Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Автоматики і комп'ютеризованих технологій
(повна назва)

Кафедра Комп'ютерно-інтегрованих технологій, автоматизації та робототехніки
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Розроблення системи автоматизованого додавання промислового обладнання на основі технічних характеристик
(тема)

Виконав:
студент IV курсу, групи АКТАКІТ-20-3
Куварзін Олексій Ігорович
(прізвище, ініціали)

Спеціальність 151 Автоматизація та комп`ютерно-інтегровані технології
(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма Автоматизація та комп`ютерно-інтегровані технології
(повна назва освітньої програми)

Керівник ст. вик. Гурін Д.В.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Невлюдов І.Ш.
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет Автоматики і комп'ютеризованих технологій

Кафедра Комп'ютерно-інтегрованих технологій, автоматизації та робототехніки

Рівень вищої освіти перший (бакалаврський)

Спеціальність 151 Автоматизація та комп`ютерно-інтегровані технології
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Автоматизація та комп`ютерно-інтегровані технології
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)

«____»_____ 2024 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Куварзін Олексій Ігорович
(прізвище, ім'я, по батькові)

1. Тема роботи Розроблення системи автоматизованого додавання промислового обладнання на основі технічних характеристик

затверджена наказом університету від 03.06.2024 р. № 544 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 21 06 2024 р.

3. Вихідні дані до роботи HTML, CSS, JavaScript, Express, PostrgreSQL, Node.js

4. Перелік питань, що потрібно опрацювати в роботі Вступ, Аналіз предметної області, Проектування автоматизованої системи, Розробка системи автоматизованого додавання промислового обладнання, Висновки.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Демонстраційний матеріал, представлений у форматі презентації PowerPoint (*.ppt). – с. Формату А4.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Аналіз предметної області	10.05.2024	виконано
2	Проектування автоматизованої системи	27.05.2024	виконано
3	Розробка системи автоматизованого додавання промислового обладнання	08.06.2024	виконано
4	Опис посібника із користування системою	10.06.2024	виконано
5	Оформлення пояснювальної записки	11.06.2024	виконано
6	Подання роботи на рецензію	19.06.2024	виконано
7	Подання роботи на підпис зав.кафедри	21.06.2024	

Дата видачі завдання _01_ _05_____2024 р.

Студент _____
(підпис)

Керівник роботи _____ ст. вик. Гурін Д.В.
(підпис) (посада, прізвище, ініціали)

Я, як студент(ка) ХНУРЕ, розумію і підтримую політику закладу із академічної доброчесності. Я не надавав(ла) і не одержував(ла) недозволену допомогу під час підготовки кваліфікаційної роботи. Використання ідей, результатів і текстів інших авторів мають посилання на віповідне джерело.

" 21" червня 2024 р.  Куварзін О.І.

РЕФЕРАТ

Пояснювальна записка: 87 с., 21 рис., 2 дод., 21 джерело.

СИСТЕМА АВТОМАТИЗОВАНОГО ДОДАВАННЯ, РОЗРОБКА САЙТУ, ПРОГРАМУВАННЯ.

Мета роботи – зменшення часу на додавання промислового обладнання шляхом створення системи автоматизованого додавання промислового обладнання на основі технічних характеристик.

Об’єкт розробки – процес додавання промислового обладнання.

Предмет розробки – програмне забезпечення, яке дозволяє адміністраторам ефективно та швидко додавати промислове обладнання відповідно до технічних вимог та характеристик.

В даній кваліфікаційній роботі приведено аналіз предметної області, було розглянуто системи додавання промислового обладнання. В результаті аналізу було обрано систему додавання як веб-додаток.

Розроблено структурні макети, розроблено алгоритми додавання та на їх основі розроблено діаграму бази даних. Для перевірки функціональності та демонстрації роботи програмних модулів було використано браузер Google Chrome.

На основі підготовлених структурних макетів та діаграми бази даних було створено веб-додаток автономного додавання обладнання. Розроблено алгоритм роботи та програмне забезпечення для керування базою даних.

Наведено опис посібника із користування системою, яке розкрило реальне користування веб-додатком.

ABSTRACT

The explanatory note contains: 84 p., 21 drawings, 2 pp, 21 sources.

SYSTEM OF AUTOMATED ADDING, WEBSITE DEVELOPMENT, PROGRAMMING.

The aim of the work is to create a system of automated addition of industrial equipment based on technical characteristics.

The object of development is a system of automated addition of industrial equipment.

The subject of development is software that allows administrators to add industrial equipment efficiently and quickly according to technical requirements and characteristics.

In this qualification work, the analysis of the subject area is presented, the system of adding industrial equipment was considered. As a result of the analysis, the addition system was chosen as a web application.

Structural layouts were developed, addition algorithms were developed, and a database diagram was developed based on them. The Google Chrome browser was used to test the functionality and demonstrate the operation of the software modules.

Based on the prepared structural layouts and the database diagram, a web application for offline hardware addition was created. Work algorithm and database management software were developed.

A description of the system user guide is given, which revealed the real use of the web application.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

КІТАР – комп’ютерно-інтегрованих технологій, автоматизації та робототехніки;

НДР – науково-дослідна робота;

ПЗ – програмне забезпечення;

ХНУРЕ – Харківський національний університет радіоелектроніки;

API – Application Programming Interface, інтерфейс програмування;

CSS – Cascading Style Sheets, спеціальна мова стилю сторінок;

HTML – HyperText Markup Language, стандартизована мова розмітки документів;

JS – JavaScript, мова програмування;

JWT – JSON Web Token, стандарт токена доступу;

ORM – Object-relational mapping;

SASS – Syntactically Awesome Style Sheets, препроцесор CSS;

SCSS – Sally Cascading Style Sheets, препроцесор CSS;

SQL – Structured Query Language, мова структурованих запитів;

TS – Typescript, мова програмування

ВСТУП

В сучасному цифровому віці інтернет став не лише невичерпним джерелом інформації, але й важливим інструментом для комунікації та бізнесу. Зараз створення веб-сайтів – це не просто процес розробки, але й мистецтво, що вимагає глибокого розуміння технічних аспектів, дизайну та користувацьких потреб. Спеціалісти з автоматизації займають центральне місце в цьому процесі, оскільки вони не лише розуміють потреби клієнта, але і забезпечують ефективне впровадження автоматизованих рішень для оптимізації функціональності та продуктивності веб-платформ.

Розробка сайтів стає все більш складною і вимагає інтеграції різноманітних технологій для забезпечення найвищої якості продукту. Спеціалісти з автоматизації, завдяки своєму унікальному баченню та навичкам, відіграють ключову роль у впровадженні автоматизованих процесів, що допомагають прискорити розробку та підвищити якість веб-сайтів. Вони поєднують технічну експертизу зі здатністю розуміти потреби користувачів, створюючи продукти, які не лише функціональні, але й зручні у використанні та естетично привабливі.

Створення системи автоматизованого додавання промислового обладнання прискорює розвиток підприємств автоматизуючи процес додавання обладнання, роблячи його доступним та ефективним для всіх користувачів. Сайт дозволяє легко та швидко додавати потрібне обладнання, не витрачаючи зайвих коштів та найму спеціалістів з підтримки бази даних. Клієнт, у свою чергу, легко зможе отримати список обладнань та детальну інформацію про них з основної бази даних.

Таким чином метою кваліфікаційної роботи є створення системи автоматизованого додавання промислового обладнання на основі технічних характеристик, що спростить процес додавання обладнання для промислових підприємств та підвищить їхню ефективність.

Об’єкт розробки – системи автоматизованого додавання промислового обладнання.

Предмет розробки – програмне забезпечення, яке дозволяє адміністраторам ефективно та швидко додавати промислове обладнання відповідно до технічних вимог та характеристик.

Для досягнення поставленої мети необхідно вирішити такі завдання:

– провести аналіз типових сайтів, що спеціалізуються на додаванні обладнання;

– розробити макет інтерфейсу користувача;

– розробити алгоритми додавання;

– розробити бази даних;

– забезпечити безпеку даних;

– зібрати макет веб-додатку;

– підключити базу даних до інтерфейсу користувача;

– описати посібник із користування веб-додатком;

Кваліфікаційна робота виконана згідно ДСТУ 3008 – 15 [1], використовуючи навчальний посібник з дипломного проекту [2] та методичні вказівки [3].

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Введення у предметну область

Промислове обладнання є основним чинником виробничого процесу майже у всіх сферах послуг. Успіх бізнесу залежить від правильного обладнання та того, як воно використовується, від фабрики до транспортної системи. Від того, наскільки швидкі, правильні та надійні ці машини, залежить, наскільки добре ведеться бізнес і наскільки якісні речі.

Але придбати відповідне промислове обладнання – це важко, і це пов’язано з багатьма проблемами. Компанії повинні думати про купу речей, наприклад, наскільки якісний продукт, скільки у них грошей, коли він потрапить до клієнта та наскільки якісним є обслуговування. набагато важче приймати рішення, коли вам доводиться думати про всі ці речі.

В останній час не багато систем які можуть допомогти з додаванням обладнання. Найбільш частіше є тільки фахівці, які вручну керують даними. У зв’язку з цим підприємства вдаються до допомоги фахівців, які зможуть реалізувати таку роботу.

Перше питання з фахівцями – цих фахівців для початку потрібно знайти, а після чого стабільно платити не маленьку суму, щоб вони постійно підтримували оновлення даних. Оскільки в подібних сферах дуже велика конкуренція, особливо для підприємців-початківців, потрібно шукати інноваційні рішення або покращувати якість без зміни плати, щоб клієнт купував продукт або користувався послугами саме у цієї компанії.

Друге питання з людським фактором – не всі фахівці знають абсолютно всі галузі та не зможуть надати професійні послуги поглиблено у певній сфері, скоріше знають лише в основних рисах та придумати щось нове потрібен зовсім інший фахівець. Витрати на такі етапи дуже виростають.

Впровадження спеціалізованих рішень та інтернет-платформ для додавання обладнання дозволяє підприємцям або підприємствам ефективно без зайвих витрат часу та ресурсів розвивати підприємство, покращувати продукт, роблячи ціну більш конкурентоспроможною.

Оскільки технології та ринки швидко змінюються, нам потрібно більше інструментів для додавання правильного обладнання. Саме з цим допоможе створення такого веб-сайту, який може автоматично додавати промислове обладнання, виходячи з того, яке саме потрібно додати та що є в наявності.

Один зі способів вирішити цю проблему – створити веб-сайт, який може автоматично додавати правильні машини для бізнесу на основі того, що їм потрібно та як вони працюють. Дослідження спрямоване на додання нового обладнання, на те, як працюють машини та чого хочуть люди, а також на створення та використання системи програмного забезпечення, яка підходить.

1.2 Аналіз існуючих рішень

Сьогодні на ринку доступно не багато онлайн-платформ, які спеціально розроблені, щоб допомогти у процесі додавання промислового обладнання. Вони мають різні сфери знань, наприклад, зосереджені на певних видах обладнання, як-от обладнання для виробництва речей або для їжі, або вони можуть надавати різні рішення для різних типів бізнесу.

Є багато онлайн-інструментів і програм, які дозволяють вибирати та порівнювати різні пристрої на основі їх функцій, цін і графіків доставки. Однак функції та їх доступність значно відрізняються в різних рішеннях.

Деякі галузі можуть надавати перевагу деяким онлайн-платформам більше, ніж іншим, тоді як інші можуть бути маловідомими або невикористовуваними. Популярність рішення серед користувачів може бути суттєвим фактором у визначенні того, чи підходить воно для конкретного бізнесу.

Оцінюючи поточні рішення, наприклад як веб-сторінка Thomasnet [4], дуже важливо враховувати їхні переваги та недоліки. Апаратне забезпечення може бути неправильно описане або не працювати належним чином на деяких платформах. З деякими продуктами може бути важко працювати або вимагають багато уваги від адміністратора.

Порівняння запропонованого проекту з поточними рішеннями дозволяє нам оцінити їхні переваги та недоліки. Це дає вам змогу розпізнати області для вдосконалення та креативності у створенні нового Інтернет-сайту для додавання промислового обладнання.

Аналіз існуючих варіантів програмного забезпечення для додавання промислового обладнання показує, що більшість із них мають обмеження та недоліки. Створення нового веб-сайту, щоб вирішити ці проблеми та запропонувати підприємствам простий та ефективний спосіб додавання обладнання, є вирішальним кроком для підвищення ефективності бізнесу на сучасному ринку.

1.3 Технічний аналіз вимог до інтернет-сайту

Цей сайт потрібен включати в себе декілька факторів для зручності користувача.

Привабливий та зручний веб-інтерфейс, який буде легко розуміти та використовувати навіть для недосвідчених користувачів. Система пошуку промислового обладнання, де користувачі повинні мати можливість здійснювати пошук промислового обладнання за різними критеріями, такими як тип, розмір, технічні характеристики тощо. Після пошуку користувачеві необхідно показати методи, детальний поетапний список необхідного обладнання. Після цих етапів користувач зможе фільтрувати отриманий результат за різними параметрами, такі як ціна, складність, популярність тощо. Також для зручності користувача, необхідно додати сортування за параметрами, ціна, складність тощо.

Для систем додавання на сайті має бути база даних, яка зберігатиме інформацію про обладнання, їх параметри, характеристики тощо. Головною задумкою є наявність системи додавання нових розділів, товарів, змін характеристик, ціни та іншої інформації про обладнання.

Одні із важливих критеріїв для сайту – його адаптивність під різні пристрої, а також швидкодія. Будь-який користувач віддасть перевагу якісному продукту, який коректно відображатиметься навіть на смартфоні і швидко завантажуватиметься, не витрачаючи зайвий час, трафік у користувачів та адміністраторів.

1.4 Аналіз вимог користувачів

Щоб зрозуміти й допомогти користувачам веб-сайту, які шукають промислове обладнання, потрібно розділити їх на групи на основі їхніх інтересів і потреб. Для створення веб-сайту вкрай важливо враховувати різноманіття користувачів та їхні особливі вимоги, щоб гарантувати найкращий досвід користувача.

Великі компанії, які мають значну присутність у промисловому секторі, можуть шукати обладнання та системи автоматизації для покращення своїх виробничих процесів, особливо для виробництва великої кількості товарів. Обладнання має відповідати суворим стандартам щодо якості, продуктивності та надійності.

Натомість малі та середні підприємства можуть мати обмежений бюджет на обладнання та шукати рентабельні та ефективні варіанти. Обладнання має відповідати їхнім вимогам і дозволяти їм покращувати виробничі процеси.

Інженери та техніки можуть знайти вичерпні технічні відомості про обладнання, включаючи його специфікації та застосування. Вони можуть захотіти вдосконалити та модернізувати свої поточні інструменти.

Сайт може допомогти продавцям і покупцям промислового обладнання зв’язатися та продемонструвати свою продукцію. Для них вкрай важливо володіти здатністю вміло демонструвати свою продукцію та захоплювати цільову аудиторію. Для ефективного дизайну веб-сайту вкрай важливо визнати, що різні групи користувачів мають різні вимоги, бажання та цілі. Розуміючи ці відмінності, потрібно налаштувати функції та макет сайту відповідно до потреб кожної групи.

Інтернет-сайт з додаванням промислового обладнання матиме функцій, які роблять його зручним і швидким у використанні.

Користувачі повинні мати можливість знайти потрібне обладнання на основі різних факторів, таких як тип обладнання, характеристики, тощо, не витрачаючи час і зусилля. Користувачі повинні порівняти характеристики та вартість різноманітного обладнання, щоб вирішити, яке з них найкраще для них. Користувачі повинні мати доступ до вичерпної інформації про обладнання, включаючи характеристики, фотографії та описи, щоб допомогти їм прийняти обґрунтоване рішення про покупку. А адміністратору в свою чергу повинні мати простий та швидкий спосіб додавання цього обладнання. Додавання цього обладнання також необхідно оптимізувати, щоб людина не робила зайвих рухів тіла.

Метою цих вимог є створення інструменту, який буде простим і корисним для користувачів та адміністраторів, щоб вони могли швидко та якісно знаходити, вибирати та додавати промислове обладнання.

Щоб розробити інтернет-сайт з додаванням промислового обладнання, необхідно зробити інтерфейс простим і зручним.

Хороший дизайн має бути простим для розуміння та використання, і не мати занадто багато речей, які відволікають користувача від вибору необхідного обладнання. Користувач повинен мати можливість легко орієнтуватися в інтерфейсі, незалежно від того, чи відвідує він сайт вперше.

Після цього вкрай важливо гарантувати просту навігацію, щоб користувачі могли швидко знаходити необхідну інформацію та переходити між різними розділами сайту, не докладаючи зайвих зусиль. Розташування навігаційних компонентів має бути таким, щоб вони були легкодоступними та придатними для використання на пристроях із маленьким екраном. Крім того, дуже важливо брати до уваги адаптивний дизайн, оскільки значна кількість користувачів Інтернету отримує доступ до мережі через мобільні пристрої.

Інтерфейс повинен адаптуватися до різних екранів і пристроїв, щоб користувачі могли використовувати його легко і комфортно, де б вони не були. Сайт для вибору промислового обладнання має бути простим у використанні, швидким і відповідати очікуванням користувачів.

Щоб переконатися, що сайт досить швидкий і стабільний для вибору промислового обладнання, необхідно дотримуватися деяких важливих правил. Ці правила допоможуть зробити користувачів задоволеними.

Сайт має бути швидким і ефективним. Коли веб-сайт завантажується надто довго, користувачам може стати нудно, і вони погано проводять час. В результаті веб-сайт має бути розроблений таким чином, щоб він швидко завантажувався, що передбачає зменшення кількості ресурсів, які використовуються кожною сторінкою, збереження часто використовуваного вмісту в пам’яті та використання високопродуктивних серверів.

