МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інженерії програмного забезпечення

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри ІПЗ, канд. техн. наук, доцент _____Є. О. Давиденко

«___»_____2022 р.

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

РОЗРОБКА ХМАРНОГО СХОВИЩА

Спеціальність «Інженерія програмного забезпечення»

121 – КРБ.1 – 409. 21810917

Студент _____А. І. Кузьмін

«___»_____2022 р.

Керівник завідувач кафедри ІПЗ, канд. техн. наук, доцент _____Є. О. Давиденко

«___»_____2022 р.

Консультант канд. техн. наук, доцент _____А. О. Алексєєва

«___»_____2022 р.

Миколаїв 2022

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет імені Петра Могили
Факультет комп'ютерних наук
Кафедра інженерії програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ІПЗ,
канд. техн. наук, доцент
_____ Є. О. Давиденко
« __ » _____ 2022 р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи бакалавра

Видано студенту групи 409 факультету комп'ютерних наук
Кузьміну Артему Ігоровичу .
(прізвище, ім'я, по батькові студента)

1. Тема кваліфікаційної роботи:

Розробка хмарного сховища.

Затверджена наказом по ЧНУ ім. П.Могили від «01» грудня 2021 р. № 314

2. Строк представлення кваліфікаційної роботи: « __ » _____ 202__ р.

3. Очікуваний результат роботи та початкові дані, якщо такі потрібні

Вебзастосунок хмарного сховища, який буде надавати можливість створити власний обліковий запис в системі. Також в системі буде можливість взаємодії з файловою системою: завантаження файлів на сервер, створення і видалення директорій, скачування файлів із сервера. Додатковими можливостями буде: сортування файлів та директорій, пошук файлів, Відстеження прогресу завантаження файлів, перегляд основних властивостей файлу, навігація по файловій системі.

4. Перелік питань, що підлягають розробці:

- дослідження предметної області та існуючих аналогів;
- формування специфікації вимог до програмного забезпечення;
- визначення архітектури для проєктування програмного забезпечення;
- моделювання та проєктування програмного забезпечення;
- розробка програмного забезпечення;
- здійснення тестування модулю;
- проведення аналізу результатів розробки.

5. Перелік графічних матеріалів:

Презентація.

6. Завдання до спеціальної частини

Мікроклімат у приміщеннях з використанням комп'ютерної техніки

7. Консультанти:

Консультант	Кафедра (організація)	Частина роботи
Алєксєєва А.О.	Кафедра екології	Спеціальна частина з охорони праці

Керівник роботи канд. техн. наук, доцент Давиденко Євген Олександрович
(посада, прізвище, ім'я, по батькові)

_____
(підпис)

Завдання прийнято до виконання
Кузьмін Артем Ігорович
(прізвище, ім'я, по батькові студента)

_____
(підпис)

Дата видачі завдання « ___ » _____ 202__ р.

КАЛЕНДАРНИЙ ПЛАН
виконання кваліфікаційної роботи

Тема кваліфікаційної роботи:

Розробка хмарного сховища

№	Найменування роботи	Початок	Закінчення	Примітки
1.	Розробка та затвердження завдання на виконання КРБ	03.10.2021р.	04.10.2021р	виконано
2.	Огляд літератури за темою роботи	06.10.2021р.	08.10.2021р.	виконано
3.	Складання календарного плану КРБ	09.10.2021р.	10.10.2021р.	виконано
4.	Аналіз предметної області	15.10.2021р.	15.10.2021р.	виконано
5.	Розробка проєктних рішень	03.11.2021р.	04.11.2021р.	виконано
6.	Моделювання та конструювання ПЗ	05.11.2021р.	10.11.2021р.	виконано
7.	Кодування, тестування розробленого ПЗ, аналіз результатів тестування, розробка керівництва користувача	16.11.2021р.	22.02.2022р.	виконано
8.	Розробка спеціальної частини з охорони праці	14.05.2022р.	28.05.2022р.	виконано
9.	Оформлення КРБ та презентації	28.05.2022р.	13.06.2022р.	виконано
10.	Попередній захист	13.06.2022р.	15.06.2022р.	виконано
11.	Відгук керівника КРБ	19.06.2022р.	19.06.2022р.	виконано
12.	Завершення оформлення КРБ та презентації	15.06.2022р.	21.06.2022р.	виконано
13.	Рецензування	21.06.2022р.	21.06.2022р	виконано
14.	Захист кваліфікаційної роботи	27.06.2022р.	30.06.2022р.	виконано

Розробив студент Кузьмін Артем Ігорович _____
(прізвище, ім'я, по батькові) *(підпис)*

« ___ » _____ 202__ р.

Керівник роботи Завідувач кафедри ІПЗ, канд. техн. наук, доцент Давиденко Є.О. _____
(посада, прізвище, ім'я, по батькові) *(підпис)*

« ___ » _____ 202__ р.

АНОТАЦІЯ

до кваліфікаційної роботи бакалавра

«Розробка хмарного сховища»

Студент 409 гр. Кузьмін Артем Ігорович

Керівник: канд. техн. наук, доцент Давиденко Є. О.

Кваліфікаційна робота присвячена розробці хмарного сховища, що базується на односторінковому вебзастосунку, який в свою чергу надає користувачу можливість взаємодіяти із вільним простором пам'яті в середовищі вебзастосунку.

Об'єкт роботи: система хмарного сховища даних.

Предмет роботи: метод передачі інформації в середовищі хмарного сховища.

Метою проєкту є удосконалення системи хмарного сховища за рахунок розробки вебзастосунку хмарного сховища.

Кваліфікаційна робота бакалавра складається з вступу, чотирьох розділів, висновків та додатків.

У вступі визначається актуальність теми, мета та завдання, об’єкт дослідження, предмет дослідження, питання удосконалення існуючого об’єкта проєктування, сфера застосування.

У першому розділі описується аналітична частина проєкту: огляд існуючих аналогів проєкту, основні функції проєкту, переваги та недоліки проєкту, створюються сценарії роботи. Далі виконується обґрунтовування плану виконання завдання, а також створюється специфікація вимог до програмного забезпечення системи хмарного сховища даних.

У другому розділі виконується моделювання об’єкту та предмету дослідження. Виконується формування функціональних та інформаційних моделей проєкту. Здійснюється огляд моделі системи, основних діаграм вебзастосунку, паттерну.

У третьому розділі будується архітектура проєкту. Виконується моделювання та проєктування програмного забезпечення в рамках

функціональних вимог проєкту. Визначається та обґрунтовується використання технологій та інтерфейсів програмного забезпечення.

У четвертому розділі виконується кодування, тестування (перевірка) програмного забезпечення. Після цього проводяться обчислення та аналіз обчислення. Описується розробка керівництва користувача та аналіз результатів програмного забезпечення.

Кваліфікаційна робота бакалавра викладена на 62 сторінки, вона містить 4 розділи, 43 ілюстрацій, 3 таблиці, 22 джерел в переліку посилань.

Ключові слова: *розробка хмарного сховища, розробка вебзастосунку, удосконаленння системи, середовище схмарного сховища, база даних, операційна система, програме забезпечення.*

ABSTRACT

of the Bachelor's Thesis

"Development of cloud storage"

Student: Kuzmin Artem

Supervisor: Candidate of Technical Sciences (Ph. D.), Associate Professor

Davydenko Y. O.

The qualification work is dedicated to the development of cloud storage based on a one-page web application, which in turn allows the user to interact with free memory space in the web application environment.

Object of work: cloud data storage system.

Subject of work: method of information transfer in the cloud storage environment.

The aim of the project is to improve the cloud storage system by developing a cloud storage web application.

The bachelor's thesis consists of an introduction, four chapters, conclusions and appendices.

The introduction determines the relevance of the topic, purpose and objectives, the object of study, the subject of research, issues of improving the existing design object, scope.

The first section describes the analytical part of the project: an overview of existing project analogues, the main functions of the project, the advantages and disadvantages of the project, job scenarios are created. Then the substantiation of the task plan is performed, and also the specification of requirements to the software of the cloud data storage system is created.

The second section performs modeling of the object and subject of research. Formation of functional and information models of the project is carried out. An overview of the system model, basic diagrams of the web application, pattern.

The third section builds the architecture of the project. Software modeling and design is performed within the functional requirements of the project. The use of software technologies and interfaces is determined and substantiated.

The fourth section performs coding, testing (verification) of software. After that,

calculations and analysis of the calculation are performed. Describes the development of user manuals and analysis of software results.

The qualification work of the bachelor is set out on 62 pages, it contains 4 sections, 43 illustrations, 3 tables, 22 sources in the list of references.

Keywords: *cloud storage development, web application development, system improvement, cloud storage environment, database, operating system, software.*

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

БД – база даних
ОС – операційна система
ПЗ – програмне забезпечення
СХС – система хмарного сховища

API – Application programming interface
JWT – JSON Web Token
MVC – Model-View-Controller
SPA – Single-page application
UML – Unified Modeling Language

ВСТУП

Хмарне сховище є актуальним інструментом для зберігання даних на сервері. Цифрове середовище хмарного сховища зазвичай належить до хостингових компаній, що надають можливість зберігання даних. Користувачі часто використовують хмарне сховище в якості зберігання резервних копій даних та інформації, що потенційно може бути втрачена під час зберігання на локальних носіях. Доступність і захист даних є суттєвою для архітектури об’єктів, тому залежно від програми додаткові технології, зусилля та витрати на додавання доступності та захисту можуть бути усунені. Хмарне сховище також є зручним інструментом у випадку використання даних, що може знадобитися для використання кількох локальним комп’ютерам.

Об'єкт роботи: система хмарного сховища даних.

Предмет роботи: метод передачі інформації в середовищі хмарного сховища.

Метою проєкту є удосконалення системи хмарного сховища за рахунок розробки вебзастосунку хмарного сховища. Для досягнення визначеної мети необхідно вирішити такі завдання:

– проаналізувати систему, що розробляється та дослідити існуючи програмні рішення: користувачів системи, структуру системи, сценарії роботи системи, засоби апаратної та програмної реалізації;

– визначити архітектуру для проєктування програмного забезпечення;

– змоделювати програмне забезпечення: діаграма прецедентів, розгортання, ERD, діяльності, компонентів;

– розробити front-end частину вебзастосунку на базі технологій: React, Redux;

– розробити back-end частину вебзастосунку на базі технологій: Node.js, Express.js, MongoDB.

Необхідність розробки нового об’єкту проєктування зумовлено попитом на використання хмарних систем для збереження певних даних. Удосконалення існуючого об’єкта проєктування полягає у використанні одного із безпечних

способів передачі інформації між учасниками, а саме – JSON Web Token (JWT). Це відкритий стандарт (RFC 7519) для створення токенів доступу. Ще одним способом модернізації існуючого об’єкта є структура односторінкового застосунку, відомого як односторінковий інтерфейс – це вебзастосунок, що вміщує в себе одну сторінку з динамічно підгруженим HTML, CSS, JavaScript, зазвичай у відповідь при взаємодії користувача з вебзастосунком. Взаємодія вебзастосунка односторінкового типу зазвичай включає в себе динамічний зв’язок з вебсервером. Структура вебзастосунку також вміщує ідею взаємодії з основним функціоналом хмарного сховища в межах кількох кліків, а саме: створення папок, завантаження на сервер файлів, завантаження із серверу та видалення.

Сферою застосування проєкту може являтися загальний користувацький простір з метою часткового збереження даних за межами локальної системи.

1 АНАЛІТИЧНИЙ ОГЛЯД СИСТЕМИ ХМАРНОГО СХОВИЩА. СПЕЦИФІКАЦІЯ ВИМОГ

1.1 Існуючі аналоги обраної предметної області

Існує чимало методів зберігання даних і одним із таких є зберігання в системі хмарного сховища, замість того, щоб зберігати файли на локальному комп'ютері. Системи хмарного сховища надають певний сервіс для взаємодії клієнта з файлами. Такий сервіс може мати обмежений характер для безкоштовного використання та розширений у випадку індивідуальних потреб споживача. В залежності від придбаного плану споживач отримує доступ до базового функціоналу СХС. В цей функціонал може входити: завантаження файлів на сервер, завантаження із серверу файлів, створення директорій, видалення файлів та директорій з файлової системи, перегляд поточної інформації про завантажений файл. Додатковими або розширеними властивостями таких системи може бути: пошук, сортування файлів, навігація по файловій системі, налаштування способу відображення файлової системи, налаштування профілю користувача, розширення доступу до файлів, перегляд прогресу завантаження, завантаження файлів за допомогою перетягування у виділену область.

Існуючи програмні рішення предметної області

Аналог 1:

- **Назва:** Google disk.
- **Виробник:** Google Inc.
- **Архітектура:** client-server, desktop application.
- **Мова реалізації:** Python.
- **Основні функції:**
  1. Сканування документів.
  2. Перетягування файлів.
  3. Програма «Автозавантаження та синхронізація».
  4. Швидке створення файлів.
  5. Управління версіями.

6. Налаштування кольору папок.
7. Коментування папок і файлів.
8. Редагування документів Office.
9. Пошук по тексту, типу файлів та картинкам.
10. Можливість поділитися файлами і папками.
11. Автоматичне завантаження фото і відео.
12. Шифрування файлів.

– **Переваги:**

1. Безкоштовне зберігання певного обсягу даних.
2. Присутній мобільний та десктопний застосунок.
3. Доповнюючи застосунки для «Google disk».
4. Велика бібліотека шаблонів, яка постійно поповнюється.
5. Можливість організувати спільну роботу над документами.
6. Система контролю версіями.
7. Простий та зручний дизайн.

– **Недоліки:**

1. Відсутність перевірки граматики та стилістики тексту.
2. Друк тільки в PDF.
3. Немає можливості редагувати вбудовані стилі.
4. Обмежений об’єм вільного простору.

– **Джерело інформації:** https://uk.wikipedia.org/wiki/Google_Drive.

– **Призначення ПЗ:** управління даними в хмарі.

– **Ролі користувачів системи:** менеджер, менеджер контенту, співавтор, коментатор, читач.

– **Зовнішній вигляд інтерфейсу ПЗ (рис. 1.1):**

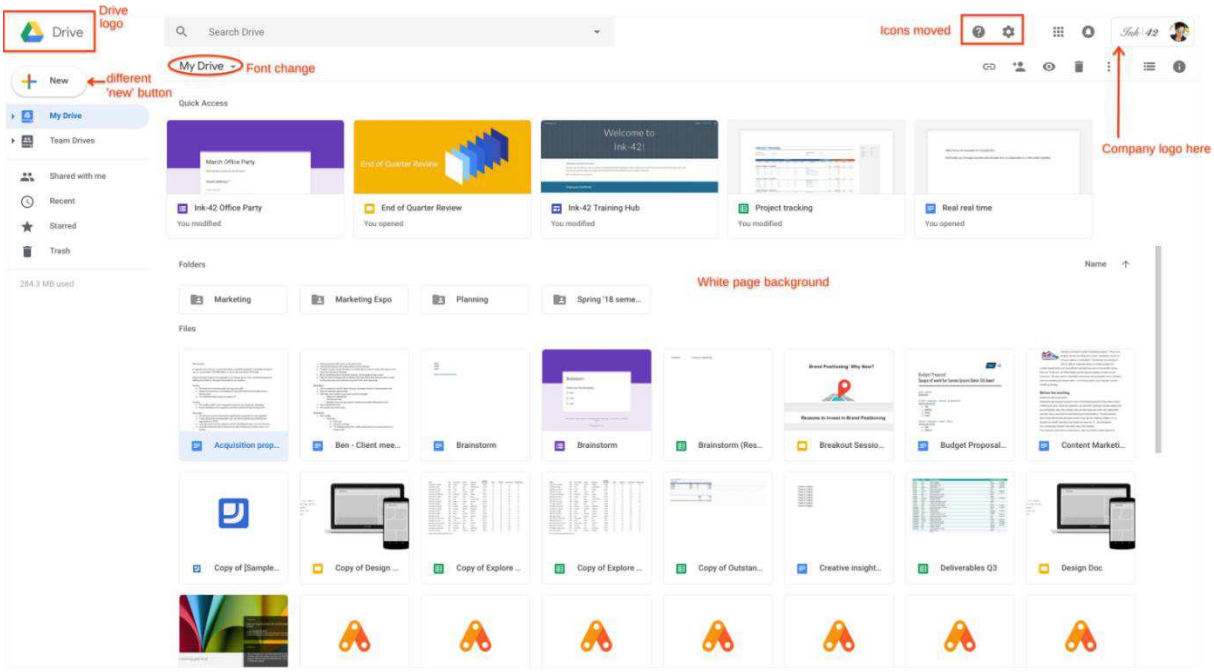

Рисунок 1.1 – Зовнішній вигляд ПЗ Google disk

Головна сторінка Google disk вміщує майже весь функціонал вебзастосунку та надає можливість виконувати взаємодію з системою в межах однією сторінки.

Аналог 2:

- **Назва:** Dropbox.
- **Виробник:** Dropbox Inc.
- **Архітектура:** client-server, desktop application.
- **Мова реалізації:** Python.
- **Основні функції:**
  1. Синхронізація папок між комп’ютерами та іншими пристроями для використання хмарного сховища.
  2. Копіювання, видалення, зміна назви папок і файлів.
  3. Можливість відкриття файлів в веббравзері, використовуючи вбудований функціонал Dropbox.
  4. Розширення доступу до файлів стороннім користувачам за допомогою посилання.
  5. Редагування документів починаючи від текстових і закінчуючи PDF.
- **Переваги:**

1. Можливість закріплювати файли, опис, посилання, списки та завдання до папок.
2. Можливість згадувати інших користувачів, створювати події та залишати коментарі.
3. Використання надійного способу шифрування (PFS, AES-256, SSL/TLS) для додатків, системи двухфакторної авторизації та строгої конфіденційності інформації.
4. Ведення історії завантажень, що в разі видалення файлів надає можливість відновити їх.
5. Безкоштовний мобільний додаток для інших популярних платформ ОС.

– **Недоліки:**
1. Не великий об’єм вільного дискового простору для безкоштовної версії.
2. Не значний рівень функціоналу застосунку.

– **Джерело інформації:** https://uk.wikipedia.org/wiki/Dropbox.
– **Призначення ПЗ:** управління даними в хмарі.
– **Ролі користувачів системи:** Власник, редактор, читач.
– **Зовнішній вигляд інтерфейсу ПЗ (рис. 1.2):**

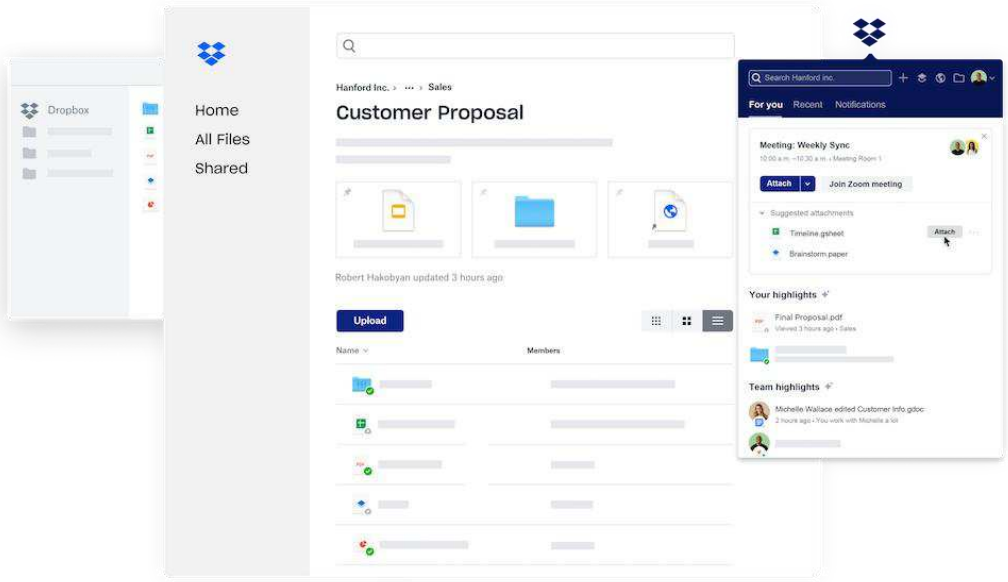

Рисунок 1.2 – Зовнішній вигляд ПЗ Dropbox

В Dropbox за замовчуванням файли виведені у вигляді списку, а інший функціонал розподілений в інших пунктах меню системи.

