ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Оптимізація ефективності веб-інтерфейсів через використання технік та інструментів JavaScript»

на здобуття освітнього ступеня бакалавра
зі спеціальності 123 Комп'ютерної інженерії
(код, найменування спеціальності)
освітньо-професійної програми Комп'ютерна інженерія
(назва)

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

_____ *Богдан КОВАЛЬСЬКИЙ*
(підпис) *Ім'я, ПРІЗВИЩЕ здобувача*

Виконав: здобувач(ка) вищої освіти гр. КІД-41
Богдан КОВАЛЬСЬКИЙ
Ім'я, ПРІЗВИЩЕ
Керівник: Світалана КУФТЕРІНА
науковий ступінь, вчене звання Ім'я, ПРІЗВИЩЕ
Рецензент:
науковий ступінь, вчене звання Ім'я, ПРІЗВИЩЕ

Київ 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут Інформаційних технологій

Кафедра 123 Комп'ютерної інженерії

Ступінь вищої освіти бакалавр

Спеціальність 123 Комп'ютерної інженерії

Освітньо-професійна програма Комп'ютерна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедрою Наталія ЛАЩЕВСЬКА
_____Ім'я, ПРІЗВИЩЕ
«_____» _____ 2024р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Ковальський Богдан Андрійович
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: Оптимізація ефективності веб-інтерфейсів через використання технік та інструментів JavaScript
керівник кваліфікаційної роботи Світлана КУФТЕРІНА,
(Ім'я, ПРІЗВИЩЕ, науковий ступінь, вчене звання)
затверджені наказом Державного університету інформаційно-комунікаційних технологій від « 27 » лютого 2024р. № 36
2. Строк подання кваліфікаційної роботи «3» червня 2024р.
3. Вихідні дані до кваліфікаційної роботи: Функціонал мов програмування таких як JavaScript. Детальний огляд області, в якій буде застосовуватися розроблене програмне забезпечення. Плану тестування програмного забезпечення, методів та інструментів, які будуть використані для перевірки коректності його роботи та відповідності вимогам.
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Огляд мови програмування JavaScript
2. Фактори, що впливають на ефективність веб-інтерфейсу
3. Фреймворки JavaScript: роль, мета, принципи роботи
5. Перелік ілюстративного матеріалу: *презентація*
6. Дата видачі завдання «27» лютого 2024р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Збір даних	28.02-22.03.2024	Виконано
2	Обробка матеріалу	23.03-05.04.2024	Виконано
3	Розробка теоретичної частини	06.04-10.04.2024	Виконано
4	Підготовка до дослідження	11.04-30.04.2024	Виконано
5	Проведення дослідження	01.05-03.05.2024	Виконано
6	Висновки за результатами аналізу	04.05-08.05.2024	Виконано
7	Розробка презентації	09.05-11.05.2024	Виконано
8	Передзахист	14.05-17.05.2024	Виконано
9	Здача в деканат	25.05-3.06.2024	Виконано

Здобувач(ка) вищої освіти _____ Богдан КОВАЛЬСЬКИЙ
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Керівник кваліфікаційної роботи _____ Світлана КУФТЕРІНА
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 59 стор., 12 рис., 30 джерел.

Мета роботи – дослідити та проаналізувати різноманітні техніки та інструменти використання JavaScript для оптимізації ефективності веб-інтерфейсів. Вона спрямована на вивчення та розуміння того, як JavaScript може бути використаний для поліпшення швидкодії, реактивності та загального користувацького досвіду веб-додатків та веб-сайтів.

Об'єкт дослідження – CRUD-додаток для веб-сторінки.

Предмет дослідження – процес створення, розгортання та оптимізації CRUD додатку з використанням JavaScript та інших технологій веб-розробки.

Короткий зміст роботи: Веб-інтерфейс відіграє ключову роль у веб-розробці, бо він є головним засобом, що забезпечує взаємодію користувачів з веб-додатками та контентом в Інтернеті. Завдяки сучасним мовам програмування, їх функціоналу, бібліотекам та фреймворкам реалізувати будь-який проект не стане великою проблемою. Робота досліджує роль JavaScript у веб-розробці та розробляє CRUD додаток з використанням JavaScript та Local Storage. Огляд різних методів оптимізації веб-інтерфейсів, вивчення стратегій оптимізації для мобільних пристроїв, аналіз фреймворків JavaScript та середовища розробки Visual Studio Code також є частиною дослідження. Результатом є створення функціонального CRUD додатку, який дозволяє користувачам здійснювати операції створення, читання, оновлення та видалення даних, а також зберігати їх локально.

КЛЮЧОВІ СЛОВА: ВЕБ-ІНТЕРФЕЙС, ВЕБ-РОЗРОБКА,ФРЕЙМВОРКИ, ОПТИМІЗАЦІЯ, SPA, JAVASCRIPT, API, TYPESCRIPT, AJAX, HTML,CSS.

ЗМІСТ

ВСТУП…….…...10
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ…...……...9
РОЗДІЛ 1. ВВЕДЕННЯ ДО ВЕБ-ІНТЕРФЕЙСІВ ТА JAVASCRIPT..…….….11
1.1 Визначення поняття веб-інтерфейсу та його роль у веб-розробці ……..11
1.2 Фактори, що впливають на ефективність веб-інтерфейсу …….……14
1.3 Огляд основних принципів оптимізації веб-інтерфейсів та їх ролі у поліпшенні ефективності користування ……..……...17
1.4 Огляд мови програмування JavaScript ……..……21
1.5 Роль JavaScript у розробці Веб-додатків ……..……24
1.6 Опис базових концепцій та функцій JavaScript ….……..……26
1.7 Переваги та обмеження використання JavaScript у веб-розробці …..……30
РОЗДІЛ 2. УДОСКОНАЛЕННЯ ТА ОПТИМІЗАЦІЯ ВЕБ-ІНТЕРФЕЙСІВ: МЕТОДИ ТА ЗАСОБИ…...……...…33
2.1 Огляд різних методів оптимізації швидкодії веб-інтерфейсів, Local та Session Storage …….….…33
2.2 Вивчення підходів до оптимізації веб-інтерфейсів для різних типів додатків……36
2.3 Стратегії оптимізації веб-інтерфейсів для мобільних пристроїв та інших платформ ……...39
2.4 Фреймворки JavaScript: роль, мета, принципи роботи…… …….....40
2.5 Середовище розробки Visual Studio Code ……..… …….....45
РОЗДІЛ 3. РОЗРОБКА ДОДАТКУ CRUD З ВИКОРИСТАННЯМ JAVASCRIPT ТА LOCAL STORAGE……..…….…48
3.1 Огляд та визначення CRUD додатку……...48
3.2 Реалізація операцій CRUD з та використання LocalStorage……50
3.3 Тестування та підведення підсумків…..……64

ВИСНОВКИ……..…..70
ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ…….…….71
Додаток А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ (Презентація)……74

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

UI - інтерфейс користувача

API - інтерфейс прикладного програмування

HTML - мова гіпертекстової розмітки

CSS - формальна мова опису зовнішнього вигляду документа, написаного з використанням мови розмітки

DOM - об’єктна модель документа

JSX - мова розширення JavaScript, яка дозволяє комбінувати JavaScript-код з розміткою, яка схожа на HTML

SPA - односторінковий додаток

URL - унікальна адреса веб ресурсу (сайту), яка зареєстрована в єдиній схемі адресації

SEO - пошукова оптимізація

JSON - текстовий формат передачі даних

ВСТУП

У сучасному світі інтернет є одним із найважливіших засобів спілкування та отримання інформації. Велика частина населення постійно шукає необхідну інформацію. Це особливо актуально в умовах інформаційного надміру, який став нормою нашого сучасного життя. З кожним днем кількість веб-додатків та веб-сайтів зростає, а разом з нею зростає і вимоги до їхніх інтерфейсів. Ефективність та зручність веб-інтерфейсів стають визначальними факторами успіху та конкурентоспроможності в сучасному цифровому середовищі.

Ця дипломна робота присвячена вивченню та оптимізації ефективності веб-інтерфейсів через використання технік та інструментів JavaScript. JavaScript, як одна з найпоширеніших мов програмування у веб-розробці, відіграє ключову роль у створенні динамічних та інтерактивних веб-інтерфейсів.

В рамках цієї дипломної роботи ми детально розглянемо поняття веб-інтерфейсу та його роль у веб-розробці. Проведемо аналіз основних принципів оптимізації веб-інтерфейсів та розглянемо різні техніки та інструменти JavaScript, які можуть бути використані для поліпшення їх ефективності.

Далі проведемо дослідження, спрямоване на вивчення впливу використання цих технік та інструментів на ефективність веб-інтерфейсів у реальних умовах. Результати цього дослідження нададуть цінні відомості та рекомендації для веб-розробників щодо оптимізації їхніх проектів.

Зрештою, ця дипломна робота має на меті внести вклад у підвищення якості та ефективності веб-інтерфейсів, що відіграють важливу роль у взаємодії користувачів з інформацією та сервісами в Інтернеті.

РОЗДІЛ 1. ВВЕДЕННЯ ДО ВЕБ-ІНТЕРФЕЙСІВ ТА JAVASCRIPT

1.1 Визначення поняття веб-інтерфейсу та його роль у веб-розробці

Для початку варто розібратись що означає поняття веб-інтерфейс. Це система взаємодії між користувачем та веб-додатком або веб-сайтом. Він представляє собою ту частину веб-додатка, яка відповідає за візуальне та функціональне спілкування користувача з системою через веб-браузер. Веб-інтерфейси стали надзвичайно популярними завдяки розширенню Інтернету та поширенню веб-браузерів.

Роль веб-інтерфейсу у веб-розробці вельми важлива. Він є основним інструментом, який забезпечує користувачам доступ до веб-додатків та інформації в Інтернеті. Тож варто розібратись які аспекти роблять роль веб-інтерфейсу настільки важливою у веб-розробці.

Варто зазначити, що веб-інтерфейс дає користувачам можливість взаємодіяти з веб-додатками та сайтами через веб-браузери на різних пристроях. Це створює можливість доступу до інформації та сервісів для широкого кола людей з різними потребами та можливостями. Гарний веб-інтерфейс сприяє зручності взаємодії користувача з веб-додатком чи сайтом. Його дизайн, навігація та розташування елементів повинні бути інтуїтивно зрозумілими та зручними для користувача. Веб-інтерфейс відповідає за візуальне відображення інформації користувачу. Це включає в себе правильне форматування тексту, використання зображень та графіків, а також відображення інтерактивних елементів, таких як кнопки та форми.

Також важливим фактором є функціональність та взаємодія з сервером. Веб-інтерфейс відповідає за передачу запитів користувача до сервера та отримання відповідей. Він забезпечує взаємодію з сервером шляхом виконання різних дій, таких як відправка форм, завантаження контенту, а також оновлення

інформації на сторінці без перезавантаження. Веб-інтерфейс повинен передбачати доступність та сумісність. Він повинен бути доступним для користувачів з різних пристроїв та браузерів. Повинен адаптуватися до різних розмірів екранів, операційних систем та технічних характеристик користувальницьких пристроїв.

В цілому, веб-інтерфейс виступає в якості основного інструменту для взаємодії користувача з веб-додатком чи сайтом, та відіграє важливу роль у забезпеченні зручності та ефективності цієї взаємодії. Його розробка та оптимізація є ключовими завданнями для веб-розробників з метою забезпечення високоякісного користувацького досвіду.

Ще одна важлива деталь, яку варто розглянути, це роль веб-інтерфейсу та які пункти вона передбачає. Першим пунктом можна вважати забезпечення доступу до функціональності. Веб-інтерфейс дозволяє користувачам виконувати різноманітні дії та операції, включаючи заповнення форм, натискання кнопок, перегляд даних, навігацію по веб-сторінках, взаємодію з різними елементами інтерфейсу, такими як меню, панелі інструментів, віджети, а також використання функціоналу, що надається веб-додатком. Наступним є покращення користувацького досвіду. Гарний веб-інтерфейс забезпечує зручну та ефективну взаємодію користувача з веб-додатком, що сприяє покращенню загального враження від використання продукту. Веб-інтерфейс відображає дані та контент користувачеві в зручному та зрозумілому форматі, дозволяючи легко знаходити та сприймати інформацію. Одним із важливих пунктів є забезпечення інтерактивності. Веб-інтерфейс може бути інтерактивним, надаючи користувачам можливість активно взаємодіяти з елементами сторінки. Наприклад, вони можуть перетягувати об'єкти, налаштовувати параметри, вибирати опції, вводити дані, взаємодіяти з анімацією та реагувати на події на сторінці. Ще одним пунктом є Підтримка різних пристроїв. Сучасні веб-інтерфейси повинні бути адаптивними та респонсивними, щоб забезпечувати оптимальний досвід користувача на різних пристроях і розмірах екрану.

Адаптивний дизайн веб-сайту забезпечує коректне та зручне відображення

сайту на будь-якому пристрої, незалежно від його розміру екрану. Цей тип дизайну автоматично адаптує розміри та компоненти сайту до розмірів вікна браузера чи пристрою, на якому відбувається перегляд. Тобто, вам не потрібно окремо налаштовувати веб-сайт для кожного типу пристрою, оскільки адаптивний дизайн забезпечує його коректне відображення на різних пристроях, таких як смартфони, планшети, ноутбуки чи навіть телевізори. При цьому користувач може зручно виконувати всі необхідні дії, такі як пошук, введення тексту, покупки тощо, без збільшення чи інших непередбачених змін в компонентах сайту.

Кожен успішний бренд або виробник розуміє, що ключ до успіху полягає в забезпеченні безперебійної роботи на всіх етапах свого бізнесу. Це стосується також його веб-сайту. Сьогодні більшість людей шукає інформацію за допомогою мобільних пристроїв, таких як смартфони та планшети, оскільки комп'ютер не завжди є під рукою. Тому, якщо ми відвідаємо веб-сайт бренду або виробника і не зможемо знайти те, що шукаємо через його неадаптивний дизайн, ми швидко покинемо цей сайт [25].

Розробка та підтримка інтерфейсу є доволі бюджетною та не довготривалою. Дозволяє уникнути проблем з просування, завдяки тому, що всі сторінки сайту доступні за одним і тим самим URL, що є дуже зручно і для користувача. Дуже красиво виглядає, та зберігає унікальний дизайн та повноцінну структуру.

Також не варто забувати про недоліки, їх обов’язково треба враховувати. Для того, щоб користувачу було зручно, можливо потрібно буде виключити певні елементи, для прикладу, прибрати якійсь зображення чи відео, сторінки сайту з адаптивним дизайном завантажуються трішки довше, ніж звичайні. Використовуючи адаптивний інтерфейс, ви отримуєте більший прибуток. Адже все більше людей частіше користуються мобільним телефоном для задоволення більшості потреб. Відповідно, якщо ваш сайт буде зручним та красивим, ви отримаєте більше клієнтів. Адаптація сайту під мобільні пристрої також означає, що сайт буде легше просувати, адже Google ранжує його набагато краще. Не

втрачайте такий необхідний для вас трафік, зробіть свій сайт адаптивним під мобільний вже зараз.

Усі ці функції та ролі роблять веб-інтерфейс ключовим елементом успішного веб-додатка або сайту, що вимагає уваги та професійного підходу до його розробки та оптимізації.

1.2 Фактори, що впливають на ефективність веб-інтерфейсу

Наразі пропоную розглянути фактори, які впливають на ефективність веб-інтерфейсу. На мою думку варто розпочати з швидкодії веб-інтерфейсу. Вона визначається часом відгуку на дії користувача, завантаженням сторінок та обробкою запитів. Чим швидше веб-сайт чи додаток відповідає на дії користувача та завантажує необхідний контент, тим краще він відповідає його очікуванням та задовольняє його потреби.

Відповідність користувальницьким потребам - це ключовий аспект успішного веб-інтерфейсу, який визначається здатністю інтерфейсу задовольняти потреби та очікування користувачів. Ефективний веб-інтерфейс повинен відповідати потребам та очікуванням користувачів. Це означає, що він повинен бути легко зрозумілим, інтуїтивно зрозумілим та відповідати їхнім конкретним потребам. Давайте розглянемо цей аспект більш детально з прикладами:

- легкість зрозуміння. Ефективний веб-інтерфейс повинен бути легко зрозумілим для користувачів, навіть для тих, хто вперше з ним стикається. Наприклад, меню навігації повинно бути структурованим та легкодоступним, щоб користувачі могли швидко знайти потрібну інформацію без зайвих зусиль;

- інтуїтивна навігація. Веб-інтерфейс повинен мати інтуїтивно зрозумілу навігацію, яка дозволяє користувачам легко переміщатися по сайту або додатку. Наприклад, наявність яскравих та чітких кнопок "Назад" або "Далі", а також логічне розташування елементів допомагає забезпечити зручну навігацію;

- відповідність контенту. Важливо, щоб вміст веб-інтерфейсу відповідав потребам цільової аудиторії. Наприклад, для веб-сайту з магазином електроніки, користувачі очікують бачити інформацію про товари та їх характеристики, а також можливість замовлення товару, тому інтерфейс повинен це враховувати;

- персоналізація. Веб-інтерфейс може бути ефективним, якщо він надає можливості персоналізації для користувачів. Наприклад, веб-сайти можуть пропонувати персоналізовані рекомендації товарів або послуг, враховуючи історію покупок або інтереси користувача.

Забезпечення відповідності користувальницьким потребам дозволяє створювати веб-інтерфейси, які максимально відповідають очікуванням та потребам користувачів, що сприяє їх задоволенню та позитивному досвіду використання.

Зручність взаємодії з веб-інтерфейсом включає в себе багато аспектів, таких як доступність елементів управління, простота навігації та зрозумілість використання форм і кнопок. Додатково, наявність допоміжних елементів, наприклад, підказок та пояснень, покращує досвід користувача та робить взаємодію з інтерфейсом більш приємною. Естетика та дизайн веб-інтерфейсу грають важливу роль у сприйнятті користувачем. Привабливий та професійно оформлений дизайн сприяє створенню позитивного враження від взаємодії з сайтом або додатком. Він може покращити користувацький досвід та збільшити ймовірність повернення користувачів на сайт у майбутньому.

Сумісність з різними пристроями та браузерами є важливим аспектом веб-інтерфейсу. Адаптивність та респонсивність дозволяють інтерфейсу оптимально виглядати та працювати на будь-якому пристрої та в будь-якому браузері. Це забезпечує зручний та приємний досвід користувача, незалежно від того, який пристрій або браузер вони використовують. Сумісність з різними пристроями та браузерами є критичним аспектом успішного веб-інтерфейсу в сучасному цифровому світі. Оскільки користувачі мають доступ до Інтернету через різноманітні пристрої, такі як комп'ютери, ноутбуки, планшети та смартфони, а також використовують різні браузери, веб-сайти та додатки повинні бути

адаптовані для оптимального відображення на будь-якому пристрої та у будь-якому браузері.

Адаптивний та респонсивний дизайн веб-інтерфейсу дозволяє автоматично адаптувати його до різних розмірів екранів та характеристик пристроїв, що використовуються користувачем. Ось деякі ключові аспекти сумісності з різними пристроями та браузерами:

Адаптивний дизайн. Цей підхід передбачає використання гнучких макетів та грідів, які автоматично перегруповують та адаптують контент до різних розмірів екранів. Наприклад, елементи меню можуть переходити у випадаючі списки на малих екранах, щоб забезпечити зручну навігацію на мобільних пристроях. Респонсивний дизайн використовує CSS медіа-запити для адаптації стилів та вигляду веб-інтерфейсу до різних пристроїв. Наприклад, текст та зображення можуть змінювати свій розмір та розташування в залежності від розміру екрану [9].

Рисунок 1.1 – Приклад адаптацію веб-сторінки під різні пристрої

Тестування на різних пристроях та браузерах. Веб-розробники повинні регулярно проводити тестування своїх проектів на різних пристроях та в різних браузерах, щоб переконатися, що вони працюють належним чином на всіх

платформах. Для цього часто використовують спеціалізовані сервіси або інструменти для хмарного тестування. Підтримка відносних одиниць та гнучкості в розмітці. Використання відносних одиниць в CSS, таких як відсотки або em, дозволяє створювати більш гнучкі та адаптивні макети, які добре працюють на різних пристроях.

Сумісність з різними пристроями та браузерами є важливим фактором для забезпечення доступності та задоволення потреб користувачів у сучасному цифровому середовищі. Це дозволяє забезпечити оптимальний досвід взаємодії незалежно від того, який пристрій або браузер використовує користувач. Врахування цих факторів дозволяє створити ефективний та успішний веб-інтерфейс, який задовольняє потреби користувачів та сприяє досягненню поставлених цілей веб-проекту.

1.3 Огляд основних принципів оптимізації веб-інтерфейсів та їх ролі у поліпшенні ефективності користування

Розглянемо основні принципи оптимізації веб-інтерфейсів. Забезпечення контролю та свободи дій є ключовим аспектом веб-інтерфейсів. Надання можливості користувачам скасовувати свої дії на сайті, схоже на наявність кнопки "1" у ліфті – вона завжди необхідна. Часто користувачі роблять скасування кроків, тому важливо забезпечити їм таку можливість, не ховаючи кнопку або посилання, оскільки це може викликати негативну реакцію і змусити їх покинути сайт. Також важливо враховувати принцип дублювання дій.

Стандартизація та послідовність грають важливу роль у покращенні ефективності користування. Використання веб-інтерфейсу, що відповідає загально прийнятим стандартам, допомагає користувачам швидше орієнтуватися на сайті, оскільки вони вже знайомі з подібними структурами. Простота та послідовність в оформленні сторінок, розділів та функцій дозволяють користувачам легше знаходити потрібну інформацію.

Попередження про можливі помилки та надання користувачам відповідних рекомендацій також важливі для покращення ефективності веб-інтерфейсу. Надання інформації про потенційні проблеми та шляхи їх вирішення дозволяє уникнути неприємностей для користувачів і сприяє позитивному досвіду використання.

Естетика та простота важливі для створення ефективного веб-інтерфейсу. Лаконічний дизайн, який не завантажений зайвими деталями, дозволяє користувачам легше фокусуватися на основній інформації і виконувати потрібні дії без зайвих зусиль.

Гнучкість та ефективність є ключовими принципами оптимізації веб-інтерфейсу. Забезпечення різних варіантів виконання дій та автоматичне заповнення форм дозволяє користувачам вибирати найзручніший спосіб взаємодії з сайтом.

Усі ці принципи сприяють поліпшенню ефективності веб-інтерфейсів і забезпечують користувачам зручний, швидкий та задовільний досвід користування.

З кожним роком розкручування в системах пошуку стає все більш складним процесом, але є обмежена кількість змінних, які мають досить помітний вплив на позиції сайту у видачі. В основному це ключові слова, структура платформи, посилання. Незважаючи на постійні зміни, ці параметри залишаються ключовими для успішного просування [30].

Структура сайту має велике значення для підвищення його рейтингу у пошукових системах. Важливо мати унікальний вміст та організовану URL-структуру. Надмірний контент та невпорядковані URL можуть негативно вплинути на позиції в пошукових видачах. При запуску або перезапуску сайту важливо приділити увагу його архітектурі, оскільки це є першим кроком до успішного продвиження.

Посилання, що ведуть на вашу веб-сторінку з інших джерел Інтернету,

відомі як вхідні посилання або "зворотні посилання". Вони мають значний вплив на позиції вашого сайту у рейтингу пошукових систем і можуть визначати його видимість у пошукових видачах.

Один з важливих аспектів є вибір теми, яка визначає основну тематику сторінки та її позиції у результатах пошуку. Теми грають вирішальну роль у формуванні авторитету веб-сторінки, що визначає її рейтинг у пошукових системах. Чим більше контенту публікується на певну тему, тим більше відповідний елемент буде високо ранжований у пошукових видачах [11].

Популярні теми грають ключову роль у привертанні уваги пошукових систем до веб-сайту. Однак використання конкретних ключових слів також має велике значення для підвищення видимості сайту в пошукових результатах. Врахування цих ключових слів є критичним аспектом в стратегії SEO, оскільки вони допомагають пошуковим системам зрозуміти тематику та спрямованість веб-сторінки. Таким чином, оптимізація ключових слів є не менш важливою, ніж обробка тем, і варто враховувати при розробці SEO-стратегії для сайту.

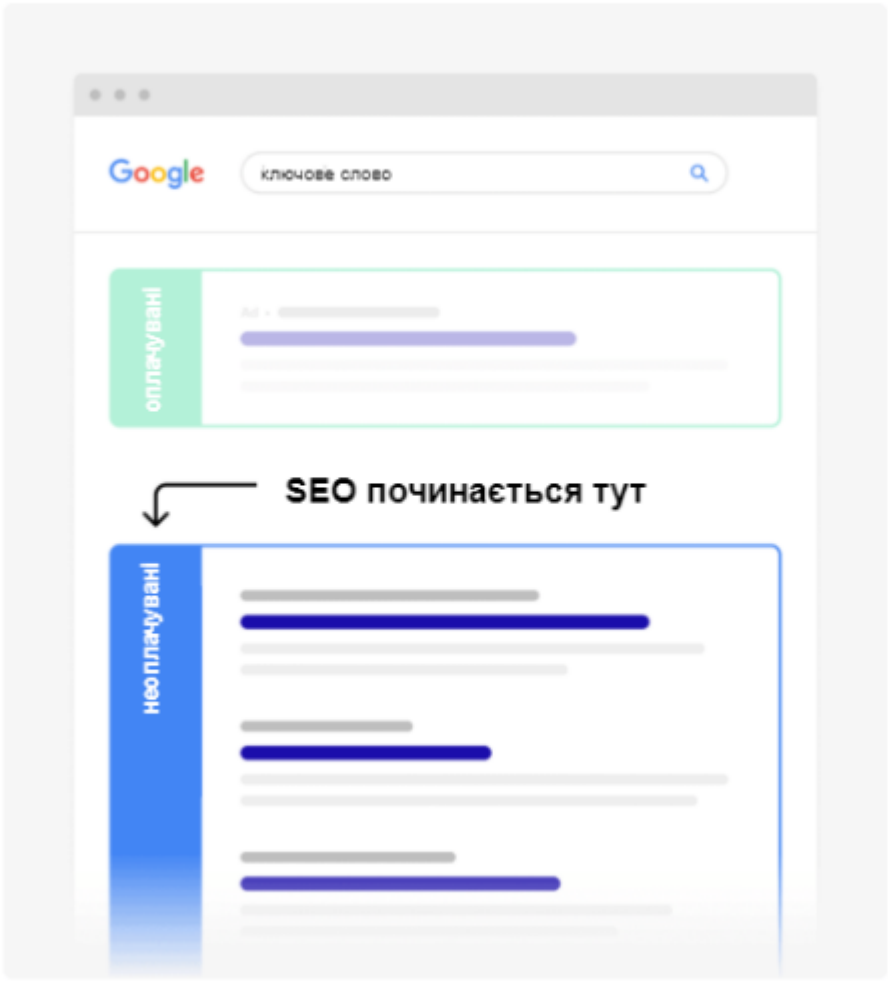


Рисунок 1.2 – Приклад розділів пошукової системи

Швидкість сторінки визначається тим, як оперативно web-сторінка завантажується, коли користувач переходить на неї з результатів пошуку або з інших джерел. Це ключовий аспект, який впливає на позиції сайту у рейтингу пошукових систем. Чим швидше сторінка завантажується, тим вище ймовірність того, що вона отримає вищі позиції у пошуковій видачі.

Зазвичай сторінки, які завантажуються менше, ніж за три секунди, вважаються досить швидкими. Однак, швидкість завантаження може бути покращена за допомогою різних методів оптимізації, таких як компресія зображень, мінімізація коду та використання кешування, що може позитивно вплинути на позиції сайту у пошукових системах.

Підсумовуючи, можна сказати що оптимізація веб-інтерфейсів є ключовим елементом для поліпшення ефективності користування. Основні принципи оптимізації, такі як контроль та свобода дій, стандартизація та послідовність, упередження помилок, наочність, гнучкість та ефективність, естетика та простота, розуміння проблем та вирішення, та довідки та документація, сприяють створенню інтуїтивно зрозумілих, ефективних і зручних у використанні інтерфейсів, допомагають забезпечити, що користувачі зможуть легко взаємодіяти з веб-сайтом або додатком, швидко знаходити потрібну інформацію та виконувати необхідні дії без зайвих зусиль чи плутанини. Такий підхід допомагає покращити загальний досвід користувача і забезпечити більшу задоволеність від використання веб-ресурсів.

1.4 Огляд мови програмування JavaScript

JavaScript – динамічна, об'єктно-орієнтована прототипна мова програмування. Головною особливістю являється те, що JavaScript – це мова програмування, що дозволяє зробити Web-сторінку інтерактивною, тобто такою, що реагує на дії користувача. Виходячи з концепції програмування JavaScript має надзвичайно багато застосувань.

Тож давайте повернемось трохи назад і дізнаємось про її історію створення. Історія розвитку інформаційних технологій свідчить про безперервний пошук нових засобів та мов програмування для полегшення створення інтерактивних та динамічних веб-додатків. В 90-ті роки XX століття була відчутна необхідність у мові програмування, яка би дозволяла створювати більш динамічні та інтерактивні веб-сторінки [26].

В кінці 80-х років XX століття компанія Netscape Communications Corporation, яка спеціалізувалася на розробці веб-браузерів, почала розробку нового веб-браузера під назвою Netscape Navigator. Однією з ключових функцій цього браузера була можливість виконання скриптів на сторінках.

Під час розробки Netscape Navigator з'явилася необхідність у мові програмування, яка би дозволяла додавати скрипти до веб-сторінок. Так народилася ідея про створення мови сценаріїв для браузера. У 1995 році, у підприємства Netscape, команда програмістів під керівництвом Брендана Айка взялася за розробку нової мови програмування. Розробка нової мови програмування розпочалася в 1995 році і тривала всього кілька тижнів. Мову назвали Mocha, а пізніше перейменували у LiveScript. У той же рік, разом з випуском першої версії браузера Netscape Navigator 2.0, LiveScript було офіційно представлено як єдина мова для програмування веб-сторінок у цьому браузері.

У той же час, у зв'язку з партнерством між Netscape та компанією Sun Microsystems, мову LiveScript перейменували у JavaScript. Це було стратегічним кроком, спрямованим на залучення уваги до мови за рахунок популярності Java, яка в той час була дуже популярною. JavaScript став стандартом для веб-програмування, а його вплив на інтернет став надзвичайно великим.

JavaScript став ключовим елементом веб-технологій, дозволяючи створювати багатофункціональні та інтерактивні веб-додатки. Його створення відбулося на хвилі потреби в інструменті для створення динамічних сторінок, і він змінив обличчя Інтернету, відкривши нові можливості для розробників та користувачів.

На сьогоднішній день JavaScript доволі компактна та гнучка мова. Є можливість створити «каруселі», галереї зображень, динамічні макети сторінок, відповіді на натиски кнопок, обробка тексту тощо. Також є можливість створювати ігри, 2D та 3D графіку, реалізувати роботу з використанням баз даних та багато іншого. Розробники забезпечили велике розмаїття інструментів, що доповнюють основу мови JavaScript, які відкривають величезну кількість додаткового функціоналу з мінімальними зусиллями. Серед них: -Програмні інтерфейси (APIs) для браузерів – API, які вбудовані у браузери, що надають функціонал на зразок динамічного створення HTML та застосування CSS-стилів, збір та обробка відео-потоків з веб-камери користувача, генерація 3D-графіки та аудіо-семплів [3].

- API третіх осіб, що дозволяють розробникам інтегрувати у власні сайти функціонал інших провайдерів, таких як Twitter або Facebook.
- фреймворки та бібліотеки третіх осіб, які ви можете застосувати до вашого HTML, щоб прискорити створення сайтів та застосунків.

JavaScript – об'єктно-орієнтована скриптова мова програмування і є діалектом мови ECMAScript. Найчастіше використовується для створення сценаріїв веб-сторінок, що надає можливість на стороні клієнта (пристрої кінцевого користувача) взаємодіяти з користувачем, керувати браузером, асинхронно обмінюватися даними з сервером, змінювати структуру та зовнішній вигляд веб-сторінки. JavaScript зазвичай використовується як вбудована мова для програмного доступу до об'єктів додатків. Найбільш широке застосування знаходить у браузерах як мова сценаріїв для надання інтерактивності вебсторінкам. Послідовність інструкцій (що називається програмою, скриптом або сценарієм) виконується інтерпретатором, вбудованим в звичайний Web-браузер. Іншими словами, код програми вбудовується в HTML - документ і виконується на боці клієнта. Для виконання програми не потрібно навіть перезавантажувати Webсторінку, всі програми виконуються в відповідь на будь-яку подію. Наприклад, перед відправленням даних форми можна перевірити їх на допустимі значення і, якщо значення не відповідають очікуваним, заборонити відправлення даних. Недоліками є відсутня жорстка типізація даних. Поки виконання коду не дійде до потрібного рядка, не взнаєш чи вона працює. Адже значну частину з пошуку помилок міг би взяти на себе компілятор, якби знав типи даних, з якими він працює. І також це незвична для багатьох програмістів об'єктна модель. Класи і наслідування класів присутні, але вони сильно відрізняється від звичної багатьом реалізацій в мовах програмування C ++ / C # / Java. Зараз JavaScript – єдина мова програмування для браузерів. Вона працює під Windows, macOS, Linux і на мобільних платформах. JavaScript розвивається кожного дня, відкриває нові напрямки та технології для її використання.

Але сувора реальність, як зазначено вище, полягає в тому, що JavaScript – це єдина мова, яка працює всередині браузера. А браузер – це найпопулярніша

платформа в інтернеті в наш час. Тому попит на frontend-розробників постійно зростає. Ба більше, їхні зарплати вже досягли зарплат backend-розробників. Але заздрити тут нема чому. Будь-який frontend-проєкт через рік перетворюється на пекло. Адже у фронтенді є ще й обмеження на розмір коду, оскільки код завантажується браузером, і це впливає на швидкість завантаження сторінки.

Щоб якось зменшити безлад у своїх проєктах, фронтендери постійно пишуть нові фреймворки, які спрощують їхнє життя. І звичайно ж, ці фреймворки застарівають буквально за 3-5 років. Якщо 5 років тому хтось вирішив написати свій проєкт на суперсучасному фреймворку, то сьогодні про нього будуть говорити, що він стародавній, як мамонти, і як таке взагалі можна використовувати.

Але є й хороші новини: винайшли нову мову на заміну JavaScript – це TypeScript. Вона дуже гарна, у ній є типізація, класи, області видимості. До того ж, є спеціальний компілятор, який вміє компілювати TypeScript в JavaScript.

Усі великі frontend-проєкти використовують TypeScript замість JavaScript. Ба більше, багато сучасних frontend-фреймворків замість JavaScript використовують TypeScript. Наприклад, Angular, на якому написано фронтенд JavaRush.

JavaScript є однією з найпопулярніших мов програмування, яка використовується для створення динамічних інтерактивних веб-додатків. Вона забезпечує можливість додавати різноманітні функціональність та взаємодію на веб-сторінках, що робить їх більш привабливими та корисними для користувачів.

1.5 Роль JavaScript у розробці Веб-додатків

JavaScript є однією з найбільш важливих мов програмування для веб-розробки. Його універсальність і можливості забезпечують широкий спектр функціональності для створення інтерактивних та динамічних веб-додатків. Це означає, що розробники можуть легко додавати анімацію, слайдшоу, форми для

збору даних та інші елементи, які зроблять веб-сайт більш привабливим та зручним для користувачів. Наприклад, анімація може зробити взаємодію з сайтом більш цікавою та приємною для користувача. Вона може використовуватися для підсвічування важливих елементів, візуального підтвердження дій користувача або створення ефектів переходу між сторінками.

Крім того, JavaScript дозволяє створювати динамічні веб-додатки, які реагують на дії користувачів без необхідності перезавантаження сторінки. Наприклад, ви можете створити інтерактивний калькулятор, який автоматично розраховує результати при зміні вхідних даних, або ж онлайн-ігри, які відрізняються відповідно до дій гравця. Також JavaScript використовується для взаємодії з Document Object Model (DOM) - це інтерфейс, який представляє структуру та вміст веб-сторінки в формі дерева об'єктів. Оскільки весь контент сторінки, включаючи тексти, зображення та інші елементи, представлений як об'єкти у DOM, JavaScript може маніпулювати цими об'єктами, змінюючи їх властивості або структуру [4].

Це означає, що розробники можуть додавати нові елементи до сторінки, видаляти чи приховувати існуючі елементи, змінювати їх стилі та властивості, а також реагувати на події, такі як клікання мишею чи натискання клавіш. Взаємодія з DOM дозволяє створювати динамічний та змістовно багатший веб-інтерфейс, який може адаптуватися до різних умов та взаємодіяти з користувачем більш ефективно.

Одним із важливих факторів є робота з AJAX. JavaScript дозволяє використовувати AJAX для отримання даних з сервера без перезавантаження сторінки. Це дозволяє вам забезпечувати більш швидку та ефективну роботу веб-додатків. Наприклад, ви можете отримувати дані з бази даних та відображати їх на сторінці без перезавантаження сторінки [2].

Валідація форм JavaScript дозволяє використовувати валідацію форм для забезпечення правильного введення даних користувачем на веб-сторінках. Валідація форм є важливим елементом веб-розробки, який допомагає уникнути неправильного введення даних, що може призвести до непередбачуваних

результатів, помилок в обробці даних і проблем з безпекою

Крім того, JavaScript використовується для створення веб-додатків, які працюють в режимі реального часу. Наприклад, додаток для спілкування в чаті, який автоматично оновлюється з новими повідомленнями, без необхідності перезавантажувати сторінку.

JavaScript відіграє ключову роль у розробці веб-додатків, надаючи широкі можливості для створення інтерактивних, динамічних та ефективних веб-застосунків. Його універсальність та широкий спектр функціональності роблять його невід'ємною частиною сучасної веб-розробки. Його мета – надати розробникам зручний набір інструментів для створення ефективних і масштабованих веб-додатків. Фреймворк js забезпечують структуру й організацію коду, а також надають готові рішення для завдань, які часто зустрічаються, таких як управління станом, маршрутизація, обробка подій і взаємодія із сервером.

Також він дозволяє розширювати можливості браузера, додаючи нові функції та функціональність. За допомогою JavaScript можна створювати графіки, анімацію, перетягування об'єктів, геолокацію та інші функції, які покращують взаємодію з користувачем.

JavaScript має одне з найбільших співтовариств розробників у світі. Існує безліч веб-фреймворків, бібліотек та інструментів, які розширюють можливості JavaScript. Це дозволяє розробникам використовувати готові рішення, прискорюючи розробку і полегшуючи тестування.

Узагалі, JavaScript є потужним інструментом для створення динамічного та інтерактивного контенту на веб-сторінках. Він дозволяє розробникам додавати нові функції, ефекти та інтерактивність до веб-сторінок, що поліпшує користувацький досвід та забезпечує більшу ефективність роботи додатків. Тому використання JavaScript є важливою частиною веб-розробки і рекомендується для використання в будь-якому проекті веб-розробки.

1.6 Опис базових концепцій та функцій JavaScript

У цьому підпункті пропоную розглянути опис базових концепцій та функцій JavaScript.

JavaScript надає потужний інструментарій для розробки веб-додатків, що включає різноманітні конструкції та функції для роботи з даними, взаємодії з користувачем та іншими аспектами веб-розробки. Це робить JavaScript важливою та необхідною мовою для веб-розробників у сучасному світі. Як і кожна мова програмування вона має змінні та типи даних

JavaScript дозволяє створювати змінні для зберігання різних типів даних, таких як рядки, числа, булеві значення тощо.

Типи даних включають рядки (strings), числа (numbers), булеві значення (booleans), масиви (arrays), об'єкти (objects), функції (functions) та інші.

JavaScript дозволяє працювати з різноманітними типами даних, такими як рядки, числа, булеві значення, масиви, об'єкти та інші. Змінні в JavaScript можуть зберігати значення різних типів даних, і їхні типи можуть змінюватися під час виконання програми.

Рядки (Strings) - це послідовність символів, таких як літери, цифри та спеціальні символи, яка записується у лапках (одинарних або подвійних). Наприклад, "John" або 'Hello, world!'.

Числа (Numbers) - вони можуть бути цілими або з плаваючою комою і використовуються для виконання математичних операцій. Наприклад, 30 або 9.99.

Булеві Значення (Booleans) - булеві значення відображають логічні стани true або false і використовуються для умовних виразів та логічних операцій. Наприклад, true або false.

Масиви (Arrays) - це структура даних, що дозволяє зберігати послідовність елементів у впорядкованому списку. Наприклад, [1, 2, 3, 4, 5] або ["red", "green", "blue"].

Об'єкти (Objects) - це колекція пар ключ-значення, де значення можуть бути будь-якого типу даних. Вони використовуються для зберігання структурованої інформації. Наприклад, {name: "John", age: 30, isStudent: false}.

Null - це спеціальне значення в JavaScript, яке позначає відсутність або

пустоту. Воно використовується для явного позначення відсутності значення. Наприклад, якщо ви хочете сказати, що змінна не має значення, ви можете присвоїти їй значення null.

Undefined - це ще один спеціальний тип даних в JavaScript, який вказує на те, що значення не було присвоєне. У випадку, коли змінна була оголошена, але їй не було присвоєно значення, її значення буде undefined.

Основна відмінність між null та undefined полягає в їхньому використанні: null використовується як виразне позначення відсутності значення, тоді як undefined вказує на те, що значення ще не було присвоєне змінній.

Оператори:

- JavaScript має різноманітні оператори для виконання арифметичних операцій, порівнянь, логічних операцій та інших дій;
- приклади операторів включають арифметичні оператори (+, -, *, /), оператори порівняння (==, ===, !=, !==), логічні оператори (&&, ||, !) тощо.

В JavaScript оператори використовуються для виконання різних операцій над змінними та значеннями.

Арифметичні оператори - вони використовуються для виконання арифметичних операцій, таких як додавання, віднімання, множення та ділення. Наприклад:

- Додавання: `+`
- Віднімання: `-`
- Множення: `*`
- Ділення: `/`

Оператори порівняння - вони порівнюють два значення та повертають булеве значення true або false. Наприклад:

- Рівність: `==`
- Нерівність: `!=`
- Більше: `>`
- Менше: `<`
- Більше або рівне: `>=`

- Менше або рівне: `<=`

Логічні оператори - вони використовуються для об'єднання та порівняння логічних значень. Наприклад:

- Логічне І: `&&`
- Логічне АБО: `||`
- Логічне НЕ: `!`

Оператори присвоювання – вони використовуються для присвоєння значень змінним. Наприклад:

- Присвоєння: `=`
- Додати та присвоїти: `+=`
- Відняти та присвоїти: `-=`
- Множити та присвоїти: `*=`
- Ділити та присвоїти: `/=`

Інші оператори – вони включають у себе тернарний оператор, оператори інкремента/декремента та інші спеціальні оператори для роботи зі змінними та значеннями. Наприклад:

- Тернарний оператор: `condition ? value1 : value2`
- Інкремент: `++`
- Декремент: `--`

Використання цих операторів дозволяє здійснювати різноманітні операції та управляти потоком програми в JavaScript.

Умовні Конструкції - умовні конструкції, такі як if, else if, else та switch, є важливою частиною будь-якої мови програмування, включаючи JavaScript. Вони дозволяють програмі реагувати на різні умови та виконувати відповідні дії в залежності від цих умов.

Конструкція if використовується для перевірки умови і виконання коду, якщо ця умова є істинною. Конструкція else if дозволяє перевірити іншу умову, якщо перша не виконується, і виконати відповідний код. Конструкція else виконується, якщо жодна з попередніх умов не є істинною.

Конструкція switch використовується, коли потрібно перевірити значення

однієї змінної на відповідність різним варіантам значень. Кожен варіант може мати свій власний блок коду для виконання.

Умовні конструкції дозволяють реалізувати різноманітну логіку в програмах, такі як керування потоком виконання програми, обробка різних варіантів даних та вирішення складних завдань в залежності від умови.

Функції - надає можливість визначати функції, які є фрагментами коду призначеними для виконання певних завдань. Визначення функції містить блок коду, який виконується, коли функція викликається. Функції можуть приймати параметри, які використовуються у внутрішньому коді функції для виконання певних операцій. Це дозволяє функціям бути більш універсальними та придатними для використання в різних контекстах.

Крім того, функції можуть повертати значення, що робить їх дуже корисними для організації коду та повторного використання. Значення, яке повертається функцією, може бути використане у іншій частині програми або передане як аргумент у іншу функцію.

Масиви та Об'єкти

- JavaScript має вбудовану підтримку для масивів та об'єктів, що дозволяє зберігати та обробляти дані в структурованому вигляді;
- масиви використовуються для зберігання списків елементів, тоді як об'єкти дозволяють зберігати дані у вигляді пар ключ-значення.

Обробники Подій - JavaScript дозволяє прив'язувати обробники подій до елементів веб-сторінки, що дозволяє реагувати на дії користувачів, такі як кліки, наведення миші, введення тексту тощо [14].

Узагальнюючи, використання функцій дозволяє покращити організацію коду, зменшити повторення та полегшити управління складністю програми. JavaScript надає потужний інструментарій для розробки веб-додатків, що включає різноманітні конструкції та функції для роботи з даними, взаємодії з користувачем та іншими аспектами веб-розробки. Це робить JavaScript важливою та необхідною мовою для веб-розробників у сучасному світі.

1.7 Переваги та обмеження використання JavaScript у веб-розробці

Використання JavaScript у веб-розробці має безліч переваг, але також має свої обмеження. Давайте розглянемо деякі з них:

Динамічність та інтерактивність. JavaScript дозволяє створювати динамічні та інтерактивні веб-сторінки, які реагують на дії користувачів без перезавантаження сторінки. Це покращує користувацький досвід та забезпечує більшу залученість.

Асинхронність - JavaScript підтримує асинхронне програмування, що дозволяє виконувати операції без блокування основного потоку веб-прикладу. Це робить можливим виконання задач на фоні і покращує швидкість реагування веб-сторінок.

Багатофункціональність - JavaScript має широкий спектр функцій та бібліотек, які дозволяють виконувати різні завдання, від анімації та валідації форм до взаємодії з сервером без перезавантаження сторінки.

Крос-платформенність - JavaScript може виконуватися у браузері на будь-якому пристрої та операційній системі, що робить його ідеальним для розробки веб-додатків, що працюють на різних пристроях [1].

Також варто було б знати чи важко вчити цю мову. Фахівці відзначають, що дана мова досить проста у вивченні для новачків й самостійно вивчити основи JavaScript можна за 3-4 місяці. Після цього вам буде простіше освоювати інші мови, зокрема, PHP. Також Джава скрипт має широке застосування і популярність в інтернеті. Вивчивши його, ви зможете заробляти на створенні та доопрацюванні скриптів для сайтів. Він містить всі фундаментальні речі: алгоритми, об'єктно-орієнтовану модель і структури даних. Писати програми цією мовою можна в будь-якому текстовому редакторі, навіть блокноті. Актуальність і потреба мови навряд чи буде знижуватися найближчим часом.

Сьогодні JavaScript працює не тільки в браузері, а й на сервері або будь-якому іншому пристрої за допомогою спеціальної програми під назвою JavaScript

Mechanical. Браузер має вбудований «движок», який іноді називають віртуальною машиною JavaScript. Різні двигуни мають різні «кодові назви». приклад:

• V8 - в Chrome і Opera.

• SpiderMonkey - у Firefox.

Сучасний JavaScript є «безпечною» мовою програмування. Низькорівневий доступ до пам’яті або ЦП не надається, оскільки він розроблений для браузерів, яким це не потрібно.

Але враховуючи всі плюси не варто забувати про обмеження які має ця мова. Одним із таких є залежність від клієнтського браузера: JavaScript виконується в середовищі клієнтського браузера, тому функціональність додатка може обмежуватися можливостями конкретного браузера. Одним із важливих факторів є безпека. Використання JavaScript може створити потенційні загрози безпеці, такі як вразливості XSS (Cross-Site Scripting), якщо не використовувати належні заходи захисту.

Продуктивність – погано оптимізований JavaScript код може призвести до погіршення продуктивності веб-сторінок, збільшення часу завантаження та затримок у відгуку. Ще одна із причин, яка може бути це наявність JavaScript існують ситуації, коли JavaScript може бути вимкнений у браузері користувача, що призведе до непрацездатності деяких функцій веб-додатка.

РОЗДІЛ 2. УДОСКОНАЛЕННЯ ТА ОПТИМІЗАЦІЯ ВЕБ-ІНТЕРФЕЙСІВ: МЕТОДИ ТА ЗАСОБИ

2.1 Огляд різних методів оптимізації швидкодії веб-інтерфейсів, Local та Session Storage

Оптимізація швидкодії веб-інтерфейсів є важливим аспектом розробки веб-додатків, оскільки впливає на користувацький досвід. JavaScript відіграє ключову роль у цьому процесі.

Деякі з динамічних удосконалень веб-сайтів, які виконуються JavaScript:

- автозаповнення;
- завантаження нового контенту або даних на сторінку без перезавантаження сторінки;
- ефекти перекидання та випадаючі меню;
- анімація елементів сторінки, таких як вицвітання, зміна розміру або переміщення;
- відтворення аудіо та відео;
- перевірка введення з веб-форм;
- виправлення проблем з сумісністю браузера.

Хоча JavaScript є мовою на стороні клієнта, деякі найпотужніші функції включають асинхронну взаємодію з віддаленим сервером. Асинхронність просто означає, що JavaScript здатна взаємодіяти з сервером у фоновому режимі, не перериваючи взаємодії користувача, що відбувається на передньому плані. Сьогодні майже всі пошукові системи мають функцію автозаповнення. Користувач починає вводити слово у вікно пошуку, а нижче перелік можливих пошукових термінів або фраз. Запропоновані пошукові терміни з’являються без перезавантаження сторінки. У фоновому режимі JavaScript зчитує літери за типом користувача, відправляє ці листи на віддалений сервер, а сервер надсилає

пропозиції назад [18].

Програмне забезпечення на стороні сервера аналізує слова та запускає алгоритми, щоб передбачити пошуковий термін користувача. Такі програми дуже великі та складні. JavaScript на машині клієнта максимально простий і маленький, щоб не сповільнювати взаємодію користувача. Зв'язок між JavaScript та серверною програмою обмежується пропускною здатністю користувача. Ось чому розробники надають пріоритет ефективності функцій JavaScript і роблять обсяг даних, переданих між програмами, якомога меншим. Лише після того, як користувач вибрав пошуковий термін, вся сторінка перезавантажується та видає результати пошуку. Такі двигуни, як Google, зменшили або усунули необхідність перезавантажувати навіть для цього кроку. Вони просто дають результати, використовуючи той самий асинхронний процес .

Одним із основних методів оптимізації роботи веб-сторінки є збереження інформації на стороні користувача. Пропоную розібратись із поняттям Local Storage та Session Storage та як це працює.

Local Storage - це механізм, який дозволяє зберігати дані локально в браузері користувача. Це надзвичайно корисний метод оптимізації, особливо для збереження та доступу до невеликих об'ємів даних без необхідності постійної взаємодії з сервером. Ось детальний огляд цього методу оптимізації:

Дані недоступні серверу- це дані, збережені в Local Storage, доступні тільки в межах браузера користувача і недоступні серверу. Це означає, що вони не відправляються на сервер при кожному запиті, що зменшує обсяг трафіку та ресурси, необхідні для обробки даних [7].

Зберігання в браузері. Дані Local Storage зберігаються непосредственно в браузері користувача. Це означає, що вони залишаються доступними навіть після закриття веб-сайту або перезавантаження сторінки, що дозволяє зберігати стан додатка або інші важливі дані.

Значення - лише рядок. Одним з обмежень Local Storage є те, що він може зберігати лише рядкові значення. Це означає, що будь-які дані, які ви зберігаєте в Local Storage, повинні бути перетворені на рядок перед збереженням. Проте це не

є великим недоліком, оскільки JSON може бути легко серіалізований та десеріалізований у рядок та навпаки.

Обсяг даних, який можна зберегти в Local Storage, обмежений приблизно 5 МБ на домен. Це означає, що для збереження великих об'ємів даних Local Storage може бути не найкращим вибором, і у таких випадках можуть бути використані інші методи, такі як IndexedDB.

Local Storage є потужним інструментом для зберігання та доступу до невеликих обсягів даних без необхідності постійного взаємодії з сервером. Він допомагає покращити швидкодію додатків та забезпечити кращий користувацький досвід, зберігаючи важливі дані локально на клієнтському пристрої [19].

Session Storage - це ще один механізм для зберігання даних локально в браузері користувача, проте він має кілька відмінностей від Local Storage. Ось детальний огляд цього методу оптимізації:

Аналогічність з Local Storage полягає у тому, що Session Storage подібний до Local Storage у тому сенсі, що обидва ці механізми дозволяють зберігати дані локально в браузері користувача. Вони обидва доступні лише на клієнтському боці і не відправляються на сервер.

Тривалість зберігання даних це основна відмінність між Session Storage та Local Storage, вона полягає в тривалості зберігання даних. Дані, збережені в Session Storage, існують лише до моменту закриття вкладки або браузера. Якщо користувач закриє вкладку або браузер, дані в Session Storage будуть втрачені, що робить його ідеальним для збереження тимчасових даних, які не потрібно зберігати на тривалий термін [20].

Подібно до Local Storage, Session Storage також зберігає лише рядкові значення. Це означає, що дані повинні бути серіалізовані у рядок перед збереженням та десеріалізовані назад при отриманні. JSON є одним з найпоширеніших форматів для цієї операції.

Використання для тимчасових даних основна мета. Session Storage часто використовується для зберігання тимчасових даних, таких як інформація про

сесію користувача або стан додатку, який потрібно зберігати лише під час активної сесії.

Тож, Session Storage є зручним інструментом для зберігання тимчасових даних в браузері користувача. Він дозволяє зберігати дані, які потрібні лише під час активної сесії, і автоматично очищує їх після закриття вкладки або браузера, що сприяє оптимізації роботи веб-інтерфейсу.

2.2 Вивчення підходів до оптимізації веб-інтерфейсів для різних типів додатків

Для різних типів веб-додатків, таких як односторінкові застосунки (Single Page Applications або SPA) та традиційні веб-сайти, існують різні підходи до оптимізації веб-інтерфейсів. Давайте розглянемо кілька прикладів для кожного типу додатка:

Односторінкові застосунки (SPA) мають такі методи та техніки:

Техніка лінивого завантаження (lazy loading) є важливою стратегією оптимізації швидкодії веб-додатків, особливо для односторінкових застосунків (SPA), де весь контент завантажується одноразово. Вона полягає в тому, що ресурси, такі як зображення, скрипти або стилі, завантажуються лише в той момент, коли вони потрібні на поточній сторінці, а не при завантаженні всієї сторінки разом зі стартовим завантаженням.

Основні переваги лінивого завантаження:

Зменшення часу завантаження сторінки. Одним з основних переваг є те, що завантаження лише необхідних ресурсів скорочує час завантаження сторінки. Це особливо корисно для сторінок з великою кількістю контенту або великими зображеннями [12].

Ефективне використання ресурсів полягає в тому, що ліниве завантаження дозволяє ефективно використовувати ресурси, тому що лише ті ресурси, які

дійсно потрібні на поточній сторінці, завантажуються, що зменшує використання пропускної здатності та обсяг даних.

Покращення продуктивності досягається зменшенням обсягу завантажуваних ресурсів також може покращити продуктивність веб-додатка, особливо на мобільних пристроях або в умовах обмеженої швидкості Інтернету.

Прикладом може бути сторінка інтернет-магазину, де зображення товарів завантажуються лише тоді, коли користувач прокручує до них на сторінці. Це дозволяє зменшити час завантаження початкової сторінки та забезпечити швидке і плавне взаємодію з інтерфейсом.

Використання кешування даних є важливою стратегією для оптимізації швидкодії веб-додатків, особливо для односторінкових застосунків (SPA), які мають справу з великим обсягом даних. Це дозволяє зменшити кількість запитів до сервера та поліпшити реактивність додатка.

Основні переваги використання кешування даних полягають у наступних методиках:

Зменшення навантаження на сервер важливий аспект для кожного сайту. Кешування дозволяє зберігати копії даних на клієнтському пристрої, що зменшує кількість запитів до сервера. Це особливо важливо для SPA, які мають велику кількість даних, таких як зображення, тексти, або інші ресурси.

Поліпшення реактивності додатка працює завдяки кешуванню, деякі дані можуть бути доступні негайно, без необхідності очікування відповіді від сервера. Це допомагає поліпшити відчуття реактивності та швидкодії додатка для користувача.

Забезпечення доступності офлайн також відіграє важливу роль. Використання service workers дозволяє кешувати ресурси та дані, що дозволяє користувачам отримувати доступ до додатка навіть у відсутності Інтернет-з'єднання. Це особливо важливо для мобільних додатків або для користувачів з обмеженим доступом до мережі.

Наприклад, у веб-додатку з соціальною мережею можна кешувати фотографії та повідомлення користувачів, які вони переглядали недавно. Коли

користувач повторно відкриває додаток або переглядає свої сторінки, ці дані можуть бути завантажені з кешу, що поліпшує швидкодію та реактивність додатка.

Для традиційні веб-сайтів користуються попитом такі методи:

Мінімізація та компресія ресурсів. Для традиційних веб-сайтів, де кожна сторінка перезавантажується при кожному запиті, важливо зменшити розмір ресурсів, таких як HTML, CSS та JavaScript. Мінімізація та компресія цих файлів допомагає зменшити час завантаження та покращити продуктивність.

Оптимізація зображень. Традиційні веб-сайти часто містять багато зображень. Важливо оптимізувати зображення, щоб зменшити їх розмір та прискорити час завантаження сторінки. Це може включати використання форматів зображень з високою стисканістю, таких як WebP, або використання технік Lazy Loading для зображень, що не відображаються на екрані безпосередньо.

В обох випадках, важливо також враховувати архітектурні особливості вашого додатка та потреби вашої аудиторії при використанні конкретних методів оптимізації. Оптимізація веб-інтерфейсів за допомогою різних методів, таких як використання асинхронних запитів, кешування даних, ліниве завантаження ресурсів та інші, грає важливу роль у поліпшенні швидкодії, реактивності та загальної ефективності веб-додатків.

Використання цих методів дозволяє зменшити час завантаження сторінок, поліпшити відчуття реактивності для користувачів, знизити навантаження на сервер та забезпечити кращий досвід використання веб-додатків як на комп'ютерах, так і на мобільних пристроях. При правильному використанні цих методів можна досягти покращення продуктивності та забезпечити більш задовільне взаємодію з користувачами.

2.3 Стратегії оптимізації веб-інтерфейсів для мобільних пристроїв та інших платформ

Оптимізація веб-інтерфейсів для мобільних пристроїв та інших платформ включає ряд стратегій, спрямованих на поліпшення швидкодії, відзивчивості та користувацького досвіду. Пропоную розглянути деякі з найефективніших стратегій:

Перша стратегія це адаптивний та респонсивний дизайн. Підходи до адаптивного дизайну дозволяють створювати веб-інтерфейси, які оптимально виглядають та працюють на різних пристроях і розмірах екранів. Наприклад, коли вміст переноситься, змінюється або приховується в залежності від розміру екрану. Прикладом для такої стратегії може бути таким: На десктопі може бути відображено повний розклад дня з тривалим описом кожної події, тоді як на мобільному пристрої цей розклад може бути згорнутим або відображати менше деталей, щоб екран не був перенаповненим [6].

Друга стратегія це мінімізація ресурсів для мобільних пристроїв. Призначення лише необхідних ресурсів для мобільних пристроїв допомагає зменшити час завантаження та споживання ресурсів. Це може включати мінімізацію та компресію зображень, використання злитих або мініфікованих CSS та JavaScript файлів, а також виключення або оптимізацію деяких функцій, які можуть бути зайвими на мобільних пристроях. Для прикладу: зображення, які використовуються на мобільних пристроях, можуть бути меншими за розміром та оптимізованими для маленьких екранів, щоб зменшити обсяг передачі даних та прискорити завантаження сторінок. Крім того, окремі функціональність або скрипти, які можуть бути потужними для десктопів, можуть бути виключені або оптимізовані для мобільних пристроїв, щоб зменшити навантаження на процесор та споживання енергії.

Третя стратегія це Touch-friendly елементи управління допомагають для створення інтерфейсів, дружніх для сенсорних екранів, використовуються

елементи управління, які легко сприймаються пальцями користувача. Наприклад, кнопки повинні мати достатньо великий розмір та відстань між ними, щоб уникнути помилкового натискання. Меню також може бути оптимізоване для сенсорних екранів, забезпечуючи достатньо великі області для взаємодії. Наприклад, розкриваючіся списки можуть бути легко розгорнуті пальцем без необхідності точного натискання.

Ще одна стратегія це тестування на різних пристроях та платформах є однією із головних задач розробки. Проведення тестів на різних пристроях та платформах допомагає виявити можливі проблеми з сумісністю та оптимізувати веб-інтерфейс для різних умов. Наприклад, веб-сайт може відображатися по-різному на різних мобільних пристроях через їхні розміри екрану та підтримку функцій. Тестування дозволяє виявити такі відмінності та виправити їх для забезпечення позитивного досвіду користувача на будь-якому пристрої.

Ці стратегії спільно допомагають покращити ефективність та користувацький досвід веб-інтерфейсів на мобільних пристроях та інших платформах, забезпечуючи оптимальну взаємодію з користувачем та задоволення його потреб.

2.4 Фреймворки JavaScript: роль, мета, принципи роботи

Фреймворки JavaScript відіграють важливу роль у трансформації звичайних веб-сторінок у вражаючі та інтерактивні користувацькі інтерфейси. Вони надають можливість розробляти складні веб-додатки з високою ефективністю та швидкістю, що допомагає скоротити час розробки і підвищити якість коду [27].

Фреймворк - це набір інструментів, функцій і структур, призначених для розробки веб-додатків на JavaScript. Це готова основа, яка допомагає програмістам упорядкувати та спростити процес створення складних програм.

Його мета – надання розробникам зручного інструментарію для створення веб-додатків, які будуть ефективними та легко масштабованими. Вони

забезпечують структуру і організацію коду, що сприяє покращенню читабельності і розуміння програмного забезпечення. Крім того, вони пропонують готові рішення для широкого спектру завдань, які зазвичай виникають у веб-розробці, таких як управління станом додатку, маршрутизація між сторінками, обробка подій та взаємодія з сервером. Це дозволяє розробникам швидше створювати функціональність додатку, скорочуючи час розробки та полегшуючи підтримку програмного забезпечення [28].

Розглянемо деякі з основних функцій фреймворків:

Керування станом. Це набір інструментів для зручного управління станом додатка, які дозволяють розробникам легко відслідковувати і оновлювати дані в додатку, не потребуючи вручну керувати станом. Вони надають зручний механізм для зберігання та маніпулювання даними, які використовуються в додатку, забезпечуючи консистентність та ефективне управління станом за допомогою спеціальних методів та функцій.

Маршрутизація. Це інструменти для керування маршрутизацією всередині додатка, що дозволяють створювати різні сторінки та маршрути, обробляти переходи між ними та оновлювати URL відповідно до поточного стану додатка.

Компонентний підхід. Фреймворки часто використовують підхід, що дозволяє розбити інтерфейс на окремі компоненти, кожен з яких відповідає за свою функціональність. Це спрощує процес розробки, сприяє повторному використанню коду та забезпечує більш чисту та організовану структуру [5].

Взаємодія із сервером. Фреймворки надають інструменти для спілкування з сервером, включаючи Ajax-запити та вбудовані API для роботи з серверними ресурсами. Це дозволяє створювати динамічні додатки, які можуть оновлюватися та взаємодіяти з сервером без перезавантаження сторінки.

Усього, JavaScript фреймворки забезпечують розробникам набір інструментів та абстракцій, які дозволяють зосередитися на створенні функціональних і ефективних веб-застосунків, зменшуючи витрати на повторну розробку і поліпшуючи процес розробки загалом.

Ось огляд п’яти популярних JavaScript фреймворків, їх особливості і області

використання.

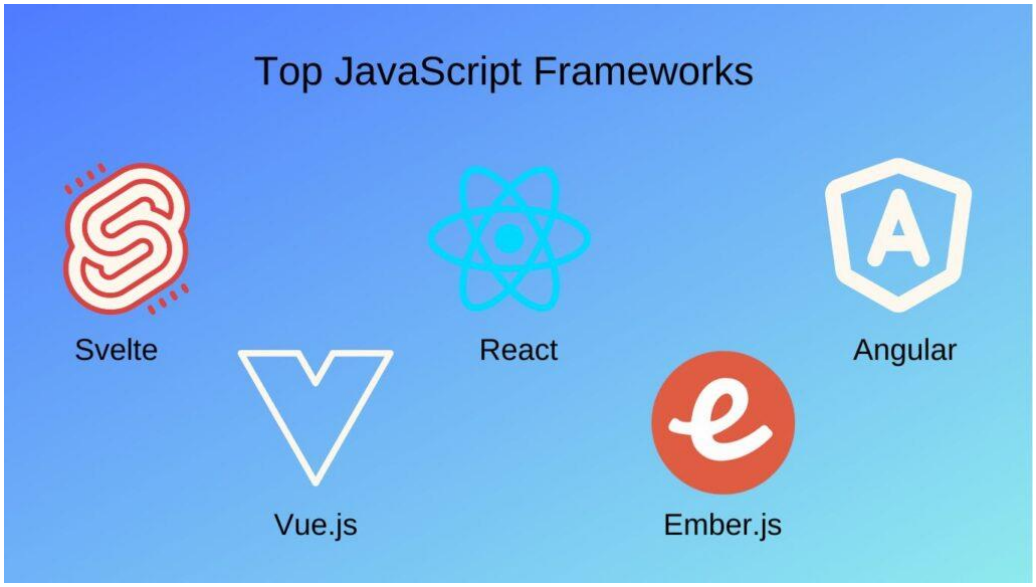


Рисунок 2.1 – Підбірка популярних фреймворків для JavaScript

React - це фреймворк, розроблений компанією Facebook, який базується на компонентах. Він дозволяє створювати перевикористовувані компоненти для вашого інтерфейсу. React відомий своєю віртуальною DOM, яка дозволяє ефективно оновлювати інтерфейс веб-додатків без повного перерисування сторінки. Цей фреймворк є популярним вибором для створення складних та інтерактивних користувацьких інтерфейсів і широко використовується в індустрії веб-розробки [15].

Angular - це фреймворк, розроблений Google, який надає повноцінне рішення для створення веб-додатків. Він має широкий набір функцій, таких як керування станом, маршрутизація, валідація форм і багато інших. Angular використовує власну синтаксичну конструкцію, відому як "директиви", для створення компонентів та управління інтерфейсом. Цей фреймворк ідеально підходить для розроблення складних застосунків, які вимагають масштабованості та розширеного функціоналу.

Vue.js - це фреймворк, який вважається простим у використанні. Він має зрозумілий синтаксис, що робить його привабливим для розробників-початківців. Крім того, він пропонує двосторонню прив'язку даних, управління станом та

багато готових компонентів. Vue.js ідеально підходить для створення інтерфейсів різної складності, від невеликих проектів до більших додатків.

Ember.js - це фреймворк, спрямований на розробку складних та амбітних веб-додатків. Він пропонує розширений набір можливостей, таких як управління станом, маршрутизація, взаємодія з сервером та інші. Основний акцент робиться на структуризації коду та використанні конвенцій, що робить Ember.js привабливим вибором для великих та складних проєктів, де організація та масштабованість є ключовими аспектами.

Svelte - це новаторський фреймворк, який пропонує зовсім новий підхід до створення інтерфейсів. Відмінною рисою Svelte є те, що він не потребує виконання клієнтського коду, а замість цього перетворює його в оптимізований JavaScript під час збірки. Це дозволяє розробникам створювати дуже швидкі та ефективні додатки. Крім того, Svelte має простий і зрозумілий синтаксис, що полегшує розробку інтерфейсів та допомагає зменшити обсяг коду [17].

Вибір JavaScript фреймворку залежить від ваших завдань, рівня навичок і переваг. Ось кілька порад, які допоможуть вам вибрати відповідний фреймворк:

Перш за все, потрібно визначити потреби та завдання: деякі фреймворки краще підходять для створення інтерактивних інтерфейсів, тоді як інші краще підходять для серверної розробки. Під час вибору варто звернiти увагу на документацію та активність спільноти: хороша документація та активна спільнота допоможуть швидко освоїти фреймворк та знайти відповіді на свої питання. Також варто врахувати рівень навичок: деякі фреймворки мають простий синтаксис і підходять для початківців, тоді як інші вимагають більш глибокого розуміння JavaScript.Дослідивши екосистему фреймворка: перевіряємо наявність розширень, плагінів та інтеграцію з іншими інструментами.

Тому, відповідь на питання, який фреймворк вибрати js, залежить від ваших завдань, рівня навичок і переваг. До того ж, практичне випробування і тестування фреймворку також можуть допомогти вам прийняти більш усвідомлене рішення.

Фреймворки JavaScript широко використовуються в реальних проектах, підтвердженням цього є безліч прикладів. Вони допомагають розробникам

створювати складні веб-додатки більш ефективно і з меншими витратами. Цим, зокрема, пояснюється популярність фреймворків js. Наведемо приклади їхнього практичного застосування:

Фреймворк React активно використовується компанією Facebook для розробки різноманітних продуктів, включаючи платформу Facebook, Instagram і WhatsApp. Завдяки React їм вдається створювати ефективні та швидкі інтерфейси, які легко підтримувати та оновлювати [16].

Google активно використовує фреймворк Angular у своїх сервісах, таких як Gmail, Google Analytics і Google Drive. Angular забезпечує потужний набір інструментів для розробки складних веб-додатків і дозволяє створювати масштабовані та високопродуктивні інтерфейси.

Netflix активно використовує фреймворк Vue.js для розробки свого користувацького інтерфейсу. Цей фреймворк славиться своїм простим та інтуїтивним синтаксисом, що дозволяє розробникам швидко створювати інтерактивні компоненти та легко інтегрувати їх у платформу Netflix.

LinkedIn, велика соціальна мережа для професіоналів, використовує фреймворк Ember.js для розробки свого веб-інтерфейсу. Цей фреймворк забезпечує потужні інструменти для управління станом та організації коду, що дозволяє створювати складні та масштабовані додатки.

Slack, відомий комунікаційний інструмент для бізнесу, використовує серверний фреймворк Node.js для обробки запитів та взаємодії з базою даних. Node.js відзначається високою продуктивністю та масштабованістю, що робить його ідеальним вибором для опрацювання великих обсягів даних у режимі реального часу.

У розробці сучасних веб-додатків, JavaScript фреймворки відіграють ключову роль, забезпечуючи розробникам необхідні інструменти та структури для створення складних та інноваційних програмних продуктів. Вони значно спрощують процес розробки, дозволяючи фахівцям концентруватися на логіці застосунку, взаємодії з користувачем та вирішенні бізнес-завдань. Благодаря фреймворкам, розробники можуть швидко та ефективно створювати інтерфейси,

які привертають увагу користувачів своїми можливостями та продуктивністю.

В цілому, JavaScript фреймворки відкривають безліч можливостей і значно полегшують життя розробників, зробивши їхню роботу більш продуктивною та приємною [29].

2.5 Середовище розробки Visual Studio Code

Visual Studio Code – редактор коду для кросплатформенної розробки веб і хмарних додатків. Він підтримує ряд мов програмування, підсвічування синтаксису, IntelliSense, рефакторинг, налагодження, навігацію по коду, підтримку Git та інші можливості. Багато можливостей Visual Studio Code недоступні через графічний інтерфейс, найчастіше вони використовуються через палітру команд, яка викликається поєднанням клавіш, або JSON файли (наприклад, налаштування користувача). Visual Studio Code має підтримку плагінів, доступних через Visual Studio Marketplace. Вони можуть включати в себе доповнення до редактора, підтримку додаткових мов програмування, статичні аналізатори коду [22].

Visual Studio Code — це редактор вихідного коду, розроблений компанією Microsoft для Windows, Linux, MacOs. Це легкий та швидкий редактор для кроссплатформеної розробки. Він містить необхідний початковий набір інструментів для роботи з Git, наприклад, термінал, який дає змогу швидко і зручно взаємодіяти з git, а також підсвітку синтаксису обраної мови, відладку, рефакторинг. Для веб проектів тут є одразу вбудована відладка коду, а для інших – можливість встановити розширення (плагіни). Найчастіше використовуванні плагіни в веб-розробників це, наприклад, Live Server, який дозволяє розробнику одразу тестувати свій код без перезавантаження сторінки, ESLint. Також є плагіни для підтримки інших мов програмування чи фреймовркв. Загалом, майже всі плагіни були розробленні сторонніми розробниками та доступні через Visual Code

Marketplace [13].

VS Code підтримує широкий спектр мов програмування від Java, C++ та Python до CSS, Go та Dockerfile. Більше того, VS Code дозволяє додавати і навіть створювати нові розширення, включаючи лінтери коду, налагоджувачі та підтримку хмарних та веб-розробок. Інтерфейс користувача VS Code забезпечує більшу взаємодію в порівнянні з іншими текстовими редакторами. Для спрощення взаємодії з користувачем VS Code поділено на п'ять основних областей:

– Панель активності;
– Бічна панель;
– Групи редакторів;
– Панель;
– Рядок стану.

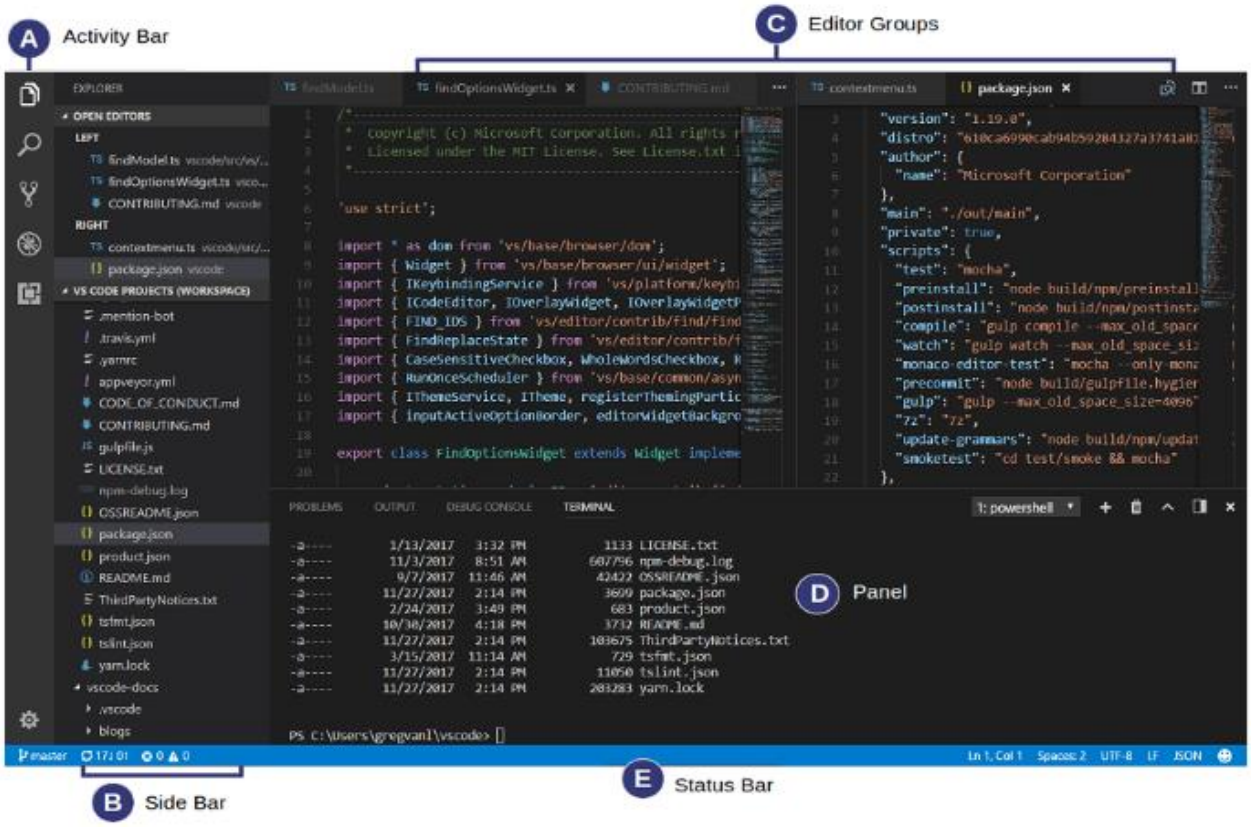


Рисунок 2.2 – Відображення робочих областей

Також даний редактор дозволяє кардинально змінювати свій зовнішній

вигляд, встановлювати теми, які можна завантажити чи обрати зі стандартних. Цей редактор являється безкоштовним, розробляється як програмне забезпечення з відкритим вихідним кодом [21].

Програма Visual Studio Code має доступ до командного рядку CMD. Цей функціонал можна знайти у вкладці “Terminal”. Користувачу не потрібно переключатись між командним рядком CMD та програмою Visual Studio Code — запустити додаток можна безпосередньо з редактору коду. Окрім доступу до 16 командного рядка CMD, вкладка “Terminal” також дає можливість роботи з консоллю PowerShell та Linux Bash. За замовчуванням, програма Visual Studio Code працює в режимі явного збереження. Для того щоб його виконати, потрібно, натиснути комбінацію клавіш “Ctrl + S”.Проте, можна також і увімкнути режим автоматичного збереження Auto Save. Для цього, потбріно натиснути клавіші “Ctrl + Shift + P”, у вікні. що відкрилося, потрібно набрати команду “ auto”. Програма Visual Studio Code працює з файлами і папками, у яких знаходяться прокети. Для того щоб відкрити, наприклад, файл для редагування потрібно або запустити команду ./code index.html, або просто обрати потрібний файл у переліку “Open Editor”. Програма Visual Studio Code сама визначає тип проекту в залежності від вмісту папки. Наприклад, якщо в папці знаходяться файли package.json, project.json, tsconfig.json, то програма Visual Studio Code включає багато нових функцій, які забезпечують IntelliSence, підказки, навігацію по коду, і багато іншого. Для того щоб вивести підкази IntelliSence, потрібно просто натиснути на “Ctrl + Space”. При наборі коду редактор буде показувати його автоматично. Програма Visual Studio Code підтримує абревіатури Emmet. Програміст може використовувати їх при редагуванні файлів HTML, Razor, CSS, Less, Sass, XML або Jade.

РОЗДІЛ 3. РОЗРОБКА ДОДАТКУ CRUD З ВИКОРИСТАННЯМ JAVASCRIPT ТА LOCAL STORAGE

3.1 Огляд та визначення CRUD додатку

У сучасному світі, коли обмін та обробка інформації стають все більш важливими, виникає необхідність у швидкому та ефективному керуванні даними. CRUD додатки відіграють важливу роль у вирішенні цієї проблеми, надаючи зручні та потужні засоби для створення, читання, оновлення та видалення інформації.

CRUD є абревіатурою, що означає чотири основні операції, які можна виконувати з даними: Create (створення), Read (читання), Update (оновлення) та Delete (видалення). Ці операції становлять основу для багатьох веб-додатків та систем управління інформацією, дозволяючи користувачам ефективно взаємодіяти з даними та змінювати їх за потребою [23].

CRUD додаток - це веб-додаток або програма, яка забезпечує можливість виконання всіх чотирьох операцій CRUD. Він дозволяє користувачам створювати нові записи, переглядати існуючі дані, оновлювати їх та видаляти, управляючи таким чином інформацією у системі.

У цьому дослідженні ми детально розглянемо особливості та переваги використання CRUD додатків, їх застосування у різних сферах діяльності та принципи їх роботи.

Операції CRUD є фундаментом для багатьох програм та систем управління інформацією, дозволяючи користувачам взаємодіяти з даними та змінювати їх відповідно до власних потреб.

- Create (створювати);

Функція Create в CRUD додатку відповідає за створення нових записів або

об'єктів в базі даних або іншій системі управління даними. Ця операція передбачає додавання нової інформації до системи. При використанні цієї функції користувач може ввести дані через відповідну форму або інтерфейс, після чого ці дані будуть збережені у відповідному сховищі [24].

– Read (читати);

Функція Read в CRUD додатку відповідає за отримання і відображення інформації, яка вже існує в базі даних або іншій системі управління даними. Ця операція передбачає читання даних з вже існуючих записів або об'єктів та їх відображення користувачеві. При використанні цієї функції користувач може переглядати інформацію про існуючі дані у вигляді списку, таблиці або іншого візуального способу.

– Update (оновлювати);

Функція update в CRUD додатку відповідає за оновлення існуючих даних в базі даних або іншій системі управління даними. Ця операція дозволяє користувачам внести зміни до існуючих записів або об'єктів, змінюючи їхні властивості або значення полів.

– Delete (видаляти)

Функція delete в CRUD додатку відповідає за видалення існуючих даних з бази даних або іншої системи управління даними. Ця операція дозволяє користувачам видаляти записи або об'єкти, які вже не потрібні або застаріли.

Операції CRUD є фундаментом для багатьох програм та систем управління інформацією, дозволяючи користувачам взаємодіяти з даними та змінювати їх відповідно до власних потреб.

Переваги використання операцій CRUD:

Ефективне управління даними. Операції CRUD дозволяють організаціям швидко та ефективно керувати своїми даними, забезпечуючи їхню актуальність та впорядкованість [8].

Підвищення продуктивності. Використання операцій CRUD допомагає збільшити продуктивність, оскільки елімінує ручне введення даних та ризик людських помилок.

Масштабованість рішень. Універсальність операцій CRUD дозволяє легко масштабувати рішення відповідно до потреб зростаючого бізнесу.

Зручний інтерфейс.Організації користуються простим та зрозумілим інтерфейсом операцій CRUD, що спрощує управління даними та забезпечує доступність інформації для всіх членів команди.

Забезпечення безпеки даних. Операції CRUD забезпечують безпеку конфіденційних даних, обмежуючи доступ до них лише авторизованим користувачам.

Спільна робота. Можливість одночасного доступу та зміни даних багатьма користувачами поліпшує командну роботу та прискорює виконання проєктів.

Оптимізація робочого процесу. Операції CRUD спрощують управління даними та автоматизують їх, що зменшує витрати часу та коштів.

Резервне копіювання та відновлення даних. Використання операцій CRUD полегшує резервне копіювання та відновлення даних, забезпечуючи стабільність та надійність інформації.

3.2 Реалізація операцій CRUD з та використання LocalStorage

Перейдемо до створення CRUD додатку. Реалізуємо список працівників з їх особистими даними та фотокартками.

Розглянемо цей HTML-код, який представляє собою основну розмітку для створення веб-сторінки.

```
<!doctype html>
<html lang="en">

<head>

    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">


    <link
```

```
href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC"
crossorigin="anonymous">


    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-
icons@1.10.5/font/bootstrap-icons.css">

    <title>CRUD Operations</title>

    <link rel="stylesheet" href="style.css">
</head>
```

В цьому розділі ми включаємо метатеги meta, які визначають кодування та масштабування сторінки.

Ми також додаємо посилання на зовнішні ресурси, такі як стилі Bootstrap та значки Bootstrap, які будуть використовуватися на сторінці. Вказуємо заголовок сторінки, який відображається в рядку заголовку браузера, а також підключаємо власний файл стилів, який буде містити додаткові стилі для нашого сайту.

```
<section class="p-3">

        <div class="row">
            <div class="col-12">
                <button class="btn btn-primary newUser" data-bs-toggle="modal" data-
bs-target="#userForm">New User <i class="bi bi-people"></i></button>
            </div>
        </div>

        <div class="row">
            <div class="col-12">
                <table class="table table-striped table-hover mt-3 text-center table-
bordered">

                    <thead>
                        <tr>
                            <th>S.No</th>
                            <th>Picture</th>
                            <th>Name</th>
                            <th>Age</th>
                            <th>City</th>
                            <th>Email</th>
                            <th>Phone</th>
                            <th>Post</th>
```

```
                        <th>Start Date</th>
                        <th>Action</th>
                    </tr>
                </thead>

                <tbody id="data"></tbody>

            </table>
        </div>
    </div>

</section>
```

Представлений код містить кнопку для створення нового користувача та таблицю для відображення інформації про користувачів

Кожен рядок таблиці буде відображати інформацію про одного користувача, а стовпці відповідають різним атрибутам користувача, таким як Ім'я, Вік, Місто тощо. Останній стовпець містить кнопки дій для кожного користувача, такі як перегляд, редагування та видалення.

```
<div class="modal fade" id="userForm">
    <div class="modal-dialog modal-dialog-centered modal-lg">
        <div class="modal-content">

            <div class="modal-header">
                <h4 class="modal-title">Fill the Form</h4>
                <button type="button" class="btn-close" data-bs-dismiss="modal"
aria-label="Close"></button>
            </div>

            <div class="modal-body">

                <form action="#" id="myForm">

                    <div class="card imgholder">
                        <label for="imgInput" class="upload">
                            <input type="file" name="" id="imgInput">
                            <i class="bi bi-plus-circle-dotted"></i>
                        </label>
                        <img src="./image/Profile Icon.webp" alt="" width="200"
height="200" class="img">
                    </div>

                    <div class="inputField">
```

```
                        <div>
                            <label for="name">Name:</label>
                            <input type="text" name="" id="name" required>
                        </div>
                        <div>
                            <label for="age">Age:</label>
                            <input type="number" name="" id="age" required>
                        </div>
                        <div>
                            <label for="city">City:</label>
                            <input type="text" name="" id="city" required>
                        </div>
                        <div>
                            <label for="email">E-mail:</label>
                            <input type="email" name="" id="email" required>
                        </div>
                        <div>
                            <label for="phone">Number:</label>
                            <input type="text" name="" id="phone" minlength="11"
maxlength="11" required>
                        </div>
                        <div>
                            <label for="post">Post:</label>
                            <input type="text" name="" id="post" required>
                        </div>
                        <div>
                            <label for="sDate">Start Date:</label>
                            <input type="date" name="" id="sDate" required>
                        </div>
                    </div>

                </form>
            </div>

            <div class="modal-footer">
                <button type="button" class="btn btn-secondary" data-bs-
dismiss="modal">Close</button>
                <button type="submit" form="myForm" class="btn btn-primary
submit">Submit</button>
            </div>
        </div>
    </div>
</div>
```

Цей HTML-код представляє модальне вікно, яке використовується для введення даних нового користувача. Тут ми можемо побачити модальне вікно , його вміст, заголовок та тіло. Також цей код реалізовує HTML-форму, яка містить поля для введення даних нового користувача. Кожне поле має свій власний мітку

(label) та введене поле (input), а також атрибути для валідації (наприклад, required). Додатково, є можливість вибору зображення профілю за допомогою поля вводу файлу.

Підвал модального вікна. Це розділ, який містить кнопки для закриття модального вікна та підтвердження введених даних. У цьому випадку є кнопки "Close" для закриття вікна та "Submit" для відправлення форми з даними користувача.

```
 <div class="modal fade" id="readData">
        <div class="modal-dialog modal-dialog-centered modal-lg">
            <div class="modal-content">

                <div class="modal-header">
                    <h4 class="modal-title">Profile</h4>
                    <button type="button" class="btn-close" data-bs-dismiss="modal"
aria-label="Close"></button>
                </div>

                <div class="modal-body">

                    <form action="#" id="myForm">

                        <div class="card imgholder">
                            <img src="./image/Profile Icon.webp" alt="" width="200"
height="200" class="showImg">
                        </div>

                        <div class="inputField">
                            <div>
                                <label for="name">Name:</label>
                                <input type="text" name="" id="showName" disabled>
                            </div>
                            <div>
                                <label for="age">Age:</label>
                                <input type="number" name="" id="showAge" disabled>
                            </div>
                            <div>
                                <label for="city">City:</label>
                                <input type="text" name="" id="showCity" disabled>
                            </div>
                            <div>
                                <label for="email">E-mail:</label>
                                <input type="email" name="" id="showEmail" disabled>
                            </div>
                            <div>
```

```
                                <label for="phone">Number:</label>
                                <input type="text" name="" id="showPhone"
minlength="11" maxlength="11" disabled>
                            </div>
                            <div>
                                <label for="post">Post:</label>
                                <input type="text" name="" id="showPost" disabled>
                            </div>
                            <div>
                                <label for="sDate">Start Date:</label>
                                <input type="date" name="" id="showsDate" disabled>
                            </div>
                        </div>

                    </form>
                </div>
            </div>
        </div>
    </div>
```

Тут ми можемо бачити схожість з попереднім модальним вікном. Головна відмінність в тому, що попередній розділ призначений для відображення даних користувача в таблиці, де можна додавати, редагувати та видаляти записи , а цей код дозволяє нам переглядати дані вже створених користувачів, без можливості їх змінення безпосередньо в модальному вікні.

```
 <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js"
integrity="sha384-MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM"
crossorigin="anonymous"></script>

    <script src="app.js"></script>
```

Останні елементи HTML документу є підключення зовнішніх файлів JavaScript.

Перший тег вказує на посилання, яке підключає функціонал розробки інтерфейсів веб-сторінок. Цей скрипт є частиною бібліотеки Bootstrap версії 5.0.2 [10]. Другий тег підключає локальний JavaScript-файл. Власне в цьому файлі містяться скрипти, які будуть виконуватися у веб-сторінці, такі як взаємодія з

елементами сторінки, обробка подій.

Наступним нашим кроком буде додавання стилізації для нашого веб-додатку.

```
table tr td{
    vertical-align: middle;
}

td button{
    margin: 5px;
}

td button i{
    font-size: 20px;
}


.modal-header{
    background: #0d6efd;
    color: #fff;
}

.modal-body form {
    display: flex;
    justify-content: space-between;
    align-items: flex-start;
    padding: 0;
}

.modal-body form .imgholder{
    width: 200px;
    height: 200px;
    position: relative;
    border-radius: 20px;
    overflow: hidden;
}

.imgholder .upload{
    position: absolute;
    bottom: 0;
    left: 10;
    width: 100%;
    height: 100px;
    background: rgba(0,0,0,0.3);
    display: none;
    justify-content: center;
    align-items: center;
    cursor: pointer;
```

```
}

.upload i{
    color: #fff;
    font-size: 35px;
}

.imgholder:hover .upload{
    display: flex;
}

.imgholder .upload input{
    display: none;
}

.modal-body form .inputField{
    flex-basis: 68%;
    border-left: 5px groove blue;
    padding-left: 20px;
}

form .inputField > div{
    width: 100%;
    display: flex;
    justify-content: space-between;
    align-items: center;
    margin-bottom: 20px;
}

form .inputField > div label{
    font-size: 20px;
    font-weight: 500;
}

#userForm form .inputField > div label::after{
    content: "*";
    color: red;
}

form .inputField > div input{
    width: 75%;
    padding: 10px;
    border: none;
    outline: none;
    background: transparent;
    border-bottom: 2px solid blue;
}

.modal-footer .submit{
    font-size: 18px;
}
```

```
#readData form .inputField > div input{
    color: #000;
    font-size: 18px;
}
```

Цей CSS-код містить стилі для оформлення елементів на веб-сторінці . Тут ми можемо побачити як реалізоване встановлення вертикального вирівнювання тексту в середині комірок таблиці, обмеження відступів для кнопок в середині комірок таблиці, збільшення розміру іконок всередині кнопок, змінення кольору кольору фону та тексту для заголовка модального вікна. А також стилі для кнопок завантаження зображення та стилі для блоку вводу даних в модальному вікні. Вони дозволяють налаштувати вигляд і поведінку різних елементів на веб-сторінці.

Переходимо до головного етапу створення додатку, реалізуємо функціонал за допомогою JavaSript та Local Storage.

```
var form = document.getElementById("myForm"),
    imgInput = document.querySelector(".img"),
    file = document.getElementById("imgInput"),
    userName = document.getElementById("name"),
    age = document.getElementById("age"),
    city = document.getElementById("city"),
    email = document.getElementById("email"),
    phone = document.getElementById("phone"),
    post = document.getElementById("post"),
    sDate = document.getElementById("sDate"),
    submitBtn = document.querySelector(".submit"),
    userInfo = document.getElementById("data"),
    modal = document.getElementById("userForm"),
    modalTitle = document.querySelector("#userForm .modal-title"),
    newUserBtn = document.querySelector(".newUser");
```

Цей JavaScript код виконує наступні дії:

- визначає змінні для доступу до елементів HTML за їх ідентифікаторами або класами. Наприклад:
- забезпечує доступ до цих елементів для подальшої обробки подій, зчитування введених даних користувачем та взаємодії з ними в JavaScript

коді. Наприклад, він може слугувати для валідації введених даних перед їх відправленням на сервер або для відображення введених даних в модальному вікні.

– забезпечує взаємодію між HTML та JavaScript, дозволяючи динамічно маніпулювати вмістом сторінки та взаємодіяти з користувачем через події, такі як кліки на кнопках або відправка форми.

```
let getData = localStorage.getItem('userProfile') ?
JSON.parse(localStorage.getItem('userProfile')) : [];
```

Тут ми можемо побачити застосування Local Storage. Використовується метод `localStorage.getItem('userProfile')` для отримання даних з локального сховища браузера за ключем 'userProfile'. Також перевіряє, чи отримані дані є непорожніми (тобто чи не дорівнюють `null` або `undefined`) та чи можна їх перетворити у об'єкт JavaScript. Якщо дані отримано успішно та вони не є непорожніми, вони розбираються з рядка JSON у відповідний об'єкт JavaScript за допомогою `JSON.parse(localStorage.getItem('userProfile'))`. Натомість якщо дані відсутні або неправильно сформовані, змінна `getData` буде ініціалізована пустим масивом `[]`.

```
let isEdit = false,
    editId;
showInfo();

newUserBtn.addEventListener('click', () => {
    submitBtn.innerText = 'Submit',
        modalTitle.innerText = "Fill the Form";
    isEdit = false;
    imgInput.src = "./image/Profile Icon.webp";
    form.reset();
});


file.onchange = function() {
    if (file.files[0].size < 1000000) { // 1MB = 1000000
        var fileReader = new FileReader();
```

```
        fileReader.onload = function(e) {
            imgUrl = e.target.result;
            imgInput.src = imgUrl;
        };

        fileReader.readAsDataURL(file.files[0]);
    } else {
        alert("This file is too large!");
    }
};


function showInfo() {
    document.querySelectorAll('.employeeDetails').forEach(info => info.remove());
    getData.forEach((element, index) => {
        let createElement = `<tr class="employeeDetails">
            <td>${index+1}</td>
            <td><img src="${element.picture}" alt="" width="50" height="50"></td>
            <td>${element.employeeName}</td>
            <td>${element.employeeAge}</td>
            <td>${element.employeeCity}</td>
            <td>${element.employeeEmail}</td>
            <td>${element.employeePhone}</td>
            <td>${element.employeePost}</td>
            <td>${element.startDate}</td>


            <td>
                <button class="btn btn-success"
onclick="readInfo('${element.picture}', '${element.employeeName}',
'${element.employeeAge}', '${element.employeeCity}', '${element.employeeEmail}',
'${element.employeePhone}', '${element.employeePost}', '${element.startDate}')" data-
bs-toggle="modal" data-bs-target="#readData"><i class="bi bi-eye"></i></button>

                <button class="btn btn-primary" onclick="editInfo(${index},
'${element.picture}', '${element.employeeName}', '${element.employeeAge}',
'${element.employeeCity}', '${element.employeeEmail}', '${element.employeePhone}',
'${element.employeePost}', '${element.startDate}')" data-bs-toggle="modal" data-bs-
target="#userForm"><i class="bi bi-pencil-square"></i></button>

                <button class="btn btn-danger" onclick="deleteInfo(${index})"><i
class="bi bi-trash"></i></button>

            </td>
        </tr>`;

        userInfo.innerHTML += createElement;
    });
}
showInfo();
```

Цей фрагмент JavaScript виконує наступні дії:

- ініціалізує змінні `isEdit` і `editId` зі значеннями `false` та `undefined` відповідно;
- встановлює обробник події для кнопки `newUserBtn`, який викликається при кліці на кнопку "New User". При кліці встановлюється значення `isEdit` на `false`, змінюється текст кнопки `submitBtn`, очищається `imgInput.src`, та скидається форма за допомогою `form.reset()`;
- встановлює обробник події для події `change` на файловому введенні `file`. При зміні файлу перевіряється розмір завантаженого файлу. Якщо розмір менше 1 МБ, виконується читання файлу за допомогою `FileReader`, та встановлюється `src` зображення `imgInput`. В іншому випадку виводиться повідомлення про те, що файл занадто великий;
- визначає функцію `showInfo()`, яка відображає інформацію про користувачів. Вона спочатку видаляє попередні дані про користувачів, які можуть вже бути відображені, а потім для кожного об'єкта `element` в `getData` генерує рядок HTML з інформацією про користувача. Кожен рядок містить кнопки для перегляду, редагування та видалення даних користувача. Кнопки для редагування та видалення викликають відповідні функції `editInfo()` та `deleteInfo()`, передаючи їм індекс об'єкта в масиві `getData`;
- функція `showInfo()` викликається для відображення інформації про користувачів після завантаження сторінки;

```
function readInfo(pic, name, age, city, email, phone, post, sDate) {
    document.querySelector('.showImg').src = pic,
        document.querySelector('#showName').value = name,
        document.querySelector("#showAge").value = age,
        document.querySelector("#showCity").value = city,
        document.querySelector("#showEmail").value = email,
        document.querySelector("#showPhone").value = phone,
        document.querySelector("#showPost").value = post,
        document.querySelector("#showsDate").value = sDate;
}
```

```
function editInfo(index, pic, name, Age, City, Email, Phone, Post, Sdate) {
    isEdit = true;
    editId = index,
        imgInput.src = pic,
        userName.value = name,
        age.value = Age,
        city.value = City,
        email.value = Email,
        phone.value = Phone,
        post.value = Post,
        sDate.value = Sdate;

    submitBtn.innerText = "Update";
    modalTitle.innerText = "Update The Form";
};
```

Розглянемо першу функцію readInfo(pic, name, age, city, email, phone, post, sDate)`. Вона призначена для відображення інформації про користувача в модальному вікні для читання. Вона приймає параметри, що представляють інформацію про користувача (зображення, ім'я, вік, місто, електронну пошту, номер телефону, посаду та дату початку роботи), та встановлює відповідні значення для елементів вікна читання модального вікна.

Друга функція editInfo(index, pic, name, Age, City, Email, Phone, Post, Sdate)` використовується для редагування інформації про користувача. Вона приймає параметри, які представляють інформацію про користувача (зображення, ім'я, вік, місто, електронну пошту, номер телефону, посаду та дату початку роботи), а також індекс користувача в масиві `getData`. Функція встановлює ці значення для відповідних елементів форми редагування в модальному вікні. Крім того, вона встановлює значення змінних `isEdit` (встановлюється в `true`) і `editId` (встановлюється в індекс користувача), а також змінює текст кнопки `submitBtn` на "Update" і текст заголовка модального вікна на "Update The Form".

```
function deleteInfo(index) {
    if (confirm("Are you sure want to delete?")) {
        getData.splice(index, 1);
        localStorage.setItem("userProfile", JSON.stringify(getData));
```

```
        showInfo();
    }
}
```

Ця функція `deleteInfo(index)` виконує видалення інформації про користувача з масиву `getData` за вказаним індексом. Вона отримує параметр `index`, що представляє індекс користувача в масиві `getData`, якого потрібно видалити.

Спочатку показується підтверджувальне діалогове вікно за допомогою `confirm`, щоб переконатися, що користувач справді хоче видалити інформацію. Якщо користувач натискає "OK" у діалоговому вікні підтвердження, виконується видалення елемента з масиву `getData` за вказаним індексом за допомогою методу `splice()`. Потім оновлена інформація про користувачі зберігається в локальному сховищі за допомогою `localStorage.setItem("userProfile", JSON.stringify(getData))`. В кінці викликається функція `showInfo()`, яка переробляє вміст таблиці для відображення оновленої інформації про користувачів.

```
form.addEventListener('submit', (e) => {
    e.preventDefault();

    const information = {
        picture: imgInput.src == undefined ? "./image/Profile Icon.webp" :
imgInput.src,
        employeeName: userName.value,
        employeeAge: age.value,
        employeeCity: city.value,
        employeeEmail: email.value,
        employeePhone: phone.value,
        employeePost: post.value,
        startDate: sDate.value
    };

    if (!isEdit) {
        getData.push(information);
    } else {
        isEdit = false;
        getData[editId] = information;
    }

    localStorage.setItem('userProfile', JSON.stringify(getData));

    submitBtn.innerText = "Submit";
    modalTitle.innerHTML = "Fill The Form";
```

```
    showInfo();
    form.reset();
    imgInput.src = "./image/Profile Icon.webp";
});
```

Ця функція викликається при поданні форми за допомогою події `submit`. Вона перешкоджає стандартній дії браузера за допомогою `e.preventDefault()`, щоб уникнути перезавантаження сторінки. Пропоную розглянути основні кроки цієї функції:

- створюється об'єкт `information`, який містить всі дані форми, що були введені користувачем;
- якщо режим редагування не активований (`!isEdit`), тобто додається новий користувач, то об'єкт `information` додається до масиву `getData` за допомогою методу `push()`;
- якщо режим редагування активований, тобто оновлюється існуючий користувач, то об'єкт `information` замінює відповідний елемент у масиві `getData` за допомогою індексу `editId`;
- оновлена інформація про користувачі зберігається в локальному сховищі за допомогою `localStorage.setItem('userProfile', JSON.stringify(getData))`;
- текст кнопки "Submit" і заголовка модального вікна змінюються на початкові значення;
- викликається функція `showInfo()`, яка переробляє вміст таблиці для відображення оновленої інформації про користувачів;
- форма очищається за допомогою `form.reset()`;
- зображення профілю повертається до значення за замовчуванням.

3.4 Тестування та підведення підсумків

Розпочавши тестування ми можемо побачити таблицю, де при додаванні працівників буде зображена інформація про них. Також тут присутня кнопка для додавання нового користувача.

Рисунок 3.1 Не заповнена таблиця нашого додатку

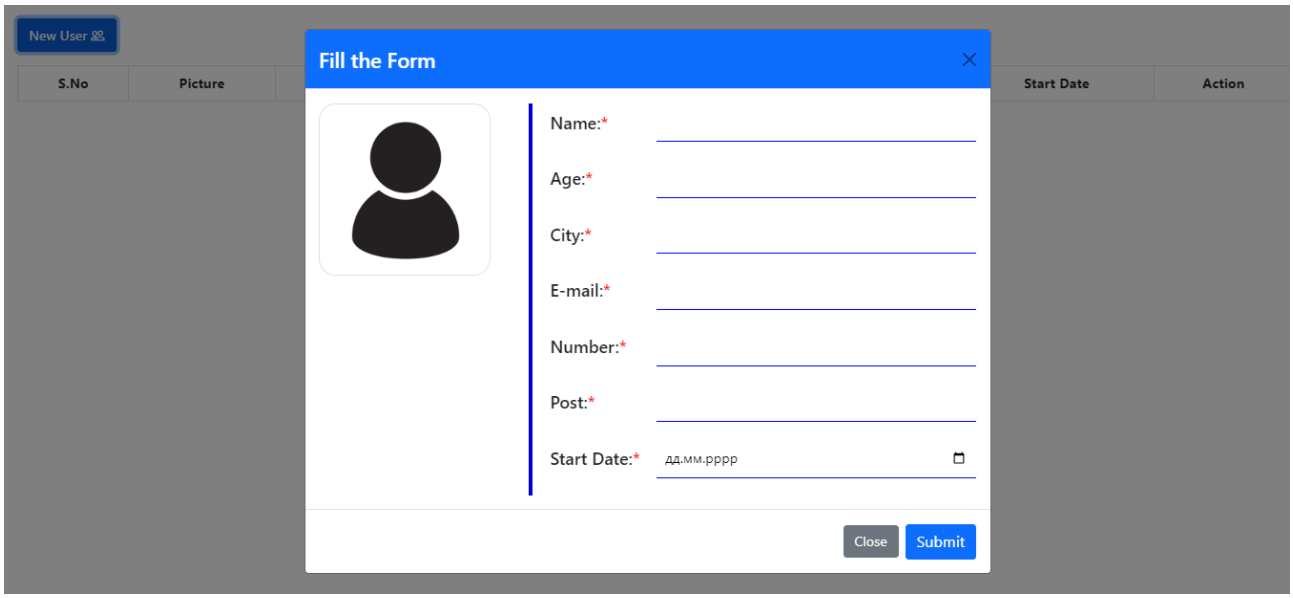


Рисунок 3.2 Форма для заповнення даних

Тут ми можемо побачити як виглядає модальне вікно з полями для введення даних та кнопки для скасування та підтвердження дії. Також при натисканні на картинку з зображенням працівника, ми зможемо завантажити власне фото.

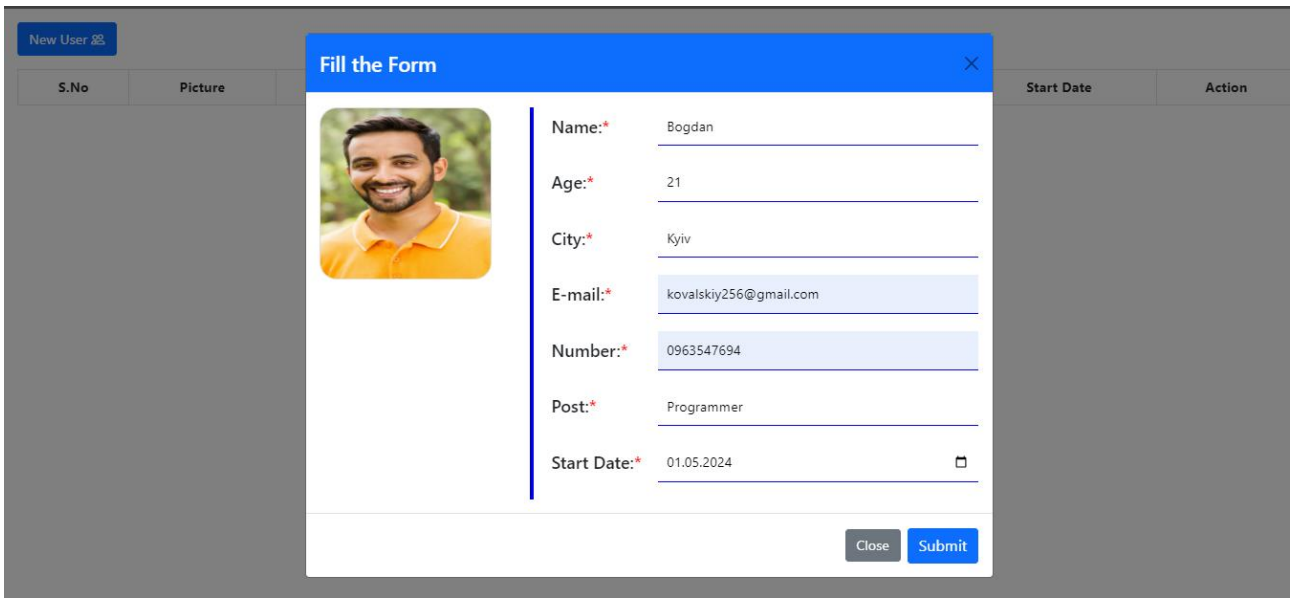


Рисунок 3.3 Заповнена форма працівника

Після заповнення даних та завантаження фото натискаємо кнопку підтвердження та перевіряємо результат.

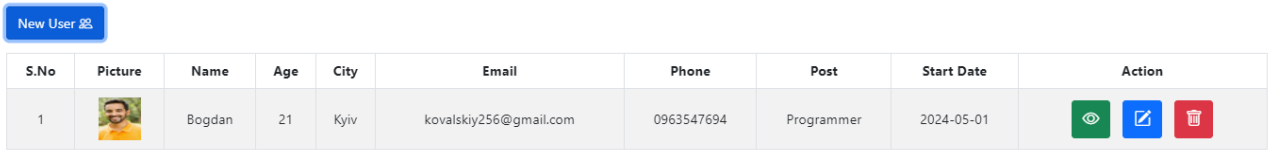


Рисунок 3.4 наявність працівника в таблиці

Як ми бачимо наш працівник додався в таблицю. Тепер можна перейти до перевірки інших функцій.

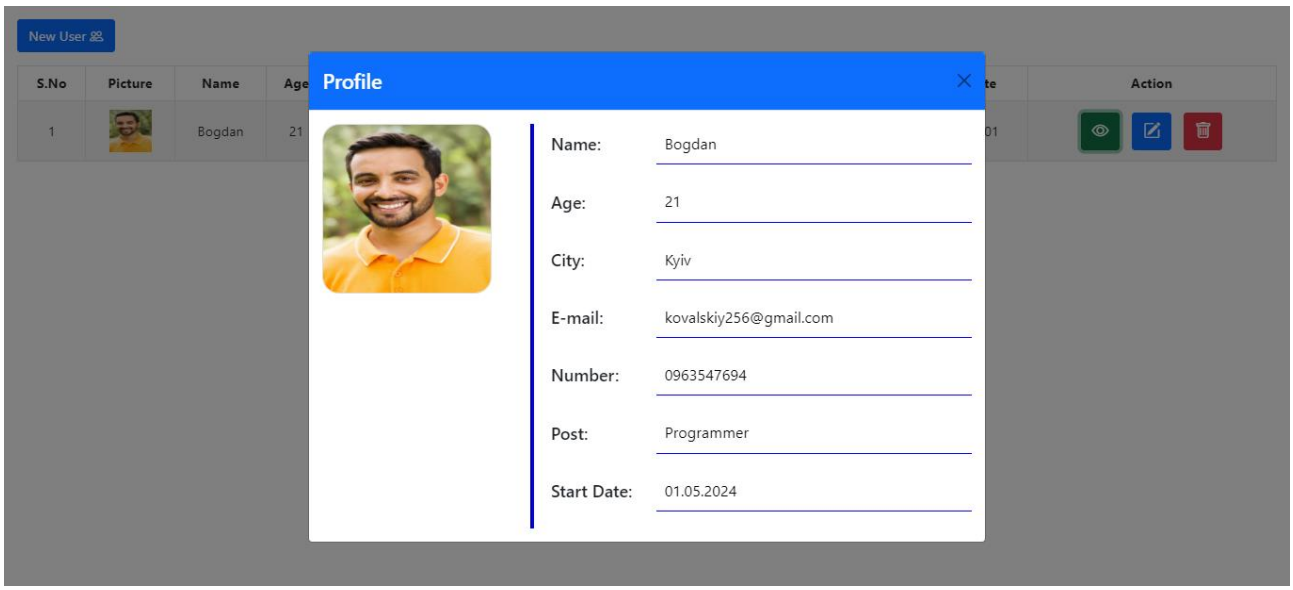


Рисунок 3.5 Огляд профілю працівника

При натисканні на кнопку огляду профілю працівника, відкривається модальне вікно де ми можемо побачити усю інформацію про нього та фотографію.

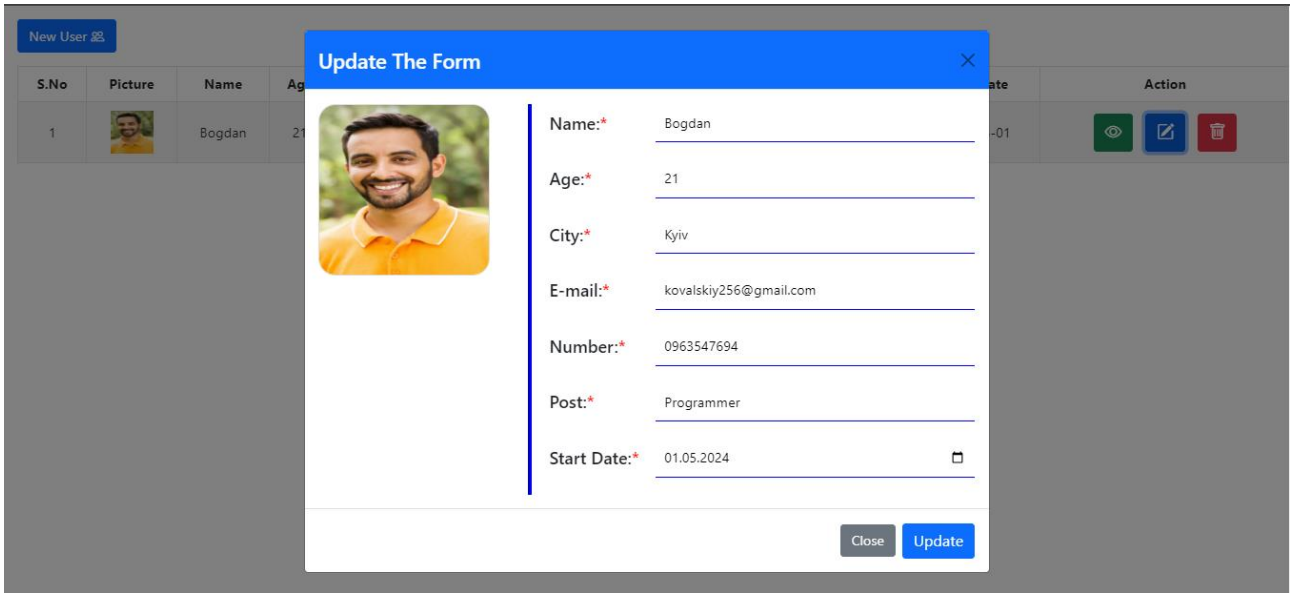


Рисунок 3.6 Огляд профілю працівника для змінення даних

Щоб відредагувати профіль, необхідно натиснути кнопку редагування. Після внесення усіх змін натискаємо кнопку оновити після чого наш додаток збереже змінити в Local Storage.

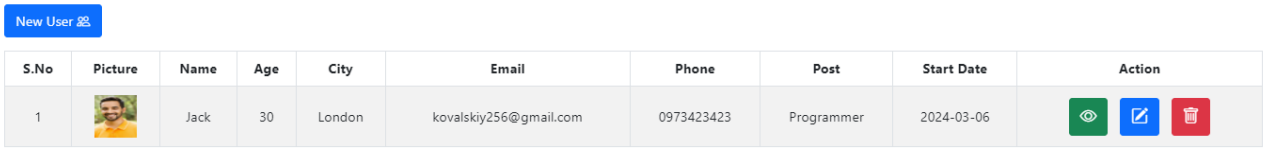


Рисунок 3.7 Оновлений профіль працівника

Для перевірки коректності роботи додатку додаємо ще декілька працівників.

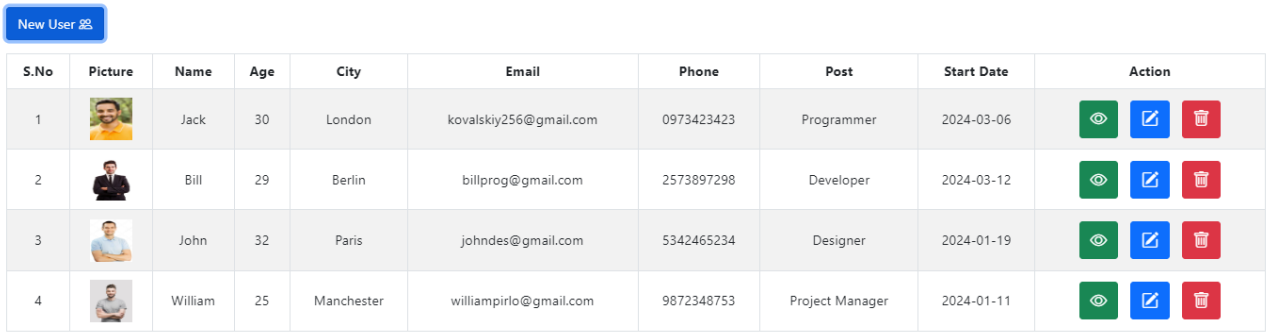


Рисунок 3.8 Приклад заповненої таблиці.

Під час тестування CRUD додатку можна визначити такі моменти:

- дані про користувачів відображаються у вигляді таблиці з відповідними стовпцями, такими як ім'я, вік, місто тощо. Це дозволяє користувачеві легко переглядати та керувати інформацією;
- локальне збереження даних дозволяє усі додані, відредаговані та видалені дані зберігати в локальному сховищі браузера за допомогою `localStorage`, що дає можливість зберегати дані між сеансами користування без необхідності підключення до сервера.

Щодо покращень, можна розглянути додавання валідації даних у формі, щоб забезпечити коректне введення інформації користувачем, а також додати

функціонал сортування та фільтрації для зручного перегляду даних у таблиці.

ВИСНОВКИ

В ході виконання даної роботи було розроблено CRUD додаток. Було досягнуто успішної інтеграції базових операцій створення, читання, оновлення та видалення даних з веб-інтерфейсу. Використання JavaScript дозволило ефективно керувати взаємодією з користувачем та обробкою даних на клієнтській стороні, забезпечуючи динамічність та інтерактивність додатку. Такий підхід дозволяє зручно та швидко вносити зміни в інтерфейс, оптимізувати процес взаємодії з користувачем та забезпечувати стабільну роботу додатку навіть при великій кількості одночасних запитів. Було проведено порівняльну роботу методами оптимізації, розглянуто фреймворки а саме їх роль та мета. JavaScript також забезпечує можливість створення динамічних елементів інтерфейсу, взаємодії зі сторонніми сервісами та інтеграції різноманітних функціональних можливостей у веб-додаток. Використання інструментів розробки, таких як Visual Studio Code, сприяє швидкому та зручному процесу написання коду та налагодженню додатку. У процесі виконання роботи було розглянуто сучасні засоби та техніки розробки, відзначені їхні позитивні та негативні сторони, проведено порівняльний аналіз між технологіями. Проведено дослідження ефективності використання різних технологій для реалізації функціонального і готового до експлуатації Web-додатку. Для цього було проаналізовано переваги програмного забезпечення, редакторів вихідного коду, фреймворків та засобів, які допомагають полегшити розробку. Розглянуті та використані технології показали, що розумний підбір технологій для розробки проекту може полегшити процес розробки та навіть зробити його цікавим.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Баров В. О. Переваги, використання та вивчення JavaScript. *Головний сайт Ніжина, де ви знайдете всю інформацію про новини, події та історію міста*., м. Київ, 1 груд. 2023 р. URL: https://mynizhyn.com/news/ukraina-i-svit/17122-perevagi-vikoristannja-ta-vivchennja-javascript.html (дата звернення: 16.04.2024).
2. Іващук І. Б. Чому JavaScript – перспективна мова програмування. *Найбільша спільнота розробників України. Все про IT: цікаві статті, інтерв'ю, розслідування, дослідження ринку, свіжі новини та події.*, м. Харків, 17 березня 2023 р.. URL: https://dou.ua/forums/topic/35184/ (дата звернення: 16.04.2024).
3. Гусієва Т. Ю. Як використовувати JavaScript для створення динамічного контенту на веб-сторінках. - IT рейтинг UA. *IT рейтинг України*., м. Київ, 8 червня 2022 р. URL: https://it-rating.ua/yak-vikoristovuvati-javascript-dlya-stvorennya-dinamichnogo-kontentu-na-veb-storinkah (дата звернення: 16.04.2024).
4. Коваль А. М. Роль JavaScript у розробці веб-додатків. *Redstone*. URL: https://redstone.media/rol-javascript-u-rozrobtsi-veb-dodatkiv (дата звернення: 16.04.2024).
5. Немчинський С. В. Фреймворки Javascript: застосування в реальних проєктах. *FoxmindEd*. URL: https://foxminded.ua/freimvorky-javascript/ (дата звернення: 16.04.2024).
6. Панасюк С. О. Основні принципи вебдизайну для створення привабливого інтерфейсу користувача While Web Production. *While Web Production*., м. Одеса, 1 груд. 2024 р. URL: https://whileweb.com/uk/blog/osnovni-principi-vebdizajnu-dlya-stvorennya-privablivogo-interfejsu/ (дата звернення: 16.04.2024).
7. Мацик О. С. Window: localStorage property - Web APIs | MDN. *MDN Web Docs*. URL: https://developer.mozilla.org/en-

US/docs/Web/API/Window/localStorage (дата звернення: 16.04.2024).

8. Денисенко А. С. Що таке CRUD простими словами: функції, переваги та приклади. *Highload.today - медіа для розробників*., м. Львів, 8 липня 2023 р. URL: https://highload.today/uk/shho-take-crud-prostimi-slovami-funktsiyi-perevagi-ta-prikladi/ (дата звернення: 17.04.2024).

9. Мушкич В. О. Що таке адаптивність веб сайту під різні пристрої - WebTune. *Webtune Web Development & Marketing*., м. Львів, 27 вересня 2023 р. URL: https://webtune.com.ua/statti/internet-marketing/yak-pereviryty-adaptyvnist-za-dopomogoyu-brauzera/ (дата звернення: 17.04.2024).

10. Терніков М. В. Бібліотеки та фреймворки JavaScript. *Anywhere Club*., м. Київ, 23 березня 2023 р. URL: https://aw.club/global/uk/blog/javascript-libraries-and-frameworks-how-to-choose (дата звернення: 17.04.2024).

11. Interfax-Ukraine. Основні принципи SEO оптимізації: перегляд основних факторів, які впливають на рейтинг сайту в пошукових системах. *Інтерфакс-Україна*. URL: https://interfax.com.ua/news/press-release/921492.html (дата звернення: 17.04.2024).

12. Посібник з оптимізації JavaScript файлів. *DevZone*. URL: https://devzone.org.ua/post/posibnyk-z-optymizatsiyi-javascript-fayliv (дата звернення: 17.04.2024).

13. Компілятор Javascript: завдання з оптимізації та етапи роботи. *FoxmindEd*. URL: https://foxminded.ua/kompiliator-javascript/ (дата звернення: 17.04.2024).

14. A Guide to Optimizing JavaScript Files â□□ SitePoint. *SitePoint â□□ Learn HTML, CSS, JavaScript, PHP, Ruby & Responsive Design*. URL: https://www.sitepoint.com/optimizing-javascript-files/ (дата звернення: 17.04.2024).

15. Johns R. 10 Best JavaScript Frameworks in 2024 [Updated]. *Hackr.io*. URL: https://hackr.io/blog/best-javascript-frameworks (дата звернення: 18.04.2024).

16. Theo - t3.gg. JavaScript Framework Tier List, 2023. *YouTube*. URL: https://www.youtube.com/watch?v=WJRf7dh5Zws (дата звернення:

18.04.2024).

17. SuperSimpleDev. What is a JavaScript Framework? (in detail), 2020. *YouTube*. URL: https://www.youtube.com/watch?v=Ka77djMkSwg (дата звернення: 18.04.2024).

18. Web Dev Simplified. JavaScript Cookies vs Local Storage vs Session Storage, 2019. *YouTube*. URL: https://www.youtube.com/watch?v=GihQAC1I39Q (дата звернення: 18.04.2024).

19. freeCodeCamp.org. cookies vs localStorage vs sessionStorage - Beau teaches JavaScript, 2017. *YouTube*. URL: https://www.youtube.com/watch?v=AwicscsvGLg (дата звернення: 18.04.2024).

20. Window localStorage Property. *W3Schools Online Web Tutorials*. URL: https://www.w3schools.com/jsref/prop_win_localstorage.asp (дата звернення: 19.04.2024).

21. Contributors to Wikimedia projects. Visual Studio Code - Wikipedia. *Wikipedia, the free encyclopedia*. URL: https://en.wikipedia.org/wiki/Visual_Studio_Code (дата звернення: 19.04.2024).

22. Microsoft. Documentation for Visual Studio Code. *Visual Studio Code - Code Editing. Redefined*. URL: https://code.visualstudio.com/docs (дата звернення: 19.04.2024).

23. Codecademy. What is CRUD? | Codecademy. *Codecademy*. URL: https://www.codecademy.com/article/what-is-crud (дата звернення: 19.04.2024).

24. Левіна Е. Что такое CRUD. *OrbitSoft Блог*. URL: https://orbitsoft.com/ru/blog/crud/ (дата звернення: 19.04.2024).

25. Основні принципи веб-дизайну інтерфейсу користувача, яких слід дотримуватися у 2024 році | Stfalcon. *Custom Software Development Company | Stfalcon.com*. URL: https://stfalcon.com/uk/blog/post/user-interface-web-design-principles (дата звернення: 19.04.2024).

26. Сучасний підручник з JavaScript. *Сучасний підручник з JavaScript*.

URL: https://uk.javascript.info/ (дата звернення: 20.04.2024).

27. Intellipaat. Top 5 JavaScript Frameworks For 2024 | Top JavaScript Frameworks For Web Development | Intellipaat, 2024. *YouTube*. URL: https://www.youtube.com/watch?v=ZhH0egZMjTs (дата звернення: 20.04.2024).

28. Understanding client-side JavaScript frameworks - Learn web development | MDN. *MDN Web Docs*. URL: https://developer.mozilla.org/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks (дата звернення: 20.04.2024).

29. Ekainu G. A. 9 Best JavaScript Frameworks to Use in 2024. *Composable Personalization and Experimentation for Dynamic Websites, Headless CMSs, and MACH | Ninetailed*. URL: https://ninetailed.io/blog/best-javascript-frameworks/ (дата звернення: 20.04.2024).

30. Перестороніна Е. Що таке SEO і навіщо потрібна пошукова оптимізація. *Netpeak Journal – медіа про інтернет-маркетинг та онлайн-бізнес у деталях*. URL: https://netpeak.net/uk/blog/shcho-take-seo-i-navishcho-potribna-poshukova-optimizatsiya/ (дата звернення: 20.04.2024).

Додаток А

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
Навчально-науковий інститут Інформаційних технологій
Кафедра Комп'ютерної інженерії

КВАЛІФІКАЦІЙНА РОБОТА
НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВРА

на тему: «Оптимізація ефективності веб-інтерфейсів через використання технік та інструментів JavaScript»

Виконав студент групи КІД-41
Ковальський Богдан

Керівник кваліфікаційної роботи:
Куфтеріна Світлана , старший викладач кафедри КІ

Київ – 2024

Мета роботи – дослідити та проаналізувати різноманітні техніки та інструменти використання JavaScript для оптимізації ефективності веб-інтерфейсів. Вона спрямована на вивчення та розуміння того, як JavaScript може бути використаний для поліпшення швидкодії, реактивності та загального користувацького досвіду веб-додатків та веб-сайтів.

Об'єкт дослідження – процес створення CRUD-додаток для веб-сторінки.

Предмет дослідження – розгортання та оптимізації CRUD додатку з використанням JavaScript та інших технологій веб-розробки.

АКТУАЛЬНІСТЬ

В рамках цієї дипломної роботи детально розглянемо поняття веб-інтерфейсу та його роль у веб-розробці. Проведемо аналіз основних принципів оптимізації веб-інтерфейсів та розглянемо різні техніки та інструменти JavaScript, які можуть бути використані для поліпшення їх ефективності. Ця дипломна робота має на меті внести вклад у підвищення якості та ефективності веб-інтерфейсів, що відіграють важливу роль у взаємодії користувачів з інформацією та сервісами в Інтернеті.

3

ЕФЕКТИВНІСТЬ ВЕБ-ІНТЕРФЕЙСУ

4

JAVASCRIPT ТА ЇЇ РОЛЬ У ВЕБ-РОЗРОБЦІ

JavaScript – динамічна, об'єктно-орієнтована скриптова мова програмування і є діалектом мови ECMAScript. Найчастіше використовується для створення сценаріїв веб-сторінок, що надає можливість на стороні клієнта (пристрої кінцевого користувача) взаємодіяти з користувачем, керувати браузером, асинхронно обмінюватися даними з сервером, змінювати структуру та зовнішній вигляд веб-сторінки.

5

LOCAL STORAGE, SESSION STORAGE

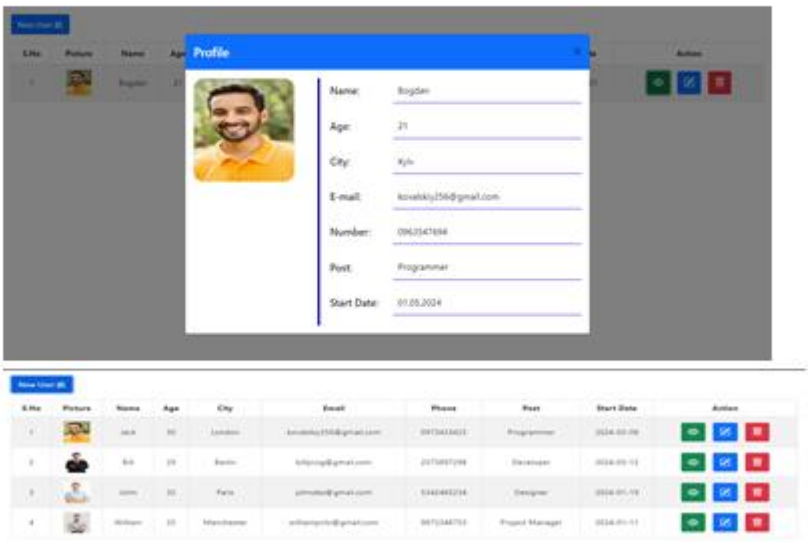
6

ФРЕЙМВОРКИ JAVACRIPT

Фреймворк - це набір інструментів, функцій і структур, призначених для розробки веб-додатків на JavaScript. Це готова основа, яка допомагає програмістам упорядкувати та спростити процес створення складних програм.
Його мета – надання розробникам зручного інструментарію для створення веб-додатків, які будуть ефективними та легко масштабованими. Вони забезпечують структуру і організацію коду, що сприяє покращенню читабельності і розуміння програмного забезпечення.

7

CRUD ДОДАТОК ТА ЙОГО РЕАЛІЗАЦІЯ



8

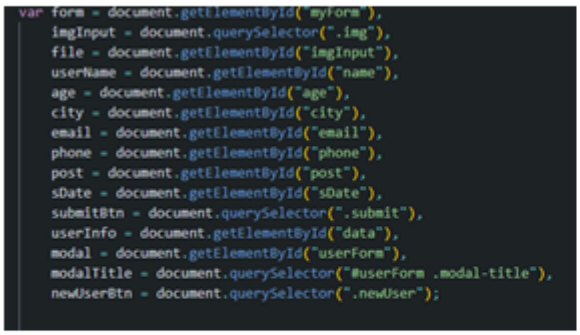
НАПИСАННЯ HTML ТА CSS КОДУ

```
<div class="modal fade" id="userform">
    <div class="modal-dialog modal-dialog-centered modal-lg">
        <div class="modal-content">

            <div class="modal-header">
                <h4 class="modal-title">Fill the Form</h4>
                <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>
            </div>

            <div class="modal-body">

                <form action="#" id="myform">

                    <div class="card imgholder">
                        <label for="imgInput" class="upload">
                            <input type="file" name="" id="imgInput">
                            <i class="bi bi-plus-circle-dotted"></i>
                        </label>
                        <img src="./image/Profile Icon.webp" alt="" width="300" height="200" class="img">
                    </div>

                    <div class="inputfield">
                        <div>
                            <label for="name">Name:</label>
                            <input type="text" name="" id="name" required>
                        </div>
```

```
.modal-body form {
    display: flex;
    justify-content: space-between;
    align-items: flex-start;
    padding: 0;
}

.modal-body form .imgholder{
    width: 200px;
    height: 200px;
    position: relative;
    border-radius: 20px;
    overflow: hidden;
}

.imgholder .upload{
    position: absolute;
    bottom: 0;
    left: 10;
    width: 100%;
    height: 100px;
    background: rgba(0,0,0,0.3);
    display: none;
    justify-content: center;
    align-items: center;
    cursor: pointer;
}
```

9

JAVASCRIPT



```
var form = document.getElementById("myForm"),
    imgInput = document.querySelector(".img"),
    file = document.getElementById("imgInput"),
    userName = document.getElementById("name"),
    age = document.getElementById("age"),
    city = document.getElementById("city"),
    email = document.getElementById("email"),
    phone = document.getElementById("phone"),
    post = document.getElementById("post"),
    sDate = document.getElementById("sDate"),
    submitBtn = document.querySelector(".submit"),
    userInfo = document.getElementById("data"),
    modal = document.getElementById("userForm"),
    modalTitle = document.querySelector("#userForm .modal-title"),
    newUserBtn = document.querySelector(".newUser");
```

Визначає змінні для доступу до елементів HTML за їх ідентифікаторами або класами. Забезпечує доступ до цих елементів для подальшої обробки подій, зчитування введених даних користувачем та взаємодії з ними в JavaScript коді. Наприклад, він може слугувати для валідації введених даних перед їх відправленням на сервер або для відображення введених даних в модальному вікні.

Забезпечує взаємодію між HTML та JavaScript, дозволяючи динамічно маніпулювати вмістом сторінки та взаємодіяти з користувачем через події, такі як кліки на кнопках або відправка форми.

10

JAVASCRIPT

```
file.onchange = function() {
    if (file.files[0].size < 1000000) { // 1MB = 1000000
        var fileReader = new FileReader();

        fileReader.onload = function(e) {
            imgUrl = e.target.result;
            imgInput.src = imgUrl;
        };

        fileReader.readAsDataURL(file.files[0]);
    } else {
        alert("This file is too large!");
    }
};
```

```
function deleteInfo(index) {
    if (confirm("Are you sure want to delete?")) {
        getData.splice(index, 1);
        localStorage.setItem("userProfile", JSON.stringify(getData));
        showInfo();
    }
}
```

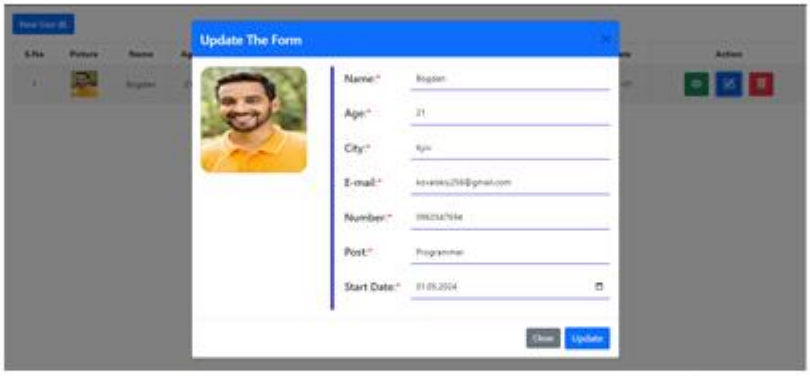
```
form.addEventListener('submit', (e) => {
    e.preventDefault();

    const information = {
        picture: imgInput.src == undefined ? "./image/Profile Icon.webp" : imgInput.src,
        employeeName: userName.value,
        employeeAge: age.value,
        employeeCity: city.value,
        employeeEmail: email.value,
        employeePhone: phone.value,
        employeePost: post.value,
        startDate: sDate.value
    };

    if (!isEdit) {
        getData.push(information);
    } else {
        isEdit = false;
        getData[editId] = information;
    }

    localStorage.setItem('userProfile', JSON.stringify(getData));

    submitBtn.innerText = "Submit";
    modalTitle.innerHTML = "Fill The Form";

    showInfo();

    form.reset();

    imgInput.src = "./image/Profile Icon.webp";
```

11

ВИГЛЯД ДОДАТКУ

12

ВИГЛЯД ДОДАТКУ



13

ВИСНОВКИ

1. Опрацьовано теоретичний матеріал. Проведено порівняльну роботу методами оптимізації.
2. Визначено роль мови програмування Javascript у розробці веб-додатків
3. Розглянуто фреймворки, їх роль та мету.
4. В ході виконання даної роботи було розроблено CRUD додаток. Успішно інтегровано операції створення, читання, оновлення та видалення даних з веб-інтерфейсу.
5. Розглянуто сучасні засоби та техніки розробки, відзначені їхні позитивні та негативні сторони.
6. Проведено порівняльний аналіз між технологіями.

14