МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХЕРСОНСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
Факультет комп'ютерних наук, фізики та математики
Кафедра комп'ютерних наук та програмної інженерії

РОЗРОБКА ТА ПРОЕКТУВАННЯ СЕРВІСУ МАТЕРІАЛЬНО-ТЕХНІЧНОЇ БАЗИ ХЕРСОНСЬКОГО ДЕРЖАВНОГО УНІВЕРСИТЕТУ

Кваліфікаційна робота
на здобуття ступеня вищої освіти «бакалавр»

Виконав: студент 4 курсу 441 групи
Спеціальності: 121 Інженерія програмного забезпечення
Освітньо-професійної програми: Інженерія програмного забезпечення
Жигун Є.С.
Керівник: Вінник М.О., кандидат педагогічних наук, доцент
Співкерівник: Співаковський О.В.
Рецензент: Щедролосьєв Д.Є., керівник центру компанії DataArt

Херсон – 2022

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

SSR – Server Side Rendering

E2E – End-to-End

UI – User Interface

SEO – Search Engine Optimization

СУБД – Система Керування Базами Даних

CRUD - Create, Read, Update, Delete

API - Application Programming Interface

ВСТУП

Актуальність

Херсонський державний університет – є багатопрофільним закладом вищої освіти зі своєю столітньою історією, заклад освіти надає класичну університетську освіту за різноманітними галузями знань і спеціальностями та є одним із культурно-освітніх центрів півдня України.[1]

У сучасному світі, для збільшення ефективної діяльності, необхідною умовою є раціональне використання наявних ресурсів. Для цього і створюють сервіси для більш зрозумілого відображення наявних ресурсів університету для подальшого планування. А так як Херсонський державний університет існує вже понад 100 років, для покращення роботи необхідно створити сервіс матеріально-технічної бази.

Мета кваліфікаційної роботи: спроектувати та розробити сервіс матеріально-технічної бази Херсонського державного університету

Об’єкт дослідження: матеріально-технічна база

Предметом дослідження є розробка сервісу матеріально-технічної бази

Завдання дослідження:

- Проектування сервісу
- Розробка сервісу
- Тестування

РОЗДІЛ 1. ТЕОРЕТИЧНИЙ ОГЛЯД

1.1. Що таке матеріально-технічна база?

Матеріально-технічна база – це сукупність усіх матеріальних умов, будівель, споруд, обладнання та інших цінностей.[2]

Інакше кажучи, матеріально-технічна база навчально закладу – це сукупність всього необхідного для проведення навчальної, методичної та наукових робіт.

Коли навчальний заклад невеликий, відобразити матеріально-технічну базу доволі просто, перечисливши що знаходиться, наприклад, у лабораторії та підкріпивши це фотографіями. В нашому випадку є багато різних аудиторій, і у кожної з них різне призначення.

Під час спроб пошуку схожих сервісів матеріально-технічної бази мені не вдалося знайти бажане, лише сторінки навчальних закладів с текстом та фотографіями.

Головна » Інститути » Навчально-науковий інститут менеджменту та психології » Загальна інформація » Матеріально-технічна база

МАТЕРІАЛЬНО-ТЕХНІЧНА БАЗА ІНСТИТУТУ

За роки свого існування Навчально-науковий інститут менеджменту та психології Державного вищого навчального закладу "Університет менеджменту освіти" має значні досягнення: створено необхідну матеріально-технічну базу для організації та забезпечення повноцінного навчально-виховного процесу, здійснення наукової діяльності, а також сприятливі умови для роботи працівників, викладачів та студентів.

Усі споруди капітально відремонтовані та відповідають санітарно-гігієнічним та інженерно-технічним нормативам. Освітній процес в Інституті проводиться в 3-х учбових корпусах, розташованих по вулиці Січових стрільців (Артема), 52А.

У розпорядженні Навчально-наукового інституту менеджменту та психології актова зала, де проводяться культурно-масові заходи. А також спеціальні кабінети та аудиторії для проведення лекційних та семінарських і практичних занять, загальною площею – 1229,53 кв. м., що цілком відповідає реальним потребам навчального закладу щодо організації повноцінного навчального процесу як для студентів денної, так і заочної форм навчання. Всі аудиторії обладнані сучасними меблями, устаткуванням, технічними засобами навчання тощо.

Всі приміщення інституту обладнані електромережами, системами протипожежної сигналізації.

Постійно працює студентська їдальня, в якій відвідувачі мають можливість скуштувати смачні і різноманітні страви, відсвяткувати урочисті події студентського життя.Матеріальна база навчального закладу дозволяє задовольняти соціально-побутові потреби студентів.

В одному з корпусів розташована бібліотека, яка має можливість забезпечити потребу читачів у сучасній навчальній, науковій та художній літературі. Читальний зал, розрахований на 60 посадочних місць, де знаходяться алфавітний та систематичний каталоги, комп'ютери з доступом до глобальної системи "Internet". Книжковий фонд

Рисунок 1.1 – Сторінка матеріально-технічної бази «Університету менеджменту освіти»

Тому буду розробляти щось нове, а нове – це завжди цікаво!

1.2. Призначення роботи

Перш за все планувалося розробити сервіс матеріально-технічної бази ХДУ з перевагою на допомогу у проведенні занять. Тому в ньому буде лише необхідна інформація для викладачів, але в майбутньому можна збільшити кількість інформації та функціонал.

Для початку, необхідно буде відобразити плани поверхів, щоб розмітити аудиторії для отримання інформації про них:

- Спеціалізація (комп'ютерна аудиторія, лекційна чи інша);
- яка група студентів займає за розкладом;
- температура приміщення;
- відображення самої аудиторії (3D модель або фотографія).

Далі необхідно додати функціонал «керування університетом». Спочатку це будуть прості CRUD операції над факультетами, так як необхідно «прив'язати» поверхи та групи студентів до факультету, а далі, за необхідності, можна розширити функціонал.

1.3. Цільові пристрої

Сервіс буде розроблено за допомогою web-технологій, що надає можливості використовувати його на будь-яких пристроях та розширити функціональність за допомогою API.

