МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«До захисту допущено»
В.о. завідувача кафедри
Оксана ШОВКОПЛЯС
_____
(підпис)
16 грудня 2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня магістр

зі спеціальності 122 - «Комп'ютерні науки»,
освітньо-професійної програми «Інформатика»
на тему: «Інформаційна технологія ергономічного проєктування вебдодатку»
здобувача групи ІН.м – 34 Кононенка Олександра Олеговича

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

Олександр
КОНОНЕНКО
_____
(підпис)

Керівник Наталія БАРЧЕНКО
к.т.н., доцент
_____
(підпис)

Суми – 2024

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«Затверджую»
В.о. завідувача кафедри
_____ Оксана ШОВКОПЛЯС
(підпис)

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня магістра

зі спеціальності 122 Комп'ютерні науки, освітньо-професійної програми «Інформатика»
здобувача групи ІН.м-34 Кононенка Олександра Олеговича

1. Тема роботи: «Інформаційна технологія ергономічного проєктування вебдодатку» затверджую наказом по СумДУ від *«03» грудня 2024 року № №1257-VI*

2. Термін здачі здобувачем кваліфікаційної роботи *до 01 грудня 2024 року*

3. Вхідні дані до кваліфікаційної роботи

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити)
1) Аналіз предметної області та постановка завдань дослідження.
2) Огляд методів проєктування інформаційної технології 3) Розробка інформаційної технології
4)Висновки

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

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

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання «14» жовтня 2024 р.

Завдання прийняв до виконання _____ (підпис) Керівник _____ (підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Термін виконання	Примітка
1	*Аналіз предметної області та постановка завдань дослідження*	31.10.2024	виконано
2	*Огляд методів проєктування інформаційної технології*	04.11.2024	виконано
3	*Розробка інформаційної технології*	15.11.2024	виконано
4	*Висновки*	20.11.2024	виконано

Здобувач вищої освіти _____ Керівник _____

(підпис) (підпис)

АНОТАЦІЯ

Записка: 54 стр., 22 рис., 1 додаток, 27 використаних джерел.

Обґрунтування актуальності теми роботи – присвячена розв'язанню важливої практичної задачі підвищення ергономічності вебдодатків шляхом розробки відповідних методів, моделей та інформаційної технології ергономічного проєктування.

Об’єкт дослідження – процес проєктування вебдодатку.

Мета роботи – розробка інформаційної технології для ергономічного проєктування вебдодатку, що забезпечує зручність використання та ефективну взаємодію користувача з інтерфейсом.

Метод дослідження – розробка прототипу вебдодатку, впровадження принципів ергономічного дизайну, а також тестування юзабіліті з використанням сучасних аналітичних інструментів.

Результати - розроблений вебдодаток з урахуванням ергономічних вимог. Реалізовано функції головного меню, адаптивний дизайн, зручну навігацію, форми введення даних та інструменти для аналітики взаємодії користувачів.

ВЕБДОДАТОК, ЮЗАБІЛІТІ, ЕРГОНОМІКА,
STRIPE, REACT.JS, NEXT.JS, БАЗИ ДАНИХ

ЗМІСТ

ВСТУП

Обґрунтування вибору теми роботи. Вибір теми зумовлений нагальною потребою у підвищенні якості та зручності вебдодатків у сучасному цифровому середовищі. З розвитком інформаційних технологій та зростанням кількості вебсервісів конкуренція між ними стає все більш жорсткою. У таких умовах успіх продукту багато в чому залежить від його зручності та ергономічності.

Актуальність. Ергономічне проектування вебдодатків - це розробка програмного забезпечення, яке відповідає сучасним вимогам зручності та ефективності використання. Його основна мета – забезпечення комфортної взаємодії користувачів із цифровим середовищем [1]. Сьогодні вебдодатки активно використовуються для різних задач: спілкування, обміну інформацією, управління бізнес-процесами, освіти тощо. Проте стрімкий розвиток інтернет-технологій та високий попит на вебсервіси призвели до розширення їх функціоналу. В умовах зростання обсягу інформації та популярності цифрових платформ ергономічність стає ключовим фактором успішності вебдодатків. Інтерфейси, розроблені без урахування принципів ергономіки, можуть знижувати продуктивність користувачів і викликати незадоволення. Тому тема створення інформаційної технології для ергономічного проектування вебдодатку є особливо актуальною.

Об’єкт дослідження. Процес проєктування вебдодатку.

Предмет дослідження. Предметом дослідження є методи та засоби ергономічного проєктування вебдодатку

Новизна. Наукова новизна роботи полягає у розробці комплексної інформаційної технології ергономічного проєктування вебдодатків, яка інтегрує сучасні принципи ергономіки з новітніми технологіями веброзробки, такими як React, Next.js, Strapi та Docker.

Структура. Випускна кваліфікаційна робота складається із вступу, трьох розділів, висновків, списку використаних джерел та додатку.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

1.1 Ергономічний дизайн вебдодатків

Ергономічний дизайн вебдодатків – це напрямок, що об'єднує технології розробки програмного забезпечення з науковими принципами зручності використання. Ергономіка спрямована на створення інтерфейсів, які забезпечують легкість взаємодії та високу ефективність роботи користувачів. В основі цього підходу лежить інтеграція методів когнітивної психології, дизайну та інформаційних технологій [2].

Сучасний розвиток цифрових платформ вимагає адаптації вебдодатків до потреб користувачів різного віку, професій і рівня цифрової грамотності. Більшість користувачів віддають перевагу платформам, що забезпечують інтуїтивно зрозумілий інтерфейс і мінімізують зусилля для досягнення бажаного результату. Основними прикладами таких систем є освітні платформи, електронна комерція, інформаційні портали [3].

Місія ергономічного проектування полягає у створенні вебдодатків, які забезпечують максимальну зручність, ефективність і задоволення користувачів під час їхньої взаємодії з інтерфейсом.

Метою ергономічного проектування є створення вебдодатків, які забезпечують ефективну, комфортну та інтуїтивну взаємодію користувачів із цифровим середовищем.

Ключовими завданнями ергономічного дизайну є:

a. мінімізація когнітивного навантаження користувача;
b. забезпечення доступності контенту для осіб із різними рівнями фізичних можливостей;
c. створення адаптивного дизайну для різних пристроїв і середовищ.

Досягнення мети ергономічного проєктування забезпечує конкурентоспроможність вебдодатків, підвищує задоволеність користувачів і сприяє успішній інтеграції цифрових рішень у бізнес-процеси.

ISO 9241 (ISO 9241, WCAG) — це міжнародний стандарт, що визначає вимоги до ергономічності взаємодії людини з комп'ютерними системами. Він акцентує увагу на: Чіткості та логічності інтерфейсу. Забезпеченні зворотного зв'язку. Зменшенні ризику помилок у роботі користувача [4].

WCAG (Web Content Accessibility Guidelines) — набір рекомендацій для забезпечення доступності вебконтенту:

1. Сприйнятливість: Контент має бути доступним для всіх користувачів, включаючи людей із порушеннями зору чи слуху.
2. Функціональність: Усі елементи інтерфейсу мають працювати через клавіатуру.
3. Зрозумілість: Інтерфейс має бути логічно структурованим і зрозумілим.
4. Стійкість: Контент повинен бути сумісним із сучасними та майбутніми технологіями [5].

1.2 Принципи ергономічного проектування

Ергономічне проєктування вебдодатків спрямоване на створення інтерфейсів, які забезпечують ефективну, зручну та безпечну взаємодію користувача з системою [6]. Кругова діаграма принципів наведена на рисунку 1.1. В сучасному світі, де кількість вебдодатків постійно зростає, важливість ергономіки стає все більш актуальною. Основні принципи ергономічного проєктування допомагають розробникам створювати продукти, які відповідають потребам та очікуванням користувачів.

Значущість ергономічного проєктування також полягає в тому, що воно сприяє підвищенню продуктивності та зниженню когнітивного навантаження на користувача. Вебдодаток, що враховує принципи ергономіки, дозволяє швидше орієнтуватися у системі, мінімізує помилки та час на виконання стандартних операцій. Таким чином, створення інтерфейсу, який відповідає ергономічним принципам, формує позитивний користувацький досвід, підвищує лояльність аудиторії та може позитивно впливати на комерційну ефективність програмних продуктів.

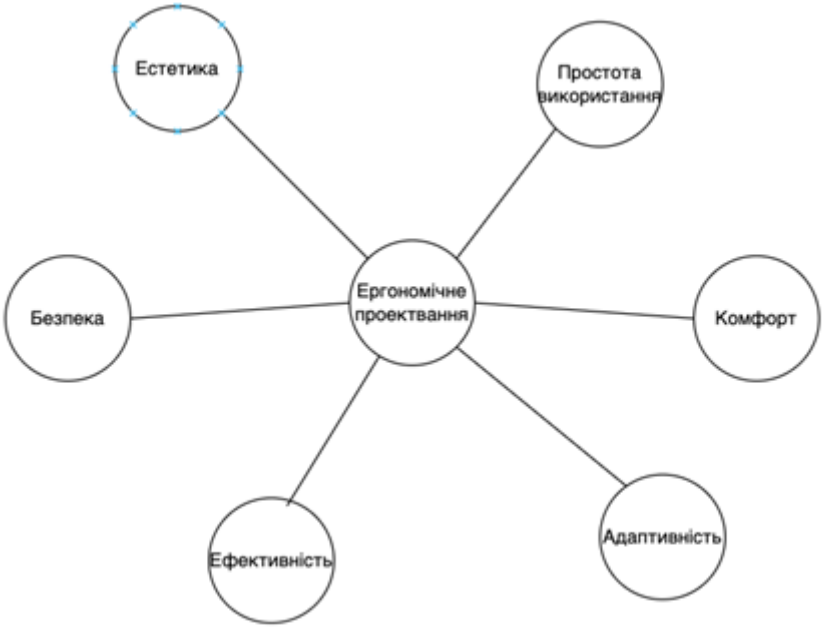

Рисунок 1.1 - Основні принципи проектування

Принцип 1: Орієнтація на користувача. Центральним елементом ергономічного проєктування є користувач. Розробка повинна базуватися на глибокому розумінні цільової аудиторії, її потреб, можливостей та обмежень. Це включає аналіз користувацьких сценаріїв, проведення опитувань та тестувань з реальними користувачами [3].

Принцип 2: Простота та інтуїтивність. Інтерфейс має відповідати критеріям простоти, бути мінімалістичним по відношенню до кількості дій для виконання завдання. Інтуїтивність має досягатися через зрозумілу навігацію та бути зрозумілою в своїй структурі для користувача [3].

Принцип 3: Консистентність. Важливо забезпечити узгодженість дизайну на всіх рівнях додатку. Це стосується використання однакових стилів, кольорів, шрифтів та поведінки елементів інтерфейсу. Консистентність допомагає користувачам швидше адаптуватися та знижує можливість виникнення помилок [3].

Принцип 4: Зворотний зв'язок. Система повинна надавати користувачеві своєчасний та зрозумілий зворотний зв'язок про результати його дій. Це

можуть бути повідомлення про успішне виконання операції, попередження про помилки чи індикатори прогресу [3].

Принцип 5: Гнучкість та адаптивність. Ергономічний дизайн передбачає можливість адаптації системи під індивідуальні потреби користувача. Це може включати налаштування інтерфейсу, вибір мови, масштабу тексту та інших параметрів [3].

Принцип 6: Попередження помилок. Дизайн повинен мінімізувати можливість виникнення помилок та спрощувати процес їх виправлення. Це досягається через чіткі інструкції, підтвердження критичних дій та можливість скасування операцій [7].

Принцип 7: Доступність. Вебдодаток повинен бути доступним для максимально широкої аудиторії, включаючи людей з обмеженими можливостями. Це передбачає відповідність стандартам доступності, таким як WCAG, та врахування особливих потреб користувачів [4].

Принцип 8: Естетичний та мінімалістичний дизайн. Візуальний дизайн має бути привабливим та не перевантаженим непотрібними елементами. Мінімалізм допомагає зосередити увагу користувача на основних функціях та покращує загальне враження від використання додатку [5].

Принцип 9: Орієнтація на завдання. Дизайн повинен підтримувати ефективне виконання користувацьких завдань. Це включає оптимізацію робочих процесів, скорочення непотрібних кроків та забезпечення швидкого доступу до основних функцій [6].