Аналог 3:

- **Назва:** iCloud.
- **Виробник:** Apple.
- **Архітектура:** client-server, desktop application.
- **Мова реалізації:** C і Objective-C.
- **Основні функції:**
  1. Зберігання та оновлення даних: зображення та відео, файлів, контактів, поміток, паролів та методів оплати, голосових поміток, карт, тем повідомлень.
  2. Розширювати доступ до світлини та відео.
  3. Використовувати локатор для пошуку втраченого пристрою або надавати дані свого місцезнаходження.
  4. Надавати доступ до файлів та папок в iCloud Drive.
  5. Тимчасове зберігання даних.
  6. Відновлення резервних копій через iCloud.
- **Переваги:**
  1. Надається безкоштовні 5 ГБ вільного місця в пам’яті.
  2. Є можливість створювати резервні копії всіх файлів з iPhone, iPad та Mac.
  3. Легкий та зручний дизайн.
  4. Цифрове успадкування хмарного сховища.
- **Недоліки:**
  1. Важко отримати доступ для користувачів інших платформ.
  2. Важкість у відновленні доступу до облікового запису.
- **Джерело інформації:** https://uk.wikipedia.org/wiki/ICloud.
- **Призначення ПЗ**: управління даними сховища.
- **Ролі користувачів системи**: власник, редактор, учасник.
- **Зовнішній вигляд інтерфейсу ПЗ (рис. 1.3):**

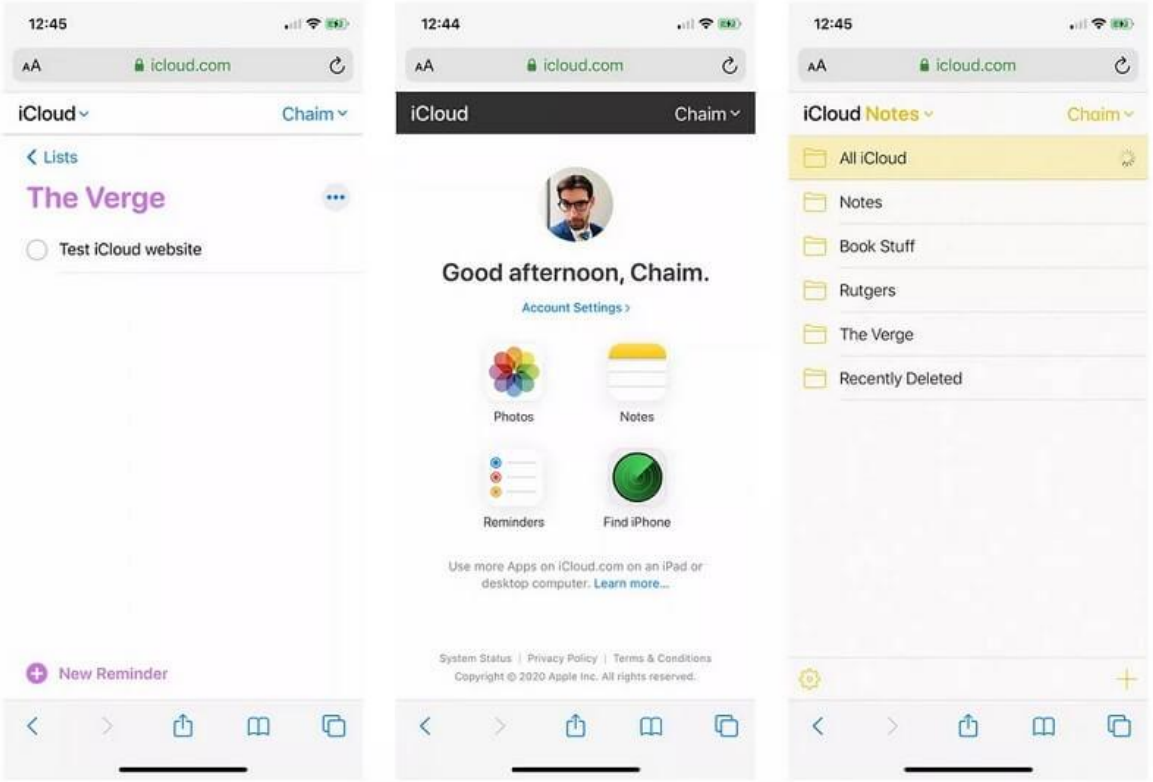

Рисунок 1.3 – Зовнішній вигляд ПЗ iCloud

В системі iCloud функціонал розподілений рівномірно для легкого його знаходження, так як ним користуються здебільшого з телефону.

1.2 Загальні відомості про створений вебзастосунок, виявлення переваг та недоліків

- **Назва:** MERN cloud storage.
- **Архітектура:** client-server.
- **Мова реалізації:** JavaScript.
- **Основні функції:**
    1. Реєстрація та авторизація в системі.
    2. Завантаження файлів на сервер.
    3. Створення папок.
    4. Навігація по файловій системі.
    5. Видалення файлів і директорій.
    6. Перегляд інформації про розмір, дату файлів та відстеження прогресу завантаження файлів.

7. Редагування даних профіля.
8. Завантаження файлів із серверу.
9. Сортування файлів та директорій.
10. Пошук файлів.

– **Переваги:**

1. Безкоштовне зберігання даних.
2. Швидкий і зручний інтерфейс.
3. Віддалений доступ до хмарного сховища.
4. Можливість завантаження широкого асортименту форматів файлу.
5. Безпечний режим передачі даних.

– **Недоліки:**

1. Відсутня можливість розширення доступу до файлів та спільного режиму.
2. Немає можливості редагувати файли всередині сховища.
3. Немає мобільного та десктопного додатка.

– **Призначення ПЗ:** управління даними сховища.

– **Роль користувачів системи:** власник.

– **Зовнішній вигляд інтерфейсу ПЗ (рис. 1.4):**

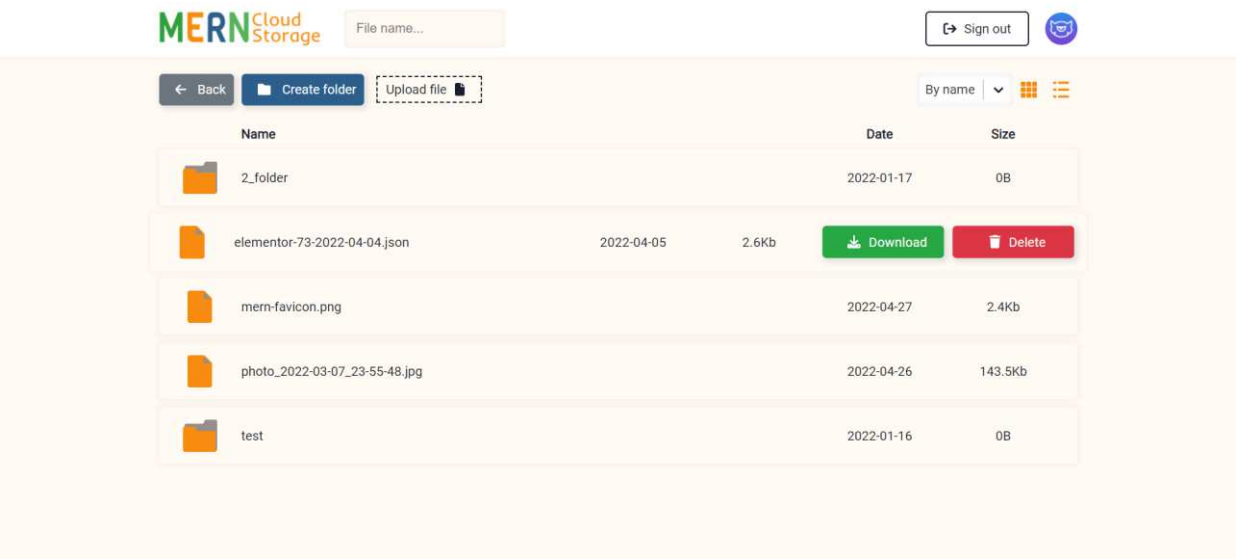

Рисунок 1.4 – Зовнішній вигляд ПЗ MERN cloud storage

Основний функціонал MERN cloud storage можна отримати в межах одного

або двох кліків мишею комп’ютера. Додатковий функціонал знаходить на сторінці профілю користувача.

1.3 Специфікації вимог до програмного забезпечення

ПРИЗНАЧЕННЯ ТА МЕЖІ ПРОЄКТУ

Призначення системи (застосунку), для якої розробляється програмне забезпечення

ПЗ призначено для можливості взаємодії з хмарним сховищем: завантаження файлів на сервер, створення і видалення папок та файлів, завантаження файлів із сервера.

Погодження, що ухвалені в програмній документації

Було погоджено, що для створення програмного забезпечення та його злагодженої роботи будуть використовуватися допоміжні бібліотеки React та Redux.

Межі проєкту ПЗ

Кінцева дата завершення ПЗ – 16.11.2021 р.

ЗАГАЛЬНИЙ ОПИС

Сфера застосування

Сферою застосування проєкту може являтися загальний користувацький простір з метою часткового збереження даних за межами локальної системи.

Характеристики користувачів

Наявність мобільного телефону, планшету, ПК чи іншого пристрою з доступом до Інтернету та сучасними версіями веббравзерів.

Загальна структура і склад системи

Структура ПЗ побудована на базі архітектури client-server та складається із: бази даних, серверу та самого вебзастосунку.

Загальні обмеження

Наявність стабільного підключення до мережі Інтернет та використання сучасних версій вебпереглядача.

ФУНКЦІЇ СИСТЕМИ

Функція завантаження файлу на сервер

Опис функції

Функція завантаження файлу надає можливість завантажити певний файл в хмарне сховище за допомогою перетягування файлу у виділену область або через стандарту форму пошуку файлу на локальному комп’ютері.

Вхідна і вихідна інформація

Вхідна інформація – обраний файл, вихідна інформація – відображення файлу в списку файлової системи.

Функціональні вимоги

Наявність файлу на локальному комп’ютері для завантаження на сервер та стабільне підключення до мережі Інтернет.

Функція завантаження файлу із сервера

Опис функції

Функція завантаження файлу із серверу надає можливість завантажити раніше завантажений із серверу файл на локальний комп’ютер.

Вхідна і вихідна інформація

Вхідна інформація – файл на сервері, вихідна – файл на локальному комп’ютері.

Функціональні вимоги

Наявність файлу на сервері для завантаження на локальний комп’ютер та стабільне підключення до мережі Інтернет.

Функція створення папки

Опис функції

Функція створення папки надає можливість встановити ім’я папці та додати її до файлової системи хмарного сховища.

Вхідна і вихідна інформація

Вхідна інформація – вказане ім’я файлу, вихідна – створена нова папка у файловій системі хмарного сховища.

Функціональні вимоги

Занести ім’я файлу в поле input та стабільне підключення до мережі Інтернет.

ВИМОГИ ДО ІНФОРМАЦІЙНОГО ЗАБЕЗПЕЧЕННЯ

Джерела і зміст вхідної інформації (даних)

Джерелом вхідної інформації в ПЗ є користувач. Користувач додає необхідні дані на сервер, тим самим зберігаючи в базі даних інформацію облікового запису та певних файлів на сервері.

Нормативно-довідкова інформація (класифікатори, довідники тощо)

Немає вимог нормативно-довідникової інформації.

Вимоги до способів організації, збереження та ведення інформації

Обмін даними між клієнтом та сервером повинен відбуватися за допомогою REST API та через JWT. В якості бази даних для програмного забезпечення обрати MongoDB.

ВИМОГИ ДО ТЕХНІЧНОГО ЗАБЕЗПЕЧЕННЯ

Будь-який пристрій з доступом до мережі Інтернет, який підтримує сучасні версії бравзерів.

ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Архітектура програмної системи

Архітектура програмної системи базується на SPA та складається з клієнтської частини, сервера та БД.

Системне програмне забезпечення

Для написання фронтенд частини вебзастосунку обрати бібліотеки React та Redux, для бекенду використовувати Node.js. Спосіб передачі даних за допомогою REST API та JWT, а в якості бази даних – MongoDB.

Мережне програмне забезпечення

Для побудови ПЗ потрібна будь-яка ОС, що підтримує сучасні веббравзери, Node.js та npm, а також редактор PhpStorm.

Програмне забезпечення ведення інформаційної бази

Ведення інформаційної бази має відбуватися через MongoDB.

Мова і технологія розробки ПЗ

ПЗ повинно бути розроблене на базі бібліотек React & Redux. Мова розробки ПЗ – JavaScript.

ВЛАСТИВОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Доступність

Будь-який користувач, що має доступ до мережі Інтернет має можливість зареєструватися та використовувати ПЗ.

Супроводжуваність

Супроводження – не передбачене у даному застосунку.

Переносимість

Дане ПЗ може працювати на будь-якій ОС та пристрою, що підтримує новітні браузери.

Продуктивність

Продуктивність вебзастосунка залежить від того, який браузер використовується та яка швидкість мережі Інтернет, тому що функція завантаження на сервер або завантаження файлів із серверу залежить напряму від швидкості мережі Інтернет. В програмному забезпеченні запити до БД асинхронні, а сама БД – MongoDB, тобто NoSQL, а даний вид БД має перевагу в швидкості перед SQL.

Надійність

Кожен користувач має доступ тільки до своїх файлів та не має можливості розширювати доступ до них, що покращує їх надійність.

Безпека

Користувачі системи мають свій токен для взаємодії із сервером, тобто без нього не можливо виконати запит до БД, а також він постійно оновлюється, що додає рівень безпеки у випадку отриманні секретного ключа.

ІНШІ ВИМОГИ

Усі вимоги описані вище.

Висновки до розділу 1

Отже, в першому розділі здійснено аналіз систем хмарного сховища, а саме:

Google disk, Dropbox та iCloud. При розгляді аналогів системи хмарного сховища було проаналізовано наступні характеристики: виробник, архітектура, мова реалізації, основні функції, переваги, недоліки, призначення ПЗ, ролі користувачів системи. Здійснено порівняльні характеристики зі створеним вебзастосунком. Виявлено загальні відомості про створений вебзастосунок, його переваги та недоліки. Також визначені специфікації вимог до ПЗ:

- призначення та межі проєкту;
- загальний опис;
- функції системи;
- вимоги до інформаційного забезпечення;
- вимоги до технічного забезпечення;
- вимоги до програмного забезпечення;
- вимоги до зовнішніх інтерфейсів;
- властивості програмного забезпечення.

2 МОДЕЛЮВАННЯ СИСТЕМИ ХМАРНОГО СХОВИЩА. ФУНКЦІОНАЛЬНІ ТА ІНФОРМАЦІЙНІ МОДЕЛІ СИСТЕМИ

2.1 Огляд моделі системи хмарного сховища

Модель системи хмарного сховища вміщує в собі ідею взаємодії з головним функціоналом вебзастосунка в межах кількох кліків (рис. 2.1). Не залежно від пристрою, що використовується для взаємодії з хмарним сховищем, весь функціонал розподілений рівномірно по всьому вебзастосунку. До функціоналу, що входить в межах кількох кліків входить:

– завантаження файлів на сервер;
– завантаження файлів із серверу;
– сортування файлів (папок), зміна режимів інтерфейсу;
– пошук файлів на сервері;
– створення папок;
– перехід до сторінки профілю;
– завантаження, видалення файлів (папок).

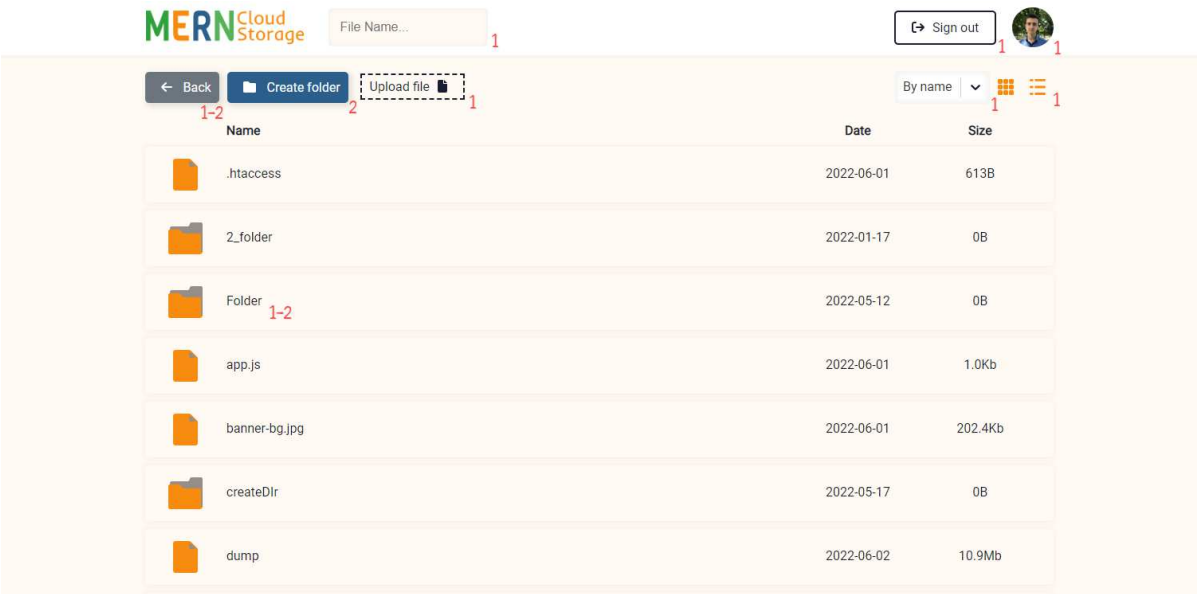

Рисунок 2.1 – Огляд головної сторінки моделі

При огляді моделі, можна легко простежити, що функціонал розподілений в межах одного або двох кліків. При чому в деяких випадках це може бути 1 або 2 кліки мишкою, а в деяких тільки 1. Це дозволяє легко та швидко ознайомитися із

системою та почати її використовувати без попереднього вивчення «user guide». Крім цього, скорочуючи дистанцію для взаємодії з функціоналом, користувач швидше виконує свій запит та не перевантажує ресурси сервера на виконання зайвих операцій.

2.2 Варіанти використання системи (usecases) хмарного сховища

Usecase – це опис низки сценаріїв. При взаємодії користувача з системою може завершитися досягненням певної мети чи навпаки призвести до певних негативних наслідків.

Сценарій в свою чергу характеризується послідовністю дій, що при взаємодії користувача з системою має певний результат.

Написання варіантів використання дає чітке визначення, хто є користувачем системи, яка мета використання системи та які сценарії входять до usecases.

Таблиця 2.1 – Авторизація до системи

Usecase section	Comment
Use Case Name	Авторизація до системи.
Scope	Хмарне сховище.
Level	Успішно авторизуватися до системи управління даними та отримати доступ до акаунту.
Primary Actor	Власник.
Stakeholders and interests	Власник – отримання доступу до власного акаунту для управління даними.
Preconditions	Власник має бути зареєстрований в системі.
Success guarantee	1. Власник має використовувати стабільне підключення до мережі Інтернет. 2. Власник має використовувати валідні дані для реєстрації в системі. 3. Власник має використовувати підтримку скриптів в браузері.

Кінець таблиці 2.1

Usecase section	**Comment**
Main Success Scenario	1. Власник переходить на сторінку авторизації. 2. Власник вводить дані свого профілю. 3. Система перенаправляє власника в особистий профіль.
Extensions	– Користувач допускає помилку при авторизації: 1. Користувач переходить до сторінки реєстрації та вводить існуючі дані. 2. Користувач не заповнюючи поля авторизації і натискає кнопку входу. – Користувач використовує автозаповнення полів та перенаправляється в особистий акаунт.
Special Requirements	Налаштування браузера повинна підтримувати виконання JavaScript коду.
Frequency of Occurrence	15%
Miscellaneous	Чи потрібно розлогінювати користувачів при неактивності впродовж певного часу.

Таблиця 2.2 – Завантаження файлів на сервер

Usecase section	**Comment**
Use Case Name	Завантаження файлів
Scope	Хмарне сховище
Level	Успішно авторизуватися та мати можливість завантажувати файли на сервер.
Primary Actor	Власник
Stakeholders and interests	Власник – завантаження файлів на сервер.

Продовження таблиці 2.2

Usecase section	**Comment**
Preconditions	Власник має бути авторизований
Success guarantee	1. Власник повинен мати файли, які можна завантажити. 2. Власник має використовувати стабільне підключення до мережі Інтернет. 3. Власник має завантажувати правильний тип файлів.
Main Success Scenario	1. Користувач відкриває файлову систему в хмарному сховищі. 2. Клієнт завантажує файл на сервер хмарного сховища. 3. Система завантажує певний об’єм даних на сервер та відображає їх в поточній файловій системі користувача.
Extensions	– Користувач генерує помилку при завантажені: 1. Користувач обирає тип файлу, який не може бути завантаженим до серверу. 2. Користувач перевищує ліміт дозволеного об’єму даних для завантаження на сервер. – Користувач не правильно використовує можливість перетягування файлів в область завантаження. 1. Користувач перетягує не вірний тип файлу. 2. Користувач не потрапляє в область завантаження файлів.
Special Requirements	Завантаження файлів на сервер повинні бути зі стабільним підключенням до Інтернету.

Кінець таблиці 2.2

Usecase section	Comment
Frequency of Occurrence	30%
Miscellaneous	Які необхідні параметри серверу для завантаження файлів.

