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 САЙТУ ДЛЯ АЛГОРИТМІЧНОГО СЕРЕДОВИЩА

Виконав студент 4-го курсу
Гелла Всеволод Вячеславович _____
(підпис)

Науковий керівник:
професор, доктор фіз.-мат. наук
Терещенко Василь Миколайович _____
(підпис)

Консультант:
Шкільняк Степан Степанович _____
(підпис)

Засвідчую, що в роботі немає запозичень з праць інших авторів без відповідних посилань
Студент _____
(підпис)

Роботу розглянуто й допущено до захисту на засіданні теорії та технології програмування
« 01 » червня 2022р.,
протокол №10
Завідувач кафедри
М. С. Нікітченко _____
(підпис)

КИЇВ – 2022

РЕФЕРАТ

Обсяг роботи 40 сторінок, 6 ілюстрацій, 29 джерел посилань.

ВЕБСАЙТ, РОЗРОБКА САЙТУ, СИСТЕМА ОЦІНЮВАННЯ, ІНТЕРФЕЙС ПРОГРАМНОГО ПРОДУКТУ, АЛГОРИТМІЧНЕ СЕРЕДОВИЩЕ, ТЕХНІЧНЕ ЗАВДАННЯ ДО ПРОДУКТУ.

Об’єктом роботи є процес розробки платформи для демонстрації користувачеві результатів оцінювання задач, котрі раніше були розв’язані самим користувачем. Предметом роботи є програмний засіб для представлення результатів оцінювання певних задач.

Метою роботи є розробка та створення сайту для зручного показу результатів оцінювання задач, які представленні в єдиному алгоритмічному середовищі.

Методи розроблення: методи проектування вебсайтів, розробка зручного інтерфейсу вебсайту, розробка структури вебсайту. Інструменти розроблення: безкоштовне, вільно поширюване інтегроване середовище розробки Microsoft Visual Studio Code 1.67.2, мова програмування JavaScript та її бібліотеки, мова розмітки HTML та каскадні таблиці стилів CSS.

Результати роботи: виконано загальний огляд технологій, які використовуються при розробці вебсайтів, зокрема покращені знання мови програмування JavaScript, проведений аналіз її популярних бібліотек і фреймворків, виявлення переваг та недоліків їх застосування у створенні продукту, розробка сайту для демонстрації результатів оцінювання задач, які представленні в єдиному алгоритмічному середовищі.

При розробці сайту також використовувалися бібліотеки CGLib, яка формує алгоритмічне середовище, в котрому представлені відповідні задачі та CGMark, яка реалізує систему оцінювання задач з попередньої бібліотеки. Обидві бібліотеки розроблені моїми колегами Германюком Всеволодом з групи ТТП-4 та Фісуненко Артемом з групи МІ-4.

Створений сайт для зручної демонстрації результатів оцінювання задач, які представленні в єдиному алгоритмічному середовищі може застосовуватися в

навчальному процесі факультету комп’ютерних наук та кібернетики під час вивчення дисципліни «Обчислювальна геометрія та комп'ютерна графіка».

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

DOM − Document Object Model, об'єктна модель документа;

BOM − Browser Object Model, об'єктна модель браузера;

HTTP − HyperText Transfer Protocol, протокол передачі гіпертекстових документів;

URL − Uniform Resource Locator, уніфікований локатор ресурсів;

API − Application Programming Interface, прикладний програмний інтерфейс;

REST − Representational State Transfer, передача репрезентативного стану;

JSON − JavaScript Object Notation, запис об'єктів JavaScript;

DFR − Django REST Framework.

ВСТУП

Оцінка сучасного стану об’єкта розробки. Час – один з найголовніших ресурсів людства. У сучасному світі кожна людина має намір оптимізувати свою працю, тобто виконати більше завдань за менший час. Проте існує необхідність у перевірці якості виконання тієї чи іншої роботи, тому люди почали створювати певні системи оцінювання, які б і дозволили зробити певні висновки. Зрозуміло, що на це також потрібний певний ресурс, тому з плином часу та розвитку технологій, зокрема сфери інформаційних технологій людство поставило перед собою нову задачу, а саме: автоматизація системи оцінювання у всіх сферах життя, де це можливо.

Наразі, багато університетів прагнуть розробити певну платформу для проведення контролю знань та оцінювання відповідних робіт, адже відомим фактом є те, що більшість викладачів здійснюють перевірку робіт ручним способом.

Актуальність роботи та підстави для її виконання. При вивченні дисципліни «Обчислювальна геометрія та комп'ютерна графіка» студенти та викладач стикаються з проведенням контролю знань, що має на меті перевірити як студенти навчились вирішувати задачі, про які йшла мова протягом дисципліни.

Постає задача спростити та автоматизувати процес перевірки контрольних робіт. Інтеграція подібної системи до навчального процесу позитивно вплине на швидкість та якість проведення та перевірки відповідних робіт студентів. Проте можливості для автоматизації присутні не лише для цієї дисципліни, тому перспектива розвитку та розширення проекту є доволі великою.

Мета й завдання роботи. Метою кваліфікаційної роботи є створення сайту для зручного показу результатів оцінювання задач, які представленні в єдиному алгоритмічному середовищі. Для досягнення цієї мети поставлено наступні завдання:

- Дослідити сучасні засоби для створення вебсайтів.
- Поглибити знання в існуючих засобах для створення вебсайтів та здобути нові в невідомих до цього раніше.

- Проаналізувати доступні відкриті системи оцінювання, або системи, що можуть використовуватися у якості таких.
- Розробити технічне завдання до сайту.
- Розробити інтерфейс та дизайн сайту.

Об'єкт, методи й засоби розроблення. Об’єктом розробки є процес створення платформи для демонстрації користувачеві результатів оцінювання задач, котрі раніше були розв’язані самим користувачем.

Розробці вебсайту передувало створення моїми колегами Германюком Всеволодом та Фісуненко Артемом бібліотек CGLib та CGMark, які представляють задачі в алгоритмічному середовищі та систему їх оцінювання відповідно. Завдяки використанню.

Під час розробки сайту були дотримані наступні етапи розробки вебсайту:

- Постановка задачі при проектуванні вебсайту.
- Вибір програмних засобів.
- Розробка інтерфейсу.
- Розробка структури вебсайту.

В якості інструменту створення засобу було обрано Microsoft Visual Studio Code 1.67.2 – безкоштовне, вільно поширюване інтегроване середовище розробки для багатьох мов програмування, зокрема і JavaScript, мови розмітки HTML та каскадних таблиць стилів CSS. Для зручної роботи в середовищі необхідно лише встановити відповідні розширення.