Крім того, стабільність сайту має вирішальне значення для забезпечення безперервної роботи. Якщо система не стабільна, це може спричинити проблеми та збої, які зроблять користувачів незадоволеними та зашкодять іміджу компанії. Отже, сайт має бути побудований із використанням надійних технологій і архітектури, і його слід часто перевіряти на стабільність і довговічність.

Основна мета дизайну сайту – зробити його швидким і стабільним, що принесе задоволення користувачам і допоможе успішному розвитку. Також описати функції та взаємодію з клієнтом без зайвих перезавантажень сторінок, що відображаються.

1.5 Аналіз технічних характеристик промислового обладнання

Оцінка технічних характеристик промислового обладнання – важливий крок у створенні онлайн-платформи для додавання обладнання. Це також дає повне розуміння технічних характеристик і функцій кожного пристрою, що допомагає користувачам приймати обґрунтоване рішення. Вивчаючи різні аспекти, такі як розмір, потужність, продуктивність, енергоефективність і параметри налаштування, можна краще зрозуміти, наскільки добре обладнання відповідає потребам і вимогам своїх користувачів.

Перед початком аналізу слід ретельно вивчити технічні характеристики кожного типу обладнання, щоб переконатися, що вони відповідають потребам користувачів. Як приклад, у випадку підприємств, яким потрібно виробляти багато товарів швидко й точно, такі фактори, як швидкість роботи машин, наскільки добре вони виконують свою роботу, як довго вони працюють і наскільки вони послідовні, можуть мати вирішальне значення.

Після цього має бути проведено детальне вивчення технічних характеристик різних типів обладнання, що дозволить нам визначити переваги та недоліки кожного. Цей аналіз може включати вивчення змінних, які впливають на продуктивність, якість продукції, операційні витрати та інші важливі аспекти.

Потрібно ретельно розглядати новітні технології та інновації в галузі промислового обладнання. Запровадження нових досягнень може значно підвищити ефективність виробничих процедур і підвищити конкурентоспроможність бізнесу. Як наслідок, нам потрібно вивчати та вивчати нові розробки та ідеї в цій галузі.

Після аналізу важливо визначитися з найкращим обладнанням, яке може задовольнити потреби та вимоги користувачів. Ретельно вивчивши технічні аспекти Інтернет-сайту, можна гарантувати його ефективну та успішну роботу при виборі доданого промислового обладнання.

1.6 Аналіз методів автоматизації підбору обладнання

Вивчення методів автоматизації процесу пошуку та додавання промислового обладнання передбачає комплексне дослідження та оцінку різних методів автоматизації процесу пошуку та вибору.

Початковий підхід до споглядання – це автоматичне визначення параметрів. Адміністратор може встановити параметри для вибору обладнання за допомогою фільтрів і алгоритмів. Важливо розуміти значення параметрів, які визначаються користувачем, і включати їх у процедуру пошуку.

Системи рекомендацій є другим методом. Система вже є більш персоналізованим підходом, який використовує алгоритми додавання та аналіз даних, щоб пропонувати обладнання користувачам на основі наявності певного обладнання та популярності серед інших користувачів. Враховуючи індивідуальні переваги та вимоги кожного користувача, ця функція покращує процес пошуку, додавання та робить його ефективнішим.

Третій спосіб — підключення до баз даних виробників. Цей метод використовує дані, надані виробниками обладнання, для автоматичного оновлення та розширення асортименту продуктів на веб-сайті. Підключившись до баз даних виробників, ми зможем отримати доступ до найновішої інформації про характеристики, вартість і асортимент товарів без необхідності оновлювати її вручну.

Під час прийняття рішення про тип автоматизації важливо враховувати унікальні вимоги та особливості передбачуваних користувачів, а також технічну можливість обраного методу. Щоб гарантувати найвищий рівень ефективності та зручності використання, дуже важливо ретельно оцінити всі можливі методи та вибрати той, який найбільше відповідає потребам і очікуванням користувачів.

Провівши аналіз методів – можна зробити висновок, що перший метод є кращим, оскільки має ряд переваг, а також незалежність від інших факторів та баз даних.

Однією з головних переваг цього методу є можливість швидко та ефективно знайти та додати потрібне обладнання, не витрачаючи час на перегляд непотрібних варіантів. Користувачам не потрібно докладно досліджувати всі доступні опції, вони можуть просто встановити необхідні параметри та отримати відповідні результати.

У порівнянні з іншими методами, де рекомендації можуть базуватися на історії перегляду чи популярних товарах, автоматизований пошук за параметрами забезпечує точніші результати, оскільки він враховує конкретні вимоги користувача.

Цей метод дозволяє користувачам охоплювати весь асортимент товарів, оскільки він не обмежується працездатністю іншої бази даних, договором взаємодії з нею, або лише наявними на складі товарами, популярними пропозиціями.

Автоматизоване додавання за створеними параметрами ефективне тоді, коли у адміністратора є конкретні типи, бренди обладнання, тощо, і він додає товар, що точно відповідає цим вимогам.

Отже, автоматизоване додання за параметрами є надійним, незалежним та ефективним методом для користувачів, які мають чіткі вимоги та додають точне відповідне обладнання, що робить його бажаним вибором на інтернет-сайтах для додавання промислового обладнання.

1.7 Аналіз технологій розробки сайту

Розвиток в сфері інтернету в останні роки дуже змінився та відкрив безмежні можливості для створення та розгортання веб-сайтів різного призначення. Веб-сайти використовують для особистих портфоліо, комерційних проектів, навчання, розваг та навіть державного управління. Технології розробки веб-сайтів стали ключовим фактором у забезпеченні їхньої ефективності, безпеки та зручності використання. Це важливі компоненти створення будь-

якого сучасного сайту. Вони визначають якість, функціональність та зовнішній вигляд веб-проекту. Застосування відповідних технологій дозволяє створювати сайти, які відповідають потребам користувачів та вирішують конкретні завдання бізнесу або особистих цілей. Ключові компоненти технологій розробки веб-сайтів можна розділити на декілька частин – Frontend розробка, Backend розробка, бази даних, препроцесори, системи керувань версіями, різноманітні інструменти та іноді API (Application Programming Interface).

Згідно з електорнним ресурсом Nure [5], Веб-розробник або Developer – багатопрофільний фахівець, який займається програмно-адміністративною та візуальною частиною сайту, програми чи мобільного додатку. Такого розробника, зазвичай, називають full-stack розробником. До речі, розробника, який займається розробкою програмно-адміністративної частини, називають back-end – розробником, а розробкою візуальної частини сайту, програми або додатка – front-end – розробником. Та найчастіше, цих фахівців називають скорочено – бекендом або фронтендом.

Frontend Development, розробка клієнтської частини, включає в себе HTML для структури сайту, CSS для стилізації та оформлення, JavaScript для динамічності на інтерактивності на стороні користувача.

Згідно з електронним ресурсом MDN Web, термін HTML визначається як [6]: код, який використовується для структурування та відображення веб-сторінки та її контенту. Наприклад, контент може бути структурований всередині множини параграфів, маркованих списків або з використанням зображень та таблиць даних. Як видно з назви, ця стаття дасть вам базове розуміння HTML та його функцій.

Згідно з методичними вказівками ВНУ [7]: CSS – Cascading Style Sheets – технологія опису зовнішнього виду документа, який написаний мовою розмітки; це набір параметрів форматування, що застосовується до елементів документа, щоб змінити їх зовнішній вигляд.

Іншими словами, CSS доповнює HTML, та робить веб-сайт привабливим для користувача. Використовується для стилізації HTML-документів і задає їх зовнішній вигляд, такий як кольори, шрифти, розміри, відступи та розташування елементів на сторінці. Дуже корисні для розробки будуть препроцесори CSS. Найбільш популярні – SCSS та SASS, вони дозволяють розробникам покращити продуктивність та підтримуваність CSS-коду, використовуючи розширений синтаксис. Вони надають багато корисних функцій, таких як змінні, вкладені селектори, міксіни та інтерполяція. Сама функція і призначення цих двох препроцесорів, як і решти, не сильно відрізняється один від одного, відмінність тільки з боку розробника – задоволення користування, методи написання. SCSS, Sassy CSS, використовує синтаксис, дуже схожий на звичайний CSS. Використовуються дужки ‘{}’ та крапки з комою ‘;’. Структура коді залишається такою ж, як і в CSS. Синтаксис SASS, Syntactically Awesome Stylesheets, більш компактним та менш вербозним, він не вимагає фігурних дужок та крапок з комою, але використовує відступи для визначення блоків коду. Однак для деяких розробників у середовищі розробки іноді не зовсім зручно працювати з відступами. Саме з цього приводу будемо використовувати саме SCSS.

JavaScript – мова програмування, яка використовується для реалізації динамічної поведінки на веб-сторінках. Вона широко використовується як для розробки клієнтської частини веб-додатків, так і для серверної розробки. Використовуючи JavaScript-сценарії, можна додати різні функції та ефекти, які дозволять вам забезпечити більш інтерактивний досвід для користувачів вашого сайту [8]. Також існують різноманітні бібліотеки для JavaScript – наприклад React.js. Вона використовується для створення користувацького інтерфейсу, дозволяючи розробити компоненти, які легко підтримувати та перевикористовувати. Для таких бібліотек існують фреймворки такі як Next.js – використовується для створення серверного рендерингу та статичного генерування веб-сторінок. Він надає можливість оптимізації швидкості завантаження сторінок та розширює функціональність React за допомогою

серверного рендерингу. В доповнення до JavaScript використовують TypeScript (TS) – строго типова мова програмування, у будівництві на JavaScript, витрачаючи вашу потребу технікі на будь-якому місці [9]. Вона дозволяє виявляти та виправляти помилки під час розробки, що полегшує великі проекти. Усе це використовується Frontend розробником для візуальної взаємодії з користувачем.

Backend розробка – це процес створення та підтримки серверної частини програмного забезпечення, яка забезпечує взаємодію клієнтської частини з базами даних, бізнес-логікою та іншими зовнішніми системами [10]. Вона відповідає за обробку запитів користувача та управління даними на серверному рівні. Вона забезпечує виконання бізнес-логіки, взаємодію з базою даних, обробку запитів від клієнтів та надання їм відповідей. У проекті з автоматизованого підбору промислового обладнання backend відповідає за обробку запитів користувачів, а також за роботу з базою даних для зберігання та обробки інформації про технічні характеристики обладнання. Іншими словами, frontend розробка – візуальна частина для користувача, а backend – невидима частина, яка відбувається на рівні сервера та надає певні дані по запиту. Для повного розуміння backend необхідно розібрати докладніше його функції.

Одна з його функцій – обробка запитів користувача, процес прийому обробки та відповіді на запити, які надходять від клієнтської частини додатку до серверної частини. У контексті проекту з автоматизованого підбору промислового обладнання, це означає, що бекенд отримує запити від користувача, які містять інформацію про їхні вимоги до обладнання, такі як технічні характеристики, потрібні функції, бюджет і т. д. Після отримання запиту бекенд аналізує його, виконує відповідні дії, такі як пошук, фільтрація або сортування в базі даних, щоб знайти відповідне обладнання, яке задовольнить потреби користувача. Після обробки запиту бекенд готує відповідь та надсилає її назад до клієнтської частини, де вона відображається користувачеві. Це дозволяє користувачам ефективно взаємодіяти з додатком та отримувати

необхідну інформацію про промислове обладнання відповідно до їхніх потреб і вимог. Більш детально по порядку про операції, які виконує бекенд:

– бекенд виконує запити до бази даних для отримання інформації про доступне обладнання. Може включати у себе отримання списків обладнання, його характеристики, ціни та інше;

– наступним кроком бекенд може оновлювати записи або додавати нові про обладнання в базу даних, коли користувач додає нові дані або реєструє нове обладнання;

– якщо потрібно, бекенд може видаляти непотрібні старі записи або дані про обладнання, коли вони більше не потрібні або застаріли.

У великих проектах також існує система аутентифікації та авторизації користувачів. Це два ключових компонента для безпеки в схожих додатках, які призначені для перевірки ідентичності користувачів і контролю доступу до різноманітних видів ресурсів. Особливо важливо це для забезпечення конфіденційності та захисту даних користувачів, якщо такий функціонал існує на платформі.

Аутентифікація в свою чергу є процесом перевірки користувача, який намагається отримати доступ до певної частини даних в системі. Зазвичай це введення логіна та пароля. Бекенд в свою чергу перевіряє надані користувачем облікові дані з даними, збереженими в базі даних, і визначає, чи має користувач доступ до даних, або ні.

Авторизація – процес визначення того, які ресурси та функціональні можливості має доступ користувач після успішної аутентифікації. Бекенд визначає права доступу користувача на основі його облікових даних та ролей, які він має, і вирішує, чи може користувач виконувати певні дії або отримувати доступ до певних ресурсів.

В контексті проекту з автоматизованого додавання промислового обладнання, аутентифікація та авторизація дозволяють забезпечити захист конфіденційності та цілісності даних користувачів, а також контролювати

доступ до функціональності системи. Наприклад, це може включати обмеження доступу до даних про обладнання тільки для авторизованих користувачів або надання спеціальних привілеїв адміністраторам системи.

Бекенд-логіка відповідає за обробку запитів користувачів та взаємодію з базою даних для забезпечення відповідності результатів пошуку та фільтрації до вимог та очікувань користувача. Бекенд може проаналізувати запит користувача та виконати необхідні дії для надання відповіді, яка відповість їх вимогам. Це може бути пошук та фільтрація обладнання за різними параметрами, такими як технічні характеристики, ціна, виробник тощо. Також у логікі є можливість аналіз даних для перевірки популярності певних типів обладнання, тенденції використання та інші фактори. Вона також може встановлювати правила та обмеження для користувача, наприклад, мінімальні та максимальні параметри обладнання, яке може бути вибране, або правила, що обмежують доступ до певних функцій в залежності від ролі користувача. Всі ці аспекти бекенд-логіки спрямовані на те, щоб забезпечити ефективну та зручну роботу системи для користувачів, враховуючи їхні потреби та вимоги щодо підбору промислового обладнання.

Як і у фронтенд частини, бекенд може використовувати фреймворки JavaScript, які зможуть додати більш ефективний та зручний функціонал для проекту. Наприклад, існує середовище виконання JavaScript на стороні сервера, Node.js. Воно допомагає виконати JavaScript поза браузером, що розширює можливості мови та надає можливість розробнику створювати ефективні та масштабовані веб-проекти та серверні програми.

Node.js має безліч переваг для реалізації, оптимізації великих додатків, ось основні з них.

Побудованність на движку JavaScript V8, який розробляється Google Chrome. Це робить його дуже швидким та ідеальним вибором для створення швидких та ефективних додатків.

Модель асинхронного програмування. Це означає, що розробники можуть створювати ефективні та чуйні додатки, спроможні одночасно обробляти безліч запитів. Node.js також відомий своєю легкістю та швидкістю роботи, що дозволяє ефективно використовувати ресурси сервера [11].

Широке співпрацювання з npm (Node Package Manager) – найбільший репозиторій пакетів для JavaScript, який дозволяє розробникам тисячі готових бібліотек для розширення функціональності їхніх додатків.

Node.js дуже універсальний, він може бути використаний для створення різноманітних типів додатків, від веб-серверів до веб-додатків, API та розширень браузера.

У цьому проекті Node.js може використовуватись для створення серверної частини додатку, реалізації бекенд-логіки, взаємодії з базою даних та надання API для клієнтської частини.

База даних може використовуватися для зберігання та управління інформацією про промислове обладнання, його технічні характеристики, ціни, виробників тощо. Вона також забезпечує можливість пошуку, фільтрації та відображення обладнання для користувачів, а також управління обліковими записами та правами доступу. База даних є основою для реалізації функціональності додатку та забезпечує ефективне управління та обробку даних.

Існує багато різних СУБД, систем управління базами даних, кожна з яких має свої особливості, переваги та недоліки. Одна з найбільш популярних – PostgreSQL. Це дуже потужна та довірена реляційна база даних з відкритим кодом. Вона відома своєю надійністю, розширюваністю та розширеними можливостями. PostgreSQL підтримує як реляційні, та нетрадиційні типи даних, включаючи JSON, XML та інші. Використовує в свою чергу мову запитів SQL для роботи з даними та має велику підтримку стандартів SQL. Безліч розширень та доповнень, які доступні для PostgreSQL, дозволяють розширювати її функціональність та використовувати її для різних завдань, від веб-додатків до аналітики даних.

Для цього проекту PostgreSQL може використовуватися для зберігання та управління інформацією про обладнання, його технічні характеристики, ціни, виробників та інші важливі дані. Вона також забезпечує можливість ефективного пошуку, фільтрації та відображення обладнання для користувачів, а також управління обліковими записами та правами доступу. База даних PostgreSQL є надійною та потужною основою для реалізації функціональності додатку та забезпечує ефективне управління та обробку даних.

Для зв’язку та взаємодією з базою даних є можливість використання ORM, Object-Relational Mapping. Це технологія програмування, яка дозволяє зв'язувати об'єктно-орієнтоване програмування з реляційним програмуванням. Sequelize, один з них, спростить взаємодію з базою даних та забезпечить безпечний доступ до даних через типізацію. Це дозволить мені легко використовувати базу даних, використовуючи звичайні Javascript або Typescript об’єкти, що відображають дані в базі даних. Sequelize наддасть можливість автоматично генерувати міграції для бази даних, що спрощує управління базою даних та зберігання її у синхронізованому стані з даними. Sequelize зможе генерувати типизовані класи та методи для виконання запитів до бази даних, що дозволить використовувати TypeScript для надійності та безпечності. Дуже великий плюс у тому, що Sequelize підтримує різні типи баз даних, включаючи PostgreSQL.