Таблиця 2.3 – Створення директорій

Usecase section	Comment
Use Case Name	Створення директорій
Scope	Хмарне сховище
Level	Успішно авторизуватися та створити директорію в файловій системі хмарного сховища.
Primary Actor	Власник
Stakeholders and interests	Власник – створення директорій в файловій системі.
Preconditions	Власник має бути авторизований в системі хмарного сховища.
Success guarantee	1. Браузер повинен підтримувати JavaScript код. 2. Власник має використовувати стабільне підключення до мережі.
Main Success Scenario	1. Користувач створює директорію в корні проєкту. 2. Користувач переходить в певну директорію для створення вкладеної папки.
Extensions	– Користувач відхиляє процес створення директорії: 1. Користувач натискає кнопку відміни створення папки. – Користувач не вказує назву директорії і натискає кнопку створити.

Кінець таблиці 2.3

Usecase section	Comment
Special Requirements	Потрібно бути підключеним до мережі Інтернет.
Frequency of Occurrence	20%
Miscellaneous	Які можуть бути обмеження при створенні директорії.

Варіанти використання складаються із трьох таблиць (варіантів): авторизація до системи, завантаження файлів на сервер, створення директорій**.**

2.3 Робота вебзастосунку з використання JSON Web Token

JSON Web Token (JWT) – це JSON об’єкт, що визначений у відкритому стандарті RFC 7519 та є одним із безпечних способів передачі даних. В склад JWT токена входить: заголовок (header) із загальной інформацією токена, корисні дані (payload), такі як id користувача, його роль і т.д. і підписи (signature) (рис. 2.2).

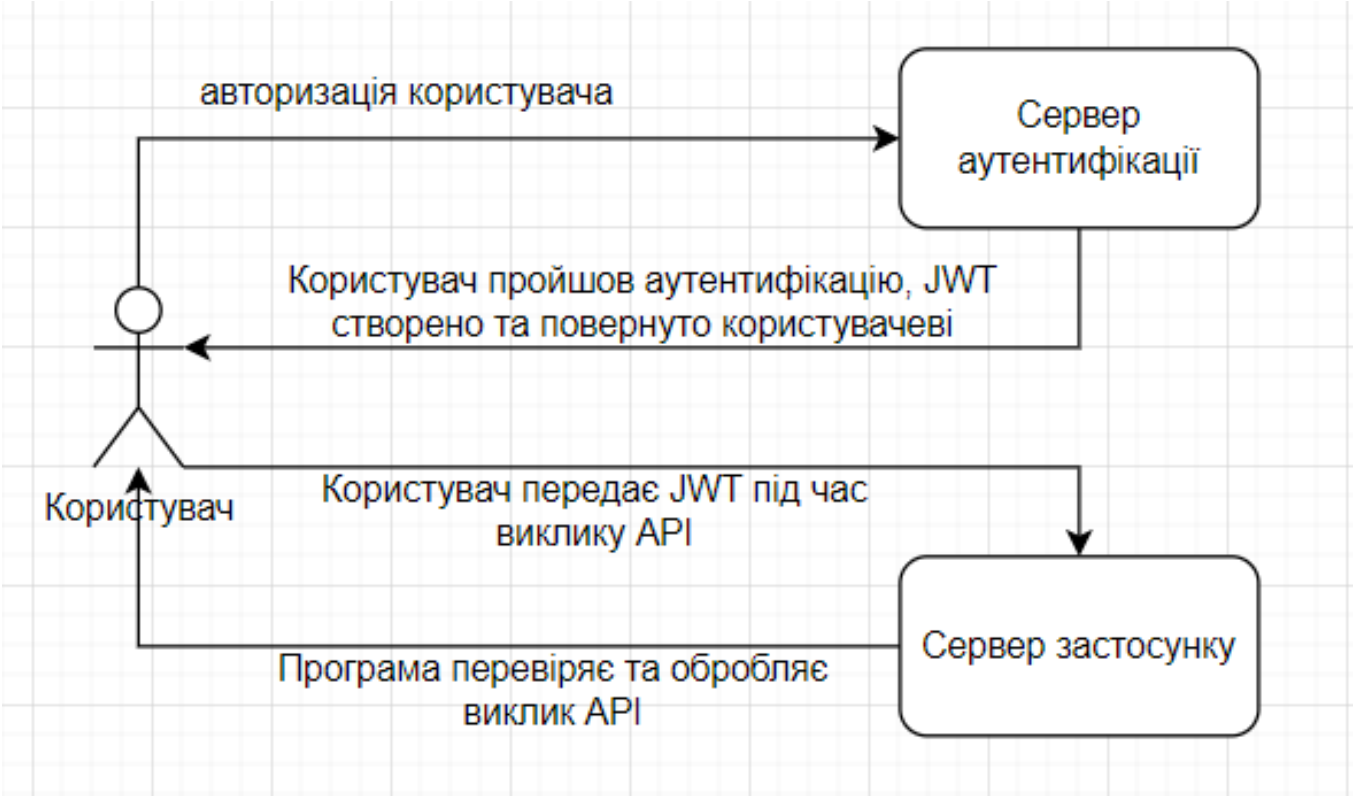

Рисунок 2.2 – Схема роботи JWT

Вебзастосунок використовує JWT для веревірки даних наступним чином:

1. Спочатку користувач відвідує сервер аутентифікації за домопогою аутентифікаційного ключа наприклад: логін/пароль, Facebook або Google ключ.
2. Сервер аутентифікації створює JWT і відправляє його користувачу.
3. Коли користувач робить запит до вебзастосунку через API, він додає до нього отриманий раніше JWT.
4. Коли користувач виконує запит API, додаток може перевірити за переданим запитом JWT чи є користувачем тим, ким є. У цій схемі додатків сервера налаштовано так, що можна перевірити, чи є JWT саме тим, що був створений сервером аутентифікації.

Система за допомогою якої можна декодувати, перевіряти та генерувати JWT (рис. 2.3) [22].

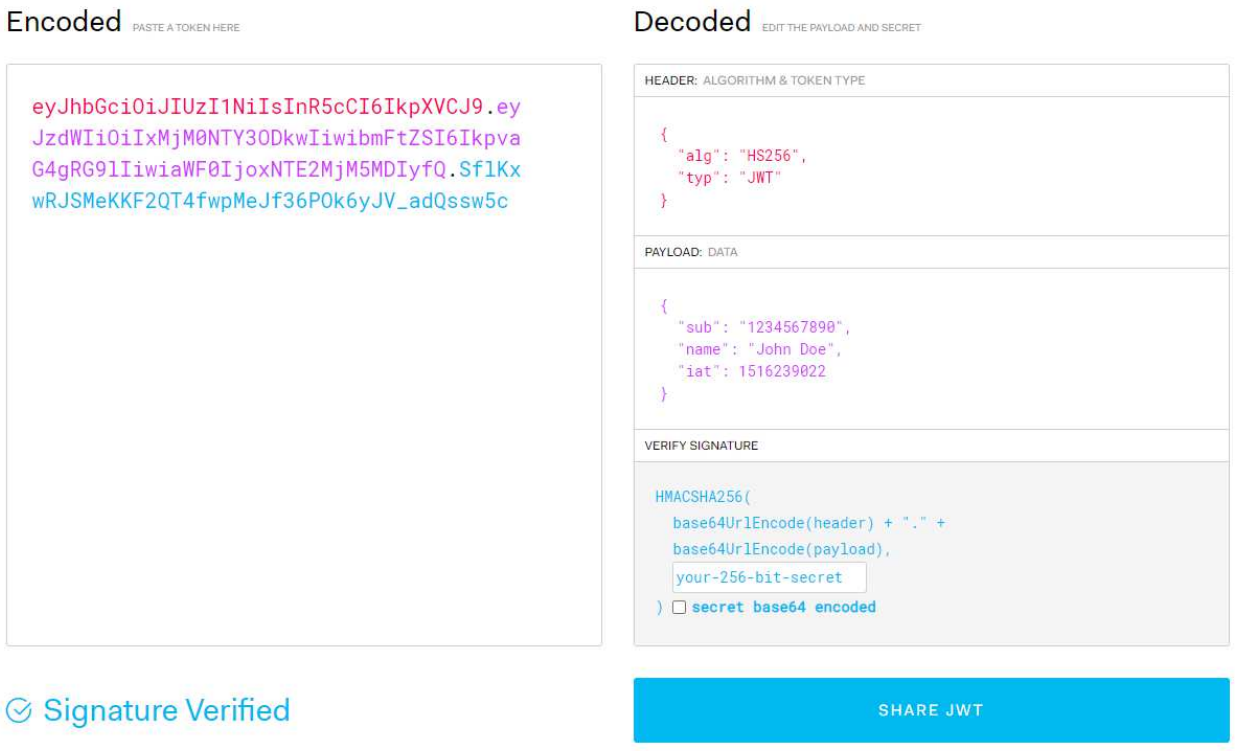

Рисунок 2.3 – Декодування, перевірка та генерування JWT сайту https://jwt.io/

При отриманні засекреченого JWT за допомогою сервісу є можливість декодувати токен або навпаки при наявності даних у вигляді JSON-у можна створити новий JWT формат даних для відправки певному об’єкту.

2.4 Моделювання діаграми прецедентів та діаграми розгортання системи хмарного сховища

Моделювання візуальних об’єктів надає можливість презентувати ідеї та схеми роботи за допомогою використання моделей.

Моделювання є важливим кроком при розробці ПЗ. Моделювання ПЗ надає можливість чітко побачити структуру майбутнього програмного забезпечення.

Зазвичай при моделюванні, розробники відображають систему графічно, так як людині зручніше зрозуміти модель в такому представленні. Саме тому, замість того, щоб писати великі описи ПЗ, розробники будують різноманітні діаграми для опису своїх систем.

Коли розробник створює застосунок, то в першу чергу постають наступні питання:

– Що буде роботи застосунок?
– Хто буде використовувати ПЗ?

Деякими застосунками може користуватися певна множина людей, тому є необхідність виділяти різноманітні кластери користувачів системи. В кожному кластері можуть бути свої правила і можливості в системі. Для того, щоб мати можливість описувати різноманітні кластери користувачів та їх функціональні можливості в застосунку, створюється – діаграма варіантів використання (use-case diagram).

Діаграма варіантів використання (use-case diagram) – модель, що описує функціонал застосунку системи, який доступний кожному кластеру користувачів.

Діаграми прецедентів використовується зазвичай для збору вимог до використання системи. В залежності від вимог, є можливість використовувати дані різноманітними способами.

Кожна група користувачів на діаграмі варіантів використання зображена людиною, під якою можна внести назву кластера користувачів, яку він визначає (рис. 2.4).

Рисунок 2.4 – Кластер користувачів СХС

В термінології UML, людина зображена на діаграмі називається актором. Актор визначає будь-яку сутність, що задіяна в системі. Цими сутностями можуть бути люди, які грають певну роль в системі: клієнт, адміністратор, редактор, читач.

Кожен кластер користувачів в системі використовує певну функцію системи. На діаграмі використання така функція системи зображується еліпсом, всередині якого записується ім’я функції с пояснювальними дієсловами.

Рисунок 2.5 – Варіант використання СХС

Згідно термінології UML, слово еліпс виражає значення варіанта використання (use-case). В цілому, варіант використання виражає набір дій, що може бути використаним актором в діаграмі варіантів використання.

Сполучними елементами між варіантами використання та акторами слугує відношення. Кожне відношення між елементами має свою назву і слугує для досягнення певної мети. До відношень входять наступні види зв’язків:

– відношення асоціації (association relationship);

– відношення узагальнення (generalization relationship);
– відношення включення (include relationship);
– відношення розширення (extend relationship);

Відношення асоціації на діаграмі відображає інформацію про то, які варіанти використання можуть бути використані кожним актором.

Наведено діаграму прецедентів з урахуванням акторів, варіантів використання та зв'язків між ними (рис. 2.6).

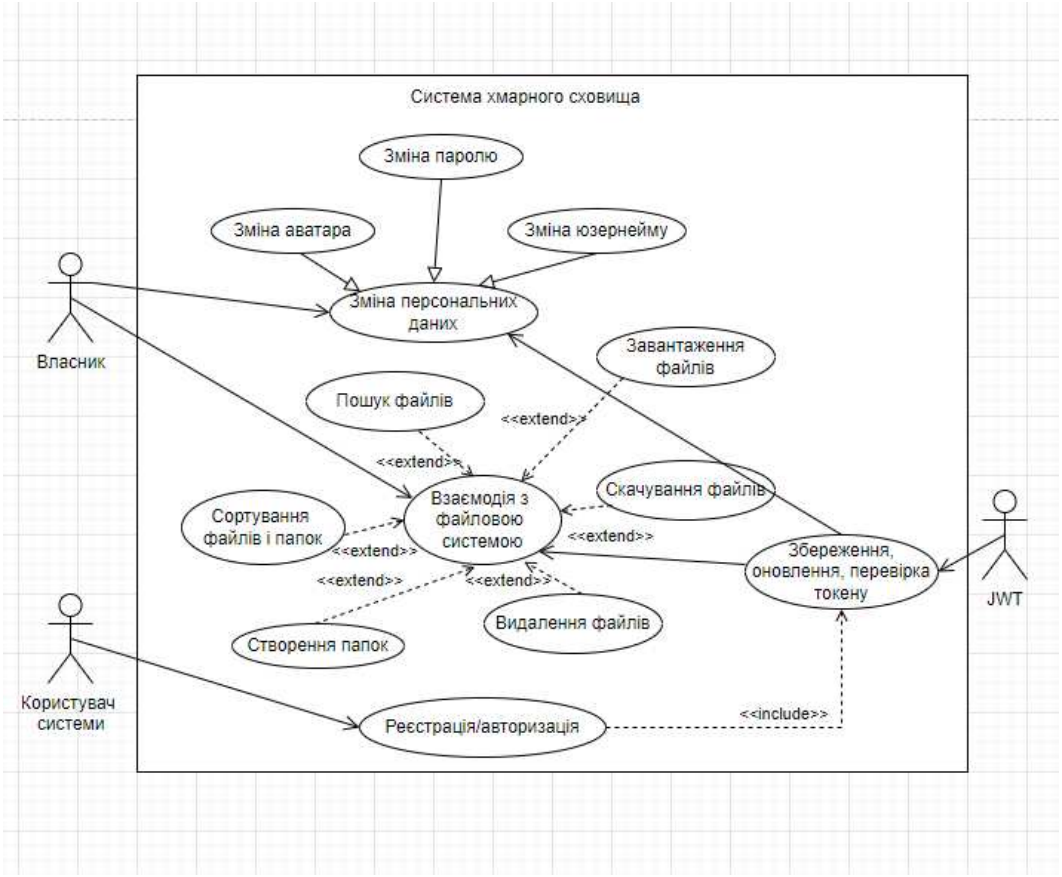

Рисунок 2.6 – Діаграма прецендентів СХС

В діаграмі варіантів використання системи хмарного сховища є 3 типа акторів: користувач системи (людина, яка перший раз потрапила та дізналася про систему), власник (персона, що вже є авторизованою та використовує весь потенціал системи) та JWT (частина системи, що відіграє роль захисту в системі). Також використовуються 3 типи зв'язків: відношення асоціації, узагальнення, включення та розширення). Крім цього наведені наступні варіанти використання: реєстрація/авторизація (включає в себе варіант використання, що відповідає за

збереження, оновлення та перевірку токена), взаємодія з файловою системою (розширює наступні варіанти використання: пошук, завантаження та видалення файлів, сортування файлів і папок, створення папок), зміна персональних даних (узагальнює наступні варіанти використання: зміна юзернейму, аватару, паролю). Всі актори системи мають відношення асоціацію до варіантів використання, що вміщують відношення розширення для інших варіантів використання.

Діаграма розгортання – фізичне розгортання інформації, що виробляється програмою на апаратних компонентах. В діаграмах UML, діаграма розгортання приймає участь в кластері структурних діаграм, тому що описує один із аспектів самої системи.

До складу діаграми розгортання вміщується декілька типів фігур UML, а саме: вузли, що символізують базові програмні або апаратні елементи та трьохмірні блоки. Лінії використовуються для позначення зв'язків від одного вузла до іншого. Фігури меншого розміру, що знаходяться всередині блоків визначаються як програмні артефакти, що розгортаються на вузлах (рис.. 2.7).

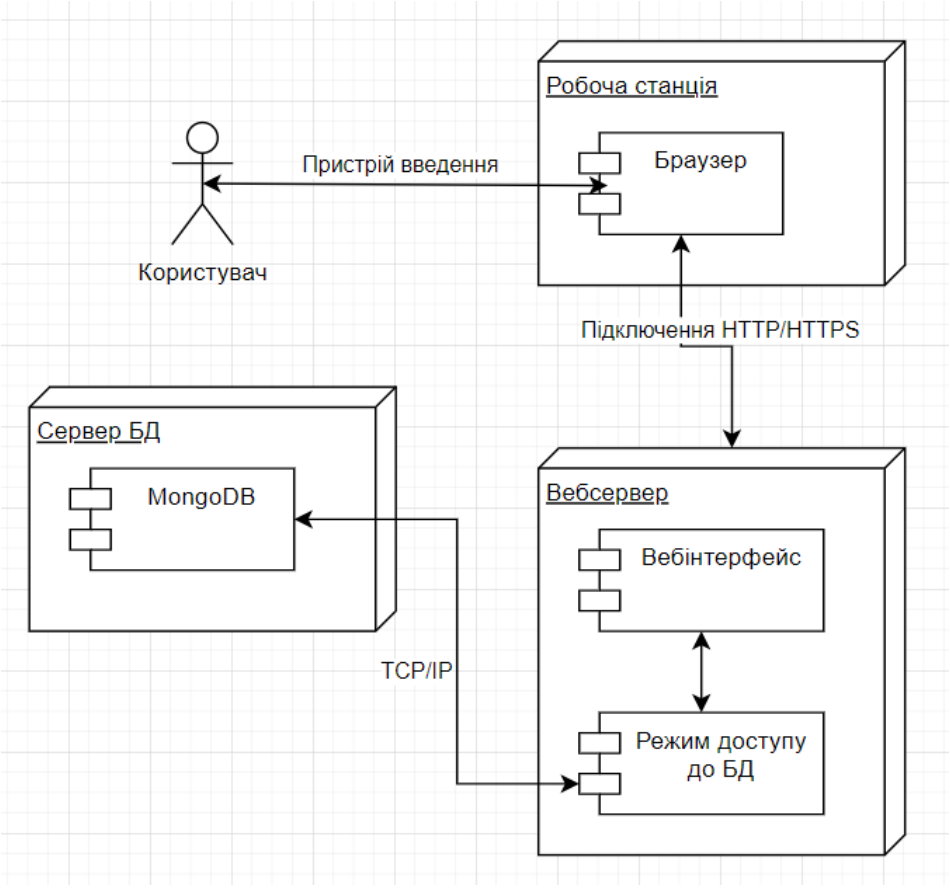

Рисунок 2.7 – Діаграма розгортання до СХС

До діаграми розгортання системи хмарного сховища входить: актор – користувач, що за допомогою пристрою введення взаємодіє із блоком – робоча станція, до якого входить браузер. Браузер в свою чергу за допомогою запиту у форматі HTTP\HTTPS звертається до вебсерверу, котрий в свою чергу за допомогою вебінтерфейсу в залежності від доступу до БД взаємодіє з сервером БД.

Діаграма розгортання має свою область використання, а саме:

– наглядно показати, які саме програмні елементи розгортаються на тих чи інших апаратних компонентах;

– зробити огляд топології апаратного комплексу;

– проілюструвати обробку процесів виконання апаратними компонентами.

2.5 Огляд парттерну MVC в системі хмарного сховища

Паттерн – архітектурне рішення, що дозволяє повторювано використовувати готове рішення для вирішення проблеми проєктування в рамках певного контексту.

MVC – це фундаментальний паттерн, який надає архітектурне рішення при проєктуванні структури проєкту, зокрема back-end частини ПЗ. Його використання значно спрошує робочий процес розробника та зберігає певний стандарт гарного тону при побудові проєкту.

Паттерн MVC перекладається наступним чином: M (model) – модель, V (view) – представлення або вид, C (controller) – контроллер.

Під блоком модель зазвичай підкреслюється функціонал, що відповідає за бізнес-логіку програмного забезпечення. Модель характеризується незалежністю від інших частин проєкту та зазвичай відповідає за звернення до БД. Модельний блок проєкту не повинен знати про інші елементи проєкту і як він має відображатися (рис. 2.8).

