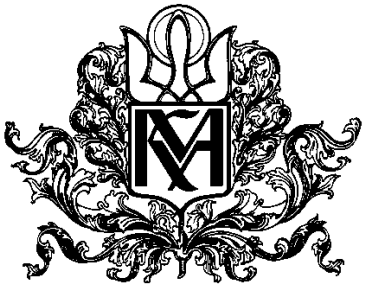
Міністерство освіти і науки України



НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
«КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра мультимедійних систем

Розробка веб застосунку з використанням React та NodeJs

Текстова частина до курсової роботи
за спеціальністю 121 «Інженерія програмного забезпечення»

Керівник курсової роботи
с.в. Борозенний С.О.

_____
(підпис)
“____” _____ 2024 р.

Виконала студентка
Малашок Н.О.
“____” _____ 2024 р.

Київ 2024

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра мультимедійних систем

ЗАТВЕРДЖУЮ
Зав.кафедри мультимедійних систем,
доцент, к.ф-м.н.
_____О. П. Жежерун (підпис)
“____”_____2023 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на курсову роботу

студентці Малашок Наталії Олександрівні факультету інформатики 3 курсу

ТЕМА: Розробка веб застосунку з використанням React та NodeJs

Зміст ТЧ до курсової роботи:
Індивідуальне завдання
Вступ
1. Огляд рішень, що існують
2. Використані технології та інструменти
3. База даних
4. Вебзастосунок та його функціонал
Висновки
Джерела
Додатки (за необхідністю)

Дата видачі “___”_____ 2023 р. Керівник _____ (підпис)

Завдання отримав _____ (підпис)

Календарний план виконання роботи

№	Назва етапу курсового проєкту (роботи)	Термін виконання етапу
1	Отримання завдання на курсову роботу	28.09.2023
2	Аналіз матеріалів та визначення актуальності	26.12.2023
3	Розробка вебзастосунку	08.01.2024
4	Написання текстової частини до курсової роботи	01.04.2024
5	Коригування виконаної роботи	02.05.2024
6	Створення презентації	08.05.2024

Малашок Н. О. _____

Борозенний О. С. _____

“_____”_____

Зміст

Анотація

Курсова робота зосереджена на створенні вебзастосунку. Додаток створений з використанням React для клієнтської частини та Node.js для частини серверу. Він напрямлений на полегшення створення заявки на подання на певне приміщення, а також її підтвердження. Окрім того, програма має функціонал публікації інформації про певний захід та перегляд усіх вільних варіантів часу у певному приміщенні.

У курсовій роботі буде проведено аналіз існуючих рішень та потреб, які вони забезпечують, описано функціональні можливості застосунку та проведено огляд технологій, використаних при розробці.

Вступ

Створення подання на певне приміщення – невід’ємний процес студентського життя, оскільки для проведення заходу завжди потрібне місце. Проте потреба уточнювати час і місце та надсилати подання часто додатково ускладнюють і без того важкий процес організації події. На даний момент відсутній сервіс для заповнення заяки на приміщення, за допомогою якого можна було б виконувати всі дії, потрібні для створення подання без використання додаткових застосунків. Тож розробка програми, що забезпечить полегшення цього процесу, є вкрай важливою.

Вебзастосунок для забезпечення покращеного процесу подання заявок на локацію створений з використанням React та Node.js (Nest.js, Express.js, TypeScript) для фронтенду та бекенду відповідно.

Мета розробки: створення застосунку для полегшення створення подання та дослідження можливостей бібліотеки React та Nest.js.

Можливості програми:

- перегляд опублікованих заходів;
- створення заявки на приміщення (одночасно з заявкою на публікацію заходу);
- перегляд своїх подань;
- підтвердження заяв для відповідальної за приміщення особи;
- перегляд календаря з вільними варіантами часу.

Такий функціонал дозволяє користувачу ефективно виконувати дії потрібні для створення або підтвердження подання, перегляду подій, відкритих для відвідування.

Розділ 1. Огляд рішень, що існують

1.1 Outlook Calendar

Outlook Calendar – інструмент від Microsoft, який використовується, щоб планувати та зберігати інформацію про події. Він повністю інтегрований з електронною поштою та контактами [1]. З точки зору використання корпоративною пошти, використання цього ресурсу є дуже зручним, оскільки можна створити групу та поширити інформацію для всіх людей доданих до неї. Проте є кілька аспектів, які зумовлюють його не оптимальність для створення подання:

- Запити, які виникають не зразу додаються до системи Outlook Calendar, через те, що мають бути опрацьовані людьми, а отже, можуть виникати накладки по часу та місцю заходу;
- Переглядати зайнятий час у календарі може бути незручно, оскільки всі приміщення (в даному випадку КМЦ) показуються одночасно і мають тільки відмінний підпис та поділ по часу;
- Запитувачу всеодно доведеться використовувати додаткове джерело комунікації (наприклад: пошту), щоб уточнити чи вільне приміщення та/або надіслати подання;
- На даний момент такий календар існує тільки для КМЦ (Культурно-Мистецького центру), а отже, дізнатися чи вільне приміщення в іншому корпусі/бібліотеці неможливо.

1.2 Microsoft Forms

Microsoft Forms – інструмент, який дозволяє створювати опитування, форми та тести [2]. Програма дозволяє користувачу створити обмеження для респондентів в межах корпоративних пошт в середині однієї організації,

що робить її одним з способів створення запиту на проведення заходу на території НаУКМА. Хоча ця програма дуже зручна для збору даних, проте не є найкращим рішенням для створення подання на приміщення з певних причин:

- В Microsoft Forms можна зробити форму з варіантами часу для можливого подання, проте це не убезпечує від накладок при створення запиту;
- Велика кількість приміщень в НаУКМА буде потребувати або великої кількості форм (що буде не зручно для людини, що хоче надіслати запит), або великої кількості варіантів у формі (що ускладнюватиме перегляд відповідей відповідальній людині);
- хоча Microsoft Forms уможливлює додавання письмового варіанту подання до форми, проте підтвердження часу всеодно доведеться надсилати на пошту;
- в Microsoft Forms ніяк не можна переглянути зайняті слоти на певний день, а отже, доведеться використовувати інші програми;
- Хоча Microsoft Forms дозвляє вивантажувати Excel файл з усіма відповідями, проте це ускладнює перегляд нових заявок та не відфільровує старі та уже опрацьовані.