JavaScript найбільше використовується як мова сценаріїв вебсторінок. А наявність великої кількості бібліотек для цієї мови програмування спрощує розробку вебсайтів.

Можливі сфери застосування. Створений сайт для зручної демонстрації результатів оцінювання задач, які представленні в єдиному алгоритмічному середовищі може застосовуватися в навчальному процесі факультету комп’ютерних наук та кібернетики під час вивчення дисципліни «Обчислювальна геометрія та комп'ютерна графіка».

Взаємозв’язок з іншими роботами. При розробці сайту також використовувалися бібліотеки CGLib, яка формує алгоритмічне середовище, в котрому представлені відповідні задачі та CGMark, яка реалізує систему оцінювання задач з попередньої бібліотеки. Обидві бібліотеки розроблені моїми колегами Германюком Всеволодом з групи ТТП-4 та Фісуненко Артемом з групи МІ-4.

РОЗДІЛ 1 ВЕБСАЙТ

1.1 Поняття вебсайту

Веб-сайт – сукупність вебсторінок та залежного вмісту, доступних у мережі Інтернет, які об’єднані як за змістом, так і за навігацією під єдиним доменним ім’ям. Фізично сайт може розміщуватися як на одному, так і на кількох серверах[1]. Вебсторінки — це текстові файли з розширенням *.html, які містять текстову інформацію та спеціальні команди — HTML-код, який визначає, яким чином інформація відображається у вікні браузера. Вся графічна, аудіо та відео інформація не міститься безпосередньо на веб-сторінці, а є окремими файлами із відповідним розширенням. HTML-код сторінки містить лише посилання на відповідні файли. Крім того, сторінка вебсайту також має свою унікальну адресу, яка складається з адреси сайту та імені файлу, котрий відповідає цій сторінці. Отже, підсумовуючи можна дати наступне загальне визначення:

Вебсайт – це ресурс, який відображає певну інформацію містить що складається з взаємопов’язаних гіпертекстових документів, розміщених на веб-серверах з індивідуальними адресами.

1.2 Класифікація вебсайтів

Головним критерієм для класифікації вебсайтів сьогодні є їх поділ за функціональністю. Тому прийнято виділяти наступні види:

- Сайт візитка.
- Корпоративний сайт.
- Інтернет-магазин.
- Односторінковий сайт.
- Маркетплейс.
- Інші[2].

Поговоримо про особливості кожного з цих видів.

Сайт візитка – це сайт, головна мета якого дати можливість його авторові лаконічно розповісти інформацію про себе, свій проект, бізнес, компанію тощо, презентувати себе оточуючим.

Корпоративний сайт – це сайт, призначений для повноцінної презентації компанії в інтернеті. Також він надає й інші можливості, зокрема продаж послуг чи товарів, які виготовляє компанія, збільшення бази клієнтів та збільшення штату співробітників. Переважно використовується компаніями великого, середнього і малого бізнесу.

Інтернет-магазин – це сайт, призначений відповідно для продажу товару чи послуги онлайн, надання консультацій щодо продукту, який пропонується компанією.

Односторінковий сайт – сайт, призначений для презентації користувачам лише якоїсь однієї пропозиції, зокрема певного розіграшу, послуги, тощо.

Маркетплейс – сайт, котрий надає можливість користувачеві серед великого розмаїття товарів чи послуг представлений на подібному сайті обрати продукт та продавця, які необхідні користувачеві.

Інші – до цієї категорії відносять усі інші типи сайтів, які існують в сучасному світі. Серед них також можна виділити певні підкатегорії. Наприклад сайти відеохостинги, де відповідно представлені різноманітні відео, інформаційні сайти,

де користувач може отримати та обговорити інформацію, поштові сервіси, пошукові системи, онлайн-енциклопедії і, звісно, соціальні мережі.

РОЗДІЛ 2 ОСНОВНІ МЕТОДИ ТА ТЕХНОЛОГІЇ ДЛЯ РОЗРОБКИ ВЕБСАЙТІВ

2.1 HTML

HTML, або ж мова розмітки гіпертексту — стандартизована мова розмітки документів для перегляду веб-сторінок у браузері[3]. Тобто це мова, яка описує структуру сторінок документа з розширенням .html та дозволяє виконувати форматування, зокрема перетворення звичайного тексту в абзаци, списки, заголовки тощо. Досягається це за допомогою відповідних інструкцій форматування, які мають назву теги. Вони вбудовуються в частини документу та повідомляють браузеру яким чином представити інформацію на екрані. Загалом, тег – це елемент мови розмітки гіпертексту, назва якого розташована у кутових дужках(<назва>). Теги можна поділити на наступні групи:

- Теги верхнього рівня.
- Теги заголовка документа.
- Блокові елементи.
- Рядкові елементи.
- Списки.
- Таблиці[4].

Теги верхнього рівня використовуються для задання структури документу. До них належать наступні теги:

- <html></html>, всередині цього тегу розміщується весь вміст вебсторінки.
- <head></head>, містить елементи, мета яких полегшити браузеру роботу з даними. Також, тут можна розташувати метатеги, які призначаються для пошукових систем, адже містять опис сайту, його ключові слова. Інформація з цього тегу не відображається безпосередньо на вебсторінці.
- <body></body>, містить зміст веб-сторінки, який буде відображатися на ній. Це можуть бути текст, таблиці, зображення, тощо.

Для того щоб браузер зрозумів, як інтерпретувати поточний документ, на початку HTML-документа треба вказати <! DOCTYPE html>. Тобто, загальна структура документу HTML складається з трьох частин:

- Декларація типу документа.
- Шапка документу, яка знаходиться в тегах<head></head>.
- Тіло документа, яке знаходиться в тегах <body></body>[3].

Теги заголовка документа – це теги, які розташовуються в <head></head>. Теги цієї групи не відображаються на вебсторінці, окрім тегу <title>, який встановлює назву відповідної сторінки.

2.2 CSS

Абревіатура CSS розшифровується як Cascading Style Sheets, що в перекладі означає «каскадні таблиці стилів». Це спеціальна мова розмітки, яка використовується для візуального дизайну сайтів.

Об'єкти, що знаходяться на сторінці, розташовуються на ній за допомогою HTML. CSS відповідає за те, як ці об'єкти виглядають, зокрема їх розмір, колір, фонове зображення, рівень прозорості, розташування порівняно з іншими складовими, поведінка при наведенні курсору.

