КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет комп'ютерних наук та кібернетики
Кафедра теорії та технології програмування

Кваліфікаційна робота

на здобуття ступеня бакалавра

за спеціальністю 122 Комп'ютерні науки

на тему:

РОЗРОБКА ПРОГРАМИ ДЛЯ УПРАВЛІННЯ ЛАБОРАТОРНИМИ РОБОТАМИ. РОЗРОБКА ВЕБ ДОДАТКУ

Виконав студент 4-го курсу
Карпенко Максим Іванович _____

Науковий керівник:
асистент
Федорус Олексій Мстиславович _____

Засвідчую, що в цій роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____

Роботу розглянуто й допущено до захисту
на засіданні кафедри теорії та
технології програмування
«____» _____ 20__ р.,

протокол № ___
Завідувач кафедри
М. С. Нікітченко _____

Київ – 2021

РЕФЕРАТ

Обсяг роботи 42 сторінки, 24 ілюстрації, 13 джерел посилань.

ВЕБ ДОДАТОК, REACT, FLUX АРХІТЕКТУРА, REDUX, WEBPACK, JSX, MATERIAL UI.

Об’єктом роботи є процес розробки веб додатка.

Предметом роботи є веб додаток для управління лабораторними роботами.

Метою роботи є створення клієнтської частини веб системи для управління лабораторними роботами.

Інструменти розробки: Visual Studio Code, JavaScript бібліотека React, Redux, JSX.

Були отримані наступні **результати**: виконано загальний огляд популярних технологій розробки клієнтської частини веб додатків. Було досліджено та застосовано бібліотеку React разом з бібліотекою для збереження стану програми Redux. Розроблено веб додаток для управління лабораторними роботами.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

HTTP – HyperText Transfer Protocol (прикладний протокол для передачі гіпертекстових документів);

TCP – Transmission Control Protocol (протокол передачі даних);

SSH – Secure Shell (мережевий протокол);

HTTPS – HyperText Transfer Protocol Secure (протокол з додатковим шаром шифрування між HTTP та TCP);

URL – Uniform Resource Locator (стандартизована адреса певного сайта);

HTML – HyperText Markup Language (мова розмітки гіпертексту);

CSS – Cascading Style Sheet (каскадні таблиці стилів);

GIF – Graphics Interchange Format (формат для обміну повідомленнями);

CMS – Content Management System (система управління контентом сайта);

PHP – Hypertext Preprocessor (мова програмування);

DOM – Document Object Model (об’єктна модель документа);

API – Application Programming Interface (спосіб взаємодії між програмами);

ПК – персональний комп’ютер;

ВСТУП

Оцінка сучасного стану об’єкта розробки. У наш час розробка веб додатків займає значну частину ринку програмного забезпечення. Цьому сприяє велика кількість користувачів мережі інтернет, що приєднується до всесвітньої павутини щодня.

З кожним днем збільшується навантаження на веб сервіси. Системи стають масштабнішими на складнішими. Веб додатки витісняють програми, що інсталюються на ПК, що є очевидним зі зростанням поширення якісного інтернет покриття по всьому світу. Цим пояснюється такий бурхливий попит на якісні системи, якими можна користуватися з будь-якого браузера.

Актуальність роботи та підстави для її виконання. Пандемія – це перша глобальна проблема сучасного людства, з якою важко впоратись але можливо пристосуватись. Однією з найвразливіших виявилась система освіти, яка потребує впровадження онлайн технологій. У зв’язку з переходом усіх сфер життя в онлайн створення веб додатка для роботи з лабораторними роботами - ковток нових можливостей для студентів та викладачів. З його допомогою можна покращити процес навчання.

Об’єктом дослідженння є процес розробки веб додатка.

Предметом дослідженння є веб додаток для управління лабораторними роботами.

Мета й завдання роботи. Метою кваліфікаційної роботи є створення веб системи для управління лабораторними роботами. Для досягнення цієї мети було поставлено такі задачі:

- дослідити існуючі веб додатки для навчання онлайн;
- обрати технології розробки для створення клієнтської частини веб додатка;
- розробити гнучку і масштабовану архітектуру додатка;

- імплементувати веб додаток за допомогою вибраних технологій та створеної архітектури

Методи та засоби розробки.

Під час розробки програмного продукту використана ітераційна модель, заснована на деяких принципах. Розробляється початкова версія продукту, яка передається кінцевим користувачам для оцінки, після чого продукт дороблюється, враховуючи думку замовника. Аналогічно розробляються, передаються й оцінюються проміжні версії програмного продукту, поки не з’явиться повністю готовий продукт, який відповідає всім вимогам замовника.

У якості середовища для розробки був використаний текстовий редактор Visual Studio Code з безкоштовними плагінами для збільшення швидкості та якості розробки веб додатка.

РОЗДІЛ 1
АНАЛІЗ АКТУАЛЬНОСТІ ТА ОГЛЯД ІСНУЮЧИХ НА РИНКУ ВЕБ ДОДАТКІВ

1.1. Аналіз актуальності

У 2020 році світ сколихнула епідемія вірусу, який змінив правила життя для всього людства. Для більшості людей пандемія внесла корективи і в робочі процеси. Багато хто був вимушений перемкнутися на роботу з дому, навіть навчання усіх видів почали проводити у віддаленому форматі. Проте індустрія не була готовою до таких змін. Ще до пандемії населення світу стикалося зі значними труднощами в реалізації права на освіту як одного з основних прав людини. Забезпечення безперервності навчання на тлі закриття шкіл стало пріоритетним завданням для урядів всіх країн світу, багато з яких задіяли інтернет технологіях і зобов'язали вчителів вести заняття онлайн.

Університети пропонують студентам власний онлайн контент. Проте багато викладачів ніколи не розробляли і не читали онлайн курси. Викладачам необхідно інакше підходити до проведення лекцій, семінарів і лабораторних та покращувати навички викладання в інтернеті.

Студентам та викладачам важко звикнути до нових інструментів, що мають вигляд веб додатків. На даний час, представлено широкий список різних інтернет ресурсів, що допомагають батькам, вчителям, учням, студентам та викладачам полегшити процес навчання. Більшість цих рішень безкоштовна та задовольняють велику кількість потреб. Важливими вимогами до систем стали їхня надійність, пропускна здатність інтернет каналів, простота створення та розміщення контенту, доступність сервісів і платформ для викладачів і учнів. Не всі системи можуть похвалитися поєднанням всіх цих вимог. Не існує єдиної платформи чи інструмента, який вирішував би всі ці проблеми. В той час існуючі додатки показали багатократний ріст і розвиток: Google Classroom, Duolingo.

Отже, через всі вищеперелічені проблеми було прийнято рішення розробити веб додаток, який допоможе викладчам та студентам спростити навчальний процес.

1.2 Огляд існуючих веб додатків

Google Classroom – один з найвідоміших веб додатків, який використовується університетами в процесі навчання. Це безплатний веб сервіс, розроблений компанією Google в 2014 році, який покликаний спростити створення, поширення і оцінку завдань безпаперовим способом. Основною метою додатка є спрощення процесу обміну файлами між вчителями та учнями [1]. Основною перевагою платформи є її ціна – вона безкоштовна.

Google Classroom поєднує в собі Google Диск для створення і поширення завдань, набір сервісів Google для створення документів, презентацій і електронних таблиць, Gmail для спілкування і Календар Google для планування. Учні можуть бути запрошені в курс за унікальним кодом або автоматично імпортовані з шкільного домену. При створенні курсу створюється окрема папка на відповідному диску користувача, де учень може представити роботу для оцінки вчителю. Мобільні додатки, доступні для пристроїв iOS і Android, дозволяють користувачам робити фотографії і прикріплювати їх до завдань, обмінюватися файлами з інших додатків і отримувати доступ до інформації в автономному режимі. Вчителі можуть стежити за успішністю кожного учня, а після оцінки вчителя можуть повертати роботу разом з коментарями.

Створивши курс, ви потрапляєте на головну панель: в горизонтальному меню три вкладки: «Стрічка», в якій за аналогією, наприклад, з Facebook, видно всі оновлення; «Завдання» з усіма матеріалами і «Користувачі», де розміщена інформація про однокласників і викладачів. Якщо ви створили курс, ви також матимете вкладку «Оцінки». Особливо мені сподобався мінімалістичний дизайн, властивий продуктам Google: від навчання ніщо не буде відволікати. Він був розроблений за допомогою Material Design. На рисунку 1.2.1 показано як виглядає стартовка сторінка курсу для викладача.

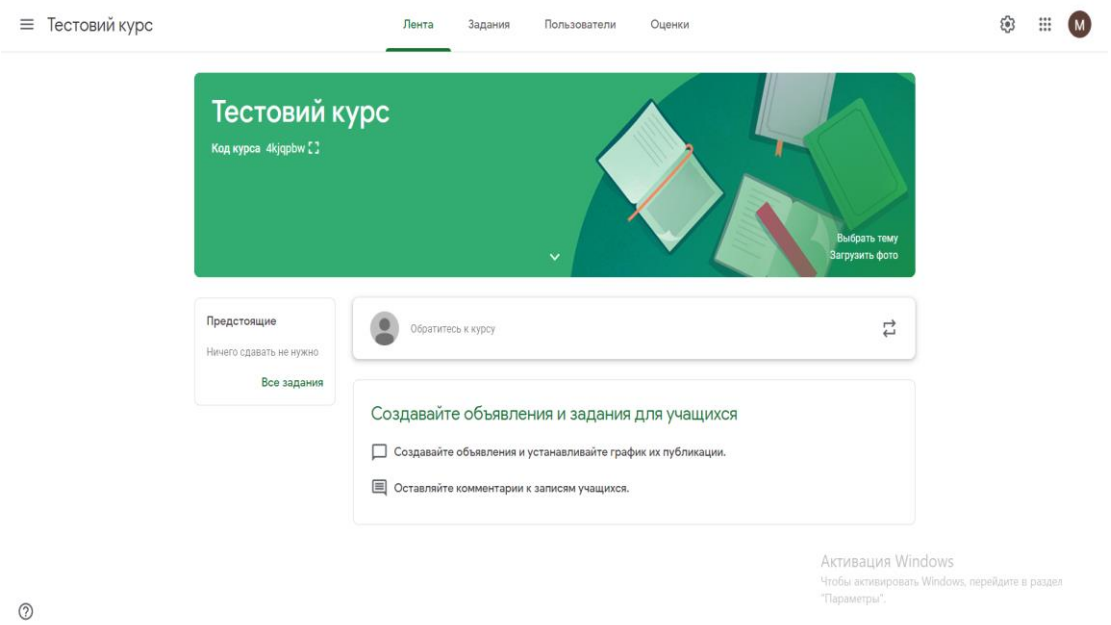

Рисунок 1.2.1 Стартова сторінка Google Classroom

У кожного додатку є свої переваги та недоліки. Перевагами даної платформи є:

- безкоштовне використання;
- відсутність реклами, що відволікає від навчання;
- наявність вкладки з оцінками;
- можливість завантаження файлів будь-якого формату;
- можливість проводити відеоконференції за допомогою Google Meets;
- відстежування дедлайнів здачі робіт;
- автоматизація процесу нотифікацій(при додаванні нового завдання учень отримує листа сповіщення на електронну адресу)

Недоліками платформи є:

- обмеження кількості учасників курсу для одного облікового запису вчителя у вигляді 200 учнів;
- необхідність навчання вчителів та учнів користування платформою;
- реєстрація лише через елетроний сервіс Gmail;
- неможливість розділити в системі університетські групи на підгрупи.

Moodle – інша не менш відома система для управління курсами [2]. Платформа являє собою простір для спільної роботи учнів та вчителів. В Moodle доступні різні можливості для відстеження успішності учнів, а також є підтримка масової реєстрації з безпечною аутентифікацією. Система має досить гнучкий інтерфейс з можливістю конфігурації макетів та дизайну окремих сторінок. Платформу можна інтегрувати з великою кількістю програмного забезпечення, включаючи інструменти для спілкування, спільної роботи, управління документами да інші додатки для підвищення продуктивності.

На платформі можна скористатися демо версією і створити тестову версію вашого курсу. На рисунку 1.2.2 показано як виглядає демо версія.

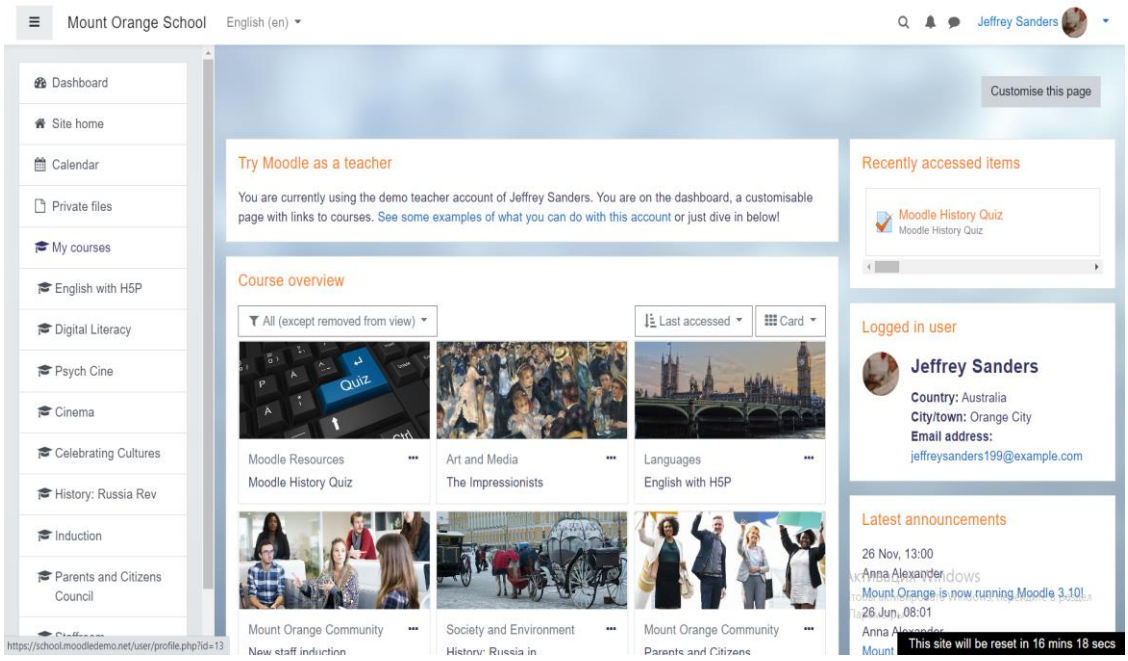

Рисунок 1.2.2 Демо версія Moodle

Найголовнішою перевагою, на яку відразу зветрають увагу користувачі, - безкоштовне використання Moodle. Система працює за схемою Open Source – відкритого вихідного коду. За рахунок цього велика кількість програмістів створила близько 1500 плагінів для системи Moodle.

Перевагами даної платформи є:

- безкоштовне використання;
- можливість розмежовувати доступ до навчальних матеріалів і курсів;

- велика кількість безкоштовних плагінів для розширення потенційних способів використання платформи;
- можливість інтеграції з іншими сервісами;
- відстежування дедлайнів здачі робіт;
- різноманітність налаштувань завдань.

Недоліками платформи є:

- складність в установці та налаштуванні системи;
- необхідність навчання вчителів та учнів користування платформою;
- доволі застарілий і не дуже товариський дизайн системи.

РОЗДІЛ 2
ТЕОРЕТИЧНІ ОСНОВИ РОЗРОБКИ ВЕБ ДОДАТКІВ

2.1 Веб додатки, їх види, місце в сучасній розробці

Веб додаток (або веб програма) - це прикладне програмне забезпечення, яке працює на веб сервері, на відміну від програм на ПК, які запускаються локально в операційній системі пристрою. Доступ до веб додатка користувач здійснює через браузер з активним підключенням до мережі інтернет. Обмін інформацією відбувається мережі. Приклади часто використовуваних веб додатків включають: веб пошту, роздрібні продажі в інтернеті та інші.

Раніше навантаження на обробку програми розподілялося між кодом на сервері та кодом, встановленим на кожному клієнті локально. Іншими словами, програма мала власну, заздалегідь скомпільовану, клієнтську програму, яка слугувала його користувальницьким інтерфейсом і повинна була встановлюватися окремо на ПК кожного користувача. Оновлення серверного коду програми, як правило, також вимагає оновлення коду на стороні клієнта, встановленого на кожній робочій станції користувача, що збільшує вартість підтримки та зменшує продуктивність. Крім того, як клієнтські, так і серверні компоненти програми були, як правило, тісно пов'язані з певною архітектурою комп'ютера та операційною системою, і перенесення їх на інші часто було надмірно дорогим для всіх, крім найбільших додатків. На відміну від них, веб програми використовують веб документи, написані у стандартному форматі, наприклад, HTML, які підтримуються різноманітними браузерами. Веб програми можна розглядати як конкретний варіант програмного забезпечення клієнт-сервер, де клієнтське програмне забезпечення завантажується на клієнтську машину під час відвідування відповідної веб сторінки, використовуючи стандартні процедури, такі як HTTP. Оновлення програмного забезпечення клієнта може відбуватися щоразу, коли відвідується веб сторінка. Під час сеансу браузер інтерпретує та відображає сторінки та виступає універсальним клієнтом для будь-якої веб програми. На рисунку 2.1 показано спрощену схему веб додатка.

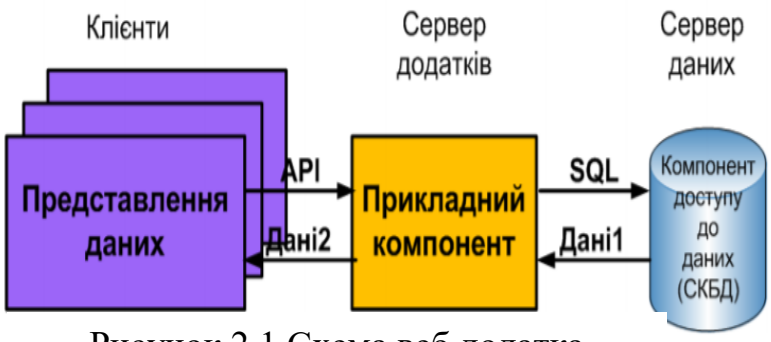

Рисунок 2.1 Схема веб додатка

Існують різні види веб додатків. Статичний веб додаток – найпростіший тип. Ці застосутнки часто створюються з використанням CSS та HTML і можуть обробляти анімований вміст, такий як GIF та відео. Зміна вмісту, включеного в статичні веб-програми, може бути складним, і для внесення серйозних змін потрібно завантажити HTML код, а також змінити та завантажити його на сервер. Поширений приклад статичних веб програм включає онлайн-портфоліо або лендінги. У цьому ж ключі ви можете уявити цільову сторінку як статичну веб-програму з відображеною на ній контактною інформацією. Динамічний веб додаток – складніший тип. Цей веб додаток є більш технічно складним у порівнянні з вищезазначеним статичним додатком. У випадку з динамічними веб-додатками вони створені для зберігання баз даних або форумів з постійною можливістю оновлювати або змінювати наявну інформацію. Зазвичай це відбувається в системі управління вмістом або CMS. Багато різних мов можна використовувати з динамічними програмами. Найпоширенішою є мова PHP, оскільки її найпростіше зрозуміти, коли йдеться про структурування вмісту.

Редагувати вміст простіше за допомогою динамічних веб додатків, а оновлення самого вмісту може бути досить простим. Однак серверна частина або частина програмування може бути більш складною залежно від сервера та інших факторів. З огляду на це, елементи дизайну можна легко модифікувати відповідно до ваших особистих уподобань.

У сучасному світі кожен бізнес портребує підвищення продуктивності та ефективності. Веб додаток допомагає використовувати менші витрати на

підтримку та розробку і дозволяє отримати доступ до ділової інформації в будь-якій точці світу. Це економить час та гроші, а також зв’язок між споживачами та діловими партнерами.

2.2 Переваги та недоліки веб додатків

Клієнтська частина веб додатків розробляються такими мовами програмування, як HTML та CSS, які є найпростішими в освоєнні. На відміну від додатків на телефон, веб додаток може використовуватись на пристроях, в яких є браузер. Веб додатки не вимогливі до ресурсів та можуть бути запущені навіть зі старого мобільного телефону. Коли з’являється нова версія нативного додатку, користувачам треба вирішувати проблеми, зв’язані з оновленням на їх пристроях копії. У випадку з браузерним додатком таких проблем не буває – існує лише одна версія, в якій працюють усі користувачі. Ця ж версія автоматично оновляється розробниками системи, тому користувачі можуть спокійно користуватися веб додатком без зайвих рухів. Веб додаток не потребує встановлення на пристрій. Ці програми працюють у браузері пристрою через просту. Їх не потрібно завантажувати та встановлювати з магазинів додатків, таких як Google Play або Apple Store. Це означає економію грошей, оскільки пряме посилання через веб-додаток є безкоштовним. Розробка веб додатків - дешевший вид розробки додатків. Розробка нативного або інтерпретованого додатка тягне за собою більш високу вартість, але шанси на успіх додатку набагато більше. Веб додатки дозволяють своїм користувачам бути насправді мобільними. Ви можете зберігати результат своєї роботи на сервері, і у випадку необхідності мати доступ до них із будь якої точки планети, де є вихід у мережу інтернет.

Як було сказано вище, один веб додаток може використовуватись на всіх пристроях. Але, звичайно ж, веб сайт повинен бути запрограмований таким чином, щоб він відображався незалежно від операційної системи пристрою. Якщо це не адаптивний веб сайт, іноді виникають проблеми відображення. Підключення до інтернету буде абсолютно необхідним для його запуску. В іншому випадку ви не

зможете переглядати веб-сайт, і веб додаток вам не принесе користі. Що ускладнює роботу з веб додатками, наприклад, під час подорожі. Кожне перезавантаження (або оновлення сторінки) викликає помітну затримку, викликану необхідністю встановити HTTP з'єднання, обробити запит на сервері, передати по мережі у відповідь HTTP повідомлення і перезавантажити сторінку браузером. Це створює переривчастий режим роботи та уповільнює його.

2.3 Принцип роботи веб додатків

В основу роботи веб додатку покладено клієнт-серверну архітектуру. Принцип роботи архітектури полягає в тому, що запити, які формуються на клієнті, який ще називають Frontend, обробляються на віддаленій машині (сервері) і записуються до бази даних. Цю частину називають Backend. На рисунку 2.3 можна це побачити.

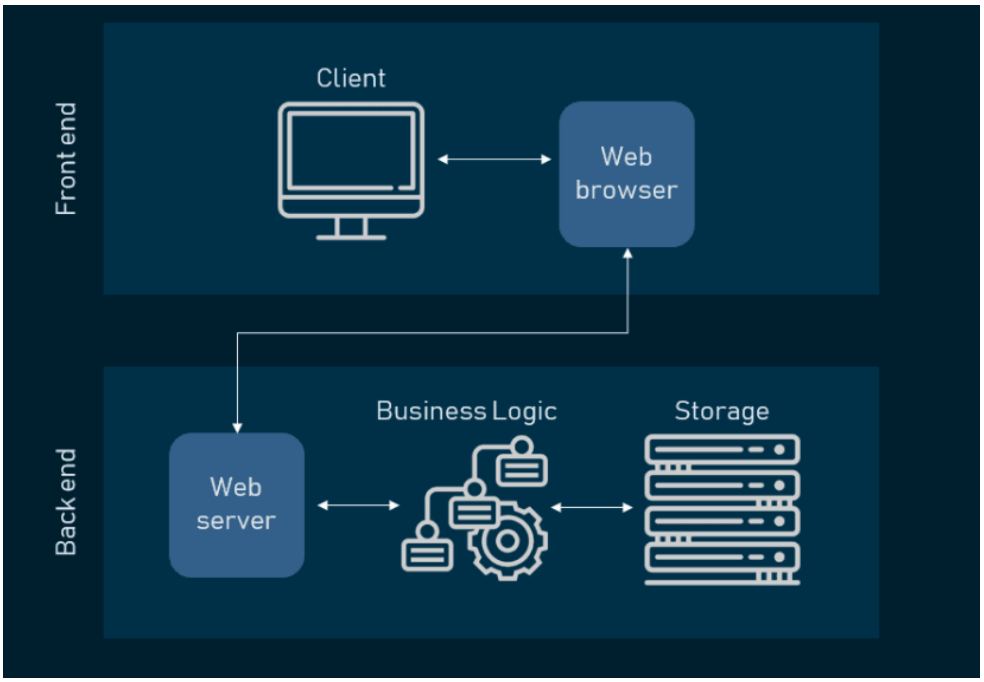

Рисунок 2.3 Клієнт-серверна архітектура

Сервер та клієнт взаємодіють в мережі інтернет або в будь-який інший комп'ютерної мережі за допомогою різних мережевих протоколів, наприклад, HTTP, HTTPS, SSH.

Як працює дана архітектура пояснемо на прикладі візиту до сайта Google. Спочатку ви відкриваєте браузер та пишете в браузері URL сайта, тобто google.com. Після того як ви натискаєте кнопку пошук, браузер готується розпізнати введений вами URL. Це робиться для того, щоб дізнатися адрес сервера, на якому розташований даний сайт. Тому бразуер надсилає запис до DNS – системи доменних імен. Якщо ви вже відвідували даний сайт перед цим, то браузер видасть адрес сервера зі свого кеша. Потім буде надіслано запит до знайденого адреса за допомогою HTTPS протоколу.

Наступним етапом веб сервер опрацює ваш запит. Цей сервер, перехоплює запит і відправляє його в область зберігання, щоб знайти сторінку та всі дані, прикріплені до неї. Його маршрут проходить через бізнес логіку. Бізес логіка керує способом доступу до даних, тобто може повернути різні дані, якщо користувачі мають різні привілегії. Коли запит оброблено, він надсилається далі до сховища даних, тобто бази даних.

З бази даних дістається необхідна інформація. Ваша відповідь повертається до вас, і ви бачите вміст сторінки на своєму дисплеї. Графічний інтерфейс, який ви бачите, вже створений за допомогою Frontend технологій.

Сучасні веб додатки розробляються шляхом розділення основних функцій на рівні. Це дозволяє гнучко керувати та вносити зміни на кожному з рівнів. Цей архітектурний шаблон називається багаторівневою архітектурою. Він складається з таких частин: презентаційний, логічний та частина роботи з даними.

Презентаційний рівень доступний для користувачів через браузер і складається з компонентів інтерфейсу користувача та компонентів процесів інтерфейсу користувача, які підтримують взаємодію з системою. Він розроблений із використанням трьох основних технологій: HTML, CSS та JavaScript. HTML - це код, який визначає, що буде містити ваш веб-сайт, CSS контролює його вигляд. JavaScript та його фреймворки роблять ваш веб-сайт інтерактивним - реагують на дії користувача. Розробники використовують такі фреймворки JavaScript, як Angular та React, щоб зробити вміст сторінки динамічним.

Логічний рівень. Цей рівень, який також називають бізнес логікою, або логікою домену, приймає запити користувачів від браузера, обробляє їх і визначає маршрути, через які буде здійснюватися доступ до даних.. Наприклад, якщо ваша заявка є веб сайтом бронювання готелів, ділова логіка нестиме відповідальність за послідовність подій, які пройде мандрівник під час бронювання номера.

Рівень роботи з даними. Це централізоване місце, яке приймає всі виклики і забезпечує постійний доступ до даних. Рівень роботи з даними тісно пов'язаний з логічним рівнем, тому логіка знає, з якою базою даних говорити, а процес отримання даних є більш оптимізованим.

РОЗДІЛ 3

СУЧАСНІ ІНСТРУМЕНТИ РОЗРОБКИ КЛІЄНТСЬКОЇ ЧАСТИНИ ВЕБ ДОДАТКІВ

3.1. Опис HTML, CSS, JavaScript

Frontend - це візуальна частина веб додатків, з якої користувач може взаємодіяти і контактувати напряму. У Frontend входить відображення функціональних завдань, призначеного для користувача інтерфейсу, що виконуються на стороні клієнта, а також обробка запитів користувачів. По суті, фронтенд - це все те, що бачить користувач при відкритті веб сторінки. Код сторінки описує все, що користувач бачить перед собою: верстку, шрифти, розташування графічних елементів тощо (рис. 3.1.1).

Рисунок 3.1.1 Принцип роботи Frontend

Фронтенд складається з трьох основних частин, які мають назву HTML, CSS і JavaScript. Дану структуру називають делегуванням відповідальності, де HTML відповідає за структуру, CSS за дизайн, а JavaScript за інтерактивність (рис 3.1.2). Щоб веб додаток був більш досконалим, фронтенд розробник співпрацює з UI-UX дизайнерами, програмістами, для створення зручного та гарного продукту [3].

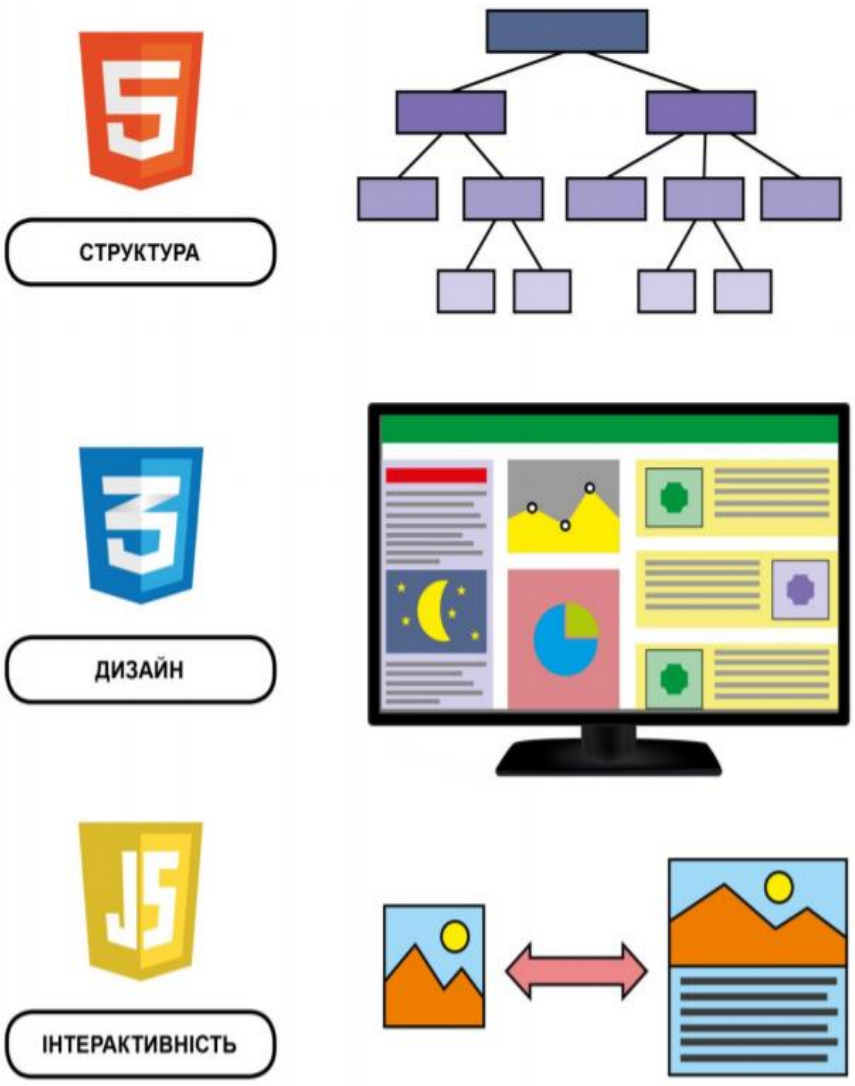

Рисунок 3.1.2 Основні складові Frontend

HTML розшифровується як мова розмітки гіпертексту. Вона дозволяє користувачеві створювати та структурувати розділи, абзаци, заголовки, посилання та цитати для веб сторінок та додатків.

HTML не є мовою програмування, тобто вона не має можливості створювати динамічну функціональність. Натомість це дозволяє упорядковувати та форматувати документи, подібно до Microsoft Word.

Під час роботи з HTML ми використовуємо прості структури коду (теги та атрибути) для розмітки сторінки веб-сайту. Наприклад, ми можемо створити абзац, помістивши вкладений текст у початковий тег і закриття .

Мова розмітки була розроблена Тімом Бернерсом-Лі. Він виступив з ідеєю створення гіпертекстової системи на основі Інтернет. Гіпертекст означає, що текст містить посилання на інші тексти. Першу версію було опубліковано в 1991 році, вона містила лише 18 HTML тегів. З того часу, з кожною версіює додаються нові види тегів. Зараз їхня кількість становить близко 140. Правда деякі з них не підтримуються сучасними браузерами. Набільшим оновленням стандарту HTML було в 2014 році, коли вийшла версія 5. Вона додала до розмітки кілька нових семантичних тегів, які розкривають значення власного змісту, наприклад <article>, <header> та <footer>. Однією з найбільш очікуваних особливостей HTML 5 є вбудована підтримка аудіо та відео контенту. Тепер можна просто вставити теги <audio> </audio> та <video> </video> на сторінку.

Каскадна таблиця стилів, широко відома як CSS, - це шар стилю над елементами HTML, або, простіше кажучи, інструмент, що дозволяє стилізувати елементи (шрифт, розмір, колір та інтервали) ваших HTML сторінок та вмісту, обережно застосовуючи класи до нього. CSS допомагає визначити, як виглядатимуть сторінки та їх вміст, створений за допомогою HTML.

Спочатку CSS був випущений в 1996 році і складався з властивостей для додавання властивостей тексту, таких як шрифт та колір акценту тексту, фонів та інших елементів. CSS2 був випущений в 1998 році з доданими стилями для інших типів носіїв, щоб їх можна було використовувати для проектування макета сторінки. CSS3 був випущений в 1999 році, і в нього були додані нові властивості стилю. CSS3 запропонував кілька ключових міркувань щодо веб дизайну, таких як

округлі межі, які допомагають округляти межі без жодних клопотів. Це виявилося величезним плюсом для розробників, які боролися з початковими версіями CSS

JavaScript - це мова програмування, яка використовується для створення в основному веб сайтів. Якщо ви думаєте про основний склад веб сайту, у вас є HTML, який описує та визначає основний вміст і структуру веб сайту, тоді у вас є CSS, який повідомляє браузеру, як цей HTML вміст повинен відображатися - визначаючи такі речі, як колір і шрифт. Маючи лише HTML і CSS, у вас є веб сайт, який виглядає добре, але насправді не робить багато. JavaScript оживляє веб сайт, додаючи функціональність. JavaScript відповідає за елементи, з якими користувач може взаємодіяти, такі як випадаючі меню, модальні вікна та форми контактів. Він також використовується для створення таких речей, як анімація, відеоплеєри та інтерактивні карти.

Зараз JavaScript використовується не лише для обслуговування веб додатків. Розробка JavaScript призвела до винаходу Node.js на основі V8 у 2009 році. З цього моменту було можливо запустити JavaScript майже на всіх платформах, і спільнота розробників програмного забезпечення широко використовувала його. Таким чином JavaScript у багатьох випадках почав замінювати попередні серверні рішення, особливо для додатків у режимі реального часу, в яких дані завантажуються двостороннім способом і не потрібно повторно підключатися до сервера.

Зараз можна за допомогою JavaScript створювати мобільні додатки. Це пов’язано з розповсюдженням використання міжплатформенних додатків за допомогою React Native - фреймворку JavaScript, створеного Facebook, який дає можливість створювати мобільні рішення. Використання React Native робить розробку програмного забезпечення простішою, дешевшою та швидшою, оскільки JavaScript забезпечує адаптацію коду до різних платформ [4].

3.2 Опис Flux архітектури

Flux-архітектура - архітектурний підхід або набір шаблонів програмування для побудови призначеного для користувача інтерфейсу веб-додатків, що поєднується з реактивним програмуванням і побудований на односпрямованих потоках даних.

Згідно із задумом творців і незважаючи на те, що Facebook надав реалізацію Flux на додаток до React, Flux не є ще одним веб-фреймворком, а є архітектурним рішенням.

Основною відмінною рисою Flux є одностороння спрямованість передачі даних між компонентами Flux-архітектури. Архітектура накладає обмеження на потік даних, зокрема, виключаючи можливість поновлення стану компонентів самими собою. Такий підхід робить потік даних передбачуваним і дозволяє легше простежити причини можливих помилок в програмному забезпеченні. У мінімальному варіанті Flux-архітектура може містити три шари, які взаємодіють один по одному: Actions(дії), Stores(сховища), Views(види). Також між діями та сховищами виділяють Dispatcher(відправник). В першу чергу Flux працює з інформаційною архітектурою, яка потім відбивається в архітектурі програмного забезпечення, тому рівень уявлень слабо зачеплений з іншими рівнями системи. На рисунку 3.2.1 зображено схему Flux.

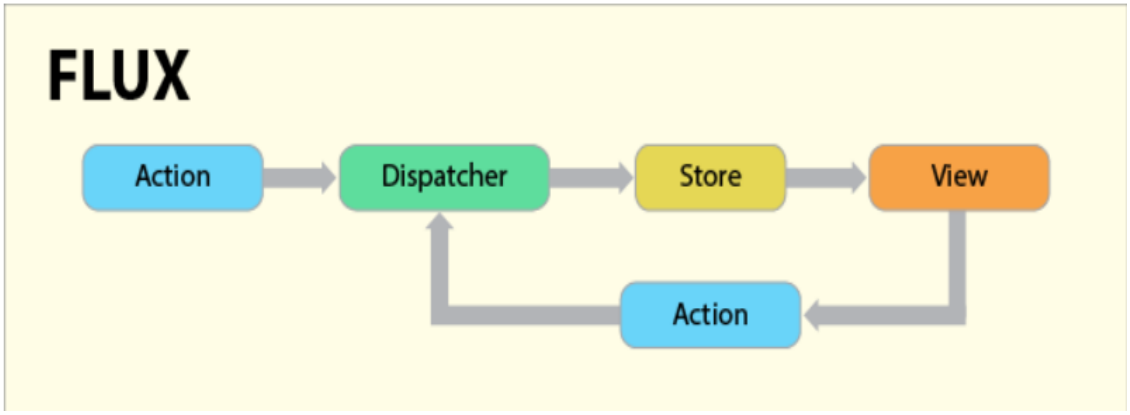

Рисунок 3.2.1 Архітектура Flux

Дії (англ. Actions) - вираз подій (часто для дій використовуються просто імена - рядки, що містять деяке дієслово). Диспетчери передають дії нижчого компонентам (сховищам) по одному. Нова дія не передається поки попередня

повністю не оброблена компонентами. Дії надходять асинхронно, але їх диспетчеризація є синхронним процесом. Крім імені, дії можуть мати корисне навантаження (англ. Payload), що містить пов'язані з дією дані.

Диспетчер (англ. Dispatcher) призначений для передачі дій до сховища. У спрощеному варіанті диспетчер може взагалі не виділятися, як єдиний на весь додаток. У диспетчері сховища реєструють свої функції зворотного виклику (callback) і залежності між сховищами.

Сховище (англ. Store) є місцем, де зосереджено стан (англ. State) додатки. Інші компоненти, згідно Flux, не мають значного (з точки зору архітектури) стану. Зміна стану сховища відбувається строго на основі даних дії і старого стану сховища.

Подання (англ. View) - компонент, звичайно відповідає за видачу інформації користувачеві. У Flux-архітектурі, яка може технічно не торкатися внутрішнього облаштування уявлень взагалі, це - кінцева точка потоків даних. Для інформаційної архітектури важливо тільки, що дані потрапляють в систему (тобто, назад в сховища) тільки через дії [5].

Основними особливостями Flux є:

− Синхронність: всі методи зворотного виклику, зареєстровані для кожної дії, синхронні у виконанні, саме ж дія може викликатися джерелом асинхронно.

− Інверсія управління: потік управління передається відповідному сховища і цільової функції зворотного виклику.

− Семантичні дії: дія, що викликається джерелом, містить смислове інформацію, що дозволяє відповідному сховища вибрати правильний метод виконання.

− Відсутність каскадів дій: всі методи зворотного виклику, зареєстровані для кожної дії, синхронні у виконанні, саме ж дія може викликатися джерелом асинхронно.

3.3 Опис React, Redux

React - відкрита JavaScript бібліотека для створення інтерфейсів користувача, яка покликана вирішувати проблеми часткового оновлення вмісту веб сторінки, з якими стикаються в розробці односторінкових застосунків [6].

Розробляється Facebook, Instagram і спільнотою індивідуальних розробників. React дозволяє розробникам створювати великі веб-застосунки, які використовують дані, котрі змінюються з часом, без перезавантаження сторінки. Його мета полягає в тому, щоб бути швидким, простим, масштабованим.

React обробляє тільки користувацький інтерфейс у застосунках. Це відповідає видові у шаблоні модель-вид-контролер (MVC), і може бути використане у поєднанні з іншими JavaScript бібліотеками таких як AngularJS. Як бібліотеку інтерфейсу користувача React найчастіше використовують разом з іншими бібліотеками, такими як Redux.

У даний час React використовують Netflix, Yahoo, Airbnb, Sony, Atlassian та інші [7].

Особливостями React є:

− Одностороння передача даних: властивості передаються в рендер компоненту, як властивості html тегу. Компонент не може напряму змінювати властивості, що йому передані, але може їх змінювати через callback функції. Такий механізм називають «властивості донизу, події нагору».

− Віртуальний DOM: React підтримує віртуальний DOM, а не покладається виключно на DOM браузера. Це дозволяє бібліотеці визначити, які частини DOM змінилися, порівняно (diff) зі збереженою версією віртуального DOM, і таким чином визначити, як найефективніше оновити DOM браузера. Таким чином програміст працює зі сторінкою, вважаючи що вона оновлюється вся, але бібліотека самостійно вирішує які компоненти сторінки треба оновити. З цим принципом можна познайомитися на рисунку 3.3.1

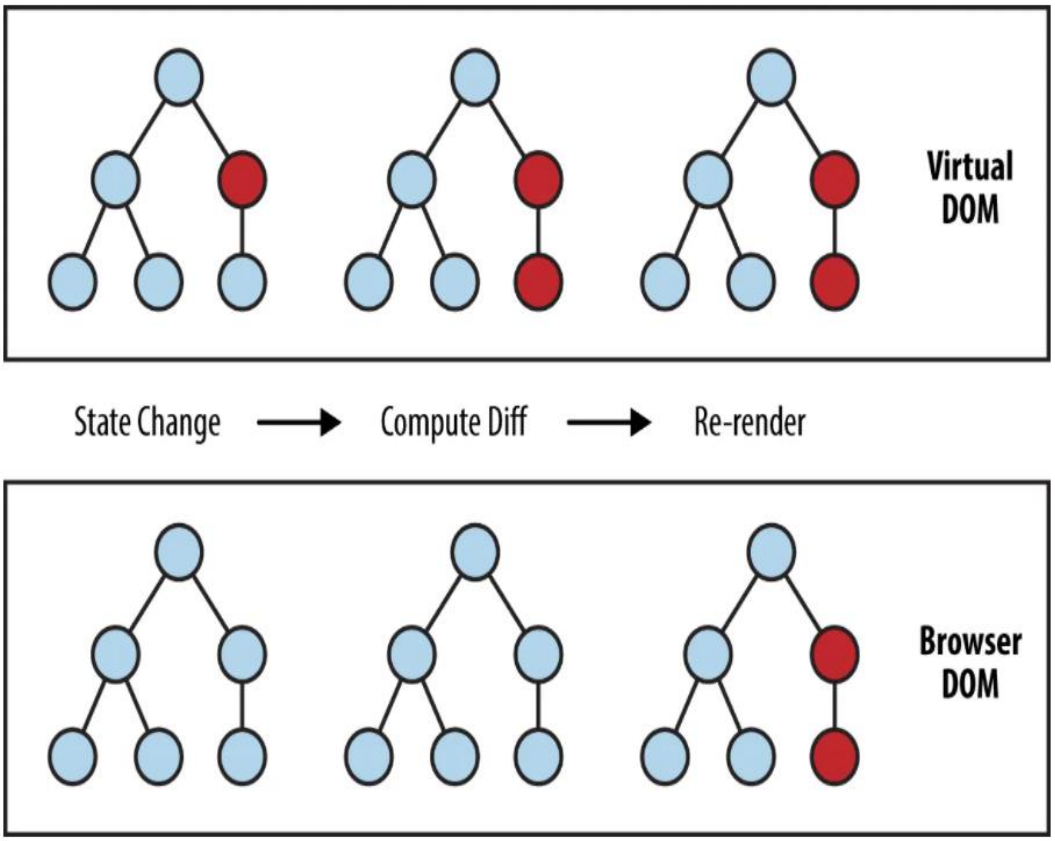

Рисунок 3.3.1 Принцип роботи Virtual DOM

– JSX: Компоненти React зазвичай написані на JSX. JSX – це розширення синтаскису JavaScript, що використовується в React, що дозволяє писати JavaScript код, схожий на HTML. Це свого роду шаблонна мова. Код написаний на JSX компілюється у виклики методів бібліотеки React. Розробники можуть так само писати на чистому JavaScript. JSX нагадує іншу мову, яку створили у компанії Фейсбук для розширення PHP, XHP. На рисунку 3.2.2 можна побачити код написаний на JSX та код, який перетворюється у виклики методів бібліотеки React.

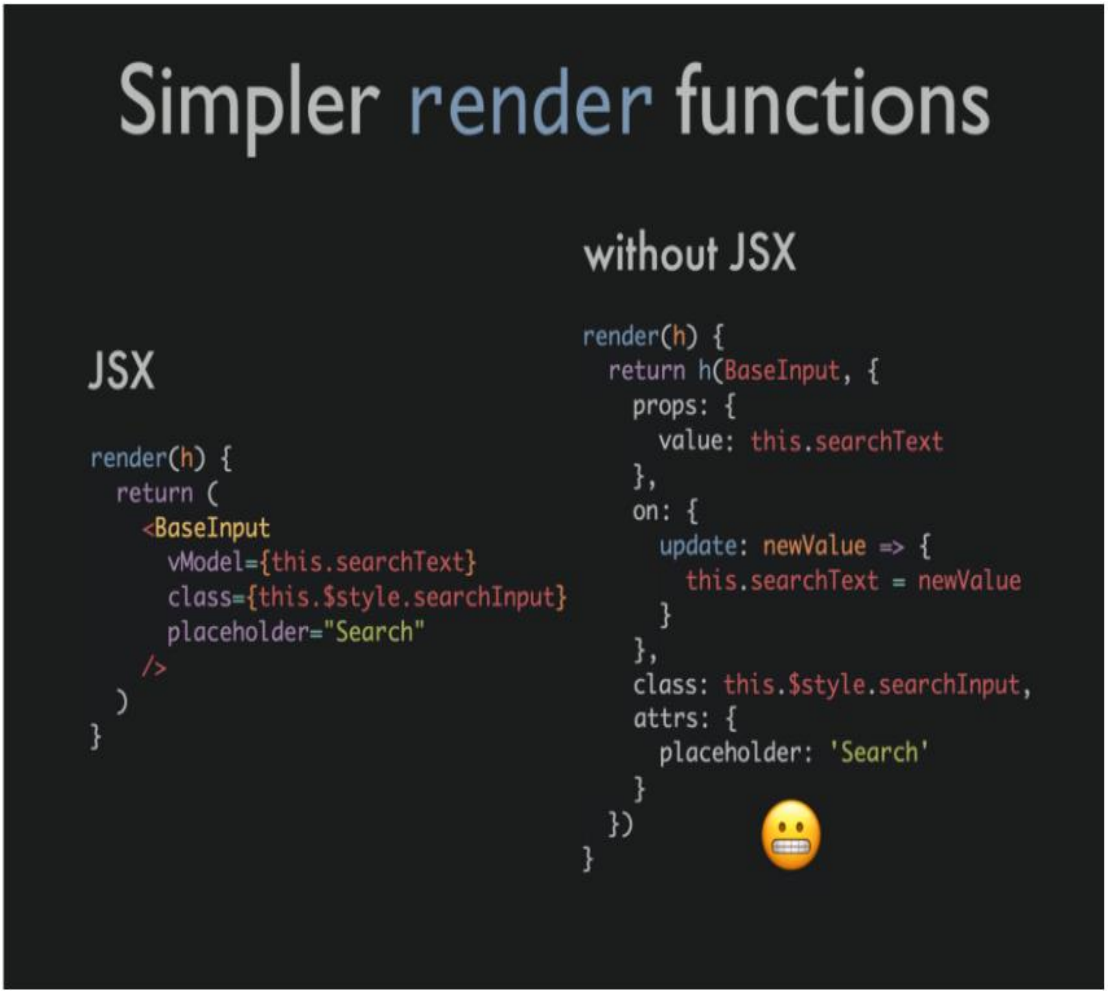

Рисунок 3.3.2 JSX розмітка

– Методи життєвого циклу: різні методи, які відбуваються за допомогою ReactJS. Вони дозволяють розробнику обробляти дані в різних точках життєвого циклу програми React. Наприклад:

- ShouldComponentUpdate - це метод життєвого циклу, який каже Javascript оновити компонент, використовуючи логічні змінні.
- ComponentWillMount - це метод життєвого циклу, який каже Javascript налаштувати певні дані перед монтуванням компонентів (вставлення у віртуальний DOM).
- ComponentDidMount - це метод життєвого циклу, подібний до компонента WillMount, за винятком того, що він працює після методу render, і може використовуватися для додавання JSON-даних, а також для визначення властивостей та станів.

- Render - є найважливішим методом життєвого циклу, необхідним у будь-якому компоненті. Метод render - це те, що з'єднується з JSX і відображати власний JSX.

Разом з React для розробки веб додатків використовується бібліотека Redux [8]. Redux – це бібліотека керування станом програми для JavaScript застосунків. Її також можна використовувати з AngularJS, другим по популярності фреймворком для розробки клієнтської частини. Ця бібліотека була створена у 2015 році, в її основу лягла архітектура Flux. Реалізація Redux містить деякі відмінності відносно архітектури Flux.

У Redux було додано нову структуру, яка була відсутня в оригінальній архітектурі. Ця структура має назву регулятор (reducer). Це такі собі функції, які мають доступ до стану програми. При викоику даної функції можна зробити зміни в нашому сховищі даних, тобто додати нові записи, змінити існуючі або видалити непотрібні. Авторами бібліотеки не рекомендовано робити сторонні операції в тілі функцій регуляторів, бо це може привести до некоректної роботи Redux. Такими операціями є виклик нечистих функцій таких як Math.random(), мутування даних або звернення до сторонього сервісу за допомогою API. Регулятори зазвичай описуються в окремих файлах, які потім компонуються разом в один великий регулятор та підключаються уже в функції створення сховища даних програми.

Сховище даних Redux являє собою простий JavaScript об’єкт. Це єдине джерело, з якого можна брати поточну інформацію програми. Наприклад, за його допомогою можна легко отримати дані про тему та стиль додатка, який використовується користувачем. Якщо користувач змінить кольорову гаму ми матимемо змогу знати поточну тему веб додатка, і на основі цього показувати відповідний дизайн. Сховище даних являє собою дерево, у вузлах якого міститься необхідна інформація для роботи з веб додатком. Це може бути наприклад інформація про контакти: ім’я, прізвище, мобільний телефон, поштова адреса.

У Redux дії (actions) представлені у вигляді об’єктів. Для цих об’єктів обов’язково має бути поле з ключем type. Зазвичай значенням цього поля є рядок виду FETCH_CONTACTS_START, який дає зрозуміти, для чого використовується дана дія. Для того, щоб зробити необхідні зміни у веб додаток, ми спочатку робимо запит до сервера або відслідковуємо дії юзера. Після отримані дані ми передаємо через поле payload далі до регуляторів. Payload може отримати будь-який тип даних – від звичайного рядка до масива з сотні елементів. З цими даними буде розбиратися регулятор.

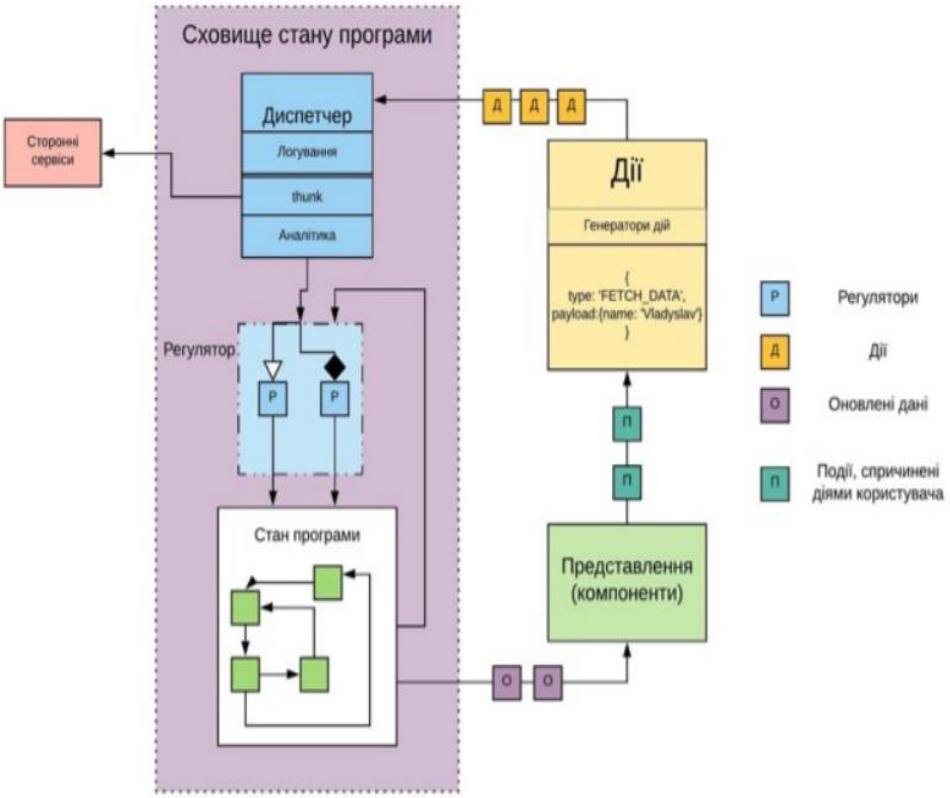

Рисунок 3.3.3 Схема роботи Redux

На рисунку 3.3.3 видно як реалізовано формування потоку даних бібліотекою Redux. Проводячи паралелі з реалізацією архітектури Flux, можна замітити деякі розбіжності в реалізації. Диспетчер отримує події, які були приведені у дію в результаті взаємодії юзера з додатком. Диспетчер може мати різні функції, які будуть надавати додаткові можливості. Наприклад, redux-thunk фукнція дозволяє роботу асинхронних функцій при диспетчеризації. Деякі утиліти надають можливість відслідковувати помилки при роботі з сервером. З їх допомогою до

диспетчера додається окремий рядок коду, який при виникненні помилки передасть дані про неї на відповідний сервіс.

Якщо не виникло ніяких помилок, то дані передаються до регуляторів шляхом виклику функції dispatch. Цей виклик містить type, по якому буде встановлено, який саме регулятор буде виконувати зміни до програми. Коли було знайдено відповідний регулятор, відбувається зміна стану програми. Стан програми змінюється, застарілі дані стають не актуальними та замінюються на нові, при цьому нові дані синхронізуються із загальним станом програми та відображаються кінцевому користувачеві.

Дані синхронізуються за допомогою так званих селекторів (selectors). Це ще одна розбіжність Redux з архітектурою Flux. Використання селекторів надає можливість отримання потрібних нам даних прямо зі сховища.

3.4 Опис React UI Framework Material UI

Material UI – один з найпопулярніших UI фреймворків для роботи з React [9].

В основу фреймворка ліг Material design – тип графічного дизайну інтерфейсів програмного забезпечення та додатків, розроблений компанією Google. Вперше був представлений 25 червня 2014 року на конференції Google. Стиль розширює ідею «карточки», що з'явилася в Google Now, більш широким застосуванням строгих макетів, анімацій та переходів, відступів та ефектів глубин (світла та тіні). За ідеєю графічних дизайнерів Google, в веб додатків не повинно бути гострих кутів, карточки повинні переключатися між собою плавно і практично незалежно.

Material UI має широкий спектр можливостей в який входить можливість перероблювати зовнішній вигляд компонентів, встановлення тем та легкий синтаксис(рис 3.4).

```
import React from "react";
import ReactDOM from "react-dom";
import Button from "@material-ui/core/Button";

function App() {
  return (
    <React.Fragment>
      <Button variant="outlined" color="secondary">
        Button 1
      </Button>
      <Button variant="contained" color="primary">
        Button 2
      </Button>
      <Button variant="contained" color="default" size="small">
        Button 2
      </Button>
    </React.Fragment>
  );
}

ReactDOM.render(<App />, document.querySelector("#app"));
```

Рисунок 3.4 Різні варіанти вигляду компоненти “кнопка”

3.5 Опис Webpack, Babel, ESLint

Webpack – це інструмент, що дозволяє збирати окремі модулі в бандли. При великій кількості файлів та папок, Webpack генерує один або декілька файлів з уже оптимізованим кодом, який матиме все потрібне для запуску веб додатку [10]. Суть роботи можна зрозуміти поглянувши на рисунок 3.5

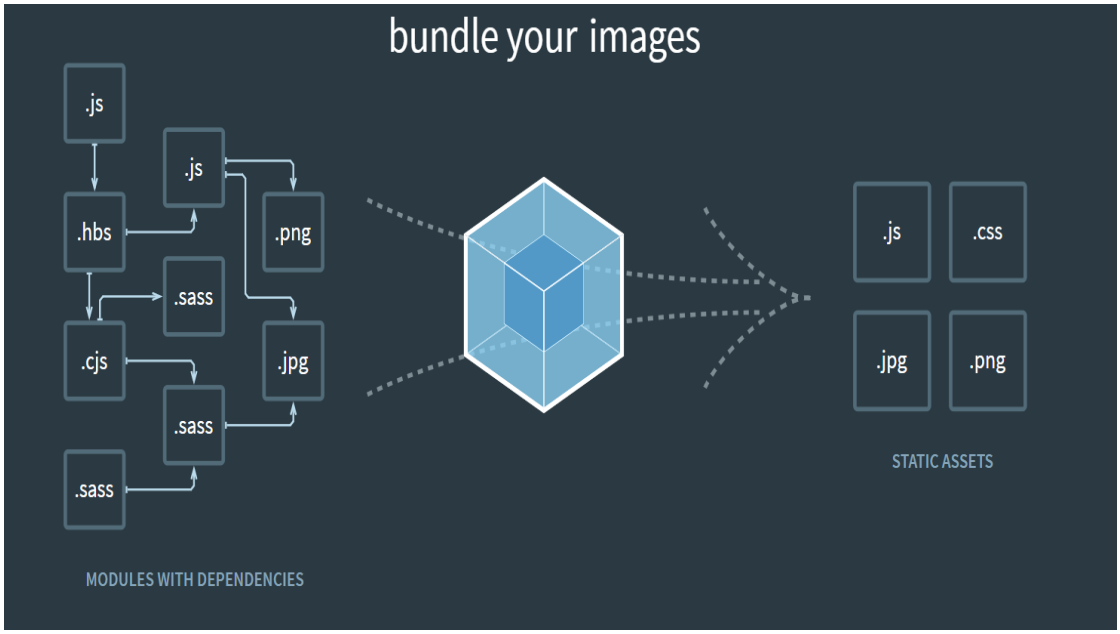

Рисунок 3.5 Приклад використання Webpack

Процес групування починається з визначених користувачем точок входу. Ці точки самі по собі є модулями і можуть вказувати на інші модулі за допомогою імпортів.

Почнемо наприклад з базової сторінки. Ми маємо один HTML файл, один або декілька JavaScript файлів, деякі файли зі стилями та файли картинки. Коли ви говорите Webpack зібрати проект, він дивиться на всі файли у вашій директорії. Він обстежує та визначає, які з цих файлів залежать один від одного. Webpack будує граф залежностей, за яким визначає як повинен бути зібраним проект.

Коли граф побудований, будується дерево, в основі якого повинен лежати корінь – модуль, у випадку Webpack. Це може бути HTML сторінка або інший файл. За замовчуванням будемо вважати що це файл написаний мовою JavaScript та має назву index.js. Від цього кореня Webpack рекурсивно переглядає кожен файл у дереві. Після того як Webpack зібрав всі наші файли, вони по замовчуванню поміщуються в папку dist. Для того щоб зібрати всі файли ефективно та оптимізовано, Webpack використовує плагіни. Деякі з них вбудовані, деякі можуть бути додані програмістом. Наприклад CopyWebpackPlugin, яких може скопіювати окремі файли та окремі директорії в створену папку dist.

Babel - це безкоштовний транскомпайлер JavaScript із відкритим кодом, який в основному використовується для перетворення коду ECMAScript 2015+ (ES6 +) у зворотну сумісну версію JavaScript, яку можуть запускати старіші механізми JavaScript. Babel - це популярний інструмент для використання новітніх функцій мови програмування JavaScript [11].

Розробники можуть використовувати нові мовні функції JavaScript, використовуючи Babel для перетворення їх вихідного коду у версії JavaScript, які браузери, що розвиваються, можуть обробляти. Основна версія Babel завантажувалася 5 мільйонів разів на місяць станом на 2016 рік, збільшуючись до 16 мільйонів разів на тиждень станом на 2019 рік.

Плагіни Babel використовуються для перетворення синтаксису, який не підтримується широко, у версію, сумісну із зворотною стороною. Наприклад, функції зі стрілками, зазначені в ES6, перетворюються у звичайні оголошення функцій. Нестандартний синтаксис JavaScript, такий як JSX, також може бути перетворений [12].

ESLint - це інструмент статичного аналізу коду для виявлення проблемних шаблонів, знайдених у коді JavaScript. Він був створений Ніколасом Закасом у 2013 році. Правила в ESLint можна налаштувати, а налаштовані правила можна визначити та завантажити. ESLint охоплює як якість коду, так і питання стилю кодування. ESLint підтримує поточні стандарти ECMAScript та експериментальний синтаксис із чернеток для майбутніх стандартів. Код, що використовує JSX або TypeScript, також може бути оброблений, коли використовується плагін або перекладач [13].

РОЗДІЛ 4

ПРАКТИЧНА ЧАСТИНА РОЗРОБКИ ВЕБ ДОДАТКА

4.1. Реалізація веб додатка

В теоретичній частині були представлені технології реалізації веб додатків, основною метою використання яких являється зручність, значні можливості, що продовжують свій розвиток, а головне економія часу та ресурсів. Для реалізації клієнтської частини веб додатка використовувавалися саме ці технології.

Фреймворк React було обрано через те, що він є найпопулярнішим фреймворком для розробки клієнтської частини. Він має широку підтримку серед програмістів.

Для середовища програмування було вибрано Visual Studio Code – редактор коду, що позиціонує себе як легкий редактор коду для кросплатформенної розробки веб та хмарних додатків. Він має широку кількість плагінів, що допомагають збільшити якість та швидкість написання коду. Редактор є безплатним.

Структура проекта являє собою директорію, що містить папку з допоміжними модулями та головну папку з трьома пакетами: App, Data, UI (див. рис. 4.1.1).

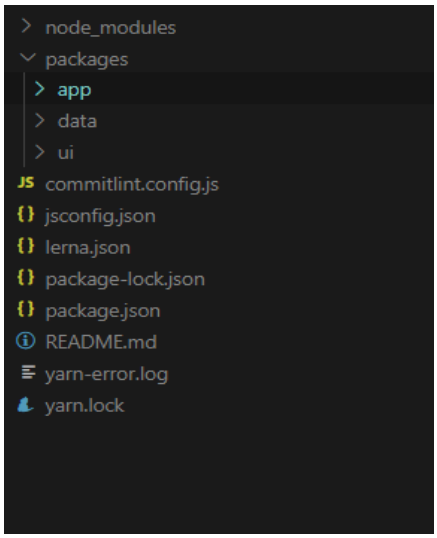

Рисунок 4.1.1 Структура проекта

Розпочнемо з пакета UI. В його основі лежать компоненти – майбутні частини, з яких буде складено візуальна частина додатка. Кнопки, форми, селекти, сайдбари, таблиці та інші зроблені в цьому пакеті. Базові компоненти імпортуються з бібліотеки Material UI, а вже потім кастомізуються. Покажемо як виглядає компонент кнопка. Спочатку компонент Button імпортується з бібліотеки Material UI. Вже оброблена кнопка виглядає як функція в JavaScript, яка приймає деякі параметри та повертає JSX. Через ці параметри можна буде передавати різні стилі та інші корисні властивості(props), які зможуть наприклад показувати іконку всередині кнопки. Також ці props повинні бути типізованими, з цим справляється React PropTypes, що являє собою ряд валідаторів, які використовуються для перевірки, що отримані дані коректні. Це зручно при розробці, бо при некоректному значені в консоль розробника буде показано помилку. На рисунку 4.1.2 наведено код компоненти кнопки.

```
import Button from '@material-ui/core/Button';
import { useButtonStyles, useFabStyles, useIconStyles } from './styles';
import { icons } from '../icons/icons';

const buttonPropTypes = {
  className: PropTypes.string,
  children: PropTypes.node.isRequired,
  size: PropTypes.string,
  disabled: PropTypes.bool,
  color: PropTypes.string,
  toolTip: PropTypes.string,
  icon: PropTypes.string,
};

export const CustomButton = ({ className, children, color, icon, disabled, toolTip, ...props }) => {
  const s = useButtonStyles(props);
  return (
    <>
      <Button className={cx(s.btn, s[color], disabled ? s.disabled : '', className)} {...props}>
        {icon && <div className={s.icon}>{icons[icon]}</div>}
        {children}
      </Button>
    </>
  );
};

CustomButton.defaultProps = {
  className: null,
  size: 'small',
  color: 'green',
  disabled: false,
  toolTip: null,
  icon: null,
};

CustomButton.propTypes = buttonPropTypes;
```

Рисунок 4.1.2 Код стилізованої кнопки в UI

Для стилізування наших компонент використовується використовується паттерн CSS-In-JS. Це паттерн, при якому стилі створюються за допомогою JavaScript, замість того, щоб писати їх в окремому файлі стилів. На рисунку 4.1.3 показано як створюються стилі, які потім використовуються в компонентах.

```
export const useButtonStyles = makeStyles(theme => ({
  root: {
    height: 40,
    width: 40,
    boxShadow: 'none',
    '& svg': {
      width: 20,
      height: 20,
      color: 'white',
    },
    '&:hover': {
      backgroundColor: theme.palette.secondary.dark,
    },
    '&:active': {
      color: 'rgba(255, 255, 255, 0.3)',
    },
  },
  [buttonColors.green]: {
    backgroundColor: theme.palette.secondary.main,
  },
}));
```

Рисунок 4.1.3 Використання стилів

Як виглядає фінальна версія показано на рисунку 4.1.4.

Рисунок 4.1.4 Вигляд кнопок в додатку

Продовжимо з пакетом Data. Це модуль, в якому відбуваються всі маніпуляції додатка з даними. Це і дані отримані за допомогою API, і дані стану нашого додатку. Основою цього модуля є імплементація бібліотеки Redux. Покажемо, як саме це це зроблено. На рисунку 4.1.5 показано структуру пакета Data.

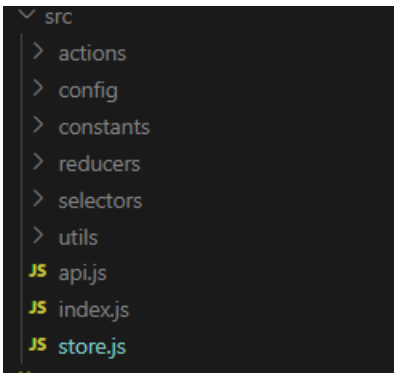

Рисунок 4.1.5 Структура Data

В папці Actions містяться події, що є частиною архітектури Redux. Саме в подіях відбуваються виклики до сторонніх сервісів, що повертають дані, які потім через регуляторів (reducers) обробляються та попадають в сховище (store) – єдине джерело даних в Redux. Через те що дані в сховищі можуть змінюватися лише через події, ми завжди можемо знати, що наш додаток має правильний стан даних. Потім ці дані можна легко дістати зі сховища за допомогою селекторів (selectors), які будуть відображатися вже наших компонентах(view). На рисунку 4.1.6 показано приклад події, в якій ми отримуємо список груп від API сервісу і за допомогою диспетчера (функції dispatch) передаємо отримані дані до регулятора.

```
export const fetchGroups = () => async (dispatch, getState, { api }) => {
  dispatch({ type: constants.FETCH_GROUPS_START });
  try {
    const res = await api.post(
      '/select/Group',
      {
        filters: {
          expression: {},
          logicalOperator: 0,
          filters: [],
        },
        columns: {
          columnNames: ['Id', 'Name', 'IsArchived'],
          relatedColumns: [],
        },
        pageSize: 100,
        pageIndex: 0,
      },
      { headers: { accept: 'text/plain', 'Content-Type': 'application/json-patch+json' } },
    );
    if (res.status % 200 >= 100) throw new Error(res.statusText);
    dispatch({
      type: constants.FETCH_GROUPS_SUCCESS,
      payload: {
        items: res.data,
      },
    });
    return true;
  } catch (e) {
    console.error(e);
    dispatch({ type: constants.FETCH_GROUPS_FAILED });
    return false;
  }
};
```

Рисунок 4.1.6 Приклад події в Redux

На рисунку 4.1.7 показано робота регулятора. При виконанні функції диспетчера відправлені дані потрапляють до регулятора та записують в сховище за допомогою команди newState.setIn().

```
import { fromJS } from 'immutable';
import * as constants from 'constants/groups';
import LoadingProgress from 'utils/reducers/loading';

export const groupsProgress = new LoadingProgress('groupsProgress');

const loadGroups = (state, action) =>
  state.withMutations(newState => {
    const { items } = action.payload;
    newState.setIn(['entities'], fromJS(items));
    groupsProgress.setLoaded(newState);
  });

const initialState = fromJS({
  entities: {},
});

export default (state = initialState, action) => {
  switch (action.type) {
    case constants.FETCH_GROUPS_START:
      return groupsProgress.setLoading(state);
    case constants.FETCH_GROUPS_SUCCESS:
      return loadGroups(state, action);
    case constants.FETCH_GROUPS_FAILED:
      return groupsProgress.setLoadFailed(state);
    case constants.CLEAR_STATE:
      return initialState;
    default:
      return state;
  }
};
```

Рисунок 4.1.7 Приклад регулятора в Redux

На рисунку 4.1.8 показано як можна дістати дані з нашого сховища.

```
import { groupsProgress } from 'reducers/groups';

const getState = store => store.groups;

export const isLoaded = store => groupsProgress.getLoaded(getState(store));
export const isLoading = store => groupsProgress.getLoading(getState(store));

export const getGroups = store => {
  return getState(store).get('entities');
};
```

Рисунок 4.1.8 Приклад селектора в Redux

Тепер перейдемо до самого ядра додатка. В основу покладено розмежування на роботу з даними та рендер розмітки. Контейнери (containers) містить файли, які надають дані та описують поведінку компонентів (components)(див. рис. 4.1.9).

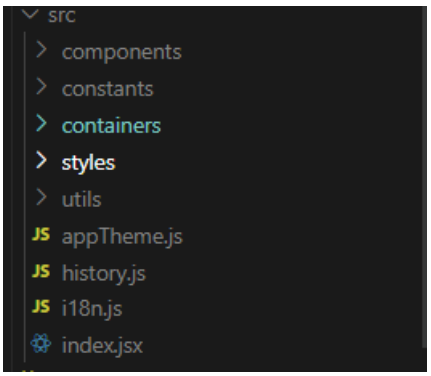

Рисунок 4.1.9 Організація файлів в App

В контейнерах відбувається взаємодія з бібліотекою Redux за допомогою функцій mapStateToProps, mapDispatchToProps та connect. Функція connect з'єднує контейнер з Redux, та надає доступ mapStateToProps до селекторів, а mapDispatchToProps - до подій. На рисунку 4.1.10 показано як використовуються ці функції.

```
const mapStateToProps = state => ({
  isLoaded: selectors.contacts.isLoaded(state),
  items: selectors.contacts.getContacts(state),
  isRemoving: selectors.contacts.isRemoving(state),
});

const mapDispatchToProps = {
  fetchContacts: actions.contacts.fetchContacts,
  deleteContact: actions.contacts.deleteContact,
  setSuccessMessage: actions.api.setSuccessMessage,
};

export default connect(mapStateToProps, mapDispatchToProps)(ContactsContainer);
```

Рисунок 4.1.10 Функції mapStateToProps, mapDispatchToProps, connect

За допомогою функції render можна відрендерити компоненти з потрібними для них параметрами. На рисунку 4.1.11 показано як виглядає звичайний функціональний компонент. Цей компонент може бути у вигляді функції або класу написаного на мові JavaScript разом з JSX. Відповідно компоненти, оголошені за допомогою JavaScript функції прийняли називати функціональними компонентами, а компоненти, оголошені за допомогою JavaScript класів, прийнято називати класовими компонентами.

```jsx
export const Contacts = ({ toggleModalDelete, ...props }) => {
  const classes = useCardsStyles();

  const tableActions = [
    {
      icon: 'delete',
      onClick: item => toggleModalDelete(item.get('id')),
    },
  ];

  const tableColumns = [
    { id: 'firstName', label: 'First name', accessor: 'firstName' },
    { id: 'lastName', label: 'Last name', accessor: 'lastName' },
    { id: 'email', label: 'Email', accessor: 'email' },
    { id: 'phoneNumber', label: 'Phone', accessor: 'phoneNumber' },
  ];
  return (
    <div className={classes.page}>
      <div className={classes.root}>
        <div className={classes.headerText}>
          <Text color="blueDark" size={24}>
            Contacts
          </Text>
          <Link to={addContactPath}>
            <Button className={cx(classes.button, classes.lastButton)}>Create contact</Button>
          </Link>
        </div>
        <Table
          linkCreator={item => contactPath(item.get('id'))}
          columns={tableColumns}
          actions={tableActions}
          fallBack="No contacts available"
          showEditIcon
          {...props}
        />
      </div>
    </div>
  );
};

Contacts.propTypes = cardsPropTypes;
```


Рисунок 4.1.11 Функціональний компонент

Тепер, коли ми маємо всі три пакети, ми можемо запустити наш додаток. Це відбувається за допомогою декількох команд, які в свою чергу запускають налаштовані скрипти для збору наших пакетів. Кінцевий результат нашої роботи можна побачити на рисунку 4.1.12

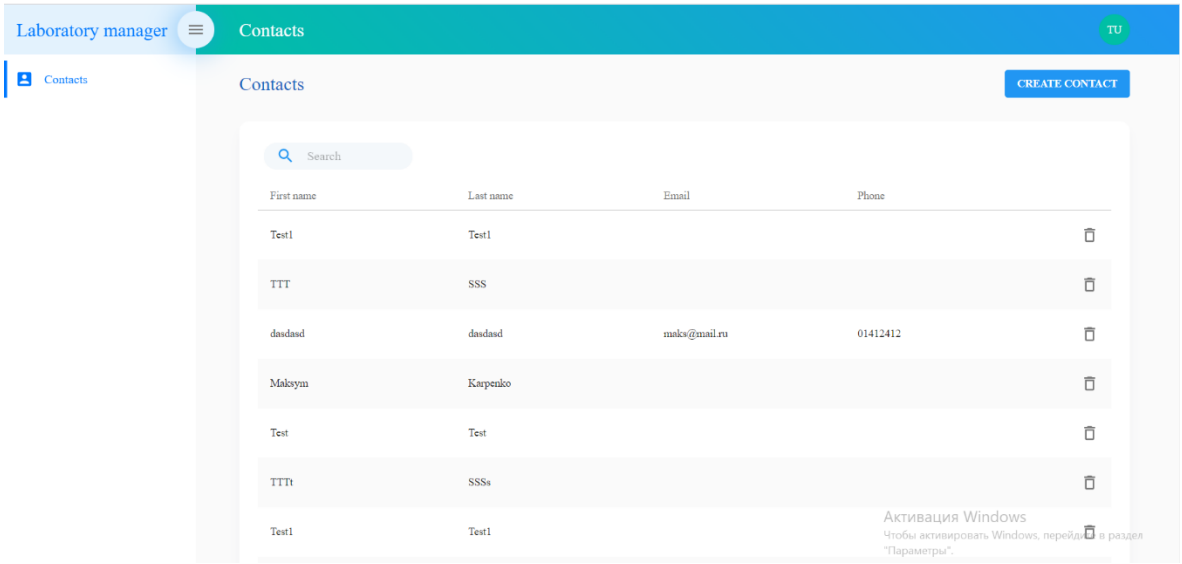

Рисунок 4.1.12 Веб додаток у дії

ВИСНОВКИ

В ході виконання кваліфікаційної роботи було досліджено існуючі на ринку веб додатки для навчання онлайн, визначено їхні плюси та недоліки, досліджено популярні технології розробки Frontend частини веб додатків, розроблено загальну архітектуру клієнтської частини сервісу та імплементовано в робочий веб додаток.

Обраними технологіями стали React, Redux, Material UI Framework, Webpack, Babel. Для розробки програмного забезпечення використовувався текстовий редактор із широким спектром можливостей Visual Studio Code.

За останні роки веб додатки набрали великої популярності серед розробників. Можливість розміщувати бізнес логіку додатку на віддаленому сервері виглядає цікавою. За допомогою наведених вище технологій можна легко втілити будь-яку ідею у вигляді веб додатка.

Під час виконання роботи було поглиблено знання фреймворку React, архітектури Flux та її імплементації у вигляді бібліотеки Redux. Також було налаштовано текстовий редактор Visual Studio Code таким чином, що є можливість автозаповнення коду, автоімпорту функцій та класів, що пришвидшує розробку програмного забезпечення.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Online Tools for Teaching and Learning [Електронний ресурс] – Режим доступу до ресурсу: https://blogs.umass.edu/onlinetools/community-centered-tools/google-classroom/
2. Система электронного обучения и тестирования Moodle [Електронний ресурс] – Режим доступу до ресурсу: https://www.ispring.ru/elearning-insights/moodle
3. Что такое фронтенд [Електронний ресурс] – Режим доступу до ресурсу: https://dan-it.com.ua/razrabotka-so-storony-front-end-chto-jeto-takoe-i-chem-otlichaetsja-ot-back-end/
4. Lindley C. Front-end Developer Handbook 2019 / C. Lindley. [Електронний ресурс] – Режим доступу до ресурсу: https://frontendmasters.com/books/front-end-handbook/2019/
5. Flux In-Depth Overview [Електронний ресурс] – Режим доступу до ресурсу: https://facebook.github.io/flux/docs/in-depth-overview/#:~:text=Flux%20is%20the%20application%20architecture,a%20lot%20of%20new%20code.
6. React [Електронний ресурс] – Режим доступу до ресурсу: https://uk.reactjs.org/
7. Learning React: Functional Web Development with React and Redux / A. Banks, E. Porcello – O'Reilly Media, 2017. – 180 с.
8. Redux [Електронний ресурс] – Режим доступу до ресурсу: https://redux.js.org/
9. Material UI [Електронний ресурс] – Режим доступу до ресурсу: https://material-ui.com/ru/
10. Webpack [Електронний ресурс] – Режим доступу до ресурсу: https://webpack.js.org/
11. Modern Full-Stack Development / F. Zammetti – Apress Media, 2020. – 143 с.
12. Learning React: A Hands-On Guide to Building Web Applications Using React and Redux / K. Chinnathambi – Pearson Education, 2017. – 182 с.
13. ESLint [Електронний ресурс] – Режим доступу до ресурсу: https://eslint.org/