1.3 Outlook

Outlook – застосунок з функціями поштового клієта, який містить інші функції від компанії Microsoft (наприклад: календар, планер, менеджер контактів). В програмі легко знайти потрібну людину, використовуючи корпоративну пошту. Щоб написати стосовно подання, потрібно знати ім’я та прізвище відповідальної за приміщення особи (або у випадку КМЦ – місце), проте не на всі приміщення в НаУКМА написати подання можна через пошту, оскільки не у всіх корпусів/бібліотек є окрема пошта (як у

КМЦ) або ж нема відомостей про відповідальну особу. Хоча Outlook є зручним для комунікації між стужентами, викладачами та адміністрацією академії, це не найкращий варіант для надсилання подання, оскільки якщо особа не має поділу на папки, то існує ризик, що лист загубиться серед іншої пошти. Окрім того, застосунок не надає можливості авоматично фіксувати зайнятий час, та пошту не можливо синхронізувати з календарем (Outlook Calendar).

1.4 Microsoft Team

Microsoft Teams – програма, яка є основним інструментом для комунікації між усіма учасниками освітнього процесу, проте для надсилання подання також не є оптимальним варіантом, оскільки:

- Microsoft Teams не має функції відображення уже забронюваних варіантів часу;
- У Microsoft Teams відсутні акаунти кожного приміщення та/або відповідальної за них особи.

1.5 Telegram

Telegram – месенджер, відомий своєю безпекою та багатофункціональністю. Однією з можливостей Telegram є створення каналів, тобто платформ, де адміністратор може публікувати повідомлення для користувачів. Ця опція добре підходить для того, щоб поширювати інформацію про заходи в КМА та поза її межами, проте:

- В Telegram не можна фільтрувати повідомлення по даті проведення саме заходу, а не відправлення повідомлення;

- Могилянські Telegram-канали не можуть знати про всі події, які відбуваються, оскільки не вони займаються підтвердженням подань на приміщення;
- Telegram не має функціоналу, який би дозволив відслідковувати всі приміщення та позначати коли те чи інше місце зайнято, він тільки надає можливість сконтактувати і дізнатися про те, чи локація вільна (в цьому випадку подання всеодно доведеться надсилати на пошту, оскільки Telegram не є офіційним способом комунікації в НаУКМА, та відповідальним за приміщення потрібно структурувати інформацію про бронювання приміщень).

1.6 Google Forms

Google Forms – інструмент, для створення опитувань та анкет, який допомагає збирати дані. За рахунок можливості підлючення різних доповнень Google Forms має ширший функціонал за Microsoft Forms. У Google Forms можна обежити кількість відповідей, які можна надати за 1 варіант, для цього можна використати плагін Choice Eliminator for Google Forms™, проте Google Forms це не оптимальний варіант з деяких причин:

- через відсутність підключення до корпоративної пошти не вийде обмежити сторонніх користувачів і форма буде доступна всім;
- у випадку відхилення запиту на приміщення доведеться використовувати інші програми для сповіщення про це;
- якщо запит було відхилено, щоб відновити варіант у формі, потрібно кожен раз її редагувати;
- час початку та час кінця заходу не можна перевірити (тобто можна ввести такі значення, що кінець заходу буде раніше за початок);

- в Google Form вивід інформації відбуваєтья у Google-таблицю, де нові запити будуть додаватися у кінець і потрібно буде налаштовувати фільтри для зручного перегляду відповідей.

1.7 Notion Calendar

Notion Calendar – інструмент для планування та зберігання інформації. Його функціонал дозврляє незареєстрованому користувачу переглядати поширений календар у браузері [3]. Хоча за рахунок зручного виводу планів, застосунок є одним з варіантів фіксування запитів на приміщення в академії, проте узгодження часу та дати всеодно має відбуватися на інших ресурсах. Окрім цього, через велику кількість приміщень – доведеться використовувати велику кількість календарів, що буде ускладнювати сприйняття для користувача.

1.8 Обгрунтуання актуальності розробки

Ознайомившись з існуючими варіантами рішення посталеної задачі, було визначено, що весь потрібний функціонал не вкладений у жодне з оглянутих рішень. Тож розробка застосунку, який би полегшував процес створення подання – є важливою. Тому у застосунку створений такий функціонал, як:

- календар, в який би автоматично додавалися створені заявки;
- можливість переглянути вільний час для певного приміщення та створити заявку, не виходячи з застосунку;
- можливість переглядати і підтверджувати заявки в одному місці, не ризикуючи загубити їх серед іншої інформації;
- відсутність ризику накладання двох подій;

- можливість переглядати усі заходи доступні до відвідування з покликаннями на них;
- можливість лишити заявку на публікацію заходу одночасно з створенням подання на приміщення.

Проєкт втілено у вигляді вебзастосунку, оскільки це робить його доступним усім користувачам, окрім того, така програма не вимагає від користувача додаткового завантаження або оновлення.

Розділ 2. Використані технології та інтсрументи

2.1 TypeScript

TypeScript – це мова програмування, яка є надбувдовою над JavaScript. Вона була розроблена Microsoft у 2012 році. Мова є статичнотипізованою, що робить код надійнішим. На відміну від JavaScript, TypeScript є компільованою мовою, що надає їй перевагу, оскільки розширює функціонал та надає можливість виявити можливі помилки на етапі компіляції [4].

TypeScript було обрано для розробки, тому що:

- TypeScript має строгу типізацію;
- легко суміщається з JavaScript та інтегрується з її бібліотеками;
- підтримує функціонал, що представлений в ES6 (ECMAScript 6) і більше;
- має вбудовану підтримку модулів для кращої організації коду;
- підтримує концепції ООП, як: інтерфейси, класи та наслідування [4];

2.2 Node.js

Node.js – це програмне середовище, яке використовується для запуску коду на JavaScript поза браузером клієнта та створене на основі JavaScript-рушія V8 [5]. За рахунок цього за допомогою Node.js на JavaScript можна писати не тільки фронтенд, а й бекенд частину.

Переваги використання Node.js:

- За рахунок обробників подій функції займають менше пам’яті та витрачають менше ресурсу сервера;
- Однопоточніть зменшує час обробки запиту завдяки тому, що всі запити збираються в циклі обробки подій (Event Loop) [6];

- Підтримка асинхронності та неблокуючий характер введення-виведення, що сприяє виконанню кількох процесів паралельно та дозволяє обробляти дані вкрай швидко [7];
- Node.js легко поєднується з React або іншими фреймворками/бібліотеками на фронтенді, що робить обробку та рендеринг швидким;
- За рахунок можливості використання JavaScript на серверній частині, розробники можуть використовувати одну мову для фронтенду та бекенду, що спроситить подальшу підтримку проєкту.