Sequelize може використовуватися для спрощення доступу до бази даних, забезпечення типізованих запитів та автоматичного управління міграціями бази даних. Його використання допоможе мені зосередитися на розробці функціональності додатку, замість рутинної роботи з базою даних.

2 ПРОЕКТУВАННЯ АВТОМАТИЗОВАНОЇ СИСТЕМИ

2.1 Розробка алгоритму роботи

Розробка схеми алгоритму роботи є одна з важливіших етапів створення системи автоматизованого додавання промислового обладнання на основі технічних характеристик. Це надає нам можливість значно покращити процес прийняття рішень, наглядно побачити порядок реалізації та обсяг роботи, що значно підвищить ефективність.

На етапі розробки веб-додатку можуть виникати помилки через людський фактор, проектування системи в свою чергу мінімізує шанс помилок. Завдяки йому, ми зможемо враховувати відразу усі технічні параметри, вимоги та ідеї, забезпечуючи найбільш точний вибір обладнання.

На основі проведеного аналізу предметної області, була розроблена схема процесу розробки системи автоматизованого додавання промислового обладнання.

Схему процесу розробки наведено на рис. 2.1.

Рисунок 2.1 – Схема процесу розробки системи автоматизованого додавання промислового обладнання

Спочатку необхідно розпочати з вирішення стеку технологій необхідних та зручних для використання у проекті, враховуючи реалізацію усіх необхідних функцій, оптимізацію та швидкодію сайту, підтримку та популярність. Після визначення стеку необхідно встановити всі необхідні NPM-пакети, що включає в себе бібліотеки та фреймворки, які будуть використовуватися в проекті.

Після цього проект розбивається на дві основні частини: візуальну та серверну. В рамках візуальної частини спершу створюється макет веб-додатка, що є основою для подальшої розробки. Далі формується скелет веб-додатка за допомогою HTML, який визначає структуру сторінок. Після цього додаються стилі, використовуючи CSS або його препроцесори SASS/SCSS, щоб забезпечити привабливий зовнішній вигляд та зручність використання.

Паралельно розробляється серверна частина. На цьому етапі створюється діаграма серверної частини, що визначає архітектуру та взаємодію між компонентами серверної сторони. Далі формується база даних, яка буде використовуватися для зберігання необхідної інформації.

До розробки серверної частини також входить формування моделей (models). Вони визначають структуру даних та взаємодіють з базою даних. Подалі будуть автоматично читати, змінювати записи в базі даних від відповідних дій користувача. Для реалізації динамічної взаємодії між користувачем та сервером існують важливі концепції, такі як контролери (controllers), маршрути (routes) та проміжне програмне забезпечення (middleware).

Контролери відповідають за обробки запиту від користувача і повернення відповіді на візуальну частину сайту користувачеві. Вони виступають посередниками між маршрутизаторами і логікою веб-додатка. Контролери отримають дані з запиту, викликають необхідні методи моделі для взаємодії з базою даних, таки як зміни, додання, видалення об’єкта, а також формують відповідь, яка надсилається користувачу.

Маршрути (routes) – відповідають за спрямування запитів на відповідні обробники (контролери). Вони визначають, який контролер і метод повинен обробляти кожен вхідний запит на основі HTTP методів GET, POST, тощо.

Проміжне програмне забезпечення (middleware) – функція, яка виконується під час обробки запитів, перш ніж вони досягнуть кінцевого маршруту або обробника. Воно використовується для виконання для різних завдань, таких як авторизація, реєстрація, обробка помилок, парсинг даних запиту і багато іншого. Парсинг даних – це процес взяття даних в одному форматі та перетворення їх в інший формат [12].

Після цього здійснюється підключення та налаштування взаємодії між Frontend та серверною частиною, щоб забезпечити коректний обмін даними. Завершальний етап включає тестування та виправлення помилок.

2.2 Аналіз та використання стеку технологій

Важлива частина проектування автоматизованої системи – аналіз та вирішення який стек технологій буде використовуватися в проекті для реалізації усіх функцій. Поділимо проект на дві частини: Backend та Frontend.

Для серверної частини (Backend) прийнято рішення використовувати середовище Node.js у зв’язку з веб-фреймворком Express. Як система управління базами даних буде використовуватися PostgreSQL. ORM для реляційних баз даних на Node.js – Sequelize. Саме вони забезпечать високу продуктивність, масштабованість та зручність у розробці.

Середовище виконання Node.js дозволяє використовувати JavaScript на серверній частини, що дуже спрощує розробку та підтримку, адже одна мова використовується як на фронтенді, так і на бекенді. Обирають Node.js завдяки його асинхронній природі, високій продуктивності та можливості обробки великої кількості одночасних запитів. У цьому проекті Node.js буде

використовуватися як основа для розробки серверної частини, яка забезпечить стабільність і масштабованість додатку.

Мінімалістичний веб-фреймворк для Node.js, який спростить створення веб-сервера і маршрутизацію запитів – Express. Він буде використовуватися для створення API, які обробляють запити від клієнтської частини, управління маршрутами (routes), та обробку проміжного програмного забезпечення (middleware).

PostgreSQL надає потужну реляційну базу даних. Вибрана саме вона завдяки її надійності, масштабованості та багатому набору функцій, таких як підтримка складних типів даних і розширюваність. Вона служить основним сховищем даних для додатка, зберігаючи інформацію про користувачів, обладнання, типи та бренди.

Sequelize спрощує взаємодію з PostgreSQL через ORM, дозволяючи працювати з базою даних за допомогою об’єктно-орієнтованого підходу. В цьому проекті він буде моделювати таблиці бази даних, створювати, оновлювати операції та забезпечувати цілісність даних.

Rest API дозволяє взаємодіяти між користувачем та сервером за допомогою методів таких як GET, POST, PUT та DELETE.

Для клієнтської частини (Frontend) прийнято рішення використовувати бібліотеку React.js, бібліотеку компонентів React Bootstrap, бібліотеку для виконання запитів Axios, бібліотеку управління маршрутизацією React Router DOM та бібліотеку управління станом у JS-додатках MobX. Використовуватимуться саме ці технології через їх ефективність і простоту. Разом вони допоможуть швидко розробити та легко підтримувати проект.

React.js – бібліотека для створення інтерфейсу користувача. Вона забезпечить створення динамічних і інтерактивних користувацьких інтерфейсів з швидким оновленням і відображенням даних.

React Bootstrap буде використано для візуалізації React, який реалізує Bootstrap – популярний CSS-фреймворк для створення адаптивних і стильних

інтерфейсів. Саме він дозволить швидко і легко створити гарні та функціональні інтерфейси користувача, дотримуючись тенденцій сучасного та адаптивного дизайну.

Axios підтримує проміси та асинхронні операції, що робить його ідеальним для взаємодії з API. Буде використаний для виконання запитів до бекенду, отримання та відправлення даних між клієнтською та серверною частинами.

React Router DOM допоможе влаштувати динамічний перехід між різними сторінками без перезавантаження. Це забезпечить реалізацію навігації між компонентами та плавний користувацький досвід.

MobX допоможе управляти станом додатку, зберігати глобальний стан та синхронізувати стану між різними компонентами. Це дозволить легко підтримувати і оновлювати стан додатку в реальному часі.

2.3 Розробка макета веб-додатка

Розробка макета є важливий етапом у створенні будь-якого веб-проекту, що дозволяє візуалізувати структуру, інтерфейс та взаємодію компонентів додатка. На основі проведеного аналізу предметної області можна обробити вимоги та потреби користувачів. Це необхідно для розуміння, які функції має виконувати додаток і як він буде взаємодіяти з користувачем.

Макет сайту складатиметься з декількох схем, оскільки в проекту буде не одна сторінка (Landing Page). Лендінг — це невеликий односторінковий сайт, який закликає клієнта зробити одну конкретну дію: наприклад, підписатися на розсилку, купити навчальний курс або замовити соковижимач [13]. Головна мета – продемонструвати основний функціонал кожної сторінки та їх взаємодію.

Для створення макету існує безліч різноманітних інструментів. Можна використовувати від графічних редакторів, таких як Adobe Photoshop, до ручного малювання. Однак вирішено використовувати додаток Figma, враховуючи його популярність та зручність використання. Відмінною особливістю Figma є те, що

працювати з пакетом можна в браузері [14]. Для детального дизайну необхідно також визначити кольори, шрифти, іконки та інші візуальні елементи, які будуть відповідати тематиці проекту та створювати приємний інтерфейс для користувачів.

Першим кроком неавторизований користувач буде попадати на основну сторінку додатка з обладнанням, в верхньому-правому куті він зможе натиснути кнопку “Авторизація” і в нього відкриється сторінка авторизації, де він зможе увійти, якщо зареєстрований, або зареєструватися. Макет входу та реєстрації представлено на рисунках 2.2, 2.3.

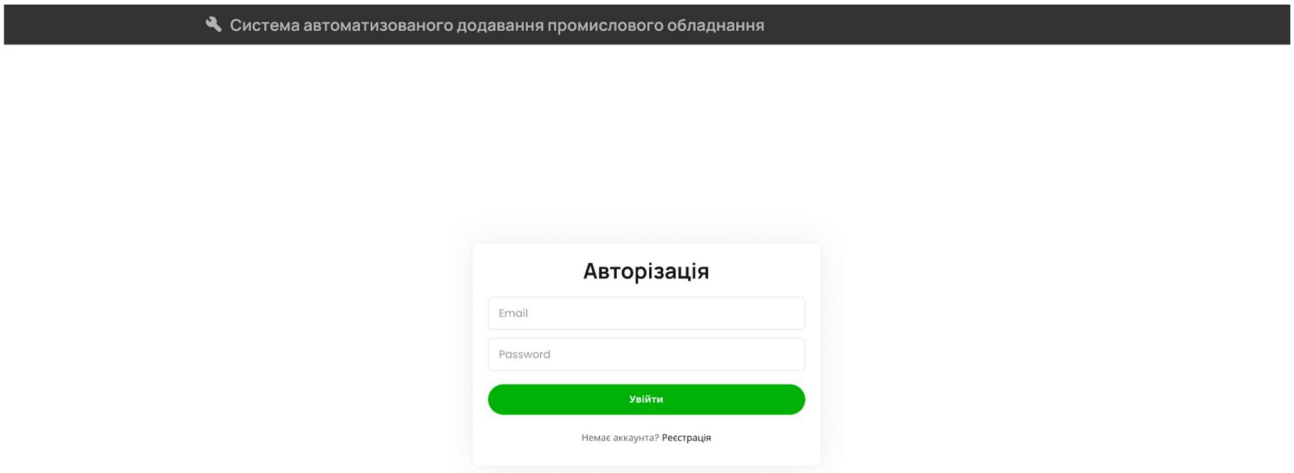

Рисунок 2.2 – Макет сторінки входу

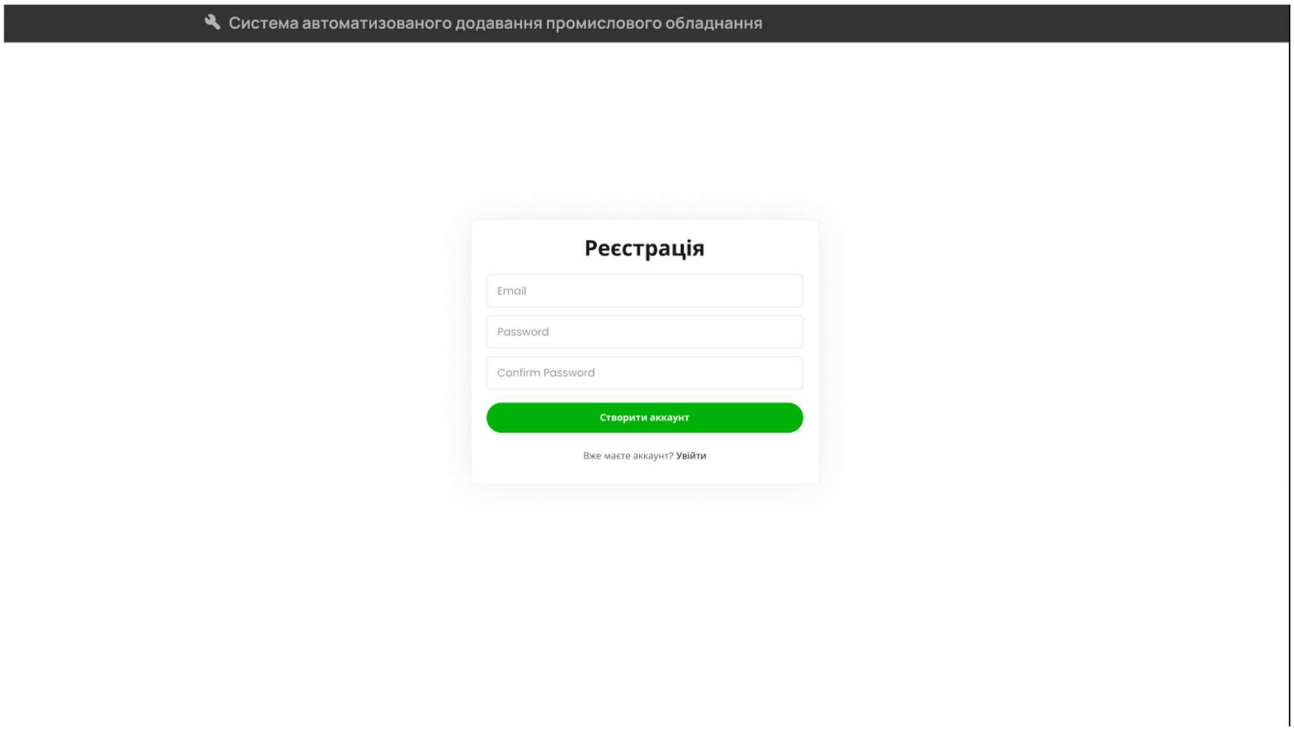

Рисунок 2.3 – Макет сторінки реєстрації

Головна сторінка складатиметься з відображення всього обладнання у формі карток, яке можна бути відобразити за типами та брендами. Список кнопок з типами буде розташований у лівий частині та займати 25 відсотків контейнеру. Він буде складатися динамічно на основі додавання адміністратором або власником сайту через адмін-панель. Кнопки з брендами будуть розташовані над обладнанням та також, як і список з типами будуть додані автоматично. Макет головної сторінки представлено на рисунку 2.4.

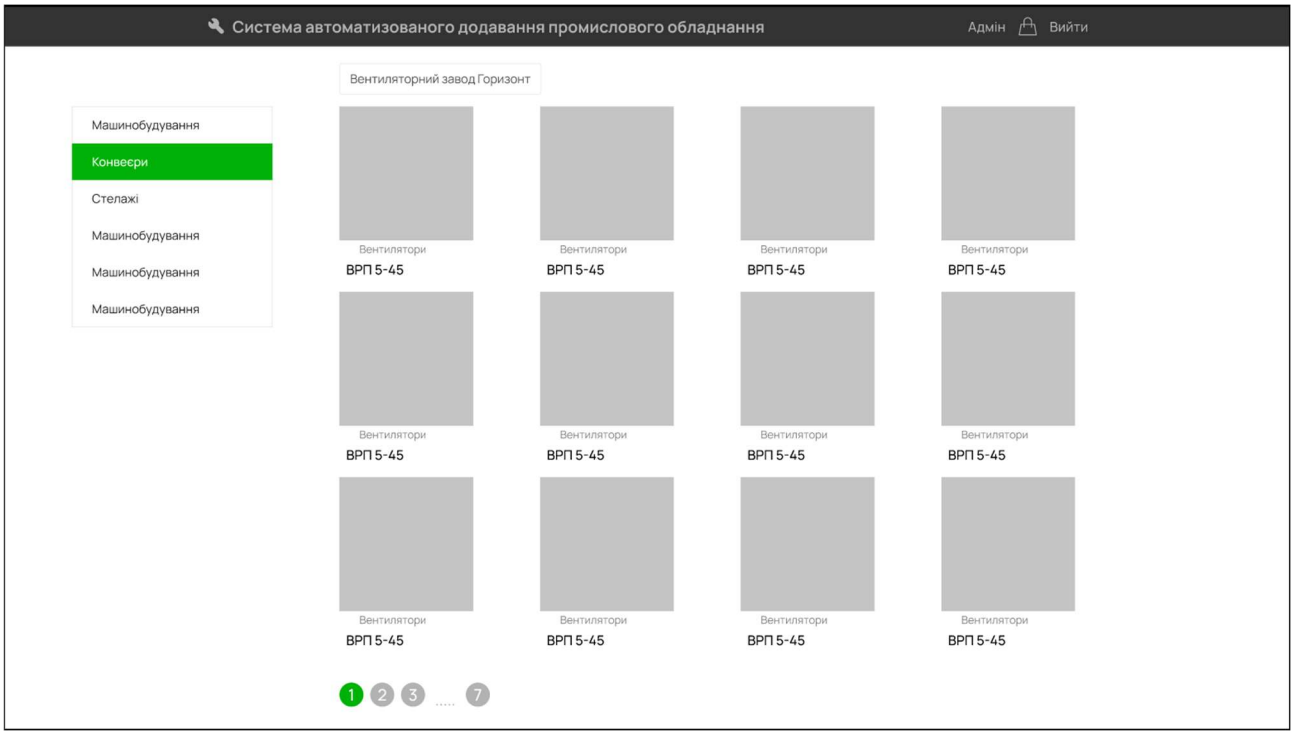

Рисунок 2.4 – Макет головної сторінки

Якщо користувача зацікавить певне обладнання, він зможе перейти до детального огляду обладнання, натиснувши на відповідну картку обладнання. На цій сторінці буде велике зображення обладнання, його назва, ціна та властивості. Над зображенням буде повна назва обладнання великим шрифтом. Під зображенням розташована ціна обладнання. У правій частині сторінки буде розміщено характеристику обладнання, яка займатиме половину контейнера. Характеристика складатиметься з назви властивості на її опис. Макет сторінки обладнання представлено на рисунку 2.5.

