**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ**

**КАФЕДРА КОМП'ЮТЕРНИХ НАУК**

# КОМПЛЕКСНА КВАЛІФІКАЦІЙНА МАГІСТЕРСЬКА РОБОТА

**на тему:**

## «Веб-сервіс зберігання та обміну файлами. Клієнтська частина»

**Завідувач випускаючої кафедри** — **Довбиш А.С.**

**Керівник роботи** — **Ободяк В.К.**

**Студента групи ІН.м – 92** — **Каліна В.В.**

**СУМИ 2020**

*Сумський державний університет*
(назва вузу)

Факультет *ЕлІТ* Кафедра *Комп'ютерних наук*
Спеціальність *«Інформатика»*

Затверджую:
зав.кафедрою _____
_____
"_____" _____ 20___р.

# ЗАВДАННЯ
# НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ) СТУДЕНТОВІ

*Каліні Воводимиру Вікторовичу*
(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) *Веб-сервіс зберігання та обміну файлами. Клієнтська частина*
затверджую наказом по інституту від "____" _____ 20 ___ р. № _____

2. Термін здачі студентом закінченого проекту (роботи) _____
3. Вхідні данні до проекту (роботи) _____
_____
_____

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити)

1) *Вивчити процеси хмарного зберігання файлів.*
2) *Провести аналіз аналогів. Постановка задачі та методи дослідження.*
3) *Визначити задачі сервісу. Моделювання та проектування.*
4) *Програмна реалізація та тестування. Створити зручний інтерфейс для користувача.*

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) _____
_____
_____
_____
_____
_____

6. Консультанти до проекту (роботи), із значенням розділів проекту, що стосується їх

| Розділ | Консультант | Підпис, дата | |
|---|---|---|---|
| | | Завдання видав | Завдання прийняв |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

7. Дата видачі завдання _____

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

## КАЛЕНДАРНИЙ ПЛАН

| № п/п | Назва етапів дипломного проекту (роботи) | Термін виконання проекту (роботи) | Примітка |
|---|---|---|---|
| *1.* | *Вивчити процеси хмарного зберігання файлів.* | | |
| *2.* | *Провести аналіз аналогів. Постановка задачі та методи дослідження.* | | |
| *3.* | *Визначити задачі сервісу. Моделювання та проектування.* | | |
| *4.* | *Програмна реалізація та тестування. Створити зручний інтерфейс для користувача.* | | |

Студент – дипломник _____
(підпис)

Керівник проекту _____
(підпис)

# РЕФЕРАТ

**Записка:** 52 стор., 19 рис., 6 табл., 1 діагр., 1 додаток, 12 джерел.

**Об’єкт дослідження** — збереження інформаційних даних в хмарному сховищі.

**Мета роботи** — науково дослідити процеси збереження інформації в хмарних сховищах (в інтернет), розробка аналогічного Веб-додатка та розробити зручний інтерфейс.

**Методи дослідження** — метод функціонально-статистичних випробувань.

**Результати** — розроблено Веб-додаток хмарного сховища за допомогою сучасних фреймворків. Досліджено і застосовано фреймворки: Bootstrap, Vue.js, Vuetify для створення клієнтської частини; а Laravel, Node.js, Express.js для створення серверної частини.

МЕТОД ФУНКЦІОНАЛЬНО-СТАТИСТИЧНИХ ВИПРОБУВАНЬ, ХВАРНЕ СХОВИЩЕ, ЗБЕРЕЖЕННЯ ДАНИХ, ВЕБ-ДОДАТОК, ФРЕЙМВОРК, JAVASCRIPPT, PHP.

# ЗМІСТ

# ВСТУП

З розвитком технологій і збільшенням кількості носіїв інформації виникає питання зберігання та обміну даними між гаджетами та електронними носіями. Також з огляду на останні новини та пандемію виникає гостра необхідність в надійному зберіганні і швидкому зручному обміні великими обсягами найрізноманітніших даних між користувачами мережі, що знаходяться в самих різних куточках нашої планети.

Одним з найбільш оптимальних, ефективних, зручних і тому затребуваних рішень стали так звані хмарні сервіси для зберігання даних, що дозволяють зберігати, оперативно редагувати і швидко передавати значні обсяги найрізноманітніших файлів. Причому доступ до тих або інших даних можуть одночасно мати багато користувачів, або ж інформація може носити конфіденційних характер і бути призначеною тільки для особистого користування.

Виходячи із визначеної актуальності проблеми зберігання файлів, було визначено, що потрібно створити сайт для того, щоб кожен бажаючий міг з легкістю звантажити файли на сервіс і без проблем мати доступ до них. Для реалізації поставленої мети потрібно вирішити такі задачі:

- Вивчити процеси хмарного зберігання файлів.
- Провести аналіз аналогів.
- Визначити задачі сервісу.
- Створити зручний інтерфейс для користувача.

# 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

## 1.1 Дослідження актуальності проблеми

Проблема своєчасного отримання інформації в бізнесі існувала завжди. Сучасні бухгалтерські програми є потужними і багатофункціональні. Передача інформації таким чином є засобом для операторів ПК і автоматичного введення і виведення даних.

Нижче наведені найбільш актуальні та типові аспекти проблем обміну даними:

- обмін інформацією між географічно віддаленими місцями;

- обмін даними між системами бухгалтерського обліку різного призначення (бухгалтерський, оперативний, управлінський);

Одержування зведених балансів (зведених балансів дочірніх компаній) з різних інформаційних баз;

Загальні вимоги до системи обміну даними полягають в забезпеченні єдиного введення інформації, використовуваної в декількох базах даних, дотриманні загальних правил цілісності бази даних, здатності системи до збоїв і запобігання несанкціонованого доступу.

Для задоволення цих вимог надаються такі послуги, як обмін файлами або хмарне сховище.

Хмарне сховище - це модель хмарних обчислень, яка зберігає дані в Інтернеті через постачальника обчислювальних ресурсів, який використовує сховище даних як послугу і забезпечує управління файлами і обмін ними. Це забезпечує гнучкість, глобальну масштабованість і надійність. Дані доступні в будь-який час і в будь-якому місці [1].

Як правило, хмарне сховище - це віртуальний загальний доступ до файлів або сервер даних, до якого можна отримати доступ з будь-якого пристрою, підключеного до мережі і використовуваної хмарної службі.

Обмін файлами - сервіс, який надає користувачам простір для зберігання файлів і доступу до них через Інтернет 24/7. Цей сервіс дозволяє

легко обмінюватися файлами. На спеціальній сторінці файлообмінника (зазвичай на домашній сторінці) користувач завантажує файл на сервер файлообмінника. Файловий менеджер надає користувачеві постійне посилання, яку він може надіслати електронною поштою, в блогах, на форумі або в миттєвому повідомленні. . Натиснувши на посилання, будь-який інший користувач може завантажити вихідний файл [2].

Однак, на відміну від традиційних серверів, що завантажуються в хмарне сховище, інформація розміщується на декількох веб-серверах одночасно, в тому числі на серверах за тисячі миль на іншому кінці земної кулі.

Зберігання даних в хмарі дозволяє фундаментально переосмислити три аспекти його діяльності.

Таблиця 1.1 – Переваги хмарного сховища

| Перевага | Опис |
|---|---|
| Сукупна вартість володіння | За допомогою хмарного сховища не потрібно купувати обладнання, розподіляти ресурси для зберігання чи витрачати місце для зберігання. Ви можете додавати або видаляти ресурси за потреби, щоб швидко змінити продуктивність та термін зберігання. Це економить багато місця. |
| Час для розгортання | Коли група розробників готова розпочати проект, інфраструктура не повинна їх зупиняти. Хмарне сховище дозволяє IT-спеціалістам швидко виділяти необхідний простір для зберігання, коли це потрібно. Тому IT-спеціалісти можуть зосередитись на вирішенні складних проблем додатків, а не на проблемах управління сховищем. |
| Управління інформацією | Централізоване хмарне сховище створює величезні можливості для нових випадків використання. Використовуючи хмарну стратегію управління життєвим |

| Перевага | Опис |
|---|---|
| | циклом, ви можете вирішити важливі завдання управління інформацією, включаючи автоматичне розподіл даних або блокування даних, щоб задовольнити вимоги відповідності [4]. |

Існує також три типи хмарного сховища: об’єктне, файлове та блокове. Кожен із них має переваги, придатні для конкретних випадків використання.

Таблиця 1.2 – Типи хмарних сховищ

| Тип | Опис |
|---|---|
| Об'єктне сховище | Додатки, розроблені в хмарі, зазвичай використовують переваги сховища об'єктів, такі як велика масштабованість та зберігання атрибутів об'єкта у вигляді метаданих. Бібліотеки об’єктів зберігання, такі як Amazon Simple Storage Service (S3), ідеально підходять для розробки сучасних додатків, які вимагають гнучкості та масштабованості з нуля. Крім того, ці сховища можуть використовуватися для імпорту даних із існуючих сховищ для аналізу, резервного копіювання чи архівування. |
| Файлове сховище | Деякі програми потребують доступу до передачі файлів, тому їм потрібна файлова система. Мережеві сервери зберігання даних (NAS) зазвичай підтримують такий тип сховища. |
| Блочне сховище | Інші корпоративні програми, такі як бази даних або системи планування корпоративних ресурсів (ERP), часто вимагають зберігання з низькою затримкою, присвячене кожному вузлу. Бібліотека |

| Тип | Опис |
|---|---|
| | сховищ працює як бібліотека сховищ із прямим підключенням (DAS) або мережа зберігання даних (SAN) [5]. |

Безпека та доступність надзвичайно важливі для забезпечення безпечного зберігання важливих даних компанії. Існує кілька основних вимог при розгляді хмарного зберігання.

Таблиця 1.3 – Вимоги до хмарного сховища

| Тип | Опис |
|---|---|
| Надійність | Даних слід зберігати занадто багато. В ідеалі вони повинні розподілятися між кількома об'єктами та кількома пристроями в кожному об'єкті. Стихійні лиха, людський фактор або механічні несправності не повинні спричинити втрату даних. |
| Доступність | За необхідності всі дані повинні бути доступними, але між виробничими даними та архівами існують відмінності. Ідеальне хмарне сховище дозволяє досягти найкращого поєднання часу збору даних та вартості. |
| Безпека | Усі дані повинні бути зашифровані під час зберігання та передачі. Дозволи та контроль доступу повинні працювати в хмарі так само, як і локальне сховище даних [7]. |

Однак в роботі з «хмарами» справедливості заради можна відзначити і кілька основних мінусів:

Перший і головний – це недостатнє опрацювання питання забезпечення безпеки. Незважаючи на пропаговану політику конфіденційності, до

інформації, що завантажується залишається відкритим доступ співробітників сервісу і його програмного забезпечення.

Другий недолік випливає частково з першого – при зломі сховища або сервісу, наприклад, в результаті хакерської атаки, конфіденційні файли можуть потрапити під загальний доступ або в руки до зловмисників.

Третій недолік пов'язаний з необхідністю очікування повної синхронізації пристрою і хмарного сховища, якщо така опція використовується. У свою чергу, тривалість очікування при зверненні до сервісу залежить від швидкості доступу в мережу. Переривати же процес синхронізації не рекомендується через високу ймовірність виникнення помилок і збоїв.

Отже сфера і можливості використання хмарних сервісів широкі. «Хмари» можуть використовуватися в корпоративних цілях для колективної командної роботи з певною інформацією, оперативного обміну актуальними даними, можуть служити Файлообмінники в особистих цілях для зберігання та обміну персональними даними. Також на відміну від локальних дата центрів хмари мають великий ряд переваг, а також деякі недоліки які було описано раніше.

## 1.2 Аналіз аналогів

Неможливо розробляти нові продукти без попереднього встановлення цілей і завдань і ретельного аналізу ринкового середовища. Проаналізуйте веб-сайти конкурентів, проаналізуйте цільову аудиторію майбутніх веб-сайтів та визначте «сильні сторони» та «слабкі сторони» проекту. Розробка веб-сайтів завжди починається з виконання подібних завдань. Перш ніж почати розробляти завдання, потрібно проаналізувати існуючі аналоги.

Для досягнення цієї мети проводимо дослідження на таких сайтах:

- mega.dp.ua
- www.google.com/drive
- www.dropbox.com
- www.mediafire.com

Одним із ресурсів з подібною тематикою і призначенням являється сайт «Google Диск» призначений для завантаження, зберігання та поширення клієнтських фалів і інформації. Веб-сайт має досить зручний і сучасний інтерфейс, зручний і зрозумілий пошук інтерфейс якого показано на (рис.1.1).