2.3 Nest.js

Nest.js – це фреймворк Node.js, який використовується для створення масштабованих серверних програм з використанням TypeScript, окрім того, Nest.js підтримує модульну архітектуру [8].

Основними перевагами використання Nest.js є:

- Написання коду на TypeScript;
- Підтримка Rest (та GraphQL) та можливість створення програми з використанням MVC (Model-View-Controller);
- Підтримка TypeORM;
- Сумісність з Express.js, завдяки чому в код легко інтегрувати бібліотеки Express;
- Наявність зручного CLI та детальної документації [9].

Організація коду в Nest.js відбувається за допомогою поділу на 3 блоки:

- Модулі (Modules): контейнери для інших частин коду, навколо яких структурована програма. Для створення модулю використовується

декоратор “@Module()”, а відповідні контроллери та сервіси оголошуються в “controllers” та “providers” відповідно.

- Контроллери (Controllers): відповідають за обробку HTTP-запитів і повернення відповіді на них. Для створення маршрутів використовують декоратори з відповідним HTTP-методом та шляхом, для самого контроллеру використовується - “@Controller()”.
- Сервіси (Services): використовуються для виокремнення бізнес-логіки та маніпулювання даними. Зазвичай у них відбуваються запити до бази даних або взаємодія з іншими зовнішніми API. Для сервісу використовується декоратор “@Injectable()”, який позначає що провайдер може бути використаний в інших компонентах (наприклад: контроллерах) [10].

2.3 Express.js

Express.js – це гнучкий та швидкий вебфреймворк для Node.js. Він пропонує набір функцій, які оптимізують застосунок та максимально полегшують його створення за допомогою проміжного програмного забезпечення та маршрутизації [11].

Express.js було використано з таких причин:

- підходить для середніх та малих проєктів;
- допомагає при створенні REST API;
- масштабований та дозволяє обробляти велику кількість запитів асинхронно [11];
- гнучкий та адаптивний, за рахунок чого дозволяє створювати потрібну розробнику архітектуру програми, а не змушує дотримуатися зазначеної;
- маршрутизація направляє запити на відповідно задані обробники;

- за допомогою проміжного програмного забезпечення легко виконувати завдання на кшталт автентифікації [11];
- фреймворк легкий для розуміння та підтримки, через свою популярність та доступну документацію.

2.4 TypeORM

ORM – об’єктно-реляційне відображення, яке є з’єднанням між базою даних та об’єктно-орієнтованим програмуваннням. Воно дозволяє керувати базою даних за допомогою мови програмування (об’єктно-орієнтованої), тобто дозволяє спроситити написання запитів та нівелює потребу змінювати SQL код при зміні бази даних [12].

TypeORM – це популярний ORM (Object Relational Mapper), який спрощує поєдання програми на TypeScript з базою даних. З його допомогою можна створювати запити до бази даних використовуючи об’єктно-орієнтований підхід [13].

Перевагами TypeORM є:

- легке поєдання з NestJs;
- синтаксис орієнтований для роботи з TypeScript, що дає можливість абстрагувати SQL-запити [14];
- декоратори дозволяють забезпечити зручність та додаткову безпеку застосунку;
- дозволяє легко створювати міграції за допомогою існуючих об’єктів (моделей);
- зменшує час розробки, усуваючи дублювання SQL коду [13];
- можливість використання як з реляційними базами даних (наприклад: PostgreSQL), так і з noSQL (нереляційними);
- швидке зростання спільноти TypeORM забезбечує хорошу документацію.

Суб’єкт (модель) – це предсталення певної сутності, через яке відбувається окреслення атрибутів. За допомогою TypeORM модель позначається декоратором “@Entity()” .

Завдяки інтегруванню TypeORM у програму, суб’єкти (моделі) отримують додаткові можливості взаємодії з базою даних. З використанням декораторів в класах моделей можна визначити перевірки та зв’язки, які будуть в базі даних.

2.5 React

React – це бібліотека JavaScript для створення інтерфейсу користувача (UI). Вона допомагає створювати односторінкові програми (SPA) та багаторазові компоненти [17].

Односторінкова програма (SPA) – це вебзастосунок, який завантажує одну HTML-сторінку, оновлюючи її вміст динамічно під час взаємодії з користувачем.

Компоненти – незалежні фрагменти коду, які можна використовувати багато разів, та у яких реалізується метод “render()”, що повертає HTML [18].

У React використовується синтаксичне розширення JavaScript – JSX, воно озволяє писати HTML-код всередині JavaScript файлу. За допомогою JSX HTML-код та логіка рендерингу зберігаються разом в одному місці (компоненті), що гарантує їх синхронність та ізольованість від інших елементів коду. TSX використовується для роботи з React в поєднанні з TypeScript, що дозволяє писати HTML-код всередині TypeScript.

Основні переваги використання React:

- можливість повторного використання коду за допомогою компонентів;

- наявність ReactDOM – віртуального DOM, який використовується для швидкого зберігання змін та миттєвого оновлення оригінального DOM браузера, що дозволяє максимально пришвидшити завантаження сторінки [17];
- не потребує реалізації конкретної архітектури, лишає вибір архітектури за розробником;
- велика кількість інструментів для розробки: вбудованих функцій, розширень, бібліотек;
- можливість використання React Hooks, таких як: useState, useReducer, useContext, useEffect, useRef та інші, окрім вбудованих хуків, React пропонує можливість створення власних. React Hooks – інструменти, що дозволяють компонентам мати доступ до стану та функцій життєвого циклу React [20].

2.5.1 Vite.js

Для створення застосунку з React було використано Vite.js – інструмент, що спрощує та прискорює процес, окрім того він:

- підтримує стандарт ESM (EcmaScript Module), тобто застосовує модульний підхід та обробляє файли проєкту як окремі модулі;
- використовує відкладене завантаження модулів, що означає, що завантажується лише код, який потрібен зараз;
- використовує вбудований сервер, який орієнтований на швидке перезавантаження та гарячу зміну модулів (HMR), що дозволяє оновлювати всю програму, а не тільки певний модуль, та бачити зміни, внесені у код, в режимі реального часу;
- використовує паралельну обробку, що забезпечує окрему обробку кожного модулю та компіляцію тільки тої частини проєкту, яку було змінено;

- використовує такі методи як стиснення файлів та видалення непотрібного змісту, що оптимізує процес збірки проєкту та пришвидшує завантаження [21].

2.5.2 React Router

React Router – бібліотека, створена для керування маршрутизацією на стороні клієнта, яка є зручним інструментом для створення динамічних односторінкових програм у React. Вона дозволяє користувачу переміщуватися між різними представленнями програми, не запитуючи нову HTML-сторінку у сервера, а змінюючи лише вміст [22].