Каскадні таблиці стилів мають наступні переваги їх використання:

- Повторне використання. У розробника є можливість написати один файл з необхідними йому стилями та використовувати його для різних вебсторінок.
- Сумісність. Каскадні таблиці дозволяють оптимізувати вміст сторінки для декількох пристроїв. Використовуючи, наприклад, медіа-запити, вебсторінка має можливість коректно відображатись як на мобільних телефонах, чи планшетах, так і на екранах комп’ютерів.
- Простота в обслуговуванні. Для внесення глобальних змін, необхідно змінити стиль у файлі зі стилями. Всі елементи на вебсторінках оновляться автоматично.

Ще одним приємним доповненням каскадних таблиць стилів є простий та зрозумілий синтаксис. В його основі покладено два показники – властивість(property), яка буде застосована до елементу, та її значення(value). Ця пара властивість-значення називається декларацією(declaration) CSS та розділяється двокрапкою. Вона має наступний вигляд(рисунок 2.1):

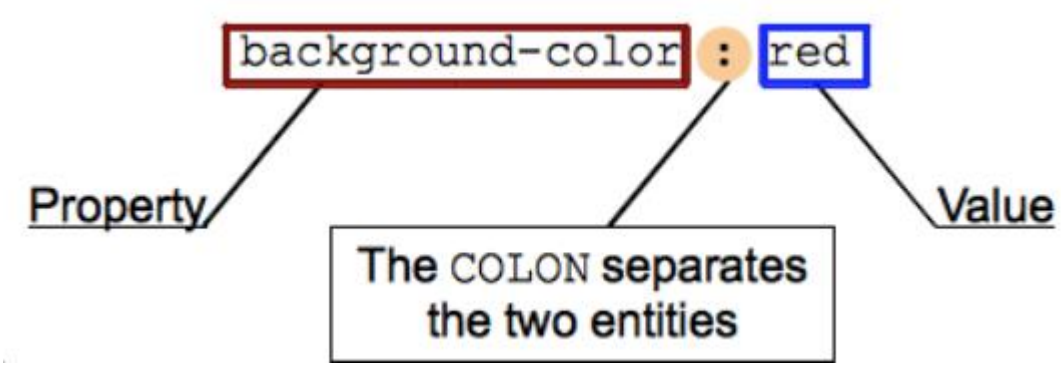

Рисунок 2.1 – Декларація CSS

Для кожного елемента може існувати більш як одна властивість, яку розробник матиме бажання змінити. Відповідно до цього, декларації групуються в блоки. Вони називаються блоками декларацій. Їх структура складається з фігурних дужок та списку декларацій, розмежованих крапкою з комою. Тобто, блоки декларацій виглядають наступним чином(рисунок 2.2):

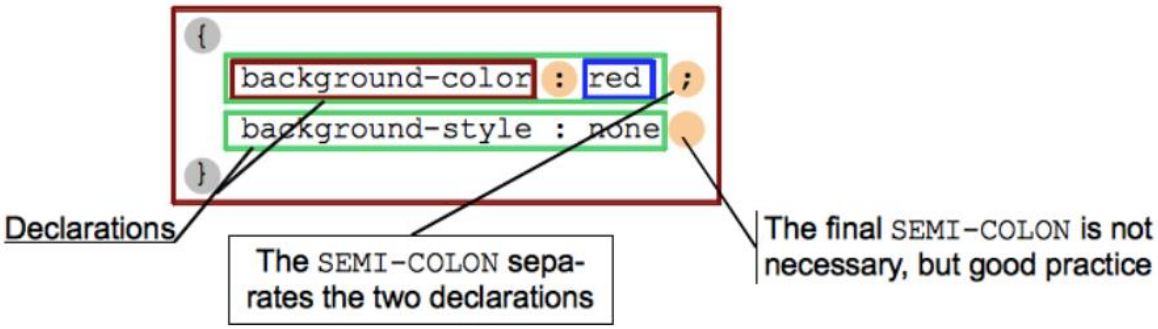

Рисунок 2.2 – Блок декларацій CSS

Усі блоки декларацій записуються до файлу з розширенням .css.

Додавання CSS до документу. Для того, щоб повідомити HTML-документу, що існують певні правила CSS, які ми хочемо застосувати існує три різні способи. Найбільш правильним і використовуваним способом зробити це є зв’язка CSS з заголовком HTML-документа. Для цього створюється окремий файл з розширенням .css. Цей файл прийнято називати «styles.css» та зберігати у тій же директорії, що й HTML-документ. Для того щоб зв’язати «styles.css» з «index.html» додається наступний рядок всередині тегу <head> відповідного HTML-документу: <link rel="stylesheet" href="назва.css" type="text/css">

Тег <link> повідомляє браузеру, існує таблиця стилів, використовуючи атрибут rel, і розташування цієї таблиці стилів як значення атрибуту href.

2.3 JavaScript

JavaScript - об'єктно-орієнтована скриптова мова програмування, яка надає можливість вебсторінці реагувати на дії користувача, тобто робити її інтерактивною. JavaScript також називають клієнтською мовою програмування, адже він працює на боці клієнта, тобто на пристрої кінцевого користувача.

Програми, написані цією мовою називаються скриптами. Існує можливість додати їх безпосередньо у HTML, або вказати посилання на файл з програмою. Таким чином скрипти будуть виконуватися автоматично під час завантаження вебсторінки, адже вони розповсюджуються і виконуються як простий текст, тому їм не потрібна спеціальна підготовка чи компіляція для запуску.

На сьогоднішній день JavaScript є однією з найпопулярнішою мовою програмування, адже він має дуже суттєві переваги:

- Незамінний у веброзробці. Скрипти підтримують усі популярні браузери, повна інтеграція із HTML та CSS.
- Швидкість та продуктивність.
- Взаємодіяти з програмою можна за допомогою звичайного текстового редактору.
- Має велику кількість фреймворків та бібліотек.
- Простота.

Звичайно, він має і певні недоліки, а саме:

- Відсутність можливості читати та завантажувати файли.
- Знижений рівень безпеки. Існує можливість впроваджувати в скрипти фрагменти шкідливий код.
- Відсутність строгої типізації.
- Неможливе використання в мережевих програмах через відсутність підтримки віддаленого доступу.

Структура JavaScript. В середовищі браузера JavaScript скаладається з наступних структурних одиниць:

- Ядро.
- BOM.

- DOM[10].

Ядро також прийнято називати ECMAScript. Воно являється основним компонентом для функціонування інших частин, адже воно описує синтаксис мови програмування, тобто визначає основні зарезервовані слова, задає правила опису функцій, циклів, тощо.

Об'єктна модель браузера призначена для управління поведінкою браузера використовуючи JavaScript.

Об'єктна модель документа призначена для надання доступу JavaScript до HTML-документа з подальшою можливістю динамічно змінювати зміст.