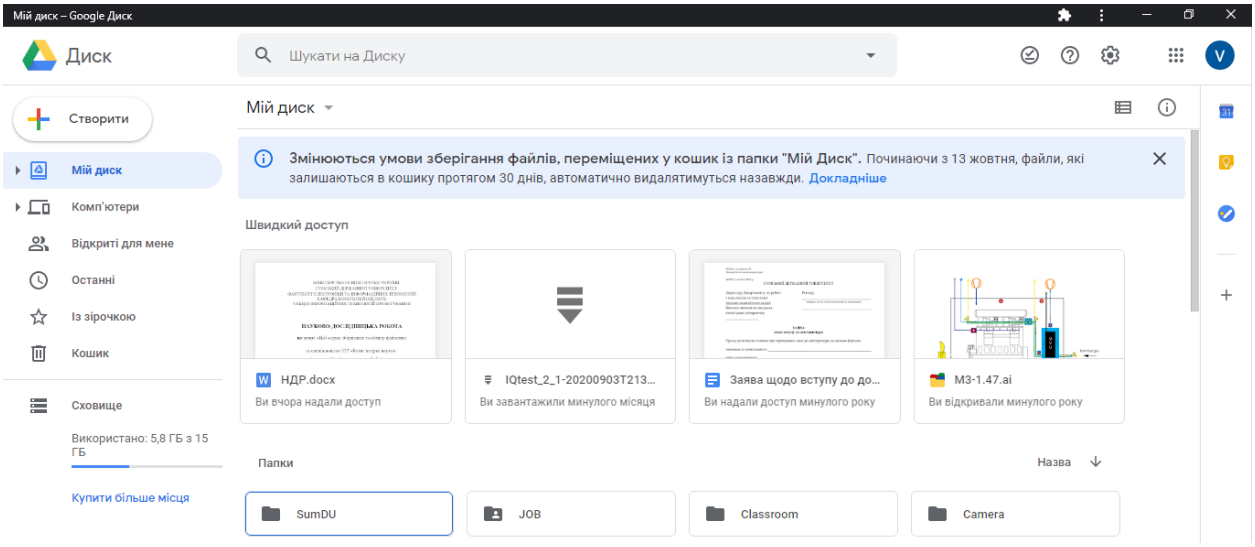


Рисунок 1.1 – головна сторінка Google диску

До переваг даного сайту можна віднести зручний інтерфейс, зручний пошук. Також на даному сайті можна працювати з різними типами файлів прямо на сайті.

Проте представлений сайт має декілька недоліків. До них можна віднести обмежена кількість пам'яті для зберігання щоб отримати великий обсяг пам'яті потрібно заплатити кошти. Також на сайті відсутня можливість підписатися на користувача. Крім того відсутня можливість оцінювати файли.

Наступним сайтом для аналізу було обрано «Mega» (рис.1.2).

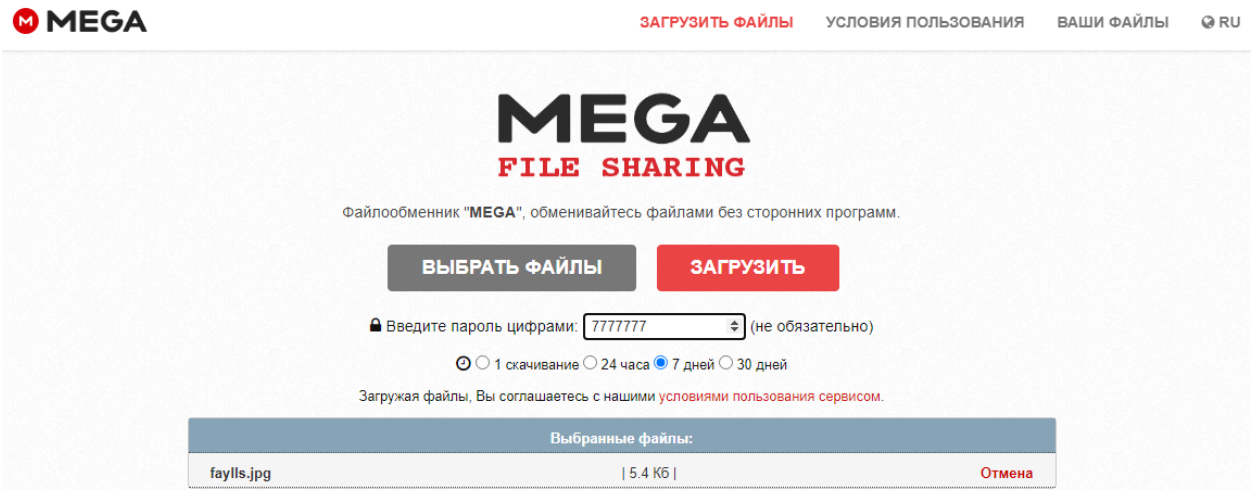


Рисунок 1.2 – головна сторінка «Mega»

Даний сайт має декілька переваг, а саме можливість завантажувати файли до 50GB. Проте представлений сайт має досить велику кількість недоліків. До них можна віднести застарілий дизайн, що є критичним для користувача. Також на сайті відсутня можливість підписатися на користувача, відсутність пошуку популярних публікацій з можливістю оцінки файлу, а також відсутність детальної інформації про файл.

Для аналізу також було обрано сайт «Dropbox» домашня сторінка якого зображена на (рис.1.3).

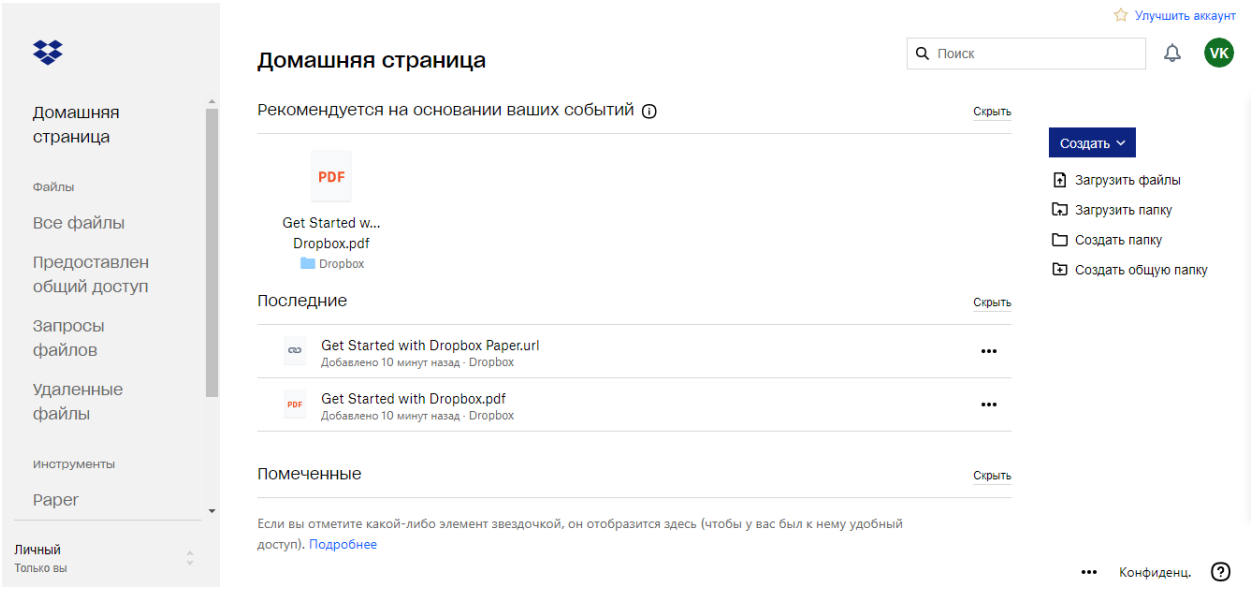


Рисунок 1.3 – домашня сторінка «Dropbox»

Даний сайт вирізняється сучасним дизайном, зручним інтерфейсом та наявність зворотного зв’язку з адміністрацією сайту.

Проте даний сайт також має ряд недоліків. На даному сервісі доступно безкоштовно лише 2GB пам`яті для завантаження та зберігання файлів.

Для аналізу також було обрано сайт mediafire.com головна сторінка якого зображена на (рис.1.4).

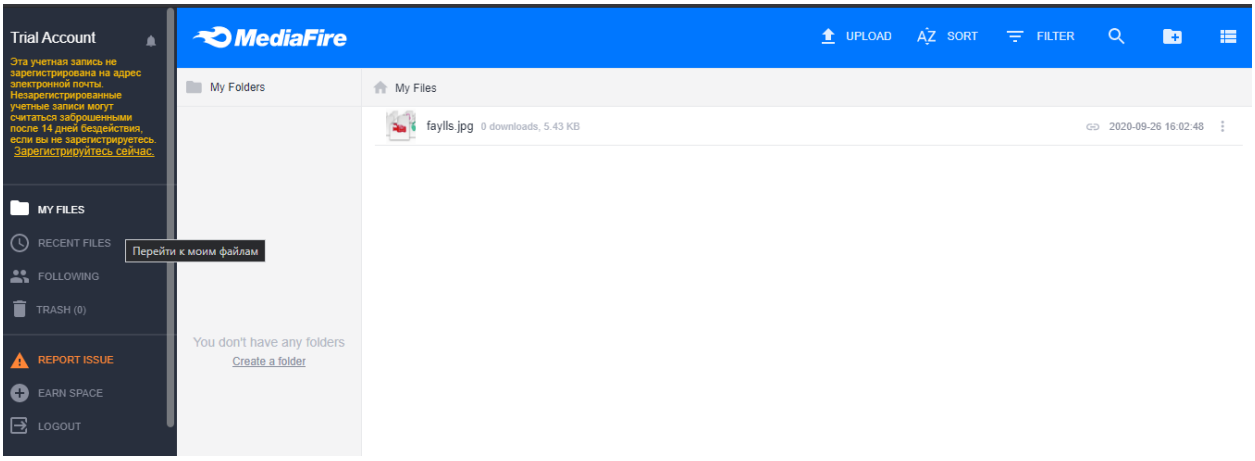


Рисунок 1.4 – головна сторінка mediafire.com

Даний сайт вирізняється унікальним дизайном, зручним пошуком, швидкістю роботи та наявністю можливості завантажувати файли без реєстрації.

Проте навіть даний сайт також має ряд недоліків. А саме базовий розмір сховища 10GB, публікувати лише за посиланням, неможливо оцінити файл. Також на даному сайті не має можливості підписуватись на інших користувачів.

Зробивши аналіз перелічених сайтів була зроблена (таб.1.4) переваг та недоліків.

Таблиця 1.4 – Аналіз розглянутих сайтів

| Критерії | Google Диск | Mega | Dropbox | Mediafire |
|---|---|---|---|---|
| Опис завантажених файлів | - | - | - | - |

| | | | | |
|---|---|---|---|---|
| Зручний інтерфейс | + | - | - | - |
| Не обмежений розмір сховища | - | - | - | - |
| Зручний пошук | + | - | + | + |
| Можливість підписатись на інших користувачів | - | - | + | - |
| Рейтингове оцінювання | - | +/- | + | - |
| Публікувати не лише за посиланням | - | - | + | - |

Тому внаслідок аналізу сайтів зі схожими темами та цілями було вирішено розробити сайт, який не відображатиме недоліків, зазначених на наступній (таб.1.4).
Ця розробка матиме всі переваги цих сайтів, з унікальним дизайном та зручним інтерфейсом.

## 1.3 Вибір засобів реалізації

Розробка веб-сайтів – це процес створення веб-сайтів та додатків для Інтернету. Від найпростіших, статичних веб-сторінок до платформ і додатків соціальних медіа, від веб-сайтів електронної комерції до систем управління контентом (CMS) і файлообмінників; всі інструменти, якими ми користуємось через інтернет щодня, були розроблені веб-розробниками. Веб-розробка може бути розбита на три шари (табл. 1.5): кодування на стороні клієнта (фронтенд), кодування на стороні сервера (бекенд) та технології баз даних.

Таблиця 1.5 – Частини веб-розробки

| Назва | Опис | Приклад технологій розробки |
|---|---|---|
| Клієнтська сторона | Сценарії на стороні клієнта або розробка інтерфейсу стосується всього, що переживає кінцевий користувач безпосередньо. | HTML, CSS, Vue.js, Vuetify, JavaScript, React.js, Bootstrap. |
| | Код клієнта виконується у браузері і безпосередньо стосується того, що бачать люди, відвідуючи веб-сайт. Такі речі, як компонування, шрифти, кольори, меню та контактні форми, керуються фронтендом. | |
| На стороні сервера | Сценарії на стороні сервера або розробка бекенду – це все, що відбувається за кадром. Бекенд – це по суті частина веб-сайту, яку користувач насправді не бачить. Він несе відповідальність за зберігання та впорядкування даних та забезпечує те, що все на стороні клієнта працює безперебійно. | PHP, Laravel, Node.js, Python, Express.js, Docker. |
| Технологія баз даних | Веб-сайти також покладаються на технологію баз даних. База даних містить усі файли та вміст, необхідний веб-сайту для функціонування, зберігаючи його | MySQL, PostgresSQL, OracleSQL, MongoDB |

| Назва | Опис | Приклад технологій розробки |
|---|---|---|
| | таким чином, щоб полегшити його роботу. База даних працює на сервері, і більшість веб-сайтів зазвичай використовують певну форму управління реляційною базою даних | |

### 1.3.1 Розробка інтерфейсу

Завдання інтерфейсного кодування розробником інтерфейсу веб-сайту або програми; тобто тієї частини веб-сайту, яку користувач бачить і взаємодіє з нею. Вони працюють над розробницькими роботами веб-дизайнерів та реалізують їх за допомогою HTML, JavaScript та CSS.