2.6 Axios

Axios – це HTTP-клієнт на основі Promise, створений для застосування у браузері або Node.js. На стороні сервера використання HTTP-модуль Node.js, а у клієнтській частині – XMLHttpRequests [23].

Axios було використано з таких причин:

- підтримується всіма браузерами;
- не потребує дотакової обробки для отримання відповіді;
- має вбудований захист від CSRF (Cross-Site Request Forgery);
- надає підтримку для скасування запитів, а також можливість перехоплення запитів [24];

Розділ 3. База даних

PostgreSQL – це об’єктно-реляційна система управління базами даних, яка базується та розширює мову SQL [15].

Для розробки застосунку було обрано PostgreSQL з таких причин:

- забезпечує швидкість роботи та може обробляти велику кількість транзакцій;
- має підтримку спільноти та відкритий код, через що наявний функціонал постійно вдосконалюється та оновлюється;
- виконує перевірку відповідності даних, а не лишає відповідальність на розробнику [16];
- легко суміщається з Node.js та багатьма іншими мовами, за допомогою бібліотек;
- PostgreSQL є гнучкою, тобто розширити функціонал можна в будь-який момент;
- PostgreSQL пропонує зручні функції для забезпечення цілосності даних, продуктивності та безпеки.

Також було створено ER-модель та реляційну модель для кращого розуміння структури та подальшого збереження цілісності.

3.1 ER-модель

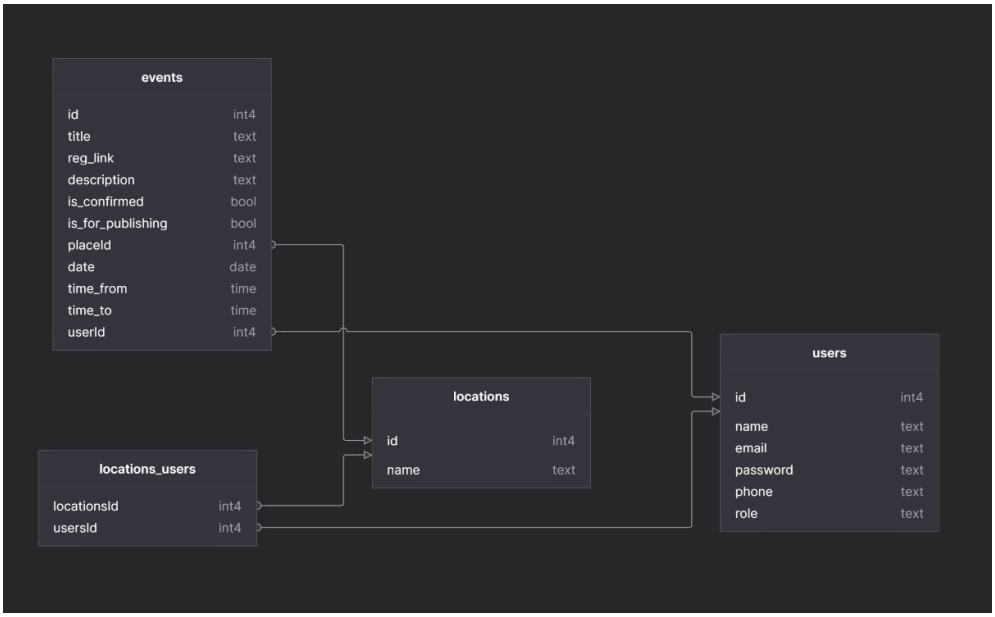


Рисунок 1. ER-модель

3.2 Реляційна модель

LOCATION Відповідає сутності “locations”				
Ключ	Ім’я	Тип	Обов’язковість	ON DELETE/ ON UPDATE
PK	id	int	NOT NULL	
	name	text	NOT NULL	

Таблиця 1. Сутність “LOCATION”

USER Відповідає сутності “users”				
Ключ	Ім’я	Тип	Обов’язковість	ON DELETE/ ON UPDATE
PK	id	int	NOT NULL	
	name	text	NOT NULL	
	emails	text	NOT NULL	
	password	text	NOT NULL	

	phone	text	NOT NULL	
	role	text	NOT NULL	

Таблиця 2. Сутність “USER”

EVENT Відповідає сутності “events”				
Ключ	Ім’я	Тип	Обов’язковість	ON DELETE/ ON UPDATE
PK	id	int	NOT NULL	
	title	text	NOT NULL	
	reg_link	text	NULL	
	description	text	NOT NULL	
	is_confirmed	bool	NULL	
	is_for_publishing	bool	NOT NULL	
FK	placeId	int	NOT NULL	ON DELETE: set null ON UPDATE: cascade
	date	date	NOT NULL	
	time_from	time	NOT NULL	
	time_to	time	NOT NULL	
FK	userId	int	NOT NULL	ON DELETE: cascade ON UPDATE: cascade

Таблиця 3. Сутність “EVENT”

LOCATIONS_USERS Відповідає зв’язку “locations_users”				
Ключ	Ім’я	Тип	Обов’язковість	ON DELETE/ ON UPDATE
PPK FK1	locationsId	int	NOT NULL	ON DELETE: no action ON UPDATE: cascade

PPK FK2	usersId	int	NOT NULL	ON DELETE: no action ON UPDATE: cascade

Таблиця 3. Зв’язок “ LOCATIONS_USERS”

Розділ 4. Вебзастосунок та його функціонал

4.1 Архітектура проєкту

4.1.1 Архітектура серверу

index.ts – кореневий файл серверу. У ньому NestFactory створює екземпляр NestApplication. Головний модуль app.module.ts імтпортує всі модулі створені для застосунку, такі як:

- ConfigModule – модуль, який допомогає керувати конфігурацією застосунку (імпортуєсть з @nestjs/config);
- TypeOrmModule – забезпечує інтеграцію ORM для роботи з базою даних (імпортуєсть з @nestjs/typeorm);
- EventModule, LocationModule, UserModule – модулі, які забезпечують функціонал програми (модулі знаходяться в директорії /Modules).

У директорії /Models розташовані суб’єкти та об’єкти передачі даних. В суб’єктах за допомогою TypeORM встановлюються назви та типи колонок у базі даних, а також за потреби зв’язки між ними (ManyToMany, ManyToOne, OneToOne) з використанням @JoinTable/@JoinColumn.

Об’єкти передачі даних (DTO) – це об’єкти, які використовуються для отримання даних користувача та їх валідації. Вони захищають програму від помилкових введень та зберігають цілісність даних.