Принцип 10: Використання стандартів та найкращих практик. Дотримання встановлених стандартів та рекомендацій у сфері вебдизайну сприяє створенню якісних та надійних продуктів. Це також полегшує користувачам взаємодію з додатком, оскільки вони знайомі з загальноприйнятими підходами [8].

1.3 Огляд сучасних методів ергономічного аналізу

Ергономічний аналіз є ключовим етапом у процесі розробки

вебдодатків, що забезпечує їх зручність, ефективність та задоволеність користувачів. Детальний процес аналізу наведений на рисунку 1.2. Сучасні методи ергономічного аналізу дозволяють глибоко оцінити взаємодію користувача з системою та ідентифікувати можливі проблеми на ранніх етапах розробки. У цьому розділі розглянемо основні сучасні підходи та методи, які використовуються для ергономічного аналізу вебдодатків.

Рисунок 1.2 - Процес ергономічного аналізу

Юзабіліті-тестування. Юзабіліті-тестування є одним із найпоширеніших методів оцінки ергономіки. Воно передбачає спостереження за тим, як реальні користувачі взаємодіють з додатком, виконуючи певні завдання. Це дозволяє виявити проблеми з навігацією, інтерфейсом та загальним досвідом користувача [9].

Переваги:

a. Пряме отримання зворотного зв'язку від користувачів.
b. Виявлення неочевидних проблем у взаємодії.
c. Можливість кількісної та якісної оцінки.

Недоліки:

a. Потребує ресурсів для організації та проведення.
b. Результати можуть бути суб'єктивними та залежати від вибору учасників.

Експертна оцінка (Heuristic Evaluation). Експертна оцінка передбачає аналіз інтерфейсу фахівцями з юзабіліті та ергономіки на основі

встановлених принципів та евристик. Метод Нільсена-Моліча є одним із найбільш відомих у цій категорії.

Переваги:

a. Швидкий та економічний спосіб оцінки.
b. Дозволяє виявити більшість критичних проблем на ранніх етапах.

Недоліки:

a. Залежить від досвіду та компетентності експертів.
b. Може не враховувати специфічні потреби кінцевих користувачів [9].

Когнітивний пройом (Cognitive Walkthrough). Цей метод фокусується на аналізі кроків, які користувач повинен виконати для досягнення певної мети. Експерти моделюють поведінку користувача, оцінюючи, наскільки інтуїтивно зрозумілий інтерфейс.

Переваги:

a. Виявляє проблеми на рівні розуміння та навчання.
b. Корисний для нових або складних систем.

Недоліки:

a. Не враховує реальну поведінку користувачів.
b. Може бути трудомістким для складних додатків [9].

Аналіз завдань (Task Analysis). Аналіз завдань передбачає детальний розбір дій, які користувач виконує для досягнення своїх цілей. Це допомагає оптимізувати робочі процеси та спростити інтерфейс.

Переваги:

a. Глибоке розуміння потреб та поведінки користувачів.
b. Допомагає виявити непотрібні або складні кроки.

Недоліки:

a. Вимагає значних ресурсів та часу.
b. Складність у застосуванні для динамічних систем [9].

Аналіз протоколів (Protocol Analysis). Метод полягає в записі думок користувача під час виконання завдань (так званий "подумковий протокол"). Це дозволяє зрозуміти когнітивні процеси та труднощі, з якими стикається

користувач.

Переваги:

a. Глибоке розуміння ментальних моделей користувача.
b. Виявлення прихованих проблем у дизайні.

Недоліки:

a. Процес може впливати на природну поведінку користувача.
b. Аналіз отриманих даних є складним та ресурсозатратним [9].

Веб-аналітика та метрики. Сучасні інструменти веб-аналітики дозволяють збирати кількісні дані про поведінку користувачів: час на сторінці, шлях навігації, показники відмов тощо. Аналіз цих даних допомагає ідентифікувати проблемні зони у додатку.

Переваги:

a. Об'єктивні дані про реальну поведінку великої кількості користувачів.
b. Можливість постійного моніторингу та вдосконалення.

Недоліки:

a. Не виявляє причини проблем, лише симптоми.
b. Потребує правильної інтерпретації даних [7].

A/B-тестування. Метод передбачає порівняння двох або більше версій інтерфейсу для визначення найбільш ефективного рішення. Користувачі випадковим чином розподіляються між версіями, і їхня поведінка аналізується.

Переваги:

a. Емпіричне підтвердження ефективності змін.
b. Дозволяє приймати рішення на основі даних.

Недоліки:

a. Потребує достатнього обсягу трафіку.
b. Може бути складним у налаштуванні та аналізі [6].

Карти кліків та теплові карти. Ці методи візуалізують дії користувачів на сторінці, показуючи, де вони клікають, переміщують мишу або переглядають. Це допомагає зрозуміти, які елементи привертають увагу та як

користувачі взаємодіють з інтерфейсом.

Переваги:

a. Візуальне представлення поведінки користувачів.
b. Допомагає оптимізувати розташування елементів.

Недоліки:

a. Не дає інформації про причини поведінки.
b. Може вимагати додаткових інструментів для збору даних [8].

Метод "Персонажів" (Personas). Створення узагальнених профілів користувачів допомагає дизайнерам та розробникам зрозуміти потреби та мотивації різних сегментів аудиторії. Це сприяє більш цілеспрямованому та користувацько-орієнтованому дизайну.

Переваги:

a. Фокусує увагу на реальних потреб користувачів.
b. Полегшує комунікацію в команді розробки.

Недоліки:

a. Персонажі можуть бути занадто узагальненими.
b. Потребує досліджень для створення достовірних профілів [9].

Сценарії використання (Use Cases). Розробка сценаріїв використання дозволяє описати типові ситуації, в яких користувач взаємодіє з системою. Це допомагає виявити вимоги до функціональності та ергономіки додатку.

Переваги:

a. Чітке визначення вимог та очікувань.
b. Сприяє розумінню контексту використання.

Недоліки:

a. Може не враховувати нестандартні ситуації.
b. Потребує оновлення при зміні вимог або функціональності [5].

1.4 Методи проведення дослідження

У процесі розробки інформаційної технології для ергономічного проєктування вебдодатків важливо застосовувати ефективні методи

дослідження, що дозволяють глибоко аналізувати взаємодію користувача з системою. У цьому розділі розглядаються методи, використані для оцінки ергономічності вебдодатку, включаючи опитування користувачів, юзабіліті-тестування та інші підходи.

Загальний підхід до дослідження. Дослідження базується на поєднанні кількісних та якісних методів, що забезпечує комплексний аналіз ергономічних аспектів вебдодатку. Основні етапи дослідження включають:

a. Підготовчий етап: визначення цілей та завдань дослідження, вибір методів та інструментів.
b. Збір даних: проведення опитувань, тестувань та збору аналітичних даних.
c. Аналіз даних: обробка отриманої інформації та виявлення закономірностей.
d. Формування висновків та рекомендацій: розробка пропозицій для покращення ергономіки вебдодатку.

Розробка та проведення опитування користувачів. Одним із ключових методів дослідження є опитування користувачів за допомогою спеціально розробленого анкета. Опитування спрямоване на виявлення суб'єктивного сприйняття користувачами ергономічних аспектів вебдодатку. На рисунку 1.3 можна побачити приклад.

Етапи роботи з опитуванням:

a. Створення анкети: розроблено опитувальник, що містить питання, спрямовані на оцінку простоти навігації, інтуїтивність інтерфейсу та оптимального розташування елементів.
b. Пілотне тестування: проведено попереднє тестування анкети на невеликій групі користувачів для виявлення можливих недоліків та їх усунення.
c. Поширення анкети: опитування було розповсюджено серед цільової аудиторії через електронну пошту та соціальні мережі.
d. Збір відповідей: отримано достатню кількість заповнених анкет для

проведення статистично значущого аналізу.

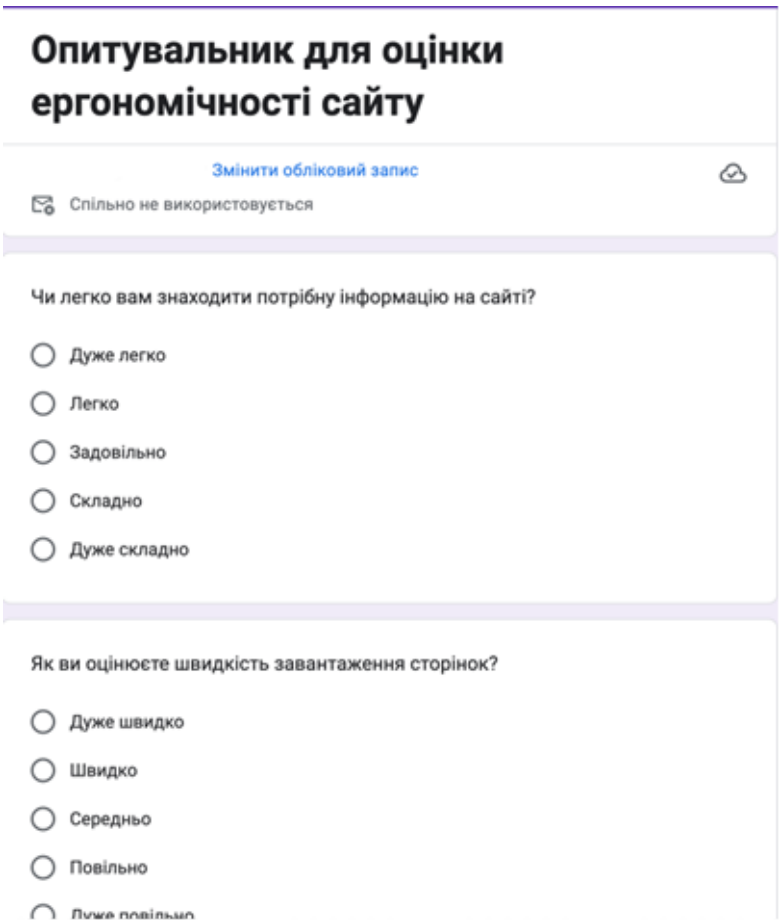

Рисунок 1.3 - Приклад питань в опитувальнику

Юзабіліті-тестування. Юзабіліті-тестування дозволяє отримати якісні дані про взаємодію користувачів з вебдодатком в реальному часі.

Процес тестування включає:

a. Відбір учасників: залучено представників цільової аудиторії з різним рівнем досвіду.
b. Розробка сценаріїв завдань: учасникам пропонується виконати певні завдання, що відображають типові сценарії використання додатку.
c. Спостереження та фіксація дій: ведеться запис дій користувачів, їхніх коментарів та реакцій.
d. Аналіз результатів: виявляються проблемні зони та можливості для покращення ергономіки.

Використання інструментів веб-аналітики. Для збору кількісних даних про поведінку користувачів застосовуються сучасні інструменти веб-аналітики, такі як Google Analytics та інші.

Завдяки веб-аналітиці отримано:

a. Статистику відвідуваності: кількість унікальних користувачів, частота відвідувань.
b. Аналіз поведінки: середній час перебування на сторінках, показник відмов, шлях користувача.
c. Дані про взаємодію: частота кліків на певні елементи, використання функціоналу.

Експертна оцінка інтерфейсу. Експерти в галузі UX/UI-дизайну провели оцінку вебдодатку на відповідність принципам ергономічного проєктування.

Підходи експертної оцінки:

a. Аналіз за евристиками Нільсена: перевірка відповідності 10 евристикам юзабіліті.
b. Перевірка консистентності: оцінка узгодженості стилів, шрифтів та поведінки елементів.
c. Виявлення потенційних проблем: прогнозування можливих труднощів користувачів.

Аналіз та обробка даних. Отримані дані з різних джерел обробляються з використанням статистичних методів та спеціалізованого програмного забезпечення.

Методи аналізу:

a. Дескриптивна статистика: розрахунок середніх значень, медіан, мод та стандартних відхилень.
b. Кореляційний аналіз: виявлення зв'язків між різними параметрами ергономіки.
c. Контент-аналіз відкритих відповідей: систематизація та інтерпретація якісних даних.

1.5 Постановка завдання дослідження

Метою даної роботи є розробка інформаційної технології для ергономічного проєктування вебдодатку, що забезпечує зручність використання та ефективну взаємодію користувача з інтерфейсом.

