МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ІНЖЕНЕРНИЙ НАВЧАЛЬНО – НАУКОВИЙ ІНСТИТУТ
ім. Ю.М. Потебні
ЗАПОРІЗЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ

КАФЕДРА ЕЛЕКТРОНІКИ, ІНФОРМАЦІЙНИХ СИСТЕМ ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Кваліфікаційна робота

перший (бакалаврський)
(рівень вищої освіти)

на тему **Розробка веб-каталогу для систем маркування вантажів будівництва**

Виконав: студент 4 курсу, групи 6.1219-пзс-з
спеціальності 121 Інженерія програмного забезпечення
(код і назва спеціальності)
освітньої програми Програмне забезпечення систем
(код і назва освітньої програми)
А.Ю. Прокопович
(ініціали та прізвище)
Керівник к.т.н., доцент, Н.П. Полякова
(посада, вчене звання, науковий ступінь, підпис, ініціали та прізвище)

Рецензент директор ТОВ «Дісітел» П.О. Лютий
(посада, вчене звання, науковий ступінь, підпис, ініціали та прізвище)

Запоріжжя
2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ІНЖЕНЕРНИЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ім. Ю.М. Потебні
ЗАПОРІЗЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ

Кафедра електроніки, інформаційних систем та програмного забезпечення
Рівень вищої освіти _____перший (бакалаврський)_____
Спеціальність ___121_Інженерія програмного забезпечення_____
(код та назва)
Освітня програма ___Програмне забезпечення систем_____
(код та назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри_____Тетяна КРИТСЬКА
“_01_” __березня_____2024_року

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

Прокоповичу Антону Юрійовичу
(прізвище, ім’я, по батькові)

1.Тема роботи Розробка веб-каталогу для систем маркування вантажів будівництва
керівник роботи _____Полякова Наталія Петрівна, к. т. н., доцент__
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом ЗНУ від ___26.12.2023 № 2212-с_____

2. Строк подання студентом кваліфікаційної роботи_____21.05.2024_____
3. Вихідні дані кваліфікаційної роботи бакалавра
- комплект нормативних документів ;
- технічне завдання до роботи.

4. Зміст розрахунково – пояснювальної записки (перелік питань, які потрібно розробити)
- огляд та збір літератури стосовно теми кваліфікаційної роботи;
- огляд та аналіз існуючих рішень та аналогів;
- аналіз сучасних технологій для створення клієнтської частини Веб застосунків;
- проектування Веб-застосунку;
- програмна реалізація застосунку.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
_____слайдів презентації

6. Консультанти розділів бакалаврської роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата Завдання прийняв

7. Дата видачі завдання 01.03.2024

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської роботи	Строк виконання етапів магістерської роботи	Примітка
1	Аналіз предметної області		виконано
2	Формулювання основної задачі дипломної роботи та узгодження її з науковим керівником		виконано
3	Аналіз існуючих методів рішення		виконано
4	Огляд та збір літератури стосовно теми кваліфікаційної роботи		виконано
5	Огляд та аналіз існуючих рішень та аналогів		виконано
6	Вивчення засобів для розробки Веб-застосунку		виконано
7	Визначення функціоналу застосунку, що розроблятиметься		виконано
8	Узгодження подальших дій з науковим керівником		виконано
9	Проектування бази даних та створення проекту застосунку		виконано
10	Проектування та конструювання застосунку		виконано
11	Програмна реалізація застосунку та його апробація		виконано
12	Оформлення звіту		виконано
13	Оформлення презентації. Отримання рецензій від опонентів		виконано

Студент _____ А.Ю. Прокопович
(підпис) (прізвище та ініціали)

Керівник роботи _____ Н.П. Полякова
(підпис) (прізвище та ініціали)

Нормоконтроль пройдено

Нормоконтролер _____ І.А. Скрипник
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Сторінок – 61

Рисунків – 12

Джерел – 8

Прокопович А. Ю. Розробка веб-каталогу вантажів будівництва: кваліфікаційна робота бакалавра спеціальності 121 «Програмне забезпечення систем» / наук. керівник Н. П. Полякова. Запоріжжя : ЗНУ, 2024, 61 с.

У сучасних умовах, будівництво потребує оптимізації будівельних процесів, та контрольованої звітності. Існують послуги моніторингу будівельних робіт за допомогою камер відеоспостереження, закріплених на підйомних будівельних кранах, що дозволяють в режимі реального часу відстежувати якість виконання робіт. Надання будівельним компаніям детальних аналітичних звітів на основі зібраних даних потребують чітко зазначених назв кожного вантажу, який використали за допомогою певного підйомного крану на протязі всього процесу будівництва. WEB-каталог стає ключовим чинником у покращені ефективності при оформлені звітності.

Мета проекту – забезпечення інформації що до існуючих категорій та типів всіх вантажів, що використовуються на певному будівельному майданчику. Основним завданням проекту є створення WEB-каталогу для оптимізації звітності.

Проведено аналіз існуючих WEB-каталогів. На основі цього аналізу був створений WEB-каталог із зручним орієнтуванням в категоріях вантажів, їх назв та фото. Реалізований веб-додаток дозволяє користувачам вибирати або шукати певні назви вантажів та отримувати їх фото. Мета системи – підвищення ефективності оформлення звітності та допомога користувачам орієнтуватися в усьому обсязі вантажів, що використовуються на певних будівництвах.

Ключові слова: *веб-каталог, вантаж, категорія, пошук фронтенд, бекенд, JavaScript, Svelte.*

ABSTRACT

Pages – 61

Drawings – 12

Sources – 8

Prokopovych A. Y. Development of a web catalog of construction cargo: qualification work of bachelor`s specialty 121 "System software" / science. Head N. P. Polyakova. Zaporizhzhia: ZNU, 61 p.

In modern conditions, construction requires optimization of construction processes and controlled reporting. There are services for monitoring construction work using video surveillance cameras mounted on lifting construction cranes, which allow monitoring the quality of work in real time. Providing construction companies with detailed analytical reports based on collected data requires the names of each load used by a specific crane throughout the construction process to be clearly specified. The WEB catalog becomes a key factor in improved efficiency in reporting.

The purpose of the project is to provide information on the existing categories and types of all cargoes used on a certain construction site. The main task of the project is to create a WEB catalog to optimize reporting.

An analysis of existing WEB catalogs was carried out. On the basis of this analysis, a WEB catalog was created with convenient orientation in cargo categories, their names and photos. The implemented web application allows users to select or search for specific cargo names and get their photos. The purpose of the system is to increase the efficiency of reporting and help users navigate the entire volume of cargo used on certain construction sites.

Keywords: web directory, cargo, category, frontend search, backend, JavaScript, Svelte.

ЗМІСТ

ВСТУП

Актуальність теми

WEB-каталоги актуальні та дуже поширені у сучасному світі. Користувачі шукають необхідну інформацію в багатьох цифрових ресурсах, як наприклад інтернет-магазини, енциклопедії, тощо. Актуальність створення будівельного веб-каталогу обумовлена кількома ключовими факторами, що впливають не тільки на сучасну будівельну галузь, а й на досвід користувача, котрий використовує його для певної мети.

Будівельні проекти стикаються з проблемою обмеження за часом. Застосунок дозволить значно скоротити час, витрачений на пошук певної назви та фото вантажу в інтернеті, що сприяє дотриманню строків виконання звітності та самого проекту.

Сучасний WEB-каталог може інтегруватися з логістичними сервісами, що забезпечить точне відстеження доставки кожного вантажу на будівельних майданчиках.

WEB-каталог дозволить користувачам своєчасно та ефективно оформити звіт роботи будівельних кранів, що використовуються на певних будівництвах, не допускаючи помилок при маркуванні кожного використаного вантажу.

Мета дослідження

Розробка простого застосунку для зручної орієнтації користувачів в усьому обсязі вантажів, що використовуються на певних будівництвах.

Завдання дослідження

Аналіз існуючих рішень щодо створення WEB-застосунку. Вивчення технологій HTML (Hyper Text Markup Language), CSS (Cascading Style Sheets) та їх основних функціональних можливостей з метою використання у створенні WEB-каталогу та розуміння принципів роботи.

Об’єкт дослідження

Об’єктом дослідження є процес розробки онлайн-каталогу.

Предмет дослідження

Предметом дослідження є вивчення основних принципів роботи HTML та CSS, аналіз сучасних тенденцій у розробці веб-застосунків для врахування актуальних вимог.

Методи дослідження

Аналіз і детальний огляд літературних джерел, підручників, статей та інших джерел, що стосуються технологій розробки HTML та CSS; аналіз популярних веб-каталогів з метою виявлення їх особливостей, накопичення досвіту для використання його на практиці.

Практичне значення одержаних результатів

Практичне значення одержаних результатів дослідження полягає у отриманні досвіду від користування необхідним інструментарієм. Результати дослідження можуть бути використані для створення веб-застосунків у майбутньому.

Глосарій

WEB-каталог – онлайн ресурс або веб-сайт, який містить інформацію про асортимент послуг, товарів або ресурсів, доступних для перегляду та пошуку. Може включати в себе категорії та підкатегоріях ресурсів, детальні описи, зображення, коментарі.

HTML (Hyper Text Markup Language) – мова розмітки, яка використовується для створення структури і вигляду веб-сторінок в інтернеті [2].

CSS (Cascading Style Sheets) – мова стилів, яка використовується для оформлення веб-сторінок, визначення їхнього зовнішнього вигляду та розміщення елементів на сторінці [2].

Користувач – особа, яка використовує веб-каталог для пошуку, перегляду та порівняння вантажів, які представленні на веб-сайті.

Вантаж – об’єкт або матеріал, що підіймається, переміщується, опускається та встановлюється за допомогою підйомного крану. Це може бути як будівельній матеріал так і обладнання, які потрібно перемістити на будівельному майданчику.

Підйомний кран – механічний пристрій, призначений для підняття та переміщення вантажів у вертикальному та горизонтальному напрямках. Використовується на будівельних майданчиках.

Інформаційна система – сукупність програмного забезпечення, апаратних даних та засобів, що призначені для зберігання, збирання та обробки інформації.

Інтерфейс користувача – графічний інтерфейс, який нажає можливість взаємодії користувача та інформаційної системи.

База даних – структуроване сховище даних, яке використовується для організації та збереження інформації про ресурси.

Backend – частина програмного забезпечення, яка відповідає за обробку даних та логіки бізнес-процесів на серверному рівні.

Frontend – частина програмного забезпечення, що відповідає за відображення інтерфейсу користувача та взаємодію з ним. Включає в себе всі елементи, що бачить користувач та з якими він взаємодіє у веб-додатку.

Оптимізація – процес покращення чи налаштування системи з метою забезпечення оптимальної ефективності.

1.1 Опис проблеми

Компанія Versatile займається розробкою технологічних рішень для будівельної індустрії. Вона створює системи, які використовують штучний інтелект (ШІ) для підвищеної ефективності та безпеки на будівельних майданчиках.

Одним із ключових аспектів діяльності Versatile є моніторинг та аналітика будівельних робіт. Використання сенсорів і камер для збору даних з будівельних майданчиків в режимі реального часу також дозволяє зробити аналітику даних для оптимізації процесів будівництва, що підвищує

продуктивність. Виявлення потенційних небезпек і попередження аварійних ситуацій сприяє покращенню умов праці робітників на будівельному майданчику. Управління і контроль за використанням матеріалів та техніки на будівельному майданчику дозволяють замовнику відстежувати ресурси.

Компанія Versatile надає інноваційні рішення, які допомагають будівельним компаніям працювати більш ефективно та безпечно, використовуючи сучасні технології.

Головним інструментом Versatile з моніторингу будівельних робіт є камери спостереження (див. Рис 1), що кріпляться на підйомних кранах які працюють на певному будівельному майданчику [4]. Ці камери дозволяють відстежувати роботу підйомного крана на протязі всього робочого процесу впродовж дня, підйом та опускання вантажів чи обладнання, їх перенесення та встановлення на будівельному майданчику. Впродовж робочого дня, камера спостереження надсилає отримані дані в режимі реального часу спеціальної програми моніторингу Piramid. Суть цієї програми полягає в створені циклів, від початку робочого дня і до його завершення. Кожен цикл складається з декількох етапів, які треба маркувати – рух підйомного крану до потрібного вантажу, закріплення вантажу на гак, перенесення та зняття його з гака (див. Рисунок 2). Також в циклах зазначається додаткова інформація про місця перенесення вантажів чи обладнання, процес (якщо потрібен, наприклад встановлення бетонної стіни) та назва самого вантажу. Маркування всіх циклів впродовж робочого дня будівельного майданчику призводить до отримання звітності про моніторинг.