«**API** – це система інструментів і ресурсів додатку, яка дозволяє розробникам створювати програмні продукти, які взаємодіють з іншими сервісами.»[3]

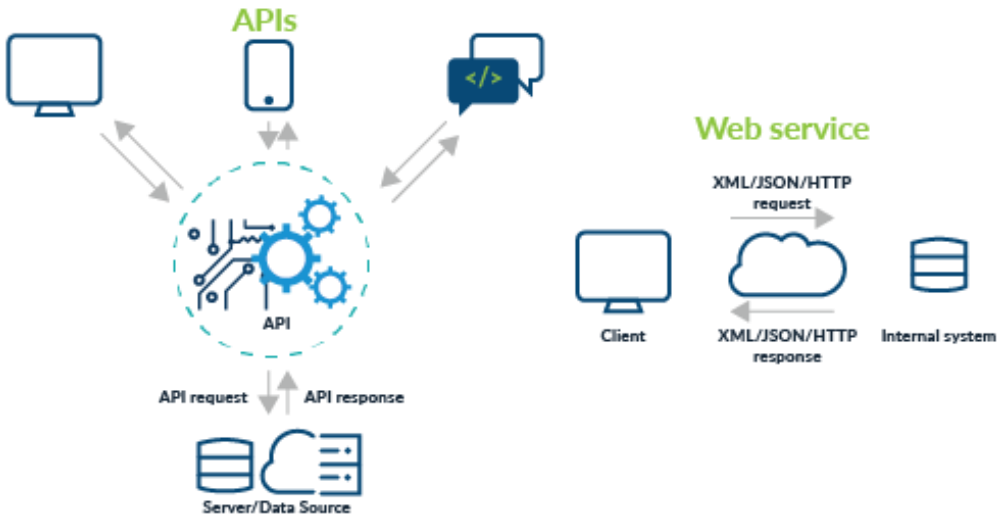

Рисунок 1.2 – Схема роботи з API

Які можливості надає розробка API? Наприклад, з’явиться необхідність створити мобільний додаток до цього сервісу, і для цього розробнику допоможе вже готовий API. Йому не знадобиться писати багато схожого коду, тому він зможе надсилати записи до нашого сервісу для отримання/зміни даних.

РОЗДІЛ 2. РОЗРОБКА СЕРВІСУ

2.1. Використані технології

Так як це буде онлайн сервісом, для його розробки необхідно створити Front-end та Back-end частини, в ідеалі навіть покрити тестами, щоб майбутні розробники могли відразу зрозуміти що відламалось.

Для проектування дизайну було обрану Figma, як один з найпопулярніших інструментів для дизайнерів.

Для Front-end було обрано:

- Next.js
- Storybook
- Material UI

Для Back-end:

- NestJS
- MongoDB

Також для тестування:

- Jest
- Cypress.io
- React Testing Library

2.1.1. Next.js

Рисунок 2.1 – Логотип Next.js

Next.js – це JavaScript фреймворк, заснований на React, який дозволяє створювати веб-додатки з покращеною продуктивністю та покращеним користувальницьким досвідом за допомогою додаткових функцій попереднього рендерингу, таких як повноцінний рендеринг на стороні сервера (SSR) та статична генерація сторінок (SSG).[4]

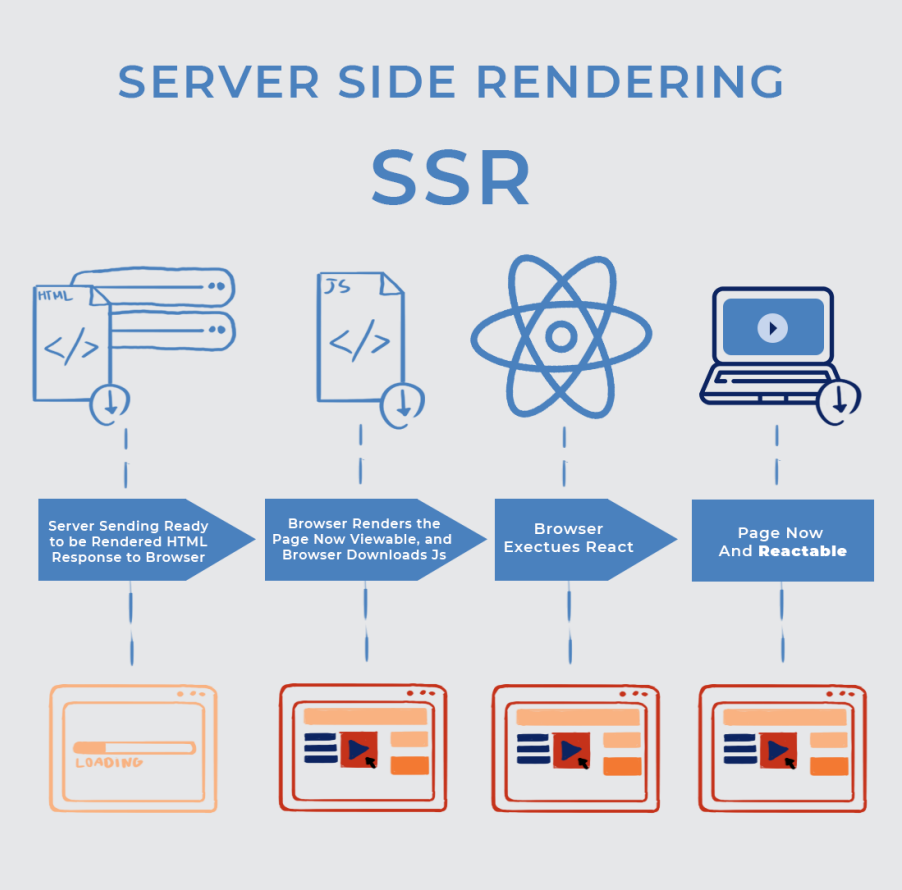

Рисунок 2.2 – Схема роботи SSR

Server Side Rendering (SSR) дозволяє отримати доступ до всіх необхідних даних для побудови сторінки на сервері. Потім сторінка повністю відправляється назад у браузер і відразу відображається. SSR дозволяє веб-сторінкам завантажуватися за менший час та підвищує зручність роботи користувачів за рахунок підвищення швидкості відгуку.[5]