Для досягнення цієї мети необхідно виконати наступні завдання:

1. Провести аналітичний огляд предметної області.
2. Визначити основні критерії оцінки ергономічності вебдодатку.
3. Застосувати принципи ергономічного проектування для розробки вебдодатку.
4. Оцінити рівень ергономічності сайту на основі даних опитування та аналізу існуючих методів.

2. ОГЛЯД МЕТОДІВ ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ

2.1 Визначення критеріїв оцінки ергономіки вебдодатків

Ефективна оцінка ергономіки вебдодатку вимагає розробки чітких та вимірюваних критеріїв, які дозволять об'єктивно аналізувати зручність використання, інтуїтивність інтерфейсу та загальний користувацький досвід. У цьому розділі будуть визначені основні критерії оцінки ергономіки вебдодатків та розроблені методи їх вимірювання.

На основі аналізу принципів ергономічного проєктування та сучасних методів ергономічного аналізу було виділено наступні ключові критерії для оцінки ергономіки вебдодатків:

1. Простота навігації
    a. Легкість переміщення по вебдодатку;
    b. Зрозумілість структури меню та логіки переходів.
    c. Наявність та зручність використання пошуку.
2. Інтуїтивність інтерфейсу
    a. Зрозумілість елементів управління та їх функцій.
    b. Використання зрозумілих іконок та підказок.
    c. Відповідність очікуванням користувача.
3. Оптимальне розташування елементів
    a. Логічне та зручне розміщення елементів на сторінці.
    b. Відповідність пріоритетів інформації та функцій.
    c. Використання принципів візуальної ієрархії.
4. Консистентність дизайну
    a. Узгодженість стилів, кольорової гами та шрифтів.
    b. Стабільність поведінки елементів на різних сторінках.
    c. Дотримання встановлених шаблонів та стандартів.
5. Доступність
    a. Відповідність стандартам доступності (наприклад, WCAG).

   b. Можливість використання вебдодатку користувачами з особливими потребами.
   c. Адаптивність до різних пристроїв та розширень екрану.
6. Зворотний зв'язок
   a. Наявність повідомлень про результати дій користувача.
   b. Інформативність помилок та підказок.
   c. Швидкість реакції системи на дії користувача.
7. Гнучкість та персоналізація
   a. Можливість налаштування інтерфейсу під потреби користувача.
   b. Підтримка різних мов та регіональних налаштувань.
   c. Варіативність способів виконання завдань.
8. Мінімізація когнітивного навантаження
   a. Уникнення перевантаження інформацією.
   b. Простота та зрозумілість формулювань.

2.2 Визначення метрик для оцінки критеріїв

Для кожного з визначених критеріїв необхідно розробити відповідні метрики, які дозволяють кількісно або якісно оцінити рівень ергономіки.

1. Простота навігації
   a. Час виконання навігаційних завдань: середній час, необхідний користувачу для знаходження потрібної інформації.
   b. Кількість кліків до цільової сторінки: середня кількість переходів для досягнення мети.
   c. Показник відмов: відсоток користувачів, які покинули сайт після перегляду однієї сторінки.
2. Інтуїтивність інтерфейсу
   a. Кількість помилок при виконанні завдань: частота неправильних дій користувача.
   b. Суб'єктивна оцінка інтуїтивності: результати опитування

користувачів за шкалою Лікерта.

c. Час навчання: період, необхідний для освоєння основних функцій додатку.

3. Оптимальне розташування елементів
    a. Середній час знаходження елементів: як швидко користувач знаходить потрібний елемент інтерфейсу.
    b. Теплові карти кліків: аналіз зон активності на сторінці.
    c. Суб'єктивна оцінка зручності розташування: опитування користувачів.
4. Консистентність дизайну
    a. Кількість невідповідностей у стилі: кількість виявлених неконсистентних елементів.
    b. Експертна оцінка консистентності: результати аналізу дизайну фахівцями.
    c. Суб'єктивна оцінка користувачів: сприйняття узгодженості інтерфейсу.
5. Доступність
    a. Відповідність стандартам WCAG: ступінь відповідності критеріям доступності.
    b. Тестування з допомогою спеціалізованих інструментів: використання програм для перевірки доступності.
    c. Зворотний зв'язок від користувачів з особливими потребами: опитування та тестування.
6. Зворотний зв'язок
    a. Час реакції системи: середній час між дією користувача та відповіддю системи.
    b. Якість повідомлень: оцінка інформативності та зрозумілості повідомлень.
    c. Кількість звернень до служби підтримки: частота запитів про допомогу.

7. Гнучкість та персоналізація
    a. Кількість доступних налаштувань: кількість параметрів, які користувач може змінити.
    b. Частота використання налаштувань: скільки користувачів змінюють стандартні налаштування.
    c. Задоволеність можливостями персоналізації: результати опитування.
8. Мінімізація когнітивного навантаження
    a. Кількість інформації на сторінці: кількість елементів та тексту.
    b. Суб'єктивна оцінка простоти сприйняття: відгуки користувачів щодо зрозумілості інформації.

Інструменти та методи вимірювання. Для збору та аналізу даних за розробленими метриками будуть використані наступні інструменти та методи:

a. Опитування користувачів: онлайн-анкети з використанням платформ типу Google Forms або SurveyMonkey.
b. Юзабіліті-тестування: спостереження за користувачами під час виконання ними типових завдань.
c. Веб-аналітика: застосування Google Analytics, Hotjar для збору статистичних даних про поведінку користувачів.
d. Експертна оцінка: залучення фахівців з UX/UI-дизайну для аналізу інтерфейсу.

Встановлення порогових значень. Для об'єктивної оцінки необхідно визначити порогові значення для кожної метрики, які будуть вказувати на прийнятний або незадовільний рівень ергономіки.

Порогові значення можуть базуватися на (рис 2.1):

a. Галузевих стандартах та рекомендаціях: наприклад, час реакції системи не повинен перевищувати 0,1 секунди для миттєвої відповіді.
b. Досвіді користувачів: середні значення, отримані в результаті опитувань або тестувань.

Таблиця 2.1 – Критерії та метрики [3]

Критерій	**Метрика**	**Метод**
Простота навігації	Час виконання	Юзабіліті тестування
	Кількість кліків до цільової сторінки	Вебаналітика
Інтуїтивність інтерфейсу	Кількість помилок при виконанні завдань	Юзабіліті тестування
	Суб’єктивна оцінка інтуїтивності	Опитування користувачів
Оптимальне розташування елементів	Середній час знаходження елементів	Юзабіліті тестування
	Теплові карти кліків	Вебаналітика
	Суб’єктивна оцінка зручності розташування	Опитування користувачів
Консистентність дизайну	Кількість невідповідностей у стилі	Експертна оцінка
	Суб’єктивна оцінка користувачів	Опитування
Доступність	Відповідність стандартам WCAG	Автоматизовані інструменти
	Зворотній зв’язок від користувачів з особливими потребами	Опитування, тестування
Зворотній зв’язок	Час реакції системи	Тестування, моніторинг
	Якість повідомлень	Експертна оцінка, опитування
Гнучкість та персоналізація	Кількість доступних налаштувань	Аналіз функціоналу
	Частота використання налаштувань	Вебаналітика

c. Конкурентному аналізі: порівняння з показниками аналогічних вебдодатків.

Матриця критеріїв та метрик. Для зручності аналізу пропонується скласти матрицю, яка відображає зв'язок між критеріями, метриками та методами їх вимірювання.

Розроблені критерії та метрики дозволяють здійснювати систематичну та об'єктивну оцінку ергономічності вебдодатків. Вони охоплюють основні

аспекти користувацького досвіду та можуть бути застосовані як під час розробки, так і при подальшому вдосконаленні продукту. У наступних розділах ці критерії будуть використані для оцінки розробленого вебдодатку та формування рекомендацій щодо його покращення.

2.3 Вибір засобів програмної реалізації

У процесі розробки інформаційної технології для ергономічного проєктування вебдодатку критично важливо обрати оптимальні технології та інструменти, які забезпечують ефективність розробки, масштабованість, продуктивність та зручність у використанні. Після аналізу доступних рішень було прийнято рішення використовувати React.js для фронтенд-розробки, Next.js як фреймворк для React, Strapi як бекенд-рішення, а також Docker для контейнеризації та розгортання додатку. Нижче наведено обґрунтування вибору кожної з цих технологій.

React.js для фронтенд-розробки. React.js — це популярна бібліотека JavaScript для створення користувацьких інтерфейсів, розроблена компанією Facebook. Вона базується на компонентному підході, що дозволяє будувати складні інтерфейси з невеликих, ізольованих шматочків коду, які називаються компонентами [10].

Переваги використання React.js:

a. Компонентний підхід: Полегшує розробку та підтримку інтерфейсу, сприяє повторному використанню коду.
b. Висока продуктивність: Використання віртуального DOM мінімізує операції з реальним DOM, що покращує швидкість роботи додатку.
c. Велика спільнота та екосистема: Наявність численних бібліотек та інструментів, що доповнюють можливості React.
d. Гнучкість: Можливість інтеграції з іншими фреймворками та бібліотеками.

React.js забезпечує високу швидкість та ефективність розробки

інтерфейсу користувача, що є критичним для створення ергономічного вебдодатку. Компонентний підхід сприяє підтримці консистентності дизайну та спрощує впровадження принципів ергономіки.

Next.js як фреймворк для React. Next.js — це фреймворк для React, який надає можливості серверного рендерингу та генерації статичних сайтів. Він спрощує розробку вебдодатків, забезпечуючи оптимізацію продуктивності та SEO [16].

Переваги використання Next.js:

a. Серверний рендеринг (SSR): Підвищує продуктивність та покращує індексацію сторінок пошуковими системами.
b. Статична генерація (SSG): Дозволяє генерувати статичні сторінки під час зборки, що покращує швидкість завантаження.
c. Автоматична оптимізація: Next.js оптимізує зображення та код для швидкого завантаження.
d. Маршрутизація з коробки: Спрощує управління маршрутами та навігацією у додатку [15].

Використання Next.js у поєднанні з React.js забезпечує додаткові переваги у продуктивності та SEO-оптимізації, що є важливим для зручності користувачів та успішності вебдодатку.

Strapi для бекенд-розробки. Strapi — це гнучкий Headless CMS на базі Node.js, який дозволяє створювати власні API швидко та ефективно [16]. Детальна модель архітектури наведено на рисунку 3.10. Він надає зручний інтерфейс для управління контентом та підтримує як RESTful, так і GraphQL API. [16]

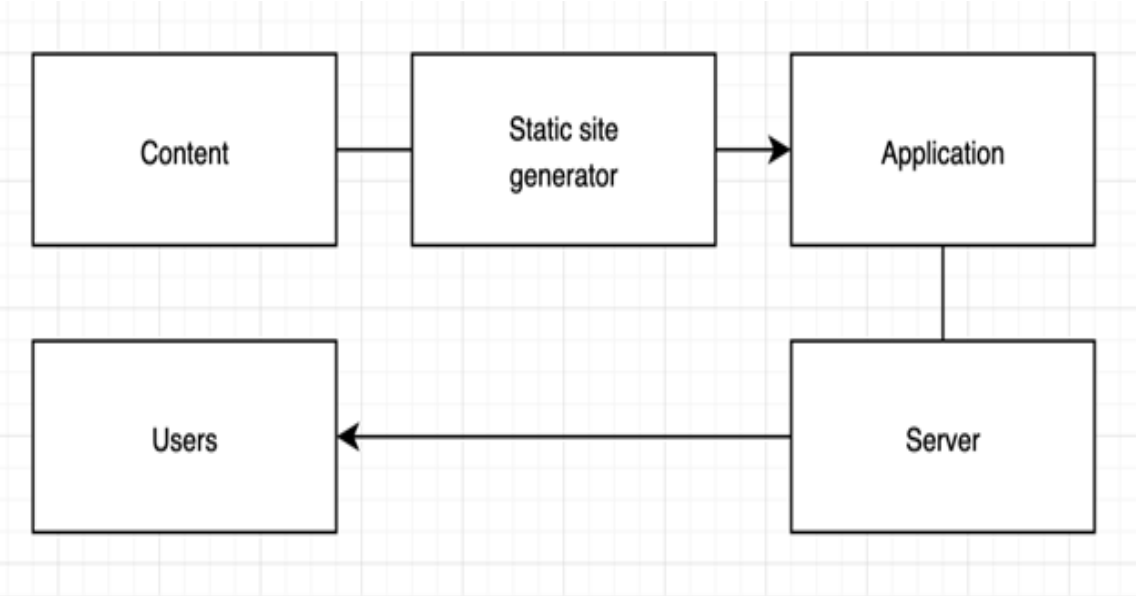

Рисунок 2.4 - Модель архітектури backend

Переваги використання Strapi:

a. Швидка розробка API: Автоматична генерація API на основі визначених моделей даних.

b. Гнучкість моделювання даних: Зручний інтерфейс для створення та управління контент-типами.

c. Підтримка аутентифікації та авторизації: Вбудовані механізми управління користувачами та ролями.

d. Масштабованість та розширюваність: Можливість додавати плагіни та кастомізувати функціонал [17].