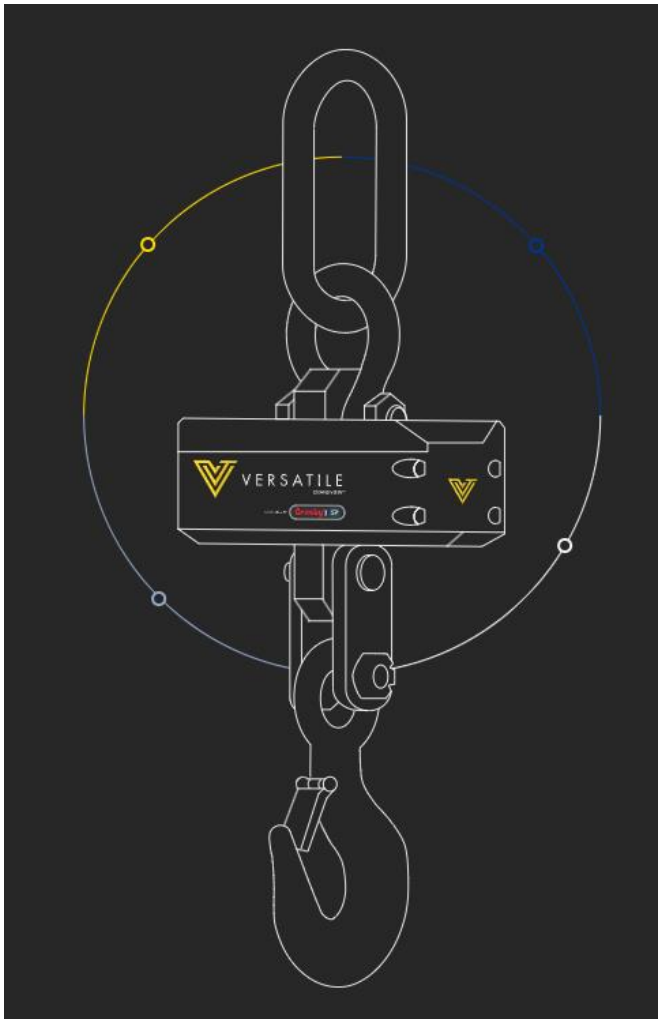

Рисунок 1 – *Камера спостереження за роботою підйомного крану*

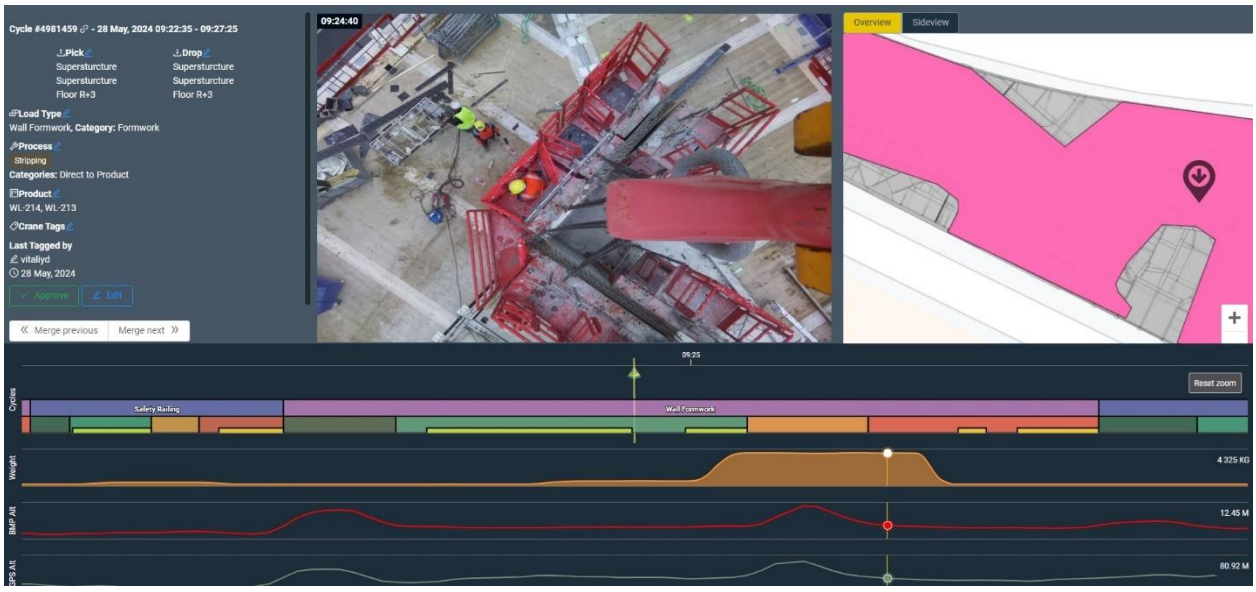

Рисунок 2 – *Цикл роботи підйомного крану в програмі Piramid*

Основною проблемою при оформлені циклів є внесення інформації про назву кожного вантажу вручну, оскільки кількість цих вантажів перебільшує відмітку в 200 пунктів і на це витрачається досить багато часу, що сприяє зниженню ефективності моніторингу. Як один із варіантів вирішення цієї проблеми можна було б вважати, що співробітник з оформлення циклів (далі користувач) повинен шукати у відкритих джерелах мережі інтернет інформацію про кожен вантаж, зокрема фото, але це призводить до ще більших витрат часу, оскільки назва певного вантажу не завжди відповідає результатам його пошуку. Вивчення кожного вантажу та відповідного йому зображення та назви теж не є рішенням цієї проблеми, оскільки враховується людський фактор – присутня висока ймовірність допущення помилок при оформленні звітності, що сприяє зменшенню ефективності моніторингу.

Враховуючи всі обставини було прийнято рішення про створення простого та зрозумілого користувачам веб-каталогу вантажів та обладнання, що використовуються на будівельних майданчиках. Маючи можливість пошуку як за відповідними категоріями так і за назвою певного вантажу, користувач повинен отримати відповідне зображення та переконатися в правильності отриманої інформації для внесення її в цикли та звітність.

1.2 Постановка задачі

Веб-каталог повинен бути простим та зрозумілим для користувачів. Не допускається присутність на його сторінках зайвої інформації, яка не стосується тематики вантажів та обладнання, як наприклад реклама. Кожен тип вантажу чи обладнання повинен відповідати певній категорії для фільтрації та мати чітке та зрозуміле відповідне зображення. Короткий опис вантажів теж має бути присутній та містити в собі його використання на відповідних будівельних майданчиках. На головній сторінці має бути присутній пошуковий рядок для швидкого доступу до вантажу за його назвою.

Веб-каталог повинен відповідати сучасним системним вимогам, швидко завантажуватися та стабільно працювати.

1.3 Огляд існуючих методів рішення

Веб-каталоги відіграють важливу роль в організації та представленні великої кількості інформації в Інтернеті. Вони збирають та класифікують різноманітні ресурси або продукти за певними темами, що значно полегшує користувачам пошук потрібної інформації. Веб-каталоги можуть бути самостійними проектами або інтегрованими частинами більших платформ, таких як інтернет-магазини, форуми, ресурси з відкритим кодом та деякі пошукові системи. Ці каталоги надають користувачам зручний спосіб знайти те, що їм потрібно, серед величезного обсягу інформації в мережі. Розглянемо деякі з найпопулярніших веб-каталогів, які завоювали визнання користувачів завдяки своїй зручності та широкому охопленню ресурсів.

Edmunds (див. Рисунок 3) – це веб-сайт, що надає інформацію про автомобілі.

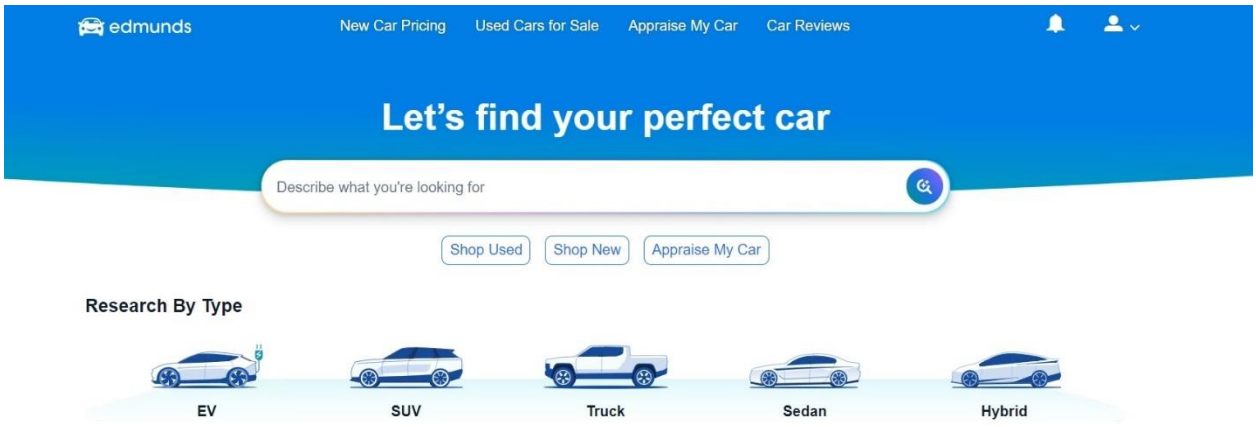

Рисунок 3 – *Головна сторінка Edmunds*

Edmunds є популярним сервісом надання інформації про автомобілі та пропонує широкий спектр послуг, такі як огляди та рейтинги нових та вживаних автомобілів, інструменти порівняння, пошук за зручною системою фільтрації, новини та високоякісні відповідні фотографії. Це потужний інструмент зокрема для автомобільних покупців та не відповідає всім вимогам проекту, оскільки містить в собі багато зайвої інформації.

Equipment Trader (див. Рис 4)– це веб-сайт, який пропонує широкий спектр послуг, пов’язаних з купівлею та продажом вживаного будівельного та сільськогосподарського обладнання. Сайт має зручну систему пошуку та фільтрації певного обладнання, роздріб на категорії та додаткову інформацію про кожен ресурс, відповідні фото-зображення. Завдяки зручному інтерфейсу та відсутності зайвої інформації користувачам зручно орієнтуватися на сайті при користуванні. За відповідне просте і зрозуміле оформлення та тематику було розглянуто саме оформлення та систему пошуку сайту.

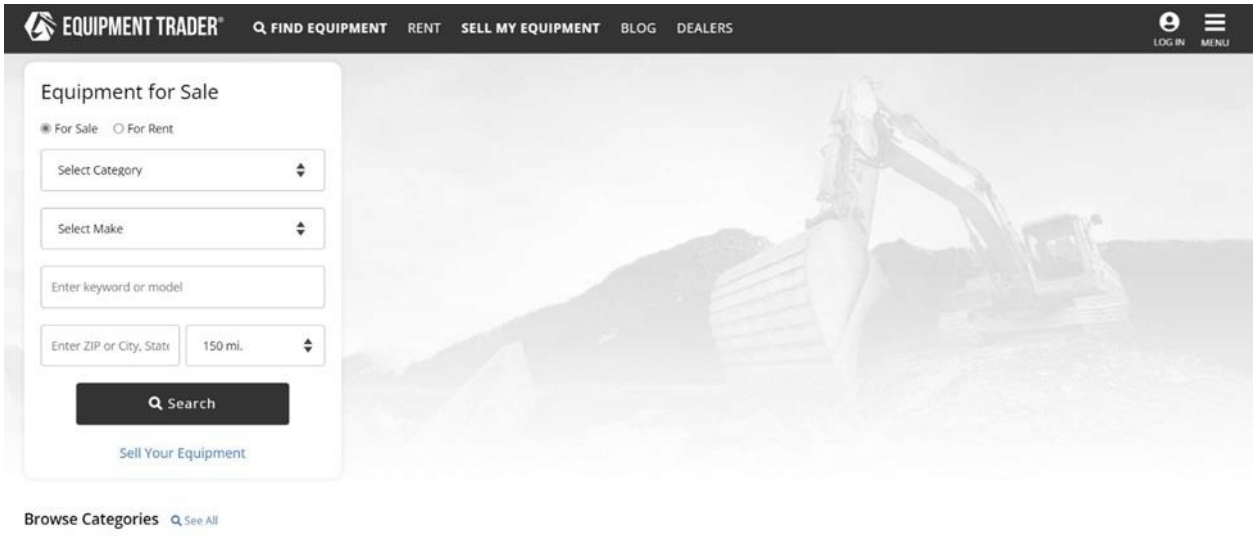

Рисунок 4 – *Головна сторінка Equipment Trader*

Unsplash (див. Рисунок 5) – веб-сайт який надає послуги з пошуку безкоштовних зображень. На сайті пропонуються тисячі високоякісних фотографій які доступні для комерційного та некомерційного використання завдяки ліцензії. Фотографії організовані в тематичні колекції та категорії, такі як архітектура, природа, люди, технології, їжа та багато інших, що значно полегшує пошук відповідних зображень. Потужний інструмент пошуку дозволяє знаходити фотографії за ключовими словами а зручна навігація дозволяє без проблем орієнтуватися на сайті. Присутній додатковий функціонал, який включає в себе вбудований редактор фотографій.