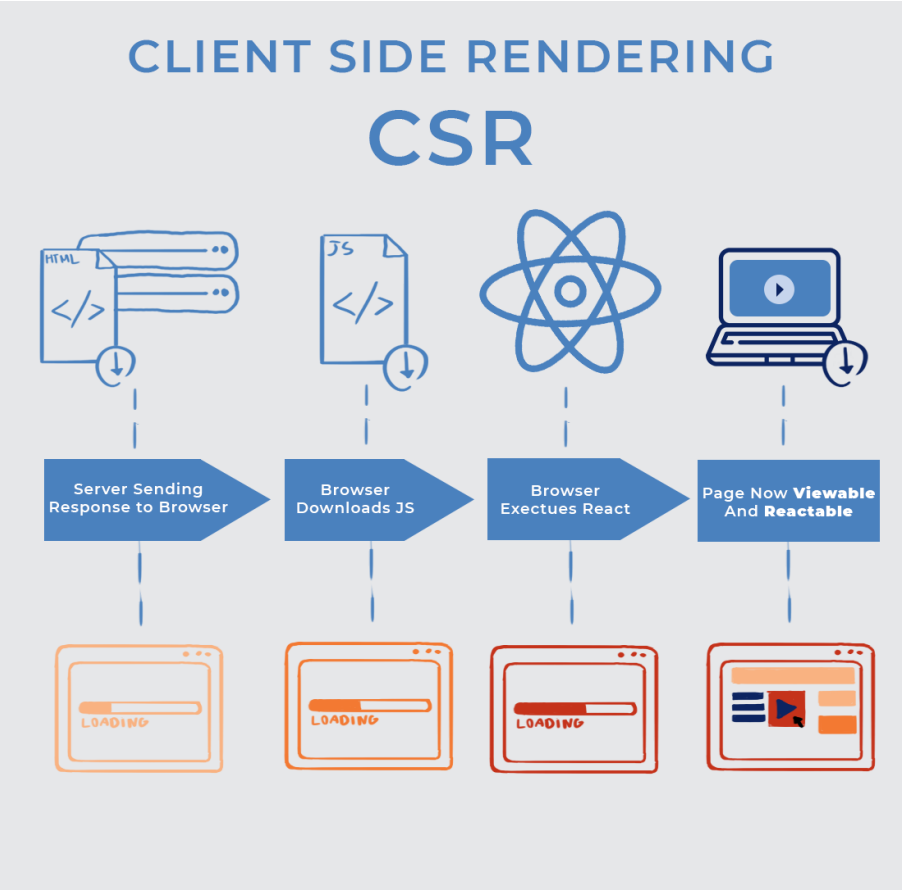

Рисунок 2.3 – Схема роботи CSR

Client Side Rendering (CSR) на відміну від SSR займає більше часу на початкове завантаження сторінки, оскільки потрібно завантажити весь JavaScript-файл.

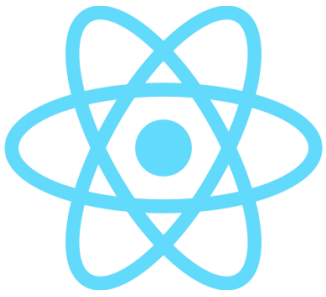

Рисунок 2.4 – Логотип React

React – це JavaScript-бібліотека, яка використовується для розробки UI.[6] Вона дозволяє компонувати складні інтерфейси з компонентів, невеликих частин коду. Ця бібліотека вирішує проблеми відображення при кожній зміні даних.

SEO – це процес вдосконалення сайту для пошукових систем та користувачів. Його мета – збільшити трафік на сайт для подальшої монетизації.

2.1.2. Storybook

Рисунок 2.5 – Логотип Storybook

Storybook – це інструмент для розробки UI, який полегшує розробку ізольованих компонентів та вести документацію по їх використанню.[7] Після створення компонента Storybook дає змогу створити файл «історії», куди можна імпортувати компонент і створити різні приклади його використання.

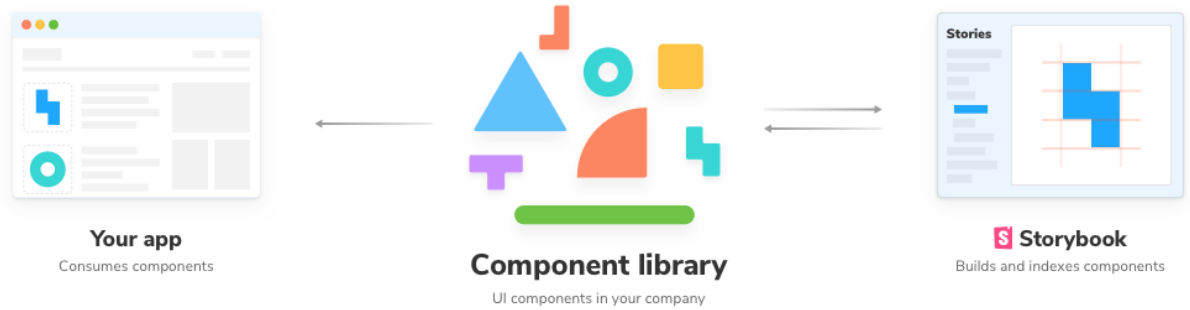

Рисунок 2.6 – Призначення Storybook

Також Storybook допомагає документувати та візуально тестувати компоненти, що запобігти помилок.

2.1.3. Material UI

Рисунок 2.7 – Логотип Material-UI

Material UI – це бібліотека компонентів для React-додатку. Завдяки ній ми одразу отримуємо всі необхідні компоненти, які реалізують Material Design від Google.[8]

Material Design – це адаптована система інструкцій, компонентів та інструментів, які підтримують найкращі методи дизайну інтерфейсу користувача.

«Основною метою Material Design є створення нової візуальної мови, яка поєднує принципи хорошого дизайну з технічними та науковими інноваціями. Дизайнер Матіас Дуарте пояснив, що «на відміну від справжнього паперу, наш цифровий матеріал може розумно розширюватися та реформуватися. Матеріал має фізичні поверхні та краї. Шви та тіні надають значення тому, до чого ви можете доторкнутися». Google стверджує, що їхня нова мова дизайну заснована на папері та чорнилах, але впровадження відбувається в просунутий спосіб.»[21]

2.1.4. NestJS

Рисунок 2.8 – Логотип NextJS

NestJS – це фреймворк для створення серверних додатків на Node.js.[9] Вона пропонує чітку архітектуру: контролер, модуль і провайдер. Завдяки цьому з початку розробки просто розділити все на мікросервіси та працювати над кожним окремо.