У файлі “User.entity.ts” виконується хешування паролю перед додаванням в базу даних (@BeforeInsert) за допомогою бібліотеки bcrypt.

Bcrypt - бібліотека, яка допомагає хешувати пароль [25]. Bcrypt поєднує хешування та додавання солі (модифікатору):

- хешування паролю – це процес перетворення поролю в рядок символів певної довжини, що важливо для безпеки застосунку;

- додавання солі – це додавання випадкового числа до паролю, після чого комбіноване значення передається до хеш-функції, що підвищує безпеку використання bcrypt [26].

Файли, що містять конфігурації проєкту:

- orm.config.ts – містить налаштування для підключенння до бази даних та вказує шлях для згенерованих міграцій. Міграції – це файл з sql-запитами для застосування змін до бази даних або її оновлення [27];
- tsconfig.json – файл, в якому знаходяться параметри для компіляції проєкта.

Папка dist створюється автоматично під час компіляції коду, у ній зберігаються .js та .d.ts файли. Це допомагає відокремити вихідний код від скомпільованого, що використовується для виконання програми.

Контроллери проєкту знаходяться в директорії /Controllers. Вони відповідають за реалізацію Restfull API: приймають HTTP-запити та виконують певну обробку або ж передають отриману інформацію у сервіси, які виконують подальшу взаємодію з базою даних (сервіси знаходяться в директорії /Services).

4.1.2 Архітектура фронтенду

main.tsx – кореневий файл клієнтської частини вебзастосунку. У ньому стоврюється BrowserRouter для забезпечення маршрутизації додатку та AuthContextProvider для створення контексту автентицікації.

Директорія /components відповідає за конмпонети, які використовуються у застосунку, такі як: форми для створення запитів та логіну, сторінки підтердження поданнів, меню, блоки інформації про подію.

В директорії /interfaces знаходяться інтерфейси, які відповідаються за визанчення формату даних у застосунку. Їх використання у TypeScript дозволяє чітко визначити структуру даних, що допомагає уникнути помилок

під час роботи програми. Директорія /services відповідає за файли, що містять методи для взаємодії з сервером.

Директорії /auth відповідає за авторизацію та автентифікацію. У ній знаходяться такі файли:

- authActions.ts – у ньому знаходять дії, які може виконувати користувач;
- authReducer.ts – у ньому визначається редуктор, що використовується для керування діями пов’язаними з автентифікацією, тобто приймає поточний стан та дію для повернення нового стану;
- AuthContextProvider.tsx – компонент провайдера контексту для забезпечення стану автентифікації для всього застосунку (вся програма огорнута в цей провайдер).

4.2 Функціонал проєкту

4.1 Логін

Для входу в акаунт користувача у застосунку створена сторінка Логіну (див. Рис. 2). Без авторизації користувач не зможе створювати подання та переглядати події, якими поділилися для публікації. Адміністратор без входу в систему не може підтверджувати подання, які надійшли на його локацію.

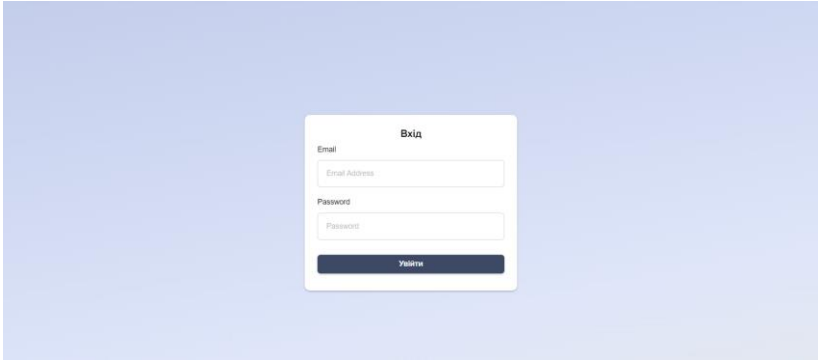


Рисунок 2. Вхід

4.2 Сторінка подій

На сторінці з подіями можна переглянути всі події, які відбуваються у певну дату (див. Рис. 3). Цей функціонал запропоновний як доповнення до могилянських телеграм каналів, оскільки всі події, що відбуваються в межах КМА та потребують розголосу, будуть занесесні в базу даних, та їх можна буде перглянути по даті. В той час як інформація про них як і раніше буде розташована в соцмережах СО (студентської організації), де про неї можна буде дізнатися більше, це спростить пошук події, на яку можна піти, а також не ускладнюватиме піар для організаторів. У блоці події буде видно її назву та короткий опис, а також місце проведення та дату і час початку (див. Рис. 4). За потреби можна буде додати посилання на оригінальний пост в соцмережах або форму для реєстрації. Не всі події будуть виведені, оскільки при створення заявки, користувач зможе обрати чи потребує його подія розголосу (наприклад: зустріч з першокурсниками від Бадді – не потребує розголосу).