Рисунок 2.5 – Макет сторінки обладнання

Для додавання певного обладнання, типу або бренда буде реалізована панель адміністратора, яка знаходиться у правому верхньому куті біля кнопки “Вийти”, яка відображається тільки для авторизованих користувачів. Якщо у авторизованого користувача є роль адміністратора, він має можливість увійти в адмін-панель, на якій знаходиться по центру екрана три кнопки з додавання властивостей. Макет сторінки панелі адміністратора представлено на рисунку 2.6.

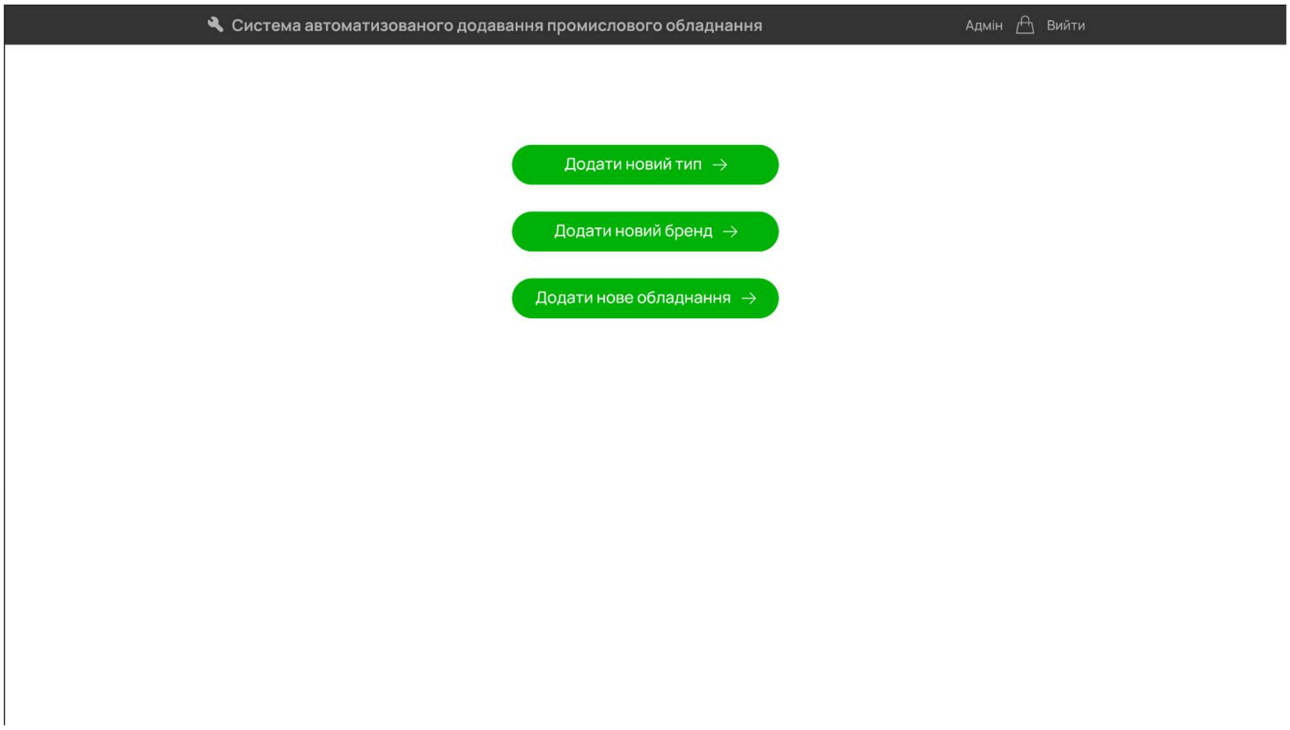

Рисунок 2.6 – Макет сторінки панелі адміністратора

Натиснувши на якусь з кнопок, буде відображатися спливаюча панель зі зручним доданням нових типів, брендів або обладнання, та його властивостей.

2.4 Розробка схеми діаграми серверної частини

Розробка діаграми серверної частини є невід'ємним етапом створення програмного забезпечення. Ця діаграма дозоляє на етапі розробки візуалізувати структуру системи та взаємозв’язки між її компонентами.

База даних призначена для зберігання інформації про користувачів, обладнання, типи та бренди обладнання, а також пов'язаної інформації. У схемі діаграми повинні бути таке призначення:

– користувачі можуть входити до системи та керувати обладнанням;
– обладнання класифікується за типами та брендами;
– додаткова інформація про обладнання зберігається в окремій таблиці.

Структура бази даних має підтримувати функції веб-застосунків, такі як керування користувачами, каталог обладнання, фільтрація обладнання за типом і брендом, а також доступ до додаткової інформації про обладнання.

Спочатку йде юзер з унікальним ідентифікатором, його імейлом, паролем та роль користувача в системі, наприклад, адміністратор, звичайний користувач. В нього має бути зв’язок один до багатьох з обладнанням. Це означає, що кожен запис у таблиці Юзер може бути пов’язаний з кількома записами в таблиці Обладнання. На діаграмі зв’язки визначаються лініями, що з’єднують таблиці. В наведеному прикладі йде лінія з одиничною зв'язкою на стороні “Юзер” і множинною зв'язкою на стороні “Обладнання”.

Кожне обладнання належить до одного типу і одного бренду, але тип та бренд можуть бути пов’язані з кількома записами про обладнання. Це наведено лінією “Багато до одного”. У кожного обладнання має бути інформація про нього, це наведено іншою таблицею з зв’язком “Один до багатьох”. Кожен запис про обладнання може мати декілька записів з інформацією про це обладнання. Обладнання може мати кілька пов’язаних записів у таблиці “Обладнання_інфо”.

Схему діаграми серверної частини наведено на рис. 2.7.

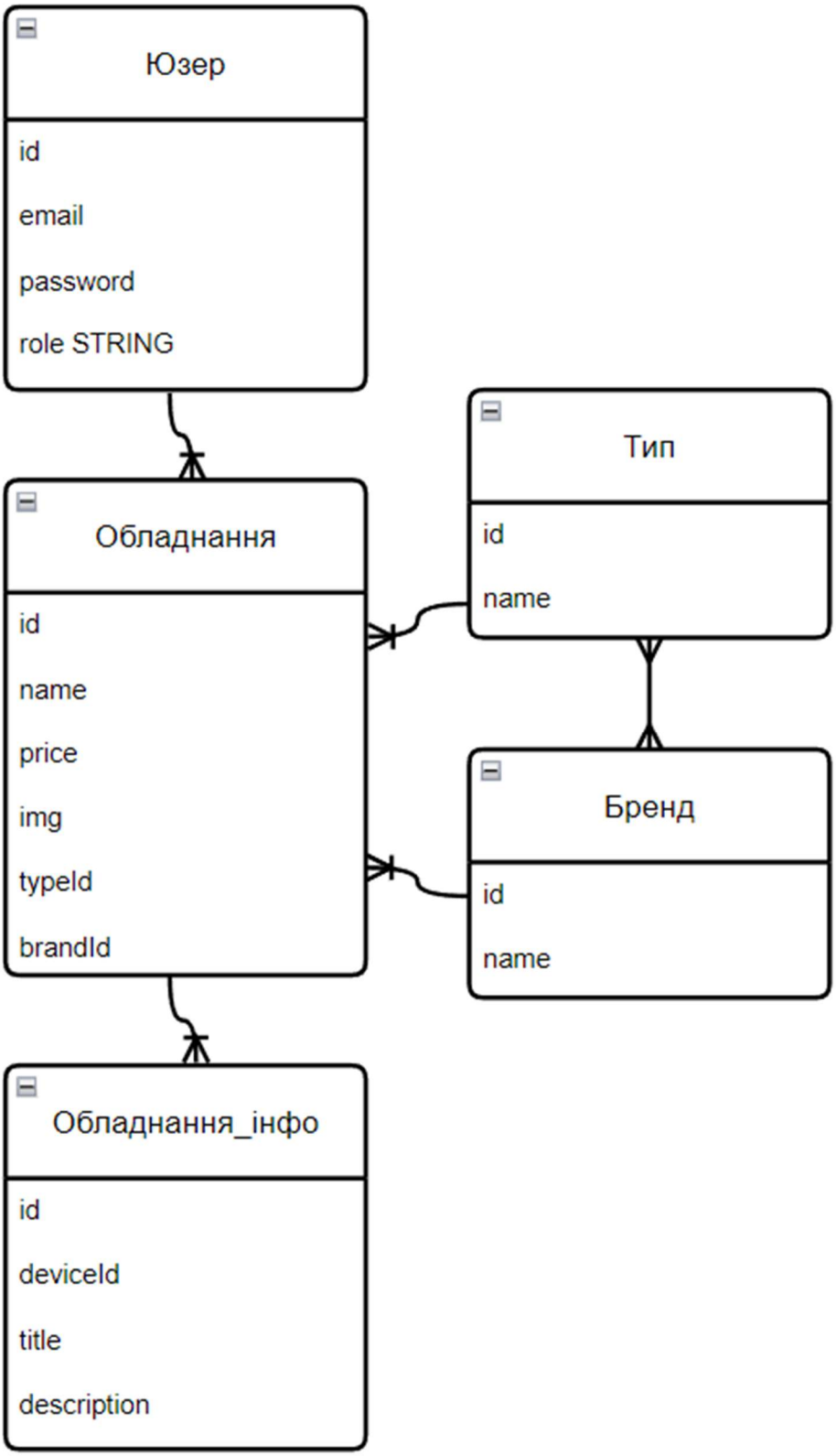

Рисунок 2.7 – Схема діаграми серверної частини

2.5 Функціональні можливості системи

Система надаватиме користувачам зручний інтерфейс для введення та редагування технічних характеристик промислового обладнання. Користувач зможе додавати нові записи, оновлювати існуючи дані та видаляти застарілу інформацію без допомоги розробника або людини, яка повинна розбиратися в управлінні бази даних. Саме це забезпечить актуальність та точність усіх даних про обладнання.

При додаванні нового обладнання система автоматично генеруватиме унікальний ідентифікатор. Це дозволить виключити ймовірність дублювання записів, спростить та прискорить керування даними, забезпечуючи унікальність кожної одиниці обладнання у базі даних.

Система включатиме інструменти фільтрації, які дозволять користувачам швидко знаходити потрібне обладнання за різними параметрами. Користувачі зможуть задавати фільтри для сортування обладнання за певними критеріями, такими як тип, виробник та інші технічні характеристики.

Система підтримуватиме управління користувачами та ролями, дозволяючи адміністраторам створювати нові облікові записи, призначати ролі та права доступу. Це забезпечить гнучкість в управлінні доступом до системи та її функціональним можливостям, гарантуючи, що користувачі зможуть виконувати лише ті дії, до яких вони мають дозвіл.

Для забезпечення безпеки даних система використовуватиме автентифікацію та авторизацію. Користувачі будуть проходити процес автентифікації для доступу до системи, а їхні права та доступ до різних функцій системи будуть визначатися через авторизацію. Це запобігатиме несанкціонованому доступу та захистить дані від загроз.

Ці функціональні можливості забезпечуватимуть рішення для автоматизації процесу додавання та управління даними про промислове обладнання. Система надаватиме зручні інструменти для введення та

редагування даних, ефективні методи фільтрації, надійні механізми безпеки та управління користувачами, що зробить її потужним інструментом для керування технічними характеристиками обладнання.

2.6 Забезпечення безпеки

Для забезпечення безпеки користувачі проходитимуть процес автентифікації під час входу до системи, використовуючи email та пароль. Паролі будуть шифруватися за допомогою бібліотеки BCrypt, що запобігає їх зберіганню у відкритому вигляді. Авторизація здійснюється за допомогою JWT (JSON Web Tokens), які генеруються при вході користувача в систему та перевіряються при кожному запиті до захищених ресурсів. Цей формат представляє собою текстовий формат обміну даними між комп'ютерами [15].

BCrypt – сучасна і популярна бібліотека для хешування паролів, розроблена для безпечного зберігання паролів. Згідно з джерелом npmjs [16], тижнева кількість завантаження пакету хешування становить приблизно 1,7 мільйони завантажень. Існує безліч функцій хешування паролів, однак BCrypt це баланс швидкості та безпеки. Такі функції хешування як MD5, SHA1..3 – загального призначення та призначені для обчислення величезних обсягів за якомога коротший час. Сучасний спеціалізований сервер може обчислювати хеш MD5 приблизно 330 мегабіт щосекунди [17].

Проведемо розрахунки.

В англійському алфавіті існує 26 літер та 10 цифр у сумі 36 символів. Таким чином, для кожного символу пароля ми маємо 36 можливих варіантів. Для пароля довжиною 6 символів у нас буде 36 у шостій ступені:

$$N = 36 \wedge 6 = 2\ 176\ 782\ 336$$

Швидкість обчислення хеша MD5: 330 Мегабіт/с. Перекладемо швидкість обчислення в мегабайти за секунду (1 байт = 8 біт):

$$S = 330 \text{ Мбіт/с} * (1 \text{ байт} / 8 \text{ біт}) = 41.25 \text{ МБ/с}$$
$$S = 41.25 \text{ МБ/с} * 10^6 = 41\,250\,000 \text{ байт/с}$$

Наступним кроком обчислимо за скільки ми можемо перебрати всі можливі паролі за формулою:

$$T = N / S,$$

де T – час, необхідний для перебору всіх можливих паролів, с;

N – загальна кількість можливих паролів;

S – швидкість обчислення, байт/с.

$$T = 2\,176\,782\,336 / 41\,250\,000 = 52.77 \text{ с}$$

Робимо висновок, що можна спробувати кожний можливий пароль такого розміру за приблизно 53 секунди.

BCrypt використовує варіант розкладу введення ключів алгоритму шифрування Blowfish і вводить коефіцієнт роботи, який дозволяє визначити, наскільки великою буде хеш-функція. Завдяки цьому, BCrypt може автоматично збільшувати стійкість з часом. Це досягається шляхом використання адаптивного алгоритму, який може змінювати кількість ітерацій, залежно від того, як швидко машина, яка виконує хешування, стає швидше.

JWT токен – це стандарт відкритого формату передачі даних як JSON-об’єкта. Переваги JWT в тому, що він має менші вимоги до участі людини в керуванні його станом і добре масштабується [18]. JWT буде використано для створення токенів аутентифікації за умови успішної аутентифікації користувача.

Ці токени потім використовуються для надання доступу до захищених ресурсів, таких як сторінки профілю або адмін-панелі, без необхідності повторної автентифікації кожного запиту.

Також JWT може використовуватися для реалізації механізму сесій або зберігання стану авторизації на стороні клієнта, що може зробити веб-додаток більш масштабованим та менш залежним від сервера.

3 РОЗРОБКА СИСТЕМИ АВТОМАТИЗОВАНОГО ДОДАВАННЯ ПРОМИСЛОВОГО ОБЛАДНАННЯ

3.1 Вибір середи розробки

При розробці програмного забезпечення вибір середи розробки є однією з важливих частин для ефективності та зручності роботи розробника. Visual Studio Code (VS Code) [19] є одним з найпопулярніших редакторів коду від Microsoft. Він пропонує безліч можливостей для оптимізації робочого процесу. VS Code пропонує ряд функцій, які роблять його дуже зручним у використанні.

Потрібно почати з того, що VS Code за замовчуванням це текстовий редактор вихідного коду (рис. 3.1).

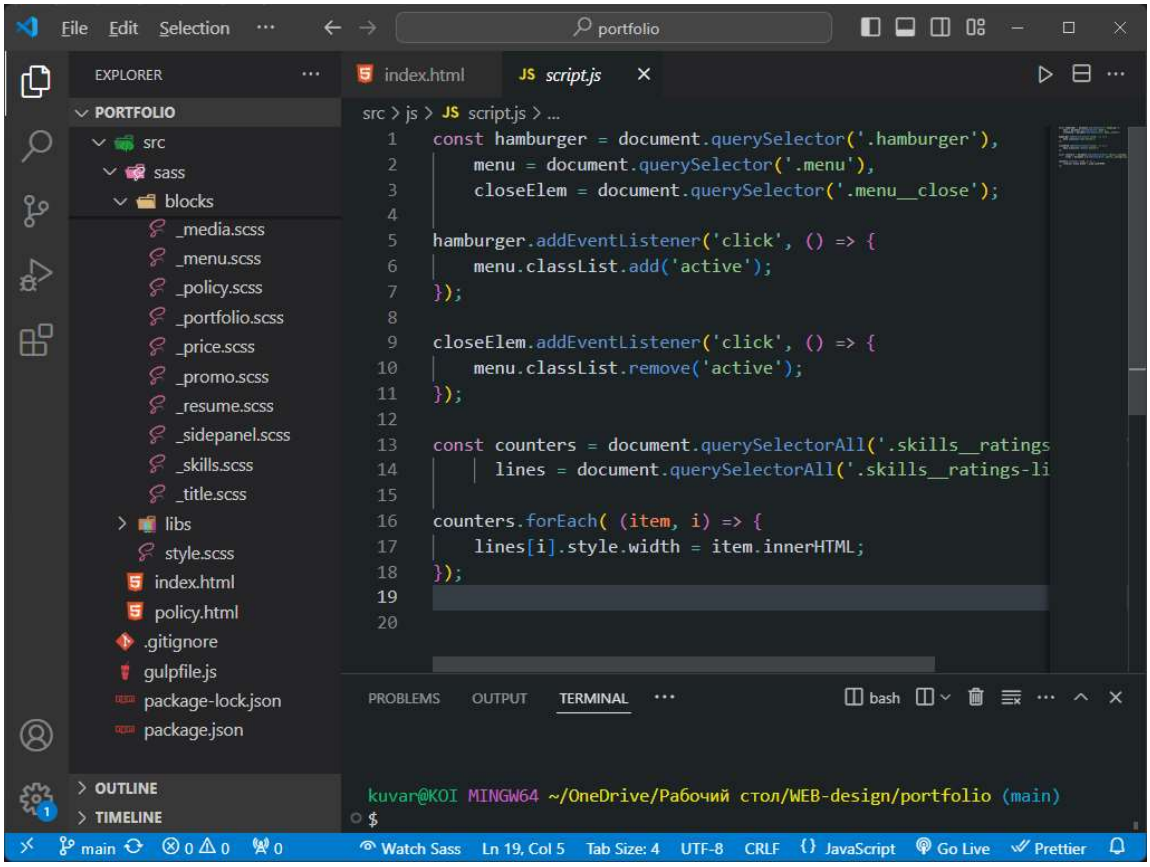

Рисунок 3.1 – Інтерфейс Visual Studio Code

Виходячи з того, що це текстовий редактор коду, ми можемо зробити висновок, що він має дуже швидко запускатися і працювати. Це робить його особливо зручним для швидкої роботи з кодом. Він підтримує величезну кількість мов і має можливість встановлення величезної кількості розширень, плагінів, які додають підтримку інших мов програмування, інструментів та інтеграцій, наприклад налагодження, тестування та робота з базами даних.