Структура коду. JavaScript програма, або ж скрипт складається з інструкцій. Інструкціями називають синтаксичні конструкції та команди, які виконують дії[11]. Кожна інструкція зазвичай пишеться з нового рядка та відокремлюється крапкою з комою.

Змінні. Для оголошення змінних використовується ключові слова let або const. У першому випадку значення змінної можна буде змінити, у другому ж випадку ні, адже таким чином оголошується константа. Щоб встановити значання змінній використовується оператор присвоювання «=». Існують наступні обмеження щодо можливих назв змінних:

- Ім’я має містити лише букви, цифри або символи «_» та «$».
- Ім’я не може починатись з числом.
- Ім’я не може співпадати з зарезервованими словами.

Типи даних. У мові програмування JavaScript розрізняють вісім основних типів даних:

- Number. Відображує цілі числа та з плаваючою крапкою. Містить спеціальні числові значення, які позначаються «Infinity» та «NaN» і означають математичну нескінченність та помилку обчислення відповідно.
- String. Відображує рядки. За правилами мови має бути оточений лапками. Допускаються одинарні, подвійні на зворотні лапки. Між першим і другим типом відсутня різниця у використанні. Останні ж

розширюють функціональність надаючи змогу додавати змінні та вирази у рядок використовуючи спеціальну конструкцію.

- Boolean. Булевий тип що зберігає лише два значення «true» або «false».
- Null. Це спеціальне значення, яке формує свій власний тип з відповідною назвою. Воно означає «нічого» або ж «порожнечу».
- Undefined. Це також спеціальне значення, яке формує свій власний тип. Воно означає, що «значення не присвоєно».
- Object. Зберігає більш складні структури, може містити збірки даних.
- Symbol. Використовується для створення унікальних ідентифікаторів в об’єктах[11].
- BigInt. Використовується для відображень великих чисел, які більші за діапазон допустимих значень типу Number.

2.4 Протокол HTTP/ HTTPS

Протокол – це система правил, які визначають спосіб обміну даними всередині або між комп’ютерами. Для зв’язку між пристроями потрібно, щоб пристрої узгодили формат даних, якими обмінюються. Набір правил, який визначає формат, називається протоколом.

HTTP – це протокол передачі даних в інтернеті[14]. Він розшифровується як HyperText Transfer Protocol, що в перекладі означає «протокол передачі гіпертекстових документів». Головне призначення цього протоколу полягає у передачі вебсторінок. Це досягається завдяки взаємодії клієнта − локального комп'ютера з браузером, який робить запит(request), та сервера − високопродуктивного спеціального комп'ютера, який надає відповідь(response)(рисунок 2.3):

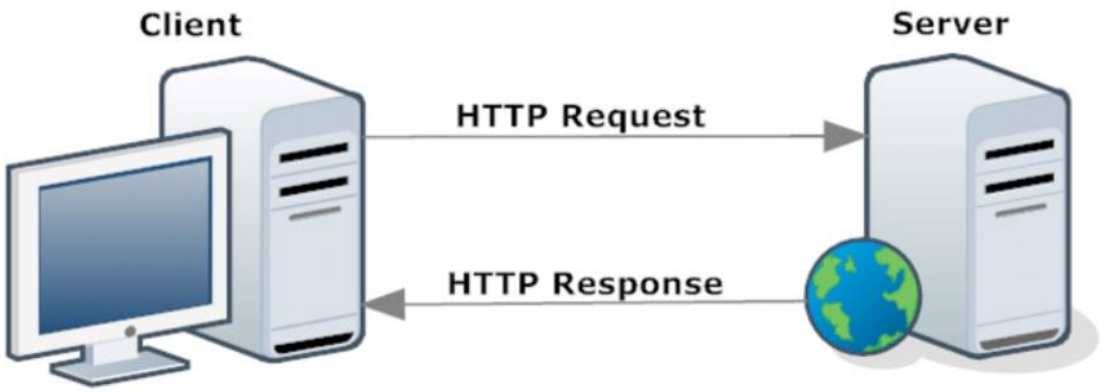

Рисунок 2.3 − Схема HTTP-запитів/відповідей

Структура протоколу. Кожні HTTP-запит чи HTTP-відповідь складаються з трьох частин:

- Стартовий рядок. Рядок, що задає параметри запиту чи відповіді.
- Заголовки. Містять інформацію для браузера.
- Тіло повідомлення. Містить дані, що передаються.

Структура запиту. Найважливішим елементом структури запиту є стартовий рядок. Він має наступний вигляд: ‹Метод› ‹URL› HTTP/‹Версія›.

Метод − визначає, яку саме дію потрібно зробити зі сторінкою. Розрізняють наступні основні методи:

- GET. Дозволяє отримати дані з серверу.
- POST. Дозволяє надіслати дані на сервер.
- DELETE. Дозволяє видалити дані з серверу.
- PATCH. Дозволяє замінити частину даних на сервері.

URL − ідентифікує ресурс та визначає його точне місцезнаходження.

Версія − визначає версію протоколу, яка буде використана у відповіді сервера.

Структура відповіді. Загалом, структура відповіді будується майже аналогічно структури запиту, проте відмінності все ж присутні. Стартовий рядок відповіді виглядає наступним чином: HTTP/‹Версія› ‹Код статусу› ‹Опис статусу›.

Версія – співпадає з версією запиту.

Код статусу – демонструє статус запиту. Це певне трицифрове число, яке повідомляє клієнту інформацію про отримання, обробку запиту, або про помилки, котрі виникли. Коди статусу поділяються наступним чином:

- 1xx. Інформаційний.
- 2xx. Успішний.
- 3xx. Свідчить про перенаправлення.
- 4xx. Повідомляє про помилку клієнта.
- 5xx. Повідомляє про помилку сервера[14].

У списку вище замість «x» підставляються цифри від 0 до 9 для отримання тризначних кодів.

Опис статусу – містить короткий опис відповіді.

Найбільш поширеними парами «код статусу»-«опис статусу» є

- 200 OK, означає що запит виконано успішно.
- 404 Not Found, означає що сервер не може віднайти те, що потребує в запиті клієнт.

– 403 Forbidden, означає що сервер не може виконати запит через обмеження пов’язаних з доступом.

Різниця протоколів HTTP та HTTPS. Протокол HTTPS – це розширення протоколу HTTP. Головна відмінність полягає у тому, що HTTPS забезпечує захист інформації, яка передається шляхом її шифрування на клієнті за допомогою SSL сертифікату(рисунок 2.4).