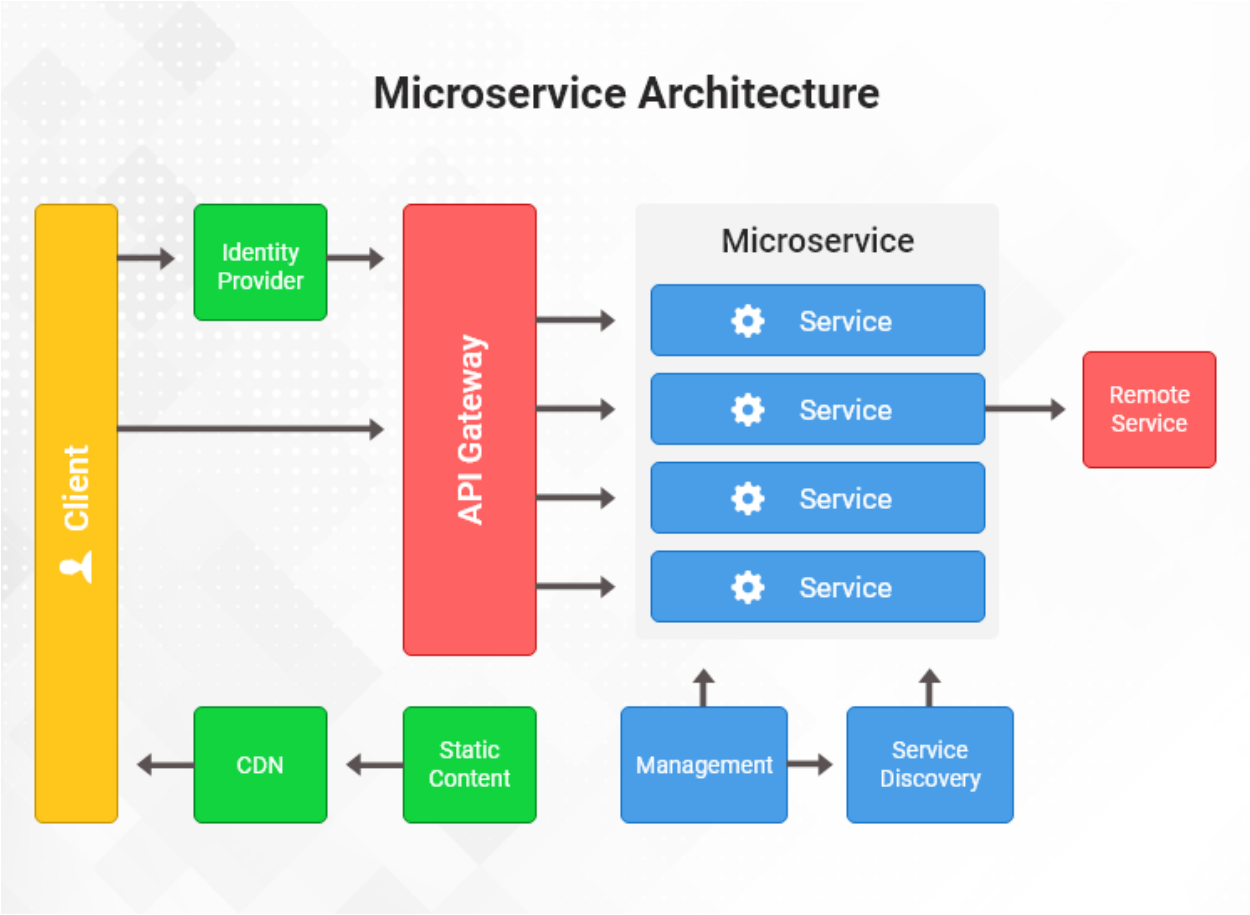

Рисунок 2.9 – Мікросервісна архітектура

Node.js – це серверна платформа для виконання мережевих застосунків, написаних мовою JavaScript. Раніше JavaScript працював лише в браузері, однак Node.js надав можливість виконувати скрипти на сервері.

2.1.5. MongoDB

Рисунок 2.10 – Логотип MongoDB

MongoDB – це документо-орієнтована СУБД, яка на відміну від MySQL не потребує опису схеми таблиці, завдяки чому вона дуже масштабована.[10]

2.1.6. Jest

Рисунок 2.11 – Логотип Jest

Jest – це фреймворк для тестування JavaScript з акцентом на простоті. Він допомагає писати тести з використанням багатофункціонального API, що дає швидкий результат.[11]

Модульне тестування (unit testing) – це метод тестування, який полягає в окремому тестуванні кожного модуля коду програми, найменшої частини програми.[12] Зазвичай її використовують щоб упевнитися, що код відповідає необхідним умовам та має очікувану поведінку.

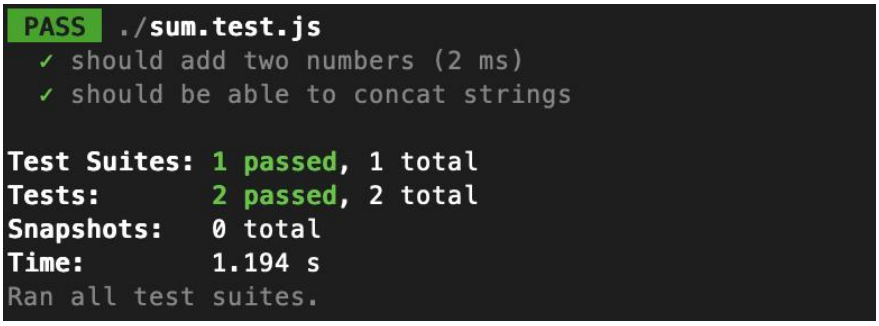

Рисунок 2.12 – Приклад відображення результатів тестування

2.1.7. Cypress.io

Рисунок 2.13 – Логотип Cypress

Cypress – це платформа e2e-тестування на основі JavaScript для автоматизації веб-тестування.[13] За допомогою можна протестувати все, що працює у веб-браузері.

End-to-End (E2E, наскрізне тестування) – це методика, яка перевіряє весь програмний продукт від початку до кінця, щоб переконатися в коректності роботи додатку.[14]