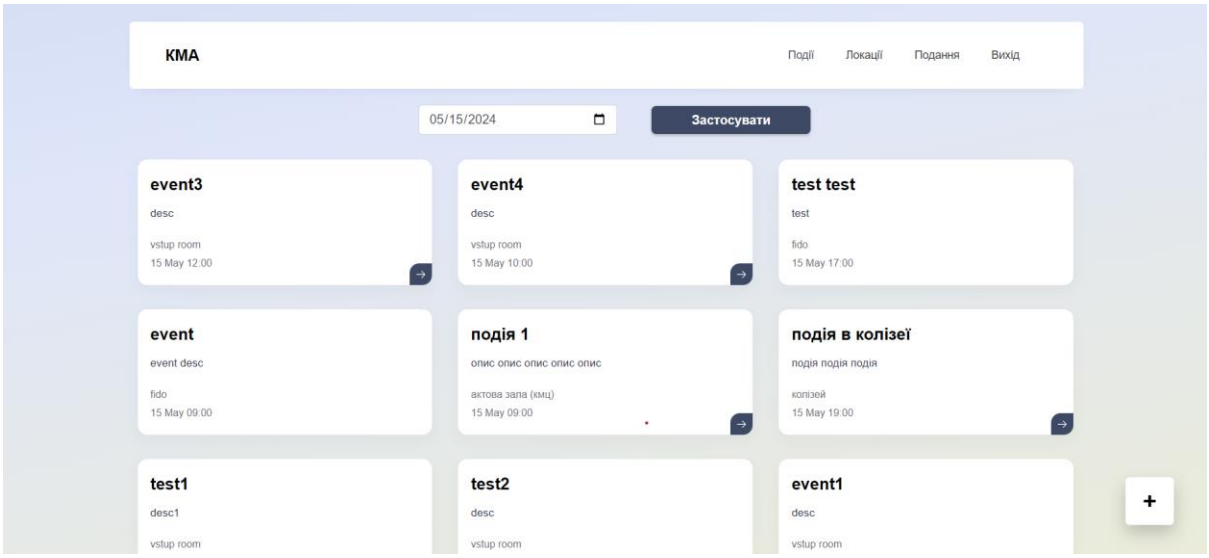


Рисунок 3. Сторінка подій

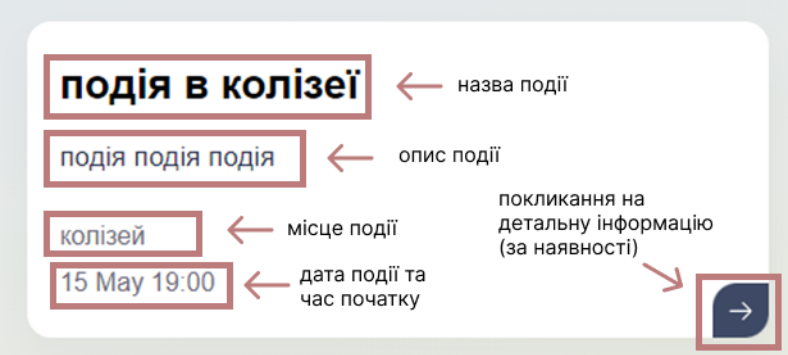


Рисунок 4. Компонент події

4.3 Створення подання

Створити запит на приміщення можна 2ма способами:

- натиснувши на кнопку “+” в правому кутку сторінки (див. Рис. 5)
- натиснувши на час та локацію (див. Рис. 5): в цьому випадку локація та час початку автоматично підтягнуться в форму.

Для вибору часу в формі доступні тільки ті варіанти часу, на які подання ще не відправлялося, через що відпадатиме потреба уточнювати чи точно вільне приміщення у певний час та не потрібно буде перевіряти чи нема накладки (див. Рис. 6).

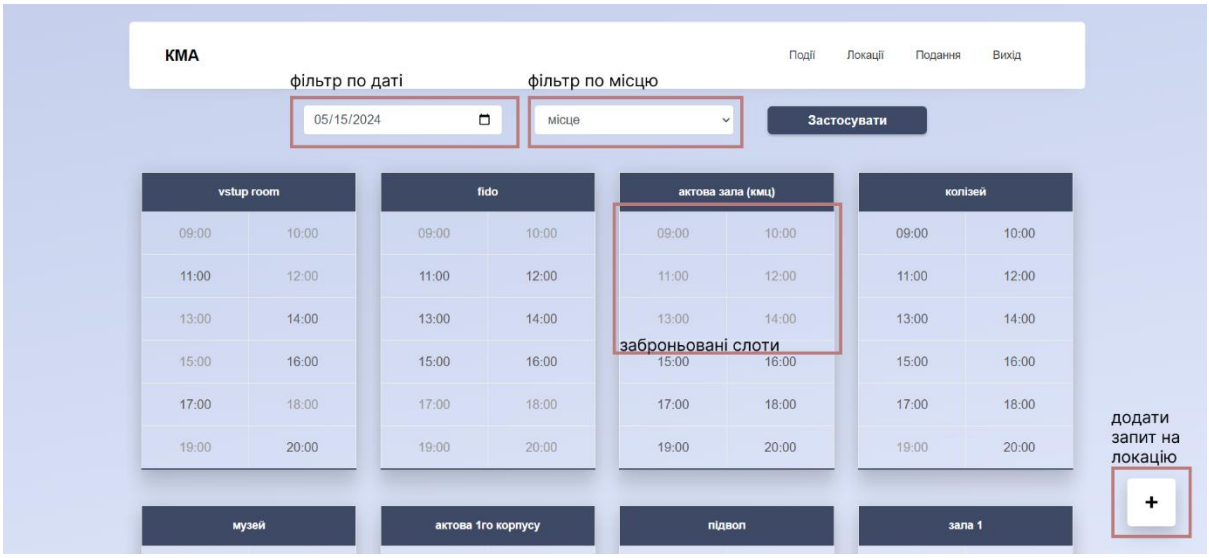


Рисунок 5. Сторінка локацій

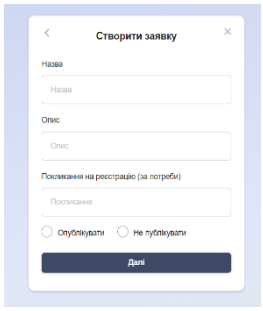


Рисунок 6.1. Створення подання (назва, опис, покликання, публікація)

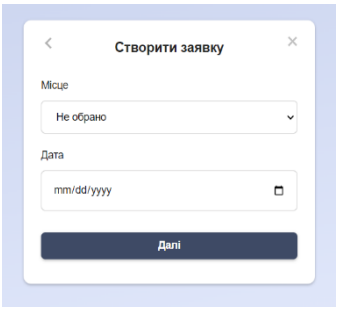


Рисунок 6.2. Ствренння подання (місце та дата)

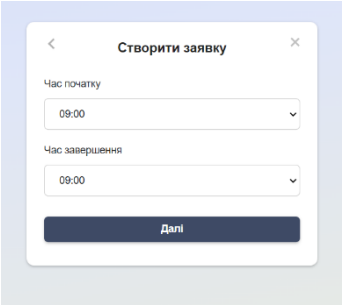


Рисунок 6.3. Створення подання (дата початку та завершення)

5.4 Підтвердження подання та перегляд своїх подій

Сторінка підтвердження подання доступна тільки користувачам з роллю “admin”. На сторінці видні всі подання, які були створені на приміщення, за якими закріплений користувач (див. Рис. 7). Для визначення чи опрацьована заявка можна переглянути детальну інформацію про подання (див. Рис. 8) або є ідентифікатор у правому кутку блоку подання:

- ⏳ - позначає, що заявка ще не була розглянута;
- ✗ - подання було відхилено;

- ✓ - подання було підтверджено.