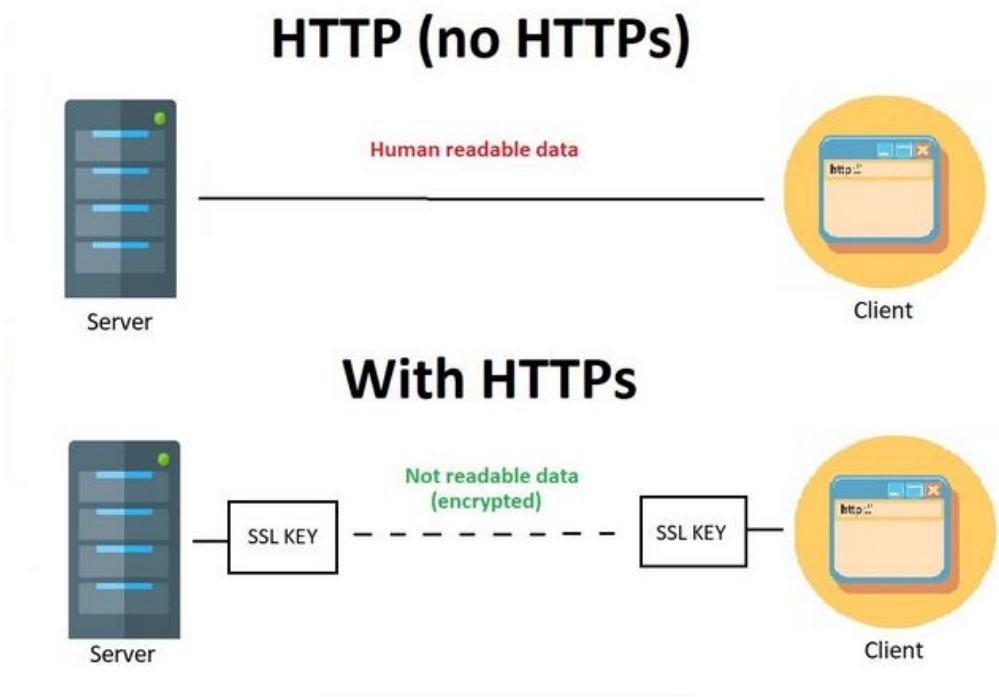

Рисунок 2.4 – Відмінність HTTP та HTTPS

2.5 REST API

Означення API. API, або ж прикладний програмний інтерфейс – це спеціальний протокол для взаємодії комп'ютерних програм, який дозволяє застосувати функції одного додатка всередині іншого. Використання API доволі різноманітне, наприклад:

- Бібліотеки та фреймворки. API описує та визначає правила, коли бібліотека є фактичною реалізацією цих правил.
- Операційні системи. API описує інтерфейс між програмою та операційною системою.
- Remote API. Цей тип API дозволяє розробникам керувати віддаленими ресурсами, використовуючи протоколи та певні стандарти зв’язку, які надають можливість різним технологіям працювати разом незважаючи на різні мови програмування чи платформи.
- Web API. Це API, доступ до якого здійснюється від клієнта до сервера за допомогою протоколу HTTP.

REST API. REST – це підхід до архітектури мережевих протоколів, які надають доступ до інформаційних ресурсів[18]. REST API − це API, який використовує HTTP-запити для отримання, вилучення, розміщення та видалення даних. Тобто, ця технологія використовується там, де користувачеві необхідно оперувати інформацією з сервера.

Принципи REST API. Були визначені наступні шість основних принципів REST API:

- Єдиний інтерфейс. Основна ідея цього принципу полягає у тому, що дані повинні отримуватись за допомогою однієї URL-адреси і лише за допомогою базових методів стандартного мережевого протоколу HTTP (GET, PUT, DELETE, POST).
- Розділення клієнту та серверу. Принцип має на увазі, що весь інтерфейс користувача має бути реалізований на боці клієнта, а доступ та безпека даних на боці серверу.

- Кешування. В основі принципу закладено що всі дані мають мати змогу кешуватися, якщо явно не вказано зворотнє.
- Багаторівневість системи. Допускається архітектура, яка складається з декількох рівнів серверів, однак кожен сервер повинен взаємодіяти лише з найближчими рівнями.
- Відсутність збереження стану. Усі клієнт-серверні операції мають бути без збереження стану. Кожен запит клієнта повинен містити лише ту інформацію, яка необхідна для отримання даних від сервера.
- Надання коду за запитом. За необхідності сервер може надсилати виконуваний код, наприклад для запуску відео, безпосередньо клієнту.

Архітектура REST API. Як було сказано вище, REST API заснований на основі протоколу передачі гіпертексту HTTP, кожен об’єкт на сервері має свою унікальну URL-адресу. У REST API для взаємодії з даними на сервері використовують наступні HTTP методи:

- GET, для отримання даних.
- POST, для додавання чи заміни даних.
- PUT, для оновлення даних.
- DETETE, для видалення даних.

Ці запити дозволяють реалізувати стандартний набір дій з даними, який ще називають CRUD: Create(Створити), Read(Прочитати), Update(Оновити), Delete(Видалити).

Загальна модель REST API має наступний вигляд(рисунок 2.5):

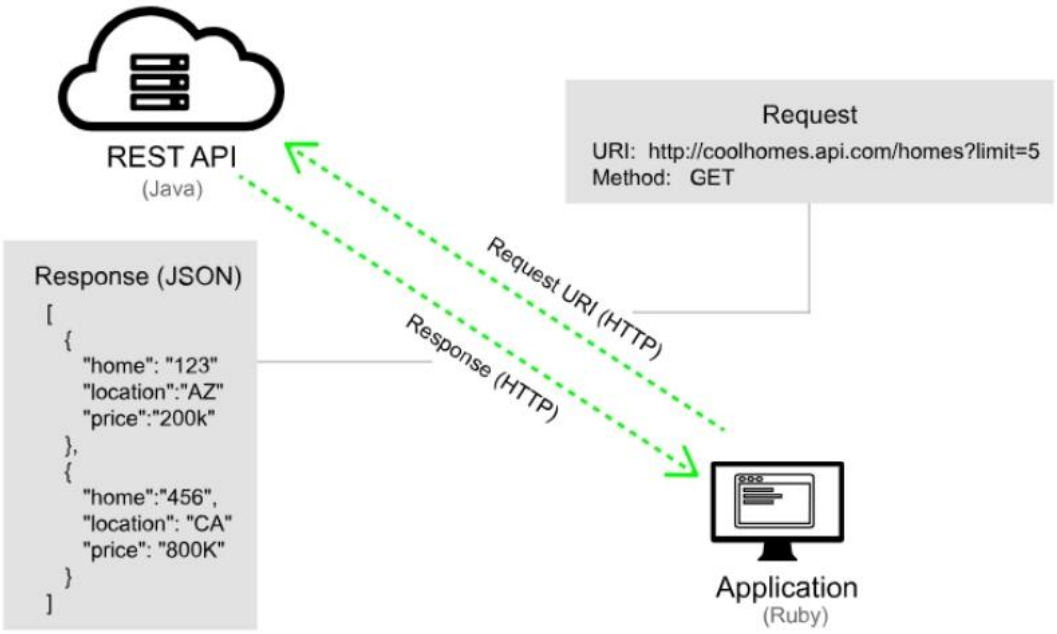

Рисунок 2.5 – Модель REST API

Між клієнтом та сервером існують HTTP-запити та відповіді. Завдяки тому що використовується протокол HTTP, клієнт та сервер можуть бути реалізовані різними мовами програмування.