Мова розмітки використовується для визначення формату текстового файлу. Іншими словами, мова розмітки повідомляє програмному забезпеченню, яке відображає текст, як форматувати текст. Мови розмітки повністю зрозумілі та легкі для читання - вони містять стандартні слова, але теги розмітки приховані від користувача в остаточній версії.

Дві найпопулярніші мови розмітки - HTML і XML. HTML розшифровується як Hypertext Markup Language і використовується для створення веб-сайтів. Усі HTML-теги додаються до простого текстового документа, що описує, як браузер повинен відображати документ.

Каскадні макети сторінок або стилі CSS є невід’ємною частиною будь-якої веб-сторінки. Стиль - це, в основному, набір правил стилю. Буквально мови таблиць стилів використовуються для оформлення документів, написаних мовами розмітки.

CSS означає каскадні таблиці стилів, і основна увага приділяється "стилю". Хоча HTML використовується для побудови веб-документів,

визначає вміст, такий як заголовки та абзаци, а також дозволяє вставляти зображення, відео та інші носії, CSS визначає стиль створеного документа.

Більшість сучасних браузерів та веб-додатків мають вбудований модуль для обробки користувацьких команд та забезпечення їх чіткої видимості. Він був створений з використанням мови веб-програмування JavaScript.

JavaScript – це текстовий документ, який можна вбудувати в певний рядок в HTML-код сторінки, або це може бути окремий файл. Її готували за спеціальними правилами. З його допомогою:

– взаємодія з користувачами на веб-сторінці - якщо форма заповнена неправильно, згенерується помилка;

– додайте логічні дії-створіть цикли, алгоритми, умови "якщо, то". Коли користувач наводить курсор на сторінку, з’являється форма передплати з проханням вказати їх контактну інформацію;

– виконайте наукові розрахунки та зробіть висновки в іншій галузі;

– додайте елементи, що рухаються спецефектами, кнопки, пункти меню, таймери зворотного відліку, анімацію, спливаючі вікна або приховані вікна;

– обробки подій та даних без використання мови програмування сервера та Інтернету перевірте, чи заповнені необхідні поля в необхідному форматі, щоб зменшити навантаження на сервер;

– керування вмістом та стиль веб-сторінки - коли відбуваються певні події, змінюють вигляд елементів сторінки та додають до них теги та атрибути HTML;

– ви можете прокручувати фотографії в галереї та портфоліо.

Це далеко не повний перелік функцій JavaScript. Його створення та вдосконалення - це багатогранний інноваційний процес.

Розглянемо слово "фреймворк". Якщо розглядати переклад слова з англійського - це "конструкція" або "структура".

Суть цієї основи полягає саме в перекладі слів. Це спеціальне програмне середовище, що використовується для значного полегшення процесу комбінування певних компонентів при створенні програми. Це фреймворк, який дозволяє додавати компоненти за потреби. За відсутності особливих труднощів основу для процедури призначення можна сформувати досить швидко.

Порівнюємо CMS, чистий код та фреймворк.

Якщо перед програмістом стоїть завдання створити сайт, він повинен негайно визначити майбутню стратегію. Існує три методи розробки, кожен програміст може вибрати той, який найбільше відповідає його навичкам.

1. Можете написати необхідний вихідний код з нуля. Головною перевагою цього варіанту є мінливість - майже немає обмежень, можна досягти будь-якої бажаної функції, потрібні лише певні навички. Основним недоліком є складність процесу та витрати часу. Вам також доведеться докласти чимало зусиль, щоб ретельно протестувати продукт - вам доведеться знайти всі його недоліки, щоб створити ідеальний веб-проект.

2. Використання фреймів. Якщо ви схожі на попередній метод, є деякі обмеження. За основу потрібно додати певну кількість необхідних компонентів. Цей варіант є економічно вигідним лише для тих, хто принаймні володіє програмуванням. Для тих, хто не може використовувати ці методи, є інший варіант.

3. Використовуйте існуючу CMS. Цей варіант ідеально підходить для людей, які мають незначні знання в галузі веб-розробки. Ви зможете швидко створити веб-сайт, який відповідає вашим вимогам. Необхідні коригування можна здійснити за допомогою панелі управління. Але цей спосіб не дуже популярний - головним недоліком є велика кількість.

Відповідно до характеристик порівняння можна зрозуміти, що цей фреймворк є «золотим методом» між написанням складних кодів та використанням системи управління вмістом з обмеженими функціями. Ви отримаєте готовий фреймворк для свого проекту, не втрачаючи

функціональної гнучкості. Фреймворки поділяються відповідно до мови програмування, якій вони належать [12].

Давайте подивимось на дивовижний шлях розвитку минулого року, досконалий JavaScript, який знайшов своє відображення у серцях понад 20 000 веб-розробників з усього світу. Ми вивчимо структуру javascript, найбільш необхідну для інтерфейсної розробки, обробки даних та внутрішньої розробки, і спробуємо знайти можливих майбутніх лідерів.

У цій статті ми базуватимемось на даних та висновках про стан JS, а також на думках, які JetBrains ділиться у своєму щорічному звіті "Стан екосистеми розробників".

Щорічне опитування щодо всього змісту JavaScript. Візуальні відповіді розробників-учасників відображають поточний стан JavaScript. Опитування охоплює такі теми: інтерфейсні фреймворки, бази даних, інструменти стану, взаємозв'язок між фреймворками, вибір розробниками та простота використання фреймворків, мови, скомпільовані в JavaScript, мобільні фреймворки, системи компіляції, засоби тестування коду JavaScript тощо.

Це перше опитування, спеціально проведене розробниками, що займаються JavaScript. Він був організований у 2016 році Олександром Грейфе та його помічниками. Зараз користується високою повагою серед цільової аудиторії. Є й інші не менш популярні опитування - від Stack Overflow (понад 100 000 респондентів) або від JetBrains (понад 6 000 респондентів). Однак сьогодні ми зосередимось на державному розслідуванні СВ - ще й тому, що воно не має собі рівних з точки зору візуалізації.

Ми живемо в епоху компонентності, а не лише передньої частини. Такі інструменти, як Bit, набирають популярності, і, об'єднавши їх та синхронізуючи з іншими проектами, ви можете легко ділитися компонентами, використовувати їх повторно та швидше створювати нові програми. Майбутнє очевидно за платформою, орієнтованою на компоненти.

Завдяки зручності користування, задоволення та простоти використання, інтерфейсний фреймворк буде розглядатися у найпопулярнішому порядку.

React – це ефективна та гнучка декларативна бібліотека JavaScript для побудови інтерфейсу від команди Facebook. Це дозволяє легко створювати інтерактивні користувальницькі інтерфейси. React не тільки підтримує побудову об’єктно-орієнтованих додатків, але й заохочує це робити. Крім того, творці фреймворку дуже серйозно ставляться до зворотної сумісності, тому ви можете бути впевнені в довговічності їх програм. Наведена діаграма наочно показує, що за останні кілька років розуміння React значно покращилося. Ось чому ця бібліотека стане гарною відправною точкою для вашого інтерфейсного розвитку.

React-розробники – одні з найбільш високооплачуваних javascript-розробників в 2018 році.

Реліз фреймворка Vue.js 3.0 відбувся ще в 2018 році, але багато власників технологічних компаній і Javascript розробники все ще не наважуються використовувати його в своїх проектах[11].

Vue.js – створений Еваном Ю і 234 іншими ентузіастами, є прогресивною структурою для створення користувальницьких інтерфейсів і отримав понад 121 000 зірок на GitHub. Він включає доступну кореневу бібліотеку (яка головним чином вирішує проблему рівня презентації) та екосистему додаткових бібліотек, що дозволяють створювати складні та величезні односторінкові програми (Single-Page Applications).

Vue.js вимовляється в точності як слово view («уявлення») і має на 4 тисячі зірок GitHub більше, ніж React.

Angular – це структура Google, яка отримала майже 44 000 зірок на GitHub. Це платформа, яка спрощує збір додатків в Інтернеті. Angular поєднує декларативні шаблони, реалізацію залежностей, двостороннє прив'язку даних та найкращі практики вирішення проблем розвитку. Платформа дозволяє збирати програми для Інтернету, мобільних пристроїв

та робочих столів. Він забезпечує найбільш зручний та зрозумілий спосіб для інтерфейсу командного рядка для початківців (CLI) або навіть консолі (Console) – клієнт з графічним інтерфейсом [10].


Діаграма 1.1 – рейтинг використання розробниками.

Проаналізувавши всі перелічені варіанти, через безліч переваг конкурентів, я вирішив розробити клієнтську частину фреймворку Vue.js.

Далі вводяться переваги:

- Реактивні інтерфейси;
- Декларативний рендеринг;
- Зв'язування даних;
- Директиви (всі директиви мають префікс «V-». В директиву передається значення стану, а в якості аргументів використовуються html атрибути або Vue JS події);
- Логіка шаблонів;
- компоненти;
- Обробка подій;
- властивості;
- Переходи і анімація CSS;

- Фільтри.

  Основна бібліотека Vue.js 3 дуже мала (лише 17 кБ). Це гарантує мінімальне навантаження на проект Vue.js і веб-сайт може швидко завантажуватися [11].

### 1.3.2 Розробка серверної частини

Код, з генерований внутрішніми розробниками, гарантує, що всі функції, створені розробниками інтерфейсний програм, функціонують належним чином. Основним завданням розробника серверної частини є забезпечення взаємодії сервера, програми та бази даних між собою. Для вирішення цієї складної проблеми розробники використовують для створення програм серверні мови, такі як PHP, Ruby, Python та Java.

Веб-технології швидко розвиваються, і для того, щоб не відставати від темпів нових технологій, веб-програмістам часто бракує часу для виконання щоденних завдань з нуля, таких як автентифікація, API, тестування, налагодження, хешування тощо. З метою спрощення розробки сайту та швидкого вирішення цих проблем була розроблена структура.

Фреймворк - це фреймворк, призначений для створення динамічних веб-сайтів, веб-додатків, послуг або ресурсів. Це спрощує розробку та позбавляє потреби писати звичайний код. Структура спрощує доступ до бази даних, розробку інтерфейсу та зменшує дублювання коду.

Розглянемо популярний фреймворк Yii 2, Yii - це універсальний фреймворк, який можна використовувати у всіх типах веб-додатків.

Нарешті, база даних є однією з найпоширеніших баз даних, посідаючи друге місце у світі, поступаючись лише корпоративній базі даних Oracle. Це означає, що у разі виникнення проблем або збоїв у роботі системи мережа може знайти багато матеріалів для вирішення цих проблем.

Виберіть мови програмування HTML, CSS та JavaScript, щоб розробити клієнтську частину веб-сайту, оскільки вони є найпопулярнішими мовами у веб-розробці, і їх дуже легко вивчити.

Серверна частина використовуватиме для розробки фреймворк Laravel та мову програмування PHP. Мова активно використовується багатьма сайтами і дуже проста у вивченні. Подібним чином, найпопулярніші системи контролю вмісту використовують цю мову програмування, яка надає можливості для розвитку інших функцій.

# 2 ПОСТАНОВКА ЗАДАЧІ І МЕТОДІВ ДОСЛІДЖЕННЯ

## 2.1 Мета та задачі роботи

Дана інформаційна система представляє з себе сервіс, що надає користувачеві місце під його файли і цілодобовий доступ до них через інтернет, як правило по протоколу http. Такий сервіс дозволяє зручно обмінюватись файлами між іншими користувачами інтернет мережі. На спеціальній сторінці файлообмінника користувач матиме можливість завантажувати файли на сервер файлообмінника, а файлообмінник віддає користувачеві постійне посилання, яку він може розсилати по e-mail, публікувати в блогах, на форумах або інших порталах. Перейшовши по такому посиланню будь-який інший користувач може завантажити початковий файл.

Сервіс повинен бути реалізований у вигляді сайту, доступного в мережі Інтернет. Сайт повинен складатися із взаємозалежних розділів із чітко розділеними функціями.

Головна мета інформаційної системи – створення повноцінної платформи для зберігання файлів з можливістю доступу до них з різних типів операційних систем і гаджетів, надання надійного зберігання файлів на сервері.

Перелік вимог до сервісу:

1. Сайт повинен складатись з двох частин, а саме користувацької та адміністративної.
2. Всі користувачі перед використанням повинні зареєструватись в системі.

Користувацька частина повинна складатися із наступних сторінок:

– Головна.
– Мої файли.
– Популярні файли.
– Обране.

– Кошик.

На головній сторінці сайту повинні бути такі елементи:

– Пошук файлів.
– Перегляд останніх публікацій користувачів.
– Перегляд інформації про файл.
– Можливість завантажувати файл на сервіс.

Доступ до адміністративної частини матимуть лише адміністратори сайту. Для них повинен бути розроблений функціонал який надає можливість блокувати користувачів, видаляти файли, переглядати список файлів і користувачів.

Адміністративна частина повинна складатися із наступних сторінок:

– Головна.
– Користувачі.
– Файли.
– Налаштування.

Для всього цього функціоналу потрібен простий та зручний дизайн. Тому потрібно приблизитись до схеми зображеної на рисунку 2.1.

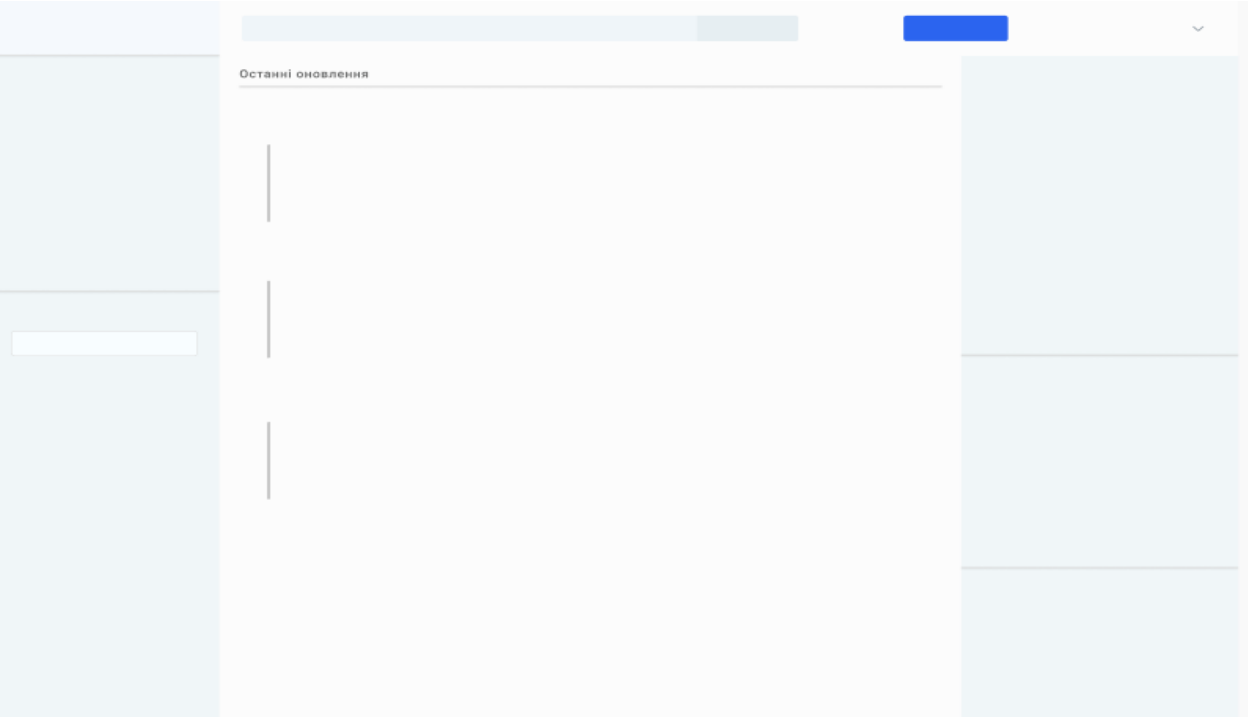


Рисунок 2.1 – Схема можливого дизайну інтерфейсу.

## 2.2 розробка клієнтської части

Насправді кожен веб-сайт містить текст, картинки,

Мультимедіа або інші об'єкти. Для того, щоб відвідувачі могли комфортно та легко шукати необхідну інформацію на сайті, сайт повинен мати чітку та продуману структуру.

Структура сайту - це логічна структура всіх сторінок ресурсу. Схема шляху до папки, категорії, підкатегорії, картки товару (якщо така є). З технічної точки зору, навігація ресурсами - це набір URL-адрес, розташованих у логічному порядку. Ця структура взаємопов’язана із семантичним ядром. Там сказано, які папки та файли повинні бути на сайті. Отже, після того, як семантика була зібрана, можна скласти схему побудови кожного майбутнього вузла. Ви можете побачити малюнок нижче (рис.2.2).

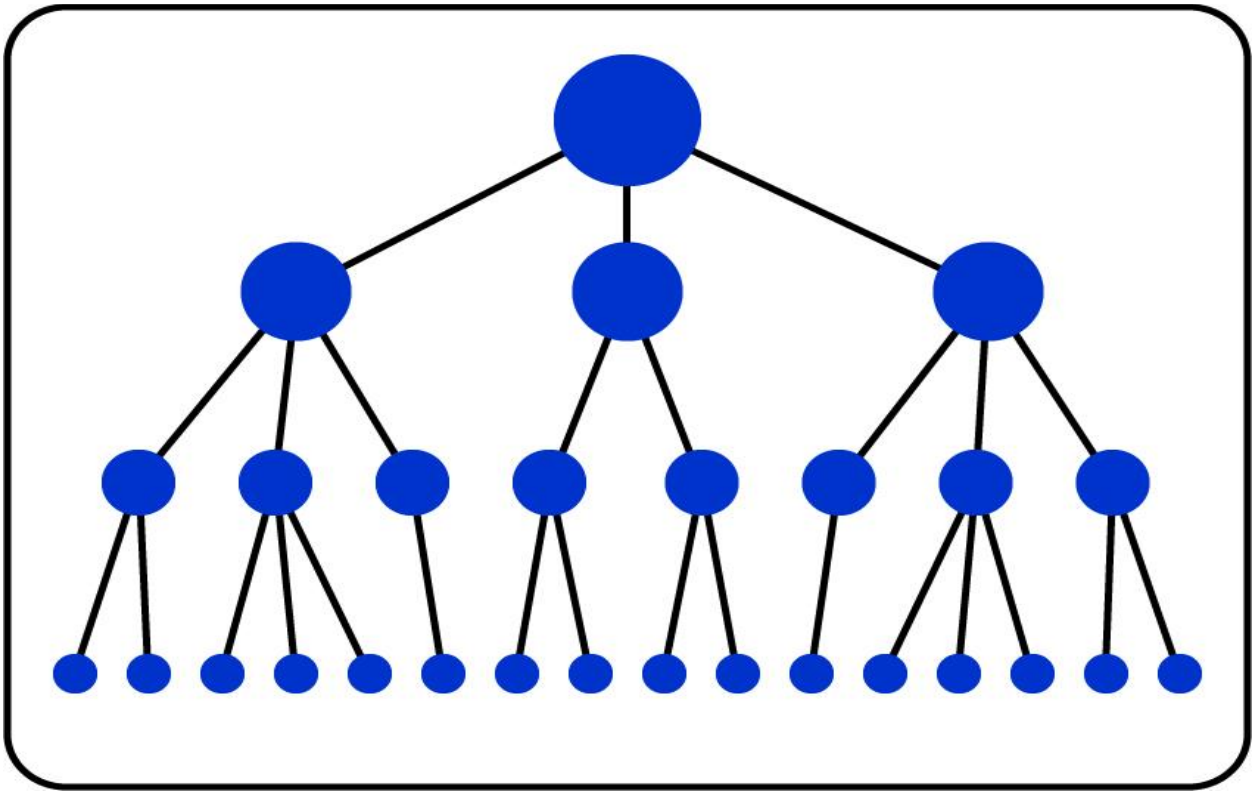

Рисунок 2.2 – Схема базових структур сайтів.

З точки зору розробника, структуру веб-сайту можна розділити на два рівні - логічний і фізичний. Логічно кажучи, структура сайту - це набір сторінок, які поєднані єдиним дизайном, стилем та посиланням.

На фізичному рівні структура сайту - це впорядкована колекція файлів різного типу (HTML-сторінки, зображення, програми, мультимедійні файли).

Внутрішня структура веб-сайту, як правило, представлена картою сайту - карта наочно показує ієрархічну структуру сторінок веб-сайту, зв'язки між ними та схему переходу.

Зовнішня структура - це розташування основних важливих елементів на кожній сторінці. Вам потрібно визначити місце розташування та спосіб меню, яким може бути пошук, основний зміст, певні оголошення про нові частини веб-сайту, оновлення, лічильники та банери (якщо вони є на веб-сайті).

Розробляючи зовнішні та внутрішні структури, вони зосереджувались на тому, щоб полегшити перегляд сайту майбутнім відвідувачам, щоб можна було легко знайти важливу та необхідну інформацію. Тому перед тим, як розпочати розробку структури сайту, потрібно дослідити ресурси на подібні теми та зрозуміти, як вирішити це завдання.

Якщо говорити більш загальною мовою, структуру сайту можна порівняти з супермаркетом. Коли ви заходите до магазину, ви очікуєте побачити молоко у молочному відділі та хліб у пекарні. Погодьтеся, що якщо ви не знайдете товар у відповідному відділі, ви не підете в магазин, щоб знайти товар, а підете лише на інший ринок. Подібна до структури. Якщо споживач не знайде потрібний йому товар у бажаній категорії, він перейде до вашого конкурента. У цьому випадку важливість програми з точки зору споживчих факторів.

Хороша структура, ефективно сприяє CEO-просування.

Пошук здійснюється за посиланнями на веб-сайті. Більше того, чим правильніше, простіше структура веб-сайту, тим менше часу витрачають пошукові системи на свої ресурси і тим швидше вони можуть обходити

ресурси. Це призведе до найшвидшого індексування веб-сайтів. Крім того, розміщення документів правильно та обґрунтовано на веб-сайті може ввести індекс пошукової системи наступного дня. Отже, структура безпосередньо впливає на рейтинг. Чим кращий ефект, тим швидше буде діяти індекс. Це незамінний елемент СЕО-просування.

Не забувайте про розподіл ваги внутрішніх посилань на сторінці. Без внутрішньої ваги сторінки можуть не індексуватися та правильно класифікуватися, оскільки пошукові системи можуть виявити, що ці сторінки мають низьку якість та низький рівень довіри. Крім того, найважливіші сторінки повинні додати ваги. Правильно структурований веб-сайт призначить права внутрішнього контролю на всі сторінки на основі його важливості для користувачів та рекламних акцій. Тому це також впливає на рейтинг.

Ви не можете налаштувати результат випуску своїми руками. Алгоритм пошуку самостійно формує цю функцію відповідно до структури, довіри пошукової системи до ресурсу та найбільш популярної категорії або частини ресурсу.

Хороша структура довіри пошукової системи до веб-сайту

Пошукові системи проводять аналогії між сайтами та ставленням користувачів до сайту. Чим привабливіші ресурси для споживачів, тим привабливішою є пошукова система. Якщо алгоритм Google вважає, що відвідувач веб-сайту має короткий час відвідування та низький рівень кліків, він вважатиме якість ресурсу низькою, що матиме дуже негативний вплив на просування сайту та ефективність роботи пошукової системи.

Якщо користувач задоволений веб-сайтом (частково залежно від структури), він витратить багато часу на усунення рівня відмов, зробить замовлення, а алгоритм пошуку однозначно помітить і підвищить рейтинг вашого веб-сайту у вигляді покращеного рейтингу пошукової системи.

Правильна структура фокусує користувачів на основних операціях на сайті. Це усуває труднощі з пошуком сторінки, яка потрібна людині. Інакше

це назавжди зменшить кількість конверсій, оскільки користувачі не знайдуть того, що хочуть, а отже, не стануть клієнтами.

Типи структури сайту: ієрархія та класифікація

Якщо освоїли семантичне ядро і можете уявити всю інформацію, яка буде використана на ресурсі, то після того, як ви зрозумієте структуру сайту, вам слід побудувати його у вигляді плану. Іншими словами, ієрархічна структура є логічним і послідовним ланцюгом побудови та подання інформації.

З точки зору ієрархічної структури, структура поділяється на кілька шарів, починаючи з першого шару. Перший рівень - це домашня сторінка та основні категорії, другий рівень - це підкатегорії тощо. Ієрархічна структура структури полегшить і пришвидшить пошук користувачів, які вони хочуть. Наприклад, користувачі шукають електричні чайники у великих інтернет-магазинах електроніки. Шлях до пошуку чайника повинен виглядати так:

Головна;

Товари для кухні;

Дрібна побутова техніка;

Електричні чайники;

Товарна картка.

Шлях від основної картки до картки товару вимагає лише трьох кліків, що дозволяє споживачам якомога легше знаходити товари.

# 3. МОДЕЛЮВАННЯ ТА ПРОЕКТУВАННЯ

## 3.1 Проектування структури Веб-додатка

Більшість додатків мають структуру, і веб-сайт повинен вимагати принаймні одну стандартну структуру. Спільний доступ до файлів також вимагає гармонійної структури зовнішніх інтерфейсів та ієрархії файлів на сервері. Функція програми полягає в тому, щоб показати користувачеві, як взаємодіяти з сервером через браузер. Цей спосіб реалізації проекту має ряд переваг. На відміну від програмних програм, користувачам не потрібно завантажувати та встановлювати програми. Веб-сайт також можна використовувати на всіх пристроях, незалежно від операційної системи та технічних характеристик.

Процес розробки можна розділити на чотири частини:

- дизайну програми, верстка зовнішнього вигляду;
- розробка модуля база даних;
- розробка модуля структурування інформації і виводу;
- розробка модуля збереження інформації і підключення до бази даних;

Застосовуючи спрощені методи та різні інструменти, проектування інформаційної системи відоме як багатоступеневий процес створення та модернізації.

Якщо підкреслити етап проектування інформаційної системи як окремий етап, її можна розмістити між етапами аналізу та розробки. Загальну структуру інформаційної системи можна розділити на кілька частин, а саме до і після авторизації. Основна відмінність полягає у доступі до інформації та певних сторінок веб-сайту.

Після авторизації користувачі мають інші можливості. Перш за все, це залежить від типу рахунку-замовника або сервісного центру.

Рядок меню містить посилання на ключові модулі системи. Розглянемо докладніше, які можливості відкриті для кожного типу рахунку акаунтів у табл.3.1.

Таблиця 3.1 – Функціонал за групами користувачів

| № | Користувач | Можливості користувача |
|---|---|---|
| 1 | Адміністратор | Перегляд загальної інформації; |
| | | Підтримка сайту; |
| | | Блокування користувачів; |
| 2 | Звичайний користувач | Створення власної інформаційної сторінки; |
| | | Редагування даних; |
| | | Завантаження файлів; |
| | | Обмін файлами; |
| | | Відстежувати публікації інших користувачів; |
| | | Залишити відгуки про файли; |
| | | Написання повідомлення адміністратору. |

Інформаційна система ретельно розроблена для досягнення якісних взаємодій для всіх користувачів. Структура інформаційної системи це повністю доводить. Кожен користувач має можливість обробляти файли, взаємодіяти з іншими користувачами тощо.

Крім того, були виконані всі завдання, поставлені на початку фактичної роботи.

### 3.2 Структурно-функціональне моделювання процесу

Багато в чому IDEF0 - це дуже простий метод. Як і в інших методах, кожне поле представляє окремий процес, але IDEF0 відрізняється використанням і розташуванням стрілок. На додаток до звичайних входів і виходів, є дві стрілки, що представляють "елементи керування" та "механізми".

Елементи керування є формою введення даних, але використовуються для управління діяльністю в процесі. Іноді існує певна міра невизначеності щодо того, є елемент вхідним елементом чи елементом управління.

Розглянемо IDEF0 інформаційної системи (рис.3.1).


Рисунок 3.1 – IDEF0 інформаційної системи

Метод IDEF1 поділяє структурні елементи інформаційної галузі, їх атрибути та взаємозв'язки на кілька категорій. Основною концепцією методології IDEF1 є основна концепція. Клас сутності - це сукупність інформації, накопиченої та збереженої в межах підприємства, що відповідає певному об’єкту або групі об’єктів у реальному світі.

Основними концептуальними властивостями сутностей в IDEF1 є:

– стійкість. Інформація, що стосується конкретних організацій, постійно накопичується.

– Унікальність Будь-яку сутність можна однозначно ідентифікувати від іншої сутності.

Кожна сутність має своє ім’я та атрибути. Атрибути - це характеристики та характеристики об’єктів у реальному світі, пов’язаних із певною сутністю. Клас атрибутів - це набір пар, що складається з імен атрибутів та їх значень для конкретних сутностей. Атрибути, які можуть однозначно відрізнити одну сутність від іншої, називаються ключовими атрибутами.

Кожна сутність може характеризуватися кількома ключовими атрибутами. Клас відносин в IDEF1 - це набір взаємозв’язків між сутностями. Якщо клас атрибутів однієї сутності містить ключові атрибути іншої сутності, вважається, що існує зв'язок між двома окремими сутностями. Відповідно до методу IDEF1, кожен із наведених класів має своє графічне представлення умов. Розглянемо інформаційну систему IDEF1 (рис.3.2).

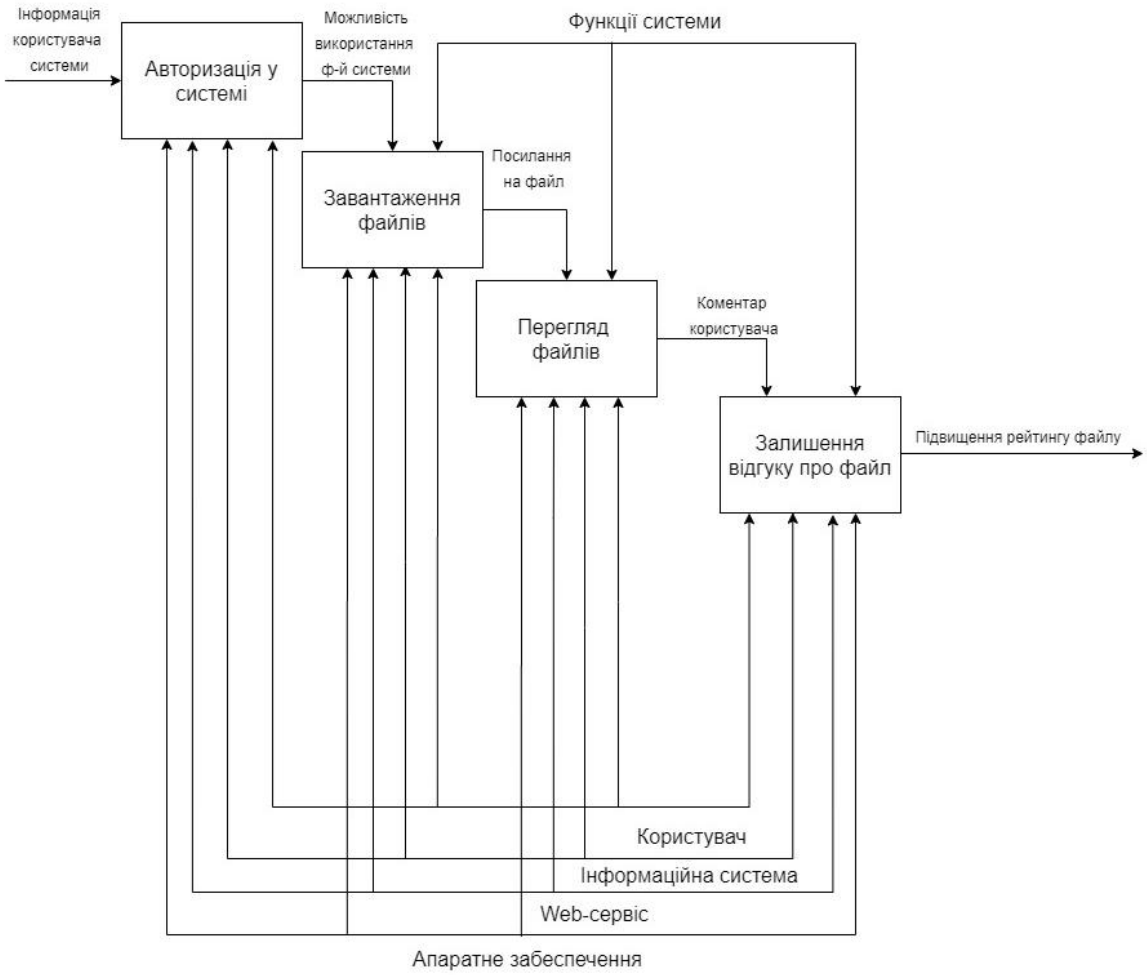


Рисунок 3.2 – IDEF1 проекту

### 3.3 Інтерфейс програми

Створено інформаційну сторінку, на якій знаходиться інформація про послуги даного Веб-додатка (рис.3.3).

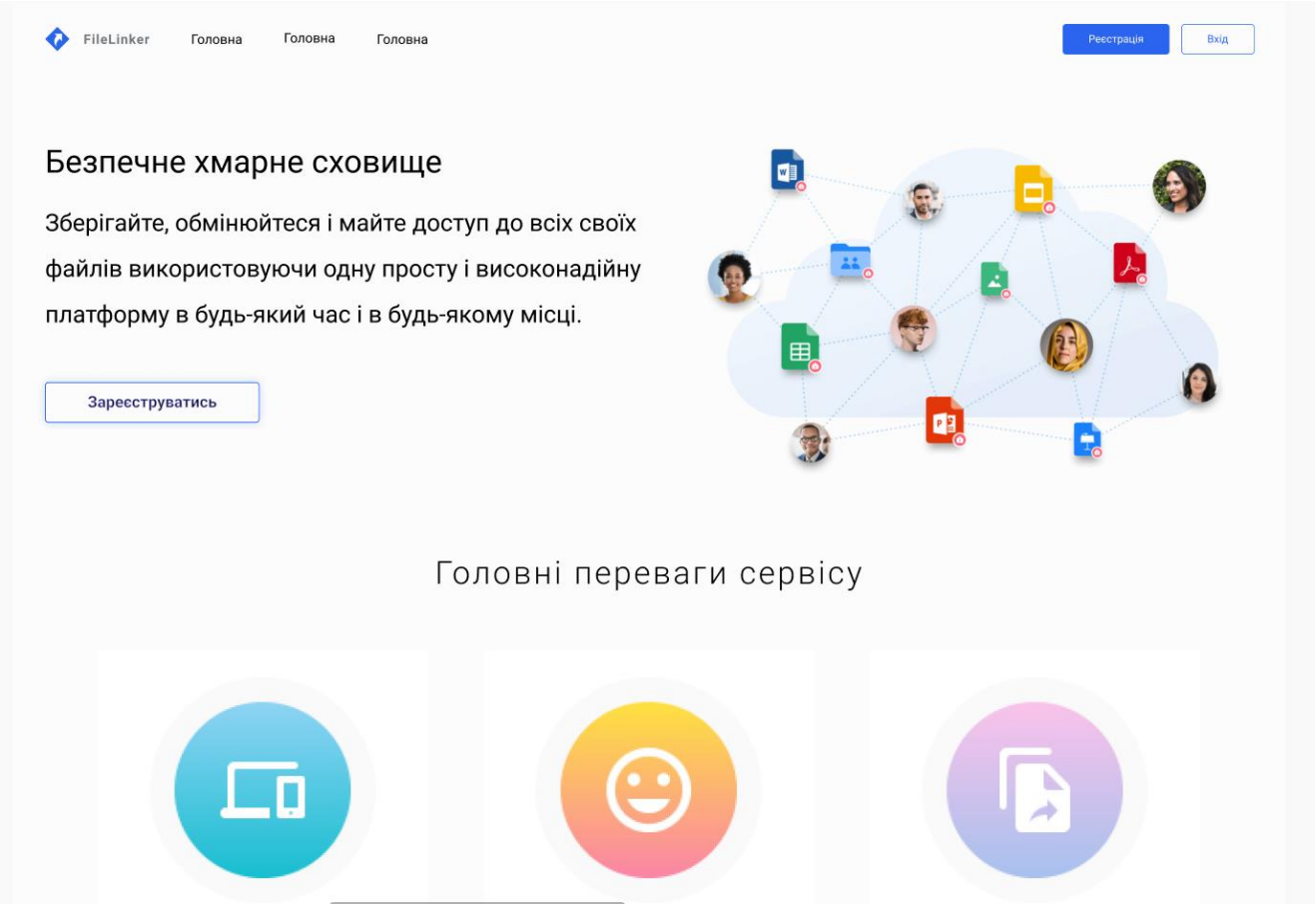


Рисунок 3.3 – Інформаційна сторінка про Веб-додаток.

На сайті потрібно зареєструватися. То ж було добавлино кнопку «Зареєструватись». Щоб створити нового акаунта користувача, потрібно вказати особисту інформацією (рис.3.4).

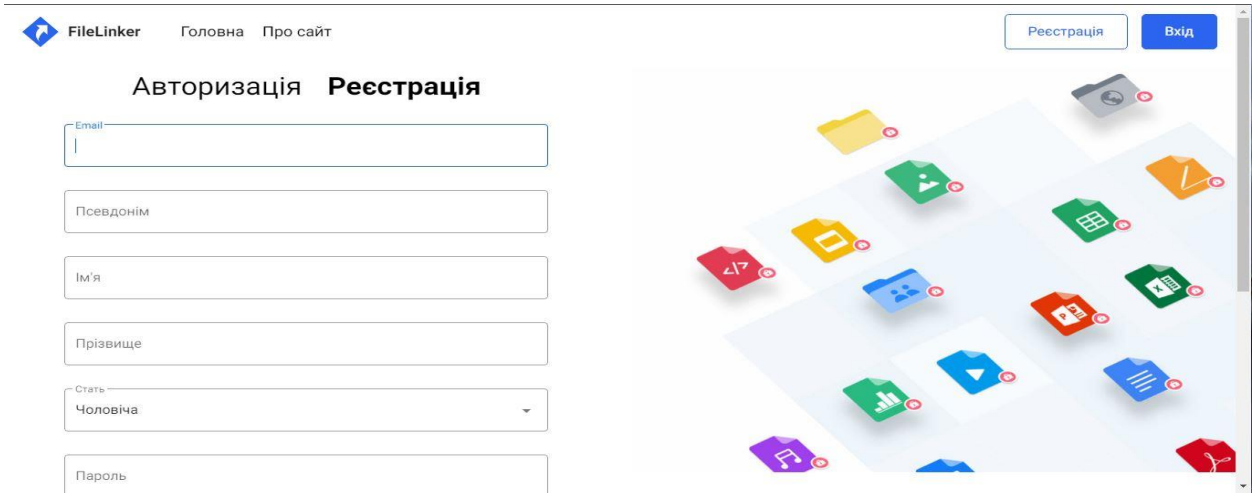


Рисунок 3.4 – Сторінка реєстрації нового користувача.

Створивши новий акаунт користувача, та натиснувши курсором на заголовок «Авторизація», форма зміниться і зареєстрований користувач може авторизуватися (рис.3.5).

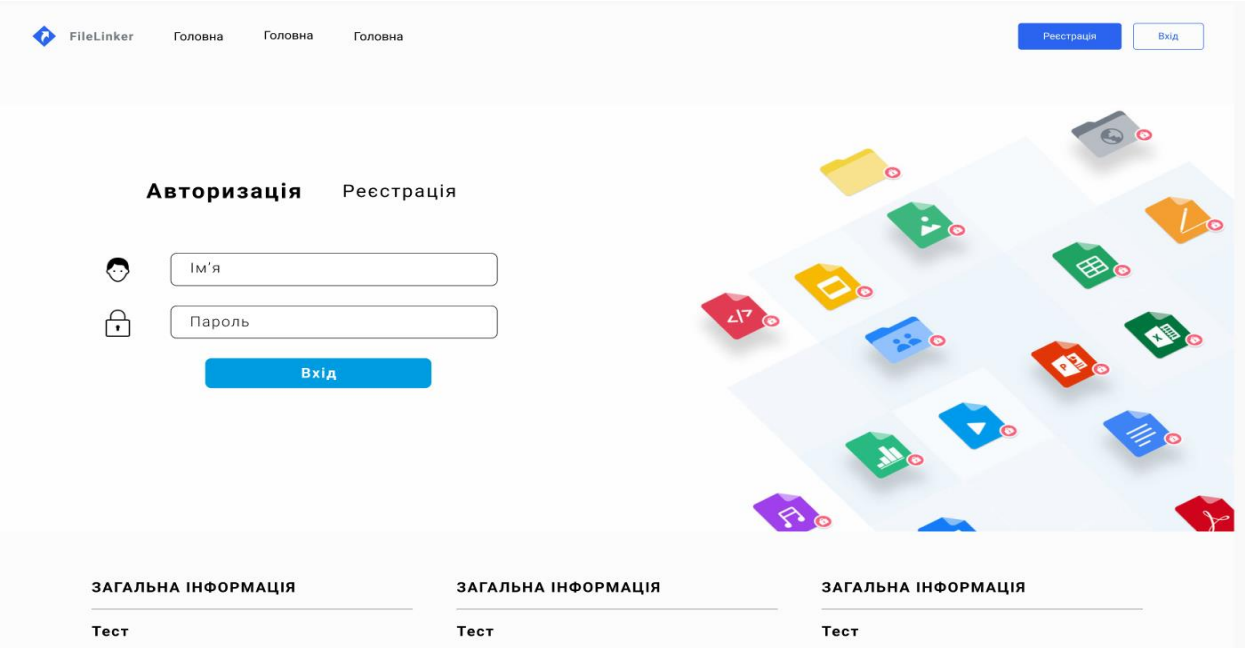


Рисунок 3.5 – Форма авторизації.

Авторизований користувач може розпочати користування на сайті (рис.3.6).

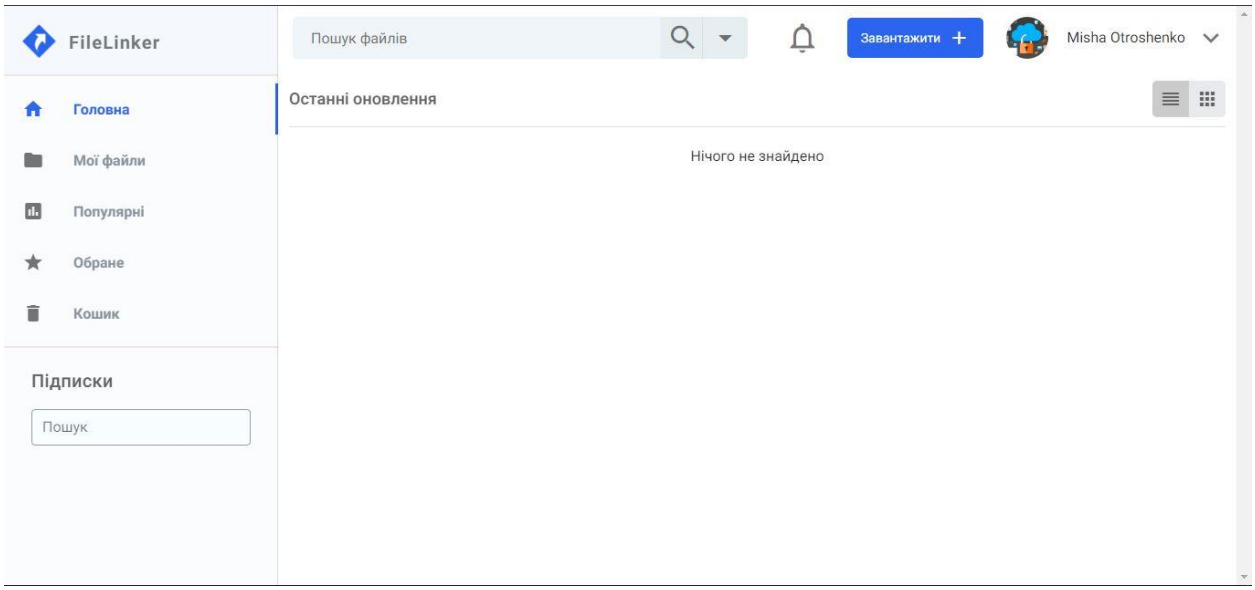


Рисунок 3.6 – Сайт готовий для користування.

Цілю користування Веб-додатка завантажити, скачати, переглянути інформаційні файли, фото або інші інформаційні ресурси, на сервері для закритого або вільного доступу.

Натиснувши курсором на кнопку «Завантажити +», яка знаходиться на сторінці користувача. Відкриється по верх сторінки модальне вікно, в який можна внести інформаційний ресурс, також можна замінить назву та додати опис ресурсу, що стане корисно для тих хто більше зацікавленій в опублікованих вами ресурсах.

Рисунок модального вікна (рис.3.7, 3.8, 3.9).

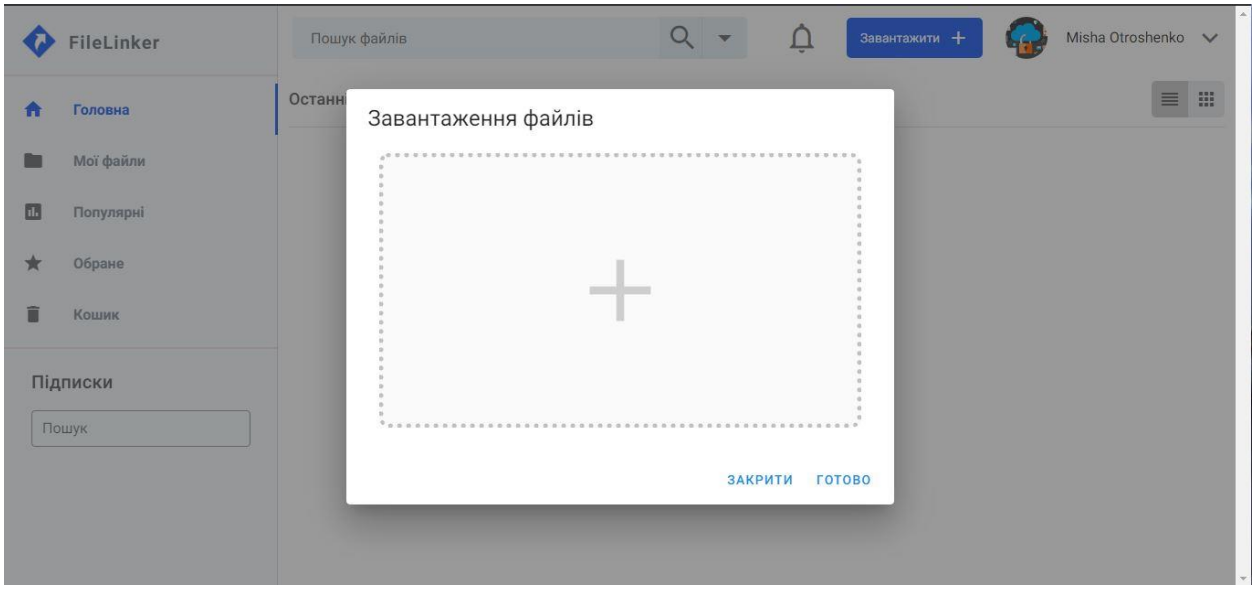


Рисунок 3.7 – Модальне вікно для заватаження.

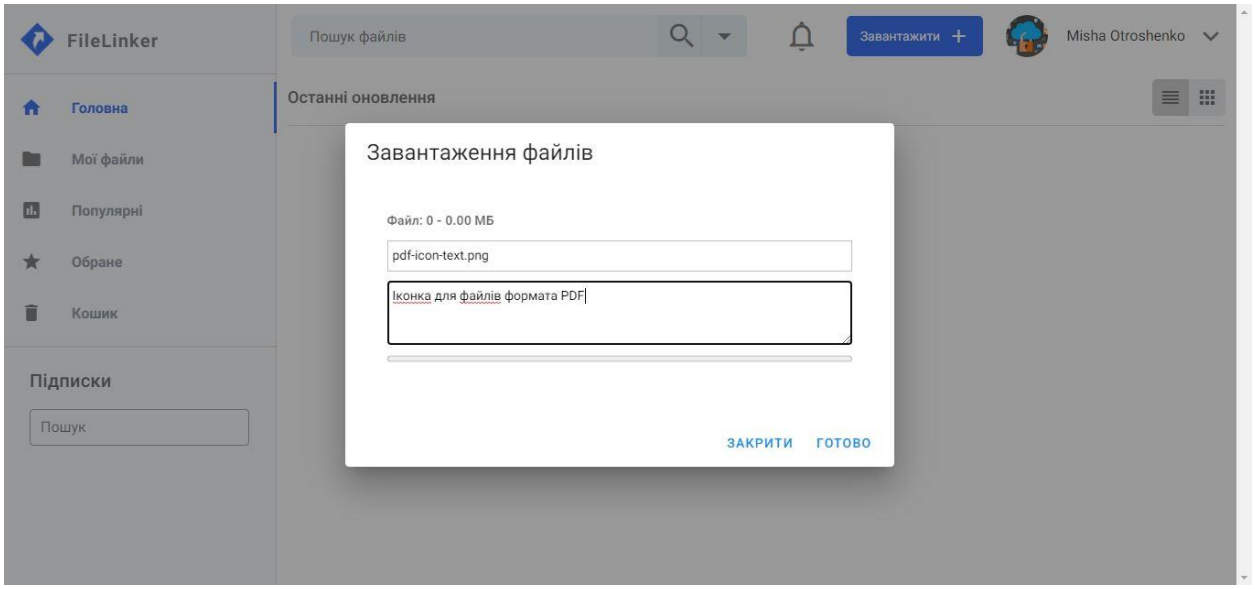


Рисунок 3.8 – Модальне з доданим файлом.

