Міністерство освіти і науки України
Харківський національний університет радіоелектроніки
(повне найменування вищого навчального закладу)

Факультет *Інфокомунікацій*
(повне найменування інституту, факультету (відділення))

Кафедра *Інформаційно-мережної інженерії*
(повна назва кафедри (предметної, циклової комісії))

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

другий (магістерський)
(рівень вищої освіти)

Порівняльний аналіз методів створення веб-додатків і розробка сайту дистанційного навчання
(тема)

Виконав: студент 2 курсу групи *ІМІм-20-1*
Спеціальність *172 Телекомунікації та радіотехніка*
(код і повна назва спеціальності)
Тип програми *освітньо-професійна*
(освітньо-професійна або освітньо-наукова)
Освітня програма *Інформаційно-мережна інженерія*
(повна назва освітньої програми)
Гриценко В.А.
(прізвище та ініціали)
Керівник *ст. викл. Калюжний М.М.*
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри _____ *Безрук В.М.*
(підпис) (прізвище, ініціали)

2021 р.

Не містить відомостей, заборонених
до відкритого публікування

Керівник _____ / *Калюжний М.М.*

Студент _____ / *Гриценко В.А.*

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

(повне найменування вищого навчального закладу)

Факультет _ *Інфокомунікацій*

Кафедра *Інформаційно-мережної інженерії*

Рівень вищої освіти *другий (магістерський)*

Спеціальність *172. Телекомунікації та радіотехніка*

(код і повна назва)

Тип програми *освітньо-професійна*

(освітньо-професійна або освітньо-наукова)

Освітня програма *Інформаційно-мережна інженерія*

(повна назва)

ЗАТВЕРДЖУЮ

Зав. кафедри «ІМІ»

_____ проф. Безрук В.М.

(підпис)

«*08*» *листопада* 2021 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові *Гриценко Владислав Анатольевич*

(прізвище, ім'я, по батькові)

1. Тема роботи *Порівняльний аналіз методів створення веб-додатків і розробка сайту дистанційного навчання*

керівник роботи *к.т.н., ст. викл. Калюжний М.М*

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом ВНЗ від «08» *листопада* 2021 р. № *1674 Ст*

2. Термін подання студентом роботи до екзаменаційної комісії *19.12.* 2021 р.

3. Вихідні дані до роботи *Порівняльний аналіз методів створення веб-додатків і розробка сайту дистанційного навчання*

4. Перелік запитань, що необхідно опрацювати в роботі

Вступ

1 Аналіз методів розробки веб-додатків.

2 Створення візуальної структури додатку для уникнення складнощів при подальшій розробці.

3 Розробка структури веб-додатку та сумижних сервісів за допомогою обраних інструментів.

Висновки

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Аналіз завдання та літературних джерел	08.11.21-10.11.21	
2	Аналіз методів розробки веб-додатків	11.11.21-14.11.21	
3	Дослідження засобів створення веб-додатків для дистанційного навчання	15.11.21-17.11.21	
4	Розробка моделі сайту та структури баз даних	18.11.21-30.11.21	
5	Створення функціонального зразка веб-додатку дистанційного навчання	01.12.21-05.12.21	
6	Аналіз та тестування функціональної частіни веб-додатку.	06.12.21-07.12.21	
7	Висновки	08.12.21-14.12.21	
8	Оформлення пояснювальної записки	15.12.21-19.12.21	

Дата видачі завдання *08 листопада* 2021 р.

Студент _____ Гриценко В.А.

(підпис) (прізвище та ініціали)

Керівник роботи _____ ст. викл. Калюжний М.М

(підпис) (посада, прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка: с., рис., 2 посилань, 1 додаток.

Мета роботи – порівняльний аналіз методів створення веб-додатків і розробка сайту дистанційного навчання.

Об'єкт розробки – сайт компанії що надає телекомунікаційні послуги.

Мета роботи - розробка сайту, що надає послуги доступу до мережі Інтернет, забезпечення коректної роботи його функціональних можливостей та повної конфіденційність персональних даних залишених користувачами сайту.

Користувачам має бути надана можливість в будь-який зручний час вільно отримувати доступ до інформації стосовно працевлаштування та тарифів на послуги. Також налаштована цілодобова гаряча лінія для зручного вирішення будь яких питань та зручна форма для відправки запитів на зворотній дзвінок.

На підставі сформульованих вимог розроблена структура сайту, розроблений дизайн і підготовлений сайт у електронному вигляді.

ІНТЕРНЕТ, JAVA SCRIPT, HTML, CSS, КОРИСТУВАЧ, БРАУЗЕР, GRAPHQL, ТЕСТУВАННЯ, MONGODB, ВЕБ-ДОДАТОК

ABSTRACT

Explanatory note: pp., fig., 13 reference, 1 app.

Object of work – comparative analysis of web application development methods and distance learning site development.

The Internet of Things concept is increasingly determining the development of communication networks, both now and in the future. The widespread application of the Internet of Things concept is wireless sensor networks (WSN). Due to the potentially widespread use of WSN in all areas of human life, they are still named as ubiquitous sensor networks. WSN belongs to the class of self-organizing systems, for which the principles of construction, routing protocols, quality of service parameters, models and characteristics of traffic etc. are significantly changed in comparison with traditional infrastructure networks.

In the presented work, the characteristics of traffic in wireless sensor networks were investigated in the context of the specifics of their work.

INTERNET, JAVA SCRIPT, HTML, CSS, USER, BROWSER, GRAPHQL, TESTING, MONGODB, WEB-APPLICATION

ЗМІСТ

С.