РОЗДІЛ 3 РОЗРОБКА ВЕБСАЙТУ ДЛЯ АЛГОРИТМІЧНОГО СЕРЕДОВИЩА

3.1 Вимоги

Головним завданням перед автором постало створення зручного вебсайту для демонстрації оцінювання задач, які є складовими алгоритмічного середовища. Перед розробкою були сформовані наступні вимоги:

- Розроблений за допомогою сучасних технологій.
- Сайт повинен мати легкий для розуміння дизайн.
- Сайт повинен мати системи ідентифікації, аутентифікації та авторизації. В залежності від користувача реалізувати різні можливості на сайті.
- Сайт повинен використовувати бібліотеки CGLib та CGMark, які реалізують алгоритмічне середовище та систему оцінювання задач відповідно.

3.2 Вибір програмних засобів.

3.2.1 Бібліотека React

React — це декларативна, ефективна і гнучка JavaScript-бібліотека, призначена для створення інтерфейсів користувача. Вона дозволяє компонувати складні інтерфейси з невеликих окремих частин коду – компонентів[19]. Компоненти можуть бути визначені як класи чи функції. Життєвий цикл кожного React-компонента складається з 3-х етапів:

- Монтування.
- Оновлення.
- Демонтуавння.

Для кожного етапу існують свої методи життєвого циклу, якими розробник може користуватись для запуску певного коду в певний момент часу. Життєвий цикл продемонстрований на наступній діаграмі(рисунок 3.1):

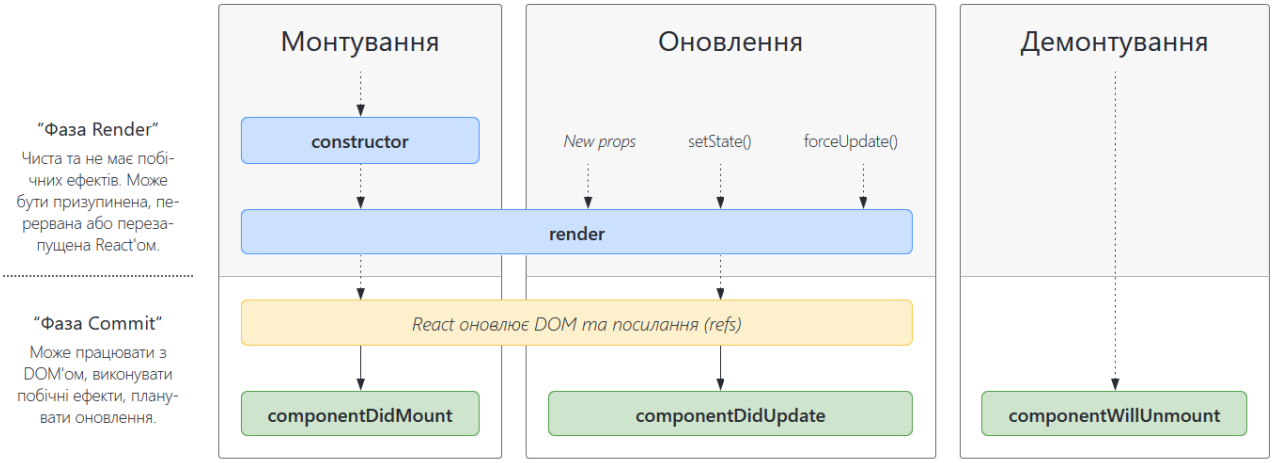

Рисунок 3.1 – Діаграма життєвого циклу

Особливості бібліотеки React. React має набір певних особливостей, які дозволяють йому бути гнучким та потужним інструментом розробки. Основні з них:

- Декларативність.
- Віртуальне DOM-дерево.
- Оновлення DOM частинами.
- Повторне використання компонентів.

- Спадний потік даних.
- JSX.
- React Hooks.

Розглянемо кожну особливість більш детально.

Під декларативністю мається на увазі декларативний принцип, який означає що розробнику достатньо один раз описати вигляд елементів у різних станах. React буде автоматично оновлювати стан цих елементів в залежності від умов.

Віртуальне DOM-дерево – це копія DOM, яку бібліотека створює і зберігає у кеші. React працює з віртуальною копією у кеші, яка важить менше, для того щоб швидко оновлювати стан вебсторінки, адже віртуальна структура оновлюється швидше ніж реальна через те, що реальна об'єктна модель може бути досить великою, тому її оновлення є повільним процесом. Тобто коли відбувається подія, наслідком якої є зміна стану об’єкту, то ця зміна швидко відображається в віртуальному DOM, після чого оновлюється реальний DOM.

Під оновленням DOM частинами мається на увазі що React оновлює DOM не повністю. Він зберігає в пам'яті дві легкі копії: актуальну та попередню. При оновленні, бібліотека порівнює ці версії між собою та вносить зміни лише в ту частину дерева, яка змінилася.

Повторне використання компонентів. React-компонент – це окремий елемент інтерфейсу, який містить у собі всі необхідні дані і методи, а також стан елементу. Завдяки інкапсуляції, або ж самостійності компонентів їх можна використовувати повторно, що в свою чергу пришвидшує розробку.

Спадний потік даних. React-компоненти можуть обмінюватися даними та властивостями між собою, але лише в напрямку від батьківських до дочірніх.

JSX − це розширення мови JavaScript, яке допомагає описувати HTML-подібні елементи за допомогою коду React[20]. Таким чином розробники створюють компоненти вебсторінки там мають змогу гнучко ними керувати.

React Hooks – це спеціальні функції, які зберігають стан чи метод елементу.

Переваги React. Бібліотека React є найпопулярнішою бібліотекою для розробки користувацького інтерфейсу, адже він має наступні переваги:

- Простота у створенні інтерфейсу. Завдяки React-компонентам можна швидко та легко розробити інтерфейс будь-якої складнощі.
- Реактивність. Бібліотека React реагує на оновлення елементу та автоматично відображає його зміни в DOM.
- Ефективність. Завдяки віртуальному DOM досягається значна економія ресурсів.
- Швидкість роботи. Досягається завдяки віртуальному DOM, котрий займає менше місця та швидше оновлюється.