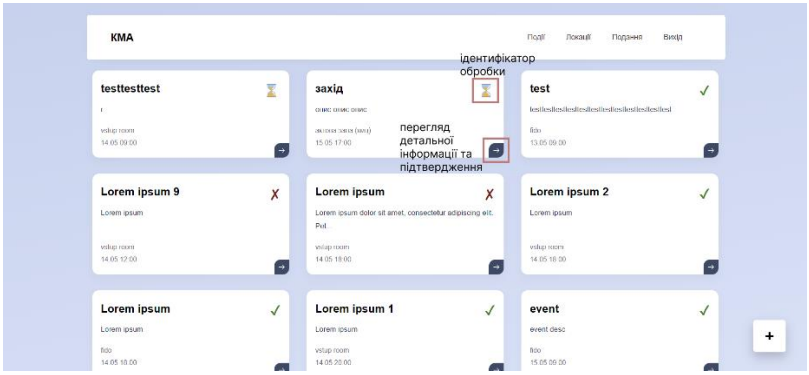


Рисунок 7. Перегляд заходів для підтвердження

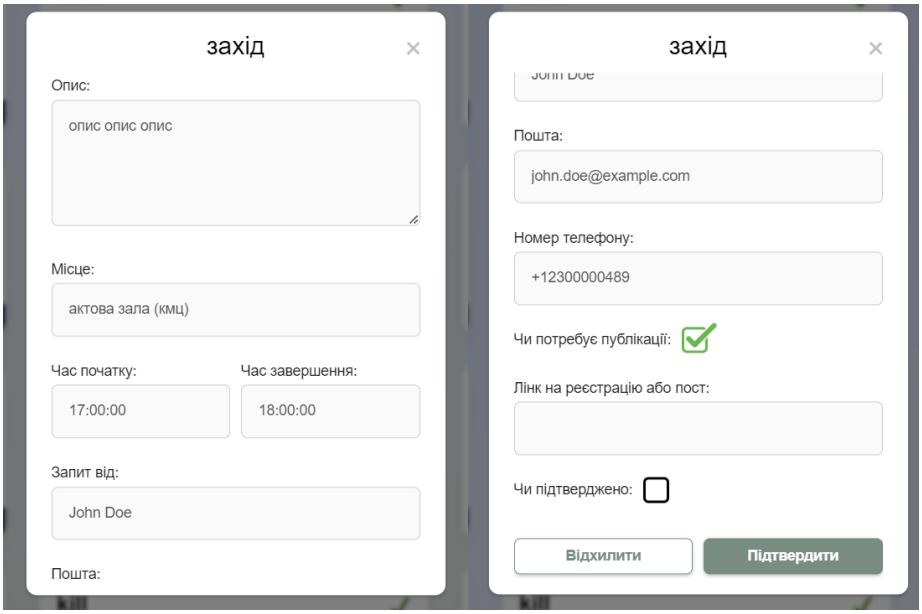


Рисунок 8. Підтвердження подання

4.5 Додавання, видалення та редагування користувача

Додавання, видалення та редагування користувача доступні лише юзерам з роллю “superuser”. Такий користувач може переглянути всіх користувачів та відсортувати їх по параметрам: ID, ПІБ, Email, Роль – натисканням на заголовок колонки (див. Рис. 9). Окрім того, він відповідає за додавання нових користувачів (див. Рис. 10) та редагування полів: ПІБ,

email, номер телефону (див. Рис. 11). При додаванні користувача можна згенерувати пароль, який буде складатися з 8 рандомних символів.

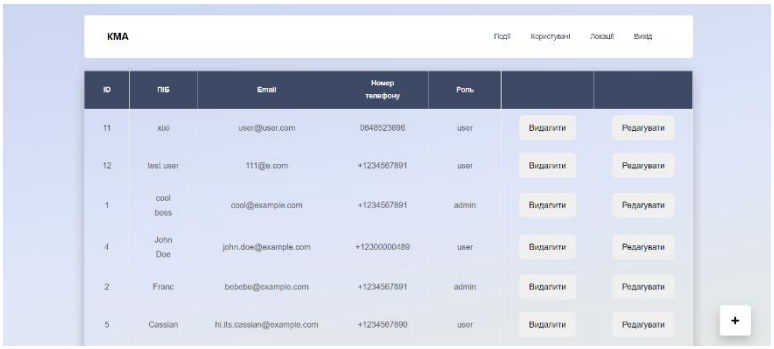


Рисунок 9. Перегляд користувачів

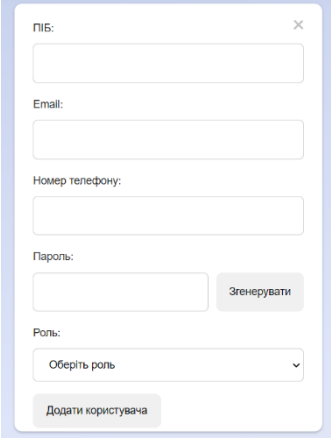


Рисунок 10. Додавання користувача

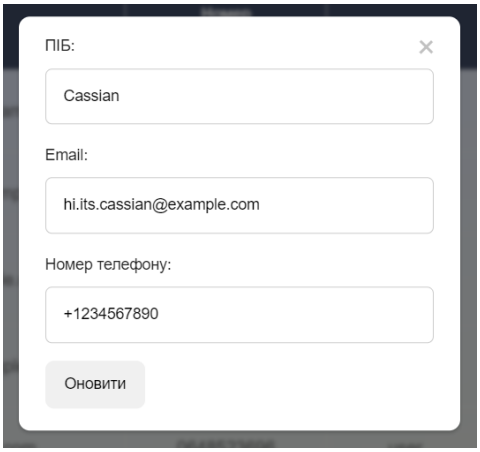


Рисунок 11. Редагування користувача

4.6 Додавання, видалення та редагування локації

Додавання, видалення та редагування локації доступні тільки користувачам з роллю “superuser”. Такому юзеру доступні:

- Видалення адміністратора локації, у випадку якщо він не один (див. Рис. 12);

- Додавання адміністратора: можна обравши керівника з переліку юзерів, які мають роль “admin” (див. Рис. 13);
- Натиснувши на “+” в правому кутку, сторінки можна додати нове приміщення доєднавши до нього відповідального користувача (адміністратора) (див. Рис. 14).