Веб-каталог Unsplash представляє собою цінний ресурс головним чином для дизайнерів, маркетологів та будь кого, хто потребує високоякісних фотографій для своїх проектів. Порівнюючи цей сайт з майбутнім проектом було взято за основу систему пошуку та фільтрацію за відповідними категоріями.

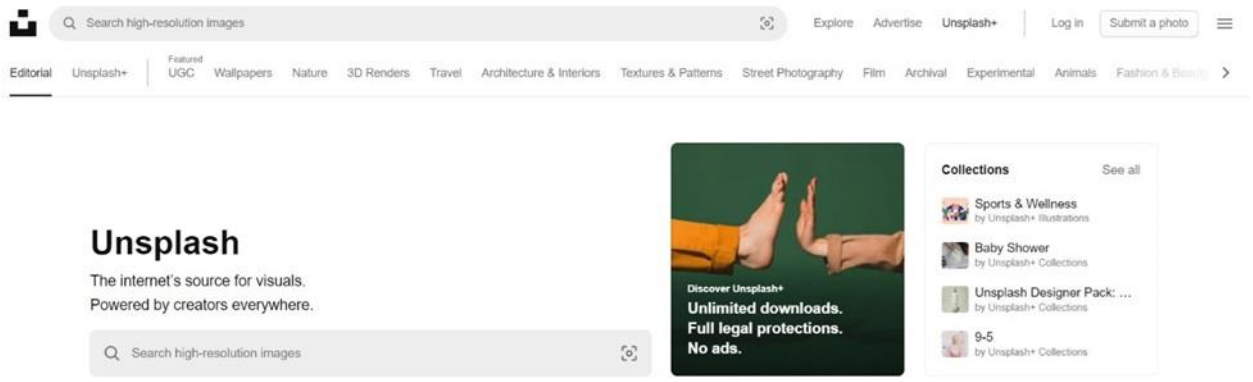

Рисунок 5 –*Головна сторінка Unsplash*

Amazon (див. Рисунок 6) – веб-сайт, який є однією з найбільших у світі інтернет-платформ для торгівля, що надає широкий спектр товарів та послуг. На сайті представлені мільйони товарів у різних категоріях, включаючи електроніку, одяг, книги, іграшки та багато іншого, які пропонують різні продавці, включаючи як малі бізнеси так і великі бренди.

Потужний пошуковий інструмент має зручну фільтраційну систему, що дозволяє користувачу здійснювати пошук товарів за ключовими словами, категоріями та іншими критеріями. Присутня можливість звуження результатів пошуку за ціною, рейтингом, типом продавця та іншими параметрами.

Присутня система відгуків про товари та продавців – система рейтингів, що дозволяє оцінити їх популярність.

Інтерактивність та зручність користування обумовлені наявністю мобільною версією сайту (додатку), що дозволяє мати доступ до сайту майже з будь якого пристрою.

Веб-каталог Amazon – потужний інструмент як для покупців так і для продавців, що забезпечує доступ до величезного асортименту товарів, інтуїтивний інтерфейс та додаткові сервіси для підвищення зручності та безпеки покупок.

Зручний інтерфейс, зручна система пошуку та структурованість – саме це було взято за приклад для майбутнього проекту під час огляду сайту.

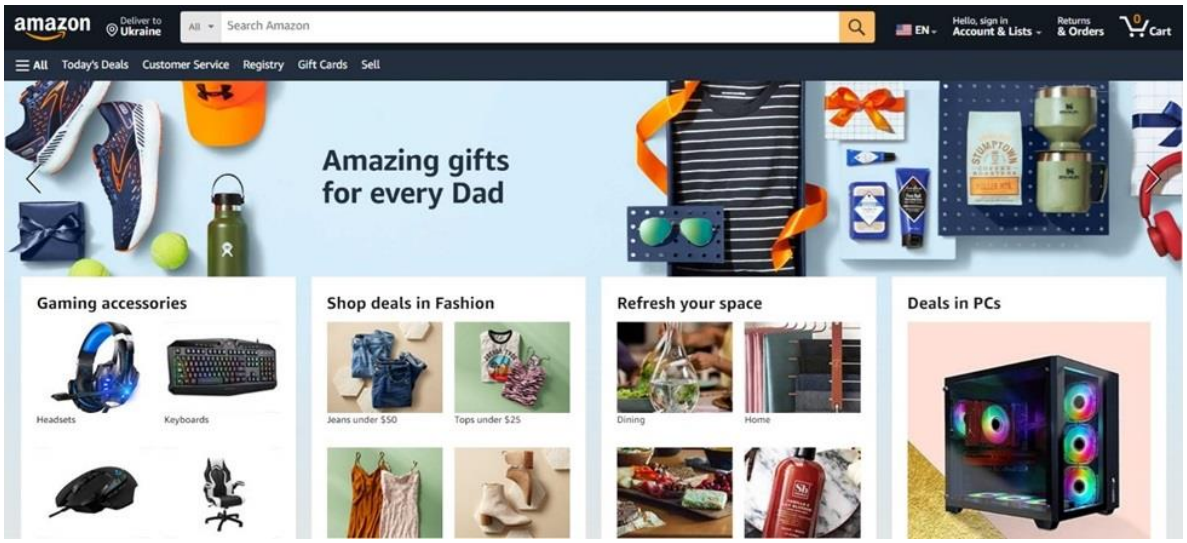

Рисунок 6 – *Головна сторінка Amazon*

TCC Catalog (див. Рис 7) – наявний каталог від компанії Versatile. Він дозволяє переглянути інформацію про наявні типи вантажів та обладнання, таку як зображення та короткий опис, здійснити пошук по категоріям та типу вантажу. Сайт має досить зручну систему фільтрації але відсутнє саме поле для пошуку певного вантажу, який шукатиме користувач.

Рисунок 7 *Головна сторінка TCC Catalog*

1.4 Висновок

Роблячи огляд перерахованих вище сайтів, зроблено висновок, що веб-каталог повинен мати ряд функціональних можливостей, серед яких слід виділити:

1. **Пошук.** Користувачі повинні мати можливість легко здійснювати пошук того, що їм потрібно за допомогою ключових слів;
2. **Перегляд.** Веб-каталог повинен мати чітку та зручну структуру, зручний та простий інтерфейс, що дозволить користувачам легко та зручно орієнтуватися в ньому;
3. **Структурована система.** Кожен тип вантажу або тип обладнання повинні відповідати певній категорії;
4. **Опис.** Кожен тип вантажу або тип обладнання повинні мати короткий опис, що може містити в собі наявність на певних будівництвах або опис використання.

2 ВИМОГИ ДО ОТОЧЕННЯ

2.1Вимоги до апаратного забезпечення

Процесор

- Мінімально: Intel Core i3 або еквівалентний.
- Рекомендовано: Intel Core i5 або еквівалентний.

Пам’ять

- Мінімально: 4 ГБ RAM.
- Рекомендовано: 8 ГБ RAM або більше.

Твердотільний накопичувач SSD

- Мінімально: 10 ГБ вільного місця.
- Рекомендовано: 20 ГБ вільного місця або більше.

Інтернет-з’єднання:

- Мінімально: 10 Мбіт/с.
- Рекомендовано: 100 Мбіт/с або більше.

2.2Вимоги до програмного забезпечення

Операційна система

- Windows 10 або 11, 64-bit.
- Браузери.
- Google Chrome версії 80 або вище, Microsoft Edge версії 125.

2.3 Вимоги до користувачів

Ролі користувачів

Адміністратор

Функції в системі:

- Управління контентом (Створення, редагування, видалення).
- Налаштування системи.

Функції в організації:

- Забезпечення безперебійної роботи веб-каталогу.
- Впровадження політик та процедур використання каталогу.

Доступні можливості системи:

- Повний доступ до всіх функцій системи.
- Доступ до аналітики та звітів.

Вимоги до знань, умінь та навиків:

- Знання основ веб-розробки та адміністрування.
- Уміння працювати з системами управління контентом (CMS).
- Навички модерації та управління користувачами.
- Базові знання в галузі інформаційної безпеки.

Користувач

Функції в системі:

- Перегляд контенту.

Функції в організації:

- Потенційний користувач.

Доступні можливості системи:

- Обмежений доступ до перегляду контенту.

Вимоги до знань, умінь і навиків:

- Навички роботи з інтернетом, веб-браузером.
- Базові навички пошуку інформації.

2.4 Організаційні вимоги

Для успішного впровадження веб-каталогу в організації-замовника, необхідно ввести зміни як в структуру організації, так і в її виробничих процесах:

Системний адміністратор

Введення ставки системного адміністратора з приблизною занятістю в половину робочого дня. Це необхідно для забезпечення стабільної роботи веб-каталогу, вирішення технічних проблем та оновлення програмного забезпечення.

Резервне копіювання

Організація щоденного резервного копіювання даних веб-каталогу. Це завдання необхідно покласти на головного менеджера або системного адміністратора.

Навчання персоналу

Проведення навчання для співробітників, які будуть працювати з веб-каталогом. Це включає базові навички роботи з системою, методи резервного копіювання та безпеки даних.

Технічна підтримка

Укладання договору з технічною підтримкою або зовнішніми фахівцями для регулярного обслуговування веб-каталогу та вирішення складних технічних питань.

Ці зміни допоможуть забезпечити безперебійну роботу веб-каталогу, що в свою чергу позитивно вплине на оптимізацію звітності організації.

3 АРХІТЕКТУРА СИСТЕМИ

Веб-каталог вантажів будівництва побудований на основі архітектури MVC (Модель-Вигляд-Контролер), що забезпечує чіткий розподіл відповідальності між компонентами, спрощуючи розробку та підтримку застосунку. Модель відповідає за дані та їх обробку, Вигляд формує інтерфейс користувача, а Контролер обробляє запити та координує взаємодію між Моделлю та Виглядом. Такий підхід сприяє підвищенню гнучкості, масштабованості та зручності внесення змін до веб-каталогу. Загальний варіант взаємодії між елементами зображено на рисунку 8.

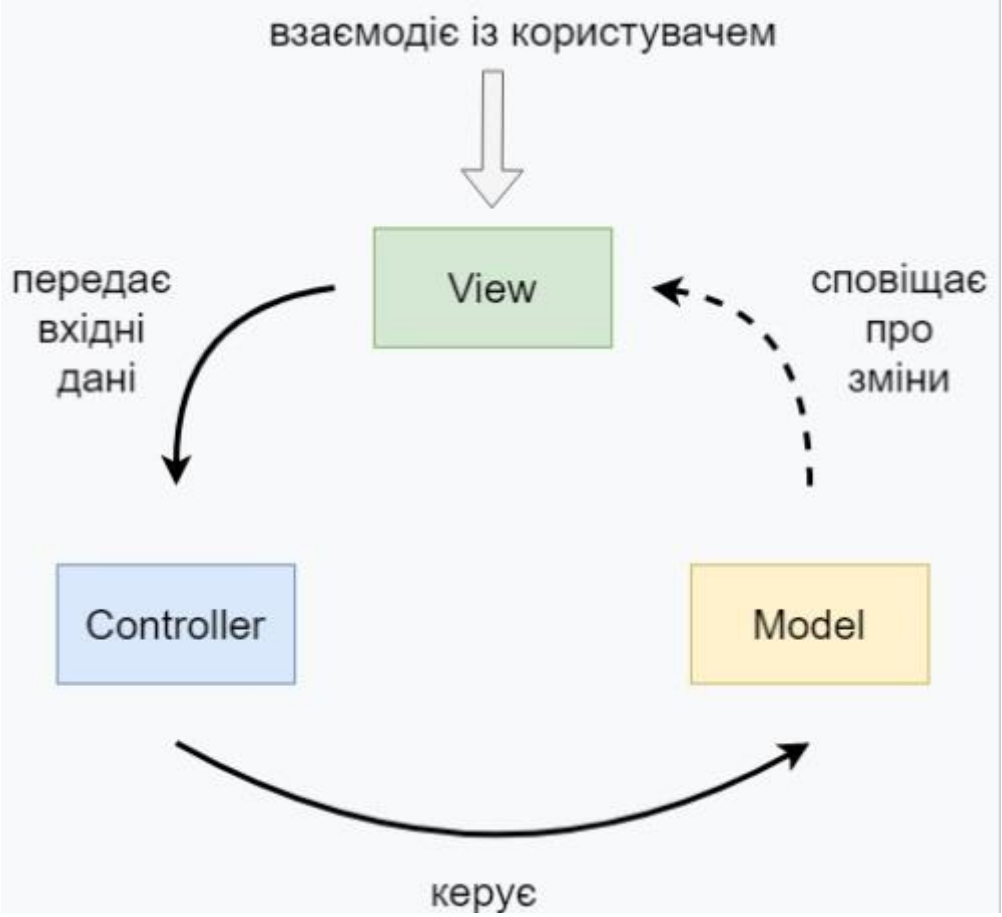

Рисунок 8 – *Схема взаємодії елементів веб-каталогу з базою даних за шаблоном MVC*

Опис взаємодії:

Користувач виконує дії, такі як перегляд каталогу, пошук вантажу (див. Рисунок 9).