Недоліки React. Бібліотека React не позбавлена недоліків, а саме:

- Складний синтаксис JSX. Через поєднання HTML та JavaScript на початкових етапах розробник може легко робити помилки.
- Складнощі з пошуковою оптимізацією. Оскільки React-компоненти написані мовою JavaScript, то пошуковому роботу їх індексація стає складнішою і повільнішою.
- Фокусування на інтерфейсі користувача.

3.2.2 Ant Design

Ant Design − це бібліотека компонентів інтерфейсу користувача React.

Ця бібліотека має наступні переваги:

- Написана мовою програмування TypeScript із передбачуваними статичними типами.
- Підтримка інтернаціоналізації для багатьох мов.
- Можливість детального налаштування теми.
- Доступна велика кількість дизайнерських ресурсів та засобів розробки.
- Набір вже розроблених якісних React-компонентів.

Однак, вона має певні недоліки:

- Постійне використання !important в CSS коді для відміни стилізацій компонентів, які не відносяться до цієї бібліотеки.
- Знижується продуктивність через величину бібліотеки.

3.2.3 Фреймворк Materialize

Materialize − це CSS фреймворк для створення сайтів, що базується на принципах material design. [23]. Тобто це певний набір необхідних компонентів, виконаних з дотриманням усіх норм сучасного дизайну сайтів.

Material design – це концепція дизайну, яка створена компанією Google для уніфікації сервісів. Тобто це певна система правил, яка дозволяє створювати єдиний інтерфейс для всіх пристроїв.

Принципи material design. Існують 9 принципів material design, однак з них прийнято виділяти наступні:

- Тактильні поверхні. Принцип має на увазі, що всі елементи інтерфейсу – це шари цифрового паперу, котрі розташовані на різних висотах та залишають за собою тінь. Цей принцип допомагає користувачеві виділяти основні елементи інтерфейсу, робить інтерфейс інтуїтивно зрозумілим.
- Поліграфічний дизайн. В основі принципу лежить ідея, що вся інформація, яка відображається на шарах, задовольняє закони друкованого дизайну. Цей принцип дозволяє акцентувати увагу користувача на необхідному елементі інтерфейсу.
- Усвідомлена анімація. Головна ідея цього принципи полягає у тому, що елементи, які відображаються на екрані не повинні просто так зникати та з’являтись. Натомість, вони повинні плавно переходити один в одного.
- Адаптивний дизайн. Цей принцип говорить нам про те, що всі попередні принципи повинні виконуватися для всіх пристроїв.

Анімація в material design. Анімація в material design це одна з його основ, яка має на меті зробити інтерфейс користувача яскравим та простим у використанні. Для цього анімація має відповідати наступним принципам:

- Інформативність. Анімація має показувати користувачу зв’язки між елементами інтерфейсу. Тобто користувач має розуміти, які дії йому доступні та їх наслідки.

- Виразність. Анімація повинна надавати унікальність та певний стиль кожному продукту.
- Орієнтованість. Анімація має акцентувати увагу лише на важливому та не відволікати від основної діяльності.

3.2.4 Django REST Framework

Django − безкоштовний і відкритий фреймворк для створення веб додатків, написаний мовою програмування Python[29].

Django REST Framework – це фреймворк для Django, який працює зі стандартними моделями Django для створення гнучкого та потужного API для проекту та підтримує ідеологію REST. Тобто, використання цього фреймворку дозволяє стандартизувати запити до бази даних та одночасно створювати REST API нашого сайту.

Архітектура DRF. API цього фреймворку складається з 3-х частин:

- Серіалізатор, котрий призначений для перетворення інформації, яка зберігається в базі даних, у певний формат для подальшої її передачі через API.
- Вид(ViewSet), визначає функції для отримання, редагування, додавання та видалення даних, які будуть доступні через API
- Маршрутизатор, визначає URL-адреси, які надаватимуть доступ до кожного виду[28].

Розглянемо кожну складову більш детально.

Серіалізатор. Серіалізатор Django REST Framework перетворює дані, які зберігаються у базі даних за допомогою складної структури − моделей Django − до більш простого формату – JSON, який використовується для передачі інформації через API. Таким чином, коли користувач надсилає дані через API, то серіалізатор перевіряє та перетворює їх у екземпляр моделі Django, а коли користувач робить запит на отримання даних, то серіалізатор перетворює їх до формату JSON.

Вид(ViewSet). На відміну від серіалізатора, котрий аналізує інформацію лише в двох напрямках: читання та запис, ViewSet визначає доступні операції. Найбільш поширеним ViewSet є ModelViewSet, який має наступні операції:

- create(), функція створення екземпляру.
- retrieve(), функція отримання екземпляру.
- update() та update_all(), функції оновлення одного екземпляру та екземплярів.

- destroy(), функція видалення екземпляру.
- list(), функція отримання списку екземплярів[28].

Маршрутизатор. Маршрутизатор – це представник верхнього рівня API. Вони об’єднують всі URL-адреси, котрі необхідні для ViewSet в один рядок.

3.3 Загальна структура сайту

Загальна структура сайту матиме наступні елементи:

- Головна сторінка, де відображається основна інформація про сайт та можливості для авторизації чи реєстрації користувача.
- Сторінка авторизації, де відображаються необхідні поля, які має заповнити користувач для успішної авторизації.
- Сторінка реєстрації, яка містить необхідні поля, які має заповнити користувач для успішної реєстрації.
- Сторінка студента, де відображається список задач, необхідних для виконання.
- Сторінка задачі, де відображаються необхідні поля, які має заповнити студент для отримання оцінки за виконання певної задачі.
- Сторінка викладача, яка містить список груп студентів та можливості для їх видалення чи створення.
- Сторінка групи. На цій сторінці викладач може задавати задачу, яку необхідно виконати для відповідної групи та додавати чи видаляти студентів до певної групи.

3.4 Розробка інтерфейсу

Відповідно до розробленої структури була спроектована головна сторінка сайту. Вона містить інформацію про назву сайту, стисло описує його призначення та пропонує пройти на сторінку авторизації чи реєстрації для розширення можливостей. Перехід можна здійснити за допомогою кнопок «Авторизуватися» та «Зареєструватися» відповідно.

На сторінці авторизації користувачу необхідно ввести ім’я користувача та пароль в формі авторизації. Якщо дані введені вірно, користувач може натиснути кнопку «Увійти» та авторизуватися у системі, інакше треба перевірити коректність введених даних та спробувати ще раз.