Strapi дозволяє зосередитися на розробці фронтенду та ергономічних аспектів додатку, забезпечуючи швидке та ефективне створення бекенду без необхідності писати багато коду з нуля.

Docker для контейнеризації та розгортання. Docker — це платформа для розробки, доставки та запуску додатків у контейнерах. Контейнери дозволяють пакувати додаток з усіма його залежностями, що забезпечує консистентність середовища розробки, тестування та продакшну. Основні переваги це:

a. Легка масштабованість: Спрощує розгортання та управління додатками в різних середовищах.

Таблиця 2.2 – Аналіз наведених технологій

Технологія	Опис	Переваги / Недоліки
Angular	Повнофункціональний фронтенд-фреймворк на основі TypeScript, розроблений Google для створення вебдодатків з багатим інтерфейсом користувача.	Комплексне рішення з вбудованими інструментами Сильна типізація завдяки TypeScript. Підтримка від Google та велика спільнота.
		Більш крута крива навчання та складність у освоєнні
		Важчий у налаштуванні та конфігурації
		Великий розмір кінцевого додатку
Vue.js	Прогресивний JavaScript-фреймворк для побудови користувацьких інтерфейсів, який фокусується на представницькому шарі та легко інтегрується з іншими бібліотеками.	Легкість в освоєнні та використанні. Висока продуктивність та швидкодія. Гнучкість та можливість поступового впровадження.
		Менша спільнота та екосистема порівняно з React. Обмежена кількість готових рішень та плагінів. Може бути складно інтегрувати у великі проєкти
Express.js	Мінімалістичний веб-фреймворк для Node.js, який надає базові функції для побудови вебдодатків та API.	Простота та гнучкість. Велика спільнота та багато модулів. Легкий та швидкий у роботі.

b. Консистентність середовища: Уникнення проблем, пов'язаних з різними конфігураціями систем.

c. Ефективне використання ресурсів: Контейнери мають менші накладні витрати порівняно з віртуальними машинами [19].

Docker спрощує процес розгортання вебдодатку та забезпечує стабільність його роботи в різних середовищах, що є важливим для підтримки високого рівня ергономіки та доступності для користувачів.

Детальну структуру docker файлу використаного в проекті, наведено у Додатку А. Даний Dockerfile є прикладом ефективного використання контейнеризації в контексті ергономічного проєктування вебдодатків. Він забезпечує швидкий та послідовний процес збірки, розгортання і масштабування системи, полегшує адаптацію до різноманітних інфраструктурних умов, сприяючи тим самим підвищенню зручності, продуктивності та надійності розробленого продукту. Такий підхід відповідає концепції інформаційної технології ергономічного проєктування, де оптимізація та зручність взаємодії користувачів із системою стоять на одному рівні з ефективною внутрішньою організацією програмного забезпечення.

Синергія обраних технологій. Використання комбінації React.js, Next.js, Strapi та Docker забезпечує:

a. Повний стек розробки: Від бекенду до фронтенду з можливістю гнучкого налаштування.
b. Продуктивність та швидкість завантаження: Завдяки Next.js та оптимізації ресурсу.
c. Швидкість розробки: Автоматизація багатьох процесів, що дозволяє зосередитися на ергономічних аспектах.
d. Легкість розгортання та підтримки: Завдяки Docker та його можливостям контейнеризації.

Альтернативні технології та їх порівняння.

Перед остаточним вибором були розглянуті альтернативні технології. В таблиці 3.5 наведені основні переваги та недоліки зазначений технологій.

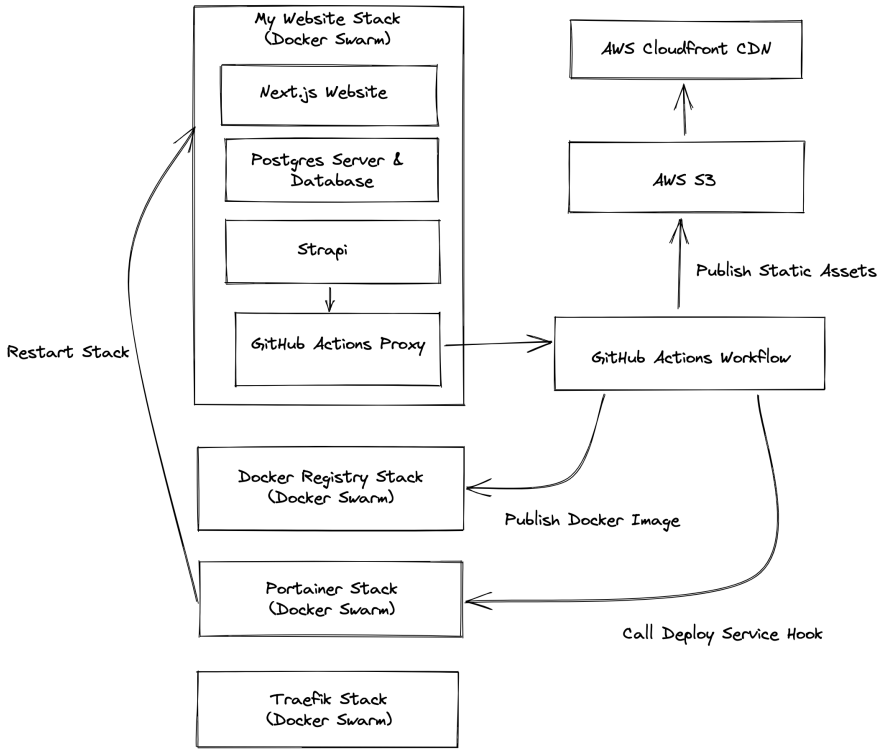

Рисунок 2.5 - Приклад архітектури з схожими технологіями [27]

На наведеному рисунку 2.5 проілюстровано архітектуру розгортання та взаємодії компонентів вебзастосунку у виробничому середовищі. Центральним елементом є «My Website Stack», який функціонує на основі оркестрації контейнерів Docker Swarm та включає в себе кілька ключових сервісів: інтерфейс на основі Next.js, сервер бази даних Postgres, бекенд на Strapi та проксі-шар для інтеграції з GitHub Actions. Такий підхід забезпечує гнучке масштабування, простоту оновлення та централізований контроль над усіма сервісами.

Вибір технологій React.js, Next.js, Strapi та Docker є оптимальним для розробки вебдодатку, який відповідає сучасним вимогам продуктивності, масштабованості та зручності використання. Ці технології забезпечують ефективний процес розробки, підтримку та подальший розвиток додатку.

3. РОЗРОБКА ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ

3.1 Опис вхідних і вихідних даних

У процесі моделювання даних потрібно врахувати потреби користувачів та кількість даних які буде отримувати додаток. Для цього потрібно виокремити ти виділити основі сущності системи для коректного опрацювання.

В контексті цього, була виконана розробка контекстної діаграми з урахуванням даних, що зазначені нижче:

− Вхідні дані: технічне завдання на розробку.

− Управління: документація Stripe, документація Next.js.

− Механізми: HTML, CSS, Next.js.

− Вихідні дані: спроектований додаток.

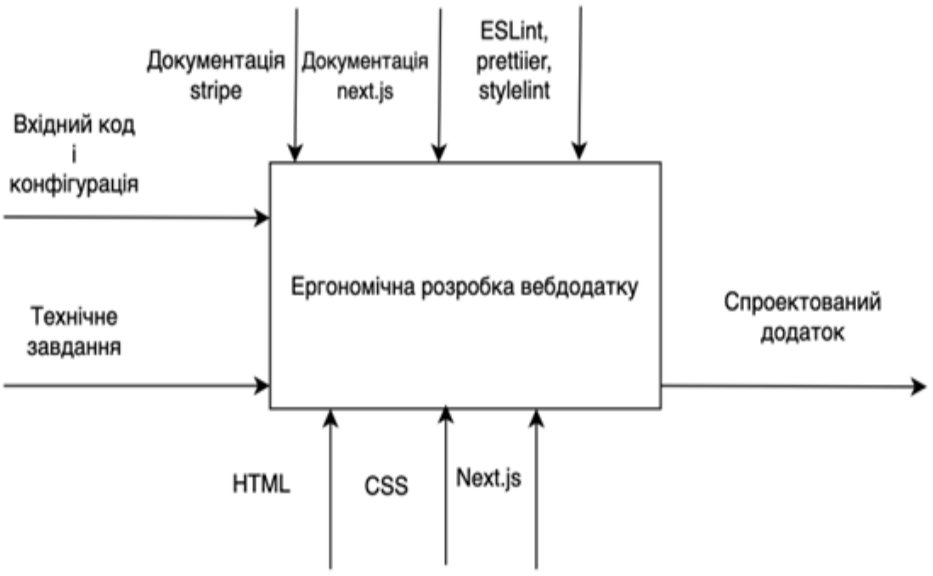

Рисунок 3.6 - IDEF0 діаграма фронтенд частини

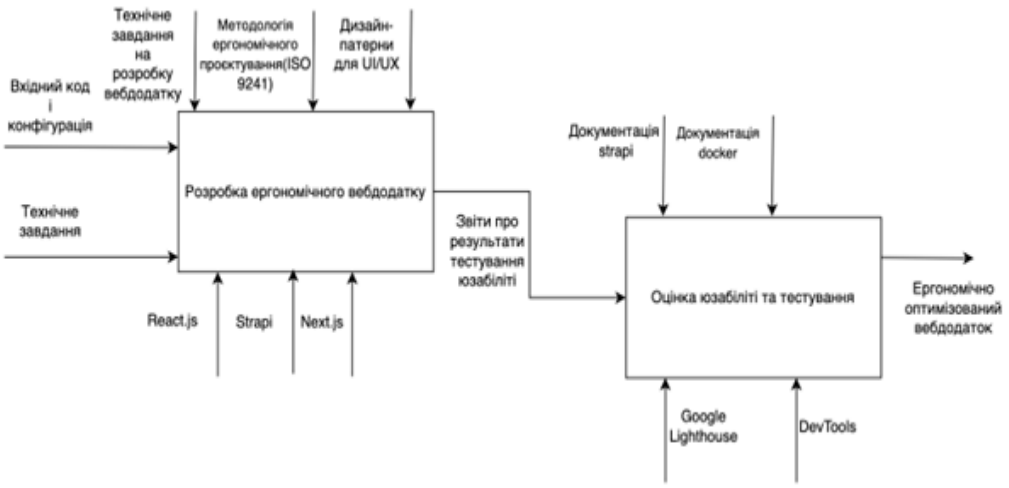

Рисунок 3.7 - Діаграма декомпозиції першого рівня

Після створення контекстної діаграми необхідно провести детальний аналіз основних завдань проєкту. Обрані завдання стають основою для формування діаграми першого рівня (рис 3.7) . Результати кожного завдання мають стати вхідними даними для наступного завдання. З урахуванням раніше сформованих завдань, було обрано два основних процеси реалізації проєкту «Ергономічне проєктування вебдодатку», які наведено нижче:

Процес «Розробка ергономічного вебдодатку»

− Вхідні дані: технічне завдання на розробку вебдодатку, вхідний код і конфігурація.

− Управління: методологія ергономічного проєктування, дизайн-патерни для UI/UX.

− Механізми: React.js, Next.js, Strapi

− Вихідні дані: прототип ергономічного вебдодатку

Процес «Оцінка юзабіліті та тестування»

− Вхідні дані: прототип ергономічного вебдодатку, тестові сценарії.

− Управління: документація Strapi, документація Docker.

− Механізми: Google Lighthouse, Chrome DevTools.

− Вихідні дані: звіт про результати тестування юзабіліті.