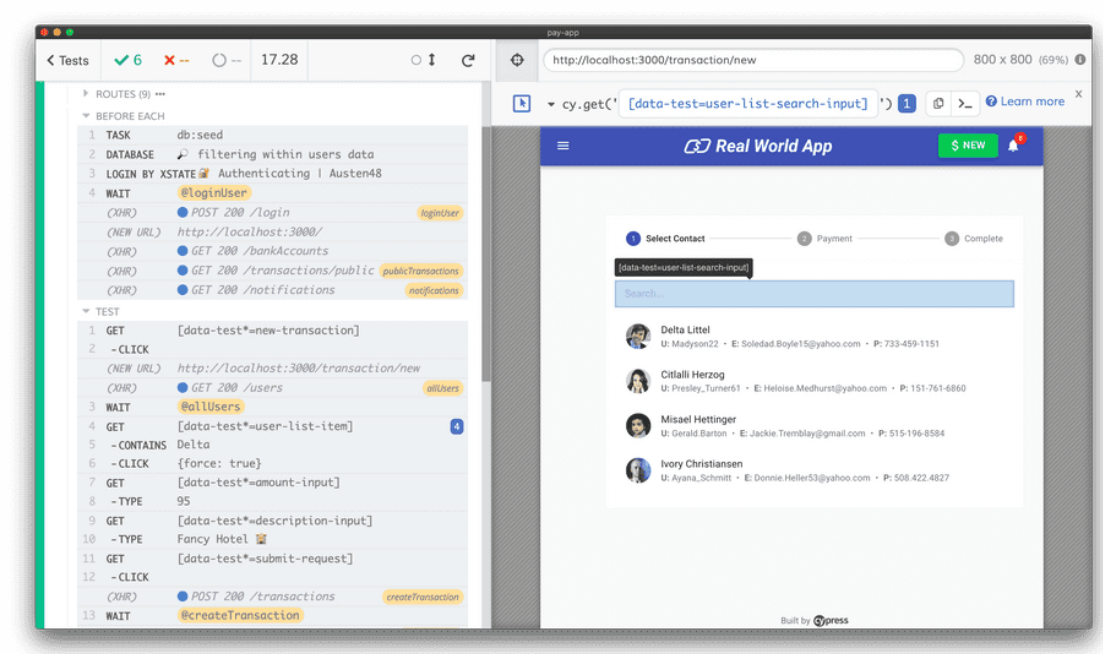

Рисунок 2.14 – Інтерфейс працюючого тесту

2.1.8. React Testing Library

Рисунок 2.15 – Логотип React Testing Library

React Testing Library – це дуже легке рішення для тестування компонентів React.[15] Він забезпечує легкі допоміжні функції, що сприяє кращим методам тестування.

На відміну від Jest та Cypress, React Testing Library призначений для тестування елементів пов’язаних з рендерингом (відображенням).\

2.1.9. Figma

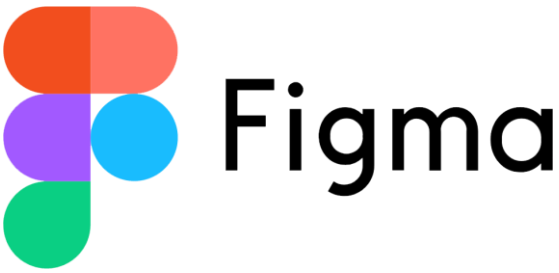

Рисунок 2.16 – Логотип Figma

Figma – це векторний графічний редактор, який дозволяє в браузері займатись дизайном інтерфейсу та прототипуванням.[16] Він корисний не лише дизайнерам, а й front-end розробникам, так як надають можливість завантажити векторні елементи та деякий CSS для Pixel Perfect.

Pixel Perfect верстка – це техніка створення, яка дозволяє зверстаному html-шаблону максимально точно співпадати з оригінальним макетом.[17]

2.2. Що таке тестування?

Тестування – це процес перевірки працездатності коду.[18] Процес тестування складається з кількох етапів, які зазвичай представлені у вигляді піраміди з 3-х рівнів: модульного, інтеграційного та наскрізного (E2E) тестування.

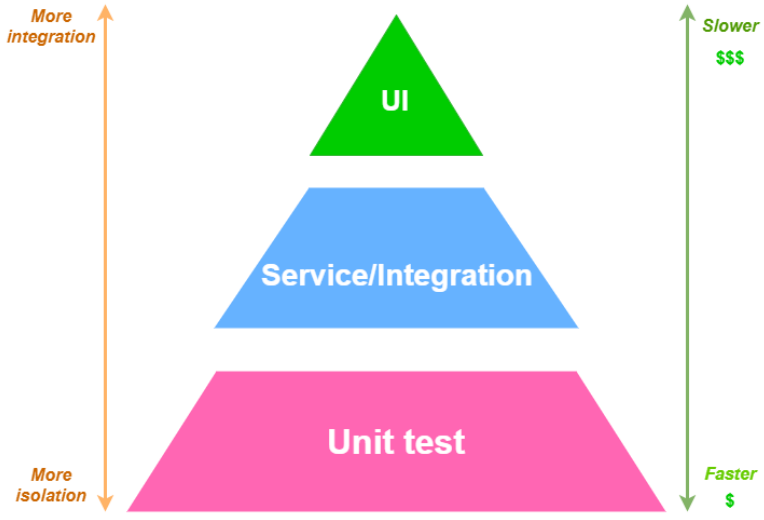

Рисунок 2.17 – Піраміда тестування

При розробці програмного забезпечення тестування використовується на ключових контрольних точках загального процесу, щоб переконатися, що

програмний продукт не містить дефектів та відповідає очікуваним вимогам. Тобто метою тестування є виявлення помилок.

Тестування поділяється також на декілька типів:[20]

- Тестування чорної скриньки (не знаємо як влаштована система);
- тестування білої скриньки (знаємо всі деталі реалізації);
- тестування сірої скриньки (відомі лише деякі особливості реалізації).

2.3. Розробка проекту

Насамперед велась розробка дизайну сервісу. За допомогою Figma було створено дизайн компонентів та прототип додатку (лише візуальний) для кращого розуміння фінального результату.