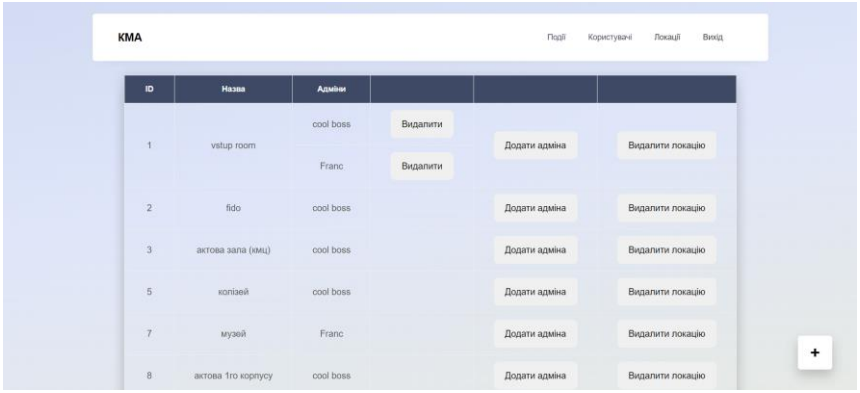


Рисунок 10. Перегляд існуючих приміщень

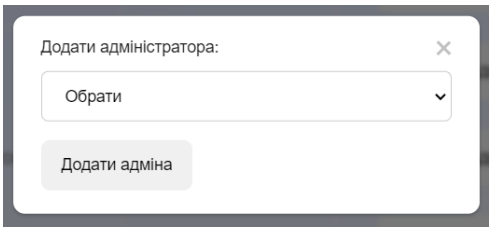


Рисунок 11. Додати адміністратора

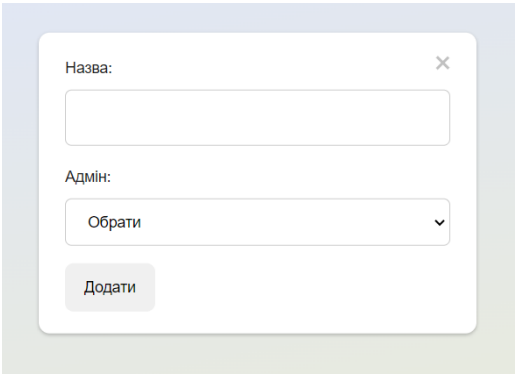


Рисунок 12. Додати локацію

Висновки

В цій курсовій було розглянуто створення вебзастосунку з використанням Node.js (TypeScript) та React. Також було досліджено роботу з базою даних PostgreSQL з використанням TypeORM. Результатом розробки став застосунок для покрашення процесу створення та підтвердження подання на приміщення. Окрім того, для полегшення розробки застосунку було використано фреймворк Nest.js в поєднання з Express.js.

Оглянувши існуючі рішення, було виявлено, що ці застосунки не надають потрібного функціоналу потрібного для комфортної роботи користувача, а забезпечують тільки частину функцій необхідних для покриття всіх потреб.

Було визначено, що вебзастосунок є кращим рішенням для поставленої задачі, оскільки він є легшим для використання користувачем та не потребує додаткового встановлення чи оновлення.

Функціонал застосунку дозволяє використовувати його як єдину платформу для створення подання, тобто відпадає потреба використання додаткових програм.

Джерела

1. Introduction to the Outlook Calendar. URL: https://support.microsoft.com/en-us/office/introduction-to-the-outlook-calendar-d94c5203-77c7-48ec-90a5-2e2bc10bd6f8
2. Загальні відомості про Microsoft Forms. URL: https://support.microsoft.com/uk-ua/topic/%D0%B7%D0%B0%D0%B3%D0%B0%D0%BB%D1%8C%D0%BD%D1%96-%D0%B2%D1%96%D0%B4%D0%BE%D0%BC%D0%BE%D1%81%D1%82%D1%96-%D0%BF%D1%80%D0%BE-microsoft-forms-6b391205-523c-45d2-b53a-fc10b22017c8
3. Raphael Schaad. Introducing Notion Calendar: an integrated calendar for work and life. URL: https://www.notion.so/blog/introducing-notion-calendar
4. What Is TypeScript? A Comprehensive Guide. URL: https://kinsta.com/knowledgebase/what-is-typescript/
5. Introduction to Node.js | Node.js Documentation. URL: https://nodejs.org/en/learn/getting-started/introduction-to-nodejs
6. The Node.js Event Loop | Node.js Documentation. URL: https://nodejs.org/en/learn/asynchronous-work/event-loop-timers-and-nexttick
7. Why use Node JS for your enterprise project? URL: https://www.miquido.com/blog/why-use-node-js/
8. Introduction to Nest.js — A Framework for Node.js. URL: https://medium.com/google-developer-indonesia/introduction-to-nest-js-a-framework-for-node-js-1d5d3f60afef
9. Nest.js Documentation. URL: https://docs.nestjs.com/

10. NestJS: Episode 6- Dependency Injections. URL: https://medium.com/@developerwhoismean/nestjs-episode-6-dependency-injections-d06abaaa5bae
11. Express.js Tutorial. URL: https://www.geeksforgeeks.org/express-js/
12. What is Object Relational Mapping? URL: https://www.educative.io/answers/what-is-object-relational-mapping
13. Mohammad Abdul Alim. An overview of TypeORM. URL: https://dev.to/alim1496/an-overview-of-typeorm-1cp2
14. Weerayut Teja. Integrating Databases in NestJS: Sequelize vs. TypeORM. URL: https://javascript.plainenglish.io/integrating-databases-in-nestjs-sequelize-vs-typeorm-bc476d3cac47
15. PostgreSQL Documentation. About. URL: https://www.postgresql.org/about/
16. Tim Davidson. Why Use PostgreSQL For Your Next Project? URL: https://cleancommit.io/blog/why-use-postgresql-for-your-next-project/
17. Why Choose React For Web Development in 2024. URL: https://www.geeksforgeeks.org/why-choose-react-for-web-development/
18. React: Documentation. URL: https://uk.legacy.reactjs.org/
19. Getting started with React | MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/React_getting_started
20. Built-in React Hooks. URL: https://react.dev/reference/react/hooks
21. New Generation Frontend Development Experience — Vite. URL: https://medium.com/bursa-bili%C5%9Fim-toplulu%C4%9Fu/new-generation-frontend-development-experience-vite-a4c7b264a157
22. Feature Overview | React Router Documentation. URL: https://reactrouter.com/en/main/start/overview

23. Getting Started | Axios Documentation. URL: https://axios-http.com/docs/intro
24. Asynchronous HTTP Requests in JavaScript: Axios vs. Fetch. URL: https://medium.com/@supraja_miryala/asynchronous-http-requests-in-javascript-axios-vs-fetch-0afdd87b22b0
25. node.bcrypt.js | NPM. URL: https://www.npmjs.com/package/bcrypt
26. How to Hash Passwords with bcrypt in Node.js. URL: https://www.freecodecamp.org/news/how-to-hash-passwords-with-bcrypt-in-nodejs/
27. Migrations | TypeORM Docs. URL: https://orkhan.gitbook.io/typeorm/docs/migrations#migrations