НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено

Завідувач кафедри

_____ О.В. Коваль

(підпис) (ініціали, прізвище)

“___”_____2020р.

ДИПЛОМНА РОБОТА

на здобуття ступеня бакалавра

з напряму підготовки 122 “ Комп’ютерні науки та інформаційні технології ”

на тему «Заповнення галузевого реєстру інформаційних ресурсів кафедри»

Виконав: студент 4 курсу, групи ТР-61

Кошмак Владислав Леонідович _____

(прізвище, ім’я, по батькові) (підпис)

Керівник ст. викладач Колумбет В. П. _____

(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Консультант _____ _____ _____

(назва розділу) (вчені ступінь та звання, прізвище, ініціали) (підпис)

Рецензент_____ _____

(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____

(підпис)

Київ – 2020 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший рівень

Напрям підготовки 6.050101 “ Комп’ютерні науки”

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ О.В. Коваль
(підпис)
”___”_____2020р.

ЗАВДАННЯ
на дипломну роботу студенту
Кошмаку Владиславу Леонідовичу
(прізвище, ім’я, по батькові)

1. Тема роботи «Заповнення галузевого реєстру інформаційних ресурсів кафедри »

керівник роботи Колумбет Вадим Петрович старший викладач
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”__”___ 2020р. № __

2. Строк подання студентом роботи _____
3. Вихідні дані до роботи програмний застосунок розроблено у середовищі Visual Studio Code на платформі Node.js з використанням мови JavaScript.

4.Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Створити галузевий реєстр інформаційних ресурсів кафедри, проаналізувати використання мов запитів для доступу до бази даних. Розробити застосунок, що дасть змогу вносити інформаційні дані до бази даних без знання мови SQL користувачами даного застосунку.
5. Перелік ілюстративного матеріалу

«Мета роботи», «Актуальність теми», «Задачі, поставлені при виконанні», «Поняття баз даних», «Створення бази даних», «Дерево класів та зв'язки», «SQL-запит», «Взаємодія компонентів», «Функціональна схема системи», «Приклади роботи програми», «Висновки».

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання "___"_____201___ р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Затвердження теми роботи	25.10.19	
2.	Вивчення та аналіз задачі	27.11.19-16.12.19	
3.	Розробка архітектури та загальної структури системи	17.12.19-27.01.20	
4.	Розробка структур окремих підсистем	28.01.20-06.04.20	
5.	Програмна реалізація системи	07.04.20-04.05.20	
6.	Оформлення пояснювальної записки	05.05.20-04.06.20	
7.	Захист програмного продукту	08.06.20	
8.	Передзахист	08.06.20	
9.	Захист		

Студент _____ Кошмак В. Л.
(підпис) (прізвище та ініціали,)

Керівник роботи _____ Колумбет В. П.
(підпис) (прізвище та ініціали,)

АНОТАЦІЯ

Метою дипломної роботи є створення галузевого реєстру інформаційних ресурсів кафедри та розробка програмного що надає змогу вносити інформаційні ресурси до бази даних.

Об’єктом дослідження є галузевий реєстр інформаційних ресурсів. Було виконано огляд існуючих інформаційних ресурсів та зроблено висновки щодо їх переваг та недоліків. Створено клієнтський застