ЖЮМІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ПРИРОДОКОРИСТУВАННЯ

ФАКУЛЬТЕТ МЕХАНІКИ, ЕНЕРГЕТИКИ
ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА

другого (магістерського) рівня вищої освіти

на тему: **“ Система інвентаризації даних інтернет-магазинів із застосуванням хмарного сховища ”**

Виконав: ст. гр. Іт-61_____

Спеціальності 126 – «Інформаційні системи та технології»_____
(шифр і назва)

_____Задерецький Роман Михайлович_____
(прізвище та ініціали)

Керівник: __к.т.н., доц. Луб П.М._____
(прізвище та ініціали)

Рецензент: _____
(прізвище та ініціали)

ДУБЛЯНИ-2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ПРИРОДОКОРИСТУВАННЯ

ФАКУЛЬТЕТ МЕХАНІКИ, ЕНЕРГЕТИКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

другий (магістерський) рівень вищої освіти
126 – «Інформаційні системи та технології»

"ЗАТВЕРДЖУЮ"
Завідувач кафедри_____
д.т.н., проф. А.М. Тригуба
"____"_____2024 р.

ЗАВДАННЯ

на кваліфікаційну роботу студенту

Задерецькому Роману Михайловичу

1. Тема роботи: «Система інвентаризації даних інтернет-магазинів із застосуванням хмарного сховища»

Керівник роботи Луб Павло Миронович, к.т.н., доцент
Затверджені наказом по університету 12 вересня 2024 року № 616/к-с.

2. Строк подання студентом роботи 06.12.2024 р.
3. Початкові дані до роботи: 1. Будова та функціонал хмарних сервісів. 2. Вимоги до побудови фронт та бекенду. 3. Вимоги до побудови клієнт-серверної системи. 4. Сервіс хмарного сховища File linker.

4. Зміст розрахунково-пояснювальної записки:

1. Аналіз завдань та способів інвентаризації даних.

2. Структурування систем інвентаризації даних.

3. Методика побудови та головні складові інформаційної системи.

4. Реалізація системи інвентаризації даних.

5. Охорона праці та безпека в надзвичайних ситуаціях.

Висновки та пропозиції.

Бібліографічний список.

Додатки.

5. Перелік графічного матеріалу: 1 та 2 – Тема, мета, завдання роботи; 3 – Аналіз головних понять з інвентаризації ІТ-активів; 4 – Аналіз хмарних сховищ та їх типів; 5 – Аналіз веб-сайти із сервісом хмарного сховища; 6 – Засоби реалізації системи інвентаризації даних; 7 – Фреймворки та архітектура організації коду; 8 – Структурно-функціональна модель системи; 9 – ER-діаграма та побудова бази даних; 10 – Архітектура проекту системи інвентаризації даних; 11 – Реалізація клієнтської та серверної частини; 12 та 13 – Результати застосування системи інвентаризації даних; 14 – Головні висновки.

6. Консультанти з розділів:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1, 2, 3, 4	*Луб П.М., доцент кафедри інформаційних технологій*		
5	*Городецький І.М., доцент кафедри фізики, інженерної механіки та безпеки виробництва*		

7. Дата видачі завдання 12 вересня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту	Строк виконання етапів роботи	Приміт-ка
1.	*Написання першого розділу та означення головних завдань роботи*	12.09 - 01.10.24	
2.	*Виконання другого розділу та формування головних показників для розрахунків*	12.09 - 01.10.24	
3.	*Виконання третього розділу, розрахунків та розробка листів*	01.10 - 01.11.24	
4.	*Написання розділу: «Охорона праці та безпека в надзвичайних ситуаціях»*	01.10 - 01.11.24	
5.	*Завершення оформлення розрахунково-пояснювальної записки та аркушів графічної частини*	01.11 - 01.12.24	
6.	*Завершення роботи в цілому*	01-10.12.24	

Студент _____ Задерецький Р.М.
(підпис)

Керівник роботи _____ Луб П.М.
(підпис)

УДК: 004.738.5:004.6:004.451.3

Кваліфікаційна робота: 71 с. текст. част., 26 рис., 12 табл., 13 слайдів, 25 джерел.

Система інвентаризації даних інтернет-магазинів із застосуванням хмарного сховища. Задерецький Р.М. Кафедра ІТ. – Дубляни, Львівський НУП, 2024.

Виконано аналіз завдань та способів інвентаризації даних. Зокрема, проаналізовано головні поняття з інвентаризації ІТ-активів. Наведено аналіз хмарних сховищ та їх типів, а також аналіз відомих систем та інструментів для хмарних сховищ.

Наведено структурування систем інвентаризації даних. Зокрема, описано функціональні особливості мов програмування, вибір засобів реалізації та представлено побудову структури інтерфейсу та серверної частини.

Виконано вибір засобів та побудовано базу даних.

Наведено методику побудови та головні складові інформаційної системи. Зокрема, описано структурно-функціональну модель ІС. Наведено діаграму варіантів використання та діяльності. Розкрито ER-діаграму та побудову бази даних.

Описано практичне використання системи інвентаризації даних. Зокрема, розкрито принцип реалізації архітектури клієнт-серверної взаємодії. Наведено результати застосування системи інвентаризації даних.

Запропоновано заходи з охорони праці за безпеки в надзвичайних ситуаціях.

Ключові слова: система, інвентаризація, дані, файли, веб-додаток, хмарне сховище, фронтенд, бекенд.

ЗМІСТ

ВСТУП

Швидкий розвиток Інтернет-послуг спричинив сплеск товарообігу, а це в свою чергу – складності із процесами контролю відстеження та управління інвентаризацією товарів. Для більшості сучасних підприємств фінансово доступним стало використання інформаційних технологій для ведення своєї бізнесової діяльності. Роль інформаційних систем зросла багатократно, а це зумовило появу нових типів завдань щодо систематизаці, опрацювання та інвентаризації даних.

Використання таких інструментів значно спростило діяльність та зумовило масштабування товарообігу, дозволило швидко відскановувати тег товару, щоб перевірити його наявність в системі матеріальних активів компанії тощо. Інвентаризація дозволяє встановити фактичну наявність товару чи послуги завдяки перерахунку залишків та перевірки наявних записів тощо. В результаті цього, стала обов'язковою процедура перегляду залишкових ресурсів та звітності а це також забезпечує перевірку достовірності показників фінансового балансу.

З іншого боку за останні кілька років розробка мобільних додатків стала індустрією, яка розвивається дуже стрімко. Сьогодні існує 2,3 мільйони розробників мобільних додатків, які прагнуть задовольнити попит різних галузей [2]. За даними Apple, у 2023 році в магазині програм Apple було зареєстровано 2,25 мільйона додатків, на які припадало 70 мільярдів завантажень та 9 мільярдів доларів, виплачених розробникам.

Завдяки цим типам галузевих цифр незабаром стає зрозуміло, що розробка інформаційних систем інвентаризації даних у спеціалізованих додатках є ключовим фактором успіху бізнесу. Виходячи з цього, ідеальним рішенням було б поєднати галузь маркетингу та цифрової інвентаризації – розробка інформаційних систем з проблематикою інвентаризації даних. Користувач такого програмного застосунку матиме можливість оперувати документами інвентарного опису на цифрових пристроях, а також проводити процедуру інвентаризації за допомогою зчитування QR коду тощо.

Відповідно до цих положень сформовано мету кваліфікаційної роботи.

Мета роботи – підвищення ефективності обміну даними та файлами, їх обліку, структурування та оновлення із використанням хмарного сховища.

Завдання роботи:

- проаналізувати завдання та способи інвентаризації даних;
- здійснити вибір засобів реалізації та описати побудову структури сервісу;
- описати структурно-функціональну модель, діаграму варіантів використання та ER-діаграму побудови бази даних;
- представити результати застосування системи інвентаризації даних.

Об'єктом роботи є система інвентаризації даних, користувачі, інтернет-магазини, хмарні сховища, процеси отримання та обміну даними й файлами.

Предметом роботи є шляхи доступу до даних, показники файлів, міграція даних у хмарних сховищах, система інвентаризації.

Новизна роботи:

- описано способи інвентаризації даних;
- описано побудова структури сервісу інвентаризації даних;
- розкрито методологію вибору засобів розробки;
- представлено результати застосування системи.

Практичне значення одержаних результатів. Результатом кваліфікаційної роботи є структура веб-додатку що реалізовує систему інвентаризації даних для завантаження файлів, даних та їх зберігання на віддаленому сервері. Запропонована система дає змогу реалізувати більш надійний і безпечний спосіб обміну й доступу до файлів і даних.

РОЗДІЛ 1
АНАЛІЗ ЗАВДАНЬ ТА СПОСОБІВ ІНВЕНТАРИЗАЦІЇ ДАНИХ

1.1. Аналіз головних понять з інвентаризації IT-активів

Ефективне управління інвентаризацією IT-системи має важливе значення для будь-якої організації для підтримки оптимальної продуктивності, безпеки та ефективності мережі. Але без правильного вибору програмного забезпечення, провести інвентаризацію може бути складним завданням.

Основні функції, які слід враховувати в програмному забезпеченні "мережевої" інвентаризації інтернет-магазинів:

- автоматизоване виявлення даних інтернет-магазинів;
- моніторинг і оповіщення в реальному часі;
- комплексна звітність і аналітика;
- інтеграція з іншими інструментами управління IT;
- масштабованість і вартість.

Автоматизоване виявлення мережі гарантує, що всі пристрої в мережі ідентифікуються та відстежуються, зменшуючи ручні зусилля та ризик непомічених активів.

Безумовно, сьогодні існує досить багато рішень для виконання такого завдання, але слід розглянути можливості саме використання PRTG (розробник Paessler). ***Paessler PRTG*** – це потужне і зручне програмне забезпечення, призначене для моніторингу мереж із обміном даними. Воно дозволяє відстежувати різноманітні параметри мережі, такі як:

- **Пропускна здатність:** Відображає хто і як використовує мережу із

обміном даних, які програми та пристрої найбільше завантажують канал тощо.

- **Навантаження на пристрої:** Моніторинг процесорів, пам’яті, дискової підсистеми та інших ресурсів ваших серверів і робочих станцій.
- **Доступність сервісів:** Відстеження роботи веб-сайтів, серверів електронної пошти, FTP-серверів та інших служб.
- **Проблеми з підключенням:** Швидке виявлення обривів кабелів, збоїв у роботі мережевого обладнання та інших проблем, які можуть призвести до перебоїв в роботі.
- **Простий у використанні:** Інтуїтивний інтерфейс і автоматичне виявлення мережі дозволяють швидко розгорнути систему моніторингу.
- **Гнучкий:** Велика кількість датчиків дозволяє відстежувати практично будь-які параметри мережі.
- **Інформативний:** Наочні графіки і звіти допомагають швидко оцінити ситуацію і виявити причини проблем.
- **Надійний:** PRTG працює цілодобово і автоматично сповіщає вас про будь-які проблеми.

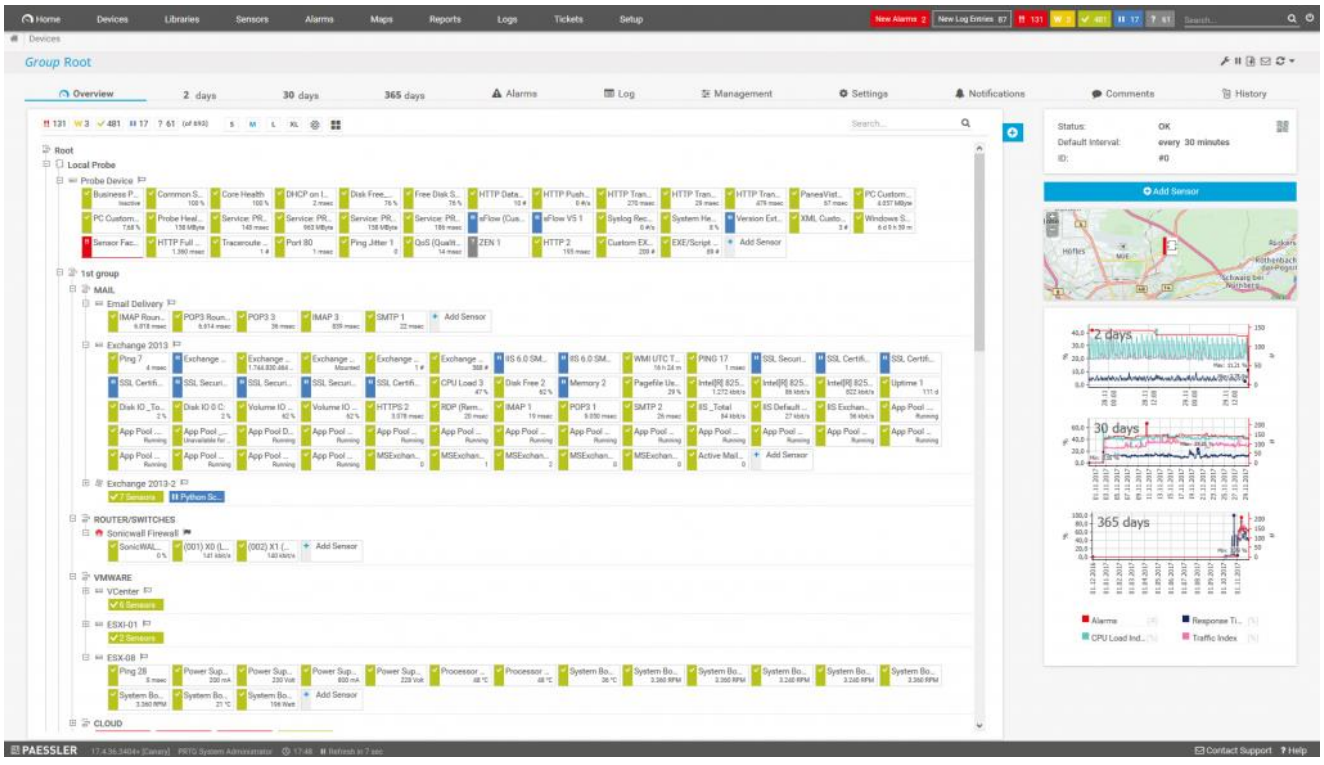


Рисунок 1.1. – Вигляд робочого вікна Paessler PRTG Network Monitor

Функціонал PRTG для мережевої інвентаризації **дозволяє контролювати все:**

- моніторинг систем, пристроїв, трафіку та додатків за всіма напрямками;
- моніторинг локальних мереж, глобальних мереж, серверів тощо;
- відстежувати веб-сайти, програми, служби тощо.

Інтегровані технології:

- підтримуються всі важливі технології – Ping, SNMP, WMI, SSH, HTTP-запити тощо;
- протоколи потоку (IPFIX, jFlow, sFlow, NetFlow).

Автоматичне виявлення мережі:

- швидке налаштування завдяки автоматичному виявленню мережі;
- інтелектуальне виявлення пристроїв мережі та обладнання із даними;
- автоматичне налаштування наборів датчиків.

Карти та інформаційні панелі:

- візуалізація мережі так, як замовнику потрібно;
- використання карт в реальному часі з поточною інформацією;
- створення інформаційних панелей за допомогою дизайнера карт PRTG.

Сповіщення

- Миттєво отримуйте сповіщення про проблеми або незвичні показники
- Спеціальні порогові значення, адаптовані до ваших потреб моніторингу
- Скористайтеся перевагами вбудованих методів сповіщення (електронна пошта, push, HTTP-запити тощо)

Саме завдяки своїй гнучкості та широкому набору функцій, він може ефективно використовуватися для інвентаризації мережі, що є невід'ємною частиною стратегій IT Asset Management (ITAM) та IT Service Management (ITSM).

Рисунок 1.2. – Візуалізація моніторингової панелі Paessler PRTG Network Monitor

Головні IT-системи для інвентаризації даних та їх особливості:

ITAM (IT Asset Management) – це комплекс заходів, спрямованих на управління життєвим циклом IT-активів. Це включає в себе інвентаризацію, облік, оцінку вартості, управління конфігурацією, технічне обслуговування та утилізацію. Мета ITAM – оптимізувати використання IT-ресурсів, знизити витрати та забезпечити відповідність нормативним вимогам.

ITSM (IT Service Management) – це набір процесів, які забезпечують надання IT-послуг кінцевим користувачам. ITSM охоплює такі процеси, як управління інцидентами, управління проблемами, управління змінами, управління рівнем послуг тощо. Мета ITSM – забезпечити безперебійну роботу IT-систем та максимальну задоволеність користувачів.

Переваги використання PRTG в рамках ITAM та ITSM:

- Збільшення видимості: завжди доступна інформація про те, що відбувається в IT-інфраструктурі.
- Зменшення ризиків: PRTG допомагає виявити потенційні проблеми до того, як вони призведуть до серйозних наслідків.

- Оптимізація витрат: Завдяки точній інвентаризації можна уникнути непотрібних витрат на ліцензії та обладнання.
- Покращення продуктивності: Швидке виявлення та усунення проблем дозволяє забезпечити безперебійну роботу IT-систем.

Використання PRTG для інвентаризації:

- Створення інвентаризаційних звітів: Регулярно створювані звіти про всі пристрої в мережі, встановлене програмне забезпечення та ліцензії.
- Моніторинг ліцензій на програмне забезпечення: Відстеження кількості встановлених ліцензій і порівняння їх із фактичною кількістю використаних ліцензій.
- Виявлення несанкціонованого програмного забезпечення: PRTG може виявити програмне забезпечення, яке не було встановлено офіційно.
- Планування оновлень: Створення планів оновлення програмного забезпечення на основі даних інвентаризації.

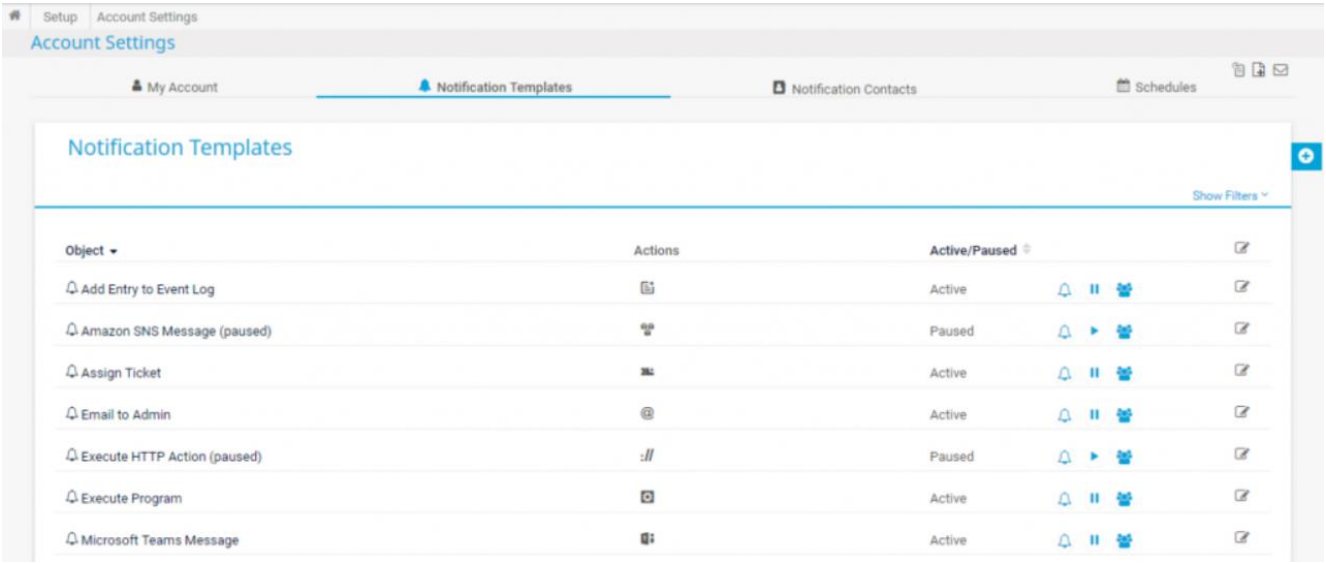


Рисунок 1.3. – Відображення збору та інвентаризації даних

Переваги застосування PRTG для інвентаризації даних:

- Автоматичне виявлення: PRTG автоматично сканує мережу та виявляє нові пристрої, програмне забезпечення та мережеві сервіси. Це значно зменшує ручну роботу.
- Комплексний огляд: Надається детальна інформація про кожен

пристрій у мережі: виробник, модель, IP-адреса, операційна система, встановлене програмне забезпечення та багато іншого.

- Моніторинг в реальному часі: PRTG дозволяє відстежувати зміни в мережі в режимі реального часу. Надання повідомлень, коли з’являються нові пристрої або видаляються старі.
- Гнучкі звіти: Можливість створювати різноманітні звіти, щоб отримати огляд інвентарю, відстежувати ліцензії на програмне забезпечення та планувати оновлення.
- Інтеграція з іншими системами: PRTG легко інтегрується з іншими системами ITAM та ITSM, такими як ServiceNow, Jira та іншими, забезпечуючи єдину точку доступу до всієї інформації про вашу IT-інфраструктуру.

1.2. Аналіз хмарних сховищ та їх типів

Проблема своєчасного отримання інформації в бізнесі існувала завжди. Сьогоднішні облікові програми потужні і багатофункціональні, а на шляху потоку цієї інформації стає оператор ПК та засоби автоматичного введення-виведення даних [6, 9, 12, 21].

Найбільш актуальні аспекти обміну даними:

– обмін між територіально віддаленими один від одного місцями введення інформації;

– обмін даними між системами обліку з різним призначенням (оперативний облік, управлінський облік);

– отримання консолідованого балансу з різних інформаційних баз (зведений баланс по дочірнім підприємствам).

Загальними вимогами, що пред'являються до систем обміну даними, є забезпечення одиничного введення інформації, використовуваної в декількох базах даних, дотримання загальних правил цілісності бази даних, стійкість системи до збоїв і захищеність від несанкціонованого доступу.

Для забезпечення даних вимог існують такі сервіси як файлоомінники, або хмарні сервіси зберігання даних.

Хмарне сховище – це модель хмарних обчислень, яка передбачає зберігання даних в Інтернеті за допомогою постачальника обчислювальних ресурсів, який надає сховище даних як сервіс і забезпечує управління та обмін файлами. Це забезпечує гнучкість, глобальну масштабованість і надійність. Дані доступні в будь-який час і в будь-якому місці [14].

У загальних рисах хмарне сховище даних являє собою віртуальний файлообмінник, або сервер з даними, доступ до яких можна отримати з будь-якого пристрою, підключеного до мережі і використовуваному хмарного сервісу.

Таблиця 1.1 – Переваги хмарного сховища

№ п/п	Перевага	Опис
1	Сукупна вартість володіння	Завдяки хмарному сховищу не потрібно купувати устаткування, виділяти ресурси для сховища або витрачати кошти зберігання даних. Можна додавати або видаляти ресурси на вимогу, швидко змінювати продуктивність та термін зберігання. Це дозволяє забезпечити економію при великих обсягах.
2	Час для розгортання	Коли група розробників готова до запуску проекту, інфраструктура не повинна їх стримувати. Хмарне сховище дозволяє IT-фахівцям швидко виділяти необхідний простір для зберігання даних саме тоді, коли це потрібно. В результаті IT-фахівці можуть зосередитися на вирішенні складних проблем, пов'язаних з додатками, а не на питаннях управління системами зберігання даних.
3	Управління інформацією	Централізоване сховище в хмарі створює величезні можливості для нових прикладів використання. Використовуючи політики управління життєвим циклом в хмарному сховищі, можна вирішувати важливі завдання, пов'язані з управлінням інформацією, включаючи автоматичний розподіл за рівнями або блокування даних з метою дотримання вимог.

Файлообмінник – сервіс, що надає користувачеві місце під його файли і

цілодобовий доступ до них через Інтернет. Такий сервіс дозволяє зручно обмінюватися файлами.

На спеціальній сторінці файлообмінника (найчастіше на головній) користувач завантажує файл на сервер файлообмінника, а він віддає користувачеві постійне посилання, яку він може розсилати по електронній пошті, публікувати в блогах, на форумах або пересилати через системи миттєвого обміну повідомленнями. Перейшовши по такому посиланню, будь-який інший користувач може завантажити початковий файл [13].

Таблиця 1.2 – Типи хмарних сховищ

№ п/п	Назва	Опис
1	Об'єктне сховище	Додатки, розроблені в хмарі, як правило, використовують такі переваги об'єктного сховища, як широкі можливості масштабування і зберігання властивостей об'єктів у вигляді метаданих. Об'єктні сховища, наприклад Amazon Simple Storage Service (S3), ідеально підходять для розробки з нуля сучасних додатків, для яких потрібна гнучкість і можливість масштабування. Крім того, ці сховища можна використовувати для імпорту даних з існуючих сховищ з метою аналітики, резервного копіювання або архівації.
2	Файлове сховище	Деяким програмам потрібен доступ до передачі файлів, отже, їм необхідна файлова система. Даний тип сховища часто підтримується сервером сховищ, підключеним до мережі (NAS).
3	Блочне сховище	Інші корпоративні додатки, наприклад бази даних або системи планування ресурсів підприємства (ERP), часто потребують виділене сховище з низькими затримками для кожного з вузлів. Таке сховище працює аналогічно сховища з прямим підключенням (DAS) або мережі зберігання даних (SAN).

Однак на відміну від традиційного сервера, що завантажується в хмарні сховища інформація розміщена одночасно на безлічі мережевих серверів, в тому числі, що знаходяться за тисячі кілометрів на іншому кінці Землі.

Зберігання даних в хмарі дозволяє принципово переглянути три аспекти

своєї діяльності.

Існує також три типи хмарних сховищ даних: об'єктні сховища, файлові сховища і блокові сховища. Кожен з них пропонує свої переваги, які підходять для конкретних прикладів використання.

Питання забезпечення надійного зберігання, безпеки та доступності критично важливих корпоративних даних мають першорядну важливість. При розгляді варіанту зберігання даних в хмарі існує кілька фундаментальних вимог.

Вимоги до хмарного сховища [6, 17, 18]:

Надійність: Дані повинні зберігатися із пересторогеністю. В ідеалі вони повинні бути розподілені між кількома об'єктами і кількома пристроями в рамках кожного з об'єктів. Стихійні лиха, людський фактор або механічні несправності не повинні призводити до втрати даних.

Доступність: Всі дані повинні бути доступними в разі необхідності, але існує різниця між виробничими даними і архівами. Ідеальне хмарне сховище пропонує оптимальне поєднання між часом отримання даних і вартістю.

Безпека: Всі дані повинні шифруватися – як при зберіганні, так і при передачі. Дозволи та контроль доступу повинні працювати в хмарі точно так само, як і в локальних сховищах даних.

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

Перший і головний – це недостатнє опрацювання 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 файлообмінниках 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.

1.3. Аналіз відомих систем та інструментів для хмарних сховищ

Аналіз web-сайтів конкурентів, аналіз цільової аудиторії майбутнього сайту, виявлення «сильних» і «слабких» сторін проекту відіграють важливу роль у розробці своїх та адаптованіших програмних продуктів. Розробка сайту завжди починається з виконання подібних завдань. Перед початком розробки поставленої задачі потрібно провести аналіз існуючих аналогів.

Нами проаналізовано найбільш популярні веб-сайти із сервісом хмарного сховища:

- ***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 рисунку 1.4.

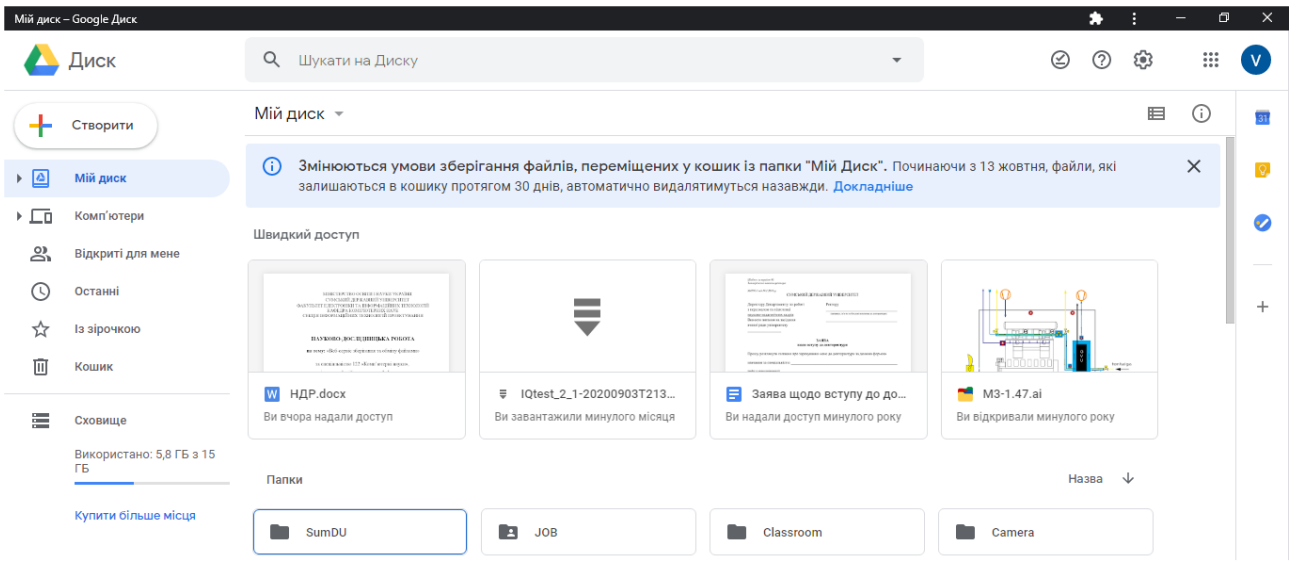


Рисунок 1.4. – Головна сторінка Google Диску

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 сайт «Mega» що наведений на рисунку 1.5.

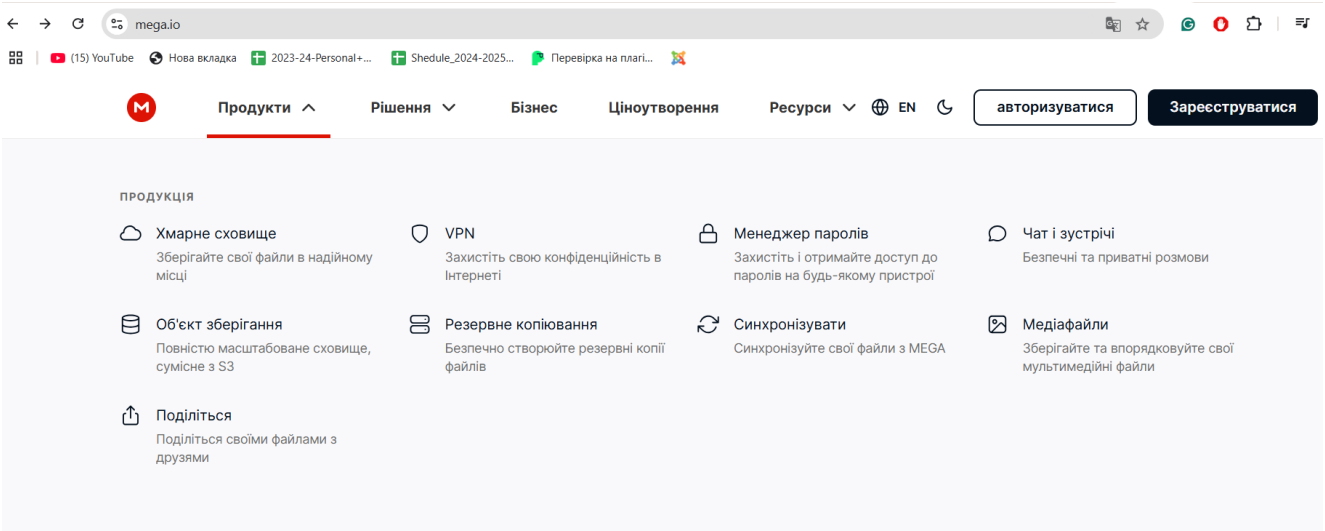


Рисунок 1.5. – Головна сторінка «MEGA cloud storage»

Цей 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.

Також нами проаналізовано сайт «Dropbox» головна сторінка якого зображена на рисунку 1.6.

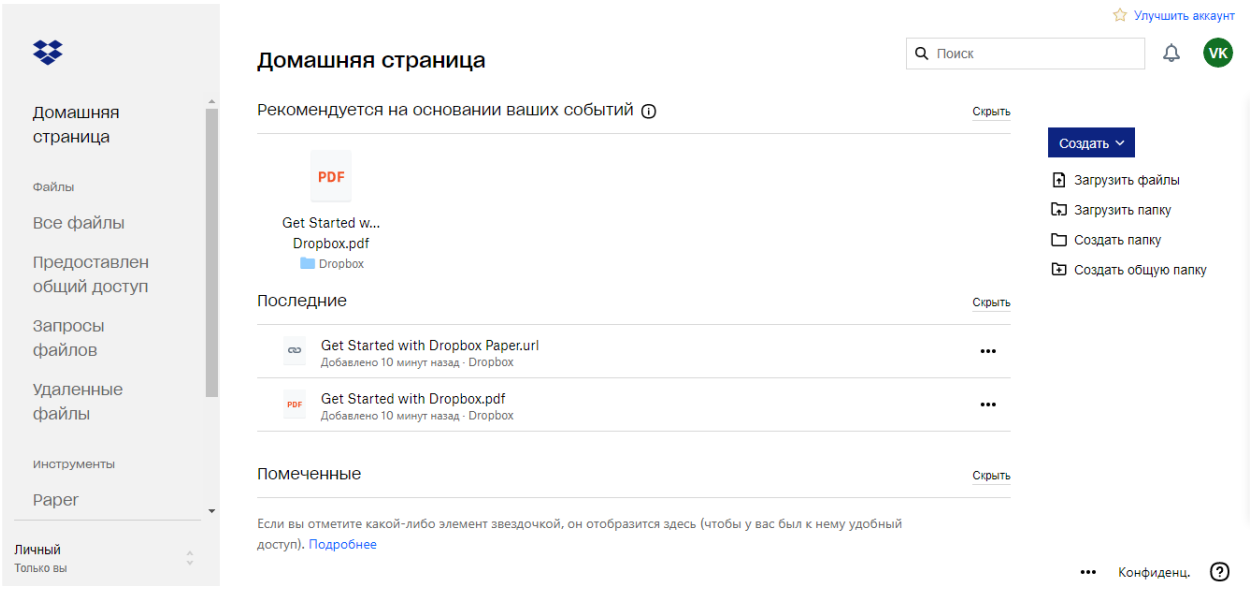


Рисунок 1.6. – Головна сторінка «Dropbox»

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 рисунку 1.7.

Отже, в результаті проведеного аналізу сайтів зі схожою тематикою і призначенням було вирішено розробити сайт який не матиме недоліків виявлених на сайтах представлених в таблиці 1.3.

Дана розробка матиме всі переваги даних сайтів, а також матиме унікальний дизайн та зручний інтерфейс.

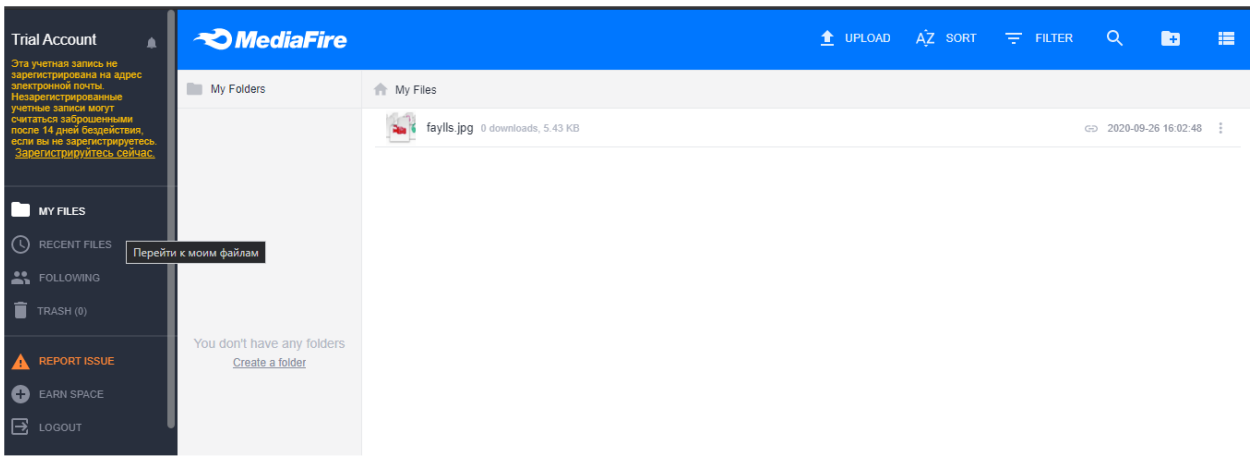


Рисунок 1.7. – Інтерфейс mediafire.com

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

Таблиця 1.3 – Аналіз сайтів із сервісами хмарних сховищ

Критерії	***Google Диск***	***Mega***	***Dropbox***	***Mediafire***
Опис завантажених файлів	+	+	+	+
Зручний інтерфейс	+	-	-	+
Не обмежений розмір сховища	-	-	-	-
Зручний пошук	+	-	+	+
Можливість підписатись на інших користувачів	+	-	+	-
Рейтнгове оцінювання	-	+/-	+	-
Публікувати не лише за посиланням	+	-	+	-

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.

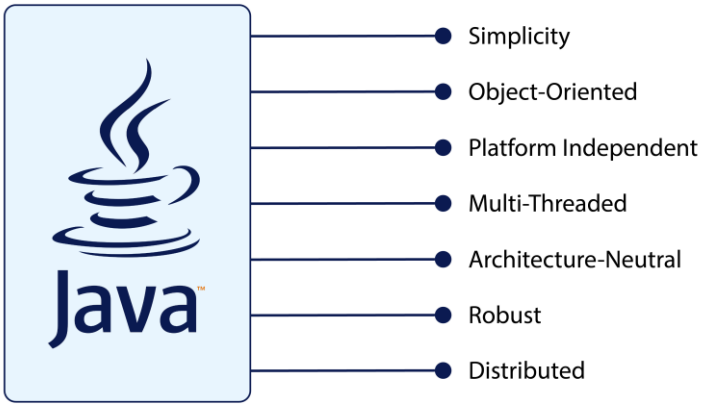
Отже, в результаті проведеного аналізу сайтів зі схожою тематикою і призначенням було вирішено розробити сайт який не матиме недоліків виявлених на сайтах представлених в таблиці. Пропонована розробка матиме всі переваги даних сайтів, а також матиме унікальний дизайн та зручний інтерфейс.

РОЗДІЛ 2
СТРУКТУРУВАННЯ СИСТЕМ ІНВЕНТАРИЗАЦІЇ ДАНИХ

2.1. Функціональні особливості мов програмування

Кількість IT компаній, яким потрібні нові розробники зростає з кожним днем. Це призводить до великого попиту на IT спеціалістів та росту заробітної плати в даній сфері [10, 18].

Освітній центр “YOLO” пропонує перелік 5-ти найпопулярніших мов програмування, на які потрібно звернути увагу.



Надзвичайно поширена мова, адже 90% компаній, що входять в список Fortune (500 найбільших корпорацій світу), так чи інакше використовують в своїх розробках Java. До речі її використовують при розробці операційної системи Android, яка на даний час є найбільш мобільною платформою в світі. Мова була розроблена ще в 1995 році, компанією Oracle і досі залишається на вершині. ЇЇ використовують для розробки десктопних додатків, операційних систем, “back end” систем та багато іншого. Основна її перевага це кросплатформеність.

Хоч і назва цієї мови схожа з попередньою мовою Javascript, це зовсім інша мова програмування. Згідно з сайтом Stackoverflow, Javascript – це найпопулярніша мова програмування серед розробників. Ця мова використовується як одна з основних технологій для створення інтерактивних сайтів разом з HTML та CSS. Адже більшість браузерів використовують саме ці три основні технології. Також, використовуючи її, можна створювати мобільні додатки, ігри та десктопні програми. Тому, якщо ви плануєте розвиватись саме в цих сферах, варто спробувати починати саме з цієї мови програмування [24].

HTML та CSS – це не зовсім мови програмування, і досвідчені програмісти вважають, якщо людина знає лише HTML та CSS, вона не може називати себе програмістом. Проте, ці технології використовуються практично на кожному сайті. Адже саме через них прописується візуальний стиль сайтів, кнопки, іконки та ефекти кожної сторінки. Тому ці мови корисно знати не лише програмістам, але і веб дизайнерам. Тому якщо ви хочете почати свій розвиток в веб розробці чи дизайні, тоді вам в першу чергу варто звернути увагу саме на ці технології.

Популярність цієї мови програмування пов’язана з тим, що її використовують в найпоширенішій платформі для створення сайтів Wordpress, а 80% найбільш відвідуваних сайтів в світі використовують мову PHP тим чи іншим способом. Вона вважається однією з базових та найпростіших мов програмування, яку має знати кожен, хто називає себе програмістом. Не важливо,

чи створюєте ви сайти, складні інтернет магазини чи серверні рішення, вам точно стане в нагоді PHP [11].

Мова, яку розробили ще у 1983 році і на якій створені Microsoft Windows і Google Chrome. Завдяки широкому набору інструментів мова легко адаптується для застосування в різноманітних сферах життя, будь то банківська сфера, розробка ігор, торгівля чи інше. Тому саме на цій мові можна створювати складні комерційні системи з багатьма елементами, так і розробляти прості застосунки та програми. І якщо ви шукаєте мову з широким інструментарієм та функцією, тоді C++ саме для вас.

Об'єктно-орієнтована мова програмування з безпечною системою типізації для платформи .NET. Розроблена Андерсом Гейлсбергом, Скотом Вілтамутом та Пітером Гольде під егідою Microsoft Research (належить Microsoft).

Синтаксис C# близький до C++ і Java. Мова має строгу статичну типізацію, підтримує поліморфізм, перевантаження операторів, вказівники на функції-члени класів, атрибути, події, властивості, винятки, коментарі у форматі XML. Перейнявши багато від своїх попередників — мов C++, Object Pascal, і Smalltalk

— C#, спираючись на практику їхнього використання, виключає деякі моделі, що зарекомендували себе як проблематичні при розробці програмних систем, наприклад, мова C#, на відміну від C++, не передбачає множинне успадкування класів.

Для побудови програми у ***Microsoft Visual Studio*** широко використовують засоби, які надає система. Є дві частини побудови: перша – проектування інтерфейсу з використанням стандартних елементів (компонент) та маніпулювання їхніми розмірами й розташуванням; друга – написання фрагментів програмного коду для виконання завдання. Visual Studio самостійно записує деякі частини програми без зовнішнього втручання, розробникові треба кодувати лише суто свою задачу. Крім того, Visual Studio формує для майбутньої програми потрібну інформацію в файлах.

Microsoft Visual Studio – серія продуктів фірми Майкрософт, які включають інтегроване середовище розробки програмного забезпечення та низку інших інструментальних засобів. Ці продукти дозволяють розробляти як консольні програми, так і програми з графічним інтерфейсом, в тому числі з підтримкою технології Windows Forms, а також веб-сайти, веб-застосунки, веб-служби як в рідному, так і в керованому кодах для всіх платформ, що підтримуються Microsoft Windows, Windows Mobile, Windows Phone, Windows CE, .NET Framework, .NET Compact Framework та Microsoft Silverlight.

Visual Studio включає один або декілька з наступних компонентів:

- ***Visual Basic .NET, а до його появи – Visual Basic;***
- ***Visual C++;***
- ***Visual C#;***
- ***Visual F# (входить до складу Visual Studio 2022);***
- ***Visual Studio Debugger.***

Багато варіантів постачання також включають – Microsoft SQL Server або MSDE Visual Source Safe – файл-серверна система управління версіями.

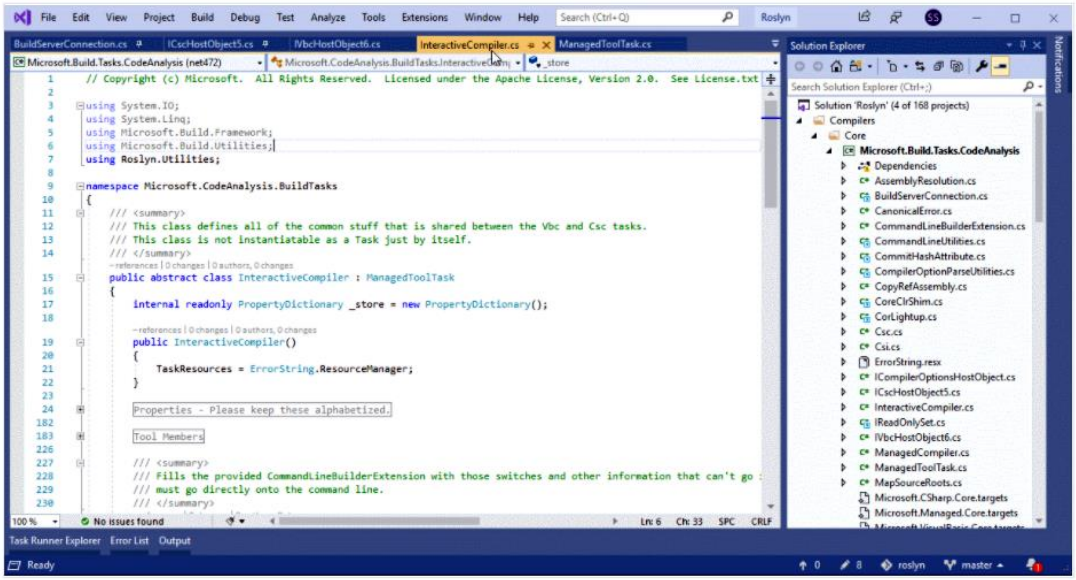


Рис. 2.1. – Інтерфейс вікна Visual Studio з інструментами мови C#

У минулому, до складу Visual Studio також входили продукти:

- ***Visual InterDev;***
- ***Visual J++;***
- ***Visual J#;***
- ***Visual FoxPro.***

Visual Source Safe – файл-серверна система управління версіями.

Для початку, треба підготувати папку (каталог), де будуть зберігатися всі файли майбутньої програми. Це ліпше робити до запуску Visual Studio. Нову папку створити за звичайними правилами операційної системи Windows. Наприклад, запустити інструмент Мій комп'ютер (MyComputer), розташований на робочому столі системи, відшукати потрібний диск та вже наявну папку, відкрити її, після чого через меню вибрати команди *Файл→Створити→Папка*.

2.2. Вибір засобів реалізації

Розроблювана нами інформаційна система представляє з себе сервіс, що надає користувачеві місце під його файли і цілодобовий доступ до них через Інтернет, як правило по протоколу https. Такий сервіс дозволяє зручно

обмінюватись файлами між іншими користувачами Інтернет мережі. На кожній сторінці файлообмінника користувач матиме можливість завантажувати файли на сервер файлообмінника, а додаток віддає користувачеві постійне посилання, яке він може розсилати через 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 нашої 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.

Наведемо головні вимоги до сервісу:

1. Всі користувачі перед використанням повинні зареєструватись в системі.

2. Користувацька частина повинна складатися із класичних сторінок (Головна; Мої файли; Популярні файли; Обране; Кошик).

3. На головній сторінці сайту повинні бути ключові елементи (Пошук файлів; Перегляд останніх публікацій користувачів; Перегляд інформації про файл; Можливість завантажувати файл на сервіс).

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 користуємось через Інтернет щодня, були розроблені веб-розробниками.

Веб-розробка може бути розбита на три шари (табл. 2.1): кодування на стороні клієнта (*фронтенд*), кодування на стороні сервера (*бекенд*) та технології баз даних.

Таблиця 2.1 – Етапи реалізації системи у веб-розробці

№ п/п	Назва	Опис	Технології розробки
1	Клієнтська сторона	Сценарії на стороні клієнта або розробка інтерфейсу стосується всього, що переживає кінцевий користувач безпосередньо. Код клієнта виконується у веб-браузері і безпосередньо стосується того, що бачать люди, відвідуючи веб-сайт. Такі речі, як компонування, шрифти, кольори, меню та контактні форми, керуються фронтендом.	HTML, CSS, Vue.js, Vuetify, JavaScript, React.js, Bootstrap.
2	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.
3	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 таким чином, щоб полегшити його роботу. База даних працює на сервері, і більшість веб-сайтів зазвичай використовують певну форму управління реляційною базою даних	MySQL, PostgresSQL, OracleSQL, MongoDB

2.3. Побудова структури інтерфейсу та серверна частина

Завдання розробника ***frontend*** – кодувати інтерфейс веб-сайту чи програми; тобто частину веб-сайту, яку бачить і взаємодіє користувач. Вони працюють над

розробками, наданими веб-дизайнером, і реалізують їх за допомогою HTML, JavaScript та CSS.

Мови розмітки використовуються для визначення форматування текстового файлу. Іншими словами, мова розмітки повідомляє програмне забезпечення, яке відображає текст, як слід форматувати текст. Мови розмітки є повністю розбірливими для читання – вони містять стандартні слова, але теги розмітки приховані від користувача в кінцевому варіанті коду.

Дві найпопулярніші мови розмітки – HTML та XML. HTML розшифровується як мова розмітки HyperText та використовується для створення веб-сайтів. Додані до звичайного текстового документа, всі теги HTML описують, як цей документ повинен відображатися веб-браузером.

Невідмінна частина будь-якої веб-сторінки – каскадні стилі розмітки сторінки, або CSS. *Стилі* – це в основному сукупність стилістичних правил. Мови аркушів стилів використовуються, доволі буквально, для оформлення документів, написаних мовами розмітки. CSS розшифровується як каскадні таблиці стилів з акцентом на «стиль». Хоча HTML використовується для структури веб-документа, визначає такі речі, як заголовки та абзаци, і дозволяє вставляти зображення, відео та інші медіа, CSS визначає стиль створеного документа.

Розробка серверної частини. Код який створюють розробники backend, гарантує, що все, що будує розробник frontend, є повністю функціональним. Головне завдання розробника серверної частини – переконатися, що сервер, програма та база даних взаємодіють між собою. Для вирішення такої складної задачі розробники використовують серверні мови, такі як PHP, Ruby, Python та Java для створення програми.

Веб-технології стрімко розвиваються, і щоб встигати за швидкістю розвитку нових технологій, у веб-програмістів часто бракує часу для реалізації повсякденних завдань з нуля, таких як: автентифікація, API, тестування, дебаг, кешування і т.д. Саме для спрощення розробки сайтів і швидкого вирішення цих завдань були придумані веб-фреймворки.

Фреймворк – це каркас, призначений для створення динамічних веб-сайтів, веб-додатків, служб або ресурсів. Він спрощує розробку і позбавляє від необхідності писати звичайний код. Фреймворк спрощує доступ до бази даних, розробку інтерфейсу та зменшує дублювання коду.

Один з найпопулярніших фреймворків є `Yii 2`.

`Yii` – це універсальний фреймворк, і він може бути задіяний у всіх типах веб-додатків.

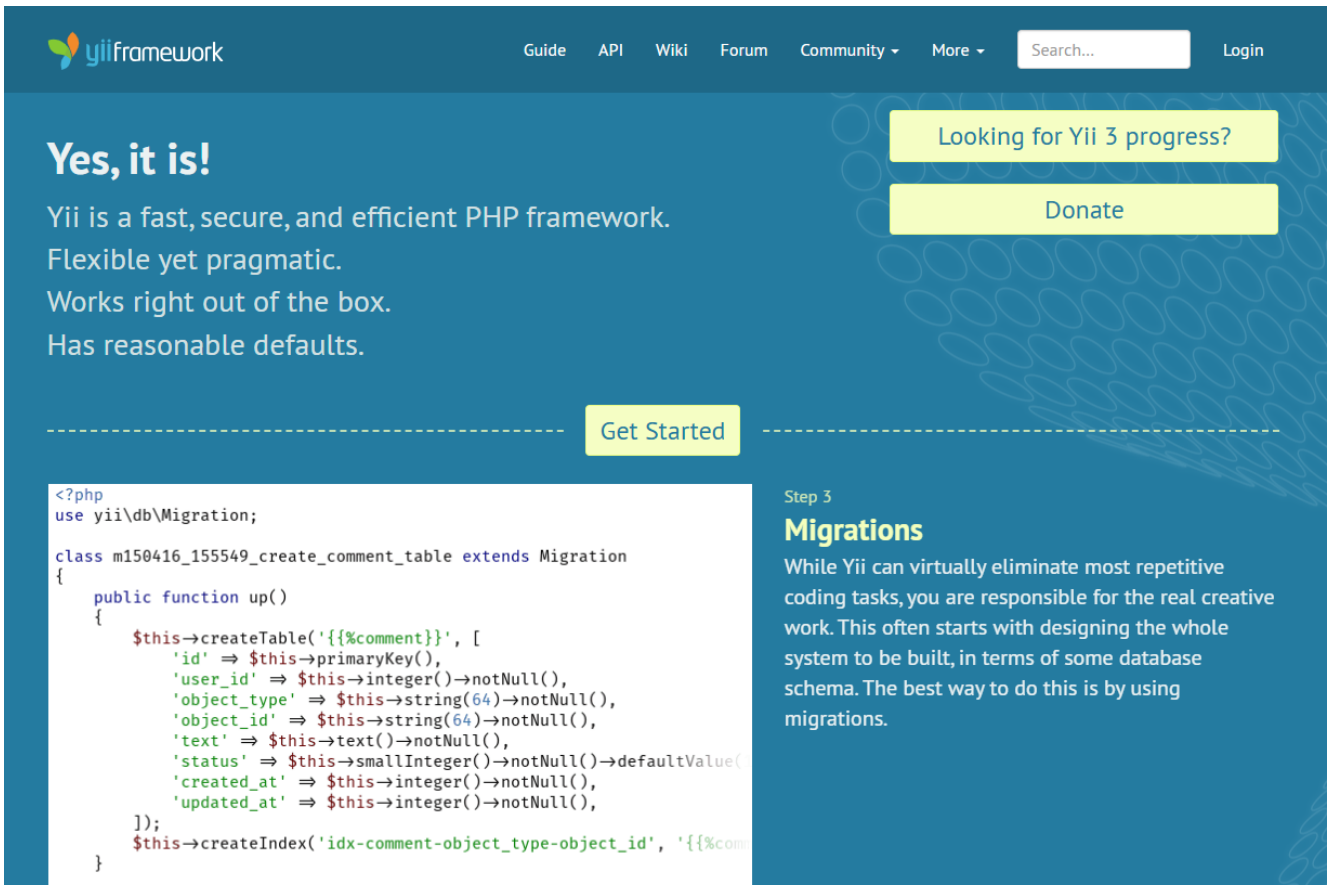


Рисунок 2.2. – Головна веб-сторінка `Yii` фреймворку

Завдяки своїй складовій структурі та чудовій підтримці кешу, фреймворки особливо підходять для розробки великих проектів, таких як портали, форуми, `CMS`, магазини або програми `RESTful`. `Yii` – це командний проект. Він підтримується і розвивається сильною командою та великою спільнотою

розробників. Найбільш підходящі можливості і кращі практики регулярно впроваджуються в фреймворк в вигляді простих і елегантних інтерфейсів:

– як і багато інших PHP-фреймворків, для організації коду Yii використовує архітектурний патерн MVC;

– Yii дотримується філософії простого і елегантного коду не намагаючись ускладнювати дизайн тільки заради проходження будь-якими шаблонами проектування;

– Yii є full-stack фреймворком і включає в себе перевірені і добре зарекомендовані в себе можливості: ActiveRecord для реляційних і NoSQL баз даних, підтримку REST API, багаторівневе кешування та інше;

– Yii відмінно масштабований – можна налаштувати, або замінити практично будь-яку частину основного коду. Використовуючи архітектуру розширень, легко ділитися кодом або використовувати код спільноти;

– одна з головних цілей Yii – продуктивність.

Не менш популярний фреймворк Laravel 7. Laravel – це фреймворк для веб-додатків з виразним і елегантним синтаксисом. Він дозволить спростити вирішення основних наболілих завдань, таких як автентифікація, маршрутизація, сесії і кешування.

Laravel – це спроба об'єднати все найкраще, що є в інших PHP фреймворк, а також Ruby on Rails, ASP.NET MVC і Sinatra.

Laravel – доступний, але потужний. Має безліч відмінних інструментів для великих, надійних додатків:

– Як і багато інших PHP фреймворків, для організації коду Laravel використовує архітектурний патерн MVC.

– Laravel дотримується філософії виразного, красивого синтаксису. Він призначений для людей, які цінують елегантність, простоту і читаність.

– Функціонал `Laravel` є простим для розуміння і використання.

– Більшість функцій `Laravel` чудово працюють, не вимагаючи додаткових налаштувань.

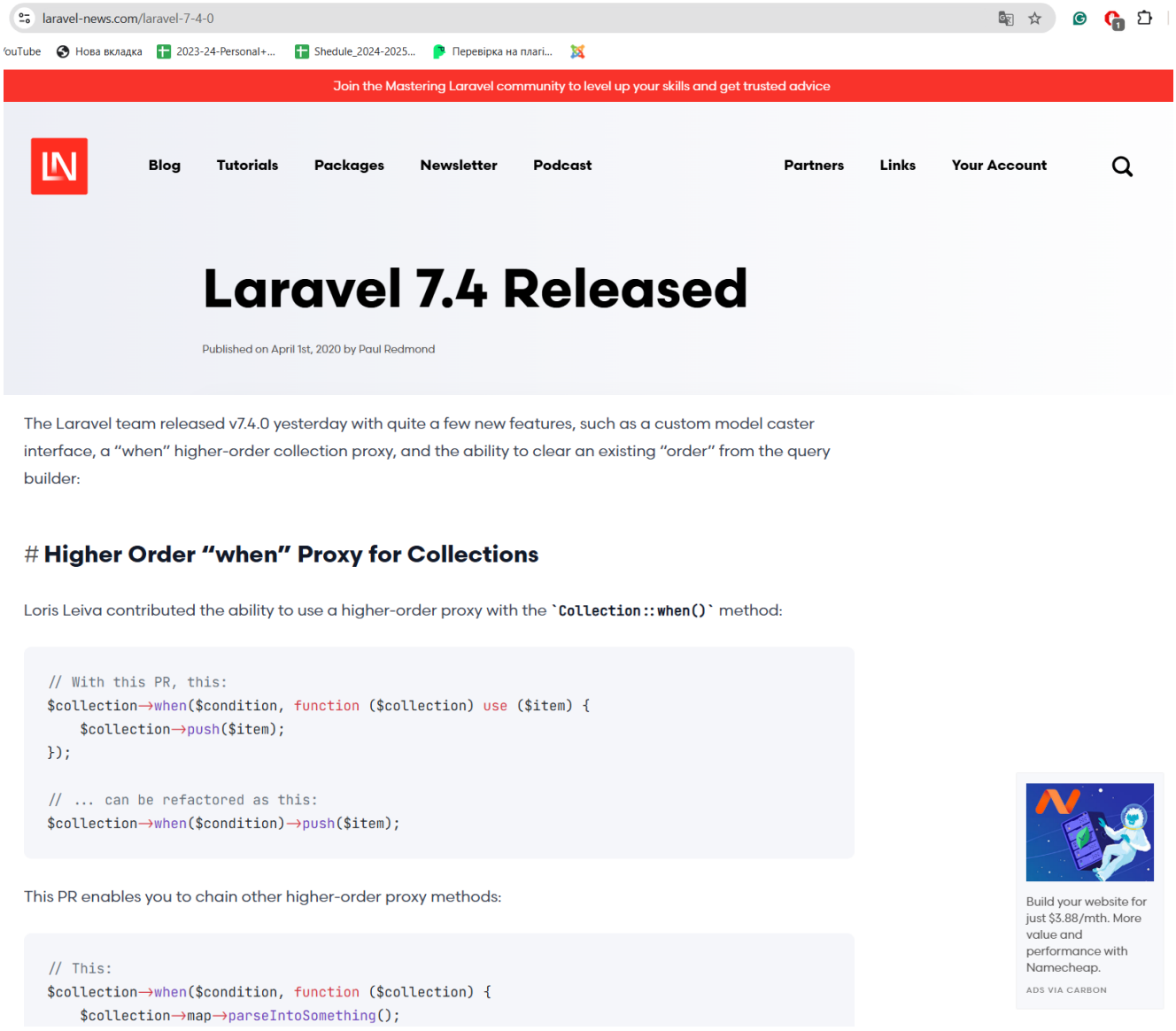


Рисунок 2.3. – Головна веб-сторінка `Laravel` фреймворку

Вони спираються на загальноприйняті стандарти написання коду, роблячи його інтуїтивно зрозумілим.

– Документація `Laravel` закінчена і постійно оновлюється.

– Чудова `IoC` (Інверсія управління).

– Зручна система міграцій.

– Інтегрована система модульного тестування.

Обидва фреймворка для організації коду використовують архітектурний патерн MVC (Модель–вигляд–контролер) (рис. 2.4).

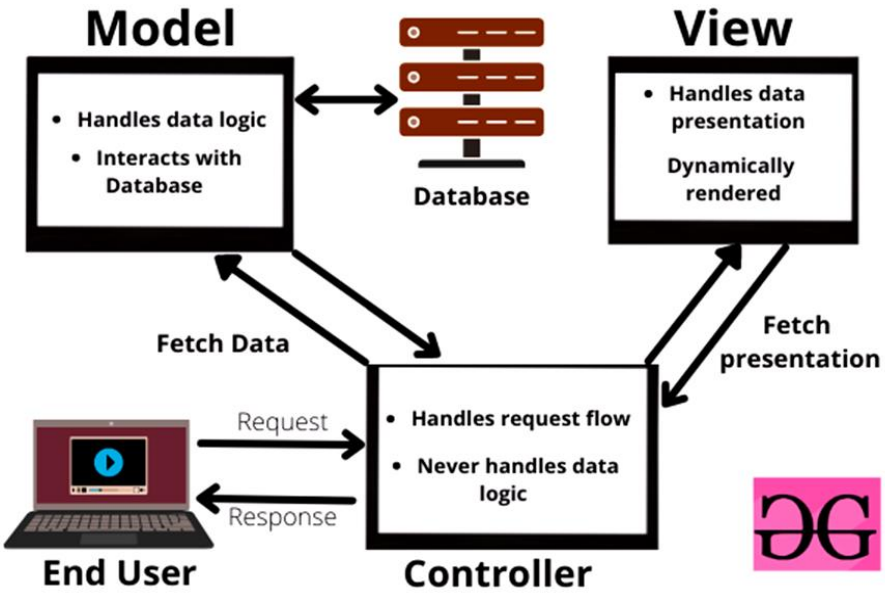


Рисунок 2.4. – Архітектурний патерн "Модель–вигляд–контролер" (MVC)

Детальний розгляд наведених фреймворків переконує в тому, що вони складаються із трьох основних компонентів:

– Model – це сам додаток, який нічого не знає про HTTP запити;

– View – це HTTP запит/відповідь і представлення даних, які вимагає клієнт від сервера;

– Controller – це кілька і більше класів, чиє завдання – абстрагування моделей від HTTP.

За архітектурою ці фрейморки схожі, оскільки реалізовують той самий архітектурний патерн MVC. Він дозволяє швидко реалізувати проект, а також легко супроводжувати і масштабувати його і надалі.

Розширення важлива частина фреймворку, тому розглянемо їх докладніше. ***Розширення*** – це можливість підключати додаткові модулі, або бібліотеки для фрейморка, які дозволять розширити його можливості. Yii і Laravel підтримують таку можливість. На даний момент, розширень набагато більше, звичайно ж, у Yii, так як фреймворк живе набагато довше, ніж Laravel, однак

другий, в свою чергу, розвивається дуже великими темпами і включає в себе багато всього для роботи відразу «з коробки».

Розширення є абсолютно для будь-якого завдання, будь то панель адміністратора, або ж платіжна система.

Для належної роботи сайту важлива підтримка REST API.

{ REST : API }

Representational State Transfer API

REST – це архітектурний стиль взаємодії між компонентами розподілених додатків у мережі. REST – це набір послідовних обмежень, які слід враховувати під час проектування розподіленої системи гіпермедіа. У деяких випадках (інтернет-магазини, пошукові системи, інші системи на основі даних) це може підвищити продуктивність та спростити архітектуру. Загалом, компоненти REST взаємодіють із клієнтами та серверами у всесвітній мережі.

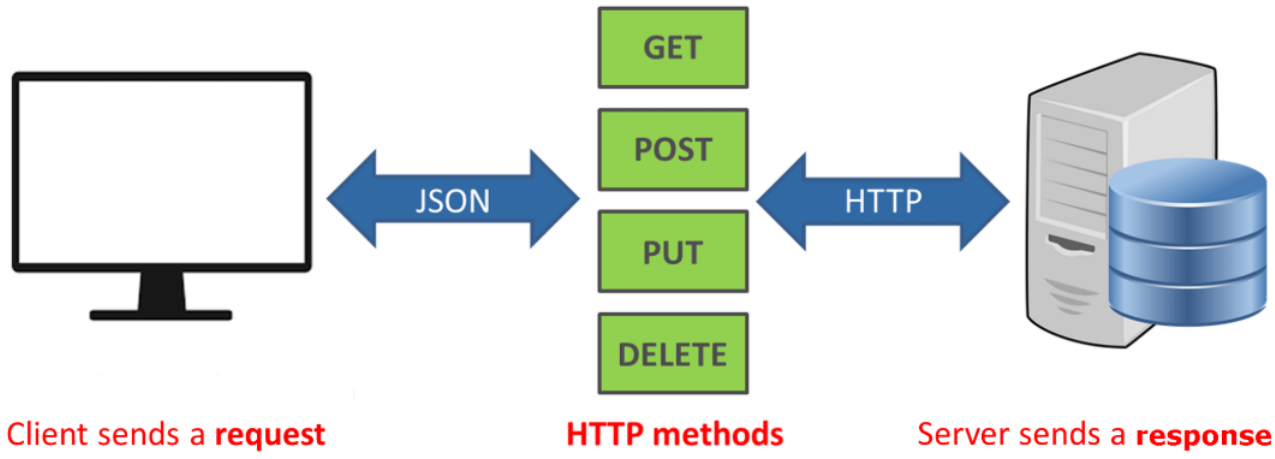


Рисунок 2.5. – Структура моделі взаємодії REST API

В Інтернеті віддалені процедурні виклики можуть бути звичайними HTTP-запитами (зазвичай "GET" або "POST"; це називається "REST-запит"), а необхідні дані передаються як параметри запиту.

Для веб-служб, які відповідають REST тобто, не порушуючи жодних обмежень, що застосовуються до нього, використовується термін "RESTful". На відміну від веб-служб на основі SOAP, веб-API RESTful не має "офіційного"

стандарту. Справа в тому, що REST – це архітектурний стиль, а SOAP – протокол. Хоча сам REST не є стандартом, більшість реалізацій RESTful використовують стандарти HTTP, URL, JSON та XML.

В кінцевому рахунку, в даний час всі сучасні PHP фреймворки зобов'язані підтримувати REST архітектуру. Нижче наведено список можливостей фреймворків для роботи з REST API.

Yii 2:

– Підтримка JSON, JSONP і XML.

– Роутинг відповідно до REST-запитами.

– Підтримка HATEAOS.

– Кешування запитів.

– Обмеження швидкості.

Laravel:

– Налаштування роутінга REST-запитів.

– Підтримка JSON, JSONP.

На основі проведеного аналізу був зроблений вибір на сторону, що активно розвивається Laravel, його структура дозволяє легко масштабувати веб-додаток. Yii є більш стабільним, але менш масштабованим.

2.4. Вибір засобів та побудова бази даних

Для розробки бази даних використовують такі інструменти, як MySQL, Oracle чи SQL Server, щоб знаходити, зберегти або відредагувати дані та повернути їх користувачеві за допомогою коду візуальної частини.

Вище перелічені мови використовуються не лише для створення веб-сайтів, програмного забезпечення та додатків; вони також використовуються для створення та управління базами даних. Бази даних не розроблені для розуміння тих же мов, на яких запрограмовані програми, тому важливо мати мову, яку вони

розуміють – як SQL, стандартну мову для доступу та маніпулювання реляційними базами даних. SQL означає структурована мова запитів.

Вона має власну розмітку і в основному дозволяє програмістам працювати з даними, що зберігаються в системі баз даних. Розглянемо детальніше різни типи, переваги та недоліки баз даних (табл. 2.2).

Таблиця 2.2 – Переваги та недоліки інструментів управління базами даних

№ п/п	Назва	Переваги	Недоліки
1	2	3	4
1	Oracle	Має останні інновації та функції, що надходять від їх продукції, оскільки Oracle прагне встановити планку для інших інструментів управління базами даних.	Вартість Oracle може бути непосильною, особливо для менших організацій.
		Інструменти управління базами даних Oracle також надзвичайно надійні, і ви можете знайти той, який може робити практично все, про що ви можете подумати.	Після встановлення система може вимагати значних ресурсів, тому для впровадження Oracle може знадобитися оновлення обладнання.
2	MySQL	Безкоштовний.	Ви можете витратити багато часу і сил, щоб змусити MySQL робити те, що інші системи роблять автоматично.
		Пропонує безліч функціональних можливостей.	Немає вбудованої підтримки для XML або OLAP.
		Існують різноманітні інтерфейси користувачів.	Підтримка доступна для безкоштовної версії, але за неї потрібно заплатити.
		Це може бути зроблено для роботи з іншими базами даних, включаючи DB2 та Oracle.	
3	Microsoft SQL Server	Це дуже швидко і стабільно.	Ціни на підприємства можуть перевищувати багато організацій
		Пропонує можливість регулювання та відстеження рівнів продуктивності, що може зменшити використання ресурсів.	Навіть при налаштуванні продуктивності Microsoft SQL Server може обробляти ресурси.
		Є можливість отримати доступ до візуалізації на мобільних пристроях.	Багато людей мають проблеми з використанням інтеграційних служб SQL Server для імпорту файлів.
		Дуже добре працює з іншими продуктами Microsoft.	

Продовження табл. 2.2.

4	PostgreSQL	Ця система управління базами даних є масштабованою і може обробляти терабайти даних.	Не чітка документація.
		Підтримує JSON.	Конфігурація може бути заплутаною.
		Існують різноманітні заздалегідь визначені функції.	Швидкість може постраждати під час великих масових операцій чи запитів.
		Доступна низка інтерфейсів.	
5	MongoDB	Швидка і проста у використанні.	Установки за замовчуванням не захищені
		Підтримує JSON та інші документи NoSQL.	Налаштування може бути тривалим процесом.
		Дані будь-якої структури можна зберігати та отримувати доступ до них швидко та легко.	Інструменти для перекладу SQL на запити MongoDB доступні, але вони додають додатковий крок до використання системи.
		SQL не використовується як мова запитів.	
		Система швидка і стабільна.	

Для реалізації завдань нашої кваліфікаційної роботи вибрано – інструментарій MySQL. Це зумовлене тим, що ця СУБД є безкоштовною, є супутній продукт MySQL Workbench (який дозволяє просто взаємодіяти зі встановленою БД, надаючи можливість через зручний інтерфейс користувача виконувати будь-які маніпуляції з даними), найбільш розповсюджена (займаючи друге місце за популярністю у світі, поступаючись лише корпоративній СУБД від Oracle). Отже, у випадку виникнення проблем чи збоїв в роботі нашох системи, в мережі можна знайти величезних кількість матеріалу для їх вирішення.

Для розробки клієнтської частини сайт обрано мови програмування HTML, CSS і JavaScript так як вони є найпопулярнішими мовами для веб-розробки і досить легкі у вивченні.

Розробка серверної частини відбуватиметься із застосуванням PHP та фреймфорку Laravel. Ця мова активно використовується багатьма сайтами і досить легка у вивченні. Також найбільш популярні системи контролю контентом використовують саме цю мову програмування, що надає можливість розробляти додатковий функціонал.

РОЗДІЛ 3
МЕТОДИКА ПОБУДОВИ ТА ГОЛОВНІ СКЛАДОВІ ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1. Структурно-функціональна модель інформаційної системи

Проектування інформаційної системи (ІС) має декілька базових етапів – найважливішим є саме розробка структури програмного додатку. Проектування ІС є багатоступінчастим процесом створення та модернізації із впорядкованими методологіями та інструментарієм. Якщо виділяти етап проектування ІС як окремий, то його можна розмістити між етапами аналізу і розробки. Однак на практиці чіткий поділ на етапи ускладнений або неможливий, оскільки проектування, формально починаючи з визначення мети проекту, часто триває на стадіях тестування і реалізації.

Структуру запропонованої ІС можна розділити на декілька частин, а саме до та після авторизації. Головна відмінність – доступ до інформації та певних сторінок сайту. До авторизації у системі користувачу передбачатимуться відкриті деякі модулі (табл. 3.1).

Таблиця 3.1 – Відкриті модулі для користувача перед етапом авторизації

№ п/п	Сторінка	Опис
1	Головна	Сторінка із основною інформацією про інформаційну систему, головні переваги та етапи реєстрації.
2	Авторизація	Сторінка для авторизації зареєстрованих користувачів.
3	Реєстрація	Сторінка для реєстрації нових користувачів системи.

Після авторизації користувачеві відкриваються додаткові можливості. Перш за все, що саме залежить від типу акаунту та масштабу діяльності – користувач, інтернет-магазин чи сукупність магазинів. Панель меню складається із посилань

на ключові модулі даної системи. Розглянемо детальніше, які можливості відкриті для кожного з типу акаунтів у табл.3.2.

Таблиця 3.2 – Функціонал за групами користувачів

№ п/п	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 добре спроектована для якісної взаємодії всіх користувачів. Це добре продемонстровано у структурі інформаційної системи. Кожен користувач має можливість працювати з файлами, взаємодіяти з іншими користувачами та інше. Крім того, було виконано всі задачі, поставлені на початку виконання даної практичної роботи.

Наведемо структуру ІС на прикладі схеми *IDEF0*.

IDEF0 – це багато в чому дуже простий метод. Кожне поле представляє окремий процес, як і в інших підходах, але IDEF0 відрізняється використанням та розміщенням стрілок. Крім звичайних входів і виходів, існують ще два типи стрілок, які представляють "елементи управління" та "механізми".

Елементи управління є формою введення, але які використовуються для керування діяльністю в процесі. Іноді виникає певна міра невизначеності щодо того, є елемент елементом введення чи контролю.

Розглянемо IDEF0 запропонованої інформаційної системи (рис. 3.1).

Методологія IDEF1 розділяє елементів структури інформаційної галузі, їх властивості та взаємозв'язки на класи. Центральним поняттям методології IDEF1 є поняття сутності.

Рисунок 3.1. – IDEF0 інформаційної системи

Клас сутностей являє собою сукупність інформації, накопиченої і зберігається в рамках інтернет-магазину і відповідає певному об'єкту або групі об'єктів реального світу. Основними концептуальними властивостями сутностей в IDEF1 є:

– *стійкість*. Інформація, що має відношення до тієї чи іншої суті постійно накопичується;

– *унікальність*. Будь-яка сутність може бути однозначно ідентифікована з іншої сутності.

Кожна сутність має своє ім'я і атрибути. Атрибути представляють собою характерні властивості і ознаки об'єктів реального світу, які стосуються певної сутності. Клас атрибутів являє собою набір пар, що складаються з імені атрибута і його значення для певної сутності. Атрибути, за якими можна однозначно відрізнити одну сутність від іншої називаються ключовими атрибутами.

Кожна сутність може характеризуватися декількома ключовими атрибутами. Клас взаємозв'язків в IDEF1 являє собою сукупність взаємозв'язків між сутностями. Взаємозв'язок між двома окремими сутностями вважається існуючої в тому випадку, клас атрибутів однієї сутності містить ключові атрибути іншої сутності.

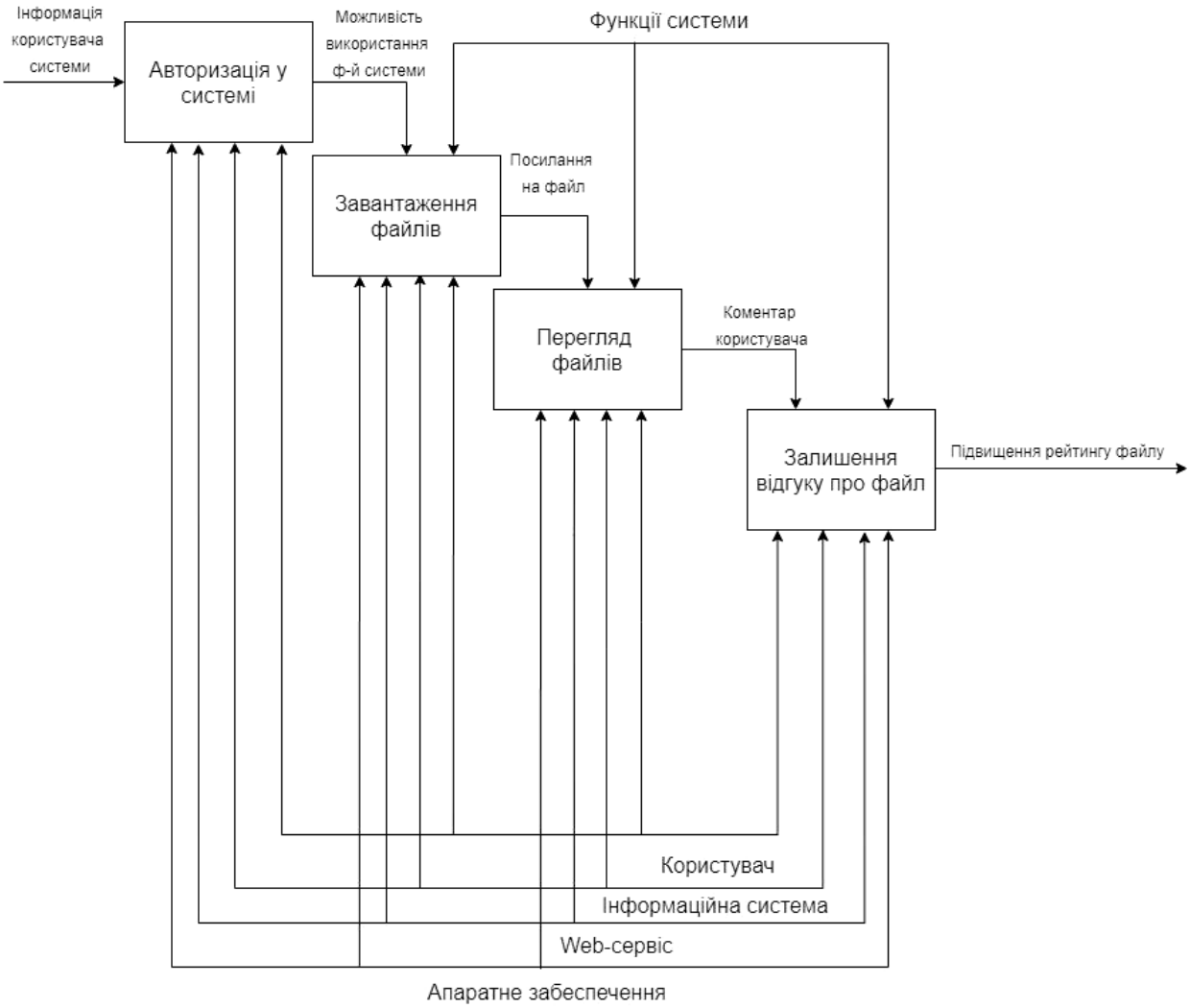


Рисунок 3.2. – IDEF1 проекту ІС обміну файлами та інвентаризації даних

Кожен з вищеописаних класів має своє умовне графічне відображення, згідно з методологією IDEF1. Наведемо IDEF1 запропонованої ІС (рис. 3.2).

3.2. Діаграми варіантів використання та діяльності

Метою діаграми варіантів використання є відображення динамічного аспекту системи. Однак це визначення є занадто загальним, щоб описати мету, оскільки інші чотири діаграми (діяльність, послідовність, співпраця та діаграма стану) також мають те саме призначення. Ми розглянемо якесь конкретне призначення, яке відрізнятиме його від інших чотирьох діаграм.

Діаграми випадків використання використовуються для збору вимог системи, включаючи внутрішні та зовнішні впливи. Ці вимоги в основному є вимогами дизайну. Отже, коли система аналізується для збору її функціональних можливостей, готуються випадки використання та визначаються дійові особи.

Коли початкове завдання виконане, діаграми використання моделюються для подання зовнішнього виду.

Цілі діаграм випадків використання:

– Використовується для збору вимог системи.

– Використовується для отримання зовнішнього вигляду системи.

– Визначте зовнішні та внутрішні фактори, що впливають на систему.

– Показати взаємодію між вимогами акторів.

Розглянемо детальніше акторів та варіанти використання в табл. 3.3 та табл. 3.4.

Таблиця 3.3 – Опис акторів

№ п/п	Назва	Опис
1	Користувач	Користувач, що має можливість завантажувати та обмінюватись файлами.
2	Адміністратор	Користувач, що має найбільші можливості на сайті, а саме редагувати та видаляти дані на сайті.

Таблиця 3.4 – Опис варіантів використання

№ п/п	Назва	Опис
1	2	3
1	Авторизація	Модуль дозволяє авторизуватися на сайті.
2	Реєстрація	Модуль дозволяє зареєструватися новому користувачеві на сайті.
3	Редагування інформації на сайті	Дозволяє змінювати загальну інформацію на сайті.
4	Перегляд завантажених файлів	Модуль дозволяє переглядати завантажені користувачами файли.

Продовження табл. 3.4.

5	Завантаженні файлів	Модуль, що дозволяє користувачам завантажувати файли та змінювати інформацію про них.
6	Залишення відгуків про файли	Кожен користувач який має доступ до файлу має можливість залишати відгуки про файл.
7	Редагування даних на сторінці	Кожен користувач має можливість редагувати особисту інформацію на сайті

Сформувавши представлення про інформацію та функціонал ІС побудовано діаграму варіантів використання (рис. 3.3).

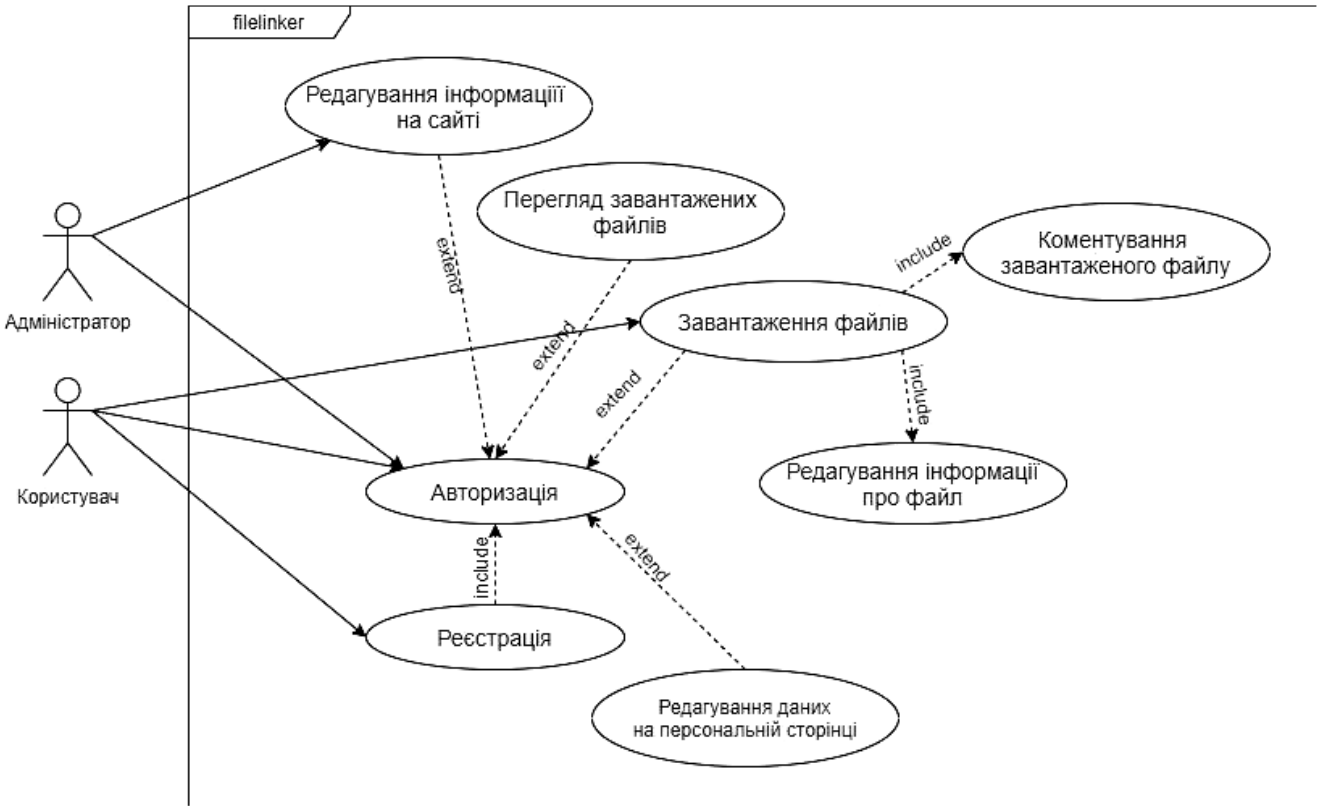


Рисунок 3.3. – Діаграма варіантів використання

Діаграма діяльності – це технологія, що дозволяє описувати логіку процедур, бізнес-процеси і потоки робіт. У багатьох випадках вони нагадують блок-схеми, але принципова різниця між діаграмами діяльності і нотацією блок-схем полягає в тому, що перші підтримують паралельне процеси.

На рисунку 3.4 3.5 зображено діаграми діяльності модулів ІС інвентаризації даних та файлів. Діаграма діяльності дозволяє будь-кому, хто виконує цей процес,

вибирати порядок дій. Іншими словами, діаграма тільки встановлює правила обов'язкової послідовності дій, яким я повинен слідувати.

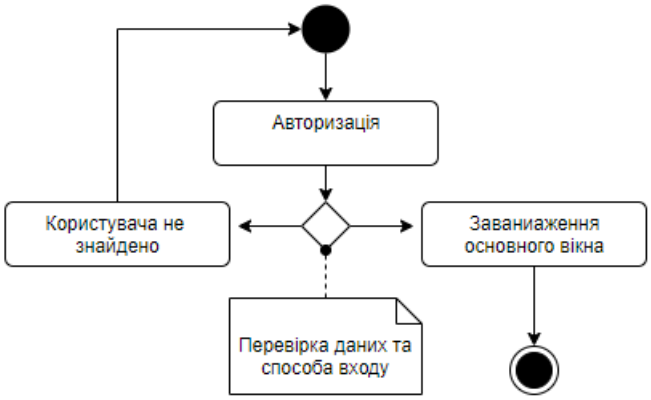


Рисунок 3.4. – Діаграма діяльності модулю авторизації

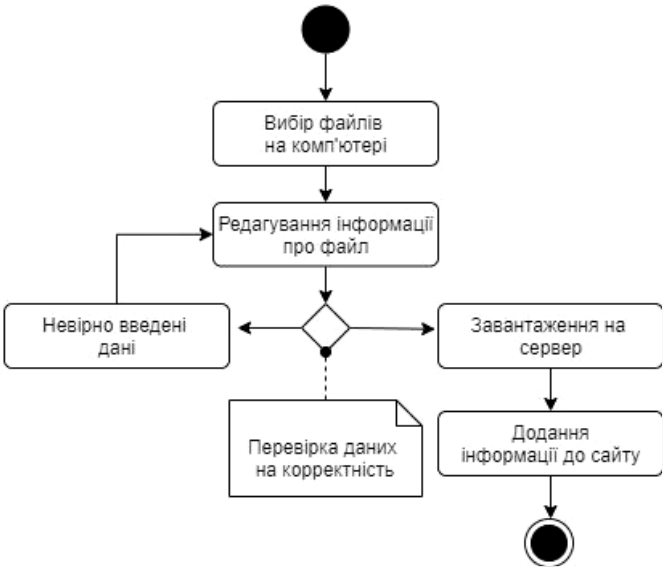


Рисунок 3.5. – Діаграма діяльності модулю завантаження файлів

Це важливо для моделювання бізнес-процесів, оскільки ці процеси часто виконуються паралельно. Такі діаграми також корисні при розробці паралельних алгоритмів, в яких незалежні потоки можуть виконувати роботу паралельно.

3.3. ER-діаграма та побудова бази даних

Основні користувачі системи – замовник та фахівець. За допомогою цієї

системи вони можуть отримувати різні відомості про виконавців та замовників, послуги та інше.

Зобразимо відносини в ER-діаграмі (рис. 3.6). Проаналізувавши сутність, використовувані в моделі ІС, перейдемо до реалізації структури БД. Для цього представимо імена необхідних таблиць, атрибутів, типів, їх призначення та обмеження в табл. 3.5.

Таблиця 3.5 – Інформація за таблицями ER-діаграми

№	Таблиця	Поле	Зміст	Тип	Ключі	Обмеження
1	users	id	Ідентифікатор аканту	INTEGER	PK	Не пустий
		email	Електронна пошта	VARCHAR(100)		Не пустий
		password	Пароль від акаунту	VARCHAR(100)		Не пустий
		nikname	Псевдонім користувача	VARCHAR(100)		Не пустий
		name	Ім’я користувача	VARCHAR(100)		Не пустий
		surmane	Прізвище користувача	VARCHAR(100)		Не пустий
		sex	Стать	BOOLEAN		Не пустий
		photo	Фото користувача	VARCHAR(100)		Не пустий

Продовження в дод. А.

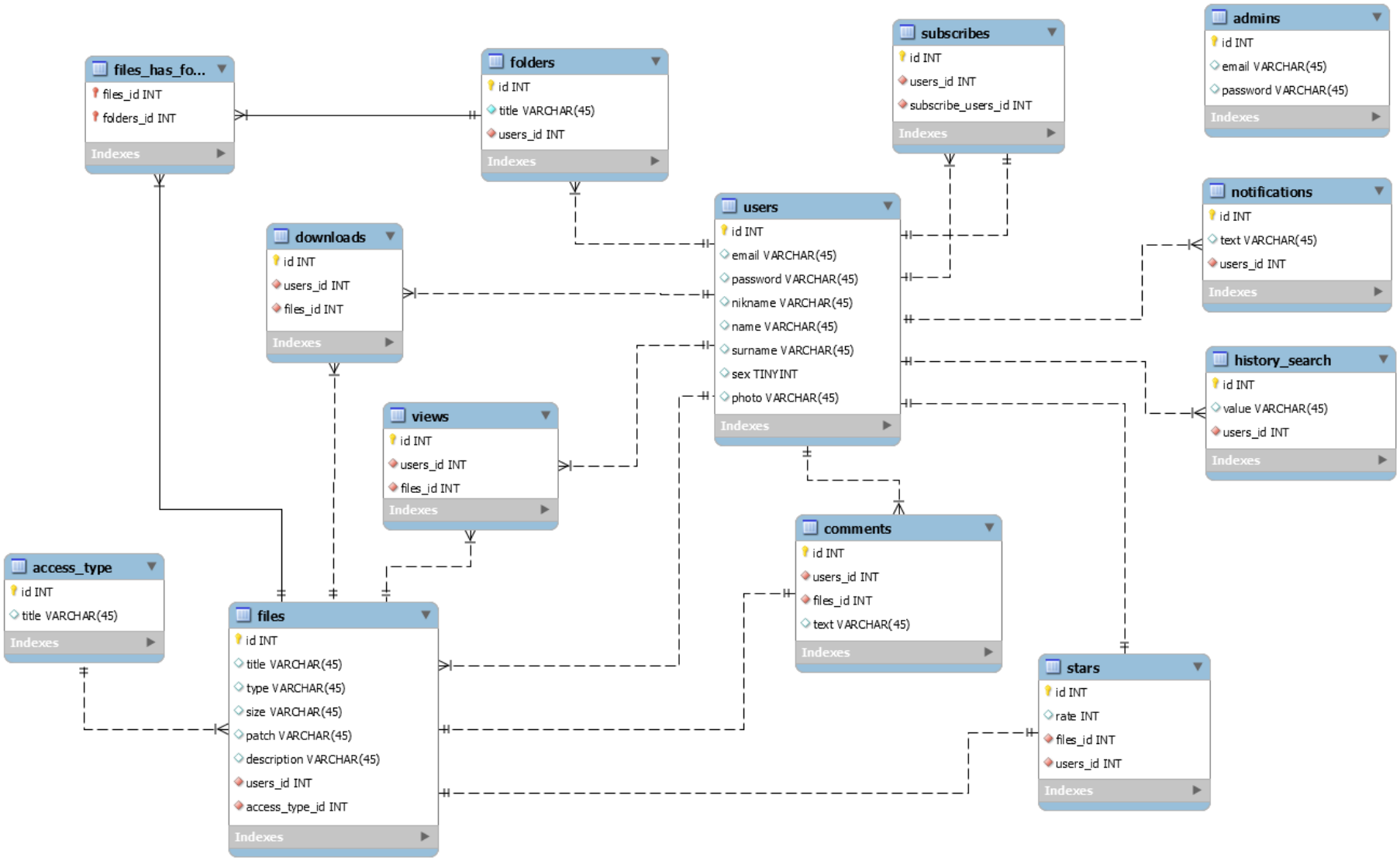


Рисунок 3.6. – ER-діаграма

РОЗДІЛ 4
РЕАЛІЗАЦІЯ СИСТЕМИ ІНВЕНТАРИЗАЦІЇ ДАНИХ

4.1. Архітектура клієнт-серверної взаємодії

Розробка ІС буде відбуватись у безкоштовному текстовому редакторі VS Code.

Visual Studio Code – це крос-платформенний редактор скриптів, створений корпорацією Майкрософт. Поряд з розширенням PowerShell він надає широкі інтерактивні можливості редагування скриптів, спрощуючи написання надійних скриптів PowerShell. Також для збереження вихідного коду, а також відслідковування версій сайту було використано систему контролю версій Git. Системи контролю версій дозволяють розробникам зберігати всі зміни, внесені в код [25]. Тому в разі, помилки, вони можуть просто відкотити код до робочого стану замість того, щоб витрачати години на пошуки маленької помилки або помилок, які ламають весь код.

Системи контролю версій також дають можливість декільком розробникам працювати над одним проектом і зберігати внесені зміни, щоб переконатися, що всі можуть стежити за тим, над чим вони працюють.

Проект буде розроблятись на віртуальному веб сервері OpenServer.

Open Server Panel – це портативна серверна платформа і програмне середовище, створена спеціально для веб-розробників з урахуванням їх рекомендацій і побажань. Також буде використаний менеджер модулів для PHP-Composer.

Розробка сайту починається з створення нового проекту з використанням фреймворку `Laravel`. Для цього в терміналі `Composer` необхідно написати наступну команду:

```
composer create-project --prefer-dist laravel/laravel filelinker
```

Де «filelinker» – ім'я проекту і може бути змінено на будь-яке. Після чого буде створено базову архітектуру проекту (рис. 4.1).

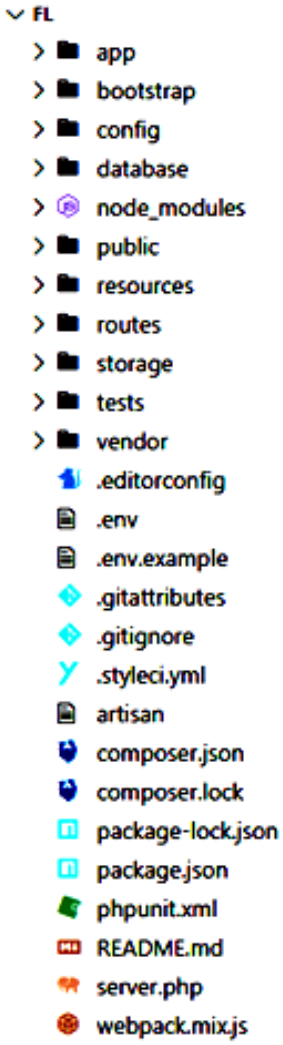


Рисунок 4.1. – Архітектура проекту ІС інвентаризації даних

Таблиця 4.1 – Таблиця опису директорій проекту

№ п/п	Пакет	Короткий опис
1	2	3
1	«app»	Містить код ядра додатку.
2	«bootstrap»	Містить файли, які завантажують фреймворк і налаштовують автозавантаження.
3	«config»	Містить всі конфігураційні файли додатку.
4	«database»	Містить міграції і класи для наповнення початковими даними БД.

Продовження табл. 4.1.

1	2	3
5	«public»	Містить файл index.php, який є вхідною точкою для всіх запитів, що надходять в додаток. Також ця папка містить ресурси, такі як зображення, JavaScript, CSS.
6	«resources»	Містить початковий код, а також сирі, некомпільовані ресурси, такі як LESS, SASS, JavaScript.
7	«routes»	Містить всі визначення маршрутів додатку.
8	«storage»	Містить скомпільовані Blade-шаблони, файл-сесії, кешу файлів і інші файли, створені фреймворком.
9	«tests»	Містить автотести.

Тоді починається розробка клієнтської частини сайту, а саме створення всіх необхідних компонентів, налаштування маршрутів доступу і верстка шаблонів та розробка модулів серверної частини сайту, створення всіх необхідних контролерів, моделей та налаштування API.

Розробка клієнтської частини. Написання сайту починається із створення «скелету» з використанням мови розмітки HTML. Після чого за допомогою мови каскадних стилів CSS формується зовнішній вигляд веб-сайту і надається візуальна форма елементів для зручного користування. Після цього, створений шаблон розбивається на окремі компоненти, що надає можливість багаторазово використовувати один компонент на різних сторінках сайту. Приклад компоненту головної сторінки наведено в додатку В.

Всі компоненти знаходяться в папці `resources/js/`.

Наступним кроком створюється єдина точка входу яка підключає всі необхідні компоненти і модулі JavaScript.

```
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';
Vue.prototype.$http = axios;
const token = localStorage.getItem('token');
```

```
if (token) {
Vue.prototype.$http.defaults.headers.common['Authorization'] = token;
}
const app = new Vue({
el: '#app',
components: {
AppComponent
},
vuetify,
store,
router
});
```

Створивши всі необхідні шаблони і компоненти. Відбувається налаштування маршрутів для навігації між сторінками сайту. Далі наведено приклад коду маршруту для головної сторінки.

```
export default new Router({
mode: "history",
routes: [
{
path: '/',
name: 'index',
component: Index
},
]
});
```

В результаті при натисканні на посилання відбувається перехід на необхідний компонент вказаний в маршрутизаторі.

Розробка серверної частини. Розробка серверної частини починається зі створення та підключення бази даних. Для цього необхідно перейти до налаштувань підключення до бази даних. Необхідно змінити файл (.env), він знаходиться в кореневій папці проекту. Необхідно змінити 4 рядки: `DB_HOST = localhost`, `DB_DATABASE = filelinker`, `DB_USERNAME = root`, `DB_PASSWORD = secret`.

Де `DB_HOST` – ім'я хоста;

`DB_DATABASE` – ім'я бази даних;

`DB_USERNAME` – ім'я користувача для доступу до бази даних;

`DB_PASSWORD` – пароль користувача для доступу до бази даних.

Потім створюються міграції для того щоб автоматично було створено базу даних. ***Міграції*** – це система контролю версій для бази даних. Вони дозволяють

команді програмістів змінювати структуру БД, в той же час залишаючись в курсі змін інших учасників. Далі наведено приклад міграції для створення таблиці користувачів.

```
public function up()
{
Schema::create('users', function (Blueprint $table) {
$table->id();
$table->string('email')->unique();
$table->string('password');
$table->string('nickname')->unique();
$table->string('name');
$table->string('surname');
$table->boolean('sex');
$table->string('photo')->default('/img/no-image.png');
$table->boolean('is_admin')->default(0);
$table->timestamps();
});
}
```

Після створення всіх необхідних міграцій потрібно написати в термінали Composer наступну команду:

```
php artisan:migrate
```

Далі створюються моделі для зручної роботи з базою даних, налаштувань залежностей між таблицями і підключення додаткових модулів. Далі наведено приклад моделі користувачів.

```
class User extends Authenticatable
{
use Notifiable, HasApiTokens;
protected $table = 'users';
protected $fillable = [
'email',
'password',
'nickname',
'name',
'surname',
'sex',
'photo',
'is_admin'
];
protected $hidden = [
'password',
];
function feed() {
return $this->hasMany('App\Models\Feed', 'users_id');
}
function files() {
```

```
    return      $this->hasMany('App\Models\Files',      'users_id')-
>where('basket', 0);
    }
    function subscribes() {
    return                  $this->HasMany('App\Models\Subscribes',
'subscribe_users_id');
    }
    function mySubscribes() {
    return $this->HasMany('App\Models\Subscribes', 'users_id');
    }
    }
```

Всі запити що надходять на сервер обробляються API маршрутизатором, що знаходиться в файлі `routes/api.php`. Маршрутизація визначає, як додаток відповідає на клієнтський запит до конкретної адреси.

Тоді в залежності від типу запиту підключається необхідний контролер з необхідним методом який опрацює дані отримані з бази даних. ***Контролер*** – з'єднує моделі, види і інші компоненти необхідні для роботи системи. А також відповідає за обробку запитів користувача, а саме отримання і обробку даних та відправку готового результату клієнтській частині системи. Приклад такого контролеру для додання коментару:

```
class FileController extends Controller
{
function commentFile(Request $request, $id) {
$model = new Comments();
$model->create([
"files_id" => $id,
"users_id" => Auth::id(),
"text" => $request->comment
]);
return response('ok', 200);
}
}
```

Фрагменти коду наведені в Додатку Б та В.

4.2. Результати застосування системи інвентаризації даних

Робота з додатком розпочинається з головної сторінки, де користувач має можливість продивитися інформацію про сайт та авторизуватися (рис.4.2).

Доступ до всіх функцій сайту мають лише авторизовані користувачі. Тому перед початком роботи потрібно зареєструватись.

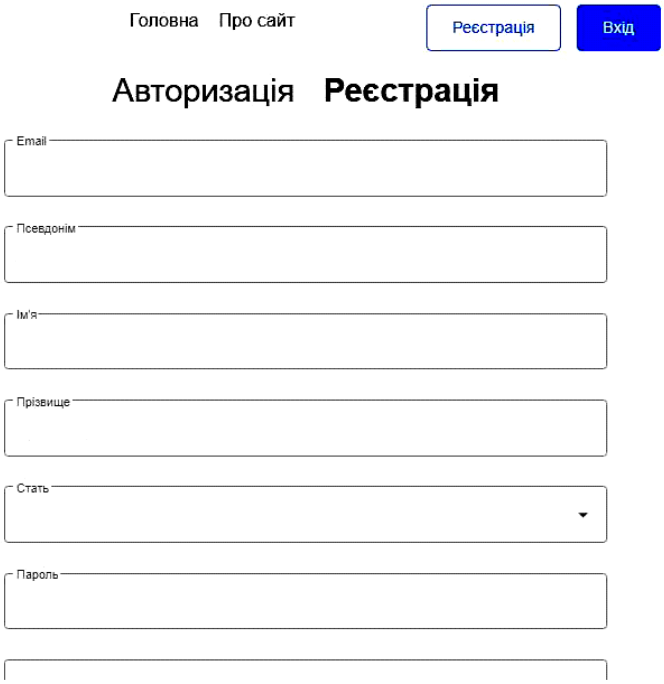


Рисунок 4.2 – Сторінка реєстрації користувача

Після реєстрації або авторизації відкривається сторінка з власним кабінетом користувача. На сторінці є можливість відредагувати власну інформацію, змінити фото та пароль (рис.4.3).

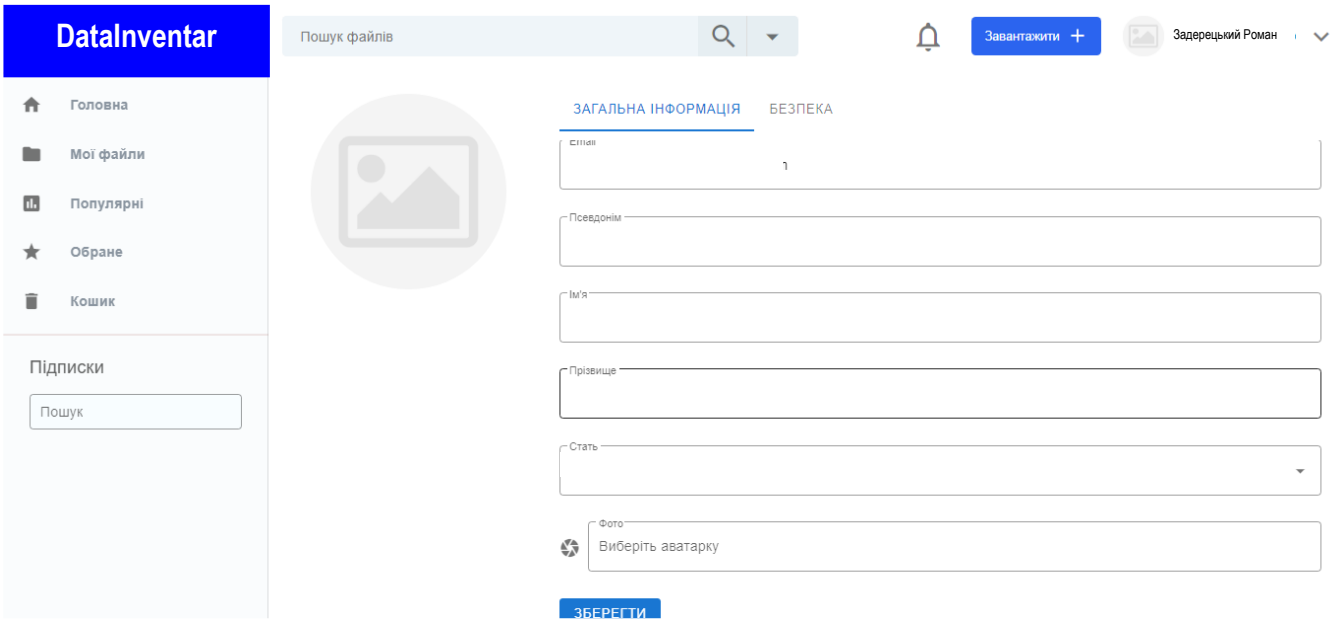


Рисунок 4.3. – Сторінка кабінету користувача

Головна функція сайту це завантаження та обмін файлами. Для завантаження нових файлів необхідно натиснути кнопку «Завантажити +». Після натискання з’явиться форма для вибору файлів на комп’ютері та результат успішного завантаження файлу.

Після вибору бажаних файлів відобразиться список файлів з додатковими тестовими полями. Ці поля надають можливість змінювати назву та додати опис кожного з файлу (рис. 4.4). Натиснувши кнопку «Готово» з’явиться повідомлення про успішну публікацію файлів та збереження даних.

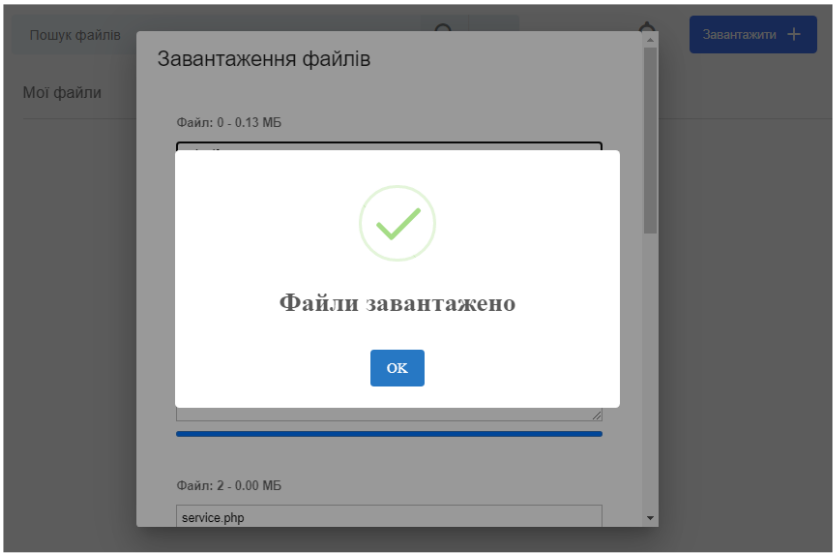


Рисунок 4.4 – Повідомлення про успішне завантаження

Всі завантажені користувачем файли можна побачити на сторінці «Мої файли». Кожна з сторінок надає можливість змінювати вид відображення файлів, а саме списком, або блоками. Для кожного з типів файлів було створено відповідне зображення для швидкого визначення типу файлу (рис. 4.5).

Web-сайт надає можливість завантажувати файл, копіювати посилання, переглядати додаткову інформацію.

Перегляд реалізований у вигляді додаткового правого блоку. Він складається з зображення типу файлу, назви, інформації про дату останньої зміцни, блоку «зірок» для можливості залишити бал фалу, що на далі вплине не його популярність.

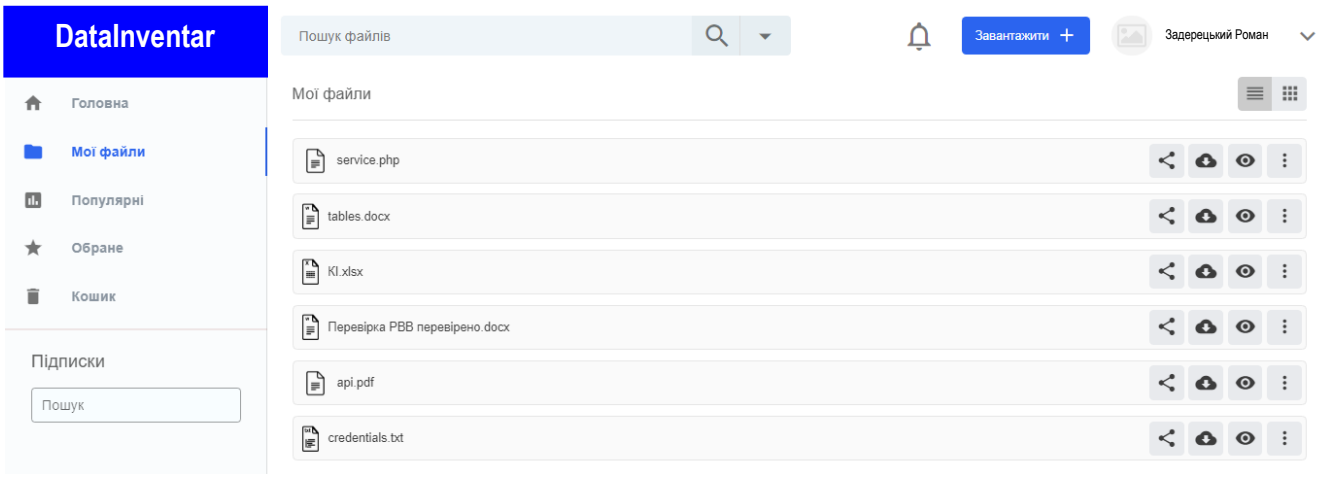


Рисунок 4.5. – Сторінка «Мої файли» зі списком файлів та даними

Також блок містить наступні кнопки:

• Відкрити – файл відкриється в новій вкладці,

• Завантажити – завантаження файлу на комп'ютер,

• Поділитись – скопіювати посилання на файл.

Додатково блок містить інформацію про автора файлу, кількість переглядів та завантажень. Також є можливість залишати коментар.

Також реалізований пошук файлів. Пошук можна виконати по назві файлу, типу та розміру. Кожен тим і максимальний розмір в формі пошуку змінюється динамічно в залежності від файлів завантаженими користувачами.

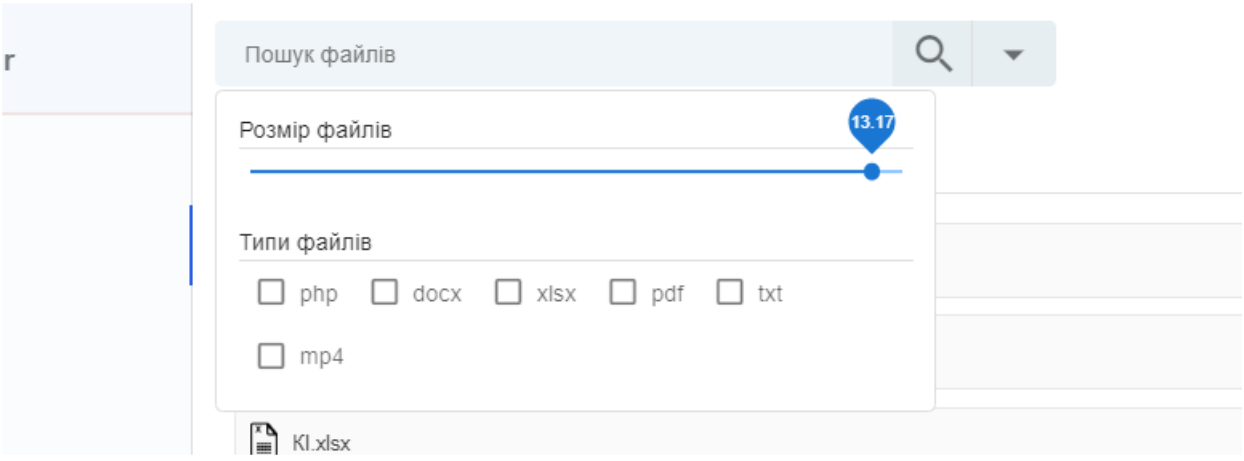


Рисунок 4.6. – Форма пошуку файлів з даними

За бажанням можна перейти на сторінку автора файлу і переглянути інформацію про нього, а також список файлів завантажених користувачем тощо.

На сайті також реалізована система підписок на аккаунти користувачів. Кожен користувач має можливість підписатись на оновлення та публікації інших користувачів сайту.

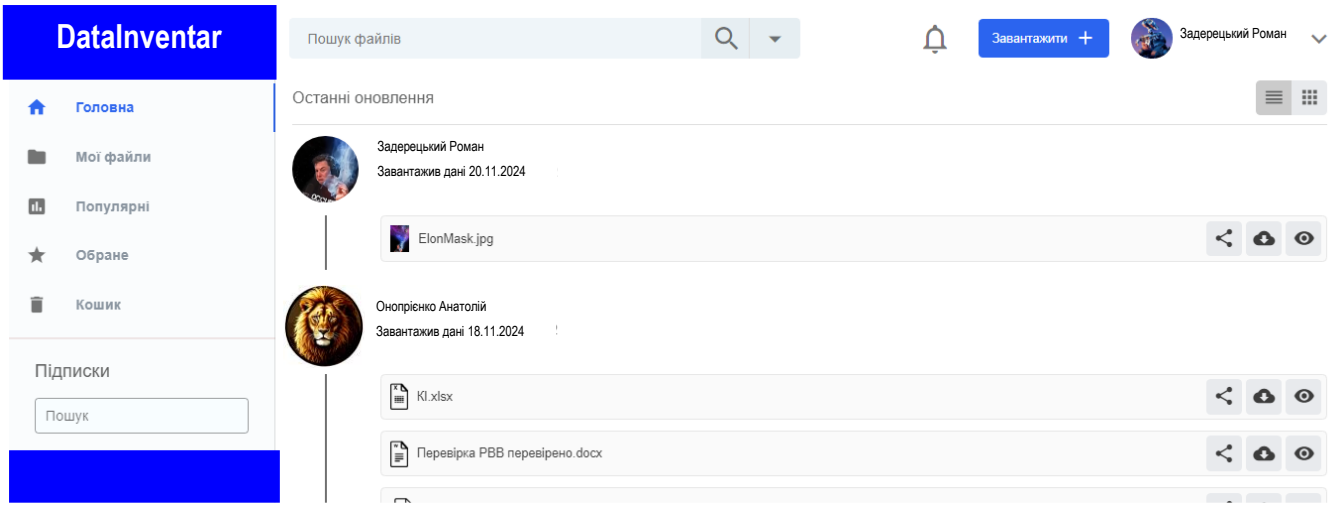


Рисунок 4.7. – Відображення змін із файлами та даними

Всі оновлення, зміни із даними та файлами відображатимуться також на сторінці оновлень інвентаризованих даних (рис. 4.7).

РОЗДІЛ 5
ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

5.1. Розробка логіко-імітаційної моделі виникнення травм і аварій

Методикою оцінки рівня небезпеки робочих місць, машин, виробничих процесів та окремих виробництв передбачено пошук об'єктивного критерію рівня небезпеки для конкретного об'єкта [7]. Таким показником вибрана ймовірність виникнення аварії, травми залежно від явища, що досліджується.

Для побудови логіко-імітаційної моделі процесу, формування і виникнення аварії та травми в процесі створення мікрокліматичних умов у приміщенні оцінюють відповідні небезпечні події. Кожній із них присвоїмо ймовірність виникнення:

Шифр	Назва події	Ймовірність
P_1	Відсутність захисного заземлення	0,02
P_2	Пошкодження захисного заземлення	0,04
P_3	Спрацювання складових захисту	0,1
P_4	Неправильна експлуатація захисту	0,02
P_5	Відсутність профілактичних заходів	0,2
P_6	Відсутність захисного щита	0,12
P_7	Недотримання правил вибору взуття	0,15
P_8	Незнання правил техніки безпеки	0,1
P_9	Відсутність засобів індивідуального захисту	0,2
P_{10}	Легковажність	0,08

На основі наведених подій будуємо матрицю логічних взаємозв'язків між окремими пунктами, графічна інтерпретація якої зображено на рис. 5.1.

Розрахуємо ймовірності виникнення подій, що формують логіко-імітаційну модель процесів створення мікрокліматичних умов. Розглянемо травмонебезпечну ситуацію, що виникає за умови роботи працівників із

електронебезпекою.

Підставивши дані ймовірностей базових подій у формулу, отримаємо ймовірність події 13: $P_{13} = 0{,}2 + 0{,}4 - 0{,}2 \cdot 0{,}4 = 0{,}0592$.

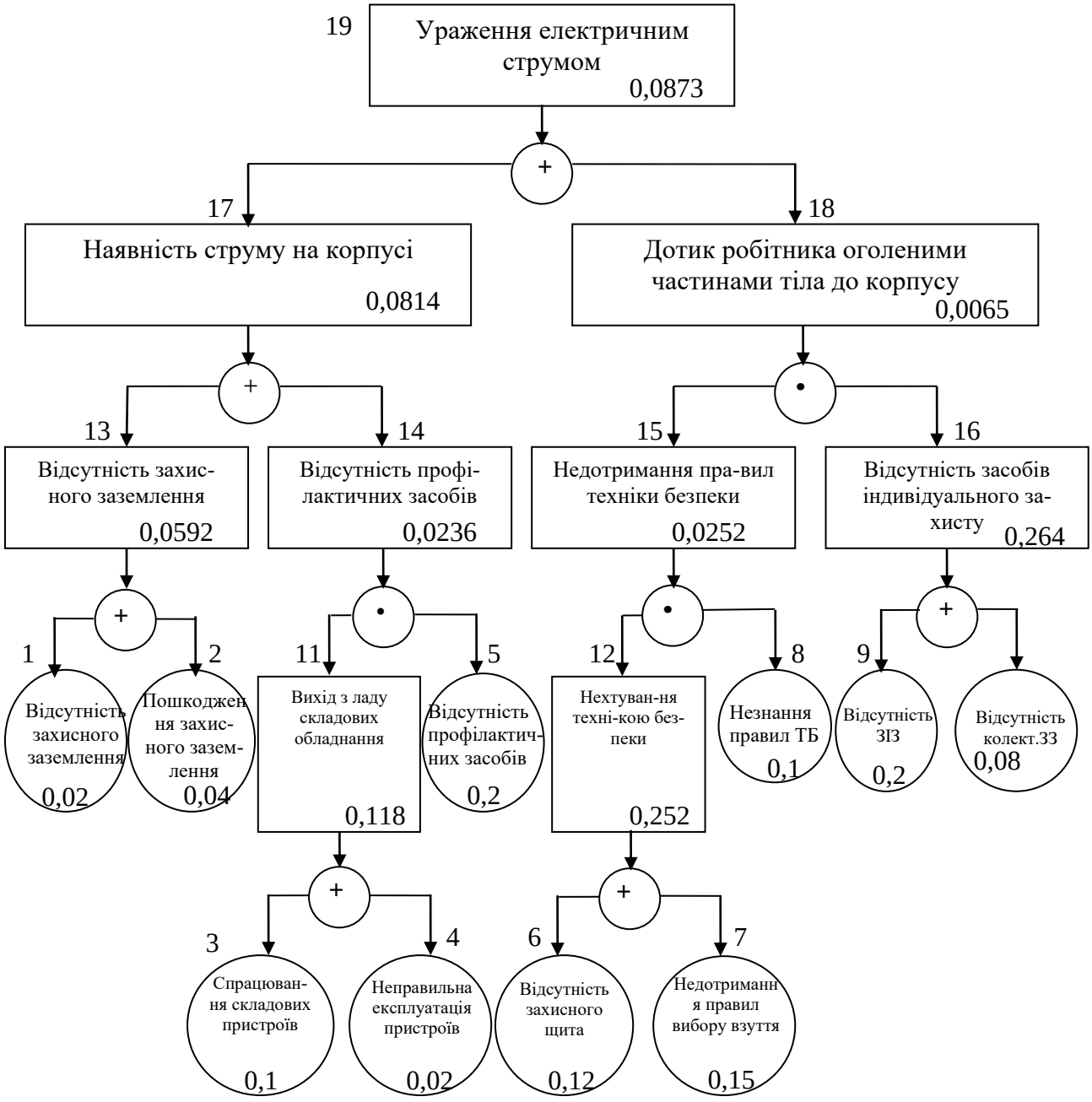


Рис. 5.1. Матриця логічних взаємозв’язків між окремими подіями травмонебезпечної ситуації [7]

Аналогічно визначаємо ймовірність інших подій:

$$P_{11} = P_4 + P_5 - P_4P_5 = 0{,}3 + 0{,}4 - 0{,}3 \cdot 0{,}4 = 0{,}118.$$
$$P_{12} = P_6 + P_7 - P_6P_7 = 0{,}3 + 0{,}5 - 0{,}3 \cdot 0{,}5 = 0{,}252.$$
$$P_{16} = P_9 + P_{10} - P_9P_{10} = 0{,}2 + 0{,}15 - 0{,}2 \cdot 0{,}15 = 0{,}264.$$

$$P_{14} = P_{11} \cdot P_5 = 0{,}118 \cdot 0{,}2 = 0{,}0236.$$
$$P_{15} = P_{12} \cdot P_8 = 0{,}252 \cdot 0{,}1 = 0{,}0252.$$
$$P_{17} = P_{13} + P_{14} - P_{13} \cdot P_{14} = 0{,}592 + 0{,}0236 - 0{,}0592 \cdot 0{,}0236 = 0{,}0814.$$
$$P_{18} = P_{15} \cdot P_{16} = 0{,}264 \cdot 0{,}0252 = 0{,}0065.$$
$$P_{19} = P_{17} + P_{18} - P_{17} \cdot P_{18} = 0{,}0065 + 0{,}0814 - 0{,}0065 \cdot 0{,}0814 = 0{,}0873.$$

Таким чином, ймовірність перекидання машини та наслідкового виникнення травми працівника є досить мала і становить – $P_{19} = 0,0873$.

5.2. Планування заходів із покращення умов праці

До заходів щодо покращення умов праці належать всі види діяльності, спрямовані на попередження, нейтралізацію або зменшення негативної дії шкідливих і небезпечних виробничих факторів на працівників.

Рівень умов праці оцінюють порівнянням за фактичними і нормативними значеннями узагальнених (групових) показників.

Заходи щодо поліпшення умов праці здійснюють з метою створення безпечних умов праці шляхом:

- доведення до нормативного рівня показників виробничого середовища за елементами умов праці;
- захисту працівників від дії небезпечних і шкідливих виробничих факторів.

До показників ефективності заходів щодо поліпшення умов праці належать:

а) зміни стану умов праці:

- зміна кількості засобів виробництва, приведених у відповідність до вимог стандартів безпеки праці;
- покращання санітарно-гігієнічних показників;
- покращання психофізичних показників, зменшення фізичних і нервово-психічних навантажень, в т.ч. монотонних умов праці;

б) соціальні результати заходів:

- збільшення кількості робочих місць, що відповідають нормативним

вимогам;

- зниження рівня виробничого травматизму;
- престиж та задоволення працею.

Отже, на покращення охорони праці потрібно виділити кошти на відновлення вентиляційних систем у ремонтних майстернях, естетично оформити приміщення офісу, відновити кабінет з охорони праці, поновити протипожежний інвентар.

5.3. Безпека в надзвичайних ситуаціях

Актуальність проблеми природно-техногенної безпеки для населення і території, зумовлена зростанням втрат людей, що спричиняється небезпечними природними явищами, промисловими аваріями та катастрофами [7]. У системі цивільної оборони окремого господарства необхідно забезпечити захист населення таким чином:

Укриття в захисних спорудах, якому підлягає усе населення відповідно до приналежності, досягається створенням фонду захисних споруд.

Евакуаційні заходи, які проводяться в містах та інших населених пунктах, які мають об’єкти підвищеної небезпеки, а також у воєнний час, основним способом захисту населення є евакуація і розміщення його у позаміській зоні.

Медичний захист проводиться для зменшення ступеня ураження людей, своєчасного надання допомоги постраждалим та їх лікування, забезпечення епідеміологічного благополуччя в районах надзвичайних ситуацій.

Радіаційний і хімічний захист включає заходи щодо виявлення і оцінки радіаційної та хімічної обстановки, організацію і здійснення дозиметричного та хімічного контролю, розроблення типових режимів радіаційного захисту, забезпечення засобами індивідуального захисту, організацію і проведення спеціальної обробки.

ВИСНОВКИ І РЕКОМЕНДАЦІЇ

Ефективне управління інвентаризацією IT-системи має важливе значення для будь-якої організації для підтримки оптимальної продуктивності, безпеки та ефективності мережі. Але без правильного вибору програмного забезпечення, провести інвентаризацію може бути складним завданням.

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 файлообмінниках 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.

Завдання розробника frontend – кодувати інтерфейс веб-сайту чи програми; тобто частину веб-сайту, яку бачить і взаємодіє користувач. Вони працюють над розробками, наданими веб-дизайнером, і реалізують їх за допомогою HTML, JavaScript та CSS. Мови розмітки використовуються для визначення форматування текстового файлу. Іншими словами, мова розмітки повідомляє програмне забезпечення, яке відображає текст, як слід форматувати текст. Мови розмітки є повністю розбірливими для читання – вони містять стандартні слова, але теги розмітки приховані від користувача в кінцевому варіанті коду.

Код який створюють розробники backend, гарантує, що все, що будує розробник frontend, є повністю функціональним. Головне завдання розробника серверної частини – переконатися, що сервер, програма та база даних взаємодіють між собою. Для вирішення такої складної задачі розробники використовують серверні мови, такі як PHP, Ruby, Python та Java для створення програми.

Веб-технології стрімко розвиваються, і щоб встигати за швидкістю розвитку нових технологій, у веб-програмістів часто бракує часу для реалізації повсякденних завдань з нуля, таких як: автентифікація, API, тестування, дебаг, кешування і т.д. Саме для спрощення розробки сайтів і швидкого вирішення цих завдань були придумані веб-фреймворки.

Зокрема, у результаті виконання завдань кваліфікаційної роботи та розробки інформаційної системи проведено збір та аналіз актуальної інформації щодо розробки сервісів для реалізації хмарного сховища. Результат роботи відіграв важливу роль для формування пріоритетів та вибору функціоналу.

Передбачено можливість завантаження даних інтернет-магазинів й обміну файлами між іншими користувачами сайту і не тільки, пошук файлів та даних з можливістю вказувати додаткові параметри, структурування даних.

Обрано функціонал для реалізації файлообмінника, візуальної та функціональної частини. Виконаний аналіз послуг та фреймворків допоміг обрати найбільш вдалі варіанти, а самеVue.js та Laravel.

У результаті формування структури інформаційної системи створено відповідні діаграми та її декомпозиції. Розроблена діаграма варіантів використання повністю відображає функціонал запропонованого сервісу. Ця інформація покладена в основу для розробки прототипу веб-додатку із реалізованою системою інвентаризації даних.

Проаналізувавши сутності, використовувані в моделі інформаційної системи виконана побудова бази даних.

У результаті виконаної кваліфікаційної роботи розроблено інформаційну систему для обміну даними, їх структурування, обміну файлами із реалізацією усіх можливостей користувача тощо.

БІБЛІОГРАФІЧНИЙ СПИСОК

1. Анді Гутманс, Стіг Баккен, Дерік Ретханс. Програмування живлення PHP5, 2015. 704 с.

2. Галаєва Л.В. Моделювання галузевої структури сільськогосподарського підприємства: практикум / Л.В. Галаєва, Н.Г. Шульга, Н.А. Рогоза. К. : НУБіП, 2018. 35 с.

3. Документація Laravel. URL: https://laravel.com/. (дата звернення: 20.11.2024

4. Документація Vue.js. URL: https://ua.vuejs.org/v2/guide/. (дата звернення: 20.11.2024

5. Комплексна система IT-рішень для управління агробізнесом. URL: https://agrichain.com.ua/ (дата звернення: 20.11.2024).

6. Кращі безкоштовні файлообмінники без реєстрації 2023 – https://comhub.ru/luchshie-besplatnye-fajloobmenniki-bez-registratsii. (дата звернення: 20.11.2024).

7. Лехман С.Д. та ін. Запобігання аварійності і травматизму у сільському господарстві / С.Д. Лехман, В.І. Рубльов, Б.І. Рябцев. К.: Урожай, 1993. 272 с.

8. Мартін Р. Чистий код. Львів. 2019. 368с.

9. Райс Ерік. Інформаційно архітектурний підхід до створення успішних веб-сайтів. [Текст] / Ерік Райс. Addison Wesley, 20020. 266 с.

10. Платформа .NET URL: https://metanit.com/sharp/mvc5/(дата звернення: 30.11.2024).

11. Сайт Metanit all about C#. URL: https://metanit.com/sharp/general.php (дата звернення: 30.11.2024).

12. Спірін О. М. Зміст навчального матеріалу спецкурсу "Хмарні інформаційно-аналітичні технології у науково-дослідному процесі". Інформаційні технології і засоби навчання. 2016. Т. 52, вип. 2. С. 108-120.

13. Файлообмінники. Зберігання та обмін файлами? – http://www.introweb.ru/inews/soft/news9591.php. (дата звернення: 20.11.2024).

14. Ямпольський Л.С., Лавров О.А. Штучний інтелект у плануванні та управлінні виробництвом. К.: Вища школа, 2015. 254с.

15. Хеслоп П. HTML з самого початку. К: Препринт, 2014. 450с.

16. Що таке CSS. URL: http://phpist.com.ua/css/5-whatcss. (дата звернення: 30.11.2024).

17. Що таке файлообмінники і як ними користуватися – https://mypclife.ru/chto-takoe-fayloobmenniki-i-kak-imi-polzovatsya/. (дата звернення: 10.12.2024).

18. Best File Hosting Services (2020): Free Storage & Sharing? – https://www.hostingadvice.com/how-to/best-file-hosting-services/. (дата звернення: 20.11.2024).

19. Digital universe of opportunities. URL: http://www.emc.com/leadership/digital-universe/index.htm?pid=landing-digitaluniverse-131212 (дата звернення: 24.11.2024).

20. Google Drive Api .NET Quickstart Complete the steps described in the rest of this page to create a simple .NET console application that makes requests to the Drive API. URL: https://developers.google.com/drive/api/v3/quickstart/dotnet . (дата звернення: 30.11.2024).

21. Issues with cloud storage. URL: https://www.datapine.com/blog/cloud-computing-risks-and-challenges/(дата звернення: 30.11.2024).

22. Kaushik, A. Web Analytics 2.0: The Art of Online Accountability and Science of Customer //Centricity (1st ed.). Indianapolis, IN: John Wiley & Sons. 2010

23. Library Jquery. URL: https://jquery.com/(дата звернення: 30.11.2024).

24. Plant Simulation. URL: https://www.csoft.ru/catalog/soft/plant-simulation/plant-simulation.html. (дата звернення: 30.11.2024).

25. Tutorial for create API with ASP.NET MVC. URL: https://docs.microsoft.com/en-us/aspnet/core/tutorials/first-mvcapp/?view=aspnetcore-3.1. (дата звернення: 30.11.2024).

ДОДАТКИ

Додаток А

ER-діаграма

Таблиця А.1 – Інформація за таблицями ER-діаграми (*продовження*)

№	Таблиця	Поле	Зміст	Тип	Ключі	Обмеження
2	Subscribes	user_id	Ідентифікатор аккаунта	INTEGER	FK	Не пустий
		subscribe_users_id	Ідентифікатор аккаунта на який виконана підписка	INTEGER	FK	Не пустий
3	Notifications	id	Ідентифікатор повідомлення	INTEGER	PK	Не пустий
		Text	Текст повідомлення	TEXT		Не пустий
		user_id	Ідентифікатор аккаунта	INTEGER	FK	Не пустий
4	history_search	id	Ідентифікатор пошукового запиту	INTEGER	PK	Не пустий
		Value	Пошуковий запит	VARCHAR(100)		Не пустий
		user_id	Ідентифікатор аккаунта	INTEGER	FK	Не пустий
5	Folders	id	Ідентифікатор папки	INTEGER	PK	Не пустий
		Title	Назва папки	VARCHAR(100)		Не пустий
		user_id	Ідентифікатор аккаунта	INTEGER	FK	Не пустий
6	files_has_folders	files_id	Ідентифікатор файлу	INTEGER	FK	Не пустий
		folders_id	Ідентифікатор папки	INTEGER	FK	Не пустий
7	files	id	Ідентифікатор файлу	INTEGER	PK	Не пустий
		title	Назва файлу	VARCHAR(100)		Не пустий
		Type	Тип файлу	VARCHAR(10)		Не пустий
		Size	Розмір файлу	FLOAT		Не пустий
		Patch	Посилання на файл	VARCHAR(100)		Не пустий
		Description	Опис файлу	TEXT		

		user_id	Ідентифікатор аккаунта	INTEGER	FK	Не пустий
		access_type_id	Ідентифікатор типу доступу	INTEGER	FK	Не пустий
8	access_type	id	Ідентифікатор рівня доступу	INTEGER	PK	Не пустий
		Title	Назва типу доступу	VARCHAR(100)		Не пустий
9	Comments	id	Ідентифікатор коментару	INTEGER	PK	Не пустий
		user_id	Ідентифікатор аккаунта	INTEGER	FK	Не пустий
		user_id	Ідентифікатор аккаунта	INTEGER	FK	Не пустий
		Text	Текст комнтару	TEXT		Не пустий
10	Stars	id	Ідентифікатор балу	INTEGER	PK	Не пустий
		Rate	Кількість балів	INTEGER		Не пустий
		user_id	Ідентифікатор аккаунта	INTEGER	FK	Не пустий
11	views	id	Ідентифікатор перегляду	INTEGER	PK	Не пустий
		user_id	Ідентифікатор аккаунта	INTEGER	FK	Не пустий
		user_id	Ідентифікатор аккаунта	INTEGER	FK	Не пустий
12	downloads	id	Ідентифікатор завантаження	INTEGER	PK	Не пустий
		user_id	Ідентифікатор аккаунта	INTEGER	FK	Не пустий
		user_id	Ідентифікатор аккаунта	INTEGER	FK	Не пустий

Додаток Б.
Фрагменти коду головних блоків ІС

Створення таблиці акауту

```
<?php
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;
class CreateUsersTable extends Migration
{
public function up()
{
Schema::create('users', function (Blueprint $table) {
$table->id();
$table->string('email')->unique();
$table->string('password');
$table->string('nickname')->unique();
$table->string('name');
$table->string('surname');
$table->boolean('sex');
$table->string('photo')->default('/img/no-image.png');
$table->boolean('is_admin')->default(0);
$table->timestamps();
});
}
public function down()
{
Schema::dropIfExists('users');
}
}
```

Створення таблиці файлів.

```
<?php
use
Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;
class CreateFilesTable extends Migration
{
public function up()
{
Schema::create('files', function
(Blueprint $table) {
$table->id();
63
$table->string('title');
$table->string('type');
$table->integer('size');
$table->string('patch');
$table->text('description')->nullable();
$table->foreignId('users_id')-
>unsigned();
$table->foreignId('access_type_id');
$table->boolean('basket')->default(0);
$table->timestamps();
});
Schema::table('files', function
(Blueprint $table) {
$table->index('users_id');
$table->foreign('users_id')-
>references('id')->on('users')-
>onDelete('cascade');
});
Schema::table('files', function
(Blueprint $table) {
$table->index('access_type_id');
$table->foreign('access_type_id')-
>references('id')->on('access_type');
});
}
public function down()
{
Schema::dropIfExists('files');
}
}
2020_11_01_092557_create_subscribes_table
.php
Міграція створення таблиці підписників.
<?php
use
Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;
class CreateSubscribesTable extends
Migration
{
public function up()
{
Schema::create('subscribes', function
(Blueprint $table) {
$table->id();
$table->foreignId('users_id');
$table->foreignId('subscribe_users_id');
});
Schema::table('subscribes', function
(Blueprint $table) {
$table->index('users_id');
$table->foreign('users_id')-
>references('id')->on('users')-
>onDelete('cascade');
$table->index('subscribe_users_id');
```

```
$table->foreign('subscribe_users_id')-
>references('id')->on('users')-
>onDelete('cascade');
});
}
public function down()
{
Schema::dropIfExists('subscribes');
}
}
```

User.php
Модель користувачів.

```
<?php
namespace App\Models;
use Illuminate\Notifications\Notifiable;
use Illuminate\Foundation\Auth\User as
Authenticatable;
use Laravel\Passport\HasApiTokens;
class User extends Authenticatable
{
use Notifiable, HasApiTokens;
protected $table = 'users';
protected $fillable = [
'email',
'password',
'nickname',
'name',
'surname',
'sex',
'photo',
'is_admin'
];
protected $hidden = [
'password',
];
function feed() {
return $this->hasMany('App\Models\Feed',
'users_id');
}
function files() {
return $this->hasMany('App\Models\Files',
'users_id')->where('basket', 0);
}
function subscribes() {
return $this-
>HasMany('App\Models\Subscribes',
'subscribe_users_id');
}
function mySubscribes() {
return $this-
>HasMany('App\Models\Subscribes',
'users_id');
}
}
```

Files.php
Модель файлів.

```
<?php
namespace App\Models;
use Illuminate\Database\Eloquent\Model;
class Files extends Model
{
protected $table = 'files';
protected $fillable = [
'users_id',
'title',
'type',
'size',
'patch',
'description',
'access_type_id',
'basket'
];
protected $casts = [
'created_at' => 'datetime:Y.m.d',
'updated_at' => 'datetime:Y.m.d',
];
function user() {
return $this-
>belongsTo('App\Models\User',
'users_id');
}
function views() {
return $this->hasMany('App\Models\Views',
'files_id');
}
function downloads() {
return $this-
>hasMany('App\Models\Downloads',
'files_id');
}
function comments() {
return $this-
>hasMany('App\Models\Comments',
'files_id');
}
function stars() {
return $this->hasMany('App\Models\Stars',
'files_id');
}
function favorites() {
return $this-
>hasMany('App\Models\Favorites',
'files_id');
}
}
```

Додаток В.
Фрагменти коду фронт та бекенд реалізації веб-сайту ІС

Index.vue – Блок головної сторінки сайту

```
<template>
<v-container>
<TitlePage :loading="loading"
title="Останні оновлення"></TitlePage>
<div class="font-weight-regular text-
center pt-5" v-if="!loading &&
data.length == 0">
Нічого не знайдено
</div>
<div v-for="(item, index) in data"
:key="index">
<v-row>
<v-col cols="1" class="avatar pb-2">
<img :src="item.user.photo">
</v-col>
<v-col class="nickname pb-2">
<div class="name"><router-link
:to="'/user/'+item.user.id">{{
item.user.name }} {{ item.user.surname
}}</router-link></div>
<div class="time">{{item.user.sex ?
'Завантажив' : 'Завантажила'}} файли {{
item.created_at }}</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" v-if="typeView ==
'line'">
<FileLineItem v-for="(file, index) in
item.files" :item="file.file"
:key="index"></FileLineItem>
</v-col>
<div style="display: flex" v-if="typeView
== 'block'">
<v-col style="padding: 11px" cols="auto"
v-for="(file, index) in item.files"
:key="index">
<FileItem :item="file.file"></FileItem>
</v-col>
</div>
</v-row>
</div>
</v-container>
</template>
<script>
import { mapGetters } from 'vuex';
import FileLineItem from
"../../components/site/FileLineItem";
import FileItem from
"../../components/site/FileItem";
import TitlePage from
"../../components/site/TitlePage";
export default {
components: {
FileLineItem,
FileItem,
TitlePage
},
data() {
return {
loading: true,
data: []
}
},
created() {
this.fetchData();
},
computed: {
typeView() {
return this.getTypeView();
}
},
methods: {
...mapGetters([
"getTypeView"
]),
fetchData() {
axios.get('/api/feed')
.then((response) => {
this.loading = false;
this.data = response.data;
})
.catch((err) => {
this.loading = false;
})
}
}
}
</script>
<style lang="css" scoped>
.nickname {
font-weight: 400;
line-height: 28px;
}
.nickname .name {
font-size: 16px;
}
.nickname .name a {
color: #000;
text-decoration: none;
} 79

.avatar img {
width: 100%;
border-radius: 50%;
}
.vertical-line .line {
width: 2px;
background: #6F6F6F;
height: 95%;
margin: 0 auto;
}
</style>
```

Header.vue – Блок форми пошуку та завантаження файлів

```
<template>
```

```
<v-app-bar app clipped-right flat
height="70" color="white">
<v-dialog v-model="formFiles" persistent
max-width="600px">
<v-card>
<v-card-title>
<span class="headline">Завантаження
файлів</span>
</v-card-title>
<v-card-text>
<v-container>
<label class="input-block" v-
if="files.length == 0">
<input
style="display: none"
type="file"
multiple
id="file"
ref="file"
v-on:change="handleFileUpload()"
>
</label>
<div v-else 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"></pro
gress>
</div>
</v-container>
</v-card-text>
<v-card-actions>
<v-spacer></v-spacer>
<v-btn
color="blue darken-1"
text
@click="formFiles = false; files = []"
>
Закрити
</v-btn>
<v-btn
color="blue darken-1"
text
@click="submitFile()"
>
Готово
</v-btn>
</v-card-actions>
</v-card>
</v-dialog>
<div class="search py-2">
<input type="text" class="search-form"
@input="searchInput" v-
model="filter.title" placeholder="Пошук
файлів">
<button class="search-button"><v-icon
text large @click="search">search</v-
icon></button>
<button class="options-button"><v-icon
text large @click="formFilter =
!formFilter">arrow_drop_down</v-
icon></button>
<div class="searchResult" v-
if="searchFiles.length > 0">
<div class="resultItem" v-for="(item,
index) in searchFiles" :key="index"
@click="setFile(item)">
{{ item.title }}
</div>
</div>
<div class="formSearch" v-
if="formFilter">
Розмір файлів
<v-divider></v-divider>
<v-slider
:max="maxSize"
:min="0"
v-model="filter.size"
thumb-label="always"
>
<template v-slot:thumb-label="{ value }">
{{ (value / 1024 / 1024).toFixed(2) }}
</template>
</v-slider>
Типи файлів
<v-divider></v-divider>
<div class="typeBlock" v-for="(item,
index) in typeFiles" :key="index">
<v-checkbox
@change="searchInput"
:label="item.type"
:value="item.type"
hide-details
v-model="filter.types"
></v-checkbox>
</div>
</div>
</div>
<v-spacer></v-spacer>
<v-btn icon>
<v-icon x-large>notifications_none</v-
icon>
</v-btn>
<button @click="formFiles = true"
class="download">Завантажити <span
class="material-icons ml-
2">add</span></button>
<img class="avatar py-2"
:src="authUser.photo">
<span>{{authUser.name}}
{{authUser.surname}}</span>
<v-menu offset-y>
<template v-slot:activator="{ on, attrs
}">
<v-btn icon class="ml-1" v-bind="attrs"
v-on="on">
<v-icon large>keyboard_arrow_down</v-
icon>
</v-btn>
</template>
<v-list>
<v-list-item link to="/profile">
<v-list-item-title>Профіль</v-list-item-
title>
</v-list-item>
<v-list-item link>
<v-list-item-title
@click="logout">Вихід</v-list-item-title>
</v-list-item>
</v-list>
</v-menu>
</v-app-bar>
</template>
<script>
import { mapMutations } from 'vuex';
```

```
export default {
data() {
return {
formFiles: false,
formFilter: false,
file: null,
typeFiles: [],
maxSize: 0,
files: [],
searchFiles: [],
filter: {
title: "",
types: [],
size: 0,
}
}
},
computed: {
authUser() {
return this.$store.getters.authUser
}
},
created() {
this.getTypeFiles();
},
methods: {
...mapMutations([
"setFile"
]),
searchInput() {
if(this.filter.title.length > 3) {
axios.get('/api/search', {
params: this.filter
}).then(response => {
this.searchFiles = response.data;
})
} else {
this.searchFiles = [];
}
},
search() {
if (this.$route.name !== 'search') {
this.$router.push({ name: 'search',
query: this.filter})
}
},
getTypeFiles() {
axios.get('/api/file-
types').then(response => {
console.log(response.data)
this.typeFiles = response.data.types;
this.filter.size =
+response.data.maxSize;
this.maxSize = +response.data.maxSize;
})
},
handleFileUpload() {
let selectFiles = this.$refs.file.files;
for(let i = 0; i < selectFiles.length;
i++) {
this.files.push({
title: selectFiles[i].name,
description: "",
file: selectFiles[i],
uploadPercentage: 0,
size: selectFiles[i].size
})
}
},
submitFile() {
axios.post('/api/feed', {
text: "додав файли"
})
.then((response) => {
let formData = new FormData();
for (let index = 0; index <
this.files.length; index++) {
this.uploadFile(this.files[index],
response.data)
}
})
.then(() => {
// this.formFiles = false;
swal.fire({
icon: 'success',
title: 'Файли завантажено'
});
})
},
uploadFile(fileItem, idFeed) {
let formData = new FormData();
formData.append('file', fileItem.file);
formData.append('title', fileItem.title);
formData.append('description',
fileItem.description);
formData.append('idFeed', idFeed);
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)
}
)
},
logout() {
this.$store.dispatch('logout').then(() =>
{
this.$router.push('/')
})
}
}
}
</script>
```