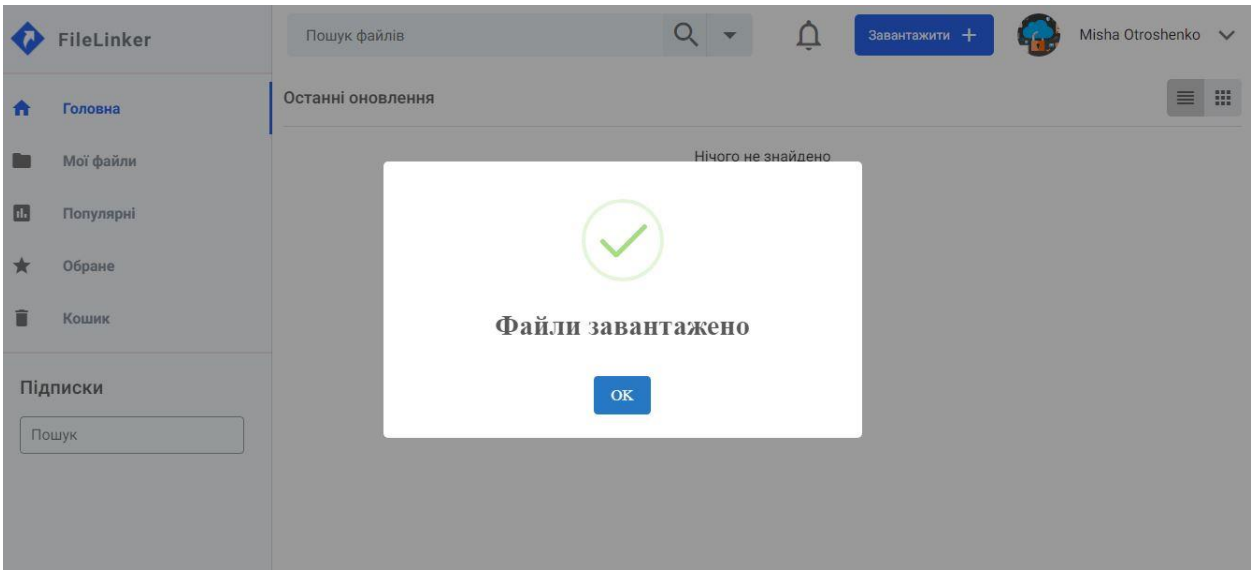


Рисунок 3.9 – Повідомлення про збеження файла на сервер.

Коли інформаційний ресурс завантажений, його можна знайти за допомогою панелі навігаційної «Мої файли», яка знаходиться з лівої сторони Веб-додатка. А також там присутні інші не менш важливі функціональні кнопки (рис.3.10).

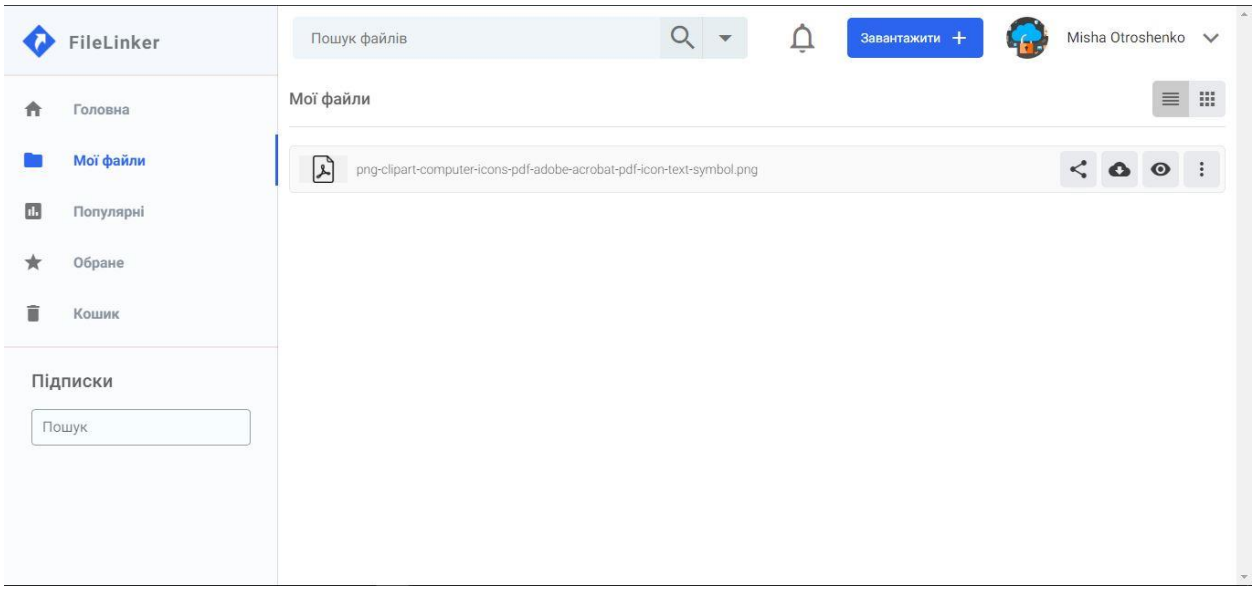


Рисунок 3.10 – Сторінка з вашими файлами.

Режими відображення ресурсів. Натиснувши курсором, на бажаний, то зміниться стиль та розмір зображення ресурсів для зручності (рис.3.11).

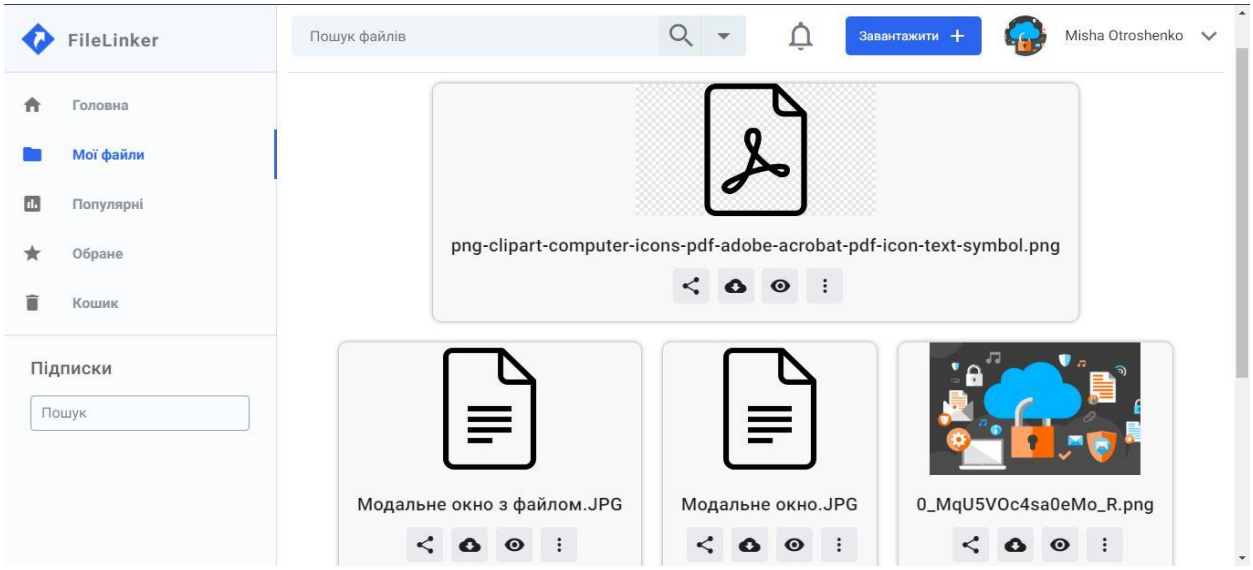


Рисунок 3.11 – Стиль змінний.

Після загрузки на сервер, ресурси можна публікувати або створить посилання на загрузку и поділиться ним в соціальних мережах.

Також створена сторінка особистої інформації користувача. На цій сторінці є можливість змінити деякі дані, та пароль за необхідності (рис.3.12, 3.13).

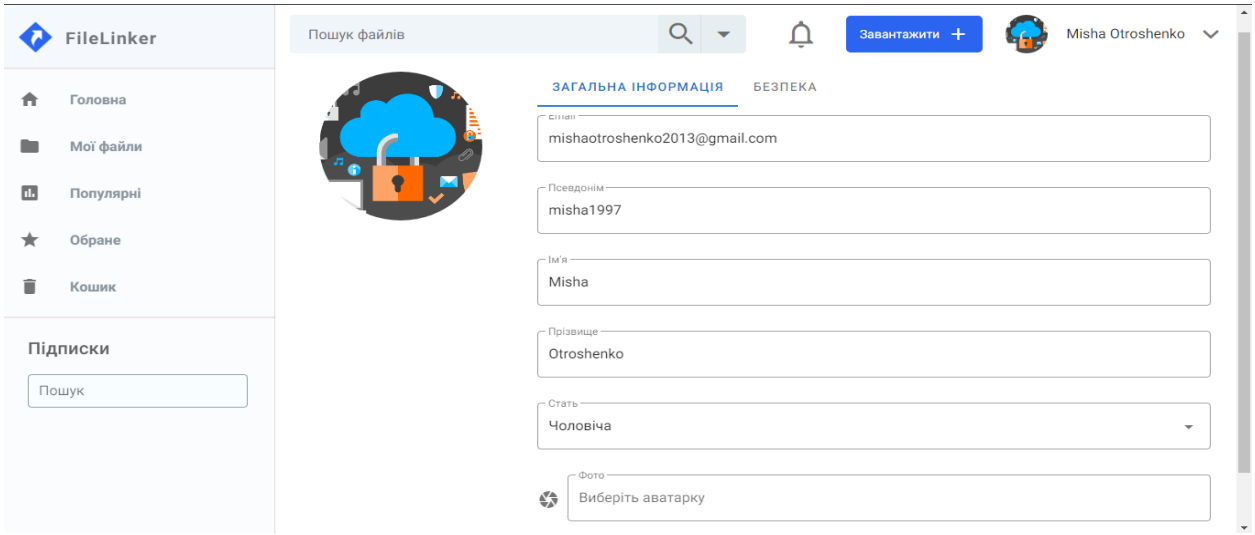


Рисунок 3.12 – Особиста інформація користувача.

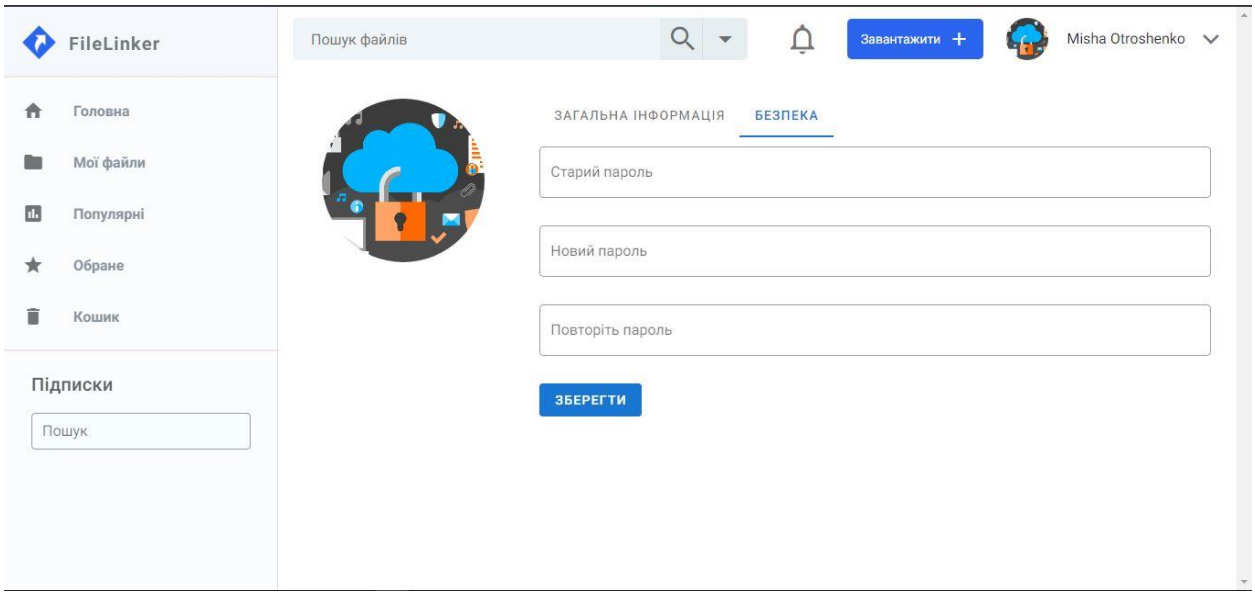


Рисунок 3.13 – Зміна пароля користувача.

Зроблено на Веб-додатку можна підписуватися на іншого користувача, та можна завантажувати опубліковані файли з цікавою для вас тематикою. І тепер завжди в курсі нових публікацій (рис.3.13).

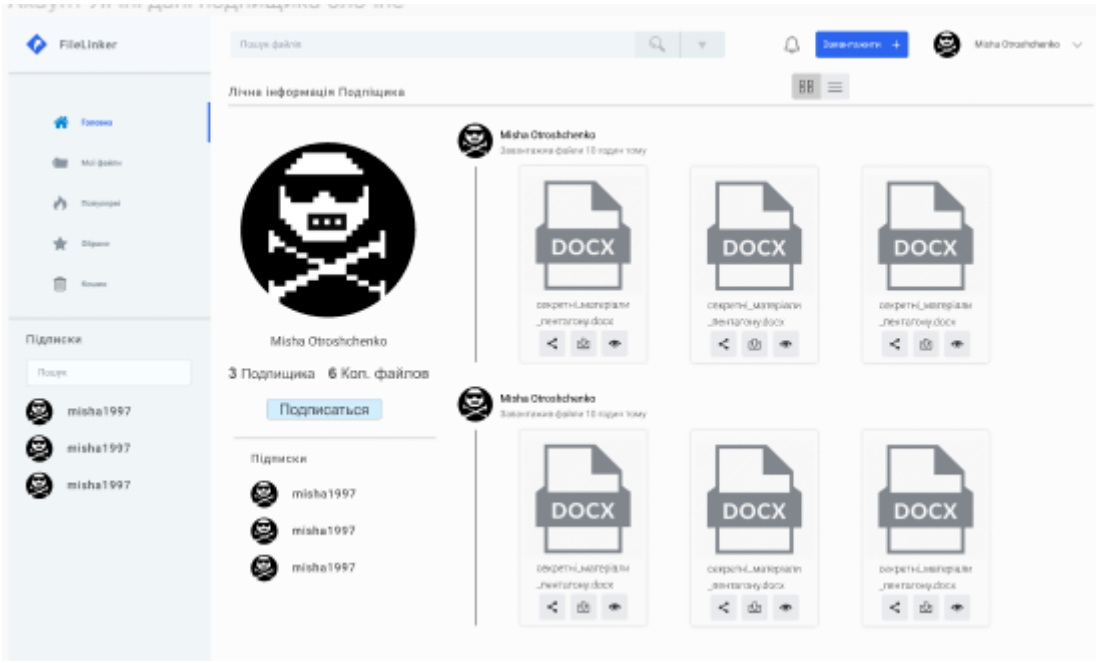


Рисунок 3.13 – Сторінка користувача на якого можна підписатись.