В багатьох серед розробки існує підтримка налагодження, де можна встановлювати точки зупинки, витворювати код покроково та слідкувати за станом змінних, що дуже спрощує процес пошуку та виправлення помилок. VS Code також підтримує налагодження для багатьох мов програмування.

У VS Code дуже зручне управління системою контролю версій. Це вбудована підтримка Git, яка дозволяє розробнику або розробникам, якщо над одним проектом працює група осіб, легко керувати версіями коду. Можна

виконувати комміти (commit), створювати гілки (branch), вирішувати конфлікти між версіями та виконувати інші операції прямо з VS Code.

Ні для кого не секрет, що розробники повинні вміти бістро друкувати код. Розробники теж звичайні люди і при друкуванні вручну - можуть припускатися помилок з роду, що розробник пропустив одну літеру в назві функції або не поставив велику літеру. Для JavaScript, нагадаю, що близько 90% коду буде написано саме на його основі, дуже важливо дотримуватись назви функцій і великі букви, оскільки функції, змінні, методи в ньому пишуться за принципом CamelCase (рис. 3.2).

```
const myVariableName; // коректно (використовується CamelCase)
myFunctionName(); // коректно (використовується CamelCase)

const my variable name; // некоректно (пробіли заборонені)
my function name(); // некоректно (пробіли заборонені)

const MyVariableName; // працює, але не рекомендується (варто починати з літери в нижньому регістрі)
MyFunctionName(); // працює, але не рекомендується
```

Рисунок 3.2 – Принцип CamelCase

Принцип CamelCase – коли кілька слів пишуться разом, без пробілів, і кожне нове слово пишеться з великої літери. CamelCase (у перекладі як «Верблюжий Стиль») отримав свою назву через використання великих букв, які нагадують верблюжі горби [20]. Саме завдяки такому принципу в коді не буде помилок і інші розробники, які будуть працювати з іншим кодом, можуть легко аналізувати та розуміти, за що саме відповідає та або інша функція або змінна.

Одна з переваг VS Code – функція IntelliSense надає інтелектуальне автодоповнення коду, тобто його функцій, підказки та інформацію про функції, методи та змінні, що значно підвищує продуктивність і точність роботи розробника. IntelliSense аналізує код та пропонує автодоповнення для назв змінних, функцій, методів та класів, що скорочує час написання коду та зменшує кількість помилок (рис. 3.3).

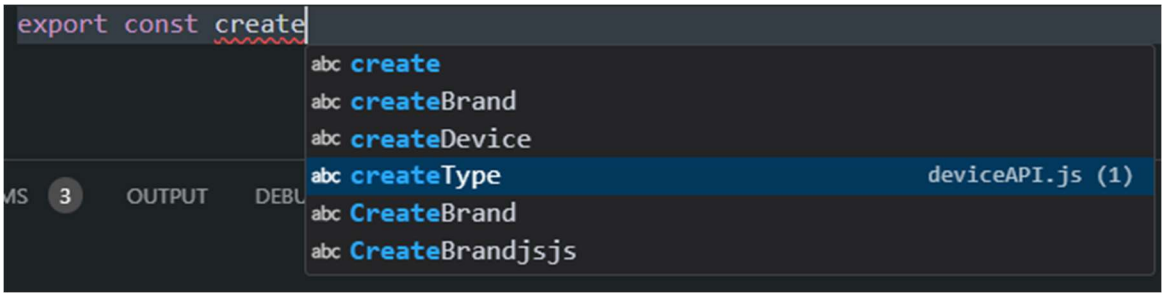

Рисунок 3.3 – Підказка IntelliSense для назв змінної

Функція відображає синтаксичні підказки, показуючи розробнику аргументи, які можуть бути передані в функції або методи, а також їх порядок, що особливо корисно при роботі з новими або складними API (рис. 3.4).

Рисунок 3.4 – Підказка IntelliSense структури функції

Під час написання функції IntelliSense відобразить інформацію про параметри, дозволяючи швидко дізнатися, які типи даних очікуються та як використовувати функцію чи метод. IntelliSense може відображати документацію за функцією, методом або класом прямо в редакторі, надаючи всю необхідну інформацію для їх правильного використання. Крім того, він полегшує навігацію за кодом, дозволяючи швидко переходити до визначення функції або методу, що спрощує роботу з громіздким кодом. Також він допомагає виявляти синтаксичні та логічні помилки у коді до його компіляції або виконання, що дозволяє виправляти помилки на ранніх етапах розробки (рис. 3.5).

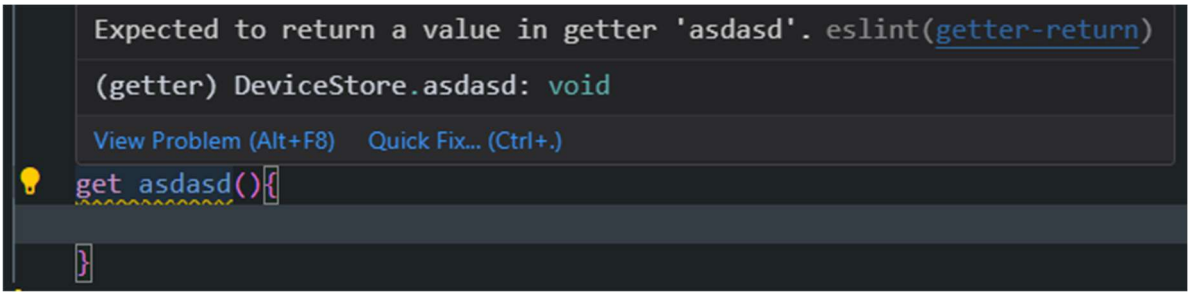

Рисунок 3.5 – Попередження IntelliSense про помилку

IntelliSense працює за замовчуванням для більшості популярних мов програмування, але його функціональність можна розширити і налаштувати за допомогою різних розширень, доступних у Visual Studio Code Marketplace, така як Eslint для JavaScript, включаючи підтримку для Python, JavaScript, TypeScript, C#, Java та багатьох інших мов. У результаті, функція IntelliSense у VS Code є потужним інструментом, який допомагає розробнику писати код швидше, точніше і з меншою кількістю помилок, спрощуючи процес розробки та покращуючи якість написаного коду.

Як і в інших середах розробку у VS Code є вбудований термінал, який розташовується в нижній частині програми (рис. 3.6).

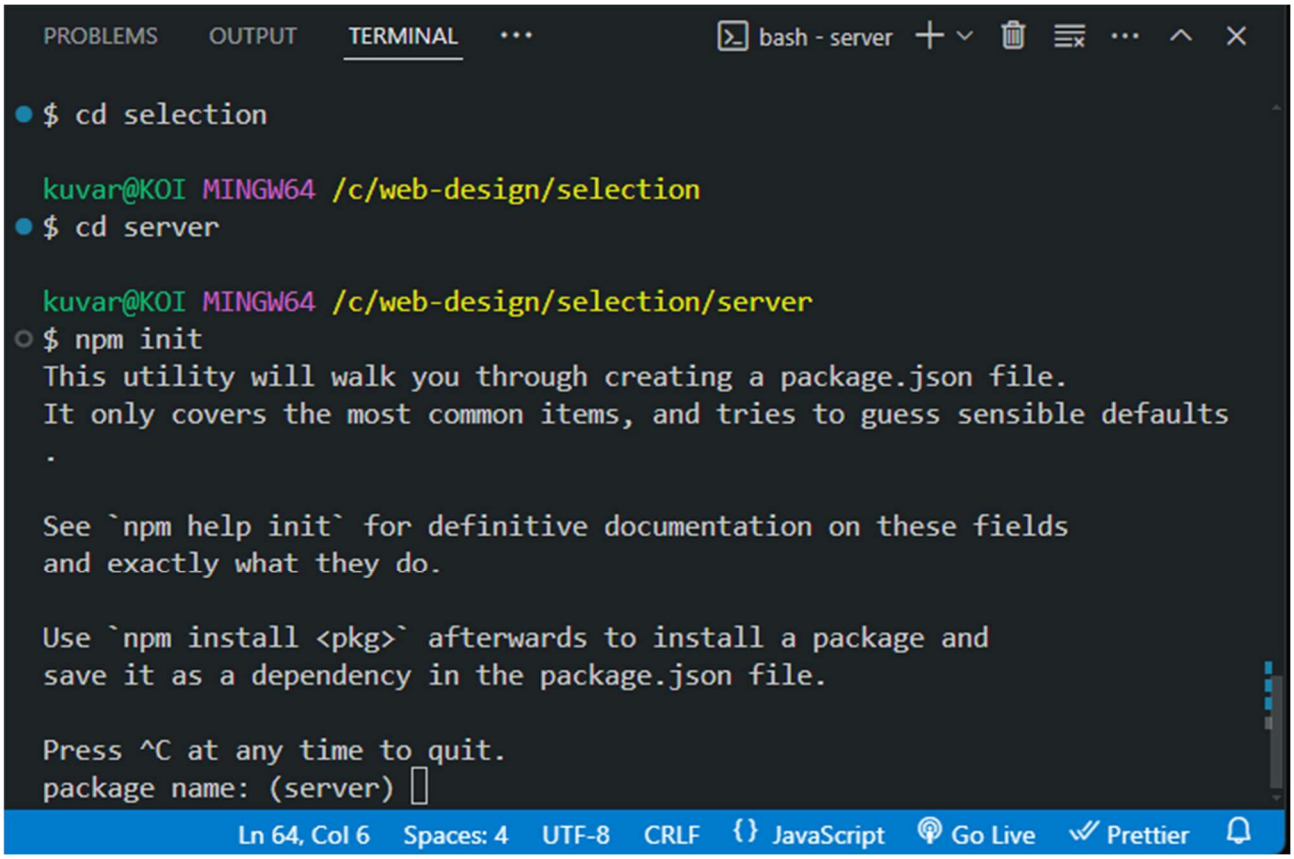

Рисунок 3.6 – Термінал Visual Studio Code

Він дозволяє виконувати необхідні для розробки команди, наприклад переміщення по папках командами "cd..", завантаження пакетів "npm install назва пакету", створення коммітів "git commit..", який зробить нову версію програмного коду і так далі.

3.2 Налаштування середовища розробки

Першим кроком в процесі розробки системи необхідно настроїти середу розробки VS Code. Настроюють поведінку редактору на зручну взаємодію з розробником, встановлюються необхідні розширення. Настройка може починатися з тем, які дозволяють налаштувати зовнішній вигляд редактора, такі як колір тексту, фону, панелей, кнопок та багато іншого, закінчуючи

налаштуванням автоматичного зберігання файлу, наприклад коли натискають будь-яке інше вікно або використовуючи комбінацію клавіш “Alt + Tab”.

З розширень для VS Code будемо використовувати Eslint, All Autocomplete, Auto Close Tag, Auto Complete Tag, Auto Rename Tag, Backticks, Import Cost, Intellicode, Path Autocomplete, Prettier, JavaScript (ES6) code snippets та Reactjs code snippets:

– ESLint статично аналізує ваш код, щоб швидко знаходити проблеми. Він вбудований у більшість текстових редакторів, і ви можете запускати ESLint як частину конвеєра безперервної інтеграції [21]. Він допоможе підтримувати високий рівень якості коду та дотримуватися стандартів стилю кодування;

– All Autocomplete розширює автозаповнення, охоплюючи всі відкриті файли у редакторі, допоможе прискорити написання коду, пропонуючи автозаповнення зі всіх відкритих файлів;

– Auto Complete Tag пропонує автозаповнення для HTML та XML тегів, прискорить процес написання коду, пропонуючи завершення тегів. Наприклад, якщо в HTML відкрити блок “ ”, після натискання “>” Auto Complete Tag автоматично додасть “ ” і залишить курсор усередині блоку. Auto Rename Tag доповнює попереднє розширення шляхом перейменування відповідного закриваючого тега, при зміні його;

– У JavaScript не можна повторювати одинарні або подвійні лапки всередині однієї строчки, оскільки це порушує синтаксичні правила мови та призведе до помилки. Backticks допоможе уникнути припущення помилок автоматично замінюючи одинарні або подвійні лапки на зворотні лапки в JavaScript, коли це необхідно;

– Import Cost відображає розмір бібліотек, що імпортуються, прямо в редакторі, що корисно для оптимізації коду;

– Intellicode розширює можливості IntelliSense покращуючи автозаповнення та роблячи його більш точним;

– Path Autocomplete пропонує автозаповнення для шляхів до файлів та папок, прискорить процес написання шляхів до файлів, знижуючи ймовірність помилок;

– Prettier – інструмент для форматування коду, який допоможе підтримувати єдиний стиль кодування, що покращує читання та підтримку коду;

– JavaScript (ES6) code snippets: надає сніпети коду для JavaScript, прискорюючи процес написання коду і підвищуючи продуктивність [22]. Замість того, щоб вручну писати великий метод наприклад:

```
array.forEach(currentItem => {
});
```

ми можемо написати “fre” та натиснути клавішу “Tab”, JavaScript code snippets автоматично розкриє метод (рис. 3.7);

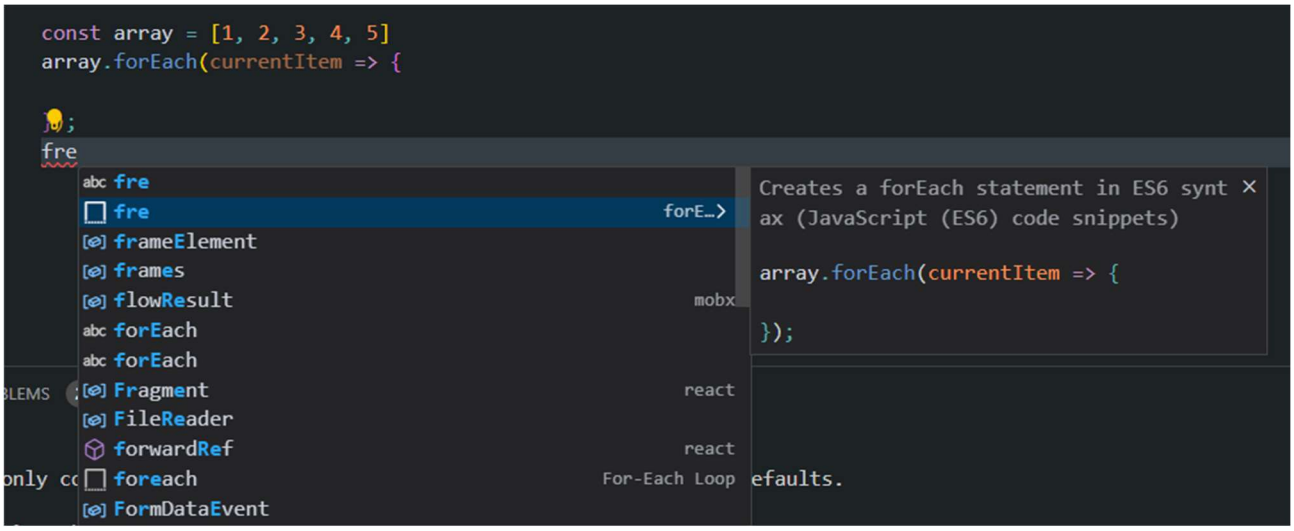

Рисунок 3.7 – Представлення роботи JavaScipt (ES6) code snippets

– Reactjs code snippets – набір сніпетів для розробки на React, що доповнює JavaScript code snippet, корисний для швидкого написання стандартних фрагментів коду в React.

Саме ці розширення забезпечать високу якість коду, зручність розробки та прискорять процес написання коду. Деякі розширення, такі як All Autocomplete,

Import Cost та Backticks, можуть бути корисними, але не є обов'язковими і кожний розробник може встановлювати їх на свою думку.

3.3 Створення серверної структури проекту та встановлення необхідних пакетів

Наступним етапом необхідно сформувати структуру проекту. Відповідно до аналізу предметної області, розділимо проект на дві частини шляхом створення двох папок - client та server. Подібні проекти завжди починаються з розробки backend частини. Тому в папці server створюємо кореневий файл index.js, з якого починається запуск. Далі необхідно ініціалізувати “npm init” для завантаження всіх пакетів. На цьому кроці VS Code створить файл package.json, файл конфігурації в проектах на Node.js, який містить інформацію про проект - назву, версія, автори, ліцензія, залежність, скрипт для автоматизації завдань та інші дані, необхідні для управління та розробки проекту. Для цього вводимо усі данні в термінал (рис. 3.8).