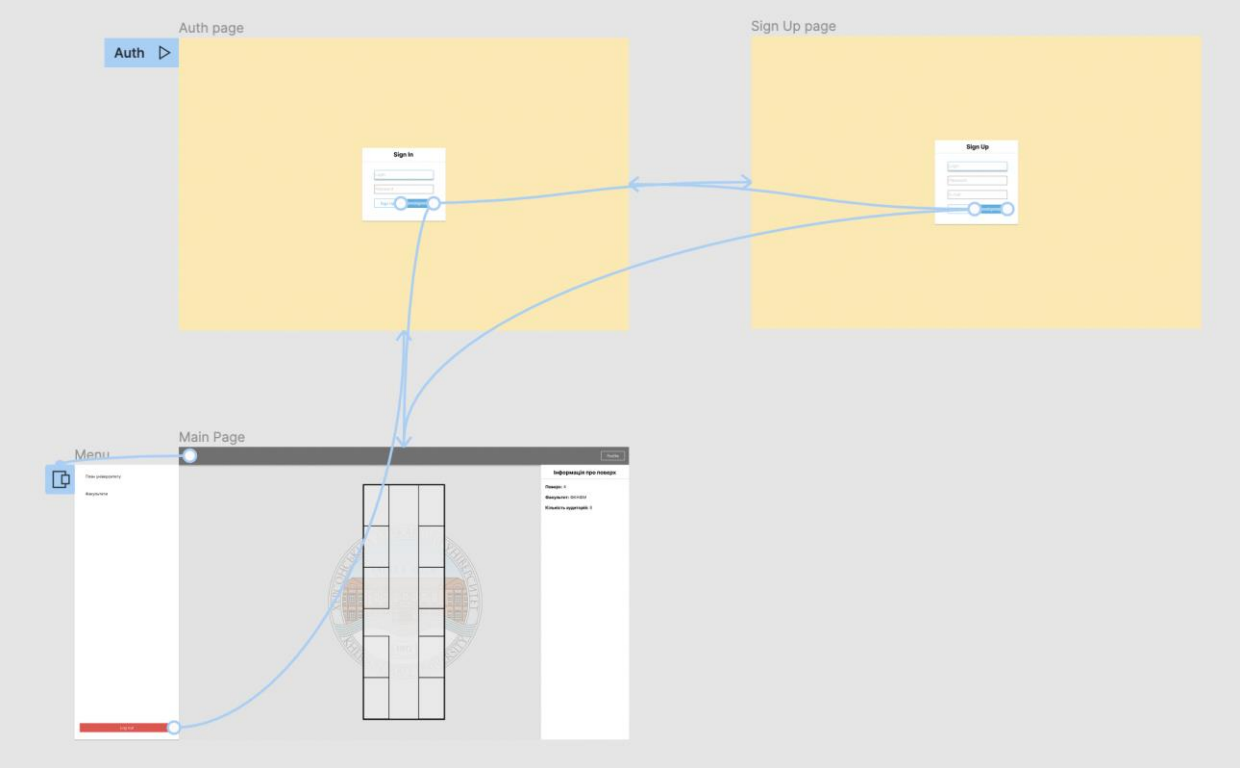

Рисунок 2.18 – Схема прототипу проекту в Figma

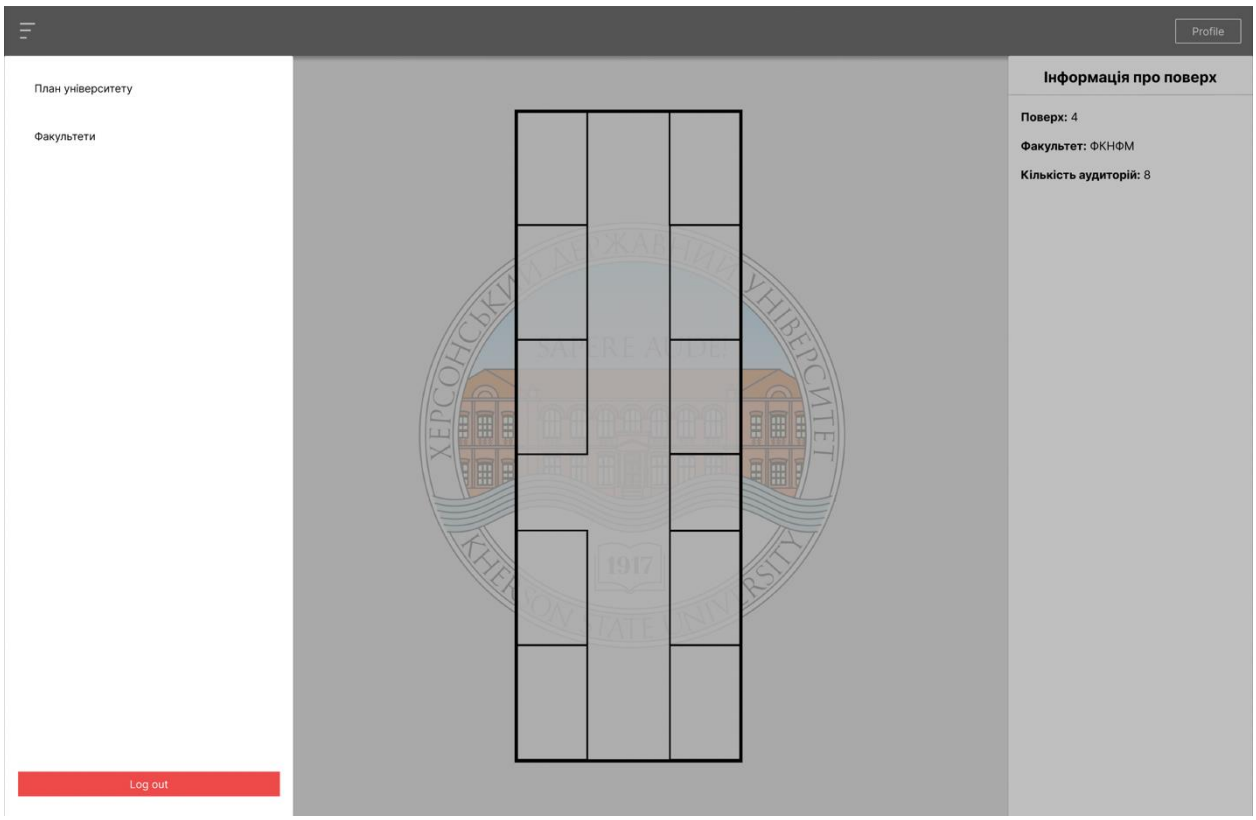

Рисунок 2.19 – Сторінка з працюючого прототипу в Figma

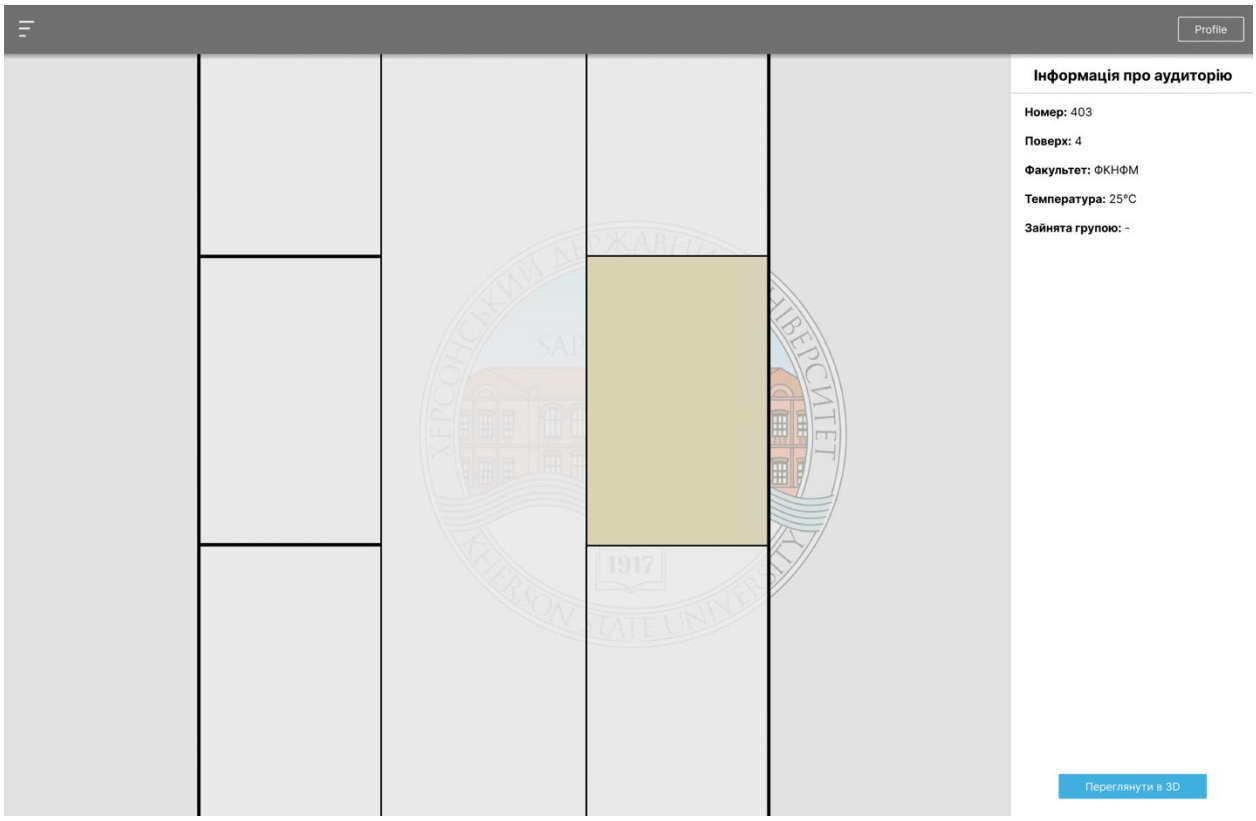

Рисунок 2.20 – Прототип інформації про аудиторію

Коли з дизайном було закінчено, можемо переходити до розробки самого проекту. Спершу я почав з back-end розробки, її архітектури та логіки, щоб під час front-end розробки можна було перевірити коректність роботи.

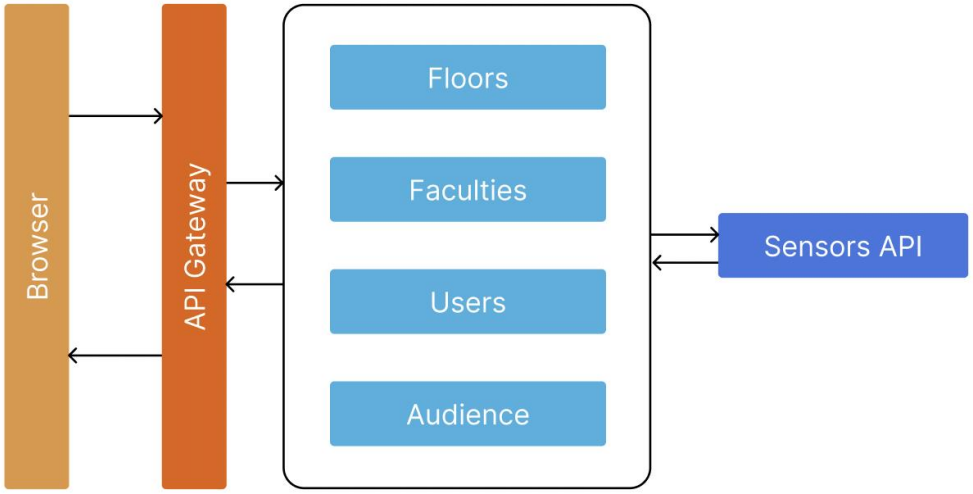

Рисунок 2.21 – Архітектура проекту

Було обрано використати мікросервісну архітектуру, так як NestJS дозволяє розділи все на модулі та працювати с кожними даними окремо. Чому саме її? У мікросервісної архітектури є певні переваги над монолітною структурою:[19]

- Більш зрозуміла структура та легкість у підтримці;
- Кожен сервіс може бути розширеним без необхідності в оновлені всіх елементів;
- Кожен сервіс працює зі своїм набором даних;
