МІНІСТЕРСТВО ФІНАНСІВ УКРАЇНИ
ДЕРЖАВНИЙ ПОДАТКОВИЙ УНІВЕРСИТЕТ

ФАКУЛЬТЕТ ФІНАНСІВ ТА ЦИФРОВИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ І СИСТЕМ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: Дослідження ефективності Single Page Application та фреймворку React при розробці вебсайту для дизайну інтер'єрів

Здобувача вищої освіти першого (бакалаврського) рівня,
IV курсу, групи ЦДБ-21-1, спеціальності 122 «Комп'ютерні науки»,
освітня програма «Технології цифрового дизайну»

(підпис) Покиньбороди Володимира Андрійовича
(прізвище, ім'я, по батькові)

Науковий керівник (підпис) Редич Олександр Володимирович, к.е.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Керівник освітньо-професійної програми (підпис) Ратушняк Тетяна Володимирівна к.ф.-м.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Допущено до захисту за рішенням кафедри комп'ютерних та інформаційних технологій і систем протокол від «13» травня 2025 № 18

Завідувач кафедри (підпис) Філоненко Михайло Миколайович, к.ф.-м.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Реєстрація № 78 29.05.2025 (дата) (підпис відповідальної особи)

Результати перевірки до захисту (до захисту, на доопрацювання) 29.05.2025 (дата) (підпис)

Результат захисту 4 (добре) (оцінка) 10.06.2025 (дата) (підпис)

Голова комісії (підпис) В.М. Франчук

Члени комісії:

(підпис) М.М. Філоненко

(підпис) Т.В. Ратушняк

(підпис) В.О. Ніжегородцев

БЛАНК ОЦІНЮВАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Об'єкти контролю	Діапазон оцінки в балах	Фактична кількість балів	Примітки	Дата, П.І.Б.: керівника, голови комісії членів комісії, секретаря, (підписи)
ОЦІНЮВАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ КОМІСІЄЮ ПО ЗАХИСТУ				
1. Зміст випускної роботи (якість теоретичного матеріалу, його зв'язок з практичним, вміння представляти матеріал в логічній послідовності, наявність висновків та пропозицій)	від 0 до 30	25		Редич О.В. Франчук В.М. Філоненко М.М. Ратушняк Т.В. Ніжегородцев В.О. Суховецька Л.П.
2. Дотримання вимог щодо оформлення	від 0 до 10	7		
3. Вчасне виконання роботи на всіх етапах у встановлені кафедрою терміни	від 0 до 5	3		
4. Рецензент	від 0 до 5	4		
РАЗОМ БАЛІВ	від 0 до 50	39		

Оцінювання кваліфікаційної роботи науковим керівником – 36 балів.

ПЕРЕВЕДЕННЯ ОЦІНКИ ЗА 100-БАЛЬНОЮ ШКАЛОЮ ДО НАЦІОНАЛЬНОЇ ШКАЛИ ТА ШКАЛИ ЄКТС

Оцінка за 100-бальною шкалою	Оцінка за шкалоюЄКТС	Оцінка за національною шкалою	Дата	П.І.Б.: голови, членів комісії, секретаря, (підпис)
75	C	4 (добре)	10.06.2025	Франчук В.М. Філоненко М.М. Ратушняк Т.В. Ніжегородцев В.О. Суховецька Л.П.

ПРАВИЛА ПЕРЕВЕДЕННЯ ОЦІНОК ДО НАЦІОНАЛЬНОЇ ШКАЛИ ТА ШКАЛИ ЄКТС

Оцінка за 100-бальною шкалою	Оцінка за шкалою ЄКТС	Оцінка за національною шкалою
90-100	A	5 (відмінно)
80-89	B	4 (добре)
70-79	C	
60-69	D	3 (задовільно)
50-59	E	
35-49	FX	2 (незадовільно)

Факультет фінансів та цифрових технологій
Кафедра комп'ютерних та інформаційних технологій і систем
Галузь знань 12 «Інформаційні технології»
Спеціальність 122 «Комп'ютерні науки»
Освітня програма «Технології цифрового дизайну»

Затверджую
Завідувач кафедри _____ Філоненко М.М.
«14» листопада 2024

Завдання на кваліфікаційну роботу
здобувачу вищої освіти
Покиньбороді Володимиру Андрійовичу

1. Тема кваліфікаційної роботи **"Дослідження ефективності Single Page Application та фреймворку React при розробці вебсайту для дизайну інтер'єрів"**
Затверджена наказом ректора від «13»листопада 2024 № 1717
2. Строк здачі закінченої роботи *«12» травня 2025*
3. Мета роботи: *дослідження переваг та недоліків використання технології Single Page Application (SPA) порівняно з традиційними багатосторінковими додатками (MPA) при створенні веб-сайтів.*
Об'єкт дослідження: *технології SPA та MPA у розробці сайтів дизану інтер'єрів.*
Предмет дослідження: *інструментарій розробки односторінкових веб-застосунків, фреймворк React.*
Орієнтовна база дослідження: *веб-сайти компаній, що спеціалізуються на дизайні інтер'єрів; приклади реалізації SPA і MPA.*
4. Вхідні дані до роботи: *відкриті дані та документація по SPA і MPA, по фреймворку React.*
5. Перелік графічного матеріалу : *архітектурні діаграми SPA та MPA; Скріншоти інтерфейсу веб-застосунку; діаграми взаємодії компонентів React; Графіки продуктивності при навантаженні; порівняльна таблиця характеристик SPA і MPA.*
6. Зміст кваліфікаційної роботи (перелік питань за кожним розділом)
РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ ОДНОСТОРІНКОВИХ ТА БАГАТОСТОРІНКОВИХ ЗАСТОСУНКІВ.
1.1 Огляд систем управління завданнями
1.2. Порівняльний аналіз функціональних можливостей систем управління завданнями
1.3 Проблеми використання стандартних систем у невеликих командах
1.4 Обґрунтування необхідності розробки власної системи
РОЗДІЛ 2. ПРОЄКТУВАННЯ СИСТЕМИ
2.1. Вибір технологій та інструментів для розробки
2.2 Архітектура системи
2.3 Проєктування бази даних та основного функціоналу

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ВЕБ-ДОДАТКУ

3.1 Налаштування середовища розробки та налаштування проєкту

3.2 Реалізація серверної частини

3.3 Реалізація клієнтської частини

3.4 Тестування системи

7. Календарний план виконання роботи

№ п/п	Назва етапів кваліфікаційноїроботи	Строк виконання етапів роботи	
		За планом	Фактично
1	2	3	4
1	Вибір теми, підбір літературних джерел, формування структури кваліфікаційної роботи	До 30.10.2024	«30» 10 2024
2	**І розділ** (розкрити сутність і стан дослідження наукової проблематики роботи, після чого навести її загальну характеристику; огляд літератури за темою і вибір конкретних напрямів досліджень)	25.12.2024	«25» 12 2024
3	**ІІ розділ** (аналіз експериментальної частини про методики досліджень)	15.02.2025	«15» 02 2025
4	**ІІІ розділ** (проведені теоретичні і (або) експериментальні дослідження та проектні рішення)	14.04.2025	«14» 04 2025
5	**Висновки**	01.05.2025	«01» 05 2025
6	Доопрацювання роботи з урахуванням зауважень наукового керівника	09.05.2025	«09» 05 2025
7	Отримання відгуку керівника	03.06.2025	«03» 06 2025
8	Отримання зовнішньої рецензії	05.06.2025	«05» 06 2025

8. Дата видачі завдання «14» листопада 2024 р.

9. Керівник кваліфікаційної роботи _____ Редич О.В.
(підпис)

10. Керівник освітньо-професійної програми _____ Ратушняк Т.В
(підпис)

11. Завдання до виконання отримав, здобувач вищої освіти _____ Покиньборода
(підпис)

АНОТАЦІЯ

Дослідження ефективності Single Page Application та фреймворку React при розробці вебсайту для дизайну інтер'єрів

Метою роботи є дослідження переваг і недоліків використання архітектури односторінкових веб-застосунків (SPA) у порівнянні з багатосторінковими застосунками (MPA) на прикладі розробки вебсайту для компаній, що спеціалізуються на дизайні інтер'єрів, із застосуванням фреймворку React.

Основними завданнями роботи є: аналіз особливостей SPA і MPA; оцінка продуктивності, гнучкості та зручності користування цими архітектурами; розгляд практичних прикладів веб-реалізацій; проєктування та розробка власного веб-застосунку; тестування та порівняння ефективності підходів.

Предмет дослідження – інструменти та методики створення SPA-застосунків, зокрема фреймворк React.

Об’єкт дослідження – веб-технології SPA і MPA в контексті реалізації сайтів для індустрії дизайну інтер'єрів.

Базою дослідження є відкриті джерела, документація React, приклади комерційних сайтів, архітектурні рішення популярних вебзастосунків.

У результаті дослідження виявлено ключові відмінності SPA та MPA, обґрунтовано доцільність використання React у конкретних умовах, реалізовано вебсайт з використанням SPA-архітектури, проведено аналіз продуктивності.

Отримані результати можуть бути використані для прийняття технічних рішень при створенні сучасних інтерфейсів користувача, оптимізації веброзробки та виборі відповідної архітектури вебзастосунку.

Кваліфікаційна робота містить 53 сторінок, 8 рисунків, 5 таблиць, список використаних джерел – 40 найменувань.

Ключові слова: Single Page Application, MPA, React, веброзробка, архітектура, інтерфейс, продуктивність.

Рік виконання кваліфікаційної роботи: 2025.

Рік захисту: 2025.

ABSTRACT

Research on the Efficiency of Single Page Applications and the React Framework in Developing a Website for Interior Design

The aim of the thesis is to explore the advantages and disadvantages of using Single Page Application (SPA) architecture compared to Multi Page Applications (MPA), based on the development of a website for companies specializing in interior design using the React framework.

The main objectives of the research include: analyzing the characteristics of SPA and MPA; evaluating performance, flexibility, and user experience of these architectures; reviewing practical examples of web implementations; designing and developing a custom web application; testing and comparing the efficiency of the approaches.

The subject of the study is the tools and methodologies for creating SPA applications, particularly the React framework.

The object of the study is SPA and MPA web technologies in the context of implementing websites for the interior design industry.

The research is based on open sources, React documentation, examples of commercial websites, and architectural solutions of popular web applications.

As a result, key differences between SPA and MPA were identified, the feasibility of using React in specific conditions was justified, a website using SPA architecture was implemented, and its performance was analyzed.

The findings can be used for making technical decisions when creating modern user interfaces, optimizing web development, and choosing the appropriate web application architecture.

The qualification paper contains 53 pages, 8 figures, 5 tables, and 40 references.

Keywords: Single Page Application, MPA, React, web development, architecture, interface, performance.
Year of completion: 2025.
Year of defense: 2025.

ЗМІСТ

ЗМІСТ.....2
ВСТУП.....3
РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ ОДНОСТОРІНКОВИХ ТА БАГАТОСТОРІНКОВИХ ЗАСТОСУНКІВ.....6
1.1 Історія та розвиток SPA.....6
1.2 Фреймворки та бібліотеки, різниця та значення в розробці веб застосунків14
1.3 Компонентна розробка SPA додатків у React.....16
РОЗДІЛ 2. МЕТОДИ ТА АЛГОРИТМИ РОЗРОБКИ SPA З ВИКОРИСТАННЯМ REACT ТА АРХІТЕКТУРА ТА ПРИНЦИПИ.....21
2.1 Основні принципи розробки односторінкових застосунків(SPA).....21
2.2 Архітектура та принципи React.....23
2.3 Тестування.....30
РОЗДІЛ 3. РЕАЛІЗАЦІЯ ІНТЕРФЕЙСУ СТУДІЇ ДИЗАЙНУ ЯК SPA НА REACT.....33
3.1 Конструювання прототипу вебсайту “ LuxDesign”.....33
3.2 Тестування.....42
3.3 Методологія дослідження.....43
3.4 Результати.....45
Висновок.....47
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....49
ДОДАТКИ

ВСТУП

Актуальність розробки. В сьогоденні інтернет-технології розвиваються надзвичайно швидкими темпами, всі сфер нашого життя стають на шлях цифрової трансформації, зокрема бізнес часто переходить з аналогових ресурсів до цифрових, вирішення проблем що постають в цьому світлі веб-застосунки.

Розвиток веб-технологій зумовив потребу у гнучких, масштабованих і зручних інтерфейсах. Односторінкові застосунки (SPA) стали популярним рішенням, що дозволяє підвищити швидкість завантаження, покращити користувацький досвід і знизити навантаження на сервер. Успішна реалізація SPA проєктів значною мірою залежить від правильного вибору інструментів розробки.

SPA технології досліджуються як у вітчизняній, так і в зарубіжній літературі. Відомі реалізації таких систем охоплюють як комерційні, так і державні інформаційні ресурси. Проте питання вибору оптимального фреймворку для реалізації SPA-проєктів у певних бізнес-сферах, таких як інтер’єрний дизайн, досі не достатньо вивчені. У сучасному світі веб- технології стали невід'ємною частиною бізнесу. Вони дозволяють компаніям будь- якого розміру та галузі досягти успіху, використовуючи такі можливості:

1. Залучення нових клієнтів. Інтернет дозволяє компаніям налагодити контакт з потенційними клієнтами в будь-якому куточку світу. Це можна зробити за допомогою веб-сайтів.
2. Продаж більшої кількості товарів та послуг. Веб-застосунки, електронна комерція та інші веб-технології дозволяють компаніям продавати свої товари та послуги більш ефективно. Вони забезпечують зручний доступ до інформації про продукти та послуги, а також можливість здійснювати покупки 24/7
3. Покращення досвіду клієнтів. Веб-технології дозволяють компаніям

надавати клієнтам підтримку 24/7. Це можна зробити за допомогою онлайн-чату, веб-форм, бази знань та інших веб-технологій.

4. Оптимізації бізнес-процесів. Веб-технології можуть автоматизувати багато завдань, що дозволяє бізнесу працювати більш ефективно. Це включає такі завдання, як управління запасами, бухгалтерський облік та управління персоналом.

Доцільність розробки та дослідження односторінкових застосунків є безперечно важливою в ері цифрових технологій.

В контексті розробки веб-застосунків важливу роль грає підбір відповідних технологій до задач котрі ставить той чи інший проєкт. Серед лідерів технологій фронтенд розробки знаходяться React та Vue.js, вони мають доволі суттєві відмінності, тому слід зважати на архітектуру і задачі проекту на етапі вибору технології.

Вибір правильних інструментів розробки може мати значний вплив на успіх веб застосунку. Правильні інструменти можуть допомогти розробникам працювати швидше та ефективніше, створювати більш інтуїтивні та зручні для користувачів застосунки та знизити витрати на розробку та підтримку.

Метою роботи є дослідження переваг та недоліків використання технології Single Page Application (SPA) порівняно з традиційними багатосторінковими додатками (MPA) при створенні веб-сайтів для компаній, що займаються дизайном інтер'єрів.

Для досягнення поставленої мети необхідно вирішити такі завдання:

1. Провести огляд сучасних веб-технологій у сфері розробки SPA та MPA.
2. Дослідити архітектурні особливості фреймворку React
3. Розробити та порівняти функціональні характеристики SPA і MPA у контексті бізнес-потреб.

4. Оцінити вплив технологій на ефективність, масштабованість і користувацький досвід.

Об’єктом дослідження є інформаційне середовище сучасної веб-розробки, зокрема процес створення інтерактивного веб-застосунку із використанням односторінкової архітектури. Це загальна IT-технологія, яка лежить в основі проєктування динамічних інтерфейсів.

Предметом дослідження виступають фреймворк React та технологічні засоби його використання при реалізації SPA — компоненти, бібліотеки, шаблони маршрутизації, підходи до керування станом застосунку. Предмет охоплює всі технічні аспекти реалізації SPA, що забезпечують досягнення поставленої мети.

Наукова новизна полягає у комплексному порівнянні SPA і MPA в прикладному контексті галузі дизайну інтер’єрів, а також у виявленні переваг застосування React у якості основного інструменту розробки, з акцентом на швидкодію, модульність і масштабованість інтерфейсу.

Результати дослідження можуть бути використані веб-розробниками для вибору технологічної платформи, а також компаніями малого і середнього бізнесу для створення зручних, ефективних і естетичних веб-сайтів. Отримані рекомендації дозволяють скоротити витрати на розробку, покращити користувацький досвід і забезпечити технічну підтримку цифрових продуктів.

РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ ОДНОСТОРІНКОВИХ ТА БАГАТОСТОРІНКОВИХ ЗАСТОСУНКІВ

1.1. Історія та розвиток SPA

Односторінкові веб-додатки - це сучасний спосіб створення веб-додатків, який забезпечує кращий досвід для користувачів. SPA робить веб-застосунки більш ефективними для бізнесу, владних структур та соціальних проектів, які потребують швидкої та плавної взаємодії з користувачем.[1]

У сучасному світі, де швидкість і зручність – ключові показники, SPA стали популярними серед бізнесу. Вони дозволяють створювати враження безперервності для користувачів, забезпечуючи більш гладку інтеракцію та швидше завантаження контенту. Додатки такого типу чудово вирішують завдання підвищення залучення користувачів, оптимізації роботи та покращення користувацького досвіду.

Україна активно використовує сучасні технології, щоб полегшити доступ до державних послуг громадянам. Нове покоління державних структур впроваджує SPA-додатки, які спрощують процедури та забезпечують швидкий доступ до різноманітних адміністративних послуг. Наприклад, Державна міграційна служба запровадила додаток "Дія", який дозволяє громадянам отримувати реєстрацію місця проживання, отримувати закордонні паспорти та посвідчення водія. Міністерство цифрової трансформації пропонує додаток "Дія.Підтримка", що надає фінансову допомогу тим, хто постраждав від військової агресії росії. Державна Фіскальна Служба України впровадила веб-додаток "Електронний кабінет платника податків" з метою спрощення взаємодії між платниками податків та службою. Цей інтерфейс дозволяє платникам податків здійснювати електронні операції, такі як подання звітності, сплата податків, перевірка статусу своєї податкової справи тощо. [2]

Використання цього додатку дозволяє забезпечити більш зручну та

ефективну взаємодію між платниками податків та податковим органом. Міністерство охорони здоров'я України розробило веб-додаток "Helsi.me", який створено з метою полегшення доступу пацієнтів до медичних послуг. Цей додаток дозволяє пацієнтам реєструватися на прийом до лікаря онлайн, отримувати електронні рецепти та медичні довідки. Крім цього, "Helsi.me" сприяє зручності спілкування між лікарем та пацієнтом, полегшуючи доступ до медичної інформації та послуг за допомогою електронних ресурсів. Це сприяє оптимізації медичної допомоги та забезпечує більш ефективне використання ресурсів системи охорони здоров'я.

Використання SPA-додатків має свої переваги для державних структур в Україні. Вони забезпечують покращений досвід користувача, дозволяючи безперервну та плавну взаємодію, цілодобову підтримку. Все це дозволяє державі надавати громадянам сучасні та якісні послуги.

SPA стали популярним рішенням для бізнесу завдяки низці переваг, які вони пропонують. Спроектовані з урахуванням користувацької зручності, вони забезпечують безперервну взаємодію без перезавантаження сторінок. Це робить взаємодію з додатком інтуїтивно зрозумілою та комфортною. Односторінкові додатки дозволяють зберігати продуктивність користувача, оскільки не потребують постійного перезавантаження. Це особливо важливо для мобільних користувачів, які цінують швидкість та зручність у використанні. І, крім того, вони забезпечують безпечне зберігання даних, маючи можливість зберігати їх в браузері, покращуючи безпеку та швидкість доступу.

Загалом, односторінкові додатки стають потужним інструментом для бізнесу, сприяючи покращенню досвіду користувача, підвищенню продуктивності та забезпеченню безпеки даних.

Односторінкові застосунки (Single Page Applications, SPA) є відносно новою парадигмою в історії веброзробки, яка виникла у відповідь на зростаючі вимоги

до інтерактивності та швидкодії вебсайтів. Початково, вебдодатки будувалися за принципом багатосторінкових застосунків (MPA), де кожна дія користувача призводила до повного перезавантаження сторінки з сервера. Проте з розвитком JavaScript та AJAX стала необхідність у більш гнучких підходах, які забезпечують більш плавний та швидкий користувацький досвід.[3]

Перші спроби створити більш інтерактивні вебдодатки включали використання окремих AJAX-запитів для оновлення певних частин сторінки без повного перезавантаження. Однак цей підхід мав певні обмеження, пов'язані зі складністю керування станом та навігацією у великих застосунках[4].

Початок цього руху тісно пов'язаний з розвитком потужних JavaScript-бібліотек та фреймворків, таких як AngularJS (представлений Google у 2009 році), Backbone.js (2010) та Ember.js (2011). Ці інструменти запропонували нові підходи до організації клієнтського коду, використовуючи такі концепції, як Model-View-Controller (MVC) або Model-View-ViewModel (MVVM), що дозволило розробникам створювати складні інтерактивні інтерфейси, які динамічно оновлювали вміст без необхідності повного перезавантаження сторінки [5].

Основні характеристики ранніх SPA включали:

- Завантаження єдиної HTML-сторінки: При першому запиті браузер отримував лише одну HTML-сторінку, а подальша навігація та оновлення контенту здійснювалися за допомогою JavaScript.
- Асинхронне отримання даних: Використання AJAX для обміну даними з сервером у фоновому режимі, що дозволяло оновлювати лише необхідні частини інтерфейсу.
- Маніпуляція DOM: JavaScript використовувався для динамічної зміни Document Object Model (DOM), відображаючи нові дані та змінюючи вміст сторінки без її повного перезавантаження.

SPA базується на фундаментальних принципах, які визначають архітектуру, функціональність та користувацький досвід. Розгляньмо основні принципи кожного підходу детальніше.[6]

Основна ідея SPA полягає в завантаженні єдиної HTML-сторінки, яка потім динамічно оновлюється за допомогою JavaScript у відповідь на дії користувача. Ключові принципи SPA включають:

- Єдина HTML-сторінка: Весь інтерфейс застосунку розміщується на одній HTML-сторінці. Подальші "переходи" між розділами відбуваються без повного перезавантаження сторінки.
- Динамічне оновлення контенту: JavaScript відповідає за асинхронне отримання даних з сервера (зазвичай у форматі JSON або XML) та оновлення DOM (Document Object Model) без перезавантаження сторінки.
- Клієнтська маршрутизація: Переходи між різними "сторінками" або розділами застосунку обробляються на клієнтській стороні за допомогою JavaScript-маршрутизатора. Це забезпечує швидку навігацію без запитів до сервера на кожну зміну відображення.
- Розділення front-end та back-end: SPA зазвичай передбачають чітке розділення між клієнтською частиною (front-end), яка відповідає за відображення та взаємодію з користувачем, та серверною частиною (back-end), яка надає API для отримання та збереження даних.
- Багатий користувацький інтерфейс: SPA дозволяють створювати складні та інтерактивні інтерфейси з плавними анімаціями та переходами, що наближає досвід користування до нативних мобільних застосунків.

Підходи до вебдизайну, такі як "Mobile First", адаптивний, гнучкий та флюїдний дизайн, є важливими незалежно від того, чи розробляється SPA, чи MPA. Однак спосіб їхньої реалізації та деякі нюанси можуть відрізнятися в залежності від архітектури застосунку.[7]

Підхід "Mobile First" є актуальним як для SPA, так і для MPA. Він передбачає розробку дизайну спочатку для мобільних пристроїв з обмеженими ресурсами та поступове розширення функціональності та стилів для більших екранів.[8]

- У SPA: "Mobile First" може впливати на спосіб структурування компонентів, оптимізацію завантаження ресурсів (наприклад, завантаження лише необхідних компонентів для мобільних пристроїв) та вибір технологій для забезпечення високої продуктивності на слабких пристроях.
- У MPA: "Mobile First" може визначати базову структуру HTML, CSS та JavaScript, які потім адаптуються за допомогою медійних запитів для більших екранів. Серверна частина також може бути оптимізована для швидкої віддачі контенту на мобільні пристрої.

Адаптивний дизайн, що передбачає використання кількох фіксованих макетів для різних діапазонів розмірів екранів, також може застосовуватися в обох архітектурах. [9]

- У SPA: Адаптивний дизайн може бути реалізований через використання медійних запитів у CSS для зміни стилів та компонентів залежно від розміру екрана. JavaScript також може динамічно змінювати відображення компонентів.
- У MPA: Адаптивний дизайн традиційно реалізується за допомогою медійних запитів у CSS, які застосовуються до різних HTML-шаблонів або окремих CSS-файлів, що завантажуються залежно від розміру екрана.

Гнучкий дизайн, що використовує гнучкі сітки, гнучкі зображення та медійні запити для створення єдиного макета, який адаптується до будь-якого розміру екрана, є дуже поширеним підходом як для SPA, так і для MPA. [10,11]

- У SPA: Гнучкі макети на основі CSS Flexbox та Grid є основою для створення адаптивних компонентів. Гнучкі зображення та медійні запити

використовуються для оптимізації відображення контенту на різних пристроях.

- У MPA: Гнучкий дизайн є стандартною практикою, що дозволяє одному набору HTML, CSS та JavaScript ефективно відображатися на різних пристроях.

Флюїдний дизайн, що використовує відсоткові одиниці вимірювання для всіх розмірів елементів, може бути використаний в обох типах застосунків, хоча й рідше в чистому вигляді через складнощі з контролем компонування на дуже великих або малих екранах.[12]

- У SPA: Флюїдні елементи можуть бути частиною загального гнучкого макета, забезпечуючи їхню пропорційну зміну розмірів.
- У MPA: Флюїдні макети можуть використовуватися для створення базової структури сторінки, яка потім доповнюється гнучкими або адаптивними елементами1.4 Порівняння підходів у контексті SPA та MPA

Вибір між SPA та MPA впливає на спосіб реалізації вебдизайну та оптимізації для різних пристроїв, відмінності подано в табл. 1.1.

Табл. 1.1

Характеристика	Single Page Application (SPA)	Multi Page Application (MPA)
Початкове завантаження	Завантажується одна HTML-сторінка та основні ресурси. Подальші "переходи" швидкі.	Кожна сторінка завантажується повністю з сервера, що може призводити до повільнішого початкового завантаження для складних сайтів.
Швидкість навігації	Швидка навігація між "сторінками" за рахунок динамічного оновлення контенту на клієнті.	Кожен перехід вимагає запиту до сервера та повного перезавантаження сторінки, що може бути повільнішим.
Серверне навантаження	Менше навантаження на сервер при навігації, оскільки сервер в основному віддає дані (API).	Вище навантаження на сервер при кожному запиті сторінки.
Клієнтське навантаження	Більше навантаження на клієнтський браузер за рахунок обробки JavaScript та рендерингу DOM.	Менше навантаження на клієнт при кожному окремому запиті, але повне перезавантаження може бути ресурсомістким.
SEO (пошукова	Традиційно складніше для SEO, але сучасні техніки, такі	Краща підтримка SEO "з коробки" завдяки окремим

оптимізація)	як SSR та пререндеринг, покращують індексацію.	URL для кожної сторінки.
Розробка	Вимагає більш складної архітектури на клієнті та використання JavaScript-фреймворків. Розділення front-end та back-end є обов'язковим.	Може бути простішим для невеликих проєктів, але підтримка великих складних MPA може стати складною.
Адаптивність	Адаптивність досягається в основному за рахунок CSS та JavaScript на клієнті. Оптимізація продуктивності на мобільних пристроях є важливим аспектом.	Адаптивність також досягається за рахунок CSS, але сервер може віддавати різні HTML або оптимізовані ресурси залежно від пристрою.
Користувацький досвід	Може забезпечити більш плавний та інтерактивний користувацький досвід, схожий на нативні застосунки.	Може бути менш плавним через повні перезавантаження сторінок, особливо на повільних з'єднаннях.

SEO: SPA складно індексуються пошуковими системами через їх динамічний вміст, що ускладнює виявлення цих додатків в результатах пошуку.

SEO (Search Engine Optimization) - це комплекс заходів, спрямованих на підвищення видимості веб-сайту в результатах пошукових систем, таких як Google, Bing чи Yahoo. Оптимізація SEO включає в себе стратегії, які роблять сайт більш привабливим та доступним для пошукових систем, серед найпопулярніших стратегій є зокрема такі: ключові слова та контент, оптимізація метатегів, технічна оптимізація, будівництво посилань, соціальні медіа та публікації, аналітика та відстеження, локальний SEO. [13]

Коли мова йде про односторінковий додаток, важливо зазначити, що вони часто відрізняються від традиційних веб-сайтів за своєю структурою. SPA використовують динамічне оновлення контенту без перезавантаження сторінки, що ускладнює їхню індексацію для пошукових систем. Це означає, що пошукові системи можуть мати проблеми з адекватним індексуванням контенту SPA, а це в свою чергу може призвести до поганої видимості цих сайтів у результатах пошуку. Тобто, якщо веб-сайт не індексується правильно, його може бути складно знайти під час пошуку. Це важливий аспект для власників сайтів, оскільки хороша видимість у пошукових

системах може значно вплинути на кількість відвідувачів та трафік на сайті, а отже, і на успіх бізнесу чи проекту.

Тому розробники SPA часто вивчають методи оптимізації, щоб забезпечити кращу індексацію своїх додатків пошуковими системами. До основних методів оптимізації відносять:

1. Структура URL-адрес: забезпечення дружніх для SEO URL- адрес. Це допомагає пошуковим системам краще індексувати сторінки додатку.
2. Метатеги та метаописи: використання ключових слів у метатегах сторінок для підвищення їх ранжування у пошуку.
3. Контент та ключові слова: розміщення унікального, релевантного контенту з ключовими словами, які відповідають запитам користувачів.
4. Оптимізація завантаження сторінок: покращення швидкості завантаження сторінок SPA, що важливо для користувачів та ранжування в пошуку.

Наступна типова проблема SPA – складність розробки. Розробка односторінкових додатків вимагає глибшого розуміння JavaScript та веб-технологій, ніж у випадку з традиційними багатосторінковими додатками. Це означає, що розробники повинні мати більше експертизи для створення складних SPA.

Питання сумісності також важливе. SPA можуть працювати не в усіх версіях браузерів, особливо в застарілих або несумісних. Це може створити проблеми для користувачів, які використовують такі браузери, оскільки додаток може працювати неправильно або навіть не запускатися.

Щодо безпеки, через те, що SPA зберігають дані в пам'яті браузера, вони можуть стати вразливими до зовнішніх атак. Це означає, що вони можуть бути більш доступні для хакерських атак.

1.2. Фреймворки та бібліотеки, різниця та значення в розробці веб застосунків

Фреймворки та бібліотеки - це два ключові інструменти для розробки веб- додатків. Вони забезпечують розробникам готові набори функцій та інструментів, які спрощують процес створення складних веб-додатків.[14]

Фреймворки - це великі конструкції, що надають базову основу для розробки веб-додатків. Вони включають різні функції, такі як маршрутизація, управління станом, шаблони та компоненти. Фреймворки спрощують життя розробників, дозволяючи використовувати готові інструменти та методи.[15]

Бібліотеки - це менші набори функцій, які можна використовувати для конкретних завдань у веб-додатках. Вони включають в себе функції для візуальних ефектів, обробки даних та зв'язку з базами даних. Використання бібліотек дозволяє розробникам додавати функціональність без написання коду з нуля.

Фреймворки зазвичай стабільніші та масштабніші, проте передають більше відповідальності розробникам. Бібліотеки, незважаючи на менший обсяг функцій, можуть бути менш складними у використанні [16].

Обидва ці інструменти мають значний вплив на розробку веб-додатків, допомагаючи скоротити час розробки, покращити якість та зробити додатки більш масштабованими. Обираючи між фреймворком і бібліотекою, розробники повинні врахувати вимоги проекту та їхні власні потреби.

React - це фреймоврк JavaScript, яка надає готові рішення для створення SPA додатків. За допомогою React можна швидко розробляти, розширяти, запускати та тестувати SPA різної складності. [17]

Під час використання цієї бібліотеки розробникам не потрібно приймати багато дизайнерських рішень, оскільки React вже надає кілька розумних рішень. Це дозволяє розробникам зосереджуватися на написанні ефективної програмної логіки, оскільки багато речей вже вирішено в рамках фреймворку. Однак, для досвідчених розробників, які працюють над великими кодовими базами,

існують певні недоліки використання фреймворків.

Фреймворки часто не дозволяють достатню гнучкість, оскільки вимагають слідування певному стилю коду. Вони можуть боротися з розробником, якщо той вирішить відхилитися від стандартних підходів. Крім того, фреймворки зазвичай мають великий обсяг і включають в себе багато функцій, навіть якщо потрібно використовувати лише частину з них.

React надає розробникам потужний інструментарій для створення інтерфейсів, що визначаються декларативно та для ефективного управління їх станом. Це робить процес розробки односторінкових додатків (SPA) більш простим та зрозумілим. У порівнянні з HTML-шаблонами, які розглядаються як рядок, React опрацьовується як дерево об'єктів, що дозволяє ефективно організувати та керувати компонентами.

JSX, як розширення синтаксису JavaScript, візуально асоціюється з XML та надає можливість трансформувати його в React-код. Незважаючи на те, що JSX на перший погляд може здатися мовою шаблонів, це насправді розширення JavaScript. Воно дозволяє представляти дерево об'єктів React, використовуючи синтаксис, що нагадує HTML.[18]

Його синтаксис, полегшує читання та написання коду, забезпечуючи при цьому всі переваги JavaScript. Важливо підкреслити, що і сам React, і браузер не обов'язково працюють з JSX, оскільки воно обробляється на етапі компіляції. Код, який потрапляє в браузер, вже не містить JSX і залишається чистим від будь-яких шаблонів.

Компонент у React діє як шаблон, план чи глобальне визначення, і може бути представлений як функцією або класом. Елемент React, який представляє об'єкт, повертається компонентами, практично описує вузли DOM. Групування логіки та розділення на компоненти роблять код більш зручним для управління та тестування.

Однією з найважливіших переваг React є можливість легко створювати односторінкові додатки (SPA), які автоматично оновлюються у реальному часі, без очікування відповіді від сервера. Розподіл коду на компоненти сприяє

ефективному управлінню та масштабуванню додатків.

Структура React-додатку дуже зручна та легко зрозуміла, а використання JSX у шаблонах компонентів додає гнучкості та зрозумілості коду роблячийого легко перевикористовуваним у будь-якому розділі додатку.

1.3. Компонентна розробка SPA додатків у React

Сучасні веб-додатки мають таку ж складність, як і будь-які інші програми, і часто розробляються командами розробників, які спільно працюють над фінальним продуктом. У таких умовах для підвищення ефективності природно використовувати стратегії розподілу роботи з мінімальними перетинами між розробниками та підсистемами. Компонентний підхід зазвичай використовується для досягнення цієї мети.

SPA, або односторінковий додаток, представляє собою веб-програму, яка завантажує лише один документ, а потім динамічно оновлює його вміст за допомогою JavaScript API, таких як XMLHttpRequest або Fetch. Це відбувається без повторного завантаження всієї сторінки, щоб показати новий вміст, що сприяє більш плавному та динамічному користувацькому досвіду.

Основна мета веб-компонентів - зменшення складності за рахунок ізоляції пов'язаних груп коду на HTML, SCSS та JavaScript для виконання загальної функціональності в межах контексту однієї сторінки. Це все добре реалізовано у React бо там фактично об’єднані HTML та JavaScript у форматі JSX.[19]

JSX – це розширення JavaScript дозволяє писати код, який нагадує HTML, прямо всередині JavaScript-файлів. JSX спрощує створення інтерфейсів, адже його синтаксис подібний до HTML, і в ньому можна вставляти вирази JavaScript для динамічного визначення вмісту. Різницю між методами відображення HTML використовуючи JSX можна подивитись на рис. 1.1.



Рис. 1.1 – Приклад відображення HTML у компоненті JSX

Необхідно підкреслити, що JSX, який використовується в React, є розширенням синтаксису JavaScript і вимагає попередньої компіляції до стандартного JavaScript перед тим, як може бути використаний в браузері. Цей процес зазвичай виконується за допомогою інструментів, таких як Babel, що конвертують JSX в еквівалентний код JavaScript.

У React, компоненти є основною будівельною одиницею інтерфейсу. Вони допомагають структурувати додаток, розбиваючи його на невеликі та самостійні частини. Компоненти можна розглядати як вузли, які спільно формують структуру додатку, полегшуючи його розробку та підтримку.[20]

У фреймворку React існують два основних типи компонентів: функціональні та класові. Функціональні компоненти є простими функціями, які приймають вхідні дані (props) і повертають елемент React. З іншого боку, класові компоненти є класами, що розширюють React.Component і можуть мати внутрішній стан.

Властивості (props) використовуються для передачі даних в компоненти, забезпечуючи їх конфігурованістю та можливістю використання в різних контекстах. Класові компоненти також можуть утримувати внутрішній стан, який дозволяє зберігати та оновлювати дані в межах компонента.

Метод render у React компонентах визначає, як саме компонент повинен відображатися на екрані. Класові компоненти також мають методи життєвого циклу, такі як componentDidMount і componentDidUpdate, які дозволяють виконати код на різних етапах життєвого циклу компонента.

Важливою особливістю React є можливість вкладення компонентів один в одного, утворюючи ієрархію. Це підтримує повторне використання та полегшує управління кодом, сприяючи читабельності та скальованості додатку.

У контексті SPA додатків, всі функціональні можливості об'єднуються в єдину веб-сторінку. Цей підхід дозволяє витягти рівень представлення за межі сервера, надаючи браузеру контроль над ним. Кожен запит на отримання контенту, такого як HTML-сторінка, подається до сервера, який відповідає на нього, взаємодіючи з браузером у двох напрямках.[21]

Коли на клієнтській стороні виникає потреба у свіжих даних, відправляється запит на сервер. На стороні сервера цей запит обробляється об'єктом контролера в межах представницького рівня. Контролер взаємодіє з модельним рівнем через сервісний шар, що визначає необхідні компоненти
для виконання завдання на рівні моделі даних. Після отримання необхідних даних від об'єкта доступу до даних або сервісного агенту, будь-які зміни вносяться в дані за допомогою бізнес-логіки на рівні бізнес-моделі.

Такий підхід розподіленої архітектури дозволяє підтримувати консистентність та ефективно керувати додатком, зменшуючи необхідність повторних завантажень сторінок і забезпечуючи динамічний та відзивчивий користувацький інтерфейс Приклад різниці між SPA та звичайним Web сайтом наведено на рис. 1.2.



Рис. 1.2 – Різниця між SPA та звичайним Web сайтом

Керування передається назад на рівень представлення, де обирається відповідне представлення. Логіка представлення диктує, як щойно отримані дані будуть представлені у вибраному поданні. Часто результуюче подання починається як вихідний файл із заповнювачами, куди потрібно вставити дані. Цей файл діє як своєрідний шаблон того, як буде виглядати контент, коли контролер надсилає до нього запит.

Перенесення процесу створення та керування поданнями в інтерфейс користувача відокремлює його від сервера. З архітектурної точки зору, це дає SPA перевагу. Якщо ви не виконуєте частковий рендеринг сторінки на сервері, сервер більше не повинен брати участь у представленні даних.

Оскільки логіка представлення здебільшого знаходиться на стороні клієнта в SPA, завдання поєднання HTML і даних переноситься з сервера в браузер. Як і на стороні сервера, вихідний HTML містить заповнювачі, куди потрібно вставити дані.[22] Цей шаблон на стороні клієнта використовується як основа для штампування нових представлень у клієнті. Однак це не шаблон HTML для всієї сторінки. Він призначений лише для частини сторінки, яку представляє подання.

Все це відбувається без необхідності оновлювати оболонку. Таким чином, замість того, щоб отримувати нову статичну сторінку для кожного навігаційного запиту, SPA може відображати новий вміст без перешкод для користувача. Для певної частини екрана вміст одного подання просто замінюється вмістом іншого подання. Це створює ілюзію, що сама сторінка змінюється під час навігації користувача. Навігація без перезавантаження є ключовою особливістю SPA додатку.

З часом концепція SPA отримала значний розвиток та широке визнання у світовій спільноті веброзробників. З'явилися нові, більш потужні та гнучкі JavaScript-фреймворки та бібліотеки, такі як React (розроблений Facebook,

представлений у 2013 році) та Vue.js (створений Еваном Ю, представлений у 2014 році). Ці інструменти запропонували покращені підходи до керування станом, компонентної архітектури та віртуального DOM, що ще більше спростило розробку складних та продуктивних SPA.[23]

Наприклад, такі інструменти, як Virtual DOM у React та Vue.js, значно покращили продуктивність оновлення інтерфейсу. Замість безпосередньої маніпуляції реальним DOM браузера, ці фреймворки спочатку оновлюють віртуальне представлення DOM, порівнюють зміни та потім ефективно застосовують лише необхідні оновлення до реального DOM [24, 25].

Клієнтська маршрутизація стала ключовою частиною архітектури SPA, дозволяючи керувати URL-адресами та історією браузера без перезавантаження сторінки. Такі бібліотеки, як React Router та Vue Router, надають потужні інструменти для створення складних навігаційних структур у SPA.

Подальше дослідження в області SPA зосереджується на поліпшенні продуктивності (особливо часу першого завантаження), SEO-оптимізації (за допомогою Server-Side Rendering та Pre-rendering), керування складним станом застосунку та покращенні інструментів розробки.[26]

Наприклад, Server-Side Rendering (SSR) передбачає рендеринг початкового HTML на сервері та відправку вже готової сторінки браузеру, що покращує час першого завантаження та індексацію пошуковими системами [27]. Pre-rendering передбачає генерацію статичних HTML-файлів для певних маршрутів під час збірки проєкту, що також сприяє кращій SEO та швидшому відображенню початкового контенту [28].В результаті, односторінкові застосунки продовжують еволюціонувати, забезпечуючи користувачам більш швидкий, плавний та інтерактивний досвід взаємодії з вебдодатками незалежно від пристрою, яким вони користуються.

РОЗДІЛ 2. МЕТОДИ ТА АЛГОРИТМИ РОЗРОБКИ SPA З ВИКОРИСТАННЯМ REACT ТА АРХІТЕКТУРА ТА ПРИНЦИПИ

2.1 Основні принципи розробки односторінкових застосунків (SPA)

Односторінкові застосунки (SPA) є методологією створення вебдодатків, які завантажують єдину HTML-сторінку та динамічно оновлюють контент у відповідь на дії користувача, забезпечуючи більш швидкий та плавний користувацький досвід. Основні принципи розробки SPA включають клієнтську маршрутизацію, асинхронне отримання даних та динамічне оновлення DOM. Розгляньмо кожен з них детальніше на прикладі простого SPA, що відображає інформацію про користувачів.

У SPA навігація між різними "сторінками" або розділами застосунку відбувається на клієнтській стороні за допомогою JavaScript, без повного перезавантаження сторінки. URL-адреса в браузері оновлюється, але сервер не залучається для отримання нової HTML-сторінки.[29]

Приклад: Уявімо SPA, що має дві основні "сторінки": "Список користувачів" та "Деталі користувача".

1. При першому завантаженні застосунку браузер отримує базову HTML-сторінку та завантажує JavaScript-код фреймворка (наприклад, React, Vue.js або Angular) та код самого застосунку.
2. Коли користувач натискає на посилання "Список користувачів", JavaScript-маршрутизатор перехоплює цю дію.
3. Замість того, щоб надсилати запит на сервер для отримання нової HTML-сторінки, маршрутизатор оновлює URL-адресу в браузері (наприклад, з #/ на #/users) та рендерить відповідний компонент "Список користувачів", який динамічно відображає дані, вже завантажені або отримані через AJAX.

4. Аналогічно, при натисканні на посилання з іменем користувача, маршрутизатор оновлює URL (наприклад, #/users/123) та відображає компонент "Деталі користувача" з інформацією про користувача з ID 123.

Таким чином, клієнтська маршрутизація забезпечує миттєві переходи між розділами застосунку, покращуючи сприйняття швидкодії.

SPA використовують асинхронні запити (зазвичай AJAX або Fetch API) для отримання даних з сервера без блокування основного потоку виконання JavaScript та без повного перезавантаження сторінки.[30]

Приклад:

1. Коли відображається компонент "Список користувачів", він може відправити асинхронний запит на сервер (наприклад, на /api/users) для отримання списку користувачів у форматі JSON.
2. Після отримання відповіді від сервера, JavaScript-код обробляє ці дані та динамічно оновлює DOM, відображаючи список користувачів на сторінці.
3. Аналогічно, коли відображається компонент "Деталі користувача" для конкретного користувача (наприклад, з ID 123), він відправляє асинхронний запит на сервер (наприклад, на /api/users/123) для отримання детальної інформації про цього користувача.
4. Після отримання даних, JavaScript оновлює DOM, відображаючи ім'я, електронну пошту та іншу інформацію про обраного користувача.

Асинхронне отримання даних дозволяє оновлювати лише необхідні частини сторінки, покращуючи продуктивність та користувацький досвід.

У SPA JavaScript відповідає за динамічну зміну вмісту вебсторінки шляхом маніпуляції Document Object Model (DOM) на основі дій користувача або отриманих даних з сервера.[31]

Приклад :

1. Після отримання списку користувачів у форматі JSON, JavaScript-код створює нові HTML-елементи (наприклад, <li> для кожного користувача) та додає їх до відповідного контейнера в DOM (наприклад, до елемента <ul>).
2. Коли користувач натискає на ім'я користувача в списку, JavaScript може динамічно змінювати вміст іншого розділу сторінки, відображаючи деталі цього користувача, або замінювати поточний компонент "Список користувачів" на компонент "Деталі користувача".
3. Якщо користувач вносить зміни до інформації про користувача у формі та зберігає їх, JavaScript може відправити асинхронний запит на сервер для оновлення даних, а після успішної відповіді динамічно оновити відображення інформації на сторінці без її повного перезавантаження.

2.2 Архітектура та принципи React

React – це один із найбільш популярних інструментів для розробки веб-додатків, який вивів розробку на новий рівень завдяки своїм особливостям та можливостям. Він почав свій шлях від команди Facebook і вибудував велику спільноту користувачів завдяки відкритості та доступності. Головна ідея Реакту полягає у розбитті користувацького інтерфейсу на окремі компоненти, що дозволяє розробникам створювати програми швидше та ефективніше. Цей підхід сприяє перевикористанню коду й уникненню зайвого повторення завдань.

Однією з ключових переваг Реакту є використання Virtual DOM – це своєрідний "віртуальний" варіант реальної моделі об'єктів у програмі, що дозволяє оптимізувати роботу з реальним DOM, модель роботи віртуально дому можна побачити на рис. 2.1. Це робить оновлення та відображення змін у інтерфейсі набагато ефективнішими [32].

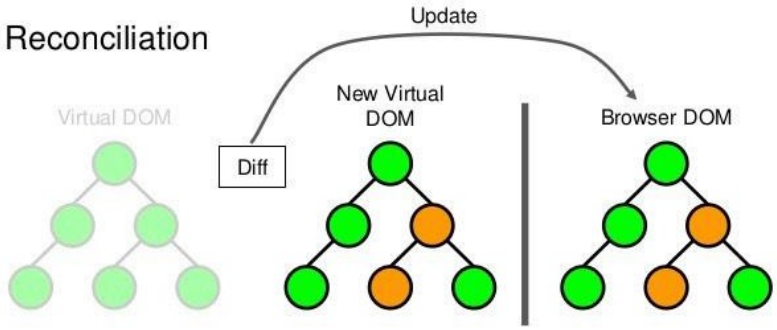


Рис. 2.1 - Віртуальний DOM

Реакт приваблює розробників завдяки великій спільноті й різноманітним інструментам, які полегшують роботу. Його гнучкість дозволяє використовувати його в проектах будь-якої складності, від невеликих до великих, що забезпечує йому стабільну популярність серед розробників.

Функціональний стиль програмування базується на використанні функцій для виконання завдань. У цьому підході функції використовуються як окремі блоки коду, які приймають вхідні дані й повертають результати. Його переваги полягають у простоті зрозуміння коду, ефективності та зручності тестування.

Класовий стиль, натомість, базується на використанні класів для створення об'єктів, які містять стан та поведінку. Його переваги включають стабільність, можливість розширення та ізоляцію стану й поведінки, що полегшує тестування та обслуговування коду.

Компоненти представляють собою ключову складову як у React, так і у Vue.js. Вони є незалежними модулями коду, які можна використовувати повторно та відповідають за відображення конкретного фрагмента контенту в програмі.

Існують два основних підходи до створення компонентів у React: функціональний і класовий [33]. Функціональні компоненти виявляються більш популярними, оскільки вони краще відповідають основним принципам

React, зокрема, простоті та можливості повторного використання коду.

Функціональний компонент React має синтаксис описаний на рис. 2.2, а класовий можна побачити на рису. 2.3.



Рис.2.2 - Ситаксис функційного компонента

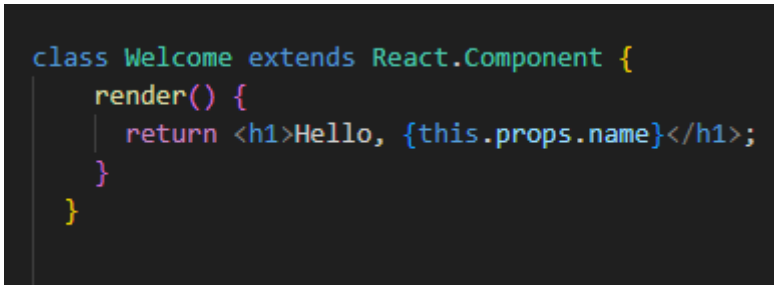


Рис. 2.3 - Синтаксис класового компонента

React, зазвичай, використовує JavaScript (JS), а саме ECMAScript 6 (або ES6), для написання компонентів. Однак, багато розробників також використовують TypeScript (TS) разом з React. TypeScript - це суперсет JavaScript, який додає статичну типізацію та деякі інші функції до JavaScript [34]. Використання TypeScript з React допомагає уникнути помилок під час розробки завдяки строгій перевірці типів, що сприяє покращенню безпеки та підтримує більшу стабільність коду.

Щодо Vue, цей фреймворк також підтримує JavaScript для розробки, але він також має додаткову підтримку для TypeScript. Vue.js 2 та Vue.js 3 мають певну ступінь інтеграції з TypeScript, що означає, що можна писати компоненти Vue за допомогою TypeScript, щоб отримати переваги статичної типізації та інших функцій, які надає TypeScript. Використання TypeScript у

Vue дозволяє зробити код більш стабільним та легким для розуміння завдяки строгій типізації, що допомагає виявляти помилки на етапі розробки.

У світі React домінує декларативний підхід до роботи з компонентами, але можливе існування елементів, які використовують імперативний стиль. Декларативний підхід в React передбачає опис бажаного стану інтерфейсу, а не напряму керує процесом оновлення. Це означає, що ви описуєте, що хочете отримати, а React сам розбирається з тим, як це відобразити. Цей підхід зазвичай забезпечує зручність, чистоту та прозорість коду.

Щодо імперативного підходу, він орієнтований на прямне керування кожним кроком оновлення інтерфейсу. Це може включати прямі маніпуляції з DOM, які зазвичай вважаються менш чистими та менш зручними для супроводу.

У React, в основному, пропагується декларативний підхід через використання властивостей та стану для відображення компонентів, але час від часу може виникати потреба у використанні імперативних підходів, наприклад, у випадках оптимізації продуктивності або взаємодії з низькорівневими DOM операціями.

Методи життєвого циклу — це основоположна концепція в розробці інтерфейсів користувача, особливо у веб-фреймворках і бібліотеках, таких як React, Vue, Angular та інші.[35] Ці методи надають можливість керувати поведінкою компонентів на різних етапах їх життя: від ініціалізації до знищення. Вони відіграють ключову роль у керуванні даними, рендерингу, обробці подій та оптимізації продуктивності застосунків.

Значення методів життєвого циклу:

1. Контроль за станом компонентів: методи життєвого циклу дозволяють розробникам виконувати код у певні моменти життя компонента. Наприклад, ініціалізація стану або встановлення початкових даних з API можуть бути виконані при створенні компонента.

2. Взаємодія з DOM: деякі методи життєвого циклу викликаються до або

після того, як компонент відображений у DOM. Це дає можливість маніпулювати DOM або виконувати дії, які залежать від розмірів чи положення елементів.

3. Оптимізація продуктивності: методи життєвого циклу дозволяють запобігти непотрібним оновленням та перерендерингам, що важливо для оптимізації продуктивності додатків, особливо при роботі з великими обсягами даних або складними інтерфейсами.
4. Керування побічними ефектами: чистка та інші дії, які мають виконуватися після знищення компонента або перед його оновленням, можуть бути реалізовані через методи життєвого циклу.
5. Реагування на зміни: ці методи можуть виявляти зміни у властивостях або стані та відповідати на них, наприклад, оновлюючи відображення або викликаючи запити до сервера.

У процесі розробки веб-додатків, методи життєвого циклу відіграють надзвичайно важливу роль, дозволяючи розробникам втілювати різноманітні функціональності та оптимізувати поведінку додатку. Наприклад, однією з ключових задач при старті компонента є завантаження даних з сервера. Це забезпечує, що користувачі мають доступ до актуальної інформації, як тільки відкривають сторінку або компонент.[36]

Крім того, важливим аспектом є управління підписками на події або сервіси. Методи життєвого циклу дозволяють не тільки підписуватися на необхідні події, але й відписуватися від них у момент, коли компонент більше не використовується. Це допомагає уникнути витоку пам'яті та забезпечити кращу продуктивність додатку. Важливою складовою є також застосування анімацій чи транзицій. Після того, як компонент було відрендерено, можна активувати анімації чи транзиції для покращення візуального враження та забезпечення більш плавного користувацького інтерфейсу. Це додає динамічності та привабливості веб-сторінкам, роблячи їх більш захоплюючими для користувачів.

Методи життєвого циклу грають ключову роль у збереженні стану компонентів при перемиканні між різними частинами додатку. Це особливо актуально в односторінкових застосунках де користувач може переходити між

різними секціями без перезавантаження сторінки. Завдяки методам життєвого циклу, стан компонентів може бути збережено або відновлено, забезпечуючи плавний та зручний перехід для користувача.

Методи життєвого циклу в класових компонентах React є ключовими для розуміння того, як компонент взаємодіє з DOM та іншими частинами додатку подано в рис. 2.4. Ці методи надають розробникам контроль над різними стадіями життя компонента: від його ініціалізації та монтажу в DOM до його оновлення та знищення. Схематично основні методи зображені на рисунку

Основні методи життєвого циклу в класових компонентах React:

Ініціалізація (Initialization): constructor(props)

- Це перший метод, який викликається в процесі створення компонента.
- Використовується для ініціалізації стану та прив'язки методів (наприклад, обробників подій) до екземпляра компонента.
- У конструкторі необхідно викликати super(props) перед будь-яким іншим виразом, щоб ініціалізувати this.props.

Монтаж (Mounting): ці методи використовуються при вставці компонента в DOM: static getDerivedStateFromProps(props, state)

- Викликається перед кожним рендерингом компонента, як під час першого монтажу, так і під час подальших оновлень.
- Використовується для оновлення стану на основі змін у властивостях (props).
- Повинен повертати об'єкт для оновлення стану або null, щоб нічого не змінювати.

render()

- Обов'язковий метод для кожного класового компонента.
- Повертає React-елементи, які представляють, має бути відображено DOM.
- Не має виконувати побічні ефекти, такі як запити до сервера або взаємодії з браузерним API.

componentDidMount()

- Викликається один раз після першого рендеринга компонента в DOM.
- Ідеальне місце для запитів до сервера, підписки або ініціалізації бібліотек, які взаємодіють з DOM.

Оновлення (Updating): ці методи використовуються при оновленні компонента через зміни в props або state.

static getDerivedStateFromProps(props, state)

- Викликається також при оновленні компонента.
- Може бути використаний для оновлення стану відповідно до нових властивостей.

shouldComponentUpdate(nextProps, nextState)

- Викликається перед кожним оновленням компонента.
- Повертає true (за замовчуванням) або false, щоб вказати, чи повинен компонент оновлюватися.
- Використовується для оптимізації продуктивності, запобігаючи непотрібним оновленням.

render()

- Викликається знову, щоб відобразити нові дані.

getSnapshotBeforeUpdate(prevProps, prevState)

- Викликається перед оновленням DOM відповідно до нових відображуваних даних.
- Використовується для збору інформації з DOM (наприклад, положення прокрутки), яку можна використовувати після оновлення.

componentDidUpdate(prevProps, prevState, snapshot)

- Викликається після оновлення компонента.

- Може бути використаний для виконання побічних ефектів, які залежать від зміненого стану чи властивостей.

Знищення (Unmounting)

componentWillUnmount()

- Викликається перед тим, як компонент буде видалений з DOM.
- Використовується для виконання прибирання: відписка від подій, зупинка таймерів, скасування запитів тощо.

Рідко використовувані методи:

componentDidCatch(error, info)

- Використовується в компонентах, що слугують "ловцями помилок" для їх перехоплення у дочірніх компонентах.

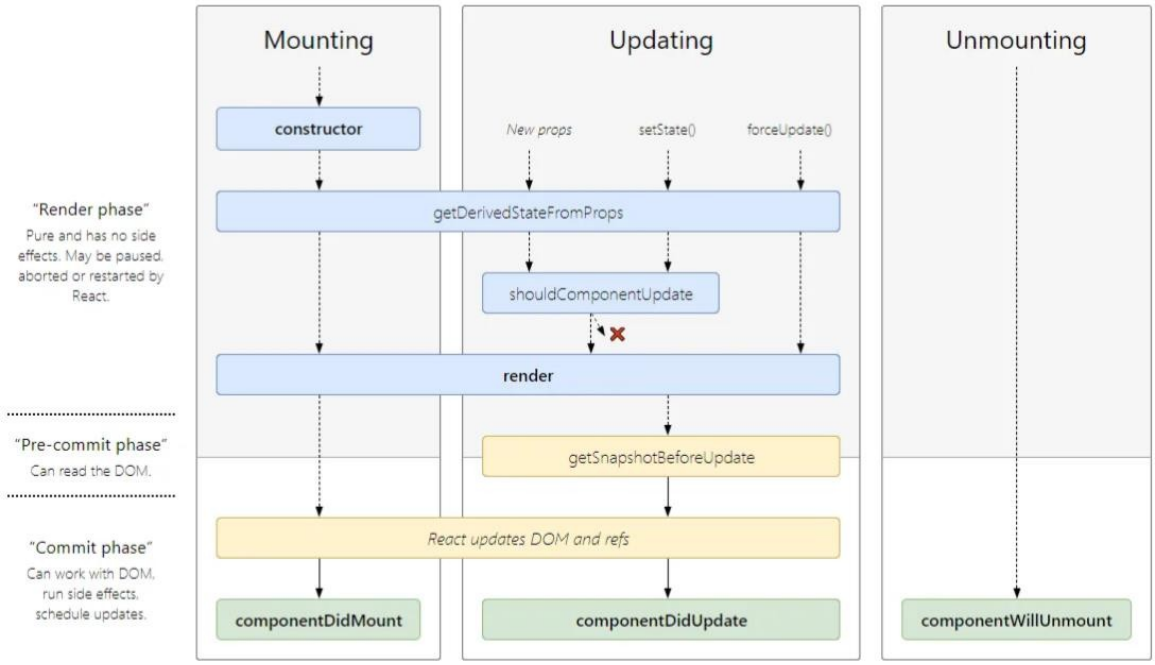


Рис. 2.4 – Методи життєвого циклу React

2.3 Тестування

Тестування в односторінкових застосунках є критично важливим для забезпечення якості, надійності та користувацького досвіду. SPA пропонують динамічний інтерфейс, змінюючи контент без перезавантаження сторінки, що створює унікальні виклики для тестування. Ось основні типи тестування:

Юніт-тестування (Unit Testing): юніт-тестування зосереджено на найменших тестових одиницях, як-то функціях або компонентах. Це включає перевірку індивідуальної функціональності для виявлення помилок у логіці або реалізації. Юніт-тести важливі для раннього виявлення помилок та спрощення процесу рефакторингу.

Інтеграційне тестування (Integration Testing): інтеграційні тести перевіряють взаємодію між різними частинами застосунку, наприклад, між компонентами UI та менеджерами стану або API. Це допомагає забезпечити, що всі частини застосунку працюють разом, як передбачено.

Кінцеве тестування (End-to-End Testing, E2E): E2E-тестування імітує реальні користувацькі сценарії та перевіряє застосунок у його повному виконанні. Це включає тестування навігації, взаємодії з користувачем, асинхронних запитів та інтеграції з зовнішніми сервісами.[37]

Інструменти, такі як Jest для юніт-тестування або Cypress для E2E тестування, є популярними серед розробників SPA.

Тестування в React та Vue.js не має суттєвих відмінностей, саме тому можна розглядати цей аспект як спільний.

Юніт-тести з використанням Jest — це потужний підхід для перевірки індивідуальних частин коду, таких як функції або компоненти, в ізоляції від інших частин програми. Jest — це тестовий фреймворк JavaScript, який надає швидке встановлення та простоту використання, роблячи його популярним вибором серед розробників. При юніт-тестуванні з Jest кожен тест фокусується на перевірці конкретної функціональності. Тести описують очікувану поведінку функцій або компонентів і перевіряють, чи відповідає реальна поведінка цим очікуванням. взаємодії з зовнішніми модулями та відстежувати, як ці модулі використовуються.[38]

Завдяки своїй простоті та гнучкості, Jest є ідеальним рішенням для написання та виконання юніт-тестів у сучасних JavaScript-проектах.

End-to-End (E2E) тестування з використанням Cypress є процесом, який імітує поведінку кінцевого користувача для перевірки загальної

функціональності веб-застосунків. Cypress — це потужний фреймворк для автоматизованого тестування, який дозволяє розробникам легко писати, запускати та відлагоджувати тести у реальному браузері. E2E тести з Cypress включають сценарії, які автоматично виконують дії у браузері, такі як введення даних у форми, натискання кнопок, перехід між сторінками та перевірка виводу.[39] Cypress підтримує автоматичне очікування елементівї.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ІНТЕРФЕЙСУ СТУДІЇ ДИЗАЙНУ ЯК SPA НА REACT

3.1 Конструювання прототипу вебсайту “ LuxDesign ”

У цьому розділі розглядається повний процес розробки вебсайту для студії дизайну “ LuxDesign ” із застосуванням принципів адаптивного дизайну, сучасних вебтехнологій та інструментів інтерфейсної розробки. Основна мета розділу — надати систематизований огляд архітектури сайту, обраних технологій, а також методологій, використаних під час створення ключових функціональних елементів вебплатформи.

Проєкт реалізовано як односторінковий додаток (SPA), що забезпечує швидке завантаження контенту та плавну навігацію між розділами без перезавантаження сторінки. Основу інтерфейсного рішення складає адаптивна HTML/CSS-структура з використанням CSS-змінних і медіа-запитів, яка доповнюється логікою на JavaScript для інтерактивної взаємодії користувача з сайтом. Для стилізації використано модульний підхід із централізованим управлінням темами, що дозволяє зручно змінювати візуальне оформлення.

Важливою складовою розробки стало оптимізоване відображення портфоліо, реалізоване через динамічну фільтрацію за категоріями, а також адаптивне контактне меню, що забезпечує легкий зворотний зв’язок для клієнтів. Крім того, були реалізовані інтерактивні блоки — акордеон з поширеними запитаннями (FAQ), анімовані елементи при прокрутці, форма з валідацією, що покращує користувацький досвід і додає професійності до сприйняття бренду студії.

У рамках розробки було також проведено тестування вебсайту на різних пристроях і розмірах екранів, включно з мобільними телефонами, планшетами та десктопами. Особлива увага приділялася оптимізації зображень та графічного контенту, що має велике значення для дизайну-орієнтованого ресурсу. Завдяки попередньому стисненню зображень та використанню сучасних форматів (WebP), було досягнуто значного зниження часу завантаження сторінки без втрати якості.

Проєктування є ключовим етапом розробки програмного забезпечення, що включає визначення структури системи, взаємодії між її компонентами та специфікацію функціональних вимог. Основною метою проєктування є створення структурованої моделі, яка визначає, як різні частини системи співпрацюють для досягнення поставлених завдань.

Сайт “LuxDesign” є інтерактивною вебплатформою, розробленою для презентації послуг студії з дизайну інтер’єрів та екстер’єрів, а також для ефективної комунікації з потенційними клієнтами. Основний акцент у проєкті зроблено на створення естетично привабливого та функціонального користувацького інтерфейсу (UI), який дозволяє легко ознайомитися з портфоліо студії, замовити консультацію або зв'язатися з представниками команди.

Під час проєктування інтерфейсу враховано потреби різних категорій користувачів — як приватних замовників, так і представників комерційних компаній. Було реалізовано інтуїтивно зрозумілу структуру сайту з чіткою навігацією, адаптивними блоками контенту та інтерактивними елементами, що сприяє комфортному перегляду навіть на мобільних пристроях.

Ключові функціональні компоненти сайту включають:

- портфоліо з фільтрацією за категоріями (житловий, комерційний, екстер’єрний дизайн);
- доступ до детального опису кожного проєкту з галереями зображень;
- розділ “Послуги” з описом напрямів діяльності студії;
- форма запиту консультації, яка забезпечує швидкий зворотний зв’язок;
- розділ “Команда”, що презентує фахівців з короткою біографією та експертизою.

Сайт реалізований з урахуванням адаптивного дизайну та принципів UX, що забезпечує комфортну навігацію та швидке завантаження навіть за умов повільного інтернет-з’єднання. Таким чином, “Lux Design” виступає не лише як вітрина студії, але й як повноцінна комунікаційна платформа, орієнтована на залучення клієнтів і підтримку іміджу бренду.

Проєктування архітектури вебсайту студії дизайну “ LuxDesign ” включає розробку структурних моделей, які визначають логіку організації основних функціональних елементів платформи. Однією з ключових моделей цього етапу є діаграма класів, що дозволяє візуалізувати сутності, їхні атрибути, методи та взаємозв’язки між ними.

Архітектура системи базується на модульному підході, що передбачає незалежність функціональних блоків і полегшує підтримку та масштабування платформи. Основні класи системи охоплюють управління вмістом портфоліо, послугами студії, командою фахівців, запитами відвідувачів та адміністративним управлінням сайтом.

Опис основних класів:

1. Visitor

Клас, що представляє відвідувача сайту. Відвідувач може надіслати запит або повідомлення через контактну форму.

- Атрибути:
  visitorID: унікальний ідентифікатор відвідувача
  name: ім’я користувача
  email: електронна пошта
  message: зміст повідомлення
- Метод:
  sendMessage() — надсилання повідомлення через форму зворотного зв’язку.

2. Project

Клас, який описує окремі дизайн-проєкти (інтер’єри, екстер’єри, комерційні об’єкти тощо), що відображаються в портфоліо студії.

- Атрибути:
  projectID: ідентифікатор проєкту
  title: назва проєкту
  description: опис роботи

category: тип дизайну (інтер'єр, екстер'єр, інше)

images: галерея візуальних матеріалів

- Метод:

  displayProject() — виведення інформації про проєкт у візуальному форматі.

3. Admin

Клас адміністратора сайту, який має доступ до системних функцій керування контентом.

- Атрибути:

  adminID: унікальний ідентифікатор

  username: ім’я користувача

  password: пароль доступу

- Методи:

  login() — авторизація адміністратора

  manageContent() — керування розділами сайту, редагування проектів, послуг, команди

  respondToMessages() — перегляд та відповідь на повідомлення відвідувачів

Для розробки нашого проєкту на React ми будемо використовувати VisualStudioCod від компанії Microsoft.

VisualStudioCod – це комерційний редактор коду, спеціалізований на роботі з JavaScript та підтримці широкого набору інструментів для веброзробки. [40] Він надає значні переваги для розробників, завдяки своїм інтелектуальним можливостям. Наприклад, потужне автозаповнення, рефакторинг та навігація по коду дозволяють значно підвищити продуктивність розробки. Цей редактор також має вбудовану підтримку різноманітних систем контролю версій, включаючи Git, GitHub та SVN, що спрощує управління версіями коду.

VisualStudioCod забезпечує спеціалізовану підтримку для популярних фреймворків, таких як React, Vue.js та Angular, що робить його особливо

корисним для сучасних веброзробників. Вбудований дебагер дозволяє швидко знаходити та виправляти помилки в коді, що підвищує ефективність розробки. Крім того, редактор легко інтегрується з різноманітними інструментами розробки, включаючи тестові фреймворки, такі як Jest, що необхідно для повсякденної роботи розробника.

Технічна реалізація

HTML-структура

У даній версії вебплатформи реалізовано класичну розмітку з контейнером , який обгортає ключові секції — шапку, основний контент та футер. Основна навігація реалізована через елементи <a> із призначеними атрибутами data-section, які відповідають певним секціям сайту (home, about, portfolio, services, contact). Це дозволяє обробляти переходи між розділами за допомогою JavaScript без оновлення сторінки.

Усередині <main> розміщено кілька <section> із унікальними id, які відповідають різним розділам сайту (home, about, portfolio, services, contact). Активна секція має клас active, що дозволяє через JavaScript перемикатися між ними, імітуючи навігацію між сторінками без оновлення вікна браузера.

Реалзація:

```
<div id="app">
   <header class="header">...</header>
   <main id="content">
      <section id="home" class="section active">...</section>
      <section id="about" class="section">...</section>
      <section id="portfolio" class="section">...</section>
      <section id="services" class="section">...</section>
      <section id="contact" class="section">...</section>
   </main>
   <footer class="footer">...</footer>
</div>
```

Навігаційне меню на головній сторінці використовує атрибути data-section для SPA-навігації:

```
<nav class="main-nav">
  <ul>
    <li><a href="#home" class="nav-link active" data-section="home">Головна</a></li>
    <li><a href="#about" class="nav-link" data-section="about">Про нас</a></li>
    <li><a href="#portfolio" class="nav-link" data-section="portfolio">Портфоліо</a></li>
    <li><a href="#services" class="nav-link" data-section="services">Послуги</a></li>
    <li><a href="#contact" class="nav-link" data-section="contact">Контакти</a></li>
</ul>
</nav>
```

Цй підхід дозволяє організувати зручну навігацію, зменшуючи затрати на рендеринг нових сторінок.

CSS-стилізація

Стилізація сайту побудована на основі CSS-змінних, що оголошуються у кореневому блоці :root. Це дозволяє централізовано керувати кольоровою палітрою, шрифтами, тінями, радіусами та ефектами переходу.

Ключова перевага такого підходу — спрощене масштабування та зміна теми сайту без необхідності глибокого втручання в структуру стилів. Крім того, кожна секція може мати власні унікальні стилі, що підвищує гнучкість дизайну.

Видимість розділів SPA реалізується наступним чином:

```
:root {
  --primary-color: #d4af37;
  --primary-light: #f1e5c0;
  --primary-dark: #a08428;
  --secondary-color: #1a1a1a;
```

```
    --text-color: #333;
    --text-light: #777;
    --background-color: #ffffff;
    --background-light: #f9f9f9;
    --background-dark: #f0f0f0;
    --accent-color: #e74c3c;
    --border-color: #e0e0e0;
    --box-shadow: 0 5px 20px rgba(0, 0, 0, 0.05);
    --transition: all 0.3s ease;
    --border-radius: 5px;
}
```

Для SPA-функціональності використовуються спеціальні стилі, що керують видимістю розділів:

```
.section { display:
    none;
}
.section.active { display:
    block;
}
```

Це дозволяє контролювати, який розділ показаний користувачеві в конкретний момент часу.

JavaScript-функціональність

Функціональна логіка SPA реалізована за допомогою нативного JavaScript. Основна функція initNavigation() забезпечує перемикання між розділами, оновлюючи класи active у відповідних DOM-елементах.

Ключова перевага цього підходу — швидка реакція інтерфейсу без затримок, оскільки навігація виконується без оновлення сторінки. Також реалізовано допоміжні функції:

Ключовий фрагмент коду:

```
// Функція для SPA-навігації
function initNavigation() {
    const spaNavLinks = document.querySelectorAll('.nav-link');
    if (spaNavLinks.length > 0) {
        spaNavLinks.forEach(link => {
            link.addEventListener('click', function(e) {
                e.preventDefault();
                // Видалення класу active з усіх посилань spaNavLinks.forEach(link =>
                link.classList.remove('active'));
                // Додавання класу active до натиснутого посилання
                this.classList.add('active');
                // Отримання цільової секції
                const targetSection = this.getAttribute('data-section');
                // Приховування всіх секцій
                const sections = document.querySelectorAll('.section');
                sections.forEach(section => section.classList.remove('active'));
                // Відображення цільової секції
                document.getElementById(targetSection).classList.add('active');
            });
        });
    }
}
```

Адаптивний дизайн

Для підтримки адаптивності застосовуються медіа-запити, які дозволяють змінювати вигляд і компонування залежно від розміру екрана. Наприклад, при ширині < 768px навігаційне меню трансформується у приховану бічну панель із бургер-кнопкою.

Цей підхід покращує досвід користувача, забезпечуючи зручну взаємодію незалежно від типу пристрою. Ретельно підібрані розміри сіток, шрифтів і

елементів керування дозволяють інтерфейсу залишатися функціональним навіть на невеликих екранах.

Також реалізовано додаткову функціональність:

- Фільтрація проєктів у портфоліо
- Акордеон для FAQ
- Обробка відправки контактної форми з валідацією
- Плавний скролінг до розділів

Приклад реалізації:

```
@media (max-width: 992px) {
    .container {
        max-width: 768px;
    }
    .project-grid,
    .services-container,
    .services-list,
    .portfolio-grid,
    .team-grid,
    .values-grid {
        grid-template-columns: repeat(2, 1fr);
    }
    /* Інші адаптивні стилі */
}
@media (max-width: 768px) {
    /* Стилі для мобільних пристроїв */
    .menu-toggle  {  display:
        block;
    }
    .main-nav    {    position:
        fixed; top: 80px;
        left: -100%;
```

```
    width: 100%;
    height: calc(100vh - 80px);
    background-color: white;
    transition: var(--transition); z-
    index: 1000;
  }
  /* Інші адаптивні стилі */
}
```

Це дозволяє забезпечити ефективну роботу сайту на будь-якому пристрої, покращуючи загальний UX і продуктивність.

3.2. Тестування

Під час тестування зверталася увага на те, чи відображаються всі компоненти (тексти, зображення, відео, графіка) чітко та без викривлень. Проводилася перевірка вебсайту у різних веббраузерах (Chrome, Firefox, Safari, Edge) та операційних системах (Windows, macOS, iOS, Android) для виявлення та усунення потенційних проблем із сумісністю рис. 3.1.

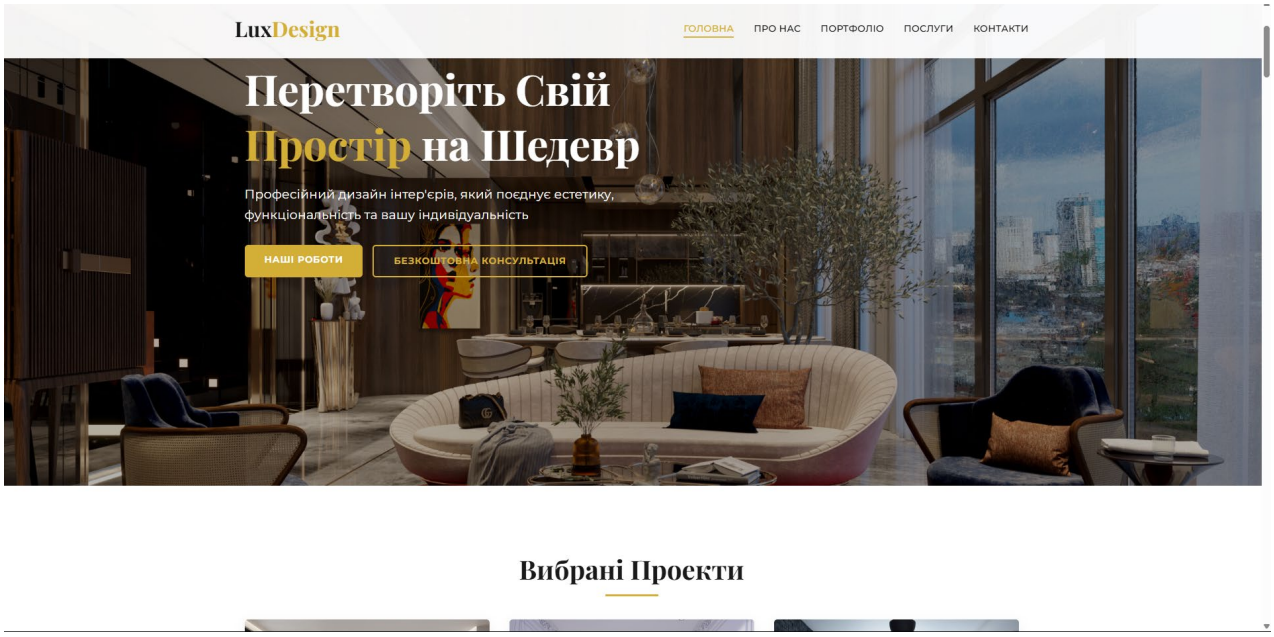


Рис. 3.1 - Сторінка Overview. Deskop

Тестування сторінки “Portfolio” проводилося аналогічно до головної

сторінки, було проведено перевірку її адаптивності, функціональності, візуальної цілісності та сумісності на різних пристроях і розмірах екранів. Аналогічно до тестування головної сторінки, перевірка включала відкриття сторінки " Portfolio " на настільних комп'ютерах, ноутбуках, планшетах та смартфонах, а також використання інструментів розробника для симуляції різних розмірів екранів рис. 3.2.

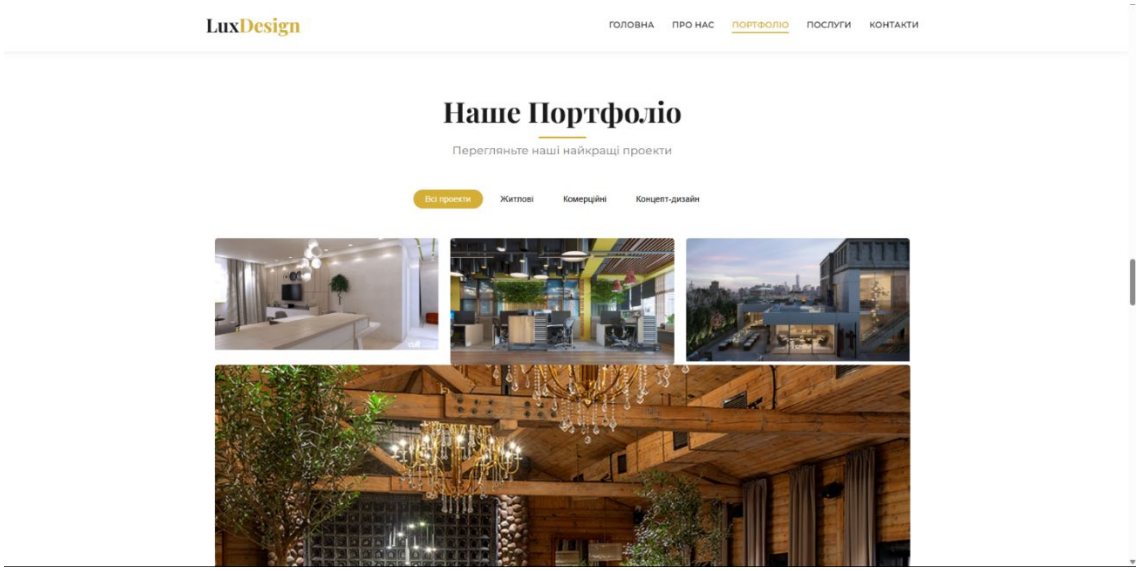


Рис. 3.2 - Сторінка “Portfolio”

У процесі тестування сторінок вебсайту на різних пристроях та розмірах екранів було виявлено, що всі елементи відображаються коректно. Дизайн веб-сайту адаптується до будь-яких розмірів екранів, забезпечуючи зручну навігацію та приємний користувацький досвід. Текст, зображення та інтерактивні елементи, такі як кнопки та посилання, зберігають свою цілісність і функціональність на настільних комп'ютерах, ноутбуках, планшетах та смартфонах. Адаптивність макета забезпечує, що контент залишається читабельним і привабливим незалежно від типу пристрою.

3.3. Методологія дослідження

Для порівняння ефективності SPA та MPA підходів було проведено наступні

дослідження:

- Вимірювання швидкості завантаження сторінок
- Аналіз користувацького досвіду (UX) через метрики взаємодії
- Оцінка SEO-оптимізації за допомогою інструментів Google
- Вимірювання навантаження на сервер при різних сценаріях використання

Порівняння швидкості завантаження для SPA та MPA

Табл. 3.1

Параметр	SPA (index.html)	MPA (about.html, contact.html)
Початкове завантаження	3.2с	1.8с
Перехід між розділами	0.1с	1.5с
Загальний час для перегляду всіх розділів	3.5с	7.5с

Оцінка аспектів користувацького досвіду

Табл. 3.2

Аспект UX	SPA	MPA
Плавність переходів	Висока	Низька
Час реакції інтерфейсу	Високий	Середній
Зручність навігації	Висока	Середня
Збереження контексту	Високе	Низьке
Зворотний зв'язок	Миттє вий	Із затримкою

Аналіз SEO-оптимізації

Табл. 3.3

Аспект SEO	SPA	MPA
Індексація пошуковими системами	Потребує додаткових заходів	Природна
Унікальні URL для розділів	Потребує додаткової реалізації	Природна
Метатеги для кожного розділу	Складно реалізувати	Просто реалізувати
Швидкість індексації	Нижча	Вища

Порівняння навантаження на сервер

Табл. 3.4

Сценарій	SPA	MPA
Початкове завантаження	Високе	Середнє
Переходи між розділами	Відсутнє	Середнє
Навантаження при 100 користувачах	Нижче	Вище
Використання трафіку	Менше	Більше

3.4. Результати дослідження

Розробка односторінкових веб-застосунків (SPA) з використанням React охоплює спектр підходів та практик, які відображають сучасні тенденції у веб-розробці.

Аналізуючи створені компоненти та їх тести для React з використанням відповідних екосистем інструментів, можна зробити висновок, що обидва фреймворки забезпечують ефективні та сучасні засоби для розробки SPA. React компоненти, з їхніми хуками та контекстом Redux, демонструють потужність управління станом і сайд ефектами. Тести для цих компонентів, написані з використанням Jest та інструментів, специфічних для кожного

фреймворку, обидві платформи підтримують надійне тестування з моками та асинхронними запитами. Використання моків для API запитів дозволяє ізолювати компоненти від зовнішніх залежностей, тим самим забезпечуючи точність та предбачуваність тестів.

В кінцевому підсумку React пропонує розробникам потужні інструменти не тільки для створення, але й для тестування односторінкових застосунків.

Загальні результати порівняльного аналізу показують, що для веб-сайтів дизайну інтер'єрів найефективнішим є гібридний підхід, що поєднує SPA та MPA:

SPA-підхід ефективний для:

- Головної сторінки з основними розділами сайту
- Демонстрації портфоліо проектів
- Реалізації інтерактивних елементів (фільтрів, галерей, тощо)

MPA-підхід ефективний для:

- Сторінок з великим обсягом унікального контенту
- Сторінок, які потребують окремої SEO-оптимізації
- Сторінок з комплексними формами (контакти, замовлення)

Гібридний підхід дозволяє максимізувати переваги обох методологій та мінімізувати їх недоліки

Висновок

У межах дослідження було проведено комплексне дослідження ефективності використання архітектури Single Page Application (SPA) у порівнянні з традиційними Multi Page Application (MPA) для розробки студії дизайну інтер'єрів та екстер'єрів "LuxDesign" з використанням технологій HTML, CSS, JavaScript та адаптивного підходу до інтерфейсу. Основна мета — створення інтерактивного, естетичного та зручного для користувачів вебресурсу, що забезпечує легкий доступ до послуг студії, перегляду портфоліо, ознайомлення з командою та зворотного зв'язку. Основною метою проєкту було визначити доцільність застосування технологій SPA з використанням фреймворку React, а також сформулювати рекомендації щодо вибору веб-архітектури відповідно до потреб бізнесу.

У результаті дослідження Було розроблено архітектуру сайту з урахуванням принципів модульності та масштабованості. Детально спроектовано структурну модель системи з використанням діаграми класів, що відображає логіку взаємодії основних компонентів: відвідувачів, адміністраторів, проєктів, послуг та команди. Реалізовано механізм SPA-навігації, що дозволяє перемикатися між секціями без перезавантаження сторінки, підвищуючи продуктивність і покращуючи користувацький досвід та досягнуто таких основних наукових та практичних результатів:

1. Проведено огляд сучасних веб-технологій, зокрема SPA і MPA, на основі якого визначено ключові переваги SPA у контексті взаємодії з користувачем, продуктивності та технічної реалізації.
2. Виявлено архітектурні особливості фреймворку React, серед яких модульність, віртуальний DOM, багатий набір екосистемних інструментів і зручність розробки масштабованих інтерфейсів.

3. Створено та проаналізовано функціональні характеристики SPA і MPA у контексті бізнес-потреб. SPA було визнано кращим вибором для динамічних інтерфейсів із високим рівнем взаємодії, тоді як MPA може бути ефективнішим у проектах з великим обсягом інформаційних сторінок.
4. Здійснено оцінку впливу технологічних рішень на користувацький досвід, адаптивність, навігацію та загальну ефективність веб-застосунків.

Здійснений аналіз підтверджує доцільність використання SPA-підходу для розробки вебпредставництва дизайн-студії, оскільки він дозволяє досягти високої інтерактивності, зниження навантаження на сервер та покращення швидкодії застосунку. Сайт дозволяє студії демонструвати професійний рівень виконання робіт, покращує взаємодію з потенційними клієнтами та сприяє підвищенню впізнаваності бренду.

На основі проведеного дослідження сформульовано такі практичні рекомендації:

1. Впровадження SPA-технологій до вебресурсів компаній, що працюють у креативних індустріях, дозволяє ефективніше демонструвати візуальний контент і взаємодіяти з клієнтами.
2. Застосування модульного підходу до архітектури дозволяє адаптувати систему під різні бізнес-моделі — від студій до архітектурних бюро, що може бути масштабовано для корпоративного використання.
3. Доцільно впровадити функціонал персоналізованого контенту, зокрема на основі аналітики користувацької поведінки, що підвищить ефективність маркетингової стратегії.

Таким чином, використання SPA-підходу в поєднанні з грамотно спроєктованою архітектурою та інтуїтивно зрозумілим UI є оптимальним та можуть слугувати підґрунтям для подальших наукових досліджень у галузі вебтехнологій в умовах сучасного цифрового середовища.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. React Single Page Application і React-Router для початківців. Medium. 2024. URL:https://medium.com/nuances-of-programming/react-single-page-application-i-react-router-для-початківців-5b43729b7100 (дата звернення 21.12.2024)
2. Міністерство цифрової трансформації України. 2024. URL:https://thedigital.gov.ua/ (дата звернення 21.12.2024)
3. Односторінковий застосунок. Вікіпедія. 2025. URL:https://uk.wikipedia.org/wiki/Односторінковий_застосунок (дата звернення 21.12.2024)
4. Single Page Application: Definition, Benefit, Architecture & Example. Synodus. 2025. URL:https://synodus.com/blog/web-development/single-page-application/ (дата звернення 21.12.2024)
5. React Ecosystem in 2024. DEV Community. 2024. URL:https://dev.to/avinashvagh/react-ecosystem-in-2024-418k (дата звернення 21.12.2024)
6. Обговорення: React Single Page Application і React-Router. DOU.ua. URL:https://dou.ua/forums/topic/25444/ (дата звернення 21.12.2024)
7. Responsive Web Design. Ethan Marcotte. 2024. URL:https://alistapart.com/article/responsive-web-design/ (дата звернення 21.12.2024)
8. Mobile First: мобільна версія сайту. Як створити, особливості, дизайн, функціонал. URL:https://example.com/mobile-first-design (дата звернення 21.12.2024)
9. Responsive vs. Adaptive vs. Fluid Design - Learn to code in 30 Days! URL:https://example.com/responsive-adaptive-fluid-design (дата звернення 21.12.2024

10. Responsive and Adaptive Design. SendPulse URL:https://sendpulse.ua/blog/responsive-and-adaptive-design (дата звернення 21.12.2024)
11. Що таке респонсивний дизайн? WizeClub. URL:https://wizeclub.education/blog/shho-take-responsivnij-dizajn/ (дата звернення 21.12.2024)
12. Як зробити один сайт для всіх екранів. Skvot. URL:https://skvot.io/uk/blog/kak-sdelat-odin-sait-dlia-vsekh-ekranov (дата звернення 21.12.2024)
13. Що таке SEO? SEO-Razum. URL:https://seo-razum.com.ua/ua/stati/sho-take-seo/ (дата звернення 21.12.2024)
14. React – The library for web and native user interfaces. URL:https://react.dev/ (дата звернення 21.12.2024)
15. React Documentation. 2024. URL:https://react.dev/learn (дата звернення 22.12.2024)
16. Обговорення: React та SPA. DOU.ua. URL:https://dou.ua/forums/topic/46244/ (дата звернення 22.12.2024)
17. Чи відходить React від SPA?. Хабр. URL:https://habr.com/ru/articles/123456/ (дата звернення 22.12.2024)
18. Using responsive images in HTML. MDN Web Docs. URL:https://developer.mozilla.org/en-US/docs/Learn/HTML/Multimedia_and_embedding/Responsive_images (дата звернення 22.12.2024)
19. Why does speed matter? web.dev. URL:https://web.dev/speed-matters/ (дата звернення 22.12.2024)
20. Angular vs React vs Vue: Core Differences. BrowserStack. URL:https://www.browserstack.com/guide/angular-vs-react-vs-vue (дата звернення 22.12.2024)

21. Single Page App (SPA). React Router. URL:https://reactrouter.com/web/guides/quick-start (дата звернення 23.12.2024)
22. clamp() - CSS: Cascading Style Sheets. MDN Web Docs. URL:https://developer.mozilla.org/en-US/docs/Web/CSS/clamp (дата звернення 23.12.2024)
23. Обговорення: SPA та React. DOU.ua. URL:https://dou.ua/forums/topic/50894/ (дата звернення 23.12.2024)
24. React Flashcards. Brainscape. URL:https://www.brainscape.com/flashcards/react-13421397/packs/21397471 (дата звернення 23.12.2024)
25. React Internals FAQ. ReactJS. URL:https://uk.legacy.reactjs.org/docs/faq-internals.html (дата звернення 23.12.2024)
26. Single Page Apps: Revolutionizing Web Development. LinkedIn. URL:https://www.linkedin.com/pulse/single-page-apps-revolutionizing-web-development-2024-cx0rc/ (дата звернення 23.12.2024)
27. Мемекод: Ardoberdag. Tseivo. URL:https://tseivo.com/b/memecode/t/ardoberdag (дата звернення 23.12.2024)
28. Web Performance. Google Developers. URL:https://developers.google.com/web/fundamentals/performance (дата звернення 23.12.2024)
29. Responsive Images. MDN Web Docs. 2024. URL:https://developer.mozilla.org/en-US/docs/Learn/HTML/Multimedia_and_embedding/Responsive_images (дата звернення 19.01.2025)
30. Asynchronous Communication. Dropbox. URL:https://experience.dropbox.com/uk-ua/resources/asynchronous-communication (дата звернення 19.01.2025)

31. Document Object Model (DOM). MDN Web Docs. URL:https://developer.mozilla.org/ru/docs/Web/API/Document_Object_Model (дата звернення 19.01.2025)
32. Ridham Z. The Reconciliation Algorithm. DEV Community. URL:https://dev.to/ridhamz/the-reconciliation-algorithm-5bab (дата звернення 19.01.2025)
33. Maria Hladka. React Single Page Application і React-Router для початківців. Medium. 2024. URL:https://medium.com/nuances-of-programming/react-single-page-application-i-react-router-для-початківців-5b43729b7100 (дата звернення 22.01.2025)
34. Що таке TypeScript? Foxminded. URL:https://foxminded.ua/typescript/ (дата звернення 22.01.2025)
35. React, Vue чи Angular: що краще? Markup UA. URL:https://markup-ua.com/react-vue-angular-shho-krashhe/ (дата звернення 19.01.2025)
36. О. Алексєєва. Методи життєвого циклу компонента в React. LinkedIn. URL:https://www.linkedin.com/pulse/методи-життєвого-циклу-компонента-в-react-один-з-на-aleksieieva/ (дата звернення 23.01.2025)
37. End-to-End тестування в React. Foxminded. URL:https://foxminded.ua/end-to-end-testuvannia/ (дата звернення 23.01.2025)
38. Jest – Тестування JavaScript-коду. JestJS. URL:https://jestjs.io/uk/ (дата звернення 23.01.2025)
39. Cypress – перше end-to-end тестування. IT Notes Wiki. URL:https://www.it-notes.wiki/automation-testing/cypress-first-end-to-end-test/ (дата звернення 23.01.2025)
40. Visual Studio Code Extension for Power Apps Portals. Microsoft Learn. URL:https://learn.microsoft.com/uk-ua/power-apps/maker/portals/vs-code-extension (дата звернення 23.01.2025)