажених файлів, зберігати їх в зашифрованому вигляді в БД та виконувати пошук файлів, що зберігаються в БД, виходячи з мета-інформації, наприклад, імені файлу, його типу, дати створення тощо, а також переглядати історію доступу до обраного файлу.

Перевагою даної системи є те, що користувач отримує доступ до єдиного універсального сховища, для роботи з яким йому не потрібно мати ніяких додаткових знань щодо розподілу даних в залежності від їх формату, їх місця знаходження тощо. Все, що від нього вимагається, – це ввести адресу WEB-ресурсу та завантажити його разом із метаданими або знайти вже існуючі документи в системі на основі введених критеріїв пошуку.

Завдяки даному застосуванню користувач при роботі з даними буде володіти широким набором інструментів для завантаження, зберігання, фільтрації та сортування даних разом зі зручним веб-інтерфейсом.