- Всі сервіси незалежні та ізольовані, тому якщо в якомусь із них виникне помилка, то вона не вплине на роботу всього додатка.

Після планування архітектури йде процес створення API для кожного модуля, в нашому випадку floors, audience, faculties та users. На даному етапі мінімальним набором методів для кожного модулю буде CRUD, так як функціональність доволі проста та не потребує складних обчислень.

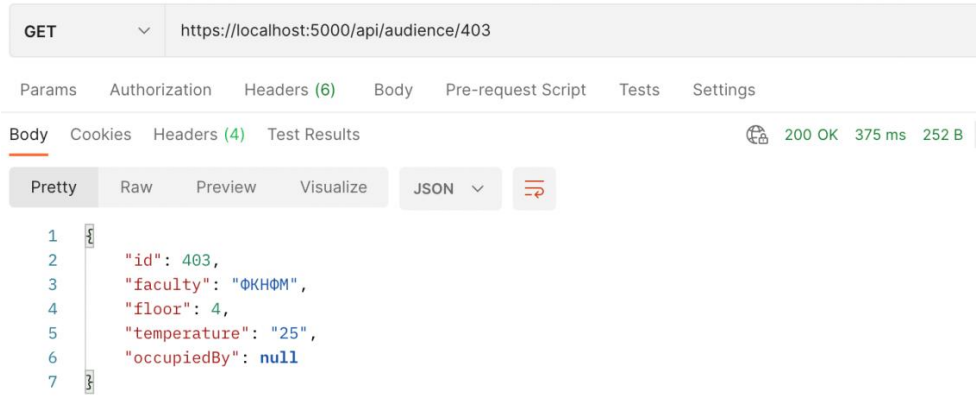

Рисунок 2.22 – Запит даних про аудиторію

За таким простим запитом стоїть більш складна робота, а ніж просто повернення даних. Сервіс audience отримує найпростіші дані про аудиторію (номер аудиторії, номер поверху, id факультету до якого належить), далі, через звернення до інших сервісів, отримуємо додаткову інформацію: назву факультету, температуру в приміщенні та номер групи, яка має знаходитись тут за розкладом.

Інші сервіси будуть працювати за схожою логікою.

Наступним кроком – розробка front-end. Після всіх початкових налаштувань йде розробка базових компонентів, з яких буде створюватися додаток.