Веб-браузер (View) відображає дані користувачеві і передає його дії Контролеру. Це може бути HTML-сторінка або форма.

Контролер обробляє запити від веб-браузера, викликає відповідні методи Моделі і повертає результат назад веб-браузеру.

Модель взаємодіє з базою даних для отримання, збереження та оновлення даних.

База даних зберігає дані про продукти, категорії, користувачів та іншу інформацію, необхідну для роботи веб-каталогу.

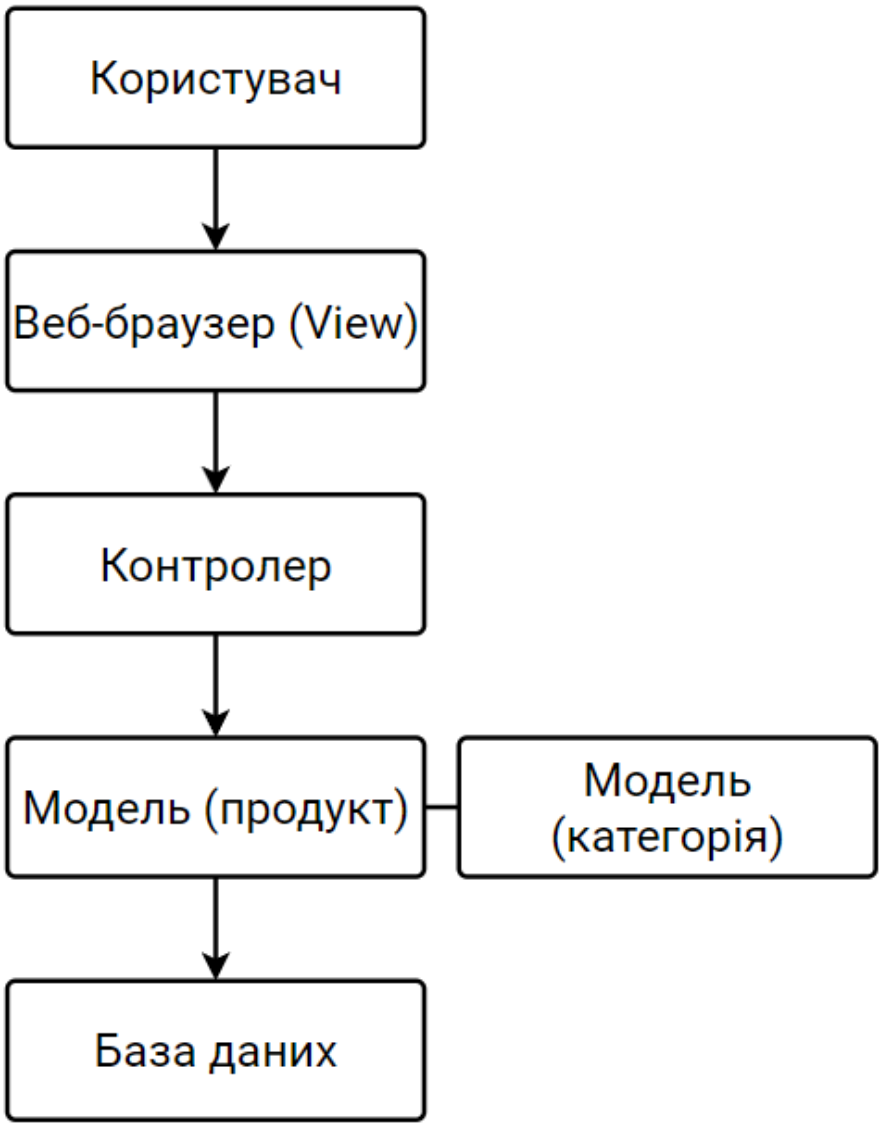

Рисунок 9 – *Діаграма взаємодії між компонентами*

Процес взаємодії:

1. Користувач надсилає запит через Веб-браузер (наприклад, переглянути список продуктів).
2. Веб-браузер надсилає цей запит Контролеру.
3. Контролер обробляє запит, взаємодіючі з Моделлю.
4. Модель виконує запит до Бази даних для отримання необхідної інформації.
5. База даних повертає дані Моделі.
6. Модель передає дані Контролеру.
7. Контролер передає отримані дані Виду.
8. Веб-браузер (View) відображає дані користувачеві.

4 ФУНКЦІОНАЛЬНІ ВИМОГИ

Система повинна дозволяти адміністратору:

1. Додавати до каталогу нові вантажі, вказуючи їх назву, категорію, опис та зображення.
2. Редагувати будь яку інформацію про існуючий в каталозі вантаж.
3. Видаляти вантаж з каталогу.
4. Створювати нову категорію для класифікації вантажів.
5. Редагувати категорії існуючих в каталозі вантажів.
6. Видаляти категорії з каталогу.

Система повинна дозволяти користувачу:

1. Переглядати каталог вантажів.
2. Сортувати список вантажів за певними категоріями.
3. Шукати потрібний вантаж за ключовими словами, які можуть бути в назві.
4. Переглядати всю наявну інформацію про вантаж, включаючи його зображення та опис.

5. ВИМОГИ ІНТЕРФЕЙСУ

Вимоги до інтерфейсу веб-каталогу включають кілька ключових аспектів, що забезпечують зручність використання, доступність та ефективність роботи з каталогом.

Загальні вимоги

1. Простота та інтуїтивність. Зрозуміле та логічне меню забезпечить чітку навігацію а мінімалістичний дизайн полегшить орієнтування користувача в застосунку;
2. Адаптація інтерфейсу до різних розмірів екрану без втрати функціональності;
3. Навігація за допомогою периферійних пристроїв, таких як миша та клавіатура, роблять веб-каталог доступним до використання.

6 ІНШІ ВИМОГИ

6.1 Вимоги до надійності

Оцінимо вірогідність виникнення, можливі наслідки та збитки від нештатних ситуацій (ризиків), що можуть виникнути у зв’язку з використанням системи.

1. Відмова устаткування
    - Вірогідність: середня.
    - Можливі наслідки: недоступність веб-каталогу, втрата доступу до даних.
    - Збитки: зниження продуктивності що до звітності.
    - Необхідність заходів: висока.
2. Виявлення помилок різного ступеня тяжкості
    - Вірогідність: висока.
    - Можливі наслідки: некоректне відображення інформації, негативний досвід користувачів.
    - Збитки: зниження довіри до сайту.
    - Необхідність заходів: висока.
3. Порушення безпеки
    - Вірогідність: висока.
    - Можливі наслідки: витік даних, репутаційні втрати.
    - Збитки: втрати від відновлення.
    - Необхідність заходів: висока.
4. Втрата або пошкодження даних
    - Вірогідність: середня.
    - Можливі наслідки: втрата важливої інформації, відновлення з резервних копій.
    - Збитки: час і ресурси на відновлення.
    - Необхідність заходів: висока.
5. Різка зміна виробничого процесу

- Вірогідність: низька.
- Можливі наслідки: невідповідність новим вимогам, потреба в додаткових ресурсах.
- Збитки: затримка в роботі, додаткові витрати на навчання.
- Необхідність заходів: середня.

Заходи щодо управління ризиками

1. Резервне копіювання даних
    - Процедура: регулярне обслуговування серверів, перевірка на наявність збоїв.
    - Періодичність: щотижнево.
    - Відповідальні: системній адміністратор.
    - Категорії користувачів: адміністратор.
2. Моніторинг стану устаткування
    - Процедура: регулярне обслуговування серверів, перевірка на наявність збоїв.
    - Періодичність: щотижнево.
    - Відповідальні: системній адміністратор.
    - Категорії користувачів: адміністратор.
3. Регулярні оновлення програмного забезпечення
    - Процедура: встановлення оновлень та патчів для CMS та інших компонентів системи.
    - Періодичність: щомісяця або при наявності нових оновлень.
    - Відповідальні: системній адміністратор.
    - Категорії користувачів: адміністратор.
4. Аудит безпеки
    - Процедура: перевірка системи на вразливості, проведення тестів на проникнення.
    - Періодичність: щоквартально.
    - Відповідальні: відповідальний за інформаційну безпеку.

- Категорії користувачів: адміністратор.

5. Навчання персоналу
    - Процедура: навчання співробітників основам інформаційної безпеки та роботі з веб-каталогом.
    - Періодичність: щорічно або при зміні програмного забезпечення.
    - Відповідальні: відповідальний за навчання.
    - Категорії користувачів: всі співробітники, які працюють з веб-каталогом.

Для забезпечення стабільної роботи веб-каталогу необхідно впровадити заходи з управління ризиками, які включають резервне копіювання, регулярне оновлення програмного забезпечення, моніторинг стану устаткування, аудит безпеки та навчання персоналу. Ці заходи допоможуть мінімізувати ймовірність виникнення нештатних ситуацій і знизити їх наслідки.

6.2 Вимоги до продуктивності

Вимоги до продуктивності веб-каталогу залежать від кількох ключових аспектів, таких як швидкість обробки запитів, здатність обслуговувати певну кількість одночасних користувачів, час завантаження сторінок та надійності системи.

Детальний перелік вимог до продуктивності:

1. Час відгука сервера
    - Вимога: час відгуку на запити до сервера не повинен перевищувати 200 мс для 95% запитів.
    - Обґрунтування: користувачі очікують швидкої реакції від веб-додатків, особливо для таких операцій, як перегляд списку вантажів чи пошук.
2. Час завантаження сторінки

- Вимога: середній час завантаження сторінки не повинен перевищувати 3 секунд при середньому навантаженні.
- Обґрунтування: швидке завантаження сторінки покращує користувацький досвід і знижує відмови користувачів від використання додатку.

3. Підтримка одночасних користувачів
    - Вимога: система повинна підтримувати не менше 100 одночасних користувачів без значної деградації продуктивності.
    - Обґрунтування: це забезпечить нормальну роботу додатку під час пікових навантажень.
4. Обробка запитів
    - Вимога: система повинна обробляти не менше 50 запитів на секунду (IPS) без зниження продуктивності.
    - Обґрунтування: ця вимога гарантує, що система зможе впоратися з навантаженням навіть при збільшенні кількості запитів.
5. Надійність і безперебійність роботи
    - Вимога: система повинна забезпечувати безперервну роботу з часом простою не більше 1% на місяць.
    - Обґрунтування: висока надійність забезпечує доступність додатку для користувачів у будь-який час.
6. Масштабованість
    - Вимога: архітектура додатку повинна підтримувати горизонтальне масштабування для збільшення потужності в разі зростання кількості користувачів або обсягу даних.
    - Обґрунтування: масштабованість забезпечує можливість розширення системи без значних змін в архітектурі.

Обґрунтування вимог до продуктивності:

1. Високий рівень користувацького досвіду: швидкий та чуйний веб-каталог вантажів забезпечує позитивний досвід користувача, дозволяючи швидко знаходити потрібну інформацію та виконувати дії без затримок. Це сприяє задоволеності користувачів, збільшує їхню лояльність та ймовірність повторного використання каталогу.

2. Ефективність обслуговування: здатність обробляти велику кількість одночасних користувачів та запитів є критичною для забезпечення безперебійної роботи веб-каталогу, особливо в періоди пікового навантаження. Це гарантує, що всі користувачі отримають доступ до інформації та зможуть виконувати необхідні операції без затримок чи помилок.

3. Надійність: висока надійність та мінімальний час простою є важливими для будь-якого веб-застосунку, особливо для тих, що використовуються в комерційних цілях. Надійний веб-каталог забезпечує постійний доступ до інформації про вантажі, що є критичним для бізнесу, який залежить від своєчасного виконання замовлень та доставки товарів.

4. Масштабованість: Можливість масштабування системи є важливою для забезпечення її довгострокової життєздатності. Зі зростанням бізнесу та збільшенням кількості користувачів та обсягу даних, масштабованість дозволяє веб-каталогу адаптуватися до нових вимог, зберігаючи при цьому високу продуктивність та надійність.

Висновок

Дотримання цих вимог забезпечить створення продуктивного, надійного та масштабованого веб-каталогу, який зможе задовільнити потреби користувачів та забезпечити високий рівень задоволеності при використанні додатку.

7 ПРОЄКТ ПРОГРАМНОЇ СИСТЕМИ

7.1 Засоби реалізації

Для створення передового веб-каталогу вантажів будівництва, який відповідає сучасним вимогам до функціональності, продуктивності та зручності використання, було ретельно підібрано комплекс інноваційних технологій та інструментів. Цей вибір гарантує не лише ефективність процесу розробки, але й високу продуктивність, надійність та масштабованість кінцевого продукту, що дозволить йому успішно адаптуватися до зростаючих потреб замовника та забезпечити бездоганний користувацький досвід.