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

$ npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help init` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (selection) server
version: (1.0.0)
description: nure
entry point: (index.js)
test command:
git repository:
keywords:
author: Kuvarzin Oleksii
license: (ISC)
About to write to C:\Users\kuvar\OneDrive\Рабочий стол\WEB-design\selection\package.json:

{
  "name": "server",
  "version": "1.0.0",
  "description": "nure",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
```

Рисунок 3.8 – Ініціалізація нового проекту Node.js

Залежності проекту встановлюються командою “npm install”. Встановлюємо фреймворк Express, як система керування базою даних PostgreSQL, об'єктно-реляційне відображення Sequelize, для ображення браузера до сервера пакет cors, а також необхідний спосіб зберігання даних конфігурації, керування змінними оточення бібліотекою dotenv. Для цього пишемо команду “npm install express pg pg-hstore sequelize cors dotenv” та відразу після завантаження всі методи можна побачити у полі dependencies в файлі package.json. Для тестування на ранньому етапі встановимо інструмент nodemon, це допоможе при кожній зміні в коді не перезапускати сервер вручну, а робити це автоматично за нас. Для його роботи в полі scripts змінюємо команду “test” на "dev": "nodemon index.js".

Наступним кроком створимо структуру програми. За допомогою “require” можна імпортувати якісь модулі у файл. Імпортуємо в файл index.js фреймворк Express та створюємо об`єкт:

```
const express = require('express')
const app = express()
```

Далі необхідно вказати порт на якому наша програма буде працювати. Порт для сервера вибрано 5000, для реалізації цього можна додати в кореневий файл “const PORT = 5000”, але порт оголошувати статично є поганою практикою, для всієї конфігурації виносимо в змінне оточення, тому створено файл .env і вказано PORT = 5000. Тепер же для отримання цього порту використовуємо змінні оточення. Замість “const PORT=5000” у файлі index.js пишемо:

```
const PORT = process.env.PORT || 5000
```

Якщо змінна PORT не задана, сервер статично братиме 5000 порт. Також щоб сервер мав змогу зчитувати файл .env використовуємо dotenv, шляхом додавання до index.js “require('dotenv').config()”.

3.4 Розробка бази даних

Наступним етапом підключимо PostgreSQL до нашого проекту. Для цього завантажуємо програму pgAdmin 4 з офіційного сайту PostgreSQL. Створюємо нову базу даних з назвою “selection” (рис. 3.9).

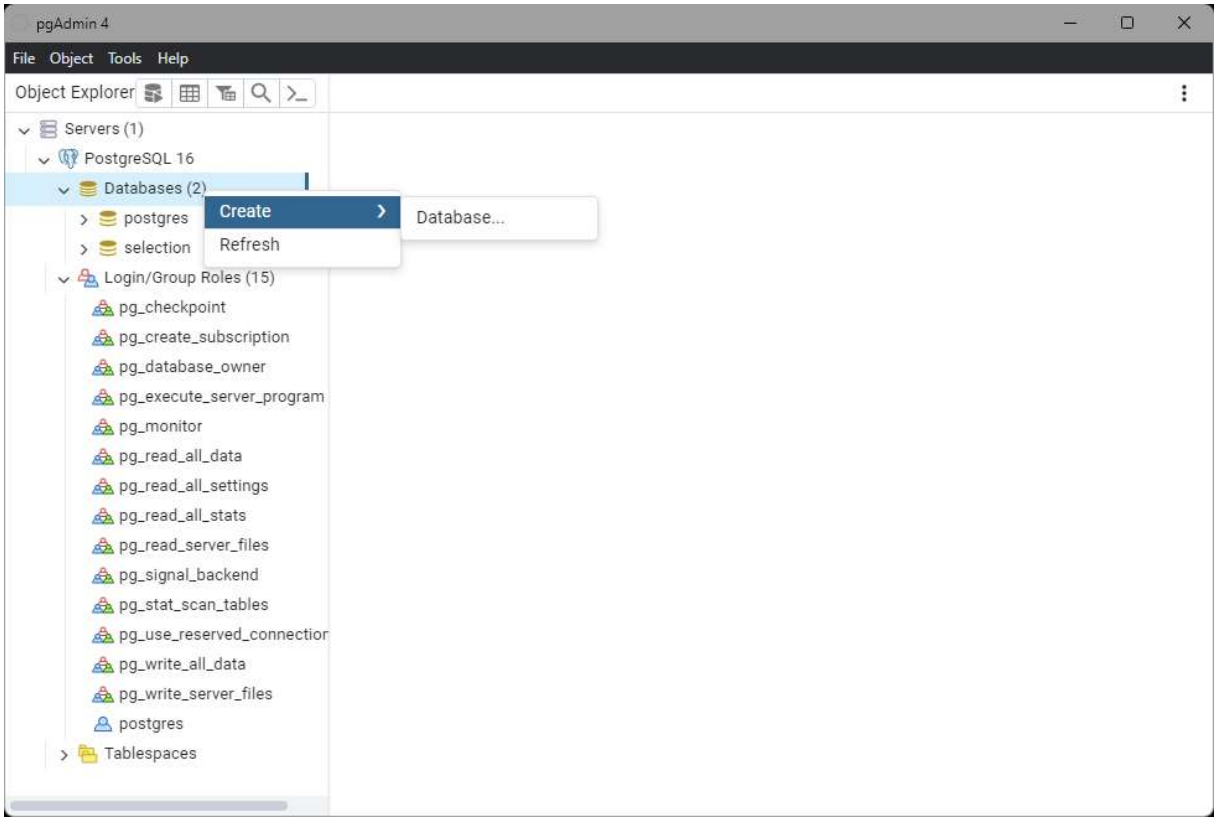

Рисунок 3.9 – Створення бази даних “selection”

Щоб підключити та надати доступ нової бази даних до проекту у файлі .env створюємо необхідні змінні, які було вказано в pgAdmin 4:

```
DB_NAME=selection
DB_USER=postgres
DB_PASSWORD=123456
DB_HOST=localhost
DB_PORT=5432
```

Відразу після цього налаштовуємо з'єднання з базою даних за допомогою бібліотеки Sequelize. Для цього створено файл db.js:

```
const {Sequelize} = require('sequelize')
module.exports = new Sequelize(
```

```
    process.env.DB_NAME,
    process.env.DB_USER,
    process.env.DB_PASSWORD,
    {
        dialect: 'postgres',
        host: process.env.DB_HOST,
        port: process.env.DB_PORT
    })
```

Для запуску сервера та синхронізації моделі даних з базою даних імпортовано об'єкт Sequelize, який зберігає конфігурацію для підключення до бази даних, після чого йде перевірка стабільного підключення. Якщо всі дані конфігурації на попередньому етапі зроблені правильно - сервер запуститься локально за допомогою функції start() і повідомить нам про це через консоль, в іншому випадку - виведе помилку:

```
const sequelize = require('./db')
const start = async () => {
try {
    await sequelize.authenticate()
    await sequelize.sync()
    app.listen(PORT, () => console.log(`Server started on port ${PORT}`))
} catch (e) {
    console.log(e)}
}
start()
```

Наступним етапом створення бази даних потрібно сформувати моделі даних згідно з діаграмою бази даних, яка була створена на етапі Проектування

автоматизованої системи (рис. 2.7). Діаграма була створена як візуальна схема того, як дані розташовуються в базі даних. Тепер потрібно перенести цю схему в код, щоб у додатку був опис які дані необхідно використовувати і як із ними взаємодіяти. Для цього створено файл models.js, який формує ці моделі та їх залежності, як зазначено на діаграмі, але на мові програмування:

```
const sequelize = require('../db')
const {DataTypes} = require('sequelize')
// формування моделей
const Device = sequelize.define('device', {
        id: {type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true},
        name: {type: DataTypes.STRING, unique: true, allowNull: false},
        price: {type: DataTypes.INTEGER, allowNull: false},
        img: {type: DataTypes.STRING, allowNull: false},
})
// формування зв'язку між моделями
Type.hasMany(Device)
Device.belongsTo(Type)
Brand.hasMany(Device)
Device.belongsTo(Brand)
Device.hasMany(DeviceInfo, {as: 'info'});
DeviceInfo.belongsTo(Device)
Type.belongsToMany(Brand, {through: TypeBrand})
Brand.belongsToMany(Type, {through: TypeBrand})
// експорт моделей для використ. в інших файлах
module.exports = {
        User, Device, Type, Brand, TypeBrand, DeviceInfo
}
```

Усі інші моделі, такі як User, Type, Brand, DeviceInfo, TypeBrand, сформовано за аналогією моделі Device.

Для того, щоб нові моделі з'явилися в базі даних, імпортуємо наш файл models.js в кореневий файл index.js:

```
const models = require('./models/models')
```

Якщо в цей момент у нас запущено сервер, у терміналі можемо побачити логи запитів до бази даних, щоб створити ті чи інші моделі (рис. 3.10).

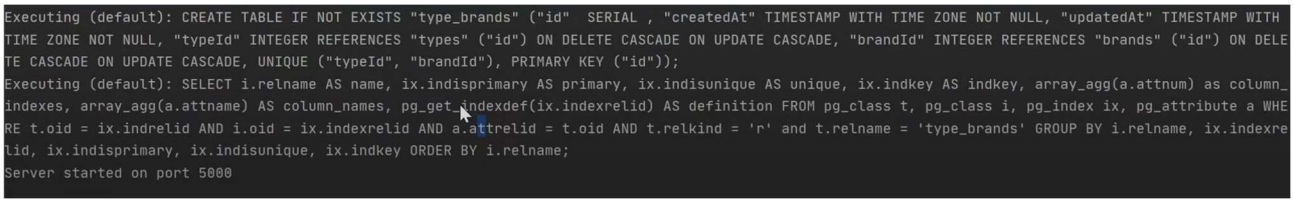

Рисунок 3.10 – Логи запитів до бази даних в Terminal

Відкриємо pgAdmin 4 для перевірки чи сформував Sequelize нові моделі в базі даних (рис. 3.11).

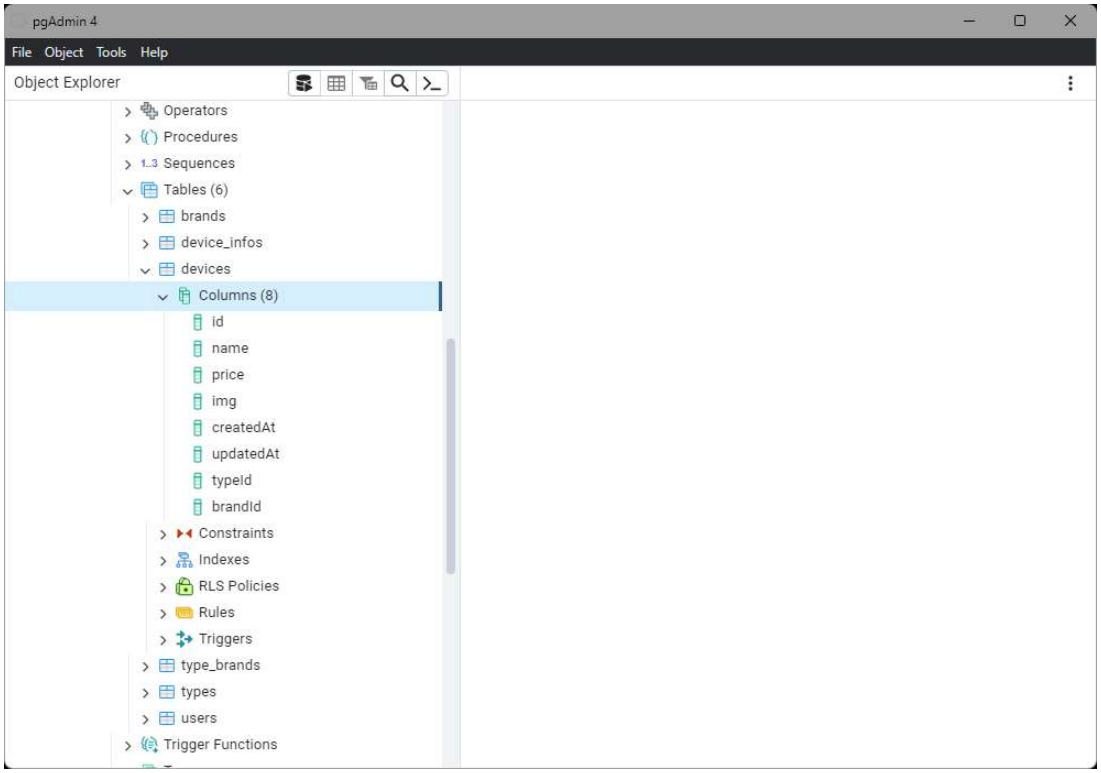

Рисунок 3.11 – Програма pgAdmin 4 зі створеними моделями

3.5 Конфігурація маршрутизації та контролерів

Наступним етапом для обробки запитів до сервера необхідно налаштувати поведінку маршрутизації. Для цього налаштовують маршрути, для визначення адреси обробки запиту, створюють контролери, які визначатимуть логіку обробки типів запитів. За необхідністю також додають проміжне програмне забезпечення (middleware) для обробки помилок, автентифікації та інші.

Для зручності розробки створено папки routes, controllers та middleware. В папці routes будуть додаткові файли, такі як brandRouter, deviceRouter, typeRouter, userRouter та основний зв'язуючий файл index.js для маршрутизації. Для кожного додаткового файлу маршрутизації необхідно задати логіку, за це відповідатимуть файли контролерів. Для реалізації додаткових функцій як авторизація користувача, перевірка доступу до додавання обладнання та виводу помилок будуть використані дані в middleware.

Починаючи з маршрутизації, розглянемо поступово на прикладі додаткового файлу deviceRouter.js, який визначить маршрут для обладнання. У Express існує об'єкт “Router”, який надасть зручний спосіб визначення маршрутів. За логіку обробки запитів буде відповідати файл deviceController.js. Для функції створення нового обладнання реалізуємо систему, яка перевірятиме користувача, чи є у нього доступ до цієї функції за допомогою файлу checkRoleMiddleware.js. Імпортуємо це в файл deviceRouter.js та додаємо методи:

```
const Router = require('express')
const router = new Router()
const deviceController = require('../controllers/deviceController')
const checkRole = require('../middleware/checkRoleMiddleware')
router.post('/', checkRole('ADMIN'), deviceController.create)
router.get('/', deviceController.getAll)
router.get('/:id', deviceController.getOne)
module.exports = router
```

Файл контролера deviceController.js містить три методи для обробки різних запитів пов'язаних з обладнанням.

Метод “create” оброблятиме POST-запит для створення нового обладнання. З запиту витягуються такі дані, як name, price, brandId, typeId, info та файл img. Кожна нова фотографія задаватиметься з унікальним ідентифікатором для зображення і переміщатиметься на сервер у папку static. Новий запис обладнання з усіма значеннями імпортуватиметься до бази даних. При детальному огляді певного обладнання буде додано опис харакетеристик (info). У разі успішного додавання повертається JSON-відповідь із даними про нове обладнання. Якщо виникає помилка, викликається функція, яка буде описана в одному з файлів middleware з інформацією про помилку.

```
const uuid = require('uuid')
// рандомні айді для зображень, щоб не було повторень
const path = require('path'); // путь до папки зображень, який є в node.js
const {Device, DeviceInfo} = require('../models/models')
const ApiError = require('../error/ApiError');

class DeviceController {
async create(req, res, next) {
try {
        let {name, price, brandId, typeId, info} = req.body
        const {img} = req.files
        let filename = uuid.v4() + ".jpg"
        img.mv(path.resolve(__dirname, '..', 'static', filename))
        const device = await Device.create({name, price, brandId, typeId, img:
        filename})
        if (info) { // перевіряємо чи додано опис
                info = JSON.parse(info)
                // вертаємо в JS об'єкт
                info.forEach(i =>
                DeviceInfo.create({
                title: i.title,
                description: i.description,
                deviceId: device.id})
                )
        } return res.json(device)
        } catch (e) {
                next(ApiError.badRequest(e.message))
        }
}…
```

Наступним методом “getAll” обробляється GET-запит для отримання списку обладнання. Він дивиться на параметри фільтрації, такі як бренд або тип обладнання, і скільки обладнань має відображатись на сторінці. Потім він звертається до бази даних, щоб знайти потрібне обладнання. Отримані дані надсилаються користувачу, щоб він міг побачити список обладнання на своєму пристрої.