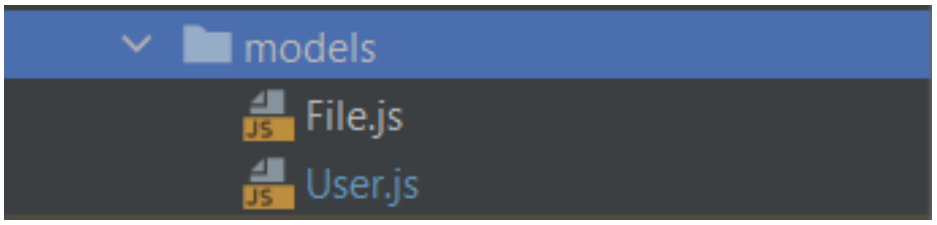

Рисунок 2.8 – Модель в структурі СХС

Модель вміщує в себе 2 файли: File.js з схемою бази данних колекції File та User.js з схемою бази даних колекції User. Модель експортується та використовується в контроллері.

В поняття представлення входить відображення отриманих даних на front-end застосунку. Проте представлення впливати на модель не може напряму. Таким чином, представлення має тільки доступ для отримання даних та його подальшого виводу (рис. 2.9).

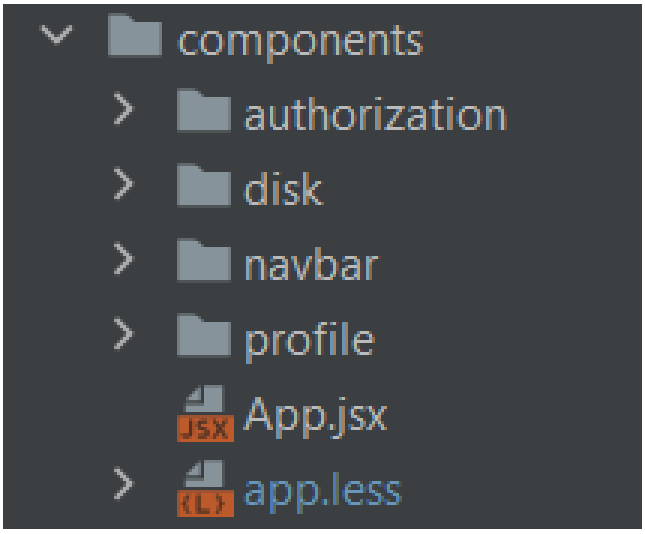

Рисунок 2.9 – Представлення в структурі СХС

В вебзастонку прдеставлення є стороною клієнта та працює за рахунок React компонентів, які отримують початкові дані за допомогою технології Redux і подальшої їх обробки через REST API. До компонентів входить authorization, disk, fileList, file, uploader, navbar та profile.

Контроллер в свою чергу оброблює дані, що були перехвачені в залежності від продуманої логіки застосунку. Контроллер визначає, яке представлення буде відображене в залежності від компоненту (функції), що було задіяне всередині контроллера (рис. 2.10).

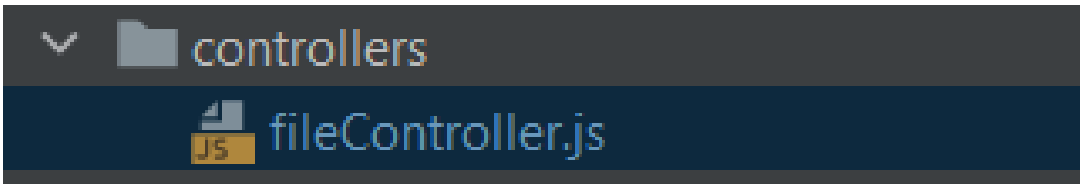

Рисунок 2.10 – Контроллер в структурі СХС

Вебзастосунок вміщує один файл контроллер – fileController.js, що вміщує в собі всю логіку маніпуляцій з файлами, якщо точніше, то асинхронні методи. Вони на основі отриманих даних з БД виконують певні маніпуляції з файлами: додавання, завантаження на сервер, завантаження із серверу, видалення і т.д.

Висновки до розділу 2

Отже, в другому розділі виконано огляд моделі системи хмарного сховища, тобто визначено архітектурну ідею вебзастосунку та функціонал, що входить до архітектури вебзастосунку. Також описані варіанти використання системи (usecases) хмарного сховища даних. Наведено 3 сценаріїв варіантів використання: авторизація до системи, завантаження файлів на сервер, створення директорій. Проаналізовано роботу вебзастосунку з використання JSON Web Token. Дано визначення поняттю JWT для подальшого його розгляду та наведено схему роботи JWT в системі хмарного сховища. Крім цього було наведено приклад, як можна верифікувати токен, декодувати, та генерувати сам JWT. Розглянуто моделювання діаграми прецедентів та діаграми розгортання системи хмарного сховища. Здійснено аналіз діаграми прецедентів на основі власної системи, де розглянуто актори, зв’язки та самі варіанти використання. Також наведено приклад діаграми розгортання на базі власного вебзастосунку. Описано з чого складається діаграма розгортання на основі власного проєкту. Виконано огляд паттерну MVC, який використовується для структуризації back-end частини вебзастосунку.

3 КОНСТРУЮВАННЯ ТА МОДЕЛЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ. АРХІТЕКТУРНІ РІШЕННЯ ДО СИСТЕМИ

3.1 Огляд стеку MERN в системі хмарного сховища

Стек – збірка технологій, що допомагає в рішенні того чи іншого завдання. Існує багато варіантів комбінації технологій, але один із затребуваних є MERN, який складається із:

- MongoDB (система управління БД);
- Express.js (фреймворк вебзастосунків під Node.js);
- React.js (бібліотека для розробки користувацьких інтерфейсів);
- Node.js (програмна платформа для розробки back-end частини).

MongoDB імплементує підхід до побудови БД без залучання таблиць, схем, SQLзапитів і всьому, що присутнє в об'єктно-реляційних БД. На відміну від реляційних БД, MongoDB застосовує документо-орієнтовану модель даних, через що є можливість швидке працювати з даними, отримувати кращу масштабованість і простоту у використанні.

Враховуючи всі недоліки та переваги, важливо мати розуміння, що задачі трапляються різні і засоби для їх вирішення можуть відрізнятися між собою. В певній ситуації використання MongoDB призведе до покращення ефективності ПЗ при збереженні важких по структурі даних. Але також може скластись ситуація, де краще все таки використати реляційні бази даних. Крім цього завжди є можливість використання різних підходів, де один тип даних зберігається в традиційних – реляційних БД, а інший в MongoDB.

Повна система MongoDB може представляти не тільки одну базу даних, що розташована на одному фізичному сервері. Функціональність MongoDB надає можливість розмістити відразу декілька баз даних на декількох фізичних серверах і такі бази даних можуть легко між собою взаємодіяти, обмінюватися даними, зберігати цільність (рис. 3.1).

```
const {model, Schema, ObjectId} = require('mongoose')


const File = new Schema({
    name: {type: String, required: true},
    type: {type: String, required: true},
    accessLink: {type:String},
    size: {type: Number, default: 0},
    path: {type: String, default: ''},
    date: {type: Date, default: Date.now()},
    user: {type: ObjectId, ref: 'User'},
    parent: {type: ObjectId, ref: 'File'},
    childs: [{type: ObjectId, ref: 'File'}],
})

module.exports = model( name: 'File', File)

```

Рисунок 3.1 – Приклад використання MongoDB в СХС

В прикладі відображається використання колекції File, що є модель в за паттерном MVC. В колекцію File входить наступні документи: name, type, accessLink, size, path, date, user, parent, childs. Кожний з документів має свій тип даних та інші довільні поля. За рахунок цього, модель File можна експортувати до інших частин часту, наприклад до контроллеру.

Node.js платформа, що використовується як середовище виконання коду JavaScript. Платформа побудована на основі двигуна JavaScript Chrome V8 з можливістю перекладу JS в машинний код. Перш за все, суть Node.js полягає в підтримки можливості розробки серверних застосунків мови JavaScript. Під час створення системи хмарного сховища використовувалася наступна версія Node.js – v14.16.0 (рис. 3.2).

```
C:\Users>node -v
v14.16.0
```

Рисунок 3.2 – Версія Node.js під час створення проєкту

В проєкті Node.js – це фундамент всього back-end вебзастосунку, починаючи з налаштування сервера, підключенням MongoDB, завершуючи написанням логіки для взаємодії client-server. Node.js враховуючи паттерн MVC (на основі якого побудований back-end) зв'язує небхідні компонени для побудови цільного застосунку, а саме: бібліотеки, маршрути, моделі, контроллери, статичні файли, конфіги, проміжне програмне забезпечення, сервіси (рис. 3.3).

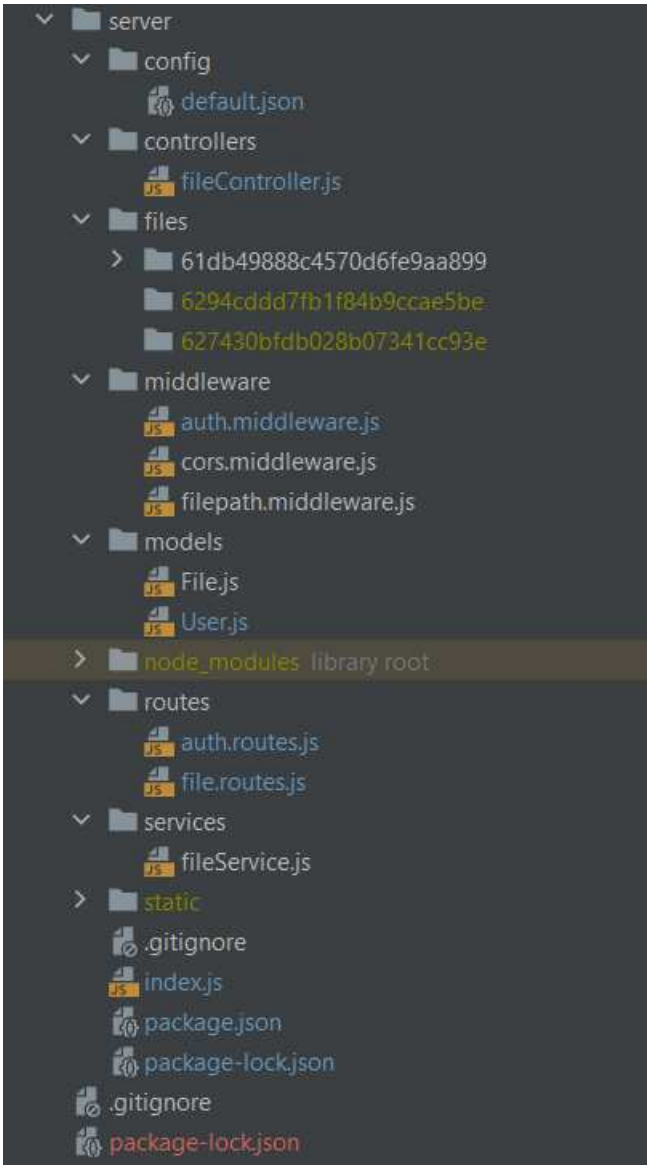

Рисунок 3.3 – Відображення структури серверної частини вебзастосунку

Проєкт также вміщує в себе стандартний пакет модулів – node_modules, .gitignore (для налаштування видимості файлів в git), package.json та package-lock.json для налаштувань проєкту, залежностей та окремих бібліотек.

React – бібліотека JavaScript. Бібліотека react задіяна для створення

користувацького інтерфейсу. Версія на якій створений проєкт – 18.0

React зручний інструмент для побудови масштабних вебзастосунків, а саме його front-end частини. З його допомогою можна також створювати вебзастосунки на базі архітектури SPA.

Особливістю React є віртуальний DOM, який має не важку за об’ємом копію звичайного DOM. І суть в тому, що React працює саме з віртуальний DOM, а не зі звичайним. В той час, як застосунку потрібно звернутися до стану елементів, то виконується звернення до віртуального DOM (рис. 3.4).

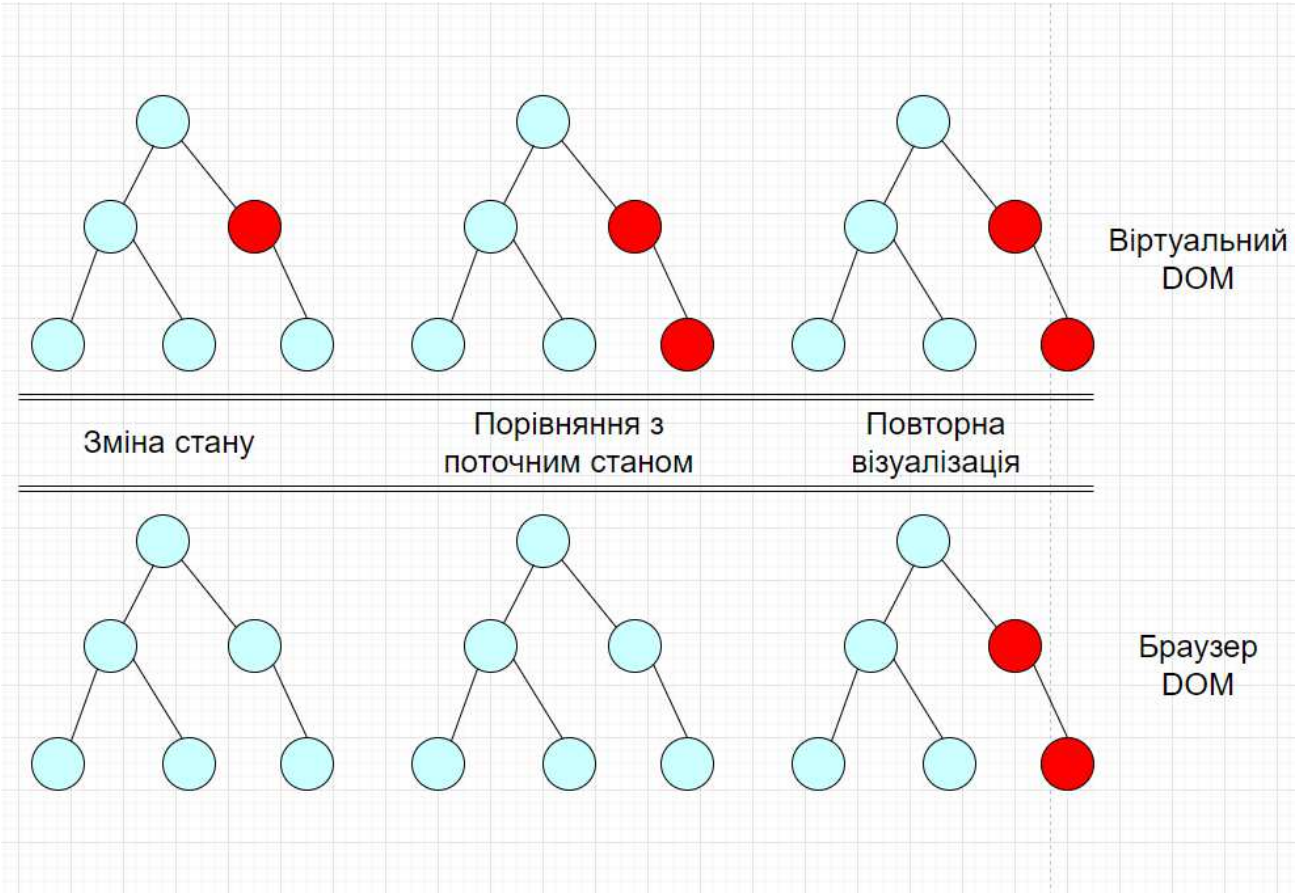

Рисунок 3.4 – Відображення порівняння віртуального та браузер DOM

При спробі дізнатися інформації про стан елементів, виконується звернення до віртуального DOM. Якщо потрібно виконати зміни в елементи якогось вебкомпоненту, то спочатку зміни вносяться у віртуальний DOM. Після цього стан віртуального DOM порівнюється із поточним станом і якщо вони відрізняються, то React знаходить мінімальну к-сть маніпуляцій, які потрібні, щоб обновити реальний DOM до нового стану. Як результат, взаємодія із елементами

вебзастосунка проходить швидше та ефективніше, на відміну від втручання JavaScript в DOM напряму.

Іншою особливістю React є компонентний підхід в побудові архітектури ПЗ. При використанні компонентів React, є можливість подальшого їх перенесення в інші проєкти.

Не менш важливо підмітити особливість використання JSX в React. JSX – це комбінація коду JS та XML, що надає простий та зрозумілий спосіб поєднання коду візуального інтерфейсу (рис. 3.5).

```
return (
  <div className="popup" onClick={() => dispatch(setPopupDisplay( display: 'none'))} style={{display: popupDisplay}}>
    <div className="popup__content" onClick={(event => event.stopPropagation())}>
      <div className="popup__header">
        <div className="popup__title">Create new folder</div>
        <button className="popup__close" onClick={() => dispatch(setPopupDisplay( display: 'none'))}>X</button>
      </div>
      <Input
        type="text"
        placeholder="Enter folder name"
        value={dirName}
        setValue={setDirName}
      />
      <button className="popup__create" onClick={() => createHandler()}>Create</button>
    </div>
  </div>
);
```

Рисунок 3.5 – Приклад використання JSX в проєкті

Express – не менш важливий фреймворк, що реалізую програмний каркас серверної частини вебзастосунку. Даний фреймворк значно спрощує роботу з серверної частиною, яка могла би бути реалізована на чистому Node.js.

Проте Express сам використовує модуль HTTP і пропонує низку готових абстракцій, що значно спрощує створення сервера і логіки вебзастосунку, зокрема робота з куками, CORS, відправка форм.

3.2 Проєктування UML-діаграм: ED, діяльності, компонентів

ERD або ER-діаграма – це певний різновид блок-схеми, де відображаються сутності, наприклад: користувачі, об’єкти, парадигми. Крім цього дані сутності можуть бути зв’язані між собою за допомогою певних зв’язків (one, many, zero or

one, one or many, zero or many). Дані діаграми часто використовуються для проєктування та відлагодження реляційних БД різноманітних сфер, що проєктуються. Для візуально відображення до ERD входять наступні фігури: прямокутники, ромби, овали, поєднуючи лінії (рис. 3.6).

В ER-моделях зазвичай відокремлюють до трьох рівнім деталізації:

– Концептуальна модель даних – схема найвищого рівня з мінімальною к-стю деталей. Особливістю такого підходу є можливість відобразити загальну структуру моделі та архітектуру системи в цілому. Не великі за об’ємом моделі можуть не використовувати даний підхід деталізації, а відразу переходити на рівень логічних моделей.

– Логічна модель – вміщує більш детальну інформацію ніж концептуальна модель. На даному рівні визначаються більш детальні транзакційні сутності. Ще одною особливістю логічної моделі є незалежність від технологію, що буде застосовуватися.

– Фізична модель – ще більш детальна ніж логічна. На базі логічної моделі, можна скласти декілька фізичних. В ній повинно бути присутнє достатньо технічних деталей для побудови самої БД.

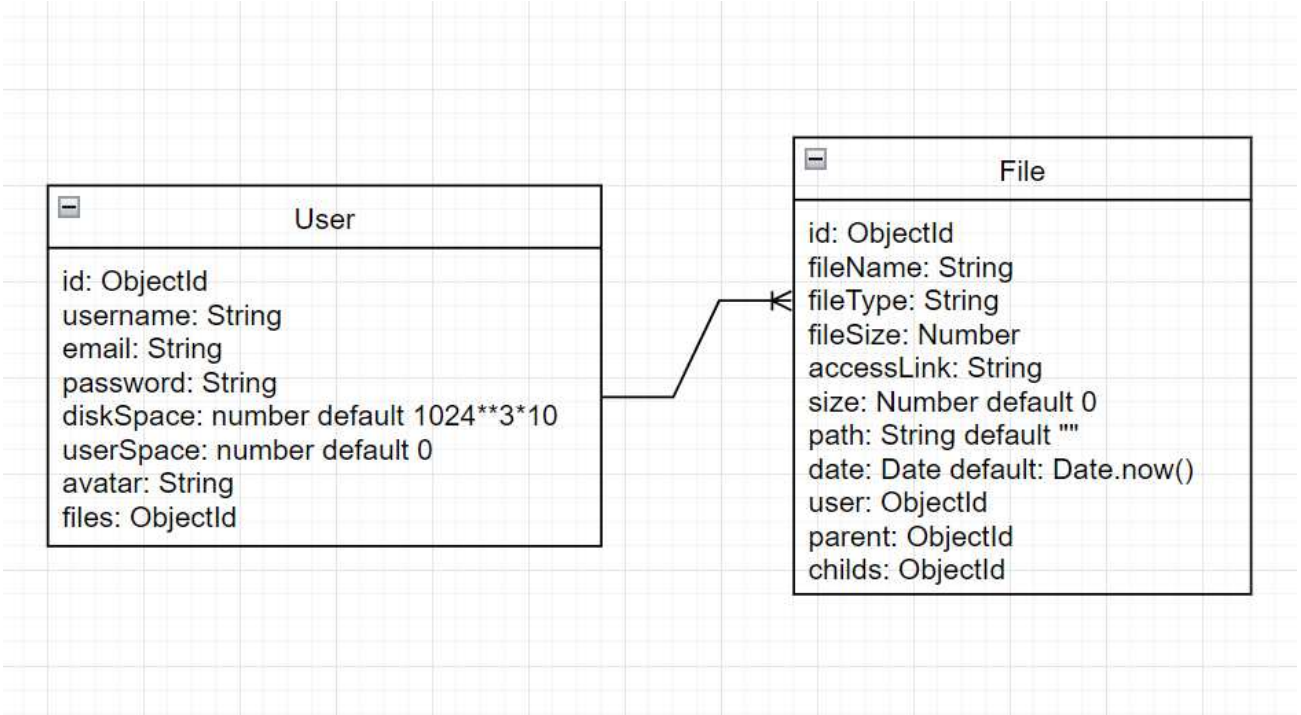

Рисунок 3.6 – ER-діаграма СХС

В даній діаграмі представлено дві сутності User та File, що поєднані зв’язком one to many. Сутність User складається з наступних полів: id, username, email, password, diskSpace, userSpace, avatar, files. Сутність File складається з таких полів: id, fileName, fileType, fileSize, accessLink, size, path, date, user, parent, childs. В архітектурі системи дані сутності є моделями паттерну MVC та поля всередині них є документами середовища MongoDB.