Backend

Node.js

Обґрунтування: Node.js є платформою JavaScript для розробки швидких та масштабованих мережевих застосунків [6]. Використання JavaScript як єдиної мови програмування для фронтенду та бекенду спрощує розробку та дозволяє використовувати спільний код. Асинхронна, неблокуюча модель вводу-виводу Node.js забезпечує ефективну обробку великої кількості запитів, що робить його ідеальним для високопродуктивних застосунків реального часу [3]. Крім того, величезна кількість доступних бібліотек та модулів npm дозволяє легко розширювати функціональність та прискорювати розробку.

Використання: Node.js виконує роль серверного середовища, де виконується JavaScript-код бекенду [6]. Він обробляє вхідні HTTP-запити від клієнтів, взаємодіє з базою даних SQLite для отримання та оновлення інформації про вантажі та категорії, а потім формує відповіді у форматі JSON та відправляє їх назад клієнту. Node.js забезпечує виконання логіки застосунку, включаючи обробку даних, маршрутизацію, аутентифікацію та інші необхідні операції на сервері.

Express.js

Обґрунтування: Express.js – це веб-фреймворк для Node.js. Мінімалістичний підхід дозволяє легко створювати масштабовані API та веб-сервери, налаштовуючи їх під конкретні потреби проекту. Велика кількість доступних middleware розширює функціональність фреймворку, забезпечуючи обробку запитів, маршрутизацію, аутентифікацію та інші важливі аспекти веб-розробки. Активна спільнота та велика кількість навчальних ресурсів роблять Express.js легким для вивчення та використання, що прискорює процес розробки та знижує поріг входження для новачків.

Використання: Express.js використовується для створення логіки бекенду, що включає визначення маршрутів (endpoints) для обробки HTTP-запитів (GET, POST, PUT, DELETE) від клієнта. За допомогою middleware Express.js обробляє запити, виконує перевірку даних, аутентифікацію користувачів та інші необхідні операції. Фреймворк дозволяє організувати структуру серверного застосунку, розділяючи логіку на окремі модулі та забезпечуючи зручну взаємодію з базою даних для отримання та оновлення інформації про вантажі та категорії.

SQLite

Обґрунтування: SQLite – це вбудована реляційна база даних, яка не вимагає окремого сервера [7]. Її архітектура дозволяє вбудовувати SQLite безпосередньо в застосунок, що спрощує розгортання та зменшує накладні витрати. SQLite підтримує більшість стандартних SQL-запитів, що забезпечує знайомий інтерфейс для роботи з даними. Крім того, вона є крос платформною та має відкритий вихідний код, що робить її доступною та гнучкою для широкого спектру застосувань.

Використання: SQLite використовується для створення таблиць, що містять інформацію про вантажі (наприклад, назва, опис, вага, розміри, статус доставки) та категорії вантажів. За допомогою SQL-запитів бекенд може додавати нові вантажі, отримувати списки вантажів з фільтрацією та сортуванням, оновлювати інформацію про існуючі вантажі та видаляти непотрібні записи. SQLite забезпечує структуроване зберігання даних, що

дозволяє бекенду ефективно керувати інформацією про вантажі та забезпечувати її цілісність.

SQLite3

Обґрунтування: SQLite3 – це модуль Node.js для роботи з SQLite базами даних. Він надає зручний інтерфейс для виконання SQL-запитів, обробки результатів та управління транзакціями, що спрощує розробку застосунків, які взаємодіють з SQLite. SQLite3 є добре підтримуваним та документованим модулем, що забезпечує стабільність та надійність роботи з базами даних в Node.js середовищі [7].

Використання: SQLite3 дозволяє створювати, модифікувати та видаляти записи про вантажі та їх категорії. За допомогою SQL-запитів, бекенд може отримувати списки вантажів з фільтрацією за категоріями, сортуванням та пошуком. Модуль також забезпечує механізми для оновлення інформації про вантажі (наприклад, зміна статусу доставки) та видалення застарілих записів. SQLite3 дозволяє бекенду ефективно управляти даними про вантажі, забезпечуючи швидкий доступ до необхідної інформації та підтримуючи цілісність бази даних.

Cors

Обґрунтування: Cors (Cross-Origin Resource Sharing) є важливим механізмом для забезпечення безпечного обміну ресурсами між різними доменами. Він дозволяє веб-серверу вказувати, Які джерела (домени, протоколи, порти) мають право робити запити до нього, тим самим запобігаючи несанкціонованому доступу до даних. Використання Cors є необхідним для сучасних веб-застосунків, які часто використовують API та ресурси з різних доменів, забезпечуючи безпеку та надійність роботи.

Використання: Cors налаштовує HTTP-заголовки у відповідях сервера, дозволяючи фронтенду (веб-каталогу), який може бути розташований на іншому домені чи порту, робити запити до бекенду для отримання даних про вантажі та категорії. Це забезпечує коректну роботу веб-каталогу, дозволяючи

йому відображати актуальну інформацію, отриману з бекенду, навіть якщо вони розташовані на різних серверах або використовують різні протоколи.

Frontend

Svelte

Обґрунтування: Svelte – це сучасний фреймворк для створення користувацьких інтерфейсів, який вирізняється серед інших фреймворків, таких як React чи Vue, своїм підходом до компіляції. Замість використання віртуального DOM під час виконання, Svelte компілює компоненти в оптимізований JavaScript під час збірки. Це призводить до меншого розміру бандлу, швидкого рендерингу та кращої продуктивності в цілому [5]. Крім того, Svelte пропонує більш декларативний та менш шаблонний синтаксис, що робить його більш інтуїтивним та легким для вивчення. Завдяки своїй легкості та ефективності, Svelte є привабливим вибором для розробників, які шукають сучасний та продуктивний інструмент для створення веб-інтерфейсів.

Використання: Svelte використовується для створення компонентів інтерфейсу веб-каталогу, таких як список вантажів, фільтри, детальна інформація про вантаж тощо. Компоненти в Svelte дозволяють легко розділяти функціональність на окремі частини, що спрощує розробку та підтримку. Динамічне оновлення стану та реактивність дозволяють створювати інтуїтивно зрозумілі та швидкі інтерфейси, що забезпечує кращий користувацький досвід [5].

Svelte Routing

Обґрунтування: Svelte Routing – це бібліотека для маршрутизації у Svelte-застосунках. Вона пропонує інтуїтивно зрозумілий та декларативний спосіб визначення маршрутів та управління переходами між ними, використовуючи знайомий синтаксис Svelte. Крім того, Svelte Routing забезпечує ефективне оновлення лише тих компонентів, які залежать від зміни маршруту, що сприяє високій продуктивності додатку. Її гнучкість та розширюваність дозволяють легко налаштовувати маршрутизацію під специфічні потреби проекту.

Використання: Svelte Routing використовується для організації навігації між різними розділами веб-каталогу. Це включає створення маршруту для кожного розділу або сторінки каталогу, що дозволяє користувачам легко переміщуватися між ними, зберігаючи при цьому стан додатку. Маршрути можуть бути налаштовані для обробки різних параметрів URL, дозволяючи передавати динамічні дані та забезпечуючи гнучкість і зручність у навігації.

Fetch API

Обґрунтування: Fetch API є сучасним та зручним інтерфейсом для роботи з мережевими запитами в JavaScript [1]. Він пропонує простий та інтуїтивно зрозумілий спосіб відправки запитів та обробки відповідей, використовуючи проміси (promises) для асинхронної роботи. Fetch API підтримує різні типи запитів (GET, POST, PUT, DELETE тощо) та дозволяє налаштовувати заголовки, тіло запиту та інші параметри. Крім того, він має вбудовану підтримку обробки JSON та інших форматів даних, що робить його зручним інструментом для розробки веб-застосунків, які взаємодіють з серверами та API.

Використання: Fetch API використовується для відправки запитів до бекенду з метою отримання даних про вантажі та категорії, а також для відправки даних на сервер. Для отримання даних застосовується метод fetch(url, options), де url вказує на адресу ресурсу, а options містять конфігурацію запиту, наприклад, метод (GET, POST тощо), заголовки та тіло запиту. Відповідь обробляється за допомогою промісів, що дозволяє працювати з даними асинхронно, полегшуючи інтеграцію з API та сервісами, які надають необхідну інформацію.

Classnames

Обґрунтування: Classnames є ідеальним інструментом для управління CSS-класами у Svelte компонентах завдяки своїй простоті та гнучкості. Classnames дозволяє легко комбінувати та маніпулювати класами, використовуючи інтуїтивно зрозумілий синтаксис. Це особливо корисно у

Svelte, де часто потрібно динамічно змінювати класи на основі стану компонента. Classnames спрощує цей процес, роблячи код більш читабельним та легким для підтримки, що особливо важливо при розробці складних компонентів з великою кількістю умовних класів. Classnames є легкою бібліотекою без залежностей, що робить її ідеальним доповненням до будь-якого Svelte проекту.

Використання: Classnames використовується для управління стилями елементів інтерфейсу, надаючи можливість динамічно змінювати класи на основі стану компонента. Наприклад, при виділенні активного фільтра або зміни вигляду кнопок залежно від дії користувача, Classnames дозволяє легко та ефективно додавати, видаляти чи комбінувати класи, роблячи код більш чистим і зручним для читання та підтримки.

Загальні

HTML, CSS, JavaScript

Обґрунтування: HTML, CSS та JavaScript є основоположними технологіями веб-розробки, які разом забезпечують створення інтерактивних та функціональних веб-сайтів. HTML надає структуру контенту, визначаючи його елементи та їх взаємозв'язки [2]. CSS відповідає за візуальне оформлення, дозволяючи налаштовувати зовнішній вигляд елементів, їх розташування та анімацію. JavaScript надає інтерактивність та динамічність, дозволяючи реалізовувати складну логіку, обробку подій та взаємодію з користувачем [1]. Така комбінація дозволяє створювати сучасні веб-додатки з багатим функціоналом та привабливим дизайном.

Використання: HTML використовується для створення основної структури сторінок, визначаючи заголовки, абзаци, посилання, зображення та інші елементи контенту. CSS забезпечує візуальне оформлення цих елементів, включаючи кольори, шрифти, відступи та розташування на сторінці. Завдяки CSS можна створювати адаптивний дизайн, який підлаштовується під різні розміри екранів. JavaScript додає інтерактивність і динаміку, дозволяючи створювати модальні вікна, каруселі зображень, обробляти кліки та інші події

користувача, а також динамічно змінювати контент без перезавантаження сторінки [8].

Git

Обґрунтування: Git – це розподілена система контролю версій, яка дозволяє відстежувати зміни в коді, працювати над проектом у команді та повертатися до попередніх версій коду. Розподілена архітектура дозволяє кожному розробнику мати повну копію репозиторію, що забезпечує швидкий доступ до історії змін та можливість працювати офлайн. Git підтримує нелінійну розробку з гілками та злиттями, що сприяє паралельній роботі над різними функціями. Git використовує ефективні алгоритми стиснення та зберігання даних, що робить його швидким та економічним у використанні дискового простору.

Використання: Git використовується для управління версіями коду проекту, дозволяючи розробникам відстежувати кожну зміну в кодовій базі через коміти. Кожен розробник може створювати гілки для розробки нових функцій або виправлення помилок, що дозволяє паралельну роботу без конфліктів. За допомогою Git можна інтегрувати зміни з різних гілок через злиття, забезпечуючи цілісність коду. Крім того, система дозволяє відстежувати авторів змін, переглядати історію проекту, а також повертатися до попередніх версій коду в разі необхідності.

Висновок

Прагнення створити сучасний, швидкий та зручний веб-каталог вантажів будівництва визначило вибір технологічного стеку. Використання Node.js забезпечило високу продуктивність та масштабованість серверної частини, а Express.js спростив створення API та маршрутизацію. SQLite, як вбудована база даних, забезпечила легкість розгортання та ефективне зберігання даних [7]. Svelte, як фреймворк для фронтенду, дозволив створити інтерактивний та швидкий інтерфейс користувача. Fetch API та Classnames забезпечили зручну взаємодію з сервером та динамічне управління стилями.

Git, як система контролю версій, гарантував надійність зберігання коду та ефективну спільну роботу над проектом.

7.2 Модулі і алгоритми

Система поділяється на два модулі, а саме сервер та веб-застосунок. Сервер зберігає та надає доступ до даних через HTTP запити.

7.2.1 Скрипти серверної частини

Для своєї роботи сервер використовує наступні скрипти:

1. Скрипт підключення до бази даних.

Цей скрипт відповідає за підключення до бази даних. Оскільки SQLite база даних представляє з себе файл database.sqlite, то цей скрипт намагається підключитися до цього файлу. Якщо цей файл відсутній у відповідній директорії, то буде створено новий файл database.sqlite з новою базою даних (див Лістинг 1).