На рис. 3.6 представлена контекстна діаграма процесу «Розробка

ергономічного вебдодатку»

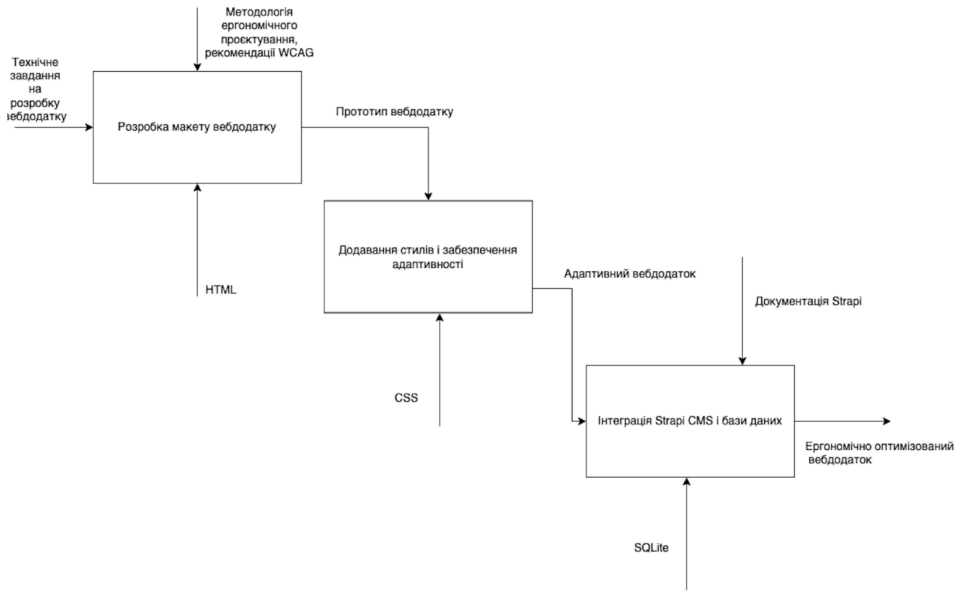

Рисунок 3.8 - Діаграма декомпозиції процесу

Для реалізації процесу були визначені три основні підзадачі (рис 3.8): розробка макету вебдодатку, додавання стилів і забезпечення адаптивності, інтеграція бази даних і CMS. Для кожної підзадачі наведено вхідні дані, механізми реалізації, управління та вихідні результати.

Підзадача «Розробка макету вебдодатку»

− Вхідні дані: технічне завдання на розробку вебдодатку.

− Управління: методологія ергономічного проєктування, загальні принципи розробки вебдодатків, рекомендації WCAG.

− Механізми: React.js, Next.js.

− Вихідні дані: базовий макет вебдодатку.

Підзадача «Додавання стилів і забезпечення адаптивності»

− Вхідні дані: базовий макет вебдодатку.

− Управління: дизайн-патерни для UI/UX, рекомендації щодо адаптивності.

− Механізми: CSS.

− Вихідні дані: адаптивний стилізований вебдодаток.

Підзадача «Інтеграція бази даних і CMS»

− Вхідні дані: адаптивний стилізований вебдодаток.

− Управління: документація Strapi CMS.

− Механізми: Strapi CMS, SQLite.

− Вихідні дані: повністю функціональний ергономічний вебдодаток.

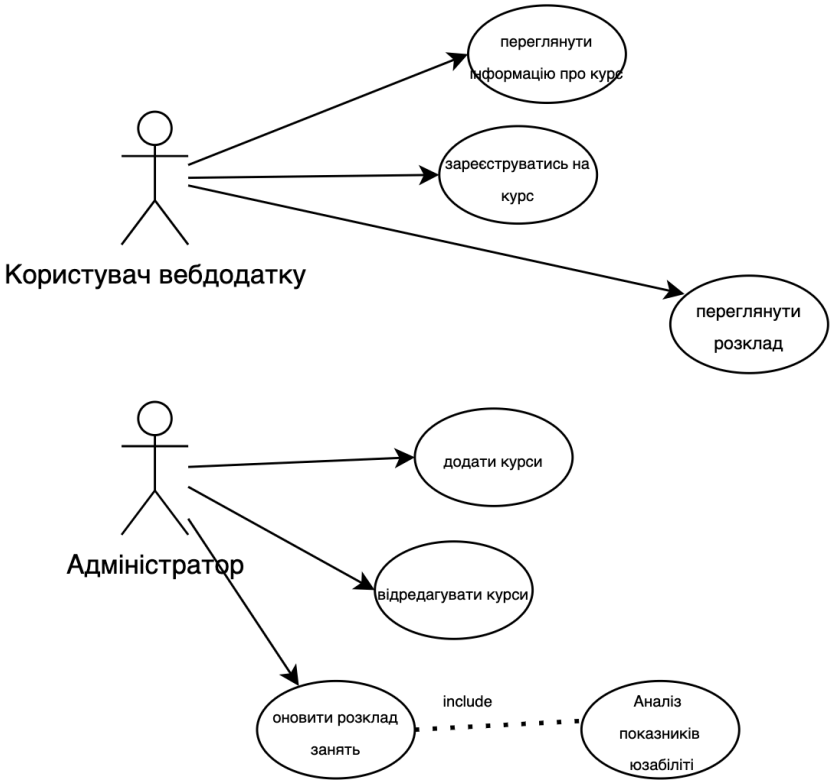

Рисунок 3.9 - Use case diagram Користувачі та функції

Рисунок 3.10 - Процес реєстрації на курс

Структура моделі даних у Strapi. У Strapi модель даних будується на основі контент-типів (Content Types), які можуть бути колекційними (Collection Type) або одноразовими (Single Type). В таблиці 2.2, наведені основі контент типи, поля та їх опис [17].

Таблиця 2.3 — Контент-типи та їх поля у Strapi [17]

Контент-тип	**Поле**	**Тип поля**	**Опис**
Користувачі	Ім’я користувача	Текстове поле	Логін або ім’я користувача
	Електронна пошта	Текстове поле	Адреса електронної пошти користувача
	Пароль	Текстове поле	Пароль для входу в систему
	Роль	Текстове поле	Роль користувача
Статті	Заголовок	Текстове поле	Назва статті
	Зміст	Текстове поле	Основний текст статті
	Автор	Зв’язок Many-to-one	Автор статті
	Категорія	Зв’язок	Категорія, до якої належить стаття
	Дата публікації	Поле дати	Дата та час публікації статті
Коментарі	Зміст	Текстове поле	Текст коментаря
	Автор	Зв’язок Many-to-One з “Користувачі”	Користувач, який залишив коментар
	Стаття	Зв’язок Many-one	Стаття, до якої належить коментар
	Дата створення	Поле дати	Дата та час створення коментаря

Представлена в Таблиці 2.3 структура відносин між контент-типами демонструє ключові зв’язки в інформаційній моделі вебзастосунку. Зокрема, визначено типи відносин «один до багатьох» між користувачами та статтями, статтями та коментарями, а також користувачами та коментарями. Така схема дозволяє забезпечити гнучкий і масштабований підхід до керування контентом та взаємодіями між користувачами. Водночас, відношення «багато до багатьох» між статтями та категоріями створює умови для ефективної

класифікації та зручної навігації у великому масиві інформації.

Таблиця 2.4 — Відносини між контент-типами [17]

Відношення	**Тип відносин**	**Опис**
Користувачі ↔ Статті	Один до багатьох (One-to-Many)	Один користувач може мати багато статей.
Статті ↔ Категорії	Багато до багатьох (Many-to-Many)	Стаття може належати багатьом категоріям, і категорія може містити багато статей.
Користувачі ↔ Коментарі	Один до багатьох (One-to-Many)	Один користувач може залишати багато коментарів.
Статті ↔ Коментарі	Один до багатьох (One-to-Many)	Одна стаття може мати багато коментарів.
Користувачі ↔ Профілі (необов'язково)	Один до одного (One-to-One)	Користувач має один профіль з додатковою інформацією.

Таблиця 2.5 — Налаштування прав доступу [17]

Роль	Дозволи
Адміністратор	Повний доступ до всіх контент-типів та налаштувань. Може створювати, редагувати та видаляти записи, управляти користувачами та правами доступу.
Редактор	Може створювати та редагувати Статті та Категорії, але не має доступу до налаштувань системи або управління користувачами.
Автор	Може створювати та редагувати лише власні Статті та Коментарі.
Користувач	Може залишати Коментарі, редагувати свій профіль.
Гість	Має доступ лише до перегляду публічного контенту (Статті, Категорії).

Наявність необов’язкового відношення «один до одного» між користувачами та профілями надає можливість розширювати функціонал

платформи відповідно до потреб користувачів, забезпечуючи гнучкість та персоналізацію при подальшому розвитку системи. Представлена в Таблиці 2.4 схема дозволів чітко визначає обов’язки та права доступу кожної ролі в системі. Адміністратор має повний контроль над усіма контент-типами та налаштуваннями: він може створювати, редагувати та видаляти матеріали, а також керувати користувачами та їхніми правами.

На рисунках 3.11, 3.12 наведена схема бази даних, яка відображає структуру модулів і компонентів, що використовуються для організації контенту в системі.

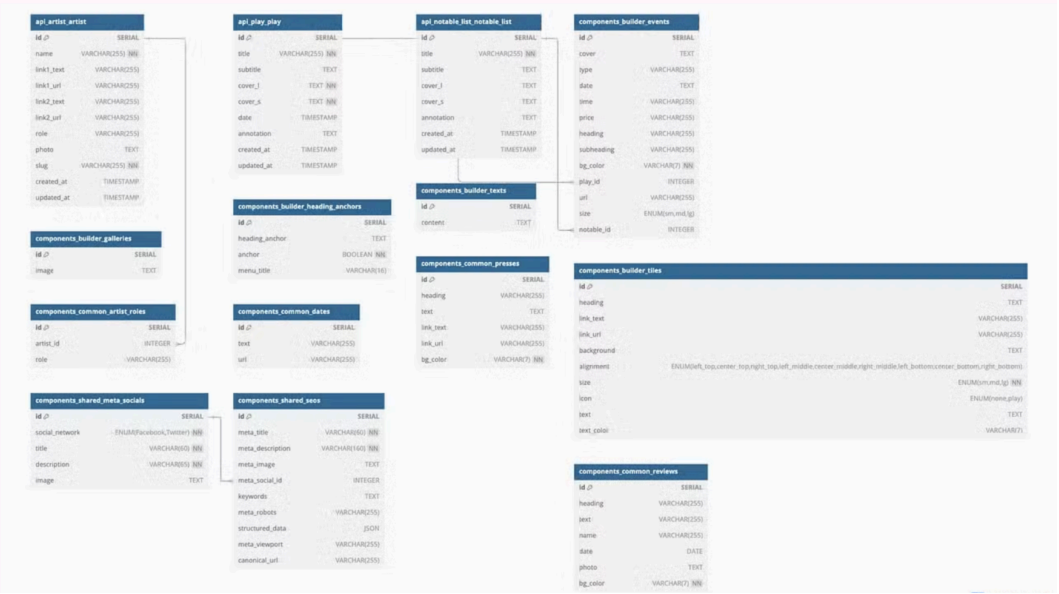

Рисунок 3.11 - Основні таблиці сутностей

Кожна таблиця відображає сутність доменної області застосунку, а стовпці таблиць містять поля з відповідними типами даних та обмеженнями. Таблиці в просторі «api»: Таблиці з префіксом api_ відповідають контентним сутностям, які користувач може безпосередньо змінювати через інтерфейс через інтерфейс. Наприклад: api_play_play: Зберігає інформацію про вистави чи події типу «play». Серед основних полів: title (VARCHAR(255), NN) — обов’язкова назва вистави. subtitle (TEXT) — підзаголовок чи додатковий опис.