```
… async getAll(req, res ) {
    let {brandId, typeId, limit, page} = req.query
    page = page || 1
    limit = limit || 8 // задано ліміт відображень на сторінку
    let offset = page * limit - limit
    let devices;
    if(!brandId && !typeId) {
        devices = await Device.findAndCountAll({limit, offset})
    }
    if(brandId && !typeId) {
        devices = await Device.findAndCountAll({
            where: {brandId}, limit, offset
        })
    }
    if(!brandId && typeId) {
        devices = await Device.findAndCountAll({
            where: {typeId}, limit, offset
        })
    }
    if(brandId && typeId) {
        devices = await Device.findAndCountAll({
```

```
                where: {typeId, brandId}, limit, offset
        })
    }
    return res.json(devices)
    }…
    …}
    module.exports = new DeviceController()
```

За аналогією з попереднім методом, розробляємо метод “getOne”, який буде обробляти такий же GET-запит, однак для отримання інформації про одне обладнання за його унікальним ідентифікатором, яке витягується з параметрів запиту. Якщо є додаткові характеристики про обладнання (info), також повертає результат у вигляді JSON-відповіді.

Для кожного з маршрутів (userRouter, typeRouter, brandRouter) та контролерів (brandController, typeController, userController) процес розробки буде приблизно схожим за попереднім описом.

Тепер більш детально – deviceRouter та deviceController призначені для управління обладнанням, вони містять логіку, яка дозволяє створювати, отримувати обладнання та інформацію з бази даних. userRouter у свою чергу визначає маршрути зв'язування з користувачами, такі як реєстрація, вхід до системи та перевірка ролі. typeRouter і brandRouter більш схожі між собою і визначають маршрути для позначення типів та брендів обладнання. Відповідні контролери містять логіку для обробки запитів, пов'язаних з користувачами, типами та брендами відповідно. Виходячи з цього, необхідність описувати кожен файл відсутня.

Після створення маршрутизації для інших компонентів як user, brand, type, необхідно об'єднати маршрут основним файлом. Це дозволить легко структурувати маршрути, роблячи їх більш читаними. У папці routes файл index.js визначить шляхи для всіх інших маршрутів:

```
const Router = require('express')
const router = new Router()
const deviceRouter = require('./deviceRouter')
const userRouter = require('./userRouter')
const brandRouter = require('./brandRouter')
const typeRouter = require('./typeRouter')

router.use('/user', userRouter)
router.use('/type', typeRouter)
router.use('/brand', brandRouter)
router.use('/device', deviceRouter)

module.exports = router
```

У результаті для підключення маршрутизації до сервера в кореневому файлі index.js, де запускали сервер, необхідно проініціалізувати маршрути, а також пакети і їх функції:

– app.use(cors()) – дозволить серверу приймати запити з різних джерел;

– app.use(express.json()) – дозволить розпізнати JSON-запит, який повертається у контролерах;

– app.use(express.static(path.resolve(__dirname, 'static'))) – дозволить звернутися до відповідного зображення з папки static;

– app.use(fileUpload({})) – дістає необхідні зображення з об'єкта req.files, доданим у файлі deviceController.js;

– app.use('/api', router) – додає маршрути, визначені основним файлом у папці routes.

Цей етап розробки важливий для забезпечення того, щоб серверна частина правильно обробляла вхідні запити та повертала відповідні значення. Було

налаштовано маршрутизацію та створено контролери, які відповідають за логіку обробки запитів.

3.6 Створення візуальної структури проекту

Створення візуальної частини веб-додатку – важливий етап розробки, який відповідає за взаємодію користувачів із системою. Переходячи до розробки frontend частини проекту необхідно сформувати структуру проекту та завантажити необхідні npm пакети.

Згідно за аналізом та використання стеку технологій, будемо використовувати такі технології:

– основний фреймворк React, який дозволить створити компоненти та керувати додатком;

– бібліотека MobX, яка дозволить керувати станом у React-додатках та полегшить знаходження та оновлення даних;

– бібліотека React Router Dom для посторінкової навігації в React-додатках;

– бібліотека Axios, яка дозволить отримувати та надсилати дані на сервер. Для завантаження в терміналі середи розробки перейдемо з папки server на client командами “cd ..” та “cd client”. Першим кроком розвернемо React-додаток командою “npx create-react-app .”. Після завантаження та встановлення необхідних даних, видалимо зайві файли в папці src, залишаючи тільки файли App.css, App.js та index.js. Наступним кроком завантажуємо останні пакети командою “npm install mobx mobx-react-lite react-router-dom axios”.

Без використання стилів, React-додаток був би дуже не зручний у використанні. Для реалізації інтерфейсу користувача згідно з макетами візуальної частини проекту використовуємо бібліотеку React Bootstrap. Вона прискорить етап налаштування стилів і розташування об'єктів на екрані користувача. Встановлюємо її командою “npm install react-bootstrap bootstap” та

імпортуємо в файл html в папці public. Одночасно з цим відредагуємо файл html видаляючи не потрібні блоки як метатегі та імпортуючи потрібний шрифт. Підсумковий варіант виглядає подібним чином:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-scale=1" />
<link
rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css"
integrity="sha384-
9aIt2nRpC12Uk9gS9baDl411NQApFmC26EwAOH8WgZl5MYYxFfc+NcPb
1dKGj7Sk"
        crossorigin="anonymous"
/>
<link rel="preconnect" href="https://fonts.googleapis.com">
<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
<link
href="https://fonts.googleapis.com/css2?family=Manrope:wght@200..800&dis
play=swap" rel="stylesheet">
<title>React App</title>
</head>
<body>
<noscript>You need to enable JavaScript to run this app.</noscript>
<div id="root"></div>
</body>
</html>
```

Доповнюючи структуру візуальної частини проекту створено папки:

– store, де відбувається взаємодія з MobX та зберігання певних даних;

– pages, де зберігаються компоненти React, які є кореневими файлами сторінок інтерфейсу користувача;

– components, де описані блоки інтерфейсу користувача з яких формуватимуться сторінки.

Після задання структури клієнтської частини веб-додатку, наповнюємо її відповідними компонентами.

3.7 Розробка сторінок та їх компонентів

Спочатку налаштовуємо маршрутизацію на конкретні сторінки веб-додатку. Створено папку utils та файл consts.js, що містить константи маршрутів. Вони будуть використані в інших файлах для навігації. Код файлу consts.js:

```
export const ADMIN_ROUTE = '/admin'
export const LOGIN_ROUTE = '/login'
export const REGISTRATION_ROUTE = '/registration'
export const SHOP_ROUTE = '/'
export const DEVICE_ROUTE = '/device'
```

Далі визначаємо які сторінки необхідні у проекті та створюємо у папці pages відповідні файли:

– Admin.js – сторінка адмін панелі, згідно з макетом представленим раніше (рис. 2.6);

– Auth.js – сторінка авторизації, згідно з макетами (Рис. 2.2 - 2.3);

– Shop.js – загальна сторінка застосунку, згідно з макетом (Рис. 2.4);

– DevicePage.js – сторінка детального перегляду обладнання, згідно з макетом (рис. 2.5).

Створюємо файл routes.js, який буде визначати маршрути програми та дозволить користувачам переміщатися між сторінками Admin, Auth, Shop та DevicePage:

```
import Admin from "./pages/Admin"
import Shop from "./pages/Shop";
import Auth from "./pages/Auth";
import DevicePage from "./pages/DevicePage";
import {ADMIN_ROUTE, DEVICE_ROUTE, LOGIN_ROUTE, REGISTRATION_ROUTE, SHOP_ROUTE} from "./utils/const";
```

Імпортовано компоненти сторінок з папки pages, які будуть використані для маршрутизації, а також константи маршрутів з файлу consts.js.

```
export const authRoutes = [
    {
        path: ADMIN_ROUTE,
        Component: Admin
    },
]
```

Визначені маршрути лише для авторизованих користувачів. У цьому випадку – сторінка адміністраторської панелі.

```
export const publicRoutes = [
    {
        path: SHOP_ROUTE,
        Component: Shop
    },
    {
```

```
        path: LOGIN_ROUTE,
        Component: Auth
    },
    {
        path: REGISTRATION_ROUTE,
        Component: Auth
    },
    {
        path: DEVICE_ROUTE + '/:id',
        Component: DevicePage
    },
]
```

Ці маршрути задають доступ для усіх користувачів, визначаючи які компоненти повинні відображатися для різних маршрутів.

Для визначення авторизованого користувача створено файл AppRouter у папці components. У ньому визначається властивість об'єкта user і якщо користувач авторизований, додається маршрут з authRoutes, інакше тільки publicRoutes. Також реалізовано перенаправлення на основний маршрут SHOP_ROUTE, якщо інший не збігається.

```
import React, { useContext } from 'react';
import { Switch, Route, Redirect } from 'react-router-dom'
import { authRoutes, publicRoutes } from '../routes';
import { SHOP_ROUTE } from '../utils/const';
import { Context } from "../index";
const AppRouter = () => {
    const {user} = useContext(Context)
    console.log(user)
```

```
        return (
        <Switch>
        {user.isAuth === true && authRoutes.map(({path, Component}) =>
                <Route key={path} path={path} component={Component} exact
                />
        )}
        {publicRoutes.map(({path, Component}) =>
                <Route key={path} path={path} component={Component} exact
                />
        )}
        <Redirect to={SHOP_ROUTE}/>
        </Switch>
        );
};
export default AppRouter;
```

Додаємо блок <AppRouter /> в кореневий файл index.js. Саме він буде задавати структуру певної сторінки.

Розглянемо етап розробки сторінок на основі створення адміністраторської панелі, до якої має доступ лише авторизований користувач з роллю “ADMIN”. Розміщуватимемо згідно макету кнопку входу в адмін-панель будемо в навігаційному меню (рис. 2.2). Після введення даних користувача на сторінці авторизації та успішної перевірки наявності таких даних у базі даних, користувачеві відображатиметься відповідна кнопка. Якщо користувач натисне на неї, на серверній стороні checkRoleMiddleware перевірить користувача на володіння таким доступом. У разі збігу та користувач є адміністратором, його перенаправить на сторінку додавання нового типу, бренду та обладнання з усіма значеннями. В папці components створено структуру навігаційного меню файлом NavBar.js. Блок <NavBar /> також додано до кореневого файлу index.js, щоб він

був на усіх сторінках. Кнопка “Адмін панель” перенаправлятиме користувача на сторінку адміністраторської панелі.

Перш ніж розробляти сторінку Admin.js, визначаємо, що необхідно сформувати модальні вікна, які будуть відображені при натисканні якоїсь з кнопок додавання. Для цього створимо в папці components ще одну папку modals, яка буде зберігати три файли з модальними вікнами.

Переходимо до розробки самої сторінки адмін-панелі. Вона дозволить адміністратору створити нові типи, бренди та обладнання через модальні вікна.

```
import React, { useState } from 'react';
import { Button, Container } from 'react-bootstrap';
import CreateBrand from '../components/modals/CreateBrand';
import CreateType from '../components/modals/CreateType';
import CreateDevice from '../components/modals/CreateDevice';
```

Імпортовано веб-хук “useState” для управління станом у відповідних компонентах. Оскільки ми будемо використовувати кнопки та обертати їх у контейнер, імпортуємо їх із бібліотеки react-bootstrap. При натисканні на відповідні кнопки відображатимуться модальні вікна, які підвантажуються з папки modals.

```
const Admin = () => {
        const [brandVisible, setBrandVisible] = useState(false);
        const [typeVisible, setTypeVisible] = useState(false);
        const [deviceVisible, setDeviceVisible] = useState(false);
…
```

Оголошено три стани для відображення модальних вікон.

```
… return (
    <Container
        <Button onClick={() => setTypeVisible(true)}>
            Додати новий тип
        </Button>
        <Button onClick={() => setBrandVisible(true)}>
            Додати новий бренд
        </Button>
        <Button onClick={() => setDeviceVisible(true)}>
            Додати нове обладнання
        </Button>
        <CreateBrand show={brandVisible} onHide={() =>
        setBrandVisible(false)}/>
        <CreateDevice show={deviceVisible} onHide={() =>
        setDeviceVisible(false)}/>
        <CreateType show={typeVisible} onHide={() =>
        setTypeVisible(false)}/>
    </Container>
    );
    };
    export default Admin;
```

Сформовано блоки з трьома кнопками "Додати новий тип", "Додати новий бренд", "Додати нове обладнання". При натисканні на одну з кнопок викликається функція зміни стану видимості модальних вікон "setTypeVisible", "setBrandVisible", "setDeviceVisible".

Розглянемо детальніше структуру модальних вікон, яка сформована файлами CreateBrand.js, CreateDevice.js, CreateType.js на прикладі модального

вікна "Додати тип". Аналогічно з файлом Admin.js імпортуємо необхідні модулі у файл CreateType.js:

```
import React, {useState} from 'react';
import Modal from "react-bootstrap/Modal";
import {Form, Button} from "react-bootstrap";
```

У цьому випадку нам не потрібно обертати в контейнер, оскільки блок є компонентом на сторінці адмін-панелі. Однак потрібно сформувати певне спливаюче вікно з кнопками та формою для передачі назви нового типу в базу даних. Для цього підвантажуємо компоненти Modal, Form, Button з бібліотеки react-bootstrap.

```
const CreateType = ({show, onHide}) => {
return (
        <Modal
                show={show}
                onHide={onHide}
                centered
        >
        <Modal.Header closeButton>
                <Modal.Title>
                Додати тип
                </Modal.Title>
         </Modal.Header>
        <Modal.Body>
        <Form>
                <Form.Control
                value={value}
```

```
                onChange={e => setValue(e.target.value)}
                placeholder={"Введіть назву типу"}
                />
        </Form>
        </Modal.Body>
        <Modal.Footer>
                <Button onClick={onHide}>Закрити</Button>
                <Button onClick={addType}>Додати</Button>
        </Modal.Footer>
        </Modal>
);
};
export default CreateType;
```

Структура коду зроблена за аналогією зразка з офіційного сайту React Bootstrap [23]. Сформовано структуру модального вікна:

– блок “Modal.Header” – заголовок модального вікна з кнопкою закриття;

– блок “Modal.Body” – тіло модального вікна, де можна ввести назву нового типу обладнання;

– блок “Modal.Footer” – нижня частина модального вікна з кнопками закриття цього вікна або додавання введеного типу.

Останнім кроком було прийнято рішення допрацювати зовнішній вигляд інтерфейсу користувача використовуючи готові блоки з бібліотеки react bootstrap і створити окремий файл custom.css, який перепризначає деякі стилі в класах bootstrap, а також імпортує шрифт Manrope. Використання готових стилів з bootstrap полягає у привласненні якомусь блоку атрибут className. Наприклад className = 'd-flex flex-column' робить блок гнучким до розташування і розташовує елементи всередині блоку один під одним займаючи всю доступну

ширину. А атрибут variant='outline-danger' змінює синій стиль за замовчуванням на червоний контур та червоний текст для кнопки.

За аналогією з попередніми файлами створюємо головну сторінку файлом Shop.js з такими компонентами:

– TypeBar.js – для сортування за типом;
– BrandBar.js – для сортування за брендом;
– DeviceItem.js – для відображення певного обладнання;
– DeviceList.js – для формування списку обладнання за DeviceItem.js;
– Pages.js – для навігації по сторінкам обладнання, якщо їх забагато.

Також створюємо більш прості, з точки зору розробки, сторінки авторизації та реєстрації файлом Auth.js та сторінку з докладним описом обладнання файлом DevicePage.js.

Підсумовуючи, на цьому етапі були розроблені всі сторінки веб-додатку.

3.8 Налаштування взаємодії клієнта з сервером

Останнім етапом розробки залишається лише налаштувати взаємодію з сервером. З цим нам допоможе бібліотека axios для виконання http-запиту до сервера. Для цього створюємо нову папку http. В неї необхідно налаштувати роботу axios та розробити функції, які відповідатимуть за виконання запитів для користувачів та обладнань.

Першим кроком створюємо головний файл в папці http – index.js:

```
import axios from 'axios';
const $host = axios.create({
baseURL: process.env.REACT_APP_API_URL
})
```

Створено axios з базовою конфігурацією.

```
const $authHost = axios.create({
baseURL: process.env.REACT_APP_API_URL
});

Створено axios з конфігурацією для авторизації.

const authInterceptor = config => {
config.headers.authorization = `Bearer ${localStorage.getItem('token')}`;
return config;
};
$authHost.interceptors.request.use(authInterceptor);
export {
        $host,
        $authHost
}
```

Розроблено функцію безпеки для додання токену авторизації до запитів з авторизацією. Тепер можемо використовувати експортований axios.

Другим кроком необхідно розробити такі функції, які будуть виконувати запити створення або отримання обладнання. Для цього створюємо файл deviceAPI.js:

```
import {$authHost, $host} from "./index";
export const createType = async (type) => {
   const {data} = await $authHost.post('api/type', type)
   return data
}

Створено функцію створення нового типу обладнання.
```

```
export const fetchTypes = async () => {
    const {data} = await $host.get('api/type')
    return data
}
```

Створено функцію отримання всіх типів обладнання.

```
export const createBrand = async (brand) => {
    const {data} = await $authHost.post('api/brand', brand)
    return data
}
export const fetchBrands = async () => {
    const {data} = await $host.get('api/brand', )
    return data
}
export const createDevice = async (device) => {
    const {data} = await $authHost.post('api/device', device)
    return data
}
export const fetchDevices = async (typeId, brandId, page, limit = 5) => {
    const {data} = await $host.get('api/device', {params: {
            typeId, brandId, page, limit
        }})
    return data
}
```

Аналогічно зі створенням та отримання типів обладнань, розроблено функції для брендів і самого обладнання.

```
export const fetchOneDevice = async (id) => {
    const {data} = await $host.get('api/device/' + id)
    return data
}
```

Також необхідно було розробити функцію для отримання певного обладнання по його ідентифікатору для нашої сторінки, описаної в файлі DevicePage.js.

Після створення функцій в окремій папці, потрібно доопрацювати компоненти та сторінки викликавши відповідні функції у цих файлах. Розглянемо на прикладі виклику функції у файлі компонента модального вікна CreateType.js, оскільки він вже був описаний у підрозділі 3.7:

```
const [value, setValue] = useState('')
const addType = () => {
    createType({name: value}).then(data => {
        setValue('');
        onHide();
    });
}
```

Ініціалізовано стан “value” за допомогою веб-хука useState і описано функцію addType. Саме змінна “value” отримує значення з форми від адміністратора і передає її на сервер за допомогою deviceAPI.

Останнім кроком розробляємо файл userAPI.js з функціями, такі як реєстрація, авторизація та перевірки валідності токена для користувачів. Докладно описувати код потреби немає, тому що реалізація відбувається як і у файлі deviceAPI.js, описаному вище.

Саме така структура забезпечує ефективну та точну взаємодію клієнта із сервером. Вона дозволяє надсилати запити на сервер та отримувати на них відповідь.