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

HOC	Higher-Order Component	Компонент найвищого порядку
REST	REpresentational State Transfer	Передача стану уявлення
SPA	Single page application	Односторінковий додаток
PWA	Progressive web app	Прогресивний web-додаток
MPA	Multi page app	Багатосторінковий додаток
CSS	Cascading Style Sheets	Каскадні таблиці стилів

ВСТУП

Розробка робочого коду та аналіз різних технологій й вибір серед них кращого варіанту для сайту такого типу.

Оскільки існує багато технологій та методів створення веб-додатків потрібно розібратися котрі з них обрати.

В дипломній роботі розглянуто основні способи створення веб-сторінок, перелік знань, що необхідно знати, та уміння застосовувати різне програмне забезпечення (ПЗ) необхідне для створення сайтів.

Також в даній роботі розглянуті основи мов програмування веб-сторінок - HTML - CSS та JavaScript, які є загальноприйнятим стандартом при створенні сайтів. Це дозволить ознайомитися зі структурою веб-сторінки і методами їх правильного оформлення.

1 АНАЛІЗ МЕТОДІВ РОЗРОБКИ ВЕБ-ДОДАТКІВ

Існує три основних підходи до розробки веб-додатків: односторінковий (SPA), багатосторінковий (MPA) і прогресивний (PWA). Вони виділяються серед інших підходів простотою розробки, зручністю використання та широкими можливостями для розвитку бізнесу.

1.1 Види веб-додатків

Односторінкові програми

SPA або Single Page Application – це односторінкове веб-додаток, який завантажується на одну сторінку HTML. Завдяки динамічному оновленню за допомогою JavaScript не потрібно перезавантажувати або завантажувати додаткові сторінки під час використання. На практиці це означає, що користувач бачить весь основний вміст у браузері, а при прокрутці або переході на інші сторінки замість повного перезавантаження просто завантажуються необхідні елементи.

У процесі роботи користувачеві може здатися, що він запустив не веб-сайт, а настільний додаток, оскільки воно миттєво реагує на всі його дії, без затримок і «зависань».

Цього ефекту можна досягти за допомогою передових фреймворків JavaScript: Angular, React, Ember, Meteor, Knockout.

Приклади динамічних програм: Gmail, Google Maps, Facebok, GitHub, Meduza.

Переваги

Висока швидкість – всі ресурси завантажуються за один сеанс, а під час дій на сторінці дані просто змінюються, що економить час;

гнучкість і швидкість реагування користувальницького інтерфейсу - завдяки тому, що є лише одна веб-сторінка, легше будувати насичений інтерфейс, зберігати інформацію про сеанс, керувати станами перегляду та анімацією;

Спрощена розробка - ви можете почати писати код з файлу: // файл URL без використання сервера, окремий код не потрібен для відтворення сторінки на стороні сервера;

кешування даних - програма відправляє лише один запит, збирає дані і після цього може працювати в автономному режимі.

Недоліки

Seo оптимізація вимагає рішень у вигляді серверного рендерингу – через те, що контент завантажується за технологією AJAX, що означає динамічну зміну вмісту сторінки, а стабільність важлива для оптимізації;

навантаження на браузер - через те, що клієнтські фреймворки важкі, вони довго завантажуються;

Необхідна підтримка JavaScript - без JS ви не можете повноцінно використовувати всю функціональність програми;

Витік пам’яті Java Script – через погану безпеку SPA більш вразливий до зловмисників і витоку пам’яті.

1.1.2 Багатосторінкові програми

MPA або багатосторінковий додаток — це багатосторінкові програми, які працюють традиційним чином. Це означає, що щоразу, коли дані незначно змінюються або завантажується нова інформація, сторінка оновлюється. Такі програми важчі за односторінкові програми, тому їх доцільно використовувати лише тоді, коли потрібно відобразити великий обсяг контенту.

Переваги

Проста SEO оптимізація - ви можете оптимізувати кожну зі сторінок програми під потрібні ключові слова;

знайомість для користувачів - завдяки простому інтерфейсу та класичній навігації.

Недоліки

Між бекендом і фронтендом існує тісний зв’язок, тому їх не можна розвивати паралельно; комплексна розробка – вони вимагають використання фреймворків як на стороні клієнта, так і на стороні сервера, що збільшує час розробки та бюджет.

комплексна розробка – вимагають використання фреймворків як на стороні клієнта, так і на стороні сервера, що збільшує час розробки та бюджет.

1.2 Відмінності між видами веб-додатків

SPA і MPA. Що вибрати?

Вибираючи тип веб-додатка, ви повинні зосередитися на тому, для чого ви його створюєте. Багатосторінковий сайт підходить для інтернет-магазину з великою кількістю товарів і послуг, а якщо у вас є, наприклад, інфобізнес, де ви можете подати всю інформацію в стиснутому веб-просторі, односторінковий сайт - це підходить.

Вибір SPA

Бажано, якщо:

- є потреба в багатофункціональному, багатому інтерфейсі користувача;
- є потреба в API - використання готових блоків для створення програми.

Вибір MPA

Бажано, якщо:

- програми використовуються тільки для читання інформації;
- є необхідність використовувати програму в браузерах без підтримки JavaScript.

Навіщо потрібні PWA

Прогресивні програми або прогресивні веб-додатки взаємодіють з користувачем як програма. Їх можна встановити на головний екран смартфона, відправляти push-повідомлення та працювати в автономному режимі.

Приклад: Google Docs.

Переваги

Кросплатформенність - може працювати з декількома операційними системами одночасно;

висока швидкість роботи і можливість запуску і відображення даних в автономному режимі з миттєвим завантаженням;

висока швидкість монтажу;

швидкий розвиток - для створення PWA не потрібен окремий сайт, потрібно лише змінити існуючий.

Недоліки

Не всі браузери підтримують основні функції цих програм (наприклад, Firefox і Edge).

2 СТВОРЕННЯ ВІЗУАЛЬНОЇ СТРУКТУРИ ДОДАТКУ ДЛЯ УНИКНЕННЯ СКЛАДНОЩІВ ПРИ ПОДАЛЬШІЙ РОЗРОБЦІ

Візуальна ієрархія на сайті — це організація та оформлення інформації таким чином, щоб відвідувач міг швидко зрозуміти інтерфейс і відрізнити головне від другорядного.

2.1 Типи конструкцій сайту

Якщо у вас є семантичне ядро і ви розумієте, які матеріали повинні бути на сайті, структура ресурсу повинна вийти у вигляді діаграми. Іншими словами, ієрархія, логічний і послідовний ланцюг побудови та подання інформації.

Яка може бути схема, залежить від призначення сайту. Виходячи з цього, вибирається найбільш підходящий тип конструкції:

Алфавітна організація. Передбачає розташування інформації в алфавітному порядку. Зручно, коли клієнт точно знає назву товару, який він шукає.

Хронологічні. Найчастіше використовується при розміщенні новин, прес-релізів, дозволяє орієнтуватися за датою виходу потрібного матеріалу.

Деревоподібна. Коли є головна сторінка, і з неї вже надходять різні категорії, картки товарів і тому подібне. Це популярна структура інтернет-магазину, розробкою якої потрібно займатися дуже ретельно.

Тематичний. Зручна навігація, коли вся інформація впорядкована за темами.

Лінійний. При цьому вся інформація розташовується в ланцюжку зі своїми компонентами. Людина тут не має можливості переходити зі сторінки на сторінку, як йому заманеться, все відбувається за певним маршрутом. Це типовий варіант візитки.

Решітка. Всі його компоненти розміщені в окремих гілках. Частіше його можна знайти в інформаційних каталогах статей.

Павутина. Це комбінована конструкція, що передбачає поєднання кількох видів. Однак через свою складність він використовується рідко.

Гібридний. Він також передбачає комбінацію кількох типів структур, що дозволяє користувачам простіше і швидше знаходити потрібну інформацію.

Ієрархічна структура сайту дозволяє користувачам швидко знайти продукт, який їх цікавить, не витрачаючи на це зайвий час.

2.2 Створення схематичного відображення структури додатку

Прототип веб-сайту, також відомий як каркас, — це схематичне зображення однієї або кількох сторінок веб-сайту, які відображають структурне розташування елементів інтерфейсу.

Як правило, прототип сайту містить такі компоненти:

- Кнопки
- Форми потенційних клієнтів
- меню сайту
- Повзунки
- Тизери
- тощо

Навіщо потрібен прототип сайту

Прототипування веб-інтерфейсів дозволяє оцінювати та аналізувати поведінкові сценарії на ресурсі на ранній стадії, покращувати їх, а також уникати глобальних покращень у майбутньому. Крім того, прототип допомагає заощадити час і гроші на розробку. Це пов’язано з тим, що внесення змін до проекту, навіть структурних, є досить простим завданням на початковому етапі. Це не вимагає особливих зусиль і часу, тому що редагувати схему легше і легше, ніж робити це на живому сайті.

Створення прототипів — це важливий крок у веб-розробці, який дозволяє визначити та сформувати ДНК проекту на ранній стадії. За останні роки значно зросла кількість онлайн-сервісів та інструментів для створення прототипів. У цьому огляді представлено 10 сервісів для створення візуальних карт сайту.

Перш ніж перейти до опису інструментів, не буде зайвим розібратися в термінології.

Карта сайту — це структура сторінок сайту, представлена в ієрархічній моделі. Ця схема допомагає оцінити обсяг сторінок, які утворюють сайт, а також зрозуміти логіку їх взаємозв'язку. Іноді це примітивні за організацією та структурою схеми. А іноді дуже складні карти з багаторівневою вкладеністю. Карти сайту допомагають спланувати поширення контенту та механіку навігації майбутнього сайту.

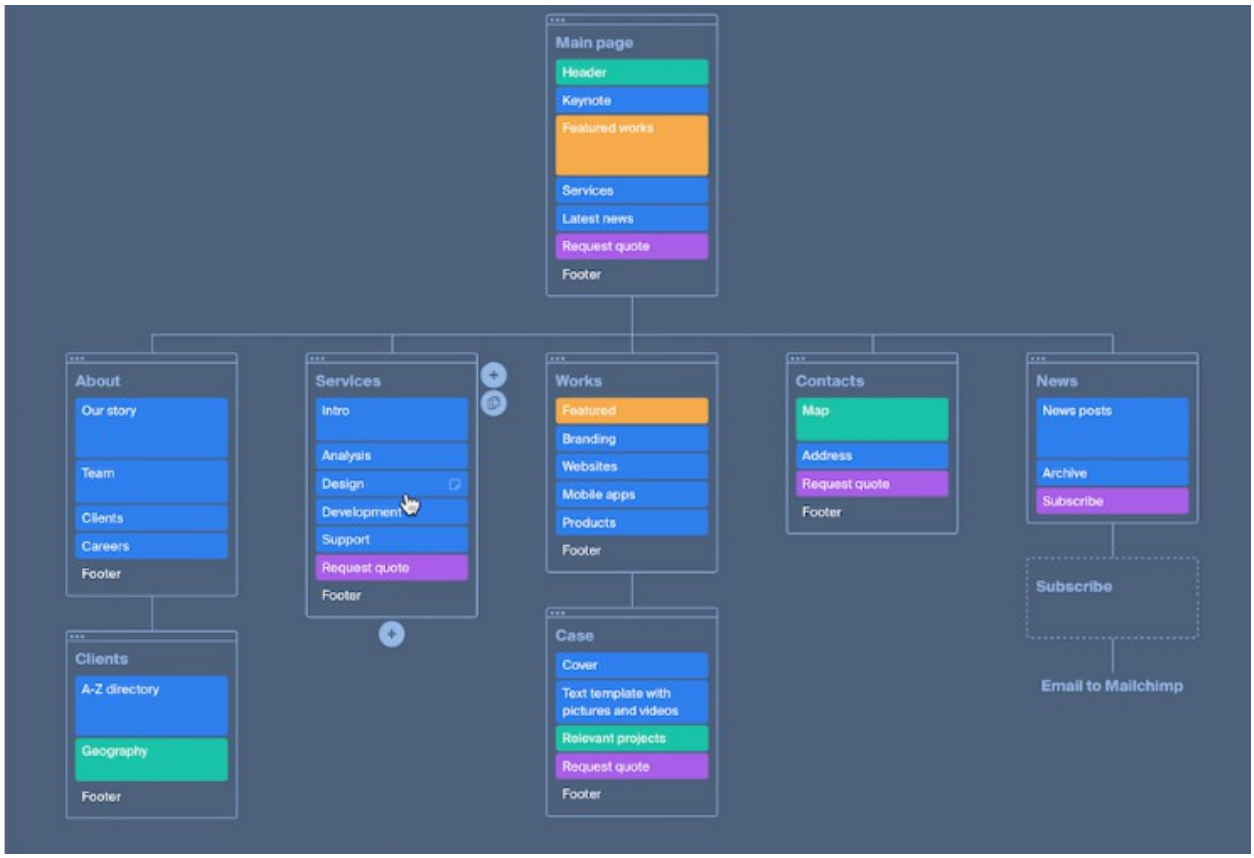

Рисунок 2.1 - Прототип сайту.

3 СТВОРЕННЯ ГОТОВОГО САЙТУ З ПОВНИМ НАБОРОМ ФУНКЦІОНАЛУ

У цьому розділі ми опишемо основні аспекти створення клієнтської частини веб-додатку та опишемо ті чи інші особливості різних функціональних частин.

3.1 Вибір фреймворку на базі якого будемо будувати наш інтернет додаток.

Є три основні фреймворки.

- React.js

React - це бібліотека javaScript з відкритим кодом для розробки інтерфейсів користувача. React розробляється і підтримується Facebook, Instagram, а також спільнотою окремих розробників і корпорацій. React можна використовувати для розробки односторінкових і мобільних додатків.

React пропонує легке та функціональне створення компонентів і сприяє їх використанню для підтримки елегантного коду API. Фреймворк дуже популярний, особливо в різних стартапах. Завдяки великому вибору легкодоступних плагінів і розширень з відкритим кодом ви можете розробити практично будь-який тип веб-сайту.

Особливості:

- компонентно орієнтований;
- декларативний;
- Продуктивно (завдяки React Virtual DOM)
- серверний рендеринг;
- наявність Redux;
- підтримка PWA;
- JSX.

- Angular.js

Angular — відкрита і безкоштовна платформа для розробки веб-додатків, написана на TypeScript, розроблена командою Google, а також спільнотою розробників з різних компаній. Angular — це повністю переписаний фреймворк від тієї ж команди, яка написала AngularJS.

Angular забезпечує кращий CLI, ніж React або Vue, що полегшує створення надійних рішень. Angular популярний у корпоративному середовищі, де він має широку аудиторію. Код дещо перенасичений і складний порівняно з іншими фреймворками.

Особливості:

- універсальний;
- нативний;
- складний;
- використовується з Typescript;
- інтелектуальне автозаповнення HTML-компонента;
- архітектура розрахована на великі проекти;
- MVVM модель.

- Vue.js

Vue.js — це фреймворк JavaScript з відкритим кодом для створення інтерфейсів користувача. Легко інтегрується в проекти з використанням інших бібліотек JavaScript. Він може функціонувати як веб-фреймворк для розробки односторінкових програм реактивного стилю.

Vue — це молодий фреймворк із зростаючою аудиторією. Найпростіший для навчання в трійці лідерів, ви можете почати “з коробки”, але досить потужний для професійних розробників. Vue не має стільки вбудованих функцій, як Angular, але більше, ніж React.

Особливості Vue:

- прив'язка даних;
- продуктивний (є віртуальний DOM);

- легкість навчання;
- адаптивність і детальна документація;
- масштабування;
- оптимізація блоків HTML;
- мініатюрний розмір;
- шаблони.

Інструменти

Усі три фреймворки забезпечують CLI, що полегшує створення нових проектів і підтримує безперервний розвиток. Крім того, всі вони добре уживаються з популярними IDE, такими як VS Code і Atom.

Продуктивність

Звичайно, продуктивність може змінюватися в залежності від ситуації, але здебільшого всі три фреймворки досить швидкі – недарма вони настільки популярні.

Продуктивність зазвичай не є основним фактором при виборі між перерахованими фреймворками.

Оскільки найбільше досвіду роботи я маю в роботі з React.js то його я й оберу для прикладів з кодом нижче.

3.2 GraphQL і REST

REST — це архітектурний стиль взаємодії між компонентами розподіленої програми в мережі. Іншими словами, REST — це набір правил щодо того, як програміст повинен організувати кодування серверного додатка, щоб усі системи могли легко обмінюватися даними, а програма могла масштабуватися.

GraphQL — це мова запитів і маніпулювання даними з відкритим вихідним кодом для API, а також середовище виконання для виконання запитів до наявних даних.

GraphQL часто представляється як революційно новий спосіб думати про API. Замість того, щоб працювати з жорстко закодованими кінцевими точками на сервері, ви можете отримати саме потрібні дані за допомогою одного запиту. І так – GraphQL є гнучким під час розгортання в організації, що робить співпрацю між зовнішніми та внутрішніми компонентами більш гладкою, ніж будь-коли. На практиці, однак, обидві ці технології передбачають надсилання HTTP-запиту та отримання якогось результату, і багато елементів з моделі REST вбудовані в GraphQL.

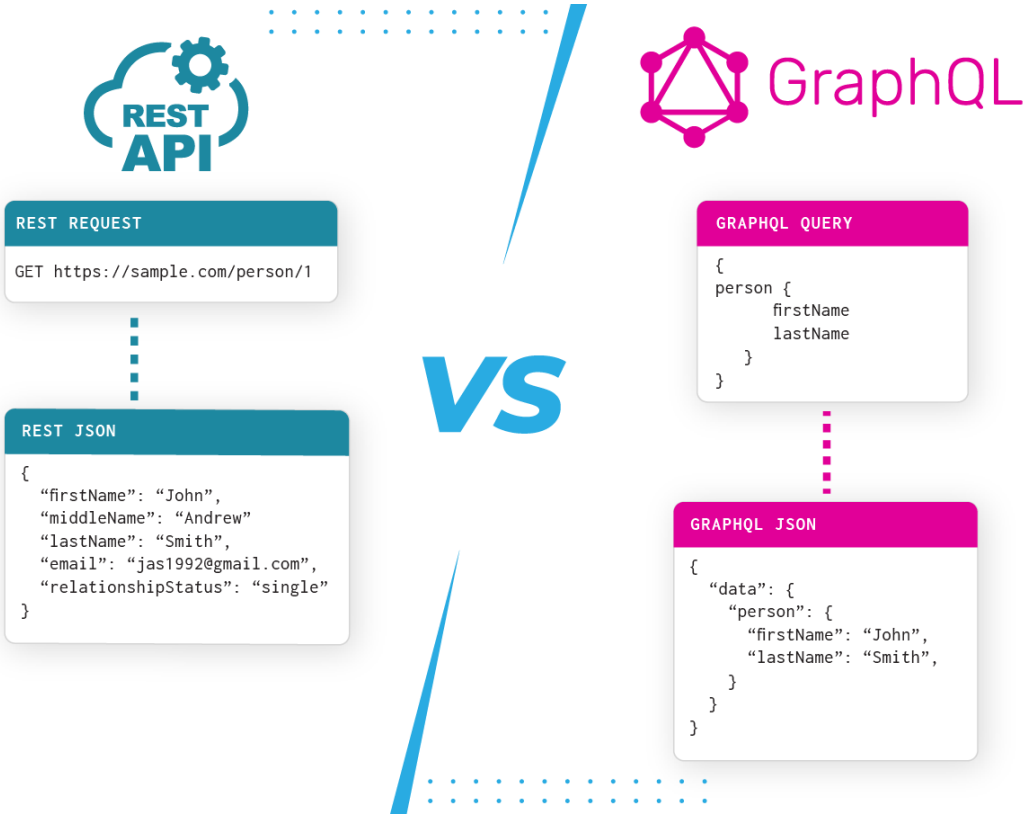

Рисунок 3.1 – Порівняння REST API та GRAPHQL

Я вважаю що для данного типу веб додатку краще підійде GRAPHQL

3.3 Реалізація авторизації

Для цього ми використовуємо HOC

```
export const withAuth = (WrappedComponent) =>
  withRouter(
    withApollo((props) => {
      const [geolocationLoading, toggleLoading] = useState(false);

      useEffect(() => {
        verifyUser();
      }, []);

      const verifyUser = async () => {
        if (!props.location.pathname.includes("post")) {
          try {
            await props.client.mutate({
              mutation: mutate.VERIFY_TOKEN,
              variables: {
                token: JSON.parse(localStorage.getItem("token")).token || "",
              },
            });
          } catch (error) {
            props.history.push(path.SIGN_IN);
            window.localStorage.removeItem("token");
          }
        }
      };

      const getLocations = () =>
        new Promise((resolve, reject) => {
          const options = {
            enableHighAccuracy: false,
            maximumAge: 0,
            timeout: 5000,
          };

          const success = (position) => {
            const latitude = position.coords.latitude;
            const longitude = position.coords.longitude;
            resolve({ latitude, longitude });
          };

          const error = (err) => reject(`ERROR(${err.code}): ${err.message}`);

          navigator.geolocation.getCurrentPosition(success, error, options);
        });

      const handleAllowGelolocation = async () => {
        try {
          toggleLoading(true);
          const getCoordinates = await getLocations();
          const response = await props.client.mutate({
            mutation: mutate.ALLOW_GEOLOCATION,
            variables: {
              isAllowed: true,
              ...getCoordinates,
            },
            refetchQueries: [{ query: query.USER }],
          });
          if (response.data.allowGeoLocation.success) {
            toggleLoading(false);
            window.location.reload();
          }
        } catch (error) {
          toggleLoading(false);
          console.log(error);
        }
      };

      return (
```

3.4 Корисні утиліти для функціонування сайту

Для більшості завдань нам знадобляться різні утіліти котрі будуть виконувати не значну роботу але зменшать кількість всього коду та полегшать його написання.

Ми будемо використовувати як локальні утіліти котрі можуть використовуватися лише в одному проекті та залежать від того як зроблена структура та інші особливості проекту та утіліти котрі не прив'язані до проекту.

Такі утіліти зазвичай виконують перетворення даних або взаємодіють з ними.

Наприклад запис токену в LocalStorage (хочя більше підходить запис до coocie).

```
export const saveData = (value) => {
  const timestamp = +new Date();
  const data = JSON.stringify({ token: value, timestamp });

  localStorage.setItem("token", data);
};

export const deleteData = () => {
  localStorage.removeItem("token");
};

export const getBase64 = (file) =>
  new Promise((resolve, reject) => {
    const reader = new FileReader();
    reader.readAsDataURL(file);
    reader.onload = () => resolve(reader.result);
    reader.onerror = (error) => reject(error);
  });

export const dataURLtoFile = (dataurl, filename) => {
  var arr = dataurl.split(","),
    mime = arr[0].match(/:(.*?);/)[1],
    bstr = atob(arr[1]),
    n = bstr.length,
    u8arr = new Uint8Array(n);

  while (n--) {
    u8arr[n] = bstr.charCodeAt(n);
  }

  return new File([u8arr], filename, { type: mime });
};

export const getItem = (maxLen) => {
  const array = [];
  let len = 1;

  while (len <= maxLen) {
    array.push(len);
    len += 1;
  }

  return array;
};

export const withPromise = (f, delay) =>
  new Promise((resolve, reject) => {
    setTimeout(() => {
      resolve(f);
    }, delay);
  });

export const showAlert = (title, message, type) =>
  store.addNotification({
    title,
    message,
    type: type || "info",
    insert: "top",
    container: "top-right",
    dismiss: {
      duration: 2000,
      pauseOnHover: true,
      showIcon: true,
      click: true,
    },
  });

export const getMilliseconds = (time) => ((time % 60000) / 1000).toFixed(0);

export const preloader = () => {
  return (
    <div className="loading">
      <Loader active inverted />
    </div>
  );
};

export const loader = (
  <div className="linear-activity">
    <div className="indeterminate"></div>
  </div>
);
```

3.5 User provider

Його ми використовуємо для збереження сесії користувача це дає змогу не вводити кожен раз логін та пароль і продовжувати роботу з сайтом навіть після вимкнення компьютеру.

```
import React, { useEffect } from "react";
import { withRouter, Redirect } from "react-router-dom";
import { withApollo } from "react-apollo";

import { useQuery } from "utils/hooks.js";
import * as query from "api/queries";
import * as path from "constants/routes";

export const withUserProfile = (Route) => {
  const RouteWithProfile = withRouter(
    withApollo((props) => {
      const { getData, loading, error, data } = useQuery({
        client: props.client,
        endpoint: query.USER,
        entity: "me",
        router: props.history,
      });

      useEffect(() => {
        getData();
      }, []);

      if (loading || !data) return null;
      if (error) return null;

      if (!data.profiles.length) {
        return <Redirect to={path.CREATE_PROFILE} />;
      }

      return <Route {...props} />;
    })
  );

  return React.memo(RouteWithProfile);
};
```

3.6 Константи для посилань.

На малюнку зображені базові константи котрі зберігають в собі посилання на сторінки та завдяки їм можна гнучко налаштувати роутинг.

Також можна зробити роутинг більш гнучким та маштабованим завдяки використанню функций котрі приймають частину посилання та динамічно их змінюючи.

```
export const CONTACT_US = "/contact-us";
export const COOKIE_POLICY = "/cookie-policy";
export const PRIVACY_POLICY = "/privacy-policy";
export const TERMS_AND_CONDITIONS = "/terms-and-conditions";
export const FAQ = "/faq";

export const SIGN_UP = "/sign-up";
export const SIGN_IN = "/sign-in";
export const FORGOT_PASSWORD = "/forgot-password";
export const SUCCESS = "/success";
export const VERIFICATION = "/verification";
export const CHECK_MAIL_BOX = "/check-mailbox";
export const CHANGE_PASSWORD = "/change-password";

export const CREATE_PROFILE = "/profile/create";
export const EDIT_PROFILE = "/profile/edit";
```

3.7 GRAPHQL запити.

Також ми додамо необхідні запити та мутації такі як:

- CATEGORIES - категорії курсів/публікацій та інш.(Потрібно створити декілька різних запитів для різних категорій або додати варіативність завдяки параметру котрий ми передамо в запит)

```
export const CATEGORIES = gql`
  Execute Query
  query categories {
    categories {
      id
      name
      hasPosts
    }
  }
`;
```

- NOTIFICATIONS - повідомлення.

Також нам знадобиться забит для повідомлень.

Вони будуть відрізнятися від звичайних запитів тому що нам потрібно постійно прослуховувати всі зміни пов'язані з користувачем.

```
export const NOTIFICATIONS = gql`
  Execute Query
  query notifications {
    notifications {
      id
      profile {
        id
        name
      }
      actionType
      notificationText
    }
  }
`;
```

- ME - цей запит відповідає за інформацію користувача.

А саме його ім’я, прізвище, стать та інше.

```
export const USER = gql`
  Execute Query
  query me {
    me {
      categories {
        id
        name
        useForMatches
        posts {
          id
        }
      }
      id
      geoLocationAllowed
      email
      isVerified
      lastVerificationCode
      searchRadius
      fullName
      gallery {
        id
        image
      }
      profiles {
        id
        name
        education
        description
        job
        age
        avatar {
          id
          image
          source {
            id
            image
          }
        }
        age
        gender
        isActive
```

```
      notificationsSettings {
        id
        likeMe
        likeMeEmail
        newReply
        newReplyEmail
        newMatch
        newMatchEmail
        newMessage
        newMessageEmail
      }
      user {
        id
        fullName
        email
      }
    }
    addMatchPost
    addListingPost
    locations {
      id
      latitude
      longitude
      city
      country
      isActive
    }
    liked
    postsLeft
    repliesLeft
    profilesLeft
    userPlan {
      id
      availableLikes
      availableReplies
      availablePosts
      availableProfiles
    }
    subscriptionValid
    currentLocation {
      id
      city
      country
      longitude
      latitude
      isCurrentLocation
      isActive
    }
    subscription {
      id
      expirationDate
      autoUpdate
    }
  }
}
`;
```

- ROOMS - цей запит поверне нам список всіх чатів які в нас є.

```
export const ROOMS = gql`
  Execute Query
  query rooms($userId: Int) {
    rooms(userId: $userId) {
      id
      isSupportChat
      users {
        id
        name
        isActive
        education
        age
        job
        description
```

```
      avatar {
        id
        image
      }
    }
    unviewedRooms
    lastMessage {
      attachments {
        id
        image {
          id
          source {
            id
            image
          }
          image
        }
      }
      id
      text
      sender {
        id
        name
        education
        description
        age
        job
        avatar {
          id
          image
        }
      }
      seen
      time
    }
    typing
  }
 }
`;
```

- ROOM - запит відповідає за чат між декількома людьми.

Окрім цього запиту потрібно зробити багато інших запитів, мутацій та підписатися на динамічне отримання нових повідомлень в чаті.

Подібним чином ми обробляємо глобальні повідомлення.

```
export const ROOM = gql`
  query room(
    $id: Int
    $firstUser: ID
    $secondUser: ID
    $first: Int
    $skip: Int
  ) {
```

```
  room(
    id: $id
    firstUser: $firstUser
    secondUser: $secondUser
    first: $first
    skip: $skip
  ) {
    id
    isSupportChat
    users {
      id
      name
      isActive
      avatar {
        id
        image
      }
    }
    lastMessage {
      id
      attachments{
        id
        image {
          id
          source{
            id
            image
          }
          image
        }
      }
      text
      sender {
        id
        name
        avatar {
          id
          image
        }
      }
      seen
      time
    }
```

```
      typing
      messages(first: $first, skip: $skip, room: $id) {
        id
        text
        post1{
          ${postFragment}
        }
        post2{
          ${postFragment}
        }
        attachments{
          id
          image {
            id
            source{
              id
              image
            }
            image
          }
        }
        sender {
          id
          name
          avatar {
            id
            image
          }
        }
        seen
        time
      }
    }
  }
`;
```

Для створення чату між користувачами використати таку мутацію

```
export const CREATE_ROOM = gql`
  Execute Mutation
  mutation createRoom($roomInput: RoomInput!) {
    createRoom(roomInput: $roomInput) {
      errors
      success
      room {
        distance
        isSupportChat
        id
        users {
          id
          name
          avatar {
            id
            image
          }
        }
        lastMessage {
          id
          text
          seen
          sender {
            id
          }
          isDeleted
          time
        }
      }
    }
  }
`;
```

Для підтвердження створення чату між користувачами використаємо мутацію APPROVE_ROOM

```
export const APPROVE_ROOM = gql`
  Execute Mutation
  mutation acceptRoom(
    $accept: Boolean!
    $roomId: ID!
    $isMatchPair: Boolean!
    $senderId: ID
    $message: String
  ) {
    acceptRoom(
      accept: $accept
      roomId: $roomId
      isMatchPair: $isMatchPair
      senderId: $senderId
      message: $message
    ) {
      success
      errors
      room {
        distance
        isSupportChat
        id
```

```
      users {
        id
        name
        avatar {
          id
          image
        }
      }
      lastMessage {
        id
        text
        sender {
          id
          name
          avatar {
            id
            image
          }
        }
        seen
        time
      }
      typing
      messages {
        id
        text
        sender {
          id
          name
          avatar {
            id
            image
          }
        }
        seen
        time
      }
    }
  }
}
`;
```

Для відправки повідомлення використаємо мутацію CREATE_MESSAGE

```
export const CREATE_MESSAGE = gql`
  Execute Mutation
  mutation createMessage($attachments: [ID], $messageInput: MessageInput!) {
    createMessage(attachments: $attachments, messageInput: $messageInput) {
      success
      errors
      message {
        id
        text
        sender {
          id
          name
        }
        room {
          # distance
          isSupportChat
          id
          users {
            id
            name
          }
        }
        seen
        time
        isDeleted
      }
    }
  }
`;
```

3.8 UI бібліотека.

У якості UI бібліотеки ми будемо використовувати Material UI.

Material UI - це набір компонентів React, які реалізує Google Material Design (нещодавно випущені бібліотеки material-ui v1). Ці компоненти працюють ізольовано, а це означає, що вони є самопідтримкою та вводять лише ті стилі, які їм потрібні для відображення.

Адаптований інтерфейс користувача Material Design заснований на макеті з 12 колонками. Ця сітка створює візуальну узгодженість між макетами.

Система сітки material-ui характеризується наступним:

Використовує Flexbox

Містить два типи макетів: контейнери та елементи

Елементам надається ширина у відсотках, тому вони завжди гнучкі та мають розмір відносно свого батька

Елементи мають відступ, щоб створити відстань між окремими елементами.

Існує п’ять точок розриву сітки: xs, sm, md, lg, xl

- Компоненти Material-ui

material-ui містить багато компонентів інтерфейсу користувача, які допоможуть вам створити додаток React у темі Material Design.

Нижче наведено зображення з прикладом Material UI компонентів.

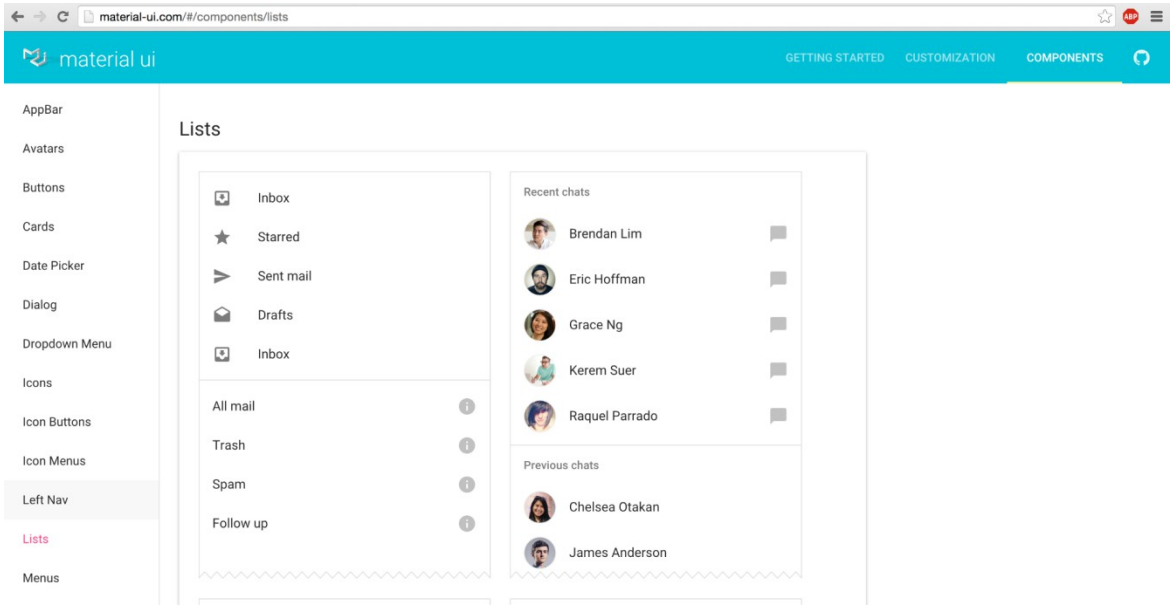

Рисунок 3.2 – Зразок компонентів бібліотеки Material UI

- Список сітки

Списки сітки є альтернативою стандартним представленням списків, які розглядалися вище. Список сітки складається з повторюваних клітинок, розташованих вертикально та горизонтально, і зазвичай відображає зображення.

Додатковий приклад списку Grid можна побачити нижче, він використовує GridListTileBar для додавання накладання до кожного GridListTile. Накладання може містити заголовки, підзаголовки та додаткову дію — IconButton у цьому прикладі — яку можна використовувати для передачі додаткової інформації.

3.9 Для реалізації відео зв’язку ми будемо використовувати web-socket

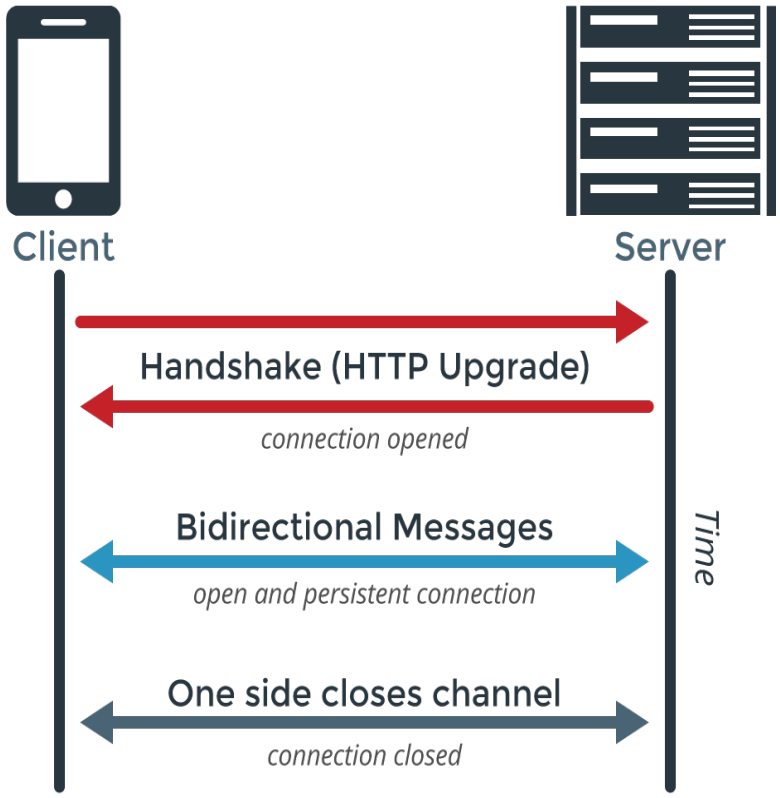

Рисунок 3.3 – Принцип дії WEBSOCKET

WEBSOCKET - Протокол з'єднання TCP призначений для обміну повідомленнями між браузером і веб-сервером в режимі реального часу.

Наразі W3C стандартизує API веб-сокетів. Проект стандарту цього протоколу схвалено IETF.

WebSocket розроблено для реалізації у веб-браузерах і веб-серверах, але його можна використовувати для будь-якого клієнтського або серверного додатка. Протокол WebSocket є незалежним протоколом, заснованим на протоколі TCP. Це забезпечує тісну взаємодію між браузером і веб-сайтом, полегшуючи поширення інтерактивного вмісту та створення додатків у реальному часі.

WebRTC (зв'язок у реальному часі) - Інтернет-протокол з відкритим вихідним кодом, призначений для організації голосового та відеозв'язку через Інтернет у реальному часі.

Його ми будемо використовувати для відеозв’язку.

Він працює на базі WEBSOCKET та є на даний момент одним з кращих інструментів для такої задачі.

ВИСНОВКИ

Розробка та управління веб-додатками з кожним роком стає все складнішими . В тому числі й через зростання потреб до додатків.

Метою роботи є порівняльний аналіз методів створення веб-додатків і розробка сайту дистанційного навчання.

Існує багато підходів до створення простих сайтів та веб-додатків.

Ми переглянули та порівняли найпопулярніші з них та отримали уявлення щодо ситуації з інструментами для створення веб-додатків.

Під час дипломного проектування було розібрано й представлено один варіант з використанням сучасних технологій.

Загалом в роботі проведено аналіз методів створення веб-сторінок на прикладі сайту дистанційного навчання. Розглянуті різні технології та інструменти котрі виконують схожі задачи але роблять це різними способами.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Фленаган Д. JavaScript. Полное руководство, 7-е издание ISBN 978-5-907203-79-2

2. React documentation https://devdocs.io/react/