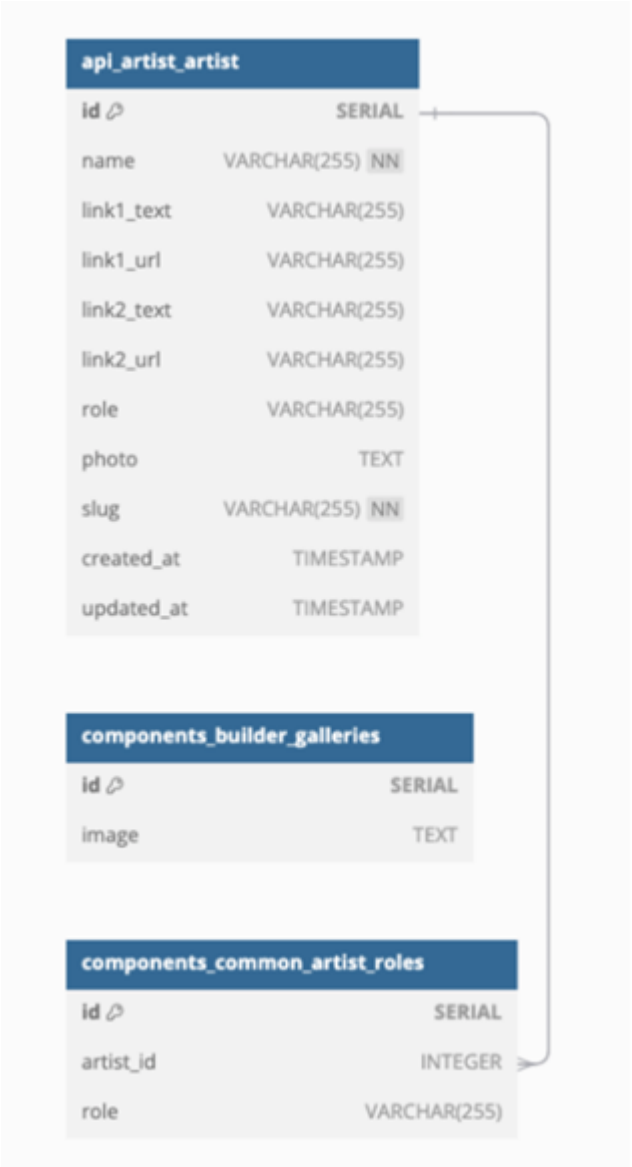

Рисунок 3.12 - Таблиці strapi

Реалізація моделі даних у Strapi. Створення контент-типів: У адміністративній панелі Strapi створюються контент-типи за допомогою вбудованого конструктора. Додавання полів: Для кожного контент-типу додаються необхідні поля, вказуються їх типи та налаштування (обов'язкові, унікальні тощо). Встановлення зв'язків: У процесі створення полів можна встановлювати зв'язки між контент-типами, вибираючи відповідний тип відносин. Генерація API: Після збереження моделі даних Strapi автоматично генерує RESTful API для доступу до даних.

Інтеграція Strapi з React.js. Для фронтенд-частини вебдодатку

використовується React.js, який отримує дані з бекенду через API, наданий Strapi.

Кроки інтеграції:

1. Налаштування запитів до API: Використовуючи бібліотеки для роботи з HTTP-запитами (наприклад, Axios або Fetch API), React-компоненти звертаються до Strapi API для отримання та відправки даних.
2. Обробка даних у React: Отримані дані зберігаються у стані компонентів або глобальному стані (з використанням Redux або Context API).
3. Відображення даних: Розробляються компоненти інтерфейсу для відображення даних користувачам. Це можуть бути списки статей, детальні сторінки, форми для введення даних тощо.
4. Управління станом та маршрутизацією: Використання React Router для навігації між сторінками та управління URL-адресами.

Забезпечення безпеки та автентифікації. Strapi надає вбудовані механізми автентифікації та управління користувачами, що важливо для захисту даних та керування доступом.

a. Ролі та дозволи: Можливість визначати різні ролі (адміністратор, редактор, користувач) та встановлювати для них дозволи.
b. JWT-токени: Використання JSON Web Tokens для автентифікації користувачів при доступі до захищених маршрутів API.
c. Захист API: Налаштування публічних та приватних маршрутів для контролю доступу до ресурсів.

Моделювання даних з використанням Strapi та React.js забезпечує ефективну взаємодію між бекендом та фронтендом, дозволяє швидко та гнучко розробляти вебдодатки з високим рівнем ергономічності. Використання цих технологій сприяє створенню зрозумілого та інтуїтивного інтерфейсу, оптимізує процес розробки та полегшує підтримку додатку. У наступних розділах буде розглянуто процес створення прототипу інформаційної технології та її тестування з точки зору ергономіки.

3.2 Реалізація функціональності вебдодатку

Загальна структура проєкту. Вебдодаток школи Малого театру реалізований за допомогою двох основних компонентів:

a. Frontend (папка SCHOOL-FE): Використано Next.js, що забезпечує серверний рендеринг і високу продуктивність. Типізація виконана за допомогою TypeScript. Інтерфейси користувача структуровані за допомогою компонентів у папці components. Дані локалізації зберігаються у файлах у папці i18n.ts. Для стилізації використовуються кастомні стилі з папки styles (рис. 3.11).
b. Backend (папка SCHOOL-CMS): Використовується Strapi CMS для управління контентом, який забезпечує гнучке API для фронтенду. База даних конфігурується у папці config/database. Основна логіка бекенду реалізована в папці src(рис. 3.14). Типи даних та інтерфейси описані у файлах у папці types (рис 3.12).

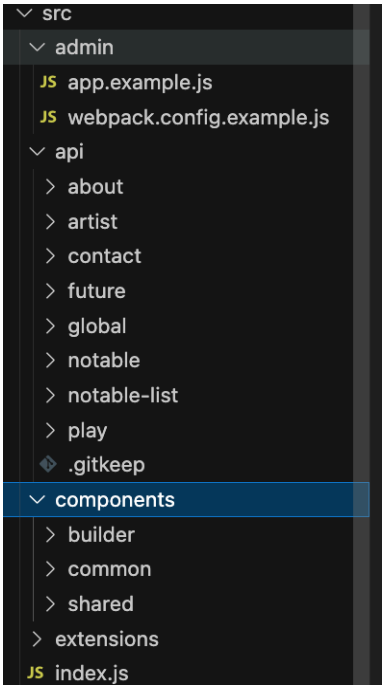

Рис 3.13 - Структура папки API та Components

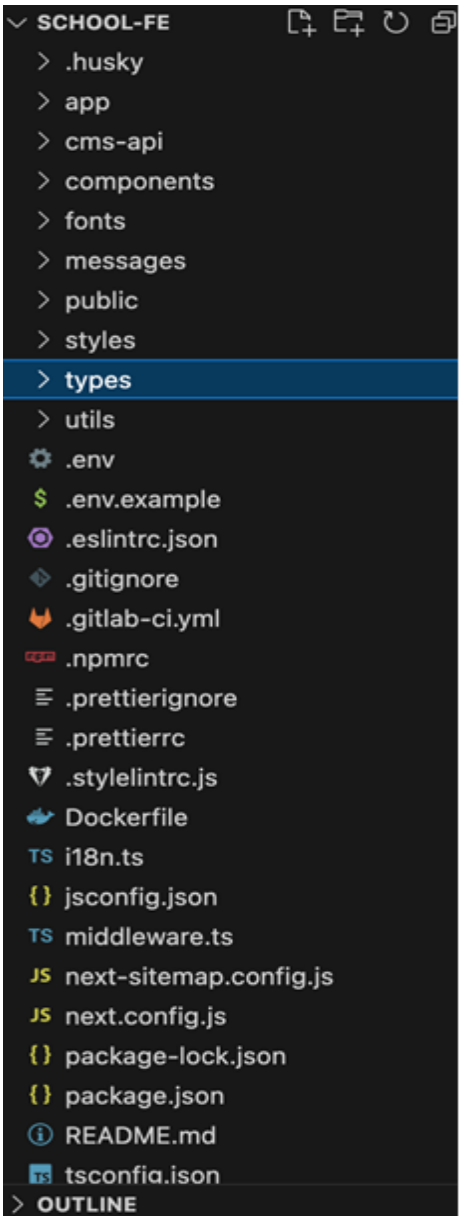

Рисунок 3.14 - Функціональність, реалізована на Frontend

Реалізація компонентів: Усі компоненти інтерфейсу зберігаються у папці components. Наприклад, компоненти для відображення списку курсів, розкладу або реєстраційної форми. Компоненти дотримуються принципу модульності, що дозволяє легко їх повторно використовувати.

Локалізація: Локалізація для багатомовної підтримки зберігається у файлі i18n.ts. Вебдодаток може бути адаптований для різних мов за допомогою цього конфігураційного файлу.

API-запити: Дані з CMS отримуються через API, реалізоване у папці cms-api. Це забезпечує динамічне оновлення інформації на фронтенді.

Оптимізація продуктивності: Використовується серверний рендеринг Next.js, що забезпечує швидке завантаження сторінок. Логіка маршрутизації налаштовується у файлі next.config.js.

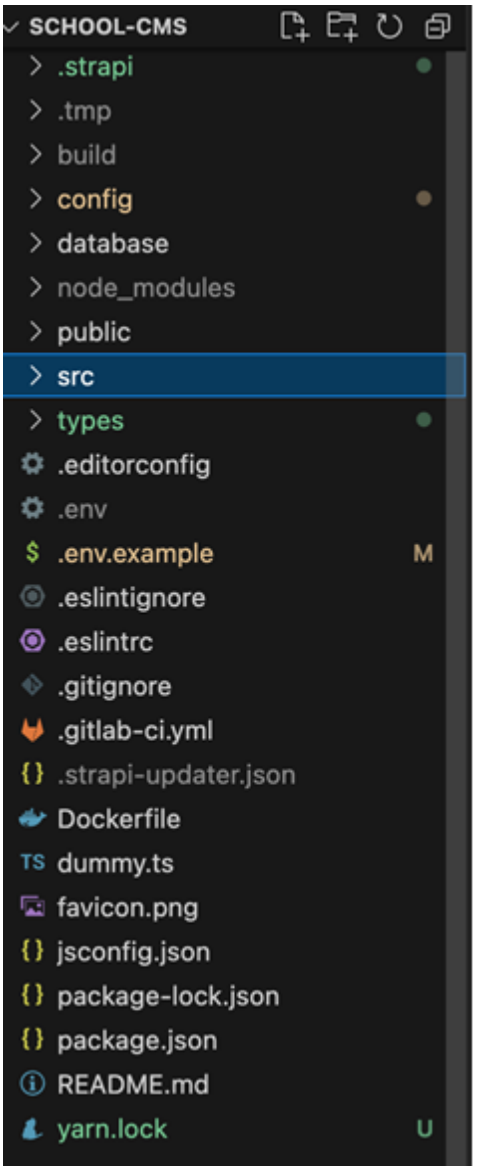

Рисунок 3.15 - Функціональність, реалізована на Backend

Управління контентом: Контент (інформація про курси, розклад, викладачів) створюється та редагується через панель Strapi CMS. Уся інформація зберігається у базі даних, конфігурація якої визначена у папці config/database.

Типізація даних: Типи та інтерфейси для контенту описані у папці types, що забезпечує перевірку коректності даних на етапі розробки.

Розширення функціональності: Додаткові ендпоінти для специфічних

запитів налаштовуються у папці src.

Автоматизація: Система побудована з використанням Docker (файл Dockerfile), що спрощує налаштування середовища для розробки та розгортання.

Інтеграція Frontend та Backend API-запити: Вебдодаток на фронтенді отримує дані про курси, розклад та іншу інформацію через REST API, який надає Strapi CMS.

Захист даних: Файли .env містять конфіденційні ключі для безпечного з'єднання з базою даних та API. Автоматизація CI/CD: У проекті реалізовано автоматизацію через .gitlab-ci.yml, що дозволяє швидке розгортання змін.

Реалізація функціональності вебдодатку для школи Малого театру базується на сучасних технологіях і забезпечує гнучкість, адаптивність та зручність управління контентом. Використання Next.js та Strapi CMS дозволяє швидко оновлювати інформацію, адаптуватися до потреб користувачів та забезпечує високий рівень продуктивності системи.

3.3 Тестування юзабіліті

Тестування юзабіліті проводиться для оцінки ефективності, доступності та швидкодії вебдодатку школи Малого театру. Основна мета полягає у виявленні проблем, які можуть перешкоджати комфортному використанню системи кінцевими користувачами, а також у перевірці відповідності інтерфейсу сучасним стандартам.Інструменти тестування.

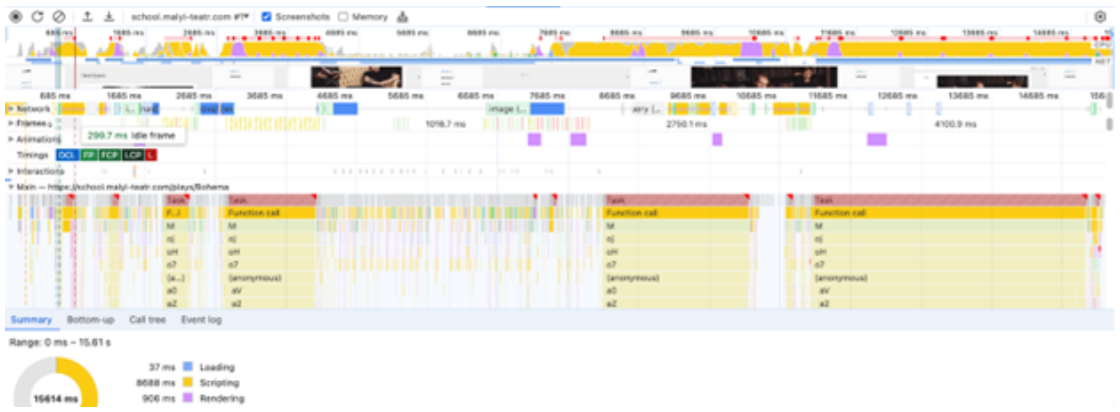

Рисунок 3.16 - Аналіз Lighthouse показників

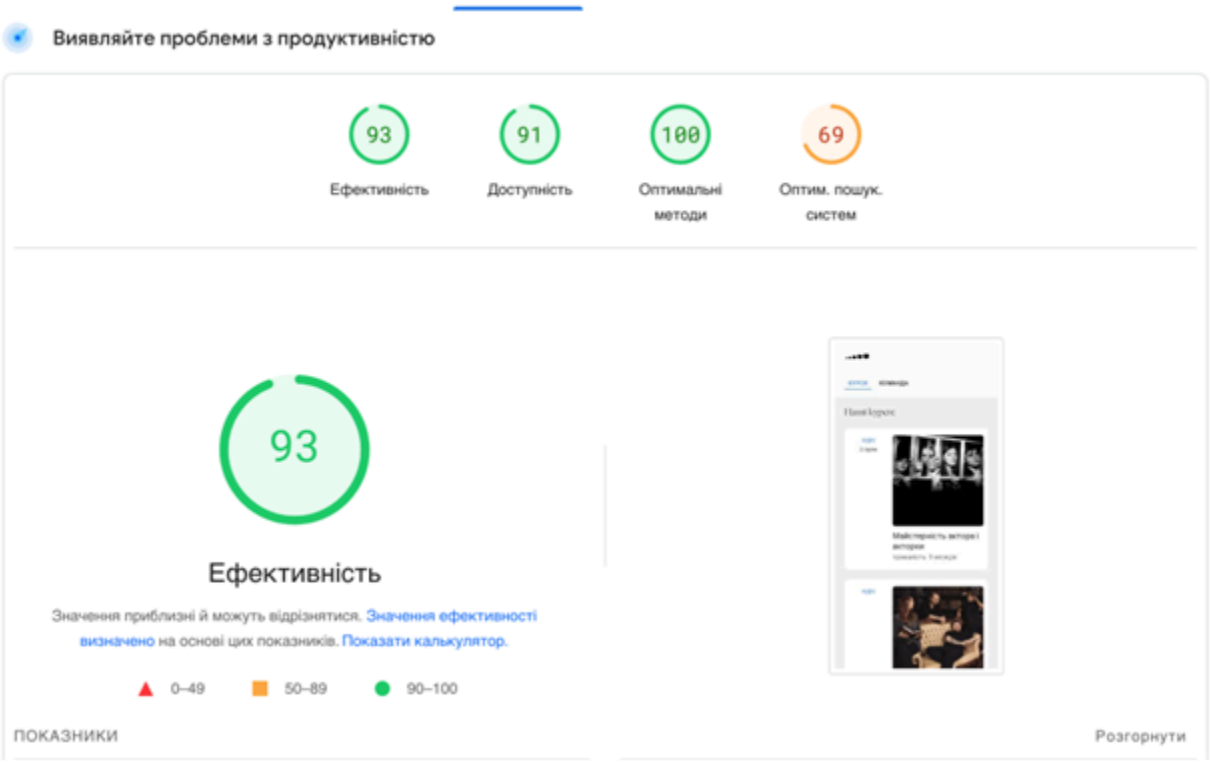

Рисунок 3.17 - Аналіз Lighthouse показників

a. Google Lighthouse: Інструмент для оцінки продуктивності, доступності та оптимізації вебдодатків.
b. Chrome DevTools: Використовувалося для аналізу продуктивності під час завантаження сторінок і виконання скриптів.

1. Ефективність За даними Google Lighthouse, загальна ефективність вебдодатку оцінюється на 93/100. На рисунку 3.14 це можна побачити. Це свідчить про високий рівень оптимізації інтерфейсу для користувачів. Основні показники ефективності: Швидкість завантаження: задовільна, але є можливості для оптимізації часу завантаження великих зображень. Логічна структура сторінки забезпечує зрозумілість для користувачів.

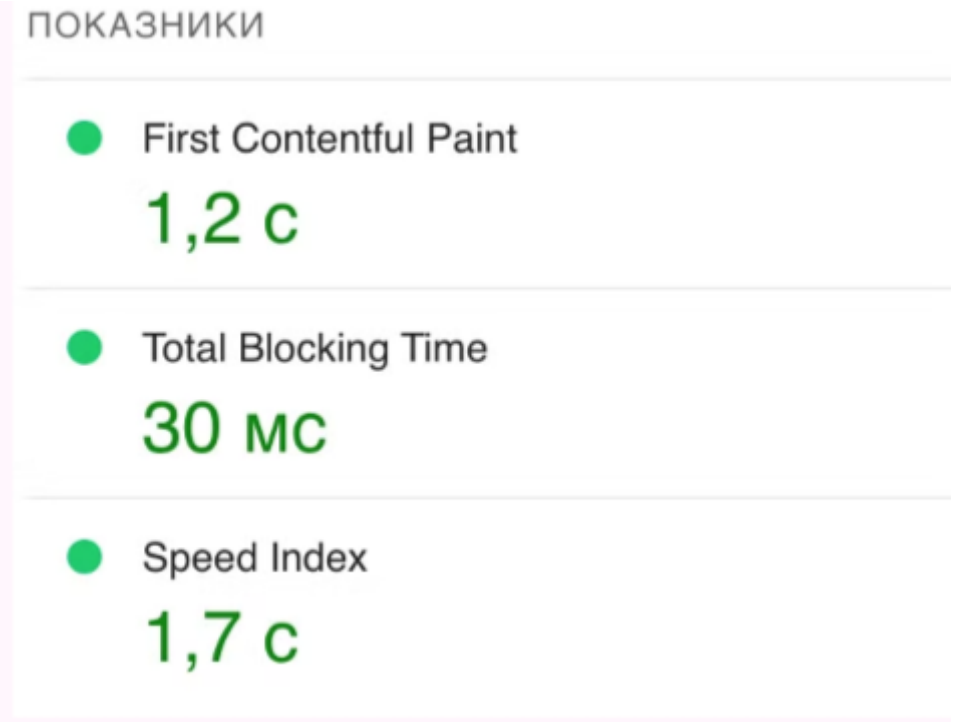

Рисунок 3.18 - Показники завантаження

2. Доступність Оцінка доступності – 91/100. Досягнута висока відповідність стандартам WCAG. Виявлено кілька дрібних недоліків, які можуть бути вирішені шляхом додавання текстових альтернатив до всіх графічних елементів.
3. Оптимізація пошукових систем Оцінка оптимізації для SEO становить 69/100, що є середнім результатом. Рекомендації для покращення: Додавання мета описів до всіх сторінок. Покращення семантичної структури HTML.
4. Продуктивність Аналіз продуктивності через Chrome DevTools виявив: Час завантаження сторінки: 15.6 секунд (включаючи всі ресурси). Основний час витрачається на рендеринг великих зображень і виконання JavaScript-скриптів. Виконання скриптів займає 8688 мс, що вказує на можливість оптимізації коду.

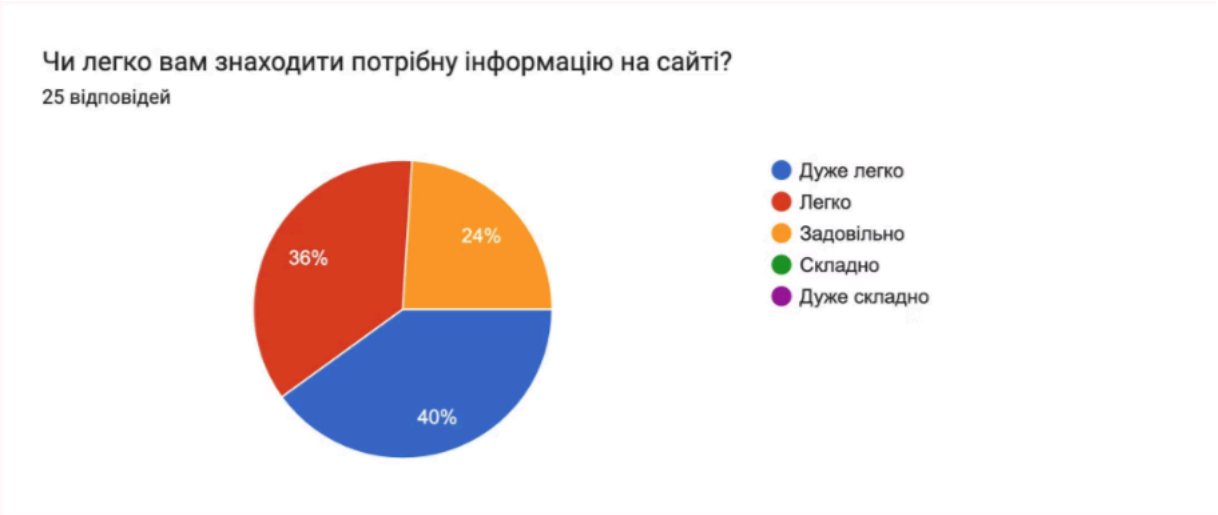

Рисунок 3.19 - Результати опитування

Перша кругова діаграма ілюструє відповіді користувачів на запитання «Чи легко вам знаходити потрібну інформацію на сайті?». Близько 40% респондентів оцінили процес як «дуже легкий», ще 36% – як «легкий», а 24% – як «задовільний». Відсутність негативних оцінок («складно» чи «дуже складно») свідчить про те, що більшість користувачів не відчуває суттєвих труднощів у пошуку необхідних даних. Таким чином, можна зробити висновок, що в цілому система забезпечує достатній рівень інтуїтивності при навігації та пошуку інформації.

Рисунок 3.20 - Результати опитування

Друга діаграма ілюструє оцінку зручності використання інтерактивних елементів (кнопок, меню, форм). Близько 20% опитаних оцінили інтерфейс як

«дуже зручний», а 52% – як «зручний». Ще 28% надали відповідь «задовільно». Жоден з респондентів не вказав на незручність чи серйозні проблеми з керуванням елементами. Така позитивна оцінка взаємодії з системними компонентами свідчить про відносно високу якість ергономіки в частині інтерфейсних рішень.

Отримані результати опитування підтверджують, що ергономічність вебдодатку знаходиться на задовільному або навіть вищому рівні, забезпечуючи ефективний та приємний користувацький досвід при пошуку інформації та роботі з інтерактивними елементами.

Рисунок 3.21 - Тестування на Iphone SE

На наведених рисунках продемонстровано результати перевірки адаптивності та юзабіліті вебзастосунку під час тестування на різних пристроях. Перегляд макету при розширенні екрану в розмірі планшету

(наприклад, iPad Pro з шириною 1024 пікселі) свідчить про збереження логічної структури та читабельності інтерфейсу: бокове меню, заголовки та графічні елементи залишаються чітко впорядкованими, а користувач може зручно орієнтуватися в контенті.