Лістинг *1 Підключення до бази даних*

```
// Шлях до файлу бази даних
const dbPath = path.resolve(__dirname, 'database.sqlite');
// Перевірка існування файлу бази даних
const dbExists = fs.existsSync(dbPath);

// Створення чи відкриття бази даних
export const db = new sqlite3.Database(dbPath, (err) => {
      if (err) {
            console.error('Помилка відкриття БД: ' + err.message);
      } else {
            console.log('З’єднання встановлено.');
            if (!dbExists) {
                  createTables(db)
                        .then(() => {
                              console.log('Таблиці створені');
                              importCatalog(db);
                        })
                        .catch((err) => {
                              console.error('Помилка створення таблиць:',
err.message);
                        });
            }
      }
});
```

2. Додавання таблиць до бази даних.

Якщо база даних була відсутня, то після створення нового файлу створюються таблиці для бази даних. За це відповідає функція createTables (див Лістинг 2).

Лістинг *2 Створення таблиць в базі даних*

```
export async function createTables(db) {
  await db.exec(`
    CREATE TABLE IF NOT EXISTS load_category (
      id INTEGER PRIMARY KEY AUTOINCREMENT,
      name TEXT
    );
    CREATE TABLE IF NOT EXISTS load_type (
      id INTEGER PRIMARY KEY AUTOINCREMENT,
      name TEXT
    );
    CREATE TABLE IF NOT EXISTS catalog (
      id INTEGER PRIMARY KEY AUTOINCREMENT,
      load_category_id INTEGER,
      load_type_id INTEGER,
      images TEXT,
      industry_images TEXT,
      special_remarks TEXT,
      supporting_info TEXT,
      FOREIGN KEY (load_category_id) REFERENCES
load_category(id),
      FOREIGN KEY (load_type_id) REFERENCES load_type(id)
    );
  `);
}
```

3. Заповнення бази даних.

Після створення таблиць виконується додавання даних про каталог. Всі дані зберігаються у файлі catalog.csv. Функція importCatalog виконує построчне зчитування цього файлу та запис даних у відповідні таблиці (див Лістинг 3).

Лістинг *3 Заповнення бази даних*

```
async function importCatalog(db) {
  const filePath = path.resolve(__dirname, 'catalog.csv');

  const loadCategories = new Map();
  const loadTypes = new Map();

  fs.createReadStream(filePath)
    .pipe(csv())
    .on('data', async (row) => {
```

```
      const { id, Load_Category_name, Load_Type_name, example_1,
example_2, industry_reference_1, industry_reference_2,
special_remarks, supporting_info } = row;

      if (!loadCategories.has(Load_Category_name)) {
        const result = await db.run(`INSERT INTO load_category (name)
VALUES (?)`, [Load_Category_name]);
        loadCategories.set(Load_Category_name, result.lastID);
      }
      const loadCategoryId = loadCategories.get(Load_Category_name);

      if (!loadTypes.has(Load_Type_name)) {
        const result = await db.run(`INSERT INTO load_type (name)
VALUES (?)`, [Load_Type_name]);
        loadTypes.set(Load_Type_name, result.lastID);
      }
      const loadTypeId = loadTypes.get(Load_Type_name);

      const images = [example_1, example_2].filter(Boolean).join(',');
      const industryImages = [industry_reference_1,
industry_reference_2].filter(Boolean).join(',');

      await db.run(
        `INSERT INTO catalog (id, load_category_id, load_type_id,
images, industry_images, special_remarks, supporting_info) VALUES (?,
?, ?, ?, ?, ?, ?)`,
        [id, loadCategoryId, loadTypeId, images || null,
industryImages || null, special_remarks || null, supporting_info ||
null]
      );
    })
    .on('end', () => {
      console.log('CSV file successfully processed');
    });
}
```

4. Отримання каталогу.

Ендпоінт: GET /api/catalog

Цей ендпоінт використовується для отримання списку елементів каталогу з можливістю фільтрації, сортування та пагінації (див Лістинг 4).

Параметри запиту:

page: Номер сторінки (за замовчуванням 1).

limit: Кількість елементів на сторінку (за замовчуванням 10).

sort: Поле для сортування (id, load_type, load_category).

order: Порядок сортування (asc, desc).

search: Пошуковий запит для часткового співпадіння.

load_category_ids: Список ID категорій вантажів для фільтрації.

load_type_ids: Список ID типів вантажів для фільтрації.

Відповідь:

total: Загальна кількість елементів.

items: Масив елементів каталогу, які відповідають критеріям запиту

Лістинг *4 Отримання каталогу*

```
router.get('/', async (req, res) => {
  try {
    const { page = 1, limit = 10, sort = 'id', order =
'asc', search = '', load_category_ids = '', load_type_ids =
'' } = req.query;
    const offset = (page - 1) * limit;

    const filters = [];
    if (load_category_ids) {
      filters.push(`catalog.load_category_id IN
(${load_category_ids.split(',').map(id =>
`'${id}'`).join(', ')})`);
    }
    if (load_type_ids) {
      filters.push(`catalog.load_type_id IN
(${load_type_ids.split(',').map(id => `'${id}'`).join(',
')})`);
    }
    if (search) {
      filters.push(`(load_type.name LIKE '%${search}%' OR
load_category.name LIKE '%${search}%' OR
catalog.special_remarks LIKE '%${search}%' OR
catalog.supporting_info LIKE '%${search}%')`);
    }

    const whereClause = filters.length ? `WHERE
${filters.join(' AND ')}` : '';
    const orderByClause = `ORDER BY ${sort === 'load_type'
? 'load_type.name' : sort === 'load_category' ?
'load_category.name' : 'catalog.id'} ${order}`;

    const total = await db.get(`SELECT COUNT(*) as count
FROM catalog JOIN load_category ON catalog.load_category_id
= load_category.id JOIN load_type ON catalog.load_type_id =
load_type.id ${whereClause}`);
    const items = await db.all(`SELECT catalog.*,
load_category.name as load_category, load_type.name as
load_type FROM catalog JOIN load_category ON
catalog.load_category_id = load_category.id JOIN load_type
```

```
ON catalog.load_type_id = load_type.id ${whereClause}
${orderByClause} LIMIT ${limit} OFFSET ${offset}`);

    res.json({
      total: total.count,
      items
    });
  } catch (error) {
    res.status(500).json({ error: 'Failed to fetch catalog'
});
  }
});
```

5. Отримання інформації про конкретний елемент каталогу

Ендпоінт: GET /api/catalog/:id

Цей ендпоінт використовується для отримання детальної інформації про конкретний елемент каталогу за його ID.

Параметри запиту:

id: ID елемента каталогу.

Відповідь: Об’єкт з інформацією про елемент каталогу, включаючи його категорію, тип, зображення та додаткові описи (див Лістинг 5).

Лістинг 5 Отримання інформації про конкретний елемент каталогу

```
router.get('/:id', async (req, res) => {
  try {
    const id = req.params.id;
    const item = await db.get(`
      SELECT catalog.*, load_category.name as
load_category, load_type.name as load_type
      FROM catalog
      JOIN load_category ON catalog.load_category_id =
load_category.id
      JOIN load_type ON catalog.load_type_id = load_type.id
      WHERE catalog.id = ?
    `, id);

    if (!item) {
      res.status(404).json({ error: 'Item not found' });
      return;
    }

    item.images = item.images ? item.images.split(',') :
[];
```

```
    item.industry_images = item.industry_images ?
item.industry_images.split(',') : [];

    res.json(item);
  } catch (error) {
    res.status(500).json({ error: 'Failed to fetch item'
});
  }
});
```

6. Отримання списку типів вантажу

Ендпоінт: GET /api/load-types

Цей ендпоінт використовується для отримання списку всіх доступних типів вантажу, відсортованих за алфавітом.

Відповідь: Масив об’єктів з типами вантажу, кожен з яких містить id і name (див Лістинг 6).

Лістинг *6 Отримання списку типів вантажу*

```
router.get('/load-types', async (req, res) => {
  try {
    const loadTypes = await db.all(`SELECT * FROM load_type ORDER BY
name ASC`);
    res.json(loadTypes);
  } catch (error) {
    res.status(500).json({ error: 'Failed to fetch load types' });
  }
});
```

7. Отримання списку категорій вантажу

Ендпоінт: GET /api/load-categories

Цей ендпоінт використовується для отримання списку всіх доступних категорій вантажу, відсортованих за алфавітом.

Відповідь: Масив об’єктів з категоріями вантажу, кожен з яких містить id і name (див Лістинг 7).

Лістинг *7 Отримання списку категорій вантажу*

```
router.get('/load-categories', async (req, res) => {
  try {
    const loadCategories = await db.all(`SELECT * FROM load_category
ORDER BY name ASC`);
    res.json(loadCategories);
  } catch (error) {
    res.status(500).json({ error: 'Failed to fetch load categories'
});
  }
});
```

7.2.2 Скрипти для веб-застосунку

Основними скриптами для застосунку виступають компоненти Svelte, який складається зі скрипта (JavaScript коду, який оброблює всю основну логіку), HTML коду та CSS стилів. Також до основної логіки входять компоненти маршрутизації застосунку та HTML файл.

1. HTML file

HTML файл виступає вхідною точкою застосунку. Svelte додає до цього файлу весь код, що необхідний для роботи веб застосунку (див Лістинг 8).

Лістинг *8 app.html*

```
<!DOCTYPE html>
<html lang="en">
	<head>
		<meta charset="utf-8" />
		<link rel="icon" href="%sveltekit.assets%/favicon.png" />
		<meta name="viewport" content="width=device-width" />
		%sveltekit.head%
	</head>
	<body data-sveltekit-preload-data="hover">
		<div style="display: contents">%sveltekit.body%</div>
	</body>
</html>
```

2. Компонент для управління макетом додатку

Для створення макету сторінки, щоб не дублювати код для відображення елементів, використовується файл +layout.svelte. Цей файл містить основну структуру веб застосунку, а саме заголовок, основна частина та футер сторінки (див Лістинг 9).

Лістинг *9 layout.svelte*

```
<script>
	import Header from '$lib/components/landing/Header.svelte';
	import Footer from '$lib/components/landing/Footer.svelte';
	import './styles.css';
</script>
<div class="app">
	<Header />
	<main>
		<slot />
	</main>
```

```
      <Footer/>
</div>
```

Компоненти можуть мати дочірній зміст так само, як і елементи. Вміст відкривається у дочірньому компоненті за допомогою елемента <slot>, який може містити запасний вміст, який відображається, якщо не надано дочірніх компонентів.

Цей макет використовується для відображення контенту на сторінці веб застосунку при маршрутизації. Окремі сторінки веб застосунку оброблюються цим макетом саме як <slot> елемент, що дозволяє правильно розташувати контент на сторінці, та не прописувати їх окремо для кожної сторінки веб-застосунку.

3. Компонент каталогу

Компонент каталогу містить всю необхідні скрипти та логіку для отримання, обробки, зберігання та відображення каталогу. Для своє роботи компонент використовує наступні дані:

let catalog = [] - зберігає елементи каталогу

let loadTypes = [] - зберігає список типів завантаження

let loadCategories = [] - зберігає список категорії завантаження

let page = 1 - поточна сторінка пагінації

let limit = 10 - кількість елементів на сторінку пагінації

let sortOrder = 'asc' - порядок сортування елементів

let selectedLoadTypes = writable([]) - список типів завантаження, що були обрані користувачем

let selectedLoadCategories = writable([]) - список категорій завантаження, що були обрані користувачем

let search = "" - текст за яким користувач шукає дані у каталозі

let totalPages = 1 - загальна кількість сторінок пагінації

let showLoadTypes = false - чи відкритий фільтр для вибору типів завантаження

let showLoadCategories = false - чи відкритий фільтр для вибору категорій завантаження

let isLoading = true — виконужться завантаження даних

3.1 Завантаження даних каталогу

Для завантаження каталогу використовується функція fetchCatalog, що оброблює всі параметри пошуку та робить запит до сервера для отримання даних (див Лістинг 10).

Лістинг *10 fetchCatalog*

```
const fetchCatalog = async () => {
    isLoading = true;
    const loadTypeIds: string =
$selectedLoadTypes.join(',');
    const loadCategoryIds: string =
$selectedLoadCategories.join(',');
    const params = new URLSearchParams({
        page,
        limit,
        sortBy,
        sortOrder,
        load_type_ids: loadTypeIds,
        load_category_ids: loadCategoryIds,
        search,
    });
    const response = await
fetch(`${import.meta.env.VITE_API_URL}/catalog?${params.toS
tring()}`);
    const { data } = await response.json();
    catalog = data.data.map(item => ({
        ...item,
        images: JSON.parse(item.images),
        industry_images: JSON.parse(item.industry_images)
    }));
    totalPages = Math.ceil(data.total / limit);
    isLoading = false;
};
```

3.2 Завантаження даних про фільтри

Для завантаження фільтрів, а саме loadTypes та loadCategories використовуються відповідні функції, які виконуються запит до серверу (див Лістинг 11).