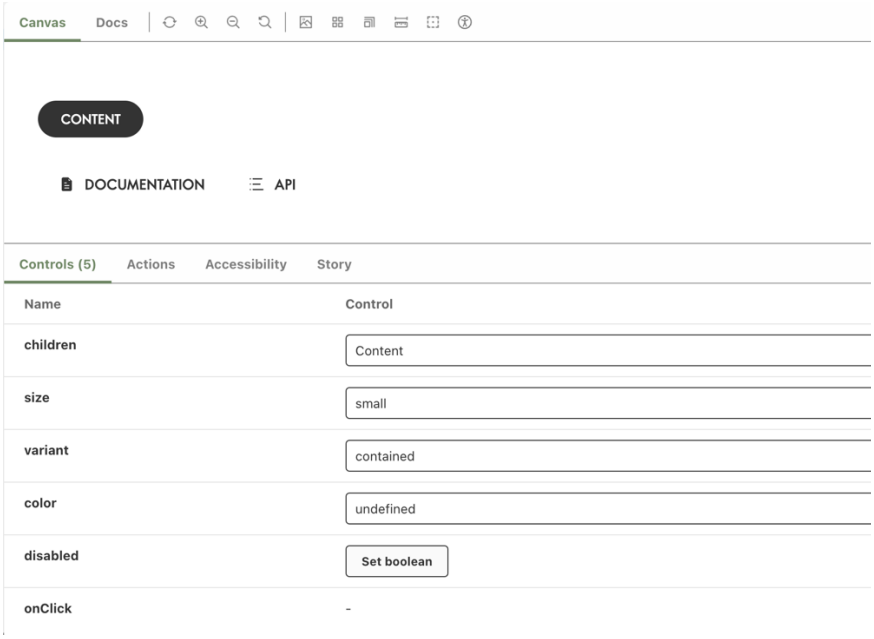

Рисунок 2.23 – Компонент в Storybook

Коли компоненти готові, то йде розробка сторінок, роутингу та взаємодія з back-end. Процес хоча і здається простішим, однак і тут можуть буди непомічені моменти. Як же перевірити, що все працює правильно? Написати тести. Спочатку пишуться unit-тести, щоб перевірити кожен компонент на коректність роботи. Наступним, у моєму випадку, йде E2E тестування – автоматична перевірка роботи всього додатку в цілому, наче користувач фінального продукту.

ВИСНОВОК

У ході виконання кваліфікаційної роботи на основі створення сервісу матеріально-технічної бази я впевнився, що створити систему, яка буде використовуватись лише в одному місці, доволі складно, необхідно врахувати всі можливості для його розширення. Світ не стоїть на місті, тому й університету знадобиться додавати щось нове.

Сама робота спонукала до глибокого вивчення та використання технологій для стабільної роботи додатку та масштабованості.

Під час написання роботи було виконано такі завдання:

- Створили дизайн, прототип та архітектуру самого додатку.
- Розроблено базову версію сервісу
- Додаток був покритий тестами, для достовірності в його коректній роботі

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Херсонський державний університет [он-лайн ресурс]. URL: https://www.kspu.edu/About/Identity.aspx?lang=uk (дата останнього звернення: 04.02.2022)
2. Стаття 44. Матеріально-технічна база закладів загальної середньої освіти [он-лайн ресурс]. URL: https://protocol.ua/ua/pro_zagalnu_serednyu_osvitu_stattya_44/ (дата останнього звернення: 04.02.2022)
3. Що таке API? [он-лайн ресурс] URL: https://seo24.kiev.ua/rozrobka/shho-take-api/ (дата останнього звернення: 25.03.2022)
4. Что такое Next.js и для чего он нужен? [он-лайн ресурс] URL: https://pxstudio.pw/blog/chto-takoe-next-js-i-dlya-chego-on-nuzhen (дата останнього звернення: 17.02.2022)
5. Рендеринг: різновиди, застосування, порівняльна характеристика [он-лайн ресурс] URL: https://medium.com/@ralabs.ua/%D1%80%D0%B5%D0%BD%D0%B4%D0%B5%D1%80%D0%B8%D0%BD%D0%B3-%D1%80%D1%96%D0%B7%D0%BD%D0%BE%D0%B2%D0%B8%D0%B4%D0%B8-%D0%B7%D0%B0%D1%81%D1%82%D0%BE%D1%81%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F-%D0%BF%D0%BE%D1%80%D1%96%D0%B2%D0%BD%D1%8F%D0%BB%D1%8C%D0%BD%D0%B0-%D1%85%D0%B0%D1%80%D0%B0%D0%BA%D1%82%D0%B5%D1%80%D0%B8%D1%81%D1%82%D0%B8%D0%BA%D0%B0-cc12e3a6707d (дата останнього звернення: 17.02.2022)
6. Створення сайтів і додатків на REACT.JS [он-лайн ресурс] URL: https://brander.ua/technologies/reactjs (дата останнього звернення: 18.02.2022)

7. Storybook: UI component explorer for frontend developers [он-лайн ресурс] URL: https://storybook.js.org/ (дата останнього звернення: 27.04.2022)
8. What is Material UI? [он-лайн ресурс] URL: https://flatlogic.com/blog/what-is-material-ui/ (дата останнього звернення: 27.04.2022)
9. Розробка веб-додатків на NEST.JS [он-лайн ресурс] URL: https://brander.ua/technologies/nestjs (дата останнього звернення: 18.02.2022)
10. MongoDB [он-лайн ресурс] URL: https://ru.wikipedia.org/wiki/MongoDB (дата останнього звернення: 18.02.2022)
11. Jest · Delightful JavaScript Testing [он-лайн ресурс] URL: https://jestjs.io/uk/ (дата останнього звернення: 26.03.2022)
12. Модульне тестування [он-лайн ресурс] URL: https://uk.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D1%83%D0%BB%D1%8C%D0%BD%D0%B5_%D1%82%D0%B5%D1%81%D1%82%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F (дата останнього звернення: 26.03.2022)
13. JavaScript End to End Testing Framework [он-лайн ресурс] URL: https://www.cypress.io/ (дата останнього звернення: 05.04.2022)
14. Що таке наскрізне тестування: Структура тестування E2E з прикладами [он-лайн ресурс] URL: https://uk.myservername.com/what-is-end-end-testing (дата останнього звернення: 05.04.2022)
15. React Testing Library [он-лайн ресурс] URL: https://testing-library.com/docs/react-testing-library/intro/ (дата останнього звернення: 27.03.2022)
16. Figma [он-лайн ресурс] URL: https://uk.wikipedia.org/wiki/Figma (дата останнього звернення: 17.04.2022)

17. Pixel Perfect верстка [он-лайн ресурс] URL: https://www.unisender.com/ru/support/about/glossary/chto-takoe-pixel-perfect-verstka/ (дата останнього звернення: 17.04.2022)
18. Тестування програмного забезпечення [он-лайн ресурс] URL: https://uk.wikipedia.org/wiki/%D0%A2%D0%B5%D1%81%D1%82%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%B7%D0%B0%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%BD%D1%8F (дата останнього звернення: 19.04.2022)
19. Микросервис vs Монолитный [он-лайн ресурс] URL: https://ru.education-wiki.com/1235570-microservice-vs-monolithic (дата останнього звернення: 23.04.2022)
20. White/Black/Grey Box-тестування [он-лайн ресурс] URL: https://qalight.ua/ru/baza-znaniy/white-black-grey-box-testirovanie/ (дата останнього звернення: 19.04.2022)
21. Material Design [он-лайн ресурс] URL: https://en.wikipedia.org/wiki/Material_Design (дата останнього звернення: 27.04.2022)