На сторінці реєстрації користувачеві необхідно заповнити наступні поля в формі реєстрації:

- Електронна адреса.
- Ім’я користувача.
- Пароль.

Якщо введені дані відповідають певним правилам валідації, то користувач може натиснути кнопку «Реєстрація» та бути зареєстрованим у системі.

У разі успішного виконання попередніх процедур користувача буде перенаправлено на відповідні сторінки: якщо це студент – на сторінку студента, якщо викладач – на сторінку викладача.

На сторінці студента відображено список задач, які йому назначені для виконання викладачем. При натисканні на назву задачі він потрапляє на сторінку задачі.

Сторінка задачі містить форму з певними полями, які необхідно заповнити студенту для того щоб система зробила перевірку його відповідей. Відповідно до відповідей студент отримає в результаті свою оцінку за виконання цієї задачі.

На сторінці викладача відображається список груп студентів. Він може їх видаляти чи додавати нові за допомогою відповідних кнопок та форм. Для того щоб редагувати вміст груп викладач має натиснути кнопку «Перейти», яка знаходиться

біля назви групи. Таким чином він буде переправлений на сторінку відповідної групи.

Сторінка групи містить список студентів цієї групи та відповідний функціонал, який надає можливість викладачу має додавати студентів до групи, обираючи їх зі списку зареєстрованих користувачів, чи видаляти їх з групи, натискаючи на відповідну кнопку біля імені користувача, додавати чи видаляти для виконання певну задачу відповідній групі.

У будь-який момент часу користувач може вийти зі свого аккаунту, натиснувши кнопку «Вийти», що призведе до його перенаправлення на головну сторінку

ВИСНОВКИ

В ході виконання кваліфікаційної роботи на здобуття ступеня бакалавра було розроблено та створено вебсайт для алгоритмічного середовища. Даний сайт орієнтований для інтегрування його в навчальний процес з дисципліни «Обчислювальна геометрія та комп'ютерна графіка». З його допомогою користувачі зможуть отримувати швидке оцінювання задач, які розглядаються в цьому курсі.

При розробці вебсайта були проаналізовані сучасні методи та технології для розробки вебсайтів. Найкращими для виконання поставленого завдання виявилися бібліотека мови JavaScript React та її бібліотеки.

Розроблений сайт задовольняє всім вимогам, однак має обмежену функціональність. Зокрема, оцінювання студента працює лише для одного типу задач, а саме для побудови опуклої оболонки методом Грехема.

Подальший розвиток сайту полягає у додаванні реалізації оцінювання інших задач, представлених у алгоритмічному середовищі. Крім того, у далекій перспективі має місце розширення продукту не тільки для задач представлених у алгоритмічному середовищі, але й у інших дисциплінах, які мають місце у навчальному процесі.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Матеріал з Вікіпедії — вільної енциклопедії про вебсайт. Доступ до сайту:https://en.wikipedia.org/wiki/Website
2. Види сайтів та їх функціонал. Доступ до сайту: https://webtune.com.ua/statti/web-rozrobka/vydy-sajtiv-ta-yih-funkczional/
3. Матеріал з Вікіпедії — вільної енциклопедії про HTML. Доступ до сайту:https://uk.wikipedia.org/wiki/HTML
4. Типи HTML тегів. Доступ до сайту: https://html-css.co.ua/self-html/tipi-tegiv/
5. Матеріал з Вікіпедії — вільної енциклопедії про CSS. Доступ до сайту:https://uk.wikipedia.org/wiki/CSS
6. Матеріал з WebDoky — документації для веброзробників про CSS. Доступ до сайту: https://webdoky.org/uk/docs/Web/CSS/
7. Матеріал з MDN Web Docs — документації для веброзробників про CSS. Доступ до сайту: https://developer.mozilla.org/en-US/docs/Web/CSS/Syntax
8. Матеріал з Вікіпедії — вільної енциклопедії про JavaScript. Доступ до сайту: https://uk.wikipedia.org/wiki/JavaScript
9. Матеріал з MDN Web Docs — документації для веброзробників про JavaScript. Доступ до сайту:https://developer.mozilla.org/en-US/docs/Learn/JavaScript
10. Структура JavaScript. Доступ до сайту:https://www.wisdomweb.ru/JS/struct.php
11. Сучасний підручник з JavaScript. Доступ до сайту:https://uk.javascript.info
12. Матеріал з Вікіпедії — вільної енциклопедії про комунікаційний протокол. Доступ до сайту: https://en.wikipedia.org/wiki/Communication_protocol
13. Матеріал з блогу SkillFactory — про HTTP. Доступ до сайту:https://blog.skillfactory.ru/glossary/http/
14. Матеріал з Вікіпедії — вільної енциклопедії про HTTP. Доступ до сайту:https://uk.wikipedia.org/wiki/HTTP
15. Матеріал з Вікіпедії — вільної енциклопедії про API. Доступ до сайту:https://en.wikipedia.org/wiki/API

16. Матеріал з блогу SkillFactory — про REST API. Доступ до сайту:https://blog.skillfactory.ru/glossary/rest-api/
17. Курс з документування API — про REST API. Доступ до сайту:https://starkovden.github.io/what-is-rest-api.html
18. Матеріал з Вікіпедії — вільної енциклопедії про REST. Доступ до сайту:https://uk.wikipedia.org/wiki/REST
19. Документація бібліотеки REACT. Доступ до сайту:https://uk.reactjs.org/
20. Матеріал з блогу SkillFactory — про REACT. Доступ до сайту:https://blog.skillfactory.ru/glossary/react/
21. Документація бібліотеки Ant Design. Доступ до сайту:https://ant.design/docs/react/introduce
22. Документація фреймворку Materialize. Доступ до сайту:https://materializecss.com/about.html
23. CSS фреймворк Materialize. Доступ до сайту:https://myrusakov.ru/css-materialize.html
24. Матеріал з блогу Arrivo Media — про принципи material design. Доступ до сайту:https://blog.arrivomedia.ru/marketing/filosofiya-google-material-design/
25. Матеріал з блогу pollskill — про основні принципи material design. Доступ до сайту:https://pllsll.com/blog/62
26. Документація фреймворку DRF. Доступ до сайту:https://www.django-rest-framework.org/
27. Матеріал з блогу AZOFT — про DRF. Доступ до сайту:https://www.azoft.ru/blog/django-rest-framework/
28. Матеріал з блогу Django.fun — про DRF. Доступ до сайту:https://django.fun/tutorials/kratko-o-django-rest-framework/
29. Матеріал з блогу DjangoGirls — про DRF. Доступ до сайту:https://tutorial.djangogirls.org/uk/django/