3.9 Посібник із користування системою автоматизованого додавання промислового обладнання

Автоматизована система додавання промислового обладнання на основі технічних характеристик реалізована за допомогою веб-програми. Більшість подібних рішень не мають такої функції з автоматизацією за допомогою зручного веб-інтерфейсу. Якщо навіть це великий і дорогий проект, в якому реалізовані подібні функції – інтерфейс для власника чи адміністратора найчастіше не виглядає так зручно та приємно. Також подібні сайти мають величезну структуру та кількість даних у базі, будь-яка взаємодія навантажує систему і може негативно вплинути на швидкодію та користування сайтом з боку звичайних користувачів. Хоча наш інтерфейс розроблений максимально простим та зручним для користувача, для полегшення використання та розуміння тих чи інших функцій сайту необхідно описати посібник із користування цим проектом.

Для того, щоб не витрачати зайвий час нового користувача, який перейшов на основну сторінку сайту, він може відразу отримати необхідну інформацію про обладнання. Реалізований варіант основної сторінки, на яку потрапляє цей користувач, складається з (рис. 3.12):

– зверху навігаційне меню, яке відкриває можливості авторизації, реєстрації та кнопка із заголовком, яка відповідає за швидке повернення на основну сторінку;

– картки з фотографією та назвою обладнання;

– список кнопок із сортуванням за типом;

– список кнопок із сортуванням за брендами.

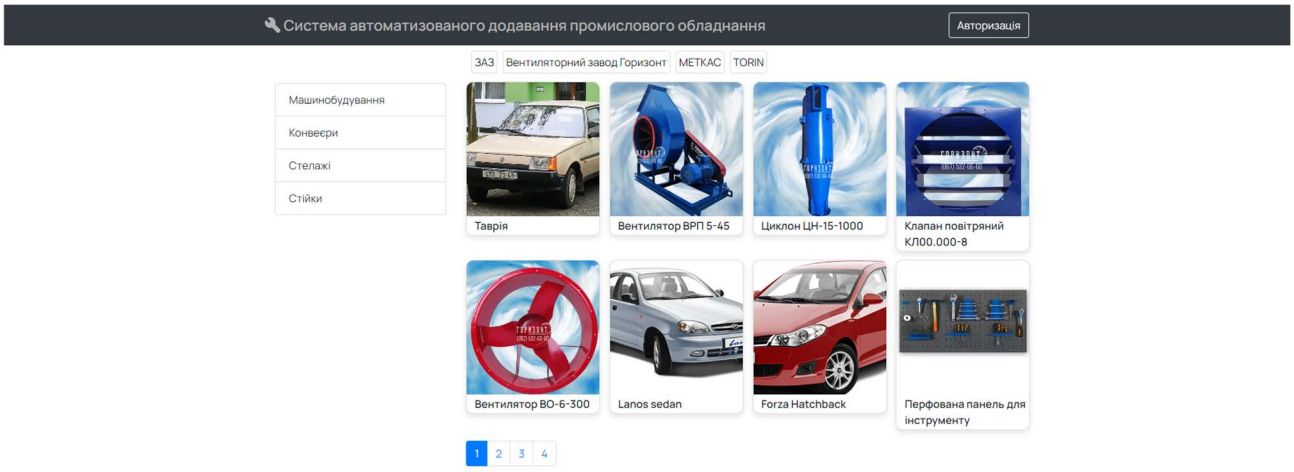

Рисунок 3.12 – Головна сторінка веб-додатку

Розглянемо ситуацію, якщо користувач зацікавлений у певному обладнанні та не збирається дивитися всі сторінки карток з обладнанням. Він може знайти потрібне йому обладнання, додавши фільтр за певним типом, розташованим у лівій частині сторінки, а також брендом, розташованим над картками, якщо це необхідно. Список та вміст карток у такому випадку будуть змінюватися відповідно до вибраних параметрів.

Після того, як користувач знайшов потрібне йому обладнання на картці і хоче докладніше прочитати про нього або подивитися велику фотографію – йому необхідно натиснути в будь-яке місце цієї картки. Після натискання, користувача перемістить на сторінку з обраним обладнанням. Якщо йому необхідно повернутись на основну сторінку з усім обладнанням – це можна виконати натиснувши на іконку або напис “Система автоматизованого додавання промислового обладнання” у навігаційному меню, розташованому зверху.

Почнемо опис функціоналу для адміністратора системи. Насамперед адміністратору необхідно зайти в систему. Виконати це можна натисканням на кнопку “Авторизація”, розташованої в навігаційному меню. Його перемістить на сторінку авторизації чи реєстрації (рис. 3.13).

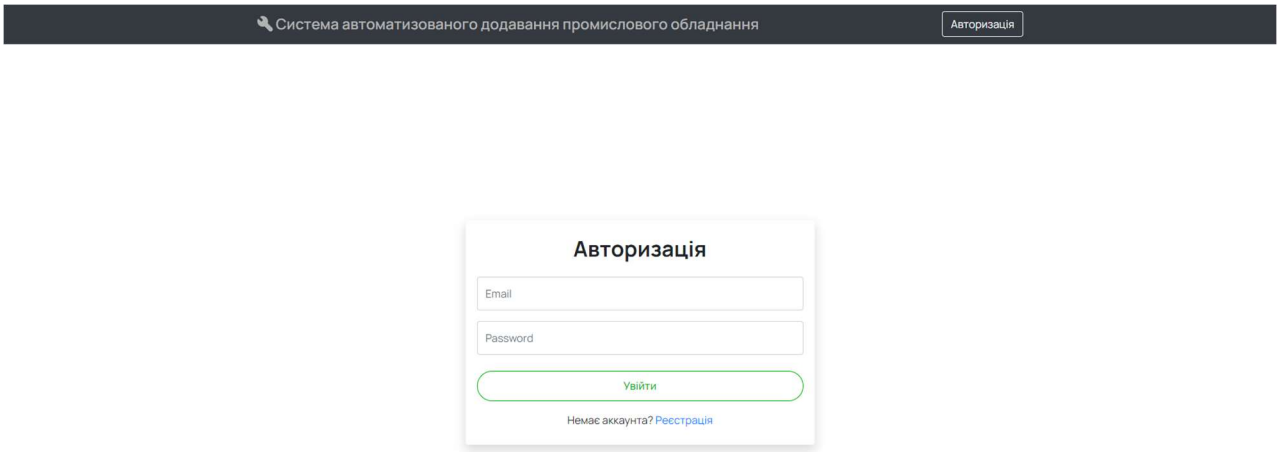

Рисунок 3.13 – Сторінка авторизації або реєстрації

У відповідні поля адміністратор повинен ввести свою пошту та пароль, за якими він реєструвався, а також натиснути клавішу “Увійти”. На цьому етапі система перевірить введенні дані користувачем і переведе на основну сторінку веб-сайту.

Після успішної авторизації користувача, на місці кнопки “Авторизація” з'являться дві нові кнопки – “Адмін панель” та “Увійти”. Для користування основними функціями проекту адміністратору необхідно натиснути клавішу адміністраторської панелі. Після чого його перекидає на сторінку адміністратора (рис. 3.14), де розташовані три кнопки – “Додати новий тип”, “Додати новий бренд” та “Додати нове обладнання”.

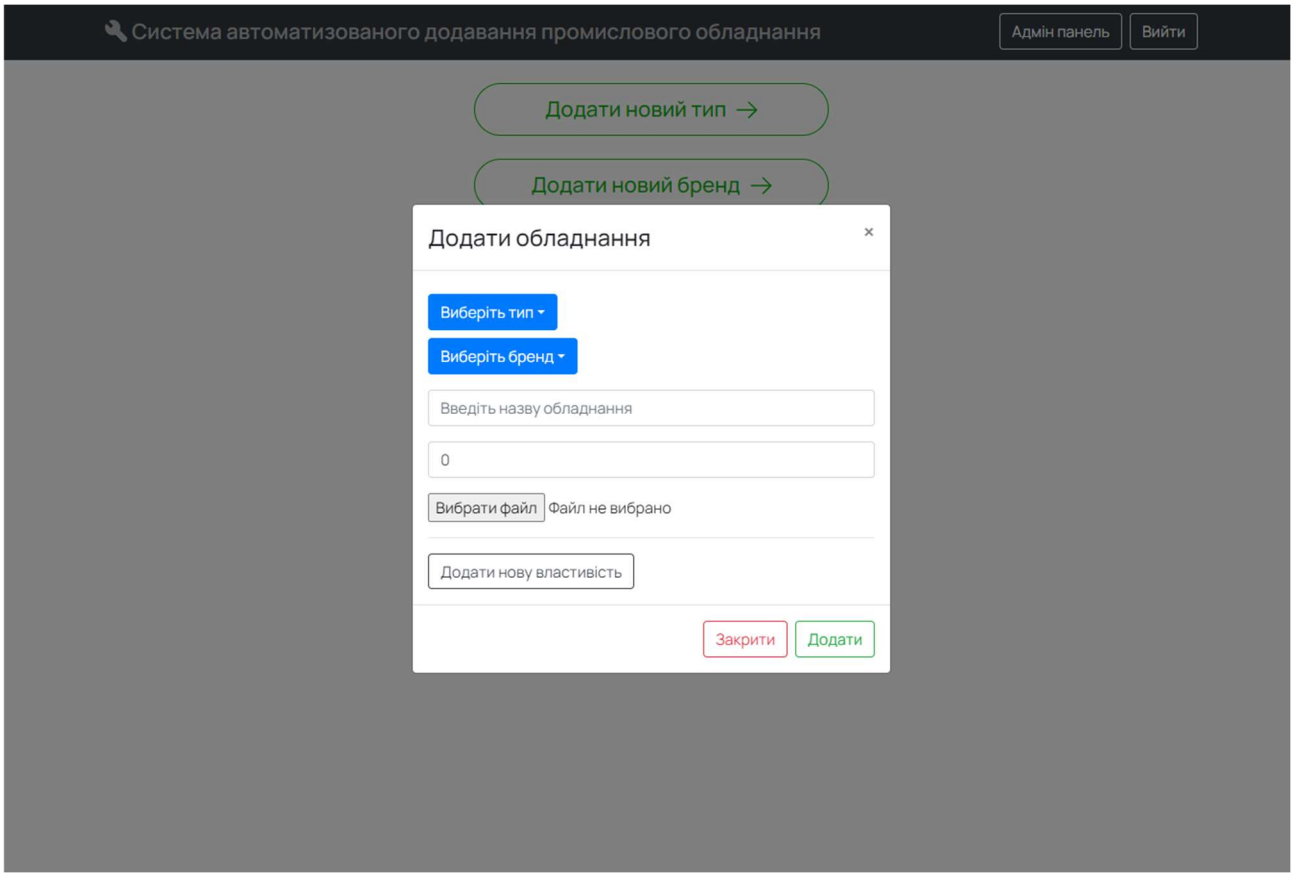

Рисунок 3.14 – Сторінка адміністратора

Якщо адміністратору потрібно додати нове обладнання, він натискає відповідну кнопку. Відкриється додаткове модальне вікно, де адміністратор може ввести всі параметри обладнання, включаючи його тип, бренд, назву, ціну, характеристики та завантажити фотографію зі свого пристрою. Після заповнення всіх необхідних властивостей натискає нижче кнопку “Додати” або, якщо передумає, кнопку “Закрити”. Ці кнопки відповідно відразу додають нове обладнання або закривають модальне вікно. За аналогією адміністратор також може додати новий тип чи бренд.

Завершуючи, описано посібник із користування системою для нових користувачів, який полегшить та прискорить їхню роботу. Саме воно створить умови приємного користування інтерфейсом, роблячи його більш зрозумілим.

Репозиторій GitHub за посиланням: https://github.com/Elronic3/selection.

ВИСНОВКИ

В ході виконання кваліфікаційної роботи був виконаний аналіз наукової діяльності кафедри комп'ютерно-інтегрованих технологій, автоматизації та мехатроніки Харківського національного університету радіоелектроніки, була описана наукова-дослідна робота та отримані результати.

В ході виконання кваліфікаційної роботи було вивчено предметну область розробки системи підбору. Було вивчено існуючи методи розробки веб-сторінок. Були виявлені існуючі проблеми даного підходу і змодельована модель її рішення. Були виконані наступні завдання:

– вивчена предметна область вже існуючих програмних рішень;

– розглянуто різні аспекти технологічної частини, такі як функціональність, масштабованість, безпека, зручність розробки, підтримки, та технічні можливості;

– здійснено пошук оптимальних технологій та інструментів, які найкраще відповідають вимогам проекту та забезпечать успішне його виконання;

– розроблено макети інтерфейсу користувача и діаграми бази даних;

– згідно з макетами, було створено веб-додаток;

– розроблено алгоритм роботи та програмне забезпечення для керування базою даних;

– наведено опис посібника із користування системою, яке розкрило реальне користування веб-додатком.

У результаті було досягнуто основна мети роботи: створено систему автоматизованого додавання промислового обладнання на основі технічних характеристик, що спростить процес додавання обладнання для промислових підприємств та підвищить їхню ефективність.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. ДСТУ 3008-15. Документація. Звіти у сфері науки та техніки. структура та правила оформлення. Введ. 2015-06-22. К. Держстандарт України, 2017. - 29 с.;

2. Невлюдов І. Ш. Дипломне проектування для студентів усіх форм навчання спеціальностей 151 «Автоматизація та комп'ютерно-інтегровані технології» [Текст]: навч. посіб. / І. Ш. Невлюдов, А. О. Андрусевич, О. В. Токарєва, Г. В. Пономарьова. – Київ: НАУ, – 2016. – 320 с.

3. Методичні вказівки з підготовки кваліфікаційної роботи бакалавра для здобувачів першого (бакалаврського) рівня вищої освіти спеціальності 151 Автоматизація та комп'ютерно-інтегровані технології освітньої програми «Системна інженерія» / Упоряд.: І.Ш. Невлюдов, О.М. Цимбал, О.В. Токарєва, А.І. Бронніков. Харків: ХНУРЕ, 2022. - 66 с.;

4. Thomasnet – Product Sourcing and Supplier Discovery: веб-сайт. URL: https://www.thomasnet.com/ (дата звернення 08.05.2024);

5. Кафедра "Інформаційно-мережна інженерія", ХНУРЕ: веб-сайт. URL: https://cn.nure.ua/veb-rozrobnik/ (дата звернення 08.05.2024);

6. MDN Mozilla Developer Network: веб-сайт. URL: https://developer.mozilla.org/ru/docs/Learn/Getting_started_with_the_web/HTML_basics (дата звернення 08.05.2024);

7. Методичні рекомендації до першого модуля нормативної дисципліни "Програмування та підтримка веб-застосувань" для студентів спеціальності 122 "Комп'ютерні науки" першого (бакалаврського) рівня / Рец. Собчук О.М., Гайдай С.І. Луцьк: ВНУ, 2022 – 26 с.;

8. Методичні аспекти розробки динамічних елементів WEB-сайтів з використанням JavaScript-сценаріїв. / Упоряд. Василенко Я.П., Прибула І.В., Тернопіль: ТНПУ, 2023 – 33 с.;

9. TypeScript: JavaScript With Syntax For Types.: веб-сайт. URL: https://www.typescriptlang.org/ (дата звернення 09.05.2024);

10. Back-end розробка. Веб-студія Brainlab.: веб-сайт. URL: https://brainlab.com.ua/uk/blog-uk/back-end-rozrobka (дата звернення 10.05.2024);

11. WEZOM. Все що потрібно знати про Node.js.: веб-сайт. URL: https://wezom.com.ua/ua/blog/vse-chto-nuzhno-znat-o-nodejs (дата звернення 10.05.2024);

12. ПАРСИНГ ДАНИХ З ВЕБ СТОРІНОК / Брітвін А., Ткачук Р., Львів: ЛДУ БЖД, 2021 – 13 с.;

13. Що таке лендінг та як його створити: веб-сайт. URL: https://hostiq.ua/blog/ukr/what-is-landing/ (дата звернення 26.05.2024);

14. FIGMA VS PHOTOSHOP / Федоров В.О., Мелітополь: ТДАТУ, 2019 – 81 с;

15. node.bcrypt.js: веб-сайт. URL: https://www.npmjs.com/package/bcrypt (дата звернення 27.05.2024);

16. Моделі та методи кіберфізичних виробничих систем в концепції Industry 4.0 : монографія / І. Ш. Невлюдов, В. В. Євсєєв, А. О. Андрусевич, С. С. Максимова ; – Oktan Print – Prague. 2023. – 321 с.;

17. Crypto++ Benchmark: веб-сайт. URL: https://www.cryptopp.com/benchmarks-amd64/benchmarks-amd64.html (дата звернення 27.05.2024);

18. БЕЗПЕЧНИЙ СПОСІБ ОБМІНУ JWT В ASP.NET CORE + SPA / Тімошенко К.О., Кропивницький: ЦНТУ, 2019 – 110 с.;

19. Аналіз інструменів для створення розширень visual studio code / Черневський, Н. О., Шатайло, В. А., Вінниця: ВНТУ, 2023 – 1 с.;

20. aCode – Ключові слова та ідентифікатори: веб-сайт. URL: https://acode.com.ua/urok-19-klyuchovi-slova-i-identyfikatory/ (дата звернення 05.06.2024);

21. ESLint: веб-сайт. URL: https://eslint.org/ (дата звернення 05.05.2024);

22. FoxmindEd – Як використовувати дебаггер у JavaScript?: веб-сайт. URL: https://foxminded.ua/debugger-js/ (дата звернення 05.06.2024);

23. Components | Modals | React Bootstrap: веб-сайт. URL: https://react-bootstrap.netlify.app/docs/components/modal/ (дата звернення 06.06.2024);