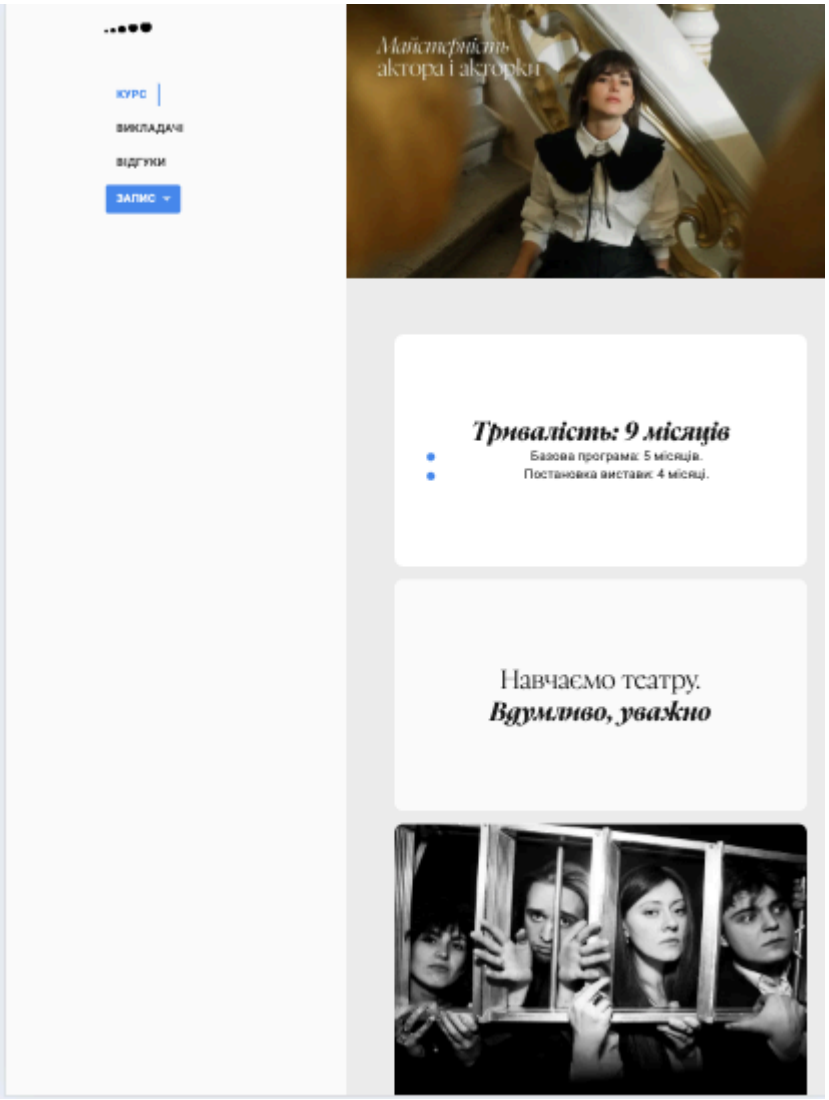

Рисунок 3.22 - Тестування на iPad Pro

На мобільному екрані (iPhone SE), інтерфейс набуває спрощеної форми, зосереджуючись на основних елементах і забезпечуючи швидкий доступ до ключових розділів сайту, таких як «Курс», «Викладачі» та «Відгуки». Даний підхід до адаптивного дизайну дозволяє ефективно застосовувати вебзастосунок до різних розмірів і пропорцій екранів.

ВИСНОВКИ

У результаті виконання магістерської роботи було досягнуто поставленої мети — розроблено інформаційну технологію, яка забезпечує ергономічне проєктування вебдодатків. Було досліджено та виявлено принципи ергономічного проектування. Застосовано та усунено проблеми ергономічного використання вебдодатку. Спроектовано та імплементовано сучасну структуру вебдодатку.

В даній роботі були виконані такі завдання:

1. Проведено аналітичний огляд предметної області.
2. Визначено основні критерії оцінки ергономічності
3. Застосовано принципи ергономічного проектування для розробки вебдодатку.
4. Оцінено рівень ергономічності сайту на основі даних опитування та аналізу існуючих методів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Chmal J., Ptasińska M., Skublewska-Paszkowska M. Analysis of the ergonomics of e-commerce websites // Journal of Computer Sciences Institute. – 2022. – DOI: https://doi.org/10.35784/jcsi.3016.
2. Fabisiak L. Web Service Usability Analysis Based on User Preferences // J. Organ. End User Comput. – 2018. – Vol. 30. – P. 1–13. – DOI: https://doi.org/10.4018/JOEUC.2018100101.
3. Statnik A. The importance of usability in developing websites and mobile apps // Culegere de lucrari stiintifice: Simpozion stiintific al tinerilor cercetatori. – 2023. – Vol. 1. – DOI: https://doi.org/10.53486/9789975359023.39.
4. Hamid S., Bawany N., Zahoor K. Assessing Ecommerce Websites: Usability and Accessibility Study // 2020 International Conference on Advanced Computer Science and Information Systems (ICACSIS). – 2020. – P. 199–204. – DOI: https://doi.org/10.1109/ICACSIS51025.2020.9263162.
5. Gundogan M. Awareness of Ergonomics in User Interface Design of Instructional Web Sites // Proceedings of the Human Factors and Ergonomics Society Annual Meeting. – 2000. – Vol. 44. – P. 481–484. – DOI: https://doi.org/10.1177/154193120004400421.
6. De Menezes M., Falco M. The Relationship of the Studies of Ergonomic and Human Computer Interfaces - A Case Study of Graphical Interfaces in E-Commerce Websites // В кн.: A. Ahram, W. Karwowski, T. Taiar (Eds.). Human Systems Engineering and Design II. IHSED 2019. Advances in Intelligent Systems and Computing, vol. 1026. – Cham: Springer, 2019. – P. 474–484. – DOI: https://doi.org/10.1007/978-3-030-23535-2_35.
7. Paschoarelli L. Ergonomic Design - a Research Line in Human-Technology Interfaces // AHFE International. – 2022. – DOI: https://doi.org/10.54941/ahfe1001306.
8. Chevalier A., Kicka M. Web designers and web users: Influence of the ergonomic quality of the web site on the information search // Int. J. Hum.

Comput. Stud. – 2006. – Vol. 64. – P. 1031–1048. – DOI: https://doi.org/10.1016/j.ijhcs.2006.06.002.

9. James, J. (2002). Usability and usefulness of ergonomics Web sites: a preliminary investigation. SA Journal of Information Management, 4. DOI: https://doi.org/10.4102/SAJIM.V4I1.151.
10. Chevalier, A., & Kicka, M. (2006). Web designers and web users: Influence of the ergonomic quality of the web site on the information search. Int. J. Hum. Comput. Stud., 64, 1031-1048. DOI: https://doi.org/10.1016/j.ijhcs.2006.06.002.
11. Leão C., Silva V., Costa S. Exploring the Intersection of Ergonomics, Design Thinking, and AI/ML in Design Innovation // Applied System Innovation. – 2024. – DOI: https://doi.org/10.3390/asi7040065.
12. Paschoarelli L. Ergonomic Design - a Research Line in Human-Technology Interfaces // AHFE International. – 2022. – DOI: https://doi.org/10.54941/ahfe1001306. (Дубль посилання №7)
13. Peres S., Mehta R., Ritchey P. Assessing ergonomic risks of software: Development of the SEAT // Applied Ergonomics. – 2017. – Vol. 59 Pt A. – P. 377–386. – DOI: https://doi.org/10.1016/j.apergo.2016.09.014.
14. Chen Z. Research on the application of ergonomics in UI interface design // Applied Mathematics and Nonlinear Sciences. – 2023. – Vol. 9. – DOI: https://doi.org/10.2478/amns.2023.2.00787.
15. Jouis C., Orús-Lacort M., Pemberton S. About Ergonomics of Computer Interfaces Designing a System: Designing a System for Human Computational Collective Use // Proceedings of the 14th International Conference on Management of Digital EcoSystems. – 2022. – DOI: https://doi.org/10.1145/3508397.3564823.
16. Fischer G. User Modeling in Human–Computer Interaction // User Modeling and User-Adapted Interaction. – 2001. – Vol. 11. – P. 65–86. – DOI: https://doi.org/10.1023/A:1011145532042.
17. Буров О. Ю. Ергономіка/людський чинник в інформатизації освіти //

Педагогіка і психологія. – 2019. – № 2(103). – С. 30–37.

18. Голобородько В. М. Ергономіка для дизайнерів : підруч. для студ. вищ. навч. закл. дизайнер. спрямування. – Харків : ХДАДМ, 2012. – 378 с.
19. React Documentation. React – офіційна документація [Електронний ресурс]. – Режим доступу: https://reactjs.org/docs/getting-started.html
20. Степаненко Л. Дизайн – синергія мистецтва та науки. – Київ, 2021. – 76 с.
21. Ярцев В. П. Організація баз даних та знань : навчальний посібник. – Київ : ДУТ, 2018. – 214 с.
22. Ситник Н. Організація баз даних. – Київ, 2022. – 38 с. (дата звернення 04.11.2024)
23. Next.js Documentation. Next.js – офіційна документація [Електронний ресурс]. – Режим доступу: https://nextjs.org/docs (дата звернення 04.11.2024)
24. Strapi Documentation. Strapi – офіційна документація [Електронний ресурс]. – Режим доступу: https://docs.strapi.io/ (дата звернення 04.12.2024)
25. Створення Progressive Web Applications (PWA) [Електронний ресурс]. – Режим доступу: https://dou.ua/lenta/articles/progressive-web-applications/ (дата звернення 04.11.2024)
26. Docker Documentation. Docker – офіційна документація [Електронний ресурс]. – Режим доступу: https://docs.docker.com/ (дата звернення 04.11.2024)
27. Richard Willis Blog [Електронний ресурс]. – Режим доступу: https://richardwillis.info/blog/how-i-use-next-js-strapi-and-docker-to-build-my-personal-website(дата звернення 01.11.2024)

ДОДАТОК А

Dockerfile

Dockerfile

Ivan S, 13 months ago | 1 author (Ivan S)

```
1   FROM node:18-alpine3.17
2
3   WORKDIR /app
4
5   COPY package.json .
6
7   RUN npm install
8
9   COPY . .
10
11  EXPOSE 1337
12
13  ENV NODE_ENV=production
14
15  RUN npm run build
16
17  CMD ["npm", "start"]
18
```

```
export async function generateMetadata({ params: { locale } }) {
  unstable_setRequestLocale(locale)
  const queryParams = { locale, populate: 'seo' }

  const ownSeoRes = await fetchFuture({
    queryParams: { ...queryParams, locale },
  })
  const ownSeo = ownSeoRes?.data?.attributes.seo

  const meta = await generateMeta(ownSeo, locale)

  return meta
}

export default function Home({ params: { locale } }) {
  unstable_setRequestLocale(locale)

  return (
    <div className={styles.page}>
      <div className={styles.nav}>
        <Nav />
      </div>
      <FutureList />
    </div>
  )
}
```

```
import { fetchGlobal } from 'cms-api/singles'
import { mergeDeep } from './deepMerge'
import { SharedSeo } from 'types/components'
import { Metadata } from 'next'

export async function generateMeta(
  ownSeo: SharedSeo['attributes'],
  locale: string,
): Promise<Metadata> {
  const queryParams = { locale, populate: 'seo' }

  const globalSeoRes = await fetchGlobal({
    queryParams: { ...queryParams, locale },
  })

  const globalSeo = globalSeoRes?.data?.attributes.seo
  const seo = mergeDeep({}, globalSeo, ownSeo)

  const metaData = {
    title: seo.metaTitle,
    description: seo.metaDescription,
    // metadataBase: new URL(`${process.env.NEXT_PUBLIC_HOST}`), see app/layout.
    alternates: {
      canonical: '/',
      languages: {
        en: '/en',
        uk: '/',
      },
    },
  }
  return metaData
}
```

```
25  const About = async ({ params: { locale } }) => {
26    unstable_setRequestLocale(locale)
27    // @todo: create short populate params
28    const queryParams = { locale, ...populateParams }
29
30    let about
31
32    try {
33      const {
34        data: { attributes },
35      } = await fetchAbout({
36        queryParams: { ...queryParams, locale },
37      })
38      about = attributes
39    } catch {
40      return (
41        <div className={styles.page}>
42          <div className={styles.nav}>
43            <Nav />
44          </div>
45          <div>
46            <h3>No content</h3>
47          </div>
48        </div>
49      )
50    }
51
52    if (about)
53      return (
54        <div className={styles.page}>
55          <div className={styles.nav}>
56            <Nav />
57          </div>
```