Лістинг *11 fetchLoadTypes та fetchLoadCategories*

```
const fetchLoadTypes = async () => {
    const response = await
fetch(`${import.meta.env.VITE_API_URL}/load_data/load_types
`);
    const data = await response.json();
    loadTypes = data.data;
};
const fetchLoadCategories = async () => {
    const response = await
fetch(`${import.meta.env.VITE_API_URL}/load_data/load_categ
ories`);
    const data = await response.json();
    loadCategories = data.data;
};
```

3.3 Відображення каталогу

Відображення сторінки каталогу поділяється на дві секції: фільтри та таблиця з елементами каталогу. Svelte надає можливість писати HTML код поряд зі скриптами компоненту, що дозволяє додати JavaScript змінні прямо до HTML коду (див Лістинг 12).

Лістинг *12 рендер компоненту каталог*

```
<div class="container">
    <div class="sidebar">
        <!--HTML код для відображення фільтрів->
    </div>
    <div class="content">
        <!--HTML код для відображення таблиці->
    </div>
</div>
```

4. Компонент елементу каталогу

Сторінка елементу каталогу поділяється на два файли. Перший файл +page.server.ts виконує запит на отримання інформації про елемент. Це зроблено для того, щоб обробка цього запиту відбувалась на стороні веб серверу, а не на клієнтській стороні. Це зменшує затримку перед відображенням контенту для користувача. Запит на стороні веб серверу відбувається за допомогою функції load, яка після завантаження даних передає їх до наступного компоненту (див Лістинг 13).

Лістинг *13 файл +page.server.ts*

```
export async function load({ params }) {
    const response = await
fetch(`${import.meta.env.VITE_API_URL}/catalog/${params.ite
m}`);
    if (!response.ok) {
        return {
            status: response.status,
            error: new Error('Could not load the item')
        };
    }
    const { data } = await response.json();
    return data;
}
```

Далі файл +page.svelte приймає дані від веб серверу (див Лістинг 14), оброблює та відображає їх користувачу (див Лістинг 15).

Лістинг *14 скрипт файлу +page.svelte*

```
<script lang="ts">
    const { id, load_category, load_type, images,
industry_images, special_remarks, supporting_info } = data
|| {};
    const renderImages = images ?
JSON.parse(images).filter(item => item !== 'null').map(item
=> ({src: item, alt: 'image'})) : null;
    const renderIndustryImages = industry_images ?
JSON.parse(industry_images).filter(item => item !==
'null').map(item => ({src: item, alt: 'image'})) : null;
</script>
```

Лістинг *15 рендер файлу +page.svelte*

```
{#if id}
    <article>
        <!--HTML код для відображення елементу каталогу->
    </article>
{:else}
    <article>
        <h1>Wrong product id</h1>
    </article>
{/if}
```

7.3 Структури даних

В розробленій базі даних для веб-каталогу вантажів будівництва використовується три взаємопов'язані таблиці (сутності): loadType,

loadCategory та catalog. Ця структура дозволяє забезпечити ефективну організацію та зберігання даних про типи вантажів, їх категорії та конкретні позиції в каталозі. Взаємозв'язки між цими таблицями дозволяють легко отримувати необхідну інформацію, фільтрувати та сортувати дані для зручного пошуку та перегляду вантажів користувачами (див. Рисунок 10).

Розглянемо сутності бази даних:

loadType - описує різні типи вантажів, що використовуються в будівництві (див. Табл. 1). Кожен запис у цій таблиці представляє унікальний тип вантажу, такий як "цегла", "пісок", "щебінь" тощо. Атрибут id служить унікальним ідентифікатором для кожного типу, а атрибут name зберігає назву цього типу.

loadCategory: Ця сутність класифікує вантажі за різними категоріями (див. Табл. 3). Кожен запис у цій таблиці представляє окрему категорію, таку як "сипучі матеріали", "стінові матеріали", "покрівельні матеріали" тощо. Атрибут id є унікальним ідентифікатором категорії, а name зберігає її назву.

catalog: Ця сутність є основною таблицею каталогу і містить інформацію про кожен окремий вантаж (див. Табл. 2). Кожен запис у цій таблиці представляє один вантаж і містить інформацію про його тип (load_type_id), категорію (Load_category_id), зображення (images, industry_images), спеціальні примітки (special_remarks) та додаткову інформацію (supporting_info). Атрибут id служить унікальним ідентифікатором для кожного запису в каталозі.

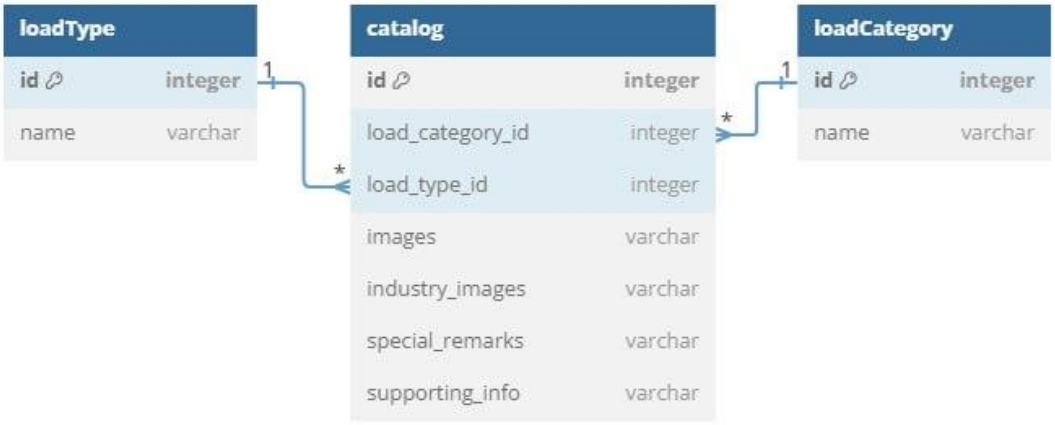

Рисунок 10 – *ER діаграма бази даних*

Таблиця 1 – *Опис сутності* ***loadType***

Сутність loadType		
Назва поля	**Опис**	**Тип**
id	Первинний ключ, який унікально ідентифікує кожен тип вантажу	integer
name	Назва типу вантажу	varchar

Таблиця 2 – *Опис сутності* ***Groups***

Сутність catalog		
Назва поля	**Опис**	**Тип**
id	Первинний ключ, який унікально ідентифікує елемент каталогу	integer

Load_category_id	Зовнішній ключ, який посилається на id з таблиці loadCategory	integer
Load_type_id	Зовнішній ключ, який посилається на id з таблиці loadType	integer
images	Масив посилань на зображення, пов'язані з даним елементом каталогу	varchar
Indystry_images	Масив посилань на промислові зображення, пов'язані з даним елементом каталогу	varchar
Special_remarks	Спеціальні зауваження щодо даного елемента каталогу	varchar
Supporting_info	Додаткова інформація щодо даного елемента каталогу	varchar

Таблиця 3 *Опис сутності* ***loadCategory***

Сутність loadCategory		
Назва поля	**Опис**	**Тип**
id	Первинний ключ, який унікально ідентифікує	integer

	кожну категорію вантажу	
name	Назва категорій вантажу	varchar

Відношення між таблицями

Взаємозв'язки між таблицями loadType, loadCategory та catalog у базі даних веб-каталогу вантажів будівництва побудовані за принципом "один до багатьох". Це означає, що кожен тип вантажу (loadType) може бути пов'язаний з кількома категоріями вантажів (loadCategory), а кожна категорія, в свою чергу, може містити безліч конкретних позицій вантажів у каталозі (catalog). Така структура забезпечує гнучкість та масштабованість бази даних, дозволяючи легко додавати нові типи, категорії та позиції вантажів, зберігаючи при цьому логічну цілісність та зв'язність даних. Крім того, цей підхід спрощує виконання запитів до бази даних, дозволяючи швидко та ефективно отримувати інформацію про вантажі за різними критеріями, такими як тип, категорія, ціна, характеристики тощо.

7.4 Проект інтерфейсу

Під час створення інтерфейсу веб-каталогу вантажів будівництва, були прийняті ключові дизайнерські рішення, спрямовані на забезпечення інтуїтивно зрозумілої навігації, привабливого візуального оформлення та максимальної зручності користування для відвідувачів. Ці рішення враховують сучасні тенденції веб-дизайну, ергономіку взаємодії з користувачем та специфіку тематики каталогу, що дозволяє створити продукт, який не лише ефективно виконує свої функції, але й залишає позитивне враження у користувачів.

Колірна гама

1. Темний фон (#151516)

Обґрунтування: Вибір темного фону (#151516) для веб-каталогу вантажів будівництва обумовлений його здатністю знижувати навантаження на очі користувачів, особливо при тривалому перегляді інформації. Це особливо актуально в умовах низької освітленості або при використанні екранів з високою яскравістю. Темний фон створює більш комфортні умови для читання та сприйняття інформації, що сприяє підвищенню задоволеності користувачів та збільшенню часу, проведеного на сайті.

2. Колір шрифтів (whitesmoke)

Обґрунтування: Вибір кольору whitesmoke для шрифтів на темному фоні (#151516) веб-каталогу вантажів будівництва забезпечує оптимальний контраст, що значно покращує читабельність тексту. Таке поєднання кольорів сприяє кращому сприйняттю інформації, зменшує навантаження на очі та робить використання веб-каталогу більш комфортним для користувачів.

3. Колір шрифтів (whitesmoke)

Обґрунтування: Використання кольору whitesmoke для основного тексту та трохи світлішого відтінку для заголовків на темному фоні (#151516) дозволяє створити візуальну ієрархію та покращити навігацію по веб-каталогу вантажів будівництва. Заголовки, виділені світлішим кольором, стають більш помітними та привертають увагу користувачів, допомагаючи їм швидко орієнтуватися в контенті та знаходити потрібну інформацію.

Шрифти

Arial, -apple-system, BlinkMacSystemFont, 'Segoe UI', Roboto, Oxygen, Ubuntu, Cantarell, 'Open Sans', 'Helvetica Neue', sans-serif.

Обґрунтування: Використання різноманітних системних шрифтів забезпечує гарний вигляд на різних платформах та пристроях. Вибрані шрифти є сучасними, легко читаними та забезпечують хороший користувацький досвід.

Розміри шрифтів:

- Заголовки (header): великий розмір (наприклад, 24px) для кращого виділення.
- Основний текст: середній розмір (наприклад, 16px) для зручного читання.

Обґрунтування: Вибір розміру шрифтів базується на рекомендаціях щодо читабельності та зручності сприйняття інформації.

Розміри, форма і розташування елементів управління

1. Розташування фільтрів

Фільтри розташовані зліва від основного контенту, як це часто робиться в інтернет-магазинах.

Обґрунтування: Такий підхід дозволяє користувачам легко знаходити і застосовувати фільтри без потреби прокручувати сторінку вниз, що покращує користувацький досвід.

2. Розміри і форма кнопок

Кнопки мають стандартний розмір (наприклад, 40px у висоту і 100px у ширину) і округлі краї.

Обґрунтування: Кнопки стандартного розміру з округлими краями є зрозумілими та зручними для користувачів, що полегшує взаємодію з інтерфейсом.

3. Інтервали і відступи

Відступи між елементами складають не менше 16px.

Обґрунтування: Достатні відступи забезпечують візуальну чистоту і роблять інтерфейс більш організованим, що полегшує сприйняття інформації.

4. Іконки та зображення

Використання SVG-іконок та зображень в каталозі.

Обґрунтування: SVG-іконки мають високу якість при будь-якому розмірі і масштабуванні, що забезпечує чіткість іконок на екранах з високою роздільною здатністю. Використання якісних зображень товарів дозволяє користувачам отримати краще уявлення про продукцію.

5. Загальна структура сторінок

Сторінка каталогу.

Заголовок сторінки розташований зверху, фільтри зліва, а основний контент з товарами - праворуч.

Обґрунтування: Така структура забезпечує логічне розташування елементів і полегшує навігацію.

Сторінка окремого елемента.

Основна інформація про елемент розташована у центрі, додаткова інформація (зображення, спеціальні примітки) розміщені нижче.

Обґрунтування: Це дозволяє користувачам швидко ознайомитися з основною інформацією, а потім детальніше переглянути додаткові деталі.

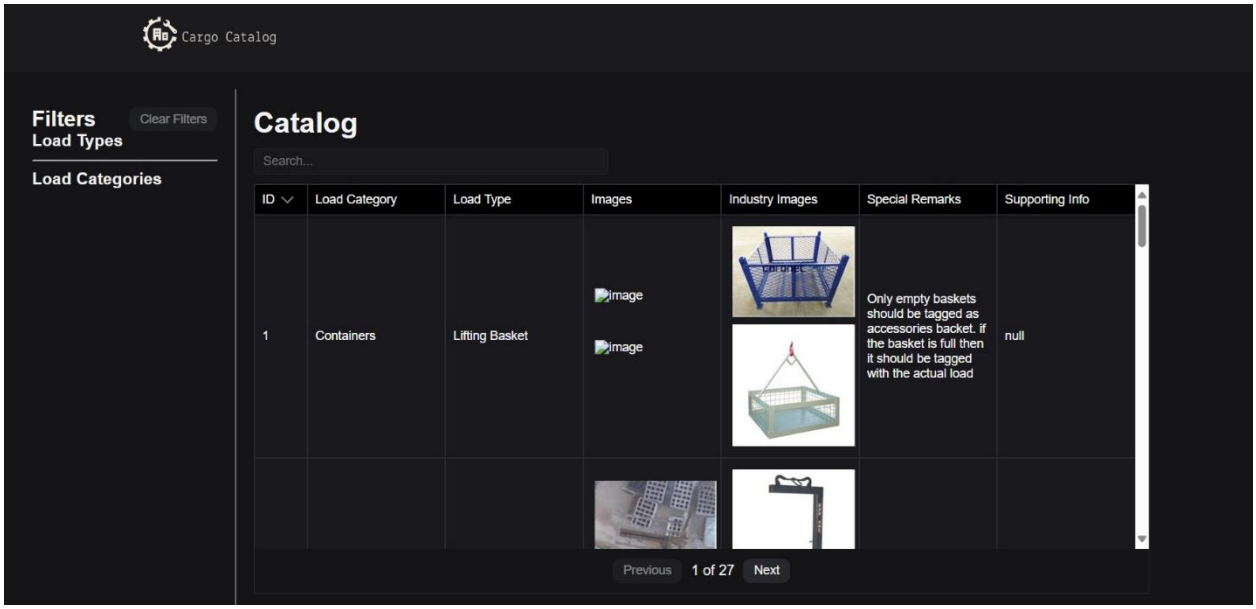

Рисунок 11 –*Головна сторінка веб-каталогу*

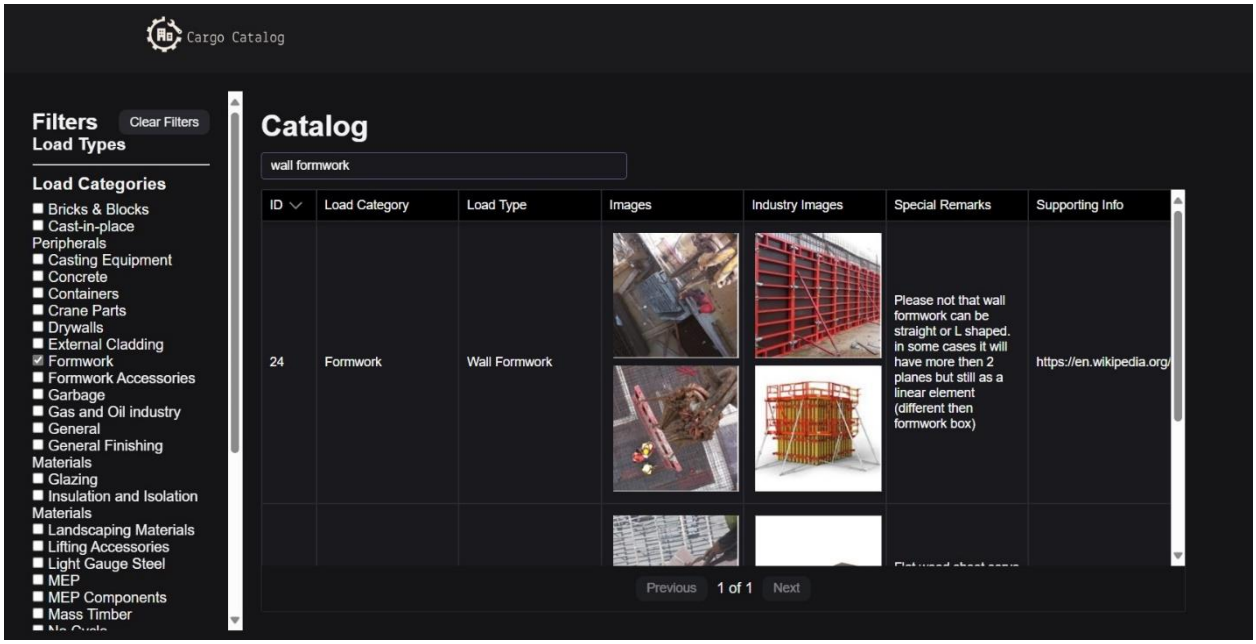

Рисунок 12 – *Головна сторінка веб-каталогу з застосованим пошуком*

Прийняті дизайнерські рішення були спрямовані на створення інтуїтивно зрозумілого та зручного інтерфейсу веб-каталогу вантажів будівництва, який дозволить користувачам легко знаходити потрібну інформацію, швидко орієнтуватися в каталозі та здійснювати необхідні дії без зайвих зусиль (див. Рис 11). Це сприятиме підвищенню ефективності роботи з каталогом, задоволеності користувачів та загальному позитивному враженню від використання продукту (див. Рис 12).

8. РЕАЛІЗАЦІЯ І ТЕСТУВАННЯ

Фізичні характеристики поточної версії системи

1. Об’єм коду Frontend
   - Загальний об’єм коду: 200 кілобайт.
   - Кількість рядків коду: 2000 рядків.
2. Об’єм коду Backend
   - Загальний об’єм: 150 кілобайт.
   - Кількість рядків коду: 1000 рядків.

Кількість модулів, форм, екранів

1. Frontend
   - Модулі: 10.
   - Форми: 2 (Форма фільтрації, форма пошуку).
   - Екрани: 2 основні (Сторінка каталогу, сторінка окремого елемента).
2. Backend

- Модулі: 7.

Кількість і об'єм програмних компонентів

1. Frontend
   - Компоненти Svelte: 9, загальним об'ємом 200 кілобайт.
2. Backend
   - Файли з логікою маршрутизації, контролерів, моделей: 7, загальним об'ємом 150 кілобайт.

Фактична швидкодія і витрати оперативної пам'яті

1. Frontend
   - Час завантаження сторінки каталогу при повільному інтернет-з'єднанні (Slow 3G): 5-10 секунд.
   - Витрати оперативної пам'яті під час роботи (при перегляді каталогу): 50-100 МБ.
2. Backend

- Час відповіді на запит до API (отримання каталогу): 200-300 мс.
- Витрати оперативної пам'яті під час роботи: 30-50 МБ.

Фактична кількість користувачів

На даний момент система використовується тільки розробниками для тестування та налагодження. Після розгортанні, система буде готова до використання широкою аудиторією

Тестування

Тестування проводилося як автоматизовано, так і вручну. Автоматизовані тести охоплювали основні функціональні можливості бекенда, включаючи перевірку коректності роботи API. Тестування фронтенду здійснювалося вручну з використанням Google Chrome DevTools для перевірки доступності та швидкодії при різних умовах інтернет-з'єднання.

1. Тестування бекенду

Мета: Перевірити коректність роботи API, включаючи обробку запитів, коректність даних, роботу з базою даних.

Метод: Використання Google Chrome DevTools для симуляції повільного інтернет-з'єднання.

Результат: Завантаження каталогу відбувається з певною затримкою при повільному з'єднанні, але всі дані відображаються правильно. Фільтрація, сортування та пошук працюють належним чином. Всі основні функції доступні і не викликають помилок при повільному інтернет-з’єднанні. Система відповідає вимогам щодо доступності.

2. Тестування фронтенду

Для тестування фронтенду було проведено ручні тести на доступність та швидкодію.

Відкриття веб-додатка у браузері Google Chrome:

- Завантажте веб-додаток у браузері.
- Відкрийте DevTools (клавіша F12).

Налаштування симуляції повільного веб-з’єднання:

- Перейдіть до вкладки «Network».
- Оберіть «Slow 3G».

3. Перевірка функціональності:

- Оновіть сторінку та спостерігайте за її завантаженням.
- Перевірте, чи завантажується таблиця каталогу та чи можна переглянути всі елементи.
- Переконайтеся, що фільтри, сортування та пошук працюють належним чином.
- Перевірте завантаження зображень та їх відображення.

Результати тестування:

4. Час завантаження сторінки: 5-10 секунд при симуляції повільного інтернет-з’єднання.

5. Відображення даних: Усі дані завантажуються та відображаються правильно.

6. Функціональність: Фільтрація, сортування та пошук працюють належним чином.

7. Зображення: Усі зображення завантажуються, хоча і з затримкою при повільному інтернет-з’єднанні. Всі зображення мають alt-теги.

8. Доступність: Всі елементи інтерфейсу доступні для взаємодії, відповідають вимогам щодо доступності.

ВИСНОВКИ

1. Аналіз існуючих рішень у сфері веб-розробки підтвердив актуальність створення спеціалізованого веб-каталогу вантажів будівництва. Такий каталог полегшує пошук та вибір необхідних матеріалів, оптимізує процес звітності для замовника послуг моніторингу за будівництвом.
2. Проведено порівняльний аналіз різних технологій та фреймворків для веб-розробки, таких як React, Vue.js та Svelte. Визначено переваги та недоліки кожного з них, що дозволило обрати оптимальний стек технологій для реалізації проекту.
3. Спроектовано архітектуру веб-каталогу, визначено основні компоненти та їх взаємодію. Розроблено функціональні вимоги до системи, враховуючи потреби адміністратора та користувачів.
4. Створено веб-каталог вантажів будівництва з використанням обраних технологій (Node.js, Express.js, SQLite, Svelte). Реалізовано основний функціонал, включаючи додавання, редагування та видалення вантажів та категорій (для адміністратора), а також перегляд каталогу, пошук та фільтрацію вантажів (для користувачів).
5. У процесі розробки застосовано сучасні підходи до веб-розробки, такі як компонентний підхід (Svelte), використання REST API для взаємодії між фронтендом та бекендом, а також оптимізація продуктивності та зручності використання.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Nixon R. Learning PHP, MySQL & JavaScript. 5-те вид. O'Reilly Media, 2018. 832 с.
2. Stumpf M. HTML5 – Pathfinder for Infotainment Innovations. *ATZelektronik worldwide*. 2013. Т. 8, № 6. С. 32–35. URL: https://doi.org/10.1365/s38314-013-0209-3 (дата звернення: 23.04.2024).
3. Shklar L., Rosen R. Web Application Architecture: Principles, Protocols and Practices. Wiley & Sons, Incorporated, John, 2011. 440 с.
4. Homepage - Versatile. *Versatile*. URL: https://www.versatile.ai/ (дата звернення: 23.04.2024).
5. Bhardwaz S., Godha R. Svelte.js: The Most Loved Framework Today. *2023 2nd International Conference for Innovation in Technology (INOCON)*, м. Bangalore, India, 3–5 берез. 2023 р. 2023. URL: https://doi.org/10.1109/inocon57975.2023.10101104 (дата звернення: 25.04.2024).
6. Prediger R., Winzinger R. Hello, Node.js. *Node.js*. München, 2015. С. 1–33. URL: https://doi.org/10.3139/9783446437586.001 (дата звернення: 25.04.2024).
7. SQLite / K. P. Gaffney та ін. *Proceedings of the VLDB Endowment*. 2022. Т. 15, № 12. С. 3535–3547. URL: https://doi.org/10.14778/3554821.3554842 (дата звернення: 25.04.2024).
8. Mayer G. E. T., Awesomeness J. JavaScript: JavaScript Awesomeness Book. CreateSpace Independent Publishing Platform, 2017. 68 с.

Декларація

академічної доброчесності

здобувача ступеня вищої освіти ЗНУ

Я, Прокопович Антон Юрійович, студент 4 курсу, форми навчання денної, Інженерного навчально-наукового інституту, спеціальність 121 Інженерія програмного забезпечення, адреса електронної пошти ipz20bz-214@stu.zsea.edu.ua, — підтверджую, що написана мною кваліфікаційна робота на тему «**Розробка веб-каталогу для систем маркування вантажів будівництва**» відповідає вимогам академічної доброчесності та не містить порушень, що визначені у ст.42 Закону України «Про освіту», зі змістом яких ознайомлений.

- заявляю, що надана мною для перевірки електронна версія роботи є ідентичною її друкованій версії;

- згоден на перевірку моєї роботи на відповідність критеріям академічної доброчесності у будь-який спосіб, у тому числі за допомогою інтернет-системи, а також на архівування моєї роботи в базі даних цієї системи.

Дата 14.06.2024 _____ Прокопович Антон Юрійович
(підпис) (прізвище та ініціали) (студент)

Дата 15.06.2024 _____ Полякова Наталія Петрівна
(підпис) (прізвище та ініціали) керівник)