# ВИСНОВОК

Отже одна з головних функцій сервісу – обмін файлами. Користувач можете завантажувати дані і відкривати до них доступ іншим користувачам, а також отримувати посилання для перегляду та скачування файлів.

В науково-дослідницькій практиці було проведено дослідження хмарного зберігання файлів, а також типів хмарних сховищ і методів забезпечення безпечного зберігання даних. Метою роботи було визначати актуальність проекту, проаналізувати як працюють хмарні сховища і провести аналіз аналогів даної системи.

Робота була проведена:

1. Визначила тема, мета проходження практики.
2. Підібрана література за темою дипломної роботи.
3. Зроблиний аналітичний огляд літератури за темою дипломної роботи.
4. Обґрунтований вибір апаратно-програмного інструментарію проектування.
5. Розроблено інформаційну, концептуальну модель додатку.
6. Здійснино збір та сформовані вхідні дані.
7. Оформлено звіт з практики.

Мета досягнута, поставлені завдання виконані.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Що таке хмарне сховище? – https://aws.amazon.com/ru/what-is-cloud-storage/.

2. Що таке файлообмінник і як ним користуватися? – https://drigin.com/article/chto-takoe-fajloobmennik.html.

3. Файлообмінники. Зберігання та обмін файлами? – http://www.introweb.ru/inews/soft/news9591.php.

4. Безпечний обмін даними через Інтернет? – http://www.taurion.ru/ie6/10/2.

5. Що таке Файлообмінники і як ними користуватися – https://mypclife.ru/chto-takoe-fayloobmenniki-i-kak-imi-polzovatsya/.

6. 12 Best File Hosting Services (2020): Free Storage & Sharing? – https://www.hostingadvice.com/how-to/best-file-hosting-services/.

7. What is File Sharing Security? – https://digitalguardian.com/blog/what-file-sharing-security.

8. Що таке файлообмінник | Відповідь на питання – https://unotices.com/page-answer.php?id=34299.

9. Кращі безкоштовні Файлообмінники без реєстрації 2019 – https://comhub.ru/luchshie-besplatnye-fajloobmenniki-bez-registratsii.

10. Javascript - фреймворки: тенденції 2019 року – https://habr.com/ru/company/plarium/blog/433926/.

11. Vue.js: особливості, застосування і відмінності від інших Javascript фреймворків – https://stfalcon.com/ru/blog/post/vue-js-guide-to-tech.

12. Фреймворк - важливий інструмент програміста – https://fructcode.com/ru/blog/features-of-popular-frameworks-html-css-php-and-python-frameworks.

# ДОДАТКИ

## Додаток А

### *Index.vue*

Компонент головної сторонки сайту.

```
<template>
    <v-container>
        <div class="title-page">
            Останні оновлення
            <v-spacer></v-spacer>
            <button         class="button         material-icons         left
active">reorder</button>
            <button class="button material-icons right">apps</button>
        </div>
        <v-divider></v-divider>
        <div v-for="i in 5" :key="i">
            <v-row>
                <v-col cols="1" class="avatar pb-2">
                    <img src="/img/user.png">
                </v-col>
                <v-col class="nickname pb-2">
                    <div class="name">Misha Otroshchenko</div>
                    <div class="time">Завантажив файли 10 годин тому</div>
                </v-col>
            </v-row>
            <v-row>
                <v-col cols="1" class="vertical-line pt-0">
                    <div class="line"></div>
                </v-col>
                <v-col class="pt-0">
                    <div class="file mb-3" v-for="i in 3" :key="i">
                        <img
src="/img/file.png">секретні_матеріали_пентагону.docx
                        <v-spacer></v-spacer>
                        <button class="button material-icons">share</button>
                        <button            class="button            material-
icons">cloud_download</button>
                        <button            class="button            material-
icons">remove_red_eye</button>
                    </div>
                </v-col>
            </v-row>
        </div>
    </v-container>
</template>

<script>
    export default {

    }
</script>

<style lang="css" scoped>
    .title-page {
        font-weight: 500;
        font-size: 18px;
        color: #626363;
        display: flex;
        align-items: center;
```

```
        margin-bottom: 10px;
    }
    .title-page .button {
        margin-left: 0px;
    }
    .title-page .button.left {
        border-top-right-radius: 0px !important;
        border-bottom-right-radius: 0px !important;
    }
    .title-page .button.right {
        border-top-left-radius: 0px;
        border-bottom-left-radius: 0px;
    }
    .title-page .button.active {
        background: #CBCBCB;
    }
    .nickname {
        font-weight: 400;
        line-height: 28px;
    }
    .nickname .name {
        font-size: 16px;
    }
    .avatar img {
        width: 100%;
        border-radius: 50%;
    }
    .vertical-line .line {
        width: 2px;
        background: #6F6F6F;
        height: 95%;
        margin: 0 auto;
    }
    .file {
        padding: 5px;
        border-radius: 5px;
        background: #fafafa;
        border: 1px solid rgb(231, 231, 231);
        color: #3C3C3C;
        font-weight: 300;
        font-size: 14px;
        display: flex;
        align-items: center;
    }
    .file:hover {
        filter: drop-shadow(0px 0px 2px rgba(0, 0, 0, 0.25));
    }
    .file img {
        height: 30px;
        margin-right: 10px;
        margin-left: 5px;
    }
</style>
```

### *Files.vue*

Компонент заватаження файлів.

```
<template>
    <div>
        <!-- <v-row>
            <v-col cols="4" v-for="i in 5" :key="i">
                <div class="icon">
                    <img src="/img/file_big.png">
```

```
                </div>
                <div class="description">
                    <div class="title">title</div>
                </div>
            </v-col>
        </v-row> -->

        <div class="container">
            <div class="large-12 medium-12 small-12 cell">
                <label>File
                    <input
                        multiple
                        type="file"
                        id="file"
                        ref="file"
                        v-on:change="handleFileUpload()"
                    />
                </label>
                <br>
                <div v-for="(item, index) in files" :key="index" class="file-
item mt-4">
                    <div    class="name">Файл:    {{    index    }}    -    {{
((item.size/1024)/1024).toFixed(2) + ' МБ' }}</div>
                    <input type="text" v-model="item.title"><br>
                    <textarea v-model="item.description"></textarea><br>
                    <progress                                          max="100"
:value.prop="item.uploadPercentage"></progress>
                    <hr>
                </div>
                <v-btn class="mt-4" @click="submitFile()">Завантажити</v-btn>
            </div>
        </div>
    </div>
</template>

<script>
export default {
    data() {
        return {
            files: []
        }
    },
    methods: {
        handleFileUpload() {
            let selectFiles = this.$refs.file.files;
            for(let i = 0; i < selectFiles.length; i++) {
                this.files.push({
                    title: selectFiles[i].name,
                    description: "Тест " + i,
                    file: selectFiles[i],
                    uploadPercentage: 0,
                    size: selectFiles[i].size
                })
            }
        },
        submitFile() {
            let formData = new FormData();
            for (let index = 0; index < this.files.length; index++) {
                this.uploadFile(this.files[index])
            }
        },
        uploadFile(fileItem) {
            let formData = new FormData();
```

```
            formData.append('file', fileItem.file);
            formData.append('title', fileItem.title);
            formData.append('description', fileItem.description);
            axios.post('/api/file-progress', formData, {
                    headers: {
                        'Content-Type': 'multipart/form-data'
                    },
                    onUploadProgress: function(progressEvent) {
                        fileItem.uploadPercentage                                    =
parseInt(Math.round((progressEvent.loaded / progressEvent.total) * 100));
                    }.bind(this)
                }
            ).then(() => {
                console.log('SUCCESS!!');
                return true;
            })
            .catch(() => {
                console.log('FAILURE!!');
                return false;
            });
        }
    }
}
</script>

<style lang="css" scoped>
    progress {
        width: 100%;
    }
    .file-item {
        padding: 10px;
        margin-bottom: 10px;
    }
    .file-item .name {
        font-weight: bold;
    }
    input, textarea {
        margin-top: 10px;
        padding: 5px;
        border: 1px solid silver;
        width: 100%;
    }
    textarea {
        height: 70px;
    }
</style>
```

***app.js***

Скрипіт підключення головних модулів сайту.

```
require('./bootstrap');

import Vue from 'vue'
import vuetify from './plugins/vuetify'
import router from "./routes"
import store from "./store"

import AppComponent from './views/site/App';

const app = new Vue({
    el: '#app',
    components: {
        AppComponent
    },
```

```
    vuetify,
    store,
    router
});
```

***store.js***

Скрипіт відправлення та отримання запитів від серверу на клієнтську частину.

```
import Vue from 'vue'
import Vuex from 'vuex'
Vue.use(Vuex)

export default new Vuex.Store({
    state: {
        status: '',
        token: localStorage.getItem('token') || '',
        tokenAdmin: localStorage.getItem('tokenAdmin') || '',
        user: JSON.parse(localStorage.getItem('user')) || null
    },
    mutations: {
        auth_request(state) {
            state.status = 'loading'
        },
        auth_user_success(state, token, user_data) {
            state.status = 'success'
            state.token = token
            state.user = user_data
        },
        auth_admin_success(state, token) {
            state.status = 'success'
            state.tokenAdmin = token
        },
        auth_error(state) {
            state.status = 'error'
        },
        logout(state) {
            state.status = ''
            state.token = ''
            state.user = ''
        },
        user_data(state, user){
            state.user = user
        },
    },
    actions: {
        login({commit}, user) {
            return new Promise((resolve, reject) => {
                commit('auth_request')
                axios({url: '/api/login', data: user, method: 'POST' })
                .then(resp => {
                    const token = resp . data . Access _ token
                    const user_data = resp . data . user
                    localStorage . setItem ('user', JSON . stringify
(user_data))
                    localStorage . setItem ('token', token)
                    axios . defaults . headers . common ['Authorization'] =
token
                    commit(' auth _ user _ success ', token, user_data)
                    resolve(resp)
                })
```

```
            .catch(err => {
                commit(' auth _ error')
                localStorage.removeItem('token')
                localStorage.removeItem('user')
                reject(err)
            })
        })
    },
    loginAdmin({commit}, user) {
        return new Promise((resolve, reject) => {
            axios({url: '/api/login-admin', data: user, method: 'POST' })
            .then(resp => {
                const token = resp.data.access_token
                localStorage.setItem('tokenAdmin', token)
                axios.defaults.headers.common['Authorization'] = token
                commit('auth_admin_success', token)
                resolve(resp)
            })
            .catch(err => {
                commit('auth_error')
                localStorage.removeItem('tokenAdmin')
                reject(err)
            })
        })
    },
    register({commit}, user) {
        return new Promise((resolve, reject) => {
            commit('auth_request')
            axios({url: '/api/register', data: user, method: 'POST' })
            .then(resp => {
                const token = resp . ata . Access _ token
                const user_data = resp.data.user
                localStorage.setItem('user', JSON.stringify(user_data))
                localStorage.setItem('token', token)
                axios.defaults.headers.common['Authorization'] = token
                commit('auth_success', token, user)
                resolve(resp)
            })
            .catch(err => {
                commit('auth_error', err)
                localStorage.removeItem('token')
                localStorage.removeItem('user')
                reject(err)
            })
        })
    },
    logout({commit}) {
        return new Promise((resolve, reject) => {
            commit('logout')
            localStorage.removeItem('token')
            localStorage.removeItem('user')
            localStorage.removeItem('tokenAdmin')
            delete axios.defaults.headers.common['Authorization']
            resolve()
        })
    },
    user({commit}, user) {
        localStorage.setItem('user', JSON.stringify(user))
        commit('user_data', user)
    }
},
getters: {
    isLoggedIn: state => !!state.token,
```

```
        isLoggedInAdmin: state => !!state.tokenAdmin,
        authStatus: state => state.status,
        authUser: state => state.user,
    }
})
```