Діаграма діяльності – візуальне відображення графу діяльностей. Дія в такій діаграмі є базовою одиницею поведінки в специфікації. В дію входить низка вхідних сигналів, що перетворюються на вихідні дані. Кожна дія в діаграмі діяльності виконується один або більше разів під час однієї реалізації діяльності. Специфікація діяльності дає можливість виконувати кілька потоків, та існування технології синхронізації для забезпечення виконання дій у вірному порядку (рис. 3.7).

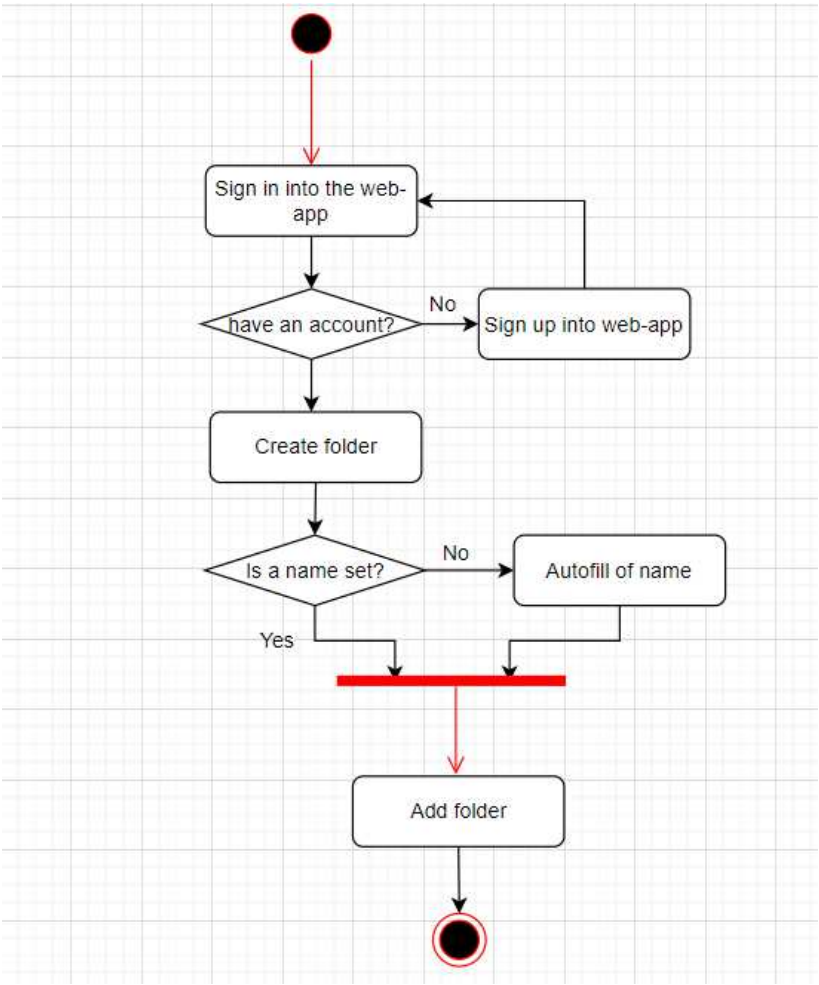

Рисунок 3.7 – Діаграма діяльності для створення директорії

На діаграмі діяльності користувач з початкової точки авторизовується до вебзастосунку. Якщо немає облікового запису, то користувач реєструється та авторизується в системі. Після цього користувач виклакає popup-форму для створення папки. В залежності було введено поле ім’я чи ні, генерується папка (рис. 3.8).

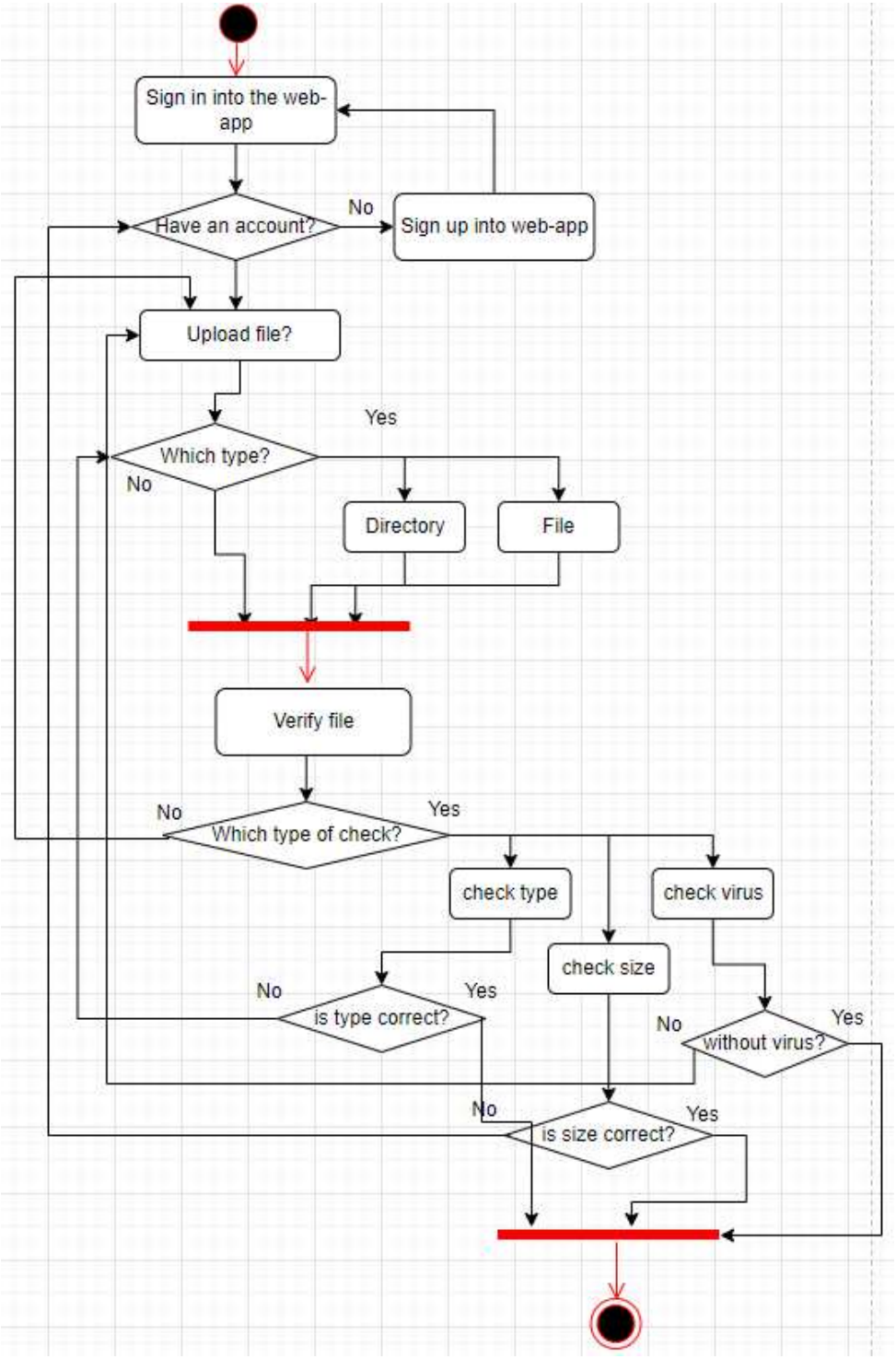

Рисунок 3.8 – Діаграма діяльності для створення директорії

На діаграмі діяльності для створення директорії користувач має зареєструватися або авторизуватися, завантажити певний тип файлу з верифікаціями та після отримати результат на front-end.

В той час як інші види UML діаграм описують функціональность системи, компонентні діаграми описють моделюючи компоненти, які допомогають створити функіональность можливою.

Діаграма компонентів – показує фізичне представлення ПЗ у вигляді групи елементів, які називаються компонентами. В списку об’єктів компонентів розглядаються тільки файли, модулі, бібліотеки, пакети і т.д.

Компонент – це фізичне представлення існуючої частини системи, завдяки якої забезпечується реалізація класів та відношень.

До компонентів можна віднести: БД, користувацький інтерфейс, апаратні компоненти (схема, мікросхема, гаджет), доставка, платіжна відомість.

Компонентні діаграми:

– використовуються в компонентно-орієнтованих відділах для документування систем архітектури, орієнтованих на сервіс;

– можуть використовуватися для фокусування на відношеннях між компонентами;

– допомагати в інформуванні і поясненні функцій системи, що розробляється зацікавленим сторонам.

Для компонентів UML визначають наступні об’єкти:

1. file (файл) – найвідоміша різновидність компонента, що приймає вид будь-якого файлу;
2. executable (виконавчий) – різновид файлу, що є виконавчим та може виконуватися на будь-якій платформі;
3. document (документ) – різновид файлу, який не виконує та не вміщує в собі код застосунку;
4. library (бібліотека) – різновид файлу в форматі бібліотеки;
5. source (джерело) – різновид файлу, що вміщує вихідний код застосунку;
6. table (таблиця) – різновид компонента у форматі БД.

Діаграма компонентів може мати декілька типів відношення: залежності та реалізації (рис. 3.9).

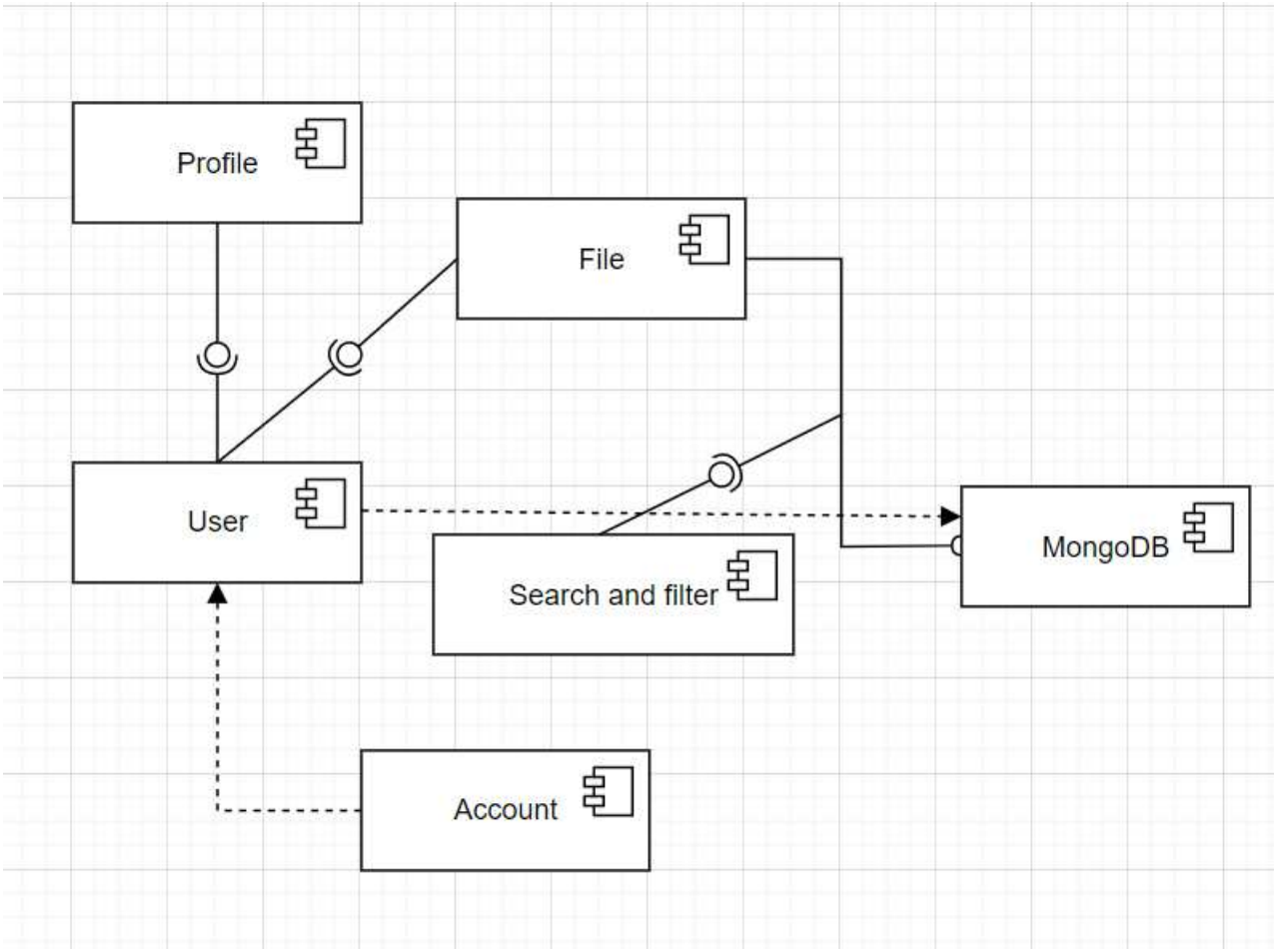

Рисунок 3.9 – Діаграма компонентів СХС

На діаграмі компонентів системи хмарного сховища відношення залежності між користувачем та БД – MongoDB та між обліковим записом та користувачем. Відношення реалізації між профілем та користувачем, між файлом та користувачем й пошуком, фільтрацією та відношенням вимоги файлу до БД.

3.3 Опис інтерфейсів програмного забезпечення

Початкова сторінка на яку потрапляє користувач – це сторінка реєстрації та авторизації до системи. Для користувача стає доступним обрати сторінку реєстрації, якщо користувач ще не був зареєстрованим або сторінку авторизації, якщо користувач не перший раз користується системою (рис. 3.10).

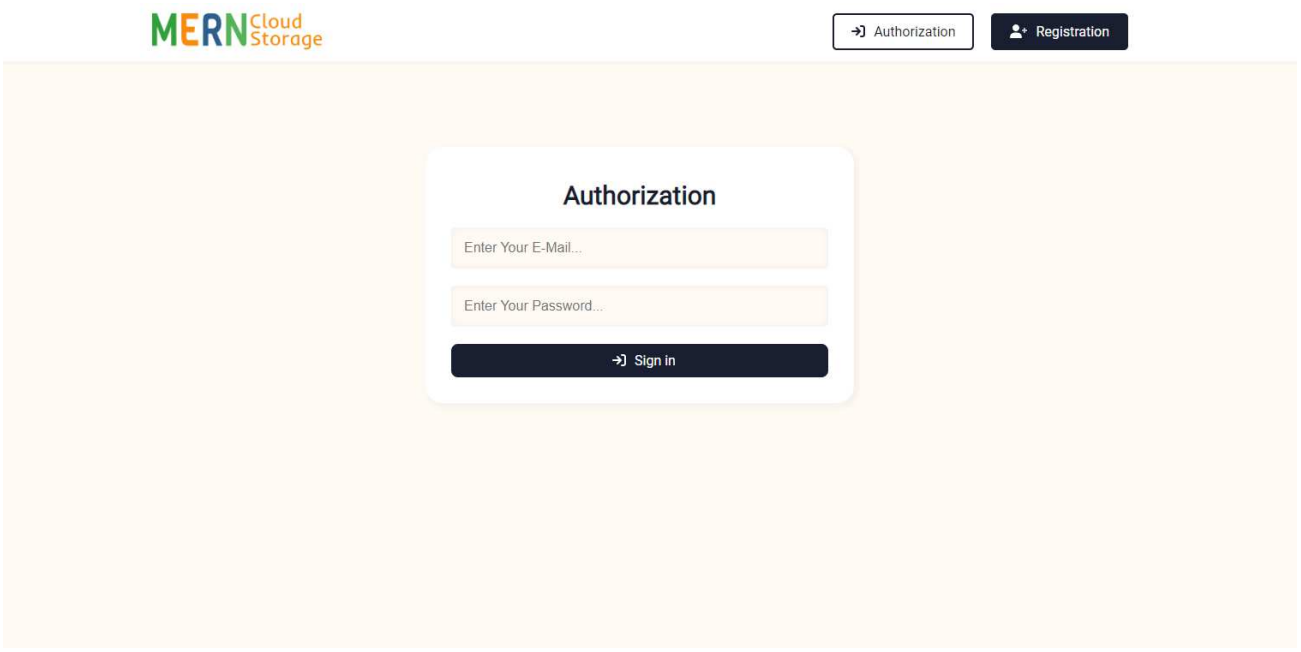

Рисунок 3.10 – Сторінка авторизації до СХС

Сторінка авторизації до системи складається із полів: email та password. На відміну від реєстрації, де ще додатково потрібно заповнити ім’я користувача. Після цього потрібно переходити на сторінку авторизації та заповнювати поля, що були зареєстровані до системи.

Головна сторінка файлової системи (рис. 3.11) вміщує список завантажений файлів, при наведені на елемент якого можна отримати додаткову можливість завантаження та видалення файлів із списку. Список може бути довільного розміру по висоті та відображатися за допомогою скролу вниз. Сторінка також має херед, тільки наразі для авторизований користувачів відображається поле пошуку, кнопка виходу із облікового запису та посилання на сторінку профілю. Під хедером розміщені деякі навігаційні елементи: кнопка повернення на попередню сторінку та кнопка для створення директорії.

Також розміщена кнопка, що відповідає за функціонал завантаження файлів на сервер, фільтр по назві, типу та даті у вигдялі випадаючого списку, а також елементи перемикання способу відображення файлової системи (зі списку до сітки).

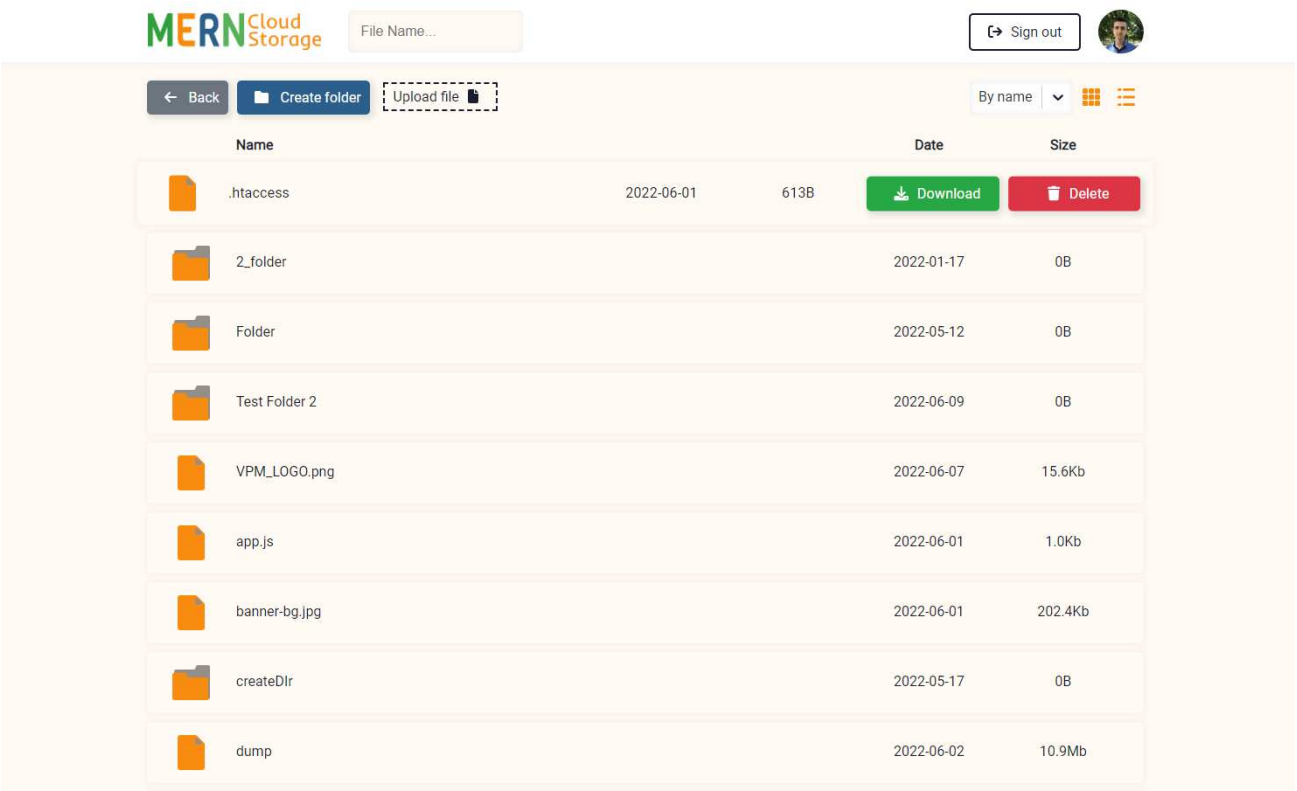

Рисунок 3.11 – Головна сторінка СХС

Режим відображення сітки можна отримати при натисненні іконки, відповідаючою за відображення сітки файлів. Для зміни на список, можна натиснути кнопку, що виводить список. (рис. 3.12).

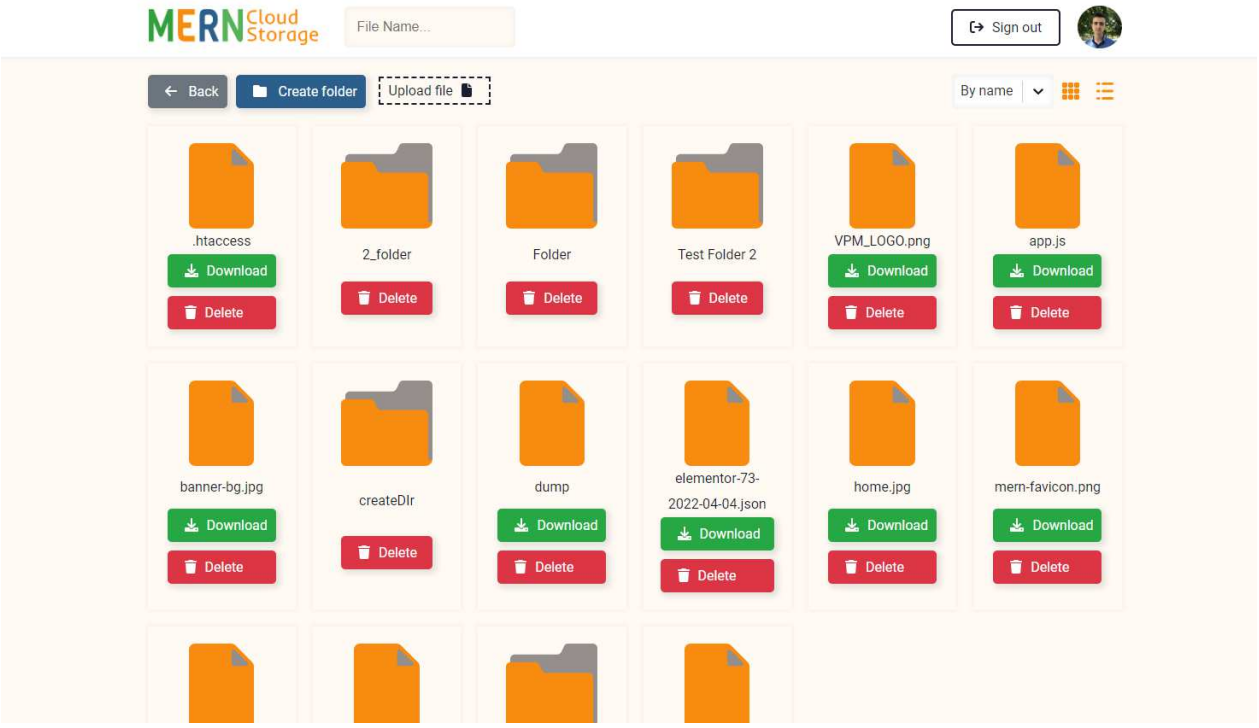

Рисунок 3.12 – Змінений режим відображення головної сторінки

Для завантаження файлів на сервер, користувач має можливість перетягнути необхідні файли до області завантаження або натиснути кнопку Upload file та перейти до режиму вибору файлів зі свого пристрою (рис. 3.13).

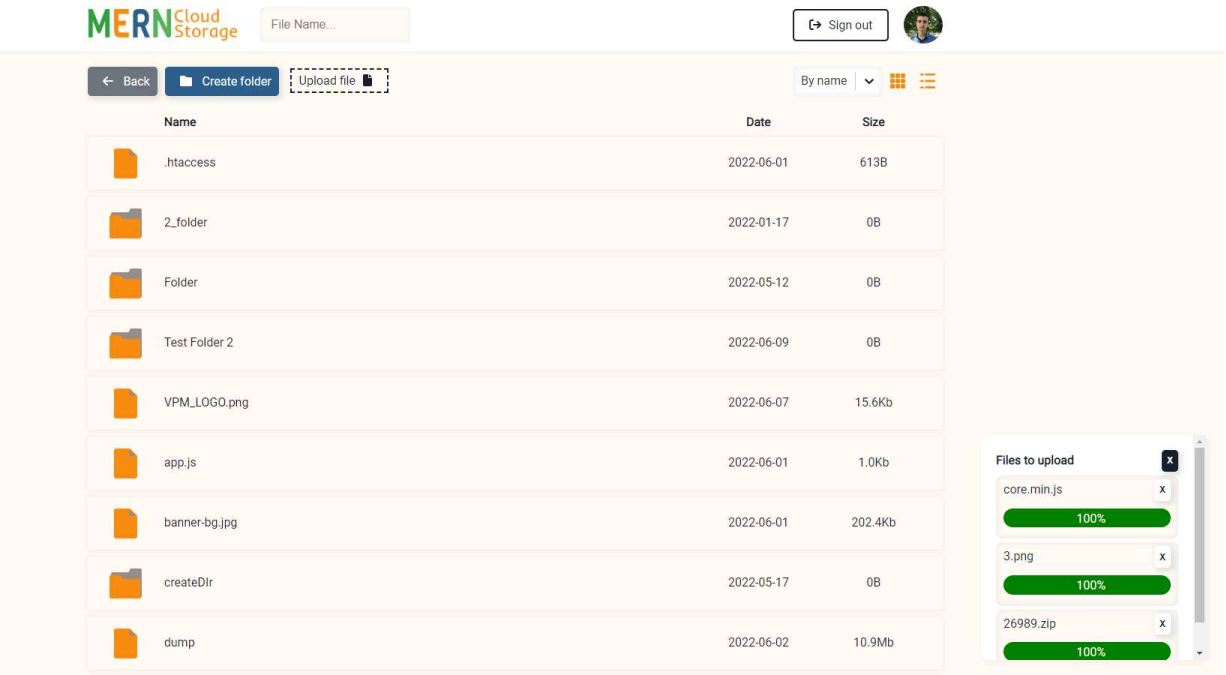

Рисунок 3.13 – Відображення списку завантажених файлів

Після завантаження файлів на сервер, відображається окреме вікно, що показує прогрес та список файлів, що були завантаженні чи ще завантажують на сервер.

Для створення папки, можна натиснути на кнопку Create folder та перейти до поп-апу, що пропонує ввести ім’я для папки (рис. 3.14).

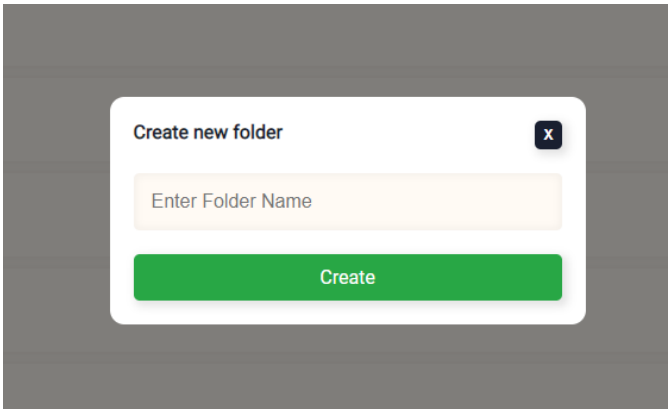

Рисунок 3.14 – Відображення pop-up для створення папок

Висновки до розділу 3

Отже, в третьому розділі виконано огляд стеку MERN, як збірку технологій для побудови проєкту. Оцінено кожну технологію в проєкті, її характеристики та переваги для побудови СХС. Спроєктовано ER-діаграму, що відображає сутності в системі. Візуально відображено діаграму діяльності, щоб показати можливу поведінку користувача в системі. Побудовано діаграму компонентів, що відображає фізичне представлення ПЗ у вигляді компонентів. Описано інтерфейси користувачів, а саме: сторінку реєстрації та авторизації користувачів (для отримання доступу до системи), головну сторінку системи (де відображено головний функціонал системи в межах декількох кліків), можливість завантаження файлів на сервер та відображення списку завантажений файлів, можливість створення папок через pop-up.

4 КОДУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ. АНАЛІЗ РЕЗУЛЬТАТІВ ОБЧИСЛЕНЬ ТА КЕРІВНИЦТВО КОРИСТУВАЧА

4.1 Кодування та аналіз компонентів головної сторінки системи. Компонентне керівництво користувача.

На головній сторінці застосунку розміщені основі функціональні компоненти системи, що взаємодіють з хмарним сховищем. До цих компонентів відноситься:

- створення папок;
- завантаження файлів на сервер;
- завантаження файлів із серверу;
- видалення файлів;

За створення папок на сервері відповідають певні функції. Для цього було створено контроллер fileController.js (рис. 4.1).

```
class FileController {
    async createDir(req, res) {
        try {
            const {name, type, parent} = req.body
            const file = new File({name, type, parent, user: req.user.id})
            const parentFile = await File.findOne({_id: parent})
            if(!parentFile) {
                file.path = name
                await fileService.createDir(req, file)
            } else {
                file.path = `${parentFile.path}\\${file.name}`
                await fileService.createDir(req, file)
                parentFile.childs.push(file._id)
                await parentFile.save()
            }
            await file.save()
            return res.json(file)
        } catch (e) {
            console.log(e)
            return res.status(400).json(e)
        }
    }
```

Рисунок 4.1 – Відображення асинхронного методу createDir

Асинхронний метод createDir приймає стандартні параметри request та result, після цього обертається в try…catch та виконується взаємодія з базою даних MongoDB. Спочатку перевіряється наявність батьківської директорії і якщо її немає, виконується сервіс createDir, що за допомогою Promise створює папку або в іншому випадку виводить помилку. Якщо ж батьківська директорії існує, то виконується додавання папки з урахуванням шляху батьківської директорії. І піся цього виконується збереження даних в БД.

Крім цього в action file.js додається функція createDir, в якій формується post запит наступного виду (рис. 4.2).

```
export function createDir(dirId, name) {
    return async dispatch => {
        try {
            const response = await axios.post( url: `${API_URL}api/files`, data: {
                name,
                parent: dirId,
                type: 'dir'
            }, config)
            dispatch(addFile(response.data))
        } catch (e) {
            alert(e.response.data.message)
        }
    }
}
```

Рисунок 4.2 – Відображення функції createDir

Назви спільних методів в контроллері та екшинах співпадають задля розуміння зв’язку між ними. В даній функції виконується post запит з передачою даних: name, parent, type. Після цього викликається метод dispatch технології Redux, що за допомогою екшена addFile передає дані до редюсера – fileReducer.js.
В редюсері в свою чергу зберігається дефолтний стан у вигляді об’єкту JavaScript. В нього входить властивість files, що приймає значення у вигляді масиву і коли потрібно змінити дане значення, fileReducer розгортає старе значення за допомогою спред-оператора та на основі старого значення додається нове, що прийшло як параметр з екшин-кріейтера addFile (рис. 4.3).

```
const ADD_FILE = "ADD_FILE"
const SET_POPUP_DISPLAY = "SET_POPUP_DISPLAY"
const PUSH_TO_STACK = "PUSH_TO_STACK"
const POP_FROM_STACK = "POP_FROM_STACK"
const DELETE_FILE = "DELETE_FILE"
const SET_VIEW = "SET_VIEW"

const defaultState = {
    files: [],
    currentDir: null,
    popupDisplay: 'none',
    dirStack: [],
    view: 'list'
}

export default function fileReducer(state = defaultState, action) {
    switch (action.type) {
        case SET_FILES: return {...state, files: action.payload}
        case SET_CURRENT_DIR: return {...state, currentDir: action.payload}
        case ADD_FILE: return {...state, files: [...state.files, action.payload]}
        case SET_POPUP_DISPLAY: return {...state, popupDisplay: action.payload}
        case PUSH_TO_STACK: return {...state, dirStack: [...state.dirStack, action.payload]}
        case DELETE_FILE: return {...state, files: [...state.files.filter(file => file._id != action.payload)]}
        case SET_VIEW: return {...state, view: action.payload}
        default:
            return state
    }
}

export const setFiles = (files) => ({type: SET_FILES, payload: files})
export const setCurrentDir= (dir) => ({type: SET_CURRENT_DIR, payload: dir})
export const addFile = (file) => ({type: ADD_FILE, payload: file})
```

Рисунок 4.3 – Відображення файлу fileReducer.js

У файлі також зберігаються константи інших типів для екшен-кріейтерів, для спрошення роботи з ними, а також інші кейси функції fileReducer.

Поп-ап в якому надається можливість створювати нові папки та додавати їх до проєкту є – компонент Popup.jsx, який за допомогою засобів Redux, а саме методів useSelector та useDispach надає можливість взаємодіяти зі стором та отримувати актуальне значення редюсера. Діспатч в свою чергу надає можливість опрокидувати певні значення в редюсер, використовуючи екшен-кріейтери: createDir, setPopupDisplay. Інші елементи компонента представляють структуру модулю у виглідя вбудований html тегів в React та допоміжних компонентів для створення поля вводу. За динамічне введення та отримання даних в поле Input відповідає хук useState, що ініціалізує стан значення та стан встановленого значення імені папки. За відправку встановленого значення до БД відповідає

функція createHandler при натисненні кнопки Create (рис. 4.4).

```
const Popup = () => {
  const [dirName, setDirName] = useState( initialState: '')
  const popupDisplay = useSelector(state => state.files.popupDisplay)
  const currentDir = useSelector(state => state.files.currentDir)
  const dispatch = useDispatch()

  function createHandler() {
    dispatch(createDir(currentDir, dirName))
    setDirName('')
    dispatch(setPopupDisplay( display: 'none'))
  }

  return (
    <div className="popup" onClick={() => dispatch(setPopupDisplay( display: 'none'))} style={{display: popupDisplay}}>
      <div className="popup__content" onClick={(event => event.stopPropagation())}>
        <div className="popup__header">
          <div className="popup__title">Create new folder</div>
          <button className="popup__close" onClick={() => dispatch(setPopupDisplay( display: 'none'))}>X</button>
        </div>
        <Input
          type="text"
          placeholder="Enter folder name"
          value={dirName}
          setValue={setDirName}
        />
        <button className="popup__create" onClick={() => createHandler()}>Create</button>
      </div>
    </div>
  );
};

export default Popup;
```

Рисунок 4.4 – Відображення коду компонента Popup.jsx

На фронтенді можна отримати вивід поп-апу при настинесні на кнопку Create folder (рис. 4.5).

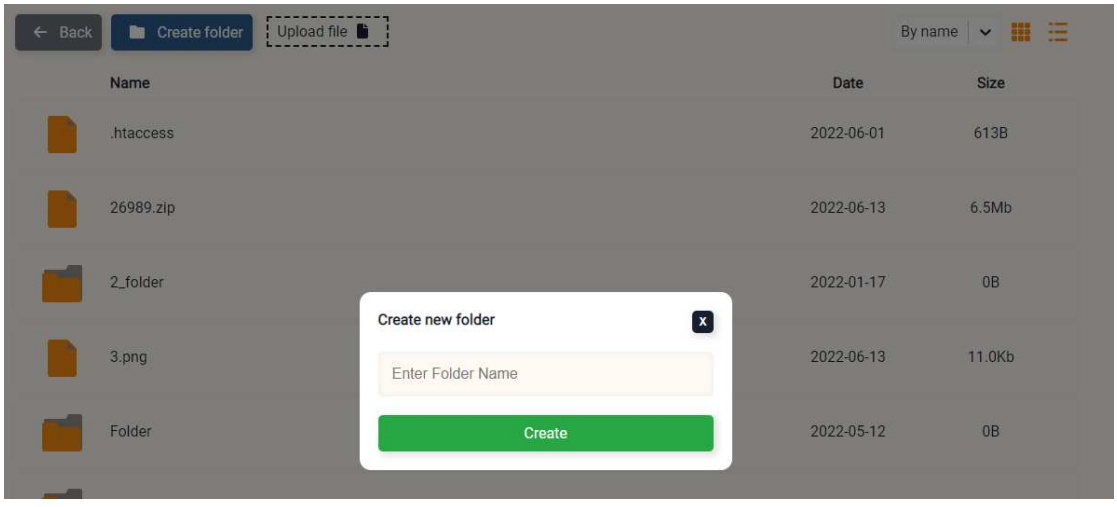

Рисунок 4.5 – Відображення компонента для створення папок

У користувача також є можливість закрити діалогове вікно або не запомнювати ім’я папки, щоб не додавати її до списку.

Для підрахунку розміру завантажених файлів підготовлена допоміжна функція, що перевіряє значення, яке передається як параметр функції (рис. 4.6).

```
export default (size) => {
  if(size > 1024*1024*1024) {
    return (size/(1024*1024*1024)).toFixed( fractionDigits: 1)+"Gb"
  }
  if(size > 1024*1024) {
    return (size/(1024*1024)).toFixed( fractionDigits: 1)+"Mb"
  }
  if(size > 1024) {
    return (size/(1024)).toFixed( fractionDigits: 1)+"Kb"
  }
  return size+"B"
}
```

Рисунок 4.6 – Відображення функції підрахунку розміру файлу

Значення перевіряється за певної умови, якій повинен відповідати розмір файлу. Так при знаходжені вірного значення файлу повертається його числовий еквівалент із приставкою Gb, Mb або Kb. І вже в компоненті є можливість використати функцію та підставити їй значення, яке було взяте із БД. Тому на екрані можна побачити вивід розміру для кожного завантаженого файлу чи створеної папки (рис. 4.7).

	Name	Date	Size
	.htaccess	2022-06-01	613B
	26989.zip	2022-06-13	6.5Mb
	2_folder	2022-01-17	0B
	3.png	2022-06-13	11.0Kb
	Folder	2022-05-12	0B

Рисунок 4.7 – Відображення компонента зі списком файлів

Для завантаження файлів на сервер створено асинхронну функцію в контроллері fileController.js, що передає з параметрів значення файлу, а з БД можна отримати батьківську папку та поточного користувача для подальшої перевірки та взаємодії з отриманими значеннями. При проходженні перевірок, визначається назва файлу разом з типом та додається до БД у вигляді json формату для подальшого збереження в БД (рис. 4.8).

```
async uploadFile(req, res) {
    try {
        const file = req.files.file
        const parent = await File.findOne({user: req.user.id, _id: req.body.parent})
        const user = await User.findOne({_id: req.user.id})
        if (user.usedSpace + file.size > user.diskSpace) {
            return res.status(400).json({message: 'There no space on the disk'})
        }
        user.usedSpace = user.usedSpace + file.size
        let path;
        if (parent) {
            path = `${req.filePath}\\${user._id}\\${parent.path}\\${file.name}`
        } else {
            path = `${req.filePath}\\${user._id}\\${file.name}`
        }
        if (fs.existsSync(path)) {
            return res.status(400).json({message: 'File already exist'})
        }
        file.mv(path)
        const type = file.name.split('.').pop()
        let filePath = file.name
        if (parent) {
            filePath = parent.path + "\\" + file.name
        }
        const dbFile = new File({
            name: file.name,
            type,
            size: file.size,
            path: filePath,
            parent: parent ? parent._id : null,
            user: user._id
        });
```

Рисунок 4.8 – Відображення коду файлу fileController.js

Паралельно запиту в контроллері, створено REST API запит на завантаження файлу. В ньому задіяний метод FormData() для подальшого

додавання типу file. Далі використано діспатч для показу прелоадера та додавання раніше створено об’єкту файлу з ім’ям, прогресом завантаження та поточною датою. Створено константну змінну response, що надсилає post запит з передачею самого файлу, заголовків (токена) та функції з відтворенням прогресу завантаження файлу на сервер. Після цього в файл редюсер додається файл, який пізніше довантажується на екран.

Для завантаження та видалення файлів створено схожі асинхронні функції downloadFile та deleteFile (рис. 4.9).

```
async downloadFile(req, res) {
    try {
        const file = await File.findOne({_id: req.query.id, user: req.user.id})
        const path = fileService.getPath(req, file)
        if (fs.existsSync(path)) {
            return res.download(path, file.name)
        }
        return res.status(400).json({message: "Download error"})
    } catch (e) {
        console.log(e)
        res.status(500).json({message: "Download error"})
    }
}

async deleteFile(req, res) {
    try {
        const file = await File.findOne({_id: req.query.id, user: req.user.id})
        if (!file) {
            return res.status(400).json({message: 'file not found'})
        }
        fileService.deleteFile(req, file)
        await file.remove()
        return res.json({message: 'File was deleted'})
    } catch (e) {
        console.log(e)
        return res.status(400).json({message: 'Dir is not empty'})
    }
}
```

Рисунок 4.9 – Відображення функцій donwloadFile та deleteFile

Функціонал подібний тим, що використовується запит до БД на основі request та подальше використання сервісу для отримання чи видалення файлу.

Після чого за допомогою Node.js використано вбудовані методи для завантаження та видалення.

Крім цього додані відповідні асинхронні функції для створення запитів на завантаження та видалення файлів, що вже мають деякі відмінності. (рис. 4.10).

```
export async function downloadFile(file) {
    const response = await fetch( input: `${API_URL}api/files/download?id=${file._id}`, config)
    if (response.status === 200) {
        const blob = await response.blob()
        const downloadUrl = window.URL.createObjectURL(blob)
        const link = document.createElement( tagName: 'a')
        link.href = downloadUrl
        link.download = file.name
        document.body.appendChild(link)
        link.click()
        link.remove()
    }
}

export function deleteFile(file) {
    return async dispatch => {
        try {
            const response = await axios.delete( url: `${API_URL}api/files?id=${file._id}`, config)
            dispatch(deleteFileAction(file._id))
            alert(response.data.message)
        } catch (e) {
            alert(e?.response?.data?.message)
        }
    }
}
```

Рисунок 4.10 – Відображення функцій donwloadFile та deleteFile

Для завантажування із серверу файлів використовується метод fetch, а для видалення бібліотека axios. В функції завантаження проходе взаємодія з DOM для формування файлу для завантаження із серверу, а в функції видалення тільки передається конкретний файл для видалення через редюсер.

На екрані можна побачити функціонал завантаження та видалення файлів тільки при наведенні на певний елемент файлу у режимі «список». Після чого виводиться дві кнопки для видалення та завантаження, а статистика файлу зміщується вліво (рис. 4.11).

Рисунок 4.10 – Відображення функціоналу для завантаження та видалення файлу.

При натисненні кнопки для завантаження файлу виконується обробка та додавання файлу до списку завантаження в браузері, а при видаленні зникає файл із списку завантажений файлів.

4.2 Тестування та аналіз результатів тестування

Функція завантаження файлу на сервер

Функція завантаження на сервер файлу надає можливість завантажити необхідний файл на сервер за допомогою особливості drag-and-drop або за допомогою стандартного пошуку файлу на локальному комп'ютері.

Вхідною інформацією є обраний файл користувачем, а вихідною – завантажений файл на сервері та подальше його відображення в списку завантажений файлів.

Для завантаження файлу на сервер потрібно мати файл, який має допустимий тип та розмір й відсутність вірусів в ньому. Крім цього потрібно мати стабільне підключення до мережі Інтернет (рис. 4.11).

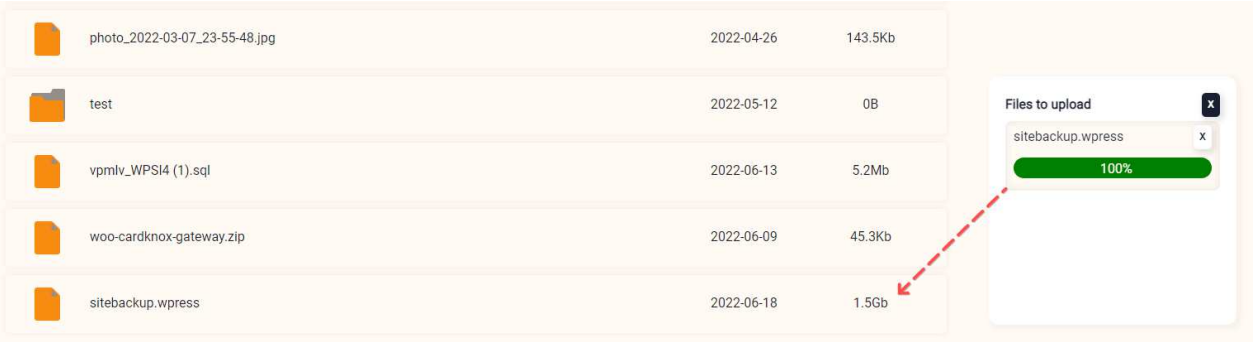

Рисунок 4.11 – Відображення функціоналу завантаження файлу на сервер

На відображеному функціоналі завантаження файлу можна побачити, що завантажено файл розміром в 1.5 Гб, що свідчить про пропускну здатність сервера на завантаження великих за розміром файлів.

Функція завантаження файлу

Функція завантаження файлів із серверу надає можливість завантажувати файли раніше завантажених із серверу файлів з локального комп'ютера.

Вхідна інформація – файл на сервері, а вихідна – файл на локальному комп'ютері.

Вимогою є стабільне підключення до мережі Інтернет та наявність вільного місця на локальному комп'ютері. Користувач має можливість завантажувати файл будь-якого типу та розміру, якщо в нього є достатньо місця на диску для певного файлу (рис. 4.12).

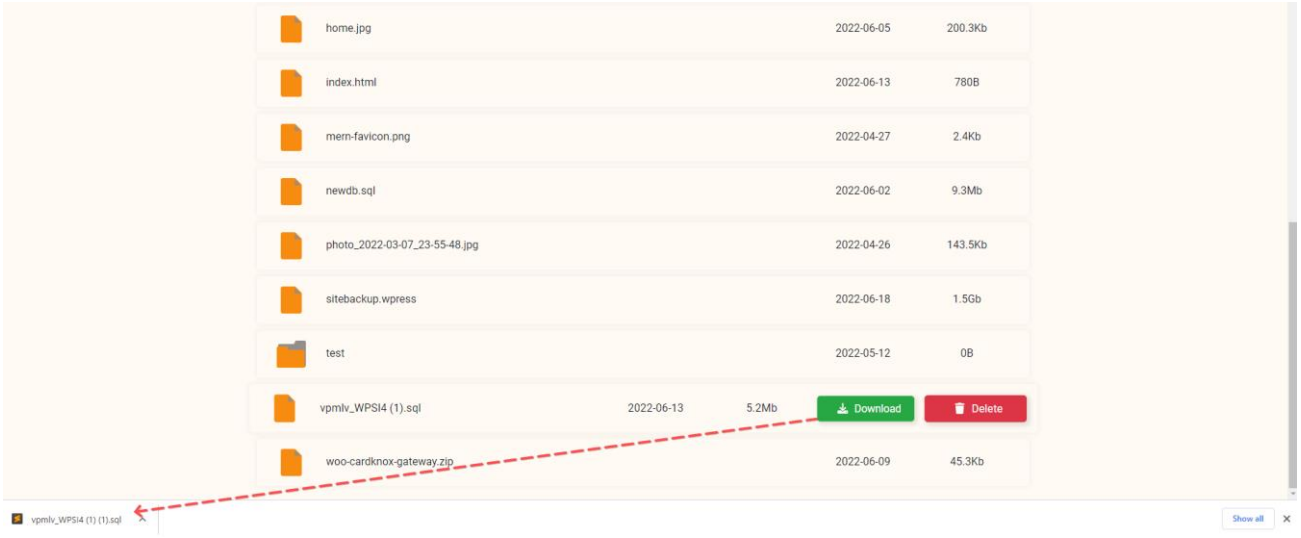

Рисунок 4.12 – Відображення завантаженого файлу із серверу

Після натиснення кнопки завантажити, файл потрапляє до списку завантажених файлів на локальному комп'ютері. Швидкість завантаження залежить від швидкості Інтернету користувача системи.

Функція зміни паролю

Кожен користувач системи має можливість змінити пароль через налаштування в профілю користувача. Для встановлення пароль, його розмір повинен складати не менше 3-ох символів. Щоб змінити пароль, користувачу

потрібно ввести старий пароль в перше поле, в друге – новий пароль та в трете повторно ввести новий пароль. При цьому новий пароль не повинен співпадати із старим паролем та містити необхідну к-сть символів (рис. 4.13).

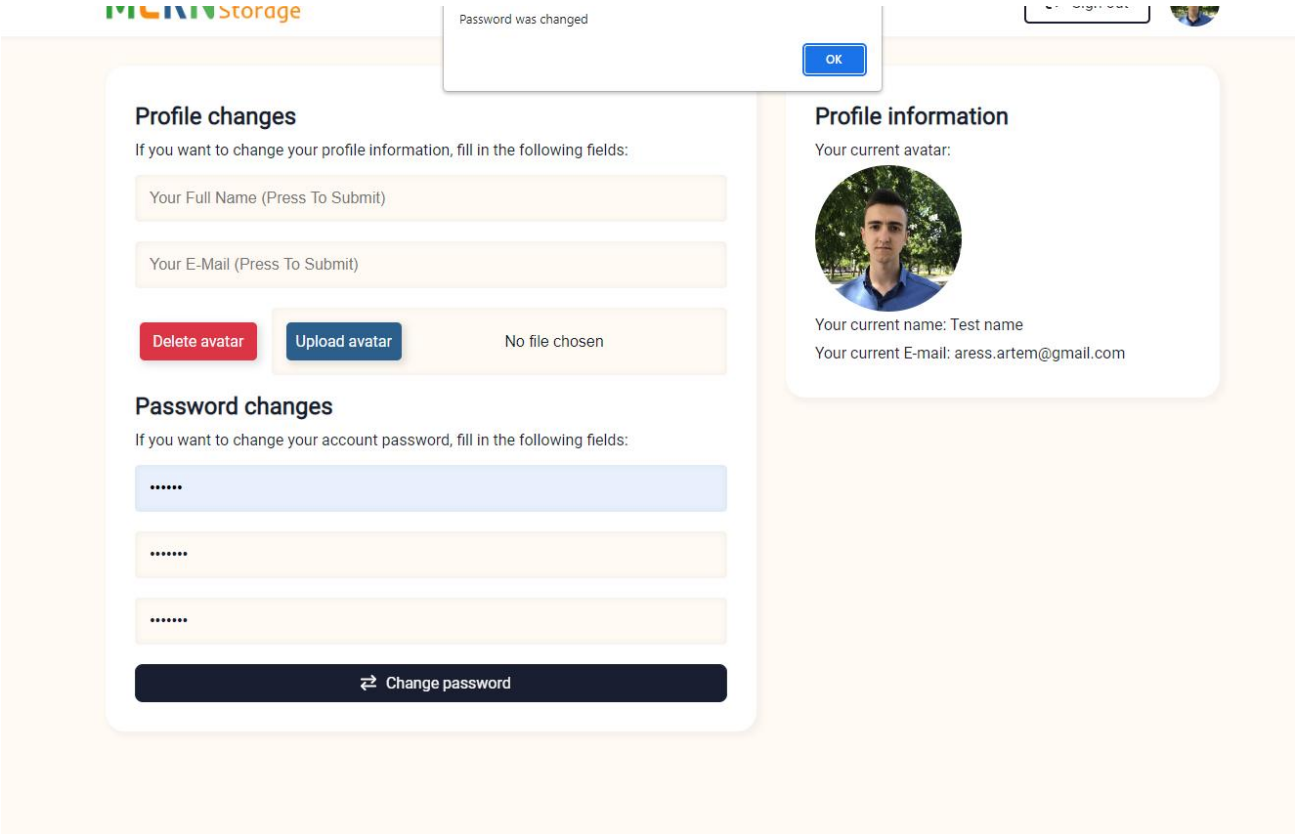

Рисунок 4.13 – Відображення функціоналу зміни паролю

Після успішної зміни паролю, користувач отримує повідомлення «Password was changed». Після цього при наступній авторизації потрібно буде ввести новий пароль. Крім цього JWT, що зберігає дані користувача оновлюються і підв’язується конкретно до того користувача, що змінював пароль в системі.

Адаптивність вебзастосунку

Вебзастосунок є адаптивним для таких пристроїв: PC, laptop, netbook, iPad Air, iPad Mini, iPhone 6+ та інші. Мобільна версія вебзастосунку не скриває основних функцій ПК версії сайту. Весь функціонал зберігається та працює при використанні сенсорного екрану. Також підтримується режим перевернутого екрану. Змінюються тільки деякі розміри елементів, таких як: картинки, текст, кнопки, списки (рис. 4.14).

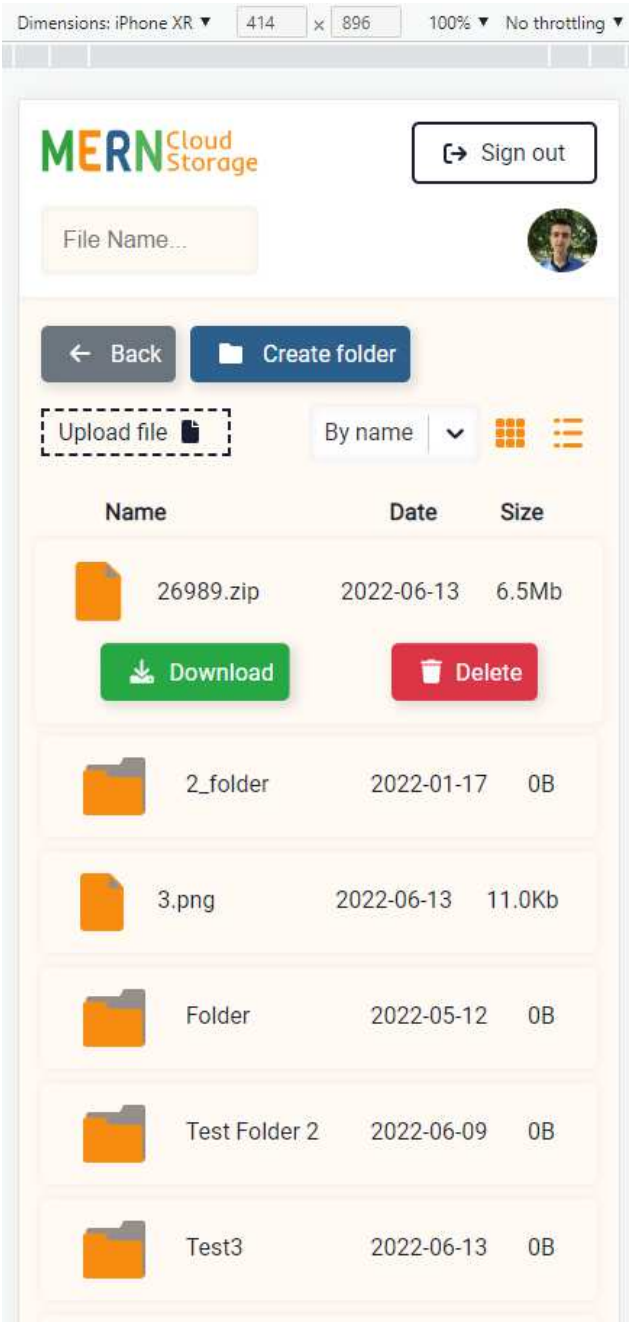

Рисунок 4.14 – Відображення адаптивного режиму вебзастосунку головної сторінки.

Кросбраузерність вебзастосунку

Крім адаптивності, вебзастосунок підтримує відображення та працездатність в різних браузерах, а саме в основних: Chrome, Opera, Safari, Internet Explorer, Mozilla Firefox. Кросбраузерність реалізована за рахунок використання новітніх технологій для створення вебзастосунків а також вендорних префіксів, а саме:

1. -o- для браузера Опера

2. -moz- для браузера Mozilla
3. -ms- для Internet Explorer
4. -webkit- для браузерів Safari та Chrome

Висновки до розділу 4

Отже, в розділі представлена робота з кодування програмного забезпечення, а також аналізу компонентів головної сторінки вебзастосунку. Розглянуто наступні компоненти системи: створення папок, завантаження на сервер файлів, завантаження із серверу та видалення файлів та папок. В процесі аналізу компонентів також описане керівництво користувача для взаємодії з ними. Описано функцію підрахунки розміру файлу, що взаємодіє з БД та в компоненті виводить на екран результат підрахунку. Також розглянуто роботу Redux для взаємодії серверу з клієнтом. Проведено тестування та аналіз результатів тестування. Розглянуто наступні функції: завантаження файлу на сервер, завантаження файлу із серверу, зміна паролю. Виконане тестування з адаптивності та кросбраузерності вебзастосунку.

ВИСНОВКИ

Результатом кваліфікаційної роботи є удосконалена система хмарного сховища за рахунок розробки вебзастосунку хмарного сховища. Для досягнення визначеної мети вирішено такі завдання:

– проаналізовано систему, що розробляється та досліджено існуючи програмні рішення: користувачів системи, структуру системи, сценарії роботи системи, засоби апаратної та програмної реалізації;

– визначено архітектуру для проєктування програмного забезпечення;

– змодельовано програмне забезпечення: діаграма прецедентів, розгортання, ERD, діяльності, компонентів;

– розроблено front-end частину вебзастосунку на базі технологій: React та Redux;

– розроблено back-end частину вебзастосунку на базі технологій: Node.js, Express.js, MongoDB.

Також проаналізовані загальні відомості про створений вебзастосунок, виявлено переваги та недоліки системи, що розроблялася. Створено та описано специфікації вимог до створеного програмного забезпечення.

Виконано огляд моделі СХС, тобто визначено архітектурну ідею вебзастосунку та функціонал, що до нього входить. Проаналізовано використання JWT в системі та наведена схема роботи JWT. Проведено огляду паттерну MVC, як архітектурне рішення для побудови СХС.

Здійснено огляд стеку MERN. Оцінено кожну технологію в проєкті, її характеристики та переваги для побудови системи. Описані інтерфейси користувачів згідно компонентів ПЗ.

Виконана робота з аналізу та тестування програмного забезпечення. Підготовлено керівництво користувача та представлена виконана робота з кодування.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Застосунок аналог: Google disk. URL: https://drive.google.com/ (дата звернення: 28.04.2022).

2. Застосунок аналог: ICloud. URL: https://www.icloud.com/ (дата звернення: 28.04.2022).

3. JSON Web Token. URL: https://uk.wikipedia.org/wiki/JSON_Web_Token (дата звернення: 28.04.2022).

4. Як використовувати JSON Web Tokens (JWT) для автентифікації. URL: https://codeguida.com/post/1567 (дата звернення: 28.04.2022).

5. Що таке діаграма класів? - визначення з техопедії - Розвиток – 2022. URL: https://uk.theastrologypage.com/class-diagram (дата звернення: 28.04.2022).

6. Діаграма класів. URL: https://uk.education-wiki.com/7917237-class-diagram (дата звернення: 28.04.2022).

7. JavaScript ES6. W3Schools Online Web Tutorials. URL: https://www.w3schools.com/js/js_es6.asp (дата звернення: 28.04.2022).

8. Node.js Tutorial. W3Schools Online Web Tutorials. URL: https://www.w3schools.com/nodejs/default.asp (дата звернення: 28.04.2022).

9. Функціональне програмування в JavaScript. Travels & Code. URL: https://travelscode.com/funktsionalne-programuvannya-v-javascript/ (дата звернення: 28.04.2022).

10. MERN Stack - GeeksforGeeks. GeeksforGeeks. URL: https://www.geeksforgeeks.org/mern-stack/ (дата звернення: 28.04.2022).

11. What Is The MERN Stack? Introduction & Examples. MongoDB. URL: https://www.mongodb.com/mern-stack (дата звернення: 28.04.2022).

12. MongoDB Documentation. URL: https://www.mongodb.com/docs/ (дата звернення: 28.04.2022).

13. Документація Node.js. URL: https://nodejs.org/uk/docs/ (дата звернення: 28.04.2022).

14. Express. URL: https://expressjs.com/ (дата звернення: 28.04.2022).

15. Getting Started. URL: https://reactjs.org/docs/getting-started.html (дата звернення: 28.04.2022).

16. Banks A., Porcello E. Learning React: Functional Web Development with React and Redux. O'Reilly Media, 2017. 350 p.

17. Stefanov S. React : up & running: Building web applications. 2016. 201 p.

18. Haverbeke M. Eloquent JavaScript: A Modern Introduction to Programming. No Starch Press, 2014. 472 p.

19. Elliott E. Programming JavaScript Applications: Robust Web Architecture with Node, HTML5, and Modern JS Libraries. O'Reilly Media, 2014. 254 p.

20. Syed B. A. Beginning Node.js. Berkeley, CA : Apress, 2014. URL: https://doi.org/10.1007/978-1-4842-0187-9 (дата звернення: 28.04.2022).

21. Mardan A. Pro Express.js: Master Express.js: The Node.js Framework For Your Web Development. Apress, 2014. 372 p.

22. JSON Web Tokens - jwt.io. URL: https://jwt.io/ (дата звернення: 28.04.2022).

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інженерії програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

РОЗРОБКА ХМАРНОГО СХОВИЩА

СПЕЦІАЛЬНА ЧАСТИНА З ОХОРОНИ ПРАЦІ

МІКРОКЛІМАТ У ПРИМІЩЕННЯХ З ВИКОРИСТАННЯМ КОМП'ЮТЕРНОЇ ТЕХНІКИ

Спеціальність «Інженерія програмного забезпечення»
121 – КРБ.1 – 409. 21810917

Студент _____ **А. І. Кузьмін**
підпис
«__» _____ 2022 р.

Консультант ст. викладач _____ **А. О. Алексєєва**
підпис
«__» _____ 2022 р.

Миколаїв – 2022

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

ДНС	–	Державні санітарні норми
ГПа	–	Гектопаскалі
МУ	–	Мікрокліматичні умови
ДСТами	–	Державними стандартами
ЗІЗ	–	Засоби індивідуального захисту

ВСТУП

Робоча діяльність працівників в приміщенні проходить в конкретних мікрокліматичних умовах, враховуючи показники температури повітря, його швидкості переміщення відносно вологості повітря, барометричні показники та інтенсивність теплового випромінювання. Дані показники можуть визначати стан (мікроклімат) середовища, в якому находяться працівники. Якщо робота працівників виконується на відкритій зоні, то метеорологічні умови забезпечуються кліматичним поясом та сезонами року. Проте у випадку роботи в приміщенні, формується певний мікроклімат для працівників, який певним чином впливає на здоров’я та умови праці.

Чимало проблем зі здоров’ям працівників можуть виникнути при ігноруванні показників: температури повітря в приміщенні, відносної вологості повітря, рухливості повітря та теплового випромінювання. Наприклад, проблеми з терморегуляцією в крові, проблеми зі шкірою, диханням та загальним самовідчуттям людини.

Тому, так як людина сильно залежить від мікроклімату приміщення, потрібно будувати та проєктувати приміщення з урахуванням параметрів мікрокліматичних умов. А також облаштовувати приміщення та сам офіс конкретними приладами для забезпечення мікроклімату: обігрівачами, вентиляторами, вентиляцією та зволожувачами повітря. Для того, щоб мати змогу контролювати умови праці та загальний мікроклімат приміщення, необхідно також встановлювати деякі прилади для моніторингу показників: температури, вологості та рухливості повітря.

У зв’язку з цим, доцільним є розглянути у спеціальній частині кваліфікаційної бакалаврської роботи питання вимог до параметрів мікроклімату приміщень, заходи та засоби для попередження небезпеки та негативного впливу на працівника при використанні комп’ютерної техніки.

1 ОСНОВНІ ПАРАМЕТРИ МЕТЕОРОЛОГІЧНИХ УМОВ (МІКРОКЛІМАТУ)

Температура (t, °С) є одним із основних параметрів повітря, що характеризує його тепловий стан (ступінь нагрівання), тобто кінетичну енергію молекулярних рухів повітря.

Вологовміст повітря у виробничому приміщенні оцінюється відносною вологістю (ф, %), тобто відношенням абсолютної вологості до максимально можливої за цією температурою.

Швидкість (рухливість) повітря (V, м/с) оцінюється вектором усередненої швидкості переміщення повітряних потоків (струменів) під дією різних сил, що виникають.

Під атмосферним тиском (Р, мм рт. ст.) розуміють модуль величини, що характеризує інтенсивність сил, зумовлених масою вищого стовпа повітря на одиницю поверхні. Нормальним прийнято вважати тиск, що дорівнює 1013,25 ГПа (760 мм рт. ст.). Для перерахування в гектопаскалі тиску, вираженого в мм рт. ст., користуються наступним співвідношенням: Р, ДПа = 4/3Р, мм рт. ст. Інші чинники мікроклімату: тиск повітря, концентрація кисню повітря, ступінь іонізації повітря, і навіть температура.

Людина під час роботи витрачає енергію, що її накопичив організм за рахунок харчування. Інтенсивність витрат енергії залежить від характеру та інтенсивності праці, а також від параметрів навколишнього середовища та, в першу чергу, від стану повітря в приміщенні. Стан повітря робочої зони у виробничому приміщенні називають мікрокліматом або метеорологічними умовами.

2 НОРМАТИВНЕ РЕГУЛЮВАННЯ МІКРОКЛІМАТУ ВИРОБНИЧИХ ПРИМІЩЕНЬ

Вимоги до мікроклімату виробничих приміщень регулюють ДСН 3.3.6.042-99 [1] «Санітарні норми мікроклімату виробничих приміщень», затверджені постановою Головного державного санітарного лікаря України від 01.12.1999 № 42. Вони ж залишаються чинними у 2021 році. За цим документом параметри мікроклімату можуть оцінювати як оптимальні, допустимі або такі, що не відповідають санітарним нормам. Державні санітарні норми мікроклімату виробничих приміщень поширюються на умови мікроклімату в межах робочої зони – визначеного простору, де розташовані робочі місця постійного або непостійного (тимчасового) перебування працівників.

Робоча зона – простір, в якому знаходяться робочі місця постійного або непостійного (тимчасового) перебування працівників.

Робоче місце – місце постійного або тимчасового перебування працюючого в процесі трудової діяльності.

3 ВИМОГИ ДО МІКРОКЛІМАТУ ТА СТАНУ ПОВІТРЯ РОБОЧОЇ ЗОНИ

МУ виробничих приміщень характеризуються такими показниками:

- температура повітря;
- відносна вологість повітря;
- швидкість руху повітря;
- інтенсивність теплового (інфрачервоного) опромінення;
- температура поверхні.

За ступенем впливу на тепловий стан людини мікрокліматичної умови поділяють на оптимальні та допустимі.

Для робочої зони виробничих приміщень встановлюються оптимальні та допустимі МУ з урахуванням важкості виконуваної роботи та періоду року. При одночасному виконанні в робочій зоні робіт різної категорії важкості рівні показників мікроклімату повинні встановлюватись з урахуванням найбільш чисельної групи працівників.

Мікрокліматичні умови, які в разі тривалого, систематичного впливу на людину забезпечують нормальний тепловий стан організму без напруження і порушення системи терморегуляції. Такі умови створюють відчуття теплового комфорту і забезпечують умови для високого рівня працездатності.

Параметри мікроклімату нормують для постійних та непостійних робочих місць залежно від таких факторів:

- періоду року;
- категорії важкості праці за фізичним навантаженням.

Розрізняють такі періоди року:

1. теплий (середньодобова температура навколишнього повітря становить більше +10 ^{o}C);

2. холодний (середньодобова температура навколишнього повітря становить менше +10 ^{o}C). Категорії важкості робіт залежно від фізичного навантаження наведені в (табл. 1).

Таблиця 3.1 – Категорії важкості праці залежно від енерговитрат

Категорія важкості праці	**Характеристика роботи**	**Енерговитрати, Дж/с (Вт)**
Легка фізична робота : Іа Іб	Виконується сидячи і не потребує систематичного фізичного напруження (Іа). Виконується сидячи, стоячи чи пов'язана з ходьбою та супроводжується певним фізичним напруженням (Іб).	До 140 141-175
Фізична робота середньої важкості: ІІ-а, ІІ-б	Пов'язана з ходьбою, і переміщенням дрібних (до 1 кг) предметів у положенні сидячи або стоячи і потребує певного фізичного навантаження (ІІ-а). Виконуються стоячи, пов'язані з ходінням, переміщенням невеликих вантажів (до 10 кг) та супроводжуються помірнім фізичним навантаженням (ІІ-б).	176-232 233-290
Важка фізична робота ІІІ	Пов'язана з постійним переміщенням, перенесенням великих вантажів (понад 10 кг), які потребують значних фізичних зусиль.	291-349

Оптимальні умови мікроклімату встановлюють для постійних робочих місць, де працівники проводять свій повний робочий день. За таких умов показники температури повітря в робочій зоні протягом робочої зміни не повинні виходити за межі нормованих величин оптимальної температури для даної категорії важкості праці. Інакше, це може вплинути на здоров'я та самопочуття працівника. Температура внутрішніх поверхонь приміщення (стін, підлоги, стелі), зовнішніх поверхонь технологічного устаткування, огороджувальних конструкцій не повинна виходити більш ніж на 2 ^{o}C за межі оптимальних величин температури повітря для даної категорії важкості праці. Допустимі величини мікрокліматичних умов встановлюють у випадках, коли на робочих місцях не можна забезпечити оптимальні величини мікроклімату внаслідок технологічних вимог виробництва або технічною недосяжності.

Величини показників, які характеризують допустимі МУ, встановлюють для постійних і непостійних робочих місць.

Оптимальні величини параметрів мікроклімату в робочій зоні засклених огороджень не повинна перевищувати 35,0 Вт/м ^{2} – у разі опромінення 50% та більше поверхні тіла, 70 Вт/м^2 – за величини опромінюваної поверхні тіла від 25 до 50%, та 100 Вт/м ^{2} – у випадку опромінення не більше 25% поверхні тіла працівника [2].

У разі наявності відкритих джерел опромінення (нагрітий метал, скло, відкрите полум'я) допустима інтенсивність опромінення складає 140,0 Вт/м ^{2}. Величина опромінюваної площі не повинна перевищувати 25 % поверхні тіла працівника за умови обов'язкового використання індивідуальних засобів захисту (спецодягу, окулярів, щитків) (табл. 2).

Таблиця 3.2 – Оптимальні величини параметрів мікроклімату

Період року	**Категорія робіт**	**Температура повітря, ^{o}C**	**Відносна вологість, %**	**Швидкість руху повітря, м/с**
Холодний період року	Легка І-а	22 - 24	40 - 60	0,1
	Легка І-б	21-23	40 - 60	0,1
	Середньої важкості ІІ-а	19-21	40 - 60	0,2
	Середньої важкості ІІ-б	17-19	40 - 60	0,2
	Важка ІІІ	16-18	40 - 60	0,3
Теплий період року	Легка І-а	23-25	40 - 60	0,1
	Легка І-б	22-24	40 - 60	0,2
	Середньої важкості ІІ-а	21-23	40 - 60	0,3
	Середньої важкості ІІ-б	20-22	40 - 60	0,3
	Важка ІІІ	18-20	40 - 60	0,4

4 ВИМОГИ ДО ЗАСОБІВ НОРМАЛІЗАЦІЇ МІКРОКЛІМАТУ ТА ТЕПЛОЗАХИСТУ

При наявності джерел тепловипромінювання вживають комплекс заходів з теплоізоляції устаткування та нагрітих поверхонь за допомогою теплозахисного обладнання.

В залежності від принципу дії теплозахисні засоби поділяються на:

1. тепловідбивні – металеві листи (сталь, залізо, алюміній, цинк, поліровані або покриті білою фарбою тощо) одинарні або подвійні; загартоване скло з плівковим покриттям; металізовані тканини; склотканини; плівковий матеріал та ін.;

2. тепловбираючі – сталеві або алюмінієві листи або коробки з теплоізоляцією з азбестового картону, шамотної цегли, повсті, вермикулітових плит та ін. теплоізоляторами; сталева сітка (одинарна або подвійна з загартованим силікатним склом); загартоване силікатне органічне скло та ін.;

3. тепловідвідні – екрани водоохолоджувальні (з металевого листа або сітки з водою, що стікає), водяні завіси та ін.;

4. комбіновані.

В залежності від особливостей технологічних процесів застосовують прозорі, напівпрозорі екрани. Вибір теплозахисних засобів обумовлюється інтенсивністю та спектральним складом випромінювання, а також умовами технологічного процесу.

Теплозахисні екрани повинні забезпечувати нормовані величини опромінення робочих; бути зручними в експлуатації; не ускладнювати огляд, чищення та змазування агрегатів; гарантувати безпечну роботу з ним; мати міцність, легкість виготовлення та монтажу; мати достатньо тривалий термін експлуатації; у процесі експлуатації зберігати ефективні теплозахисні якості.

При неможливості технічними засобами забезпечити допустимі гігієнічні нормативи опромінення на робочих місцях використовуються засоби

індивідуального захисту (ЗІЗ) – спецодяг, спецвзуття, ЗІЗ для захисту голови, очей, обличчя, рук.

В залежності від призначення передбачаються такі ЗІЗ:

– для постійної роботи в гарячих цехах – спецодяг (костюм чоловічий повстяний), а при ремонті гарячих печей та агрегатів – автономна система індивідуального охолодження в комплексі з повстяним костюмом;

– при аварійних роботах – тепловідбиваючий комплект з металізованої тканини;

– для захисту ніг від теплового випромінення, іскор і бризок розплавленого металу, контакту з нагрітими поверхнями – взуття шкіряне спеціальне для працюючих в гарячих цехах;

– для захисту рук від опіків – вачеги, рукавиці суконні, брезентові, комбіновані з надолонниками з шкіри та спилку;

– для захисту голови від теплових опромінень, іскор та бризок металу – повстяний капелюх, захисна каска з підшоломником, каски текстолітові або з полікарбонату;

– для захисту очей та обличчя – щиток теплозахисний сталевара, з приладнаними для нього захисними окулярами із світлофільтрами, маски захисні з прозорим екраном, окуляри захисні, козиркові з світлофільтрами.

У виробничих приміщеннях, в яких на робочих місцях неможливо встановити регламентовані інтенсивності теплового опромінення працюючих через технологічні вимоги, технічну недосяжність або економічно обґрунтовану недоцільність, використовуються обдування, душування, водоповітряне душування та ін.

При тепловому опроміненні, що перевищує 350 Вт/кв. м., доцільно застосовувати повітряне душування робочих місць (табл. 4.1) [1].

Таблиця 4.1 – Температура та швидкість руху повітря при повітряному душуванні

Категорія робіт	**Температура повітря в робочій зоні, град. С**	**Швидкість руху повітря, м/сек.**	**Температура повітря в струмені, що душує (гр. С) при інтенсивності інфрачервоного опромінення, Вт/кв. м**				
			350	700	1400	2100	2800
Легка Іа, Іб	до 28	1	28	24	21	16	-
		2	-	28	26	24	20
		3	-	-	28	26	24
		3,5	-	-	21	27	25
Середньої важкості IIа, IIб	до 27	1	27	22	-	-	-
		2	28	24	21	16	-
		3	-	27	24	21	18
		3,5	-	28	25	22	19
Важка	до 26	2	25	19	16	-	-
		3	26	22	20	18	17
		3,5	-	23	22	20	19

5 ВИМОГИ ДО МЕТОДІВ ВИМІРЮВАННЯ ПАРАМЕТРІВ МІКРОКЛІМАТУ ТА ЇХ ОЦІНКИ

Вимірювання параметрів мікроклімату проводяться на робочих місцях і в робочій зоні на початку, в середині та в кінці робочої зміни. При коливаннях мікрокліматичних умов, пов'язаних з технологічним процесом та іншими причинами, вимірювання проводяться з урахуванням найбільших і найменших величин термічних навантажень протягом робочої зміни.

Вимірювання здійснюються не менше 2-х разів на рік (теплий та холодний періоди року) у порядку поточного санітарного нагляду, а також при прийманні до експлуатації нового технологічного устаткування, внесенні технічних змін в конструкцію діючого устаткування, організації нових робочих місць тощо.

При проведенні вимірювання в холодний період року температура зовнішнього повітря не повинна бути вищою за середню розрахункову температуру, в теплий період - не нижчою за середню розрахункову температуру, що приймається для опалення та кондиціонування за оптимальними та допустимими параметрами (табл. 4).

Таблиця 4 – Вимоги до вимірювальних приладів

Вимірювальні величини	**Діапазон вимірювання**	**Допустима похибка**	**Рекомендовані прилади**
1. Температура повітря, град. С	-30 до +5	+-0,1	Аспіраційний психрометр із ртутними термометрами
2. Відносна вологість повітря, %	15 до 100	+-5,0	Ті ж самі та записуючі гігрографи
3. Температура поверхні, град. С	-30 до 100	+-1,0	Електротермометри, термопари та ін.
4. Швидкість руху повітря, м/сек.	0,1 - 0,5 до 0,6 - 5,0	+-0,1 +-0,2	Анемометри ротаційної дії
5. Інтенсивність інфрачервоного опромінення	10,0 - 20000,0	+-10%	Актинометри, термостовбці, радіометри зі спектральною чутливістю

6 ЗАХОДИ ТА ЗАСОБИ ПОПЕРЕДЖЕННЯ ЗАБРУДНЕННЯ ПОВІТРЯ РОБОЧОЇ ЗОНИ

Вентиляція – видалення повітря з приміщення і заміна його свіжим, в необхідних випадках, обробленим повітрям. Вентиляція створює умови повітряного середовища, сприятливі для здоров'я і самопочуття людини, що відповідають вимогам технологічного процесу.

За способом переміщення повітря вентиляція поділяється на два види: природну та механічну.

За способом організації повітрообміну вентиляція може бути: місцевою та загальнообмінною.

За принципом дії вентиляційне устаткування поділяється на:

1. витяжне (загальне та місцеве);
2. припливне воно буває місцеве (повітряні душові ванни, оазиси, завіси) і загальне.

Кондиціонування повітря – це створення і автоматична підтримка у приміщеннях незалежно від зовнішніх умов постійних або змінних за відповідною програмою параметрів мікроклімату, які найбільш придатні для людини та нормального проходження технологічного процесу.

Основний ***принцип повітряного балансу*** будівлі полягає в тому, що обсяг що надходить в будівлю повітря зовні повинен відповідати обсягу що виходить з нього. В ідеалі обсяг зовнішнього повітря, подається через системи вентиляції та кондиціонування будівлі, повинен перевищувати обсяг вихідного повітря, щоб забезпечити деякий надлишковий тиск усередині будівлі. Це запобігає неконтрольовану інфільтрацію зовнішнього повітря на входах і виходах.

Кратність повітрообміну – це показник, який показує, скільки разів протягом години змінюється повітря в приміщенні.

ВИСНОВКИ

Під час виконання спеціальної частини з охорони праці було проаналізовано умови праці працівника в приміщенні в конкретних мікрокліматичних умовах враховуючи наступні показники: температуру повітря, відносну вологість повітря, швидкість руху повітря, інтенсивність теплового (інфрачервоного) опромінення, температуру поверхні.

Також проаналізовано доцільність розгляду основних параметрів та вимог до приміщень, заходів та засобів для попередження при використанням комп’ютерної техніки.

Наведені правила по нормативному регулювання мікроклімату виробничих приміщень за ДСН, які залишаються чинними до цього часу. Також описані вимоги до мікроклімату та стану повітря робочої зони, засобів нормалізації мікроклімату та теплозахисту, методів вимірювання параметрів мікроклімату та їх оцінки. Також описані заходи та засоби попередження забруднення повітря робочої зони.

До кожного параметру мікроклімату було наведено таблицю з оптимальними величинами, які мають підтримуватися в межах умов праці та мають бути відстеженні за допомогою спеціальних пристроїв та засобів попередження перевищення нормативних значень умов праці. Дотримання зафіксованих вимог до умов праці дозволить уникнути негативні наслідки, що впливають на здоров’я та самопочуття людини.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Санітарні норми мікроклімату виробничих приміщень ДСН 3.3.6.042-99. URL: https://zakon.rada.gov.ua/rada/show/va042282-99#Text (дата звернення: 02.06.2022).

2. Санітарні норми мікроклімату виробничих приміщень. URL: https://www.sop.com.ua/article/82-osnovn-vimogi-ta-zahodi-z-normalzats-mkroklmatichnih-umov-na-robochih-mstsyah (дата звернення: 02.06.2022).

3. ДСН 3.3.6.042-99. Санітарні норми мікроклімату виробничих приміщень (34094). URL: https://dnaop.com/html/34094/doc-%D0%94%D0%A1%D0%9D_3.3.6.042-99 (дата звернення: 02.06.2022).

4. Гігієна праці та експертиза умов праці. URL: http://surl.li/ccrzw (дата звернення: 02.06.2022).

5. Мікроклімат виробничих приміщень та його нормування URL: https://studfile.net/preview/2425101/page:4/#7 (дата звернення: 02.06.2022).

6. Повітря робочої зони. URL: https://pidru4niki.com/92787/bzhd/povitrya_robochoyi_zoni (дата звернення: 02.06.2022).

7. ДБН В.2.5-67:2013. Опалення, вентиляція та кондиціонування (32609) URL: https://dnaop.com/html/32609_26.html (дата звернення: 02.06.2022).

8. Державні санітарні правила "Підприємства чорної металургії" (32609) URL: https://ips.ligazakon.net/document/MOZ8523 (дата звернення: 02.06.2022).

9. Гандзюк М. П., Желібо Е. П., Халімовський М. О. Основи охорони праці / За ред.. Гандзюка М. П. - К.: Каравела 2003 - 405 с.

10. ДСТУ 2293-99 Охорона праці. Терміни та визначення основних понять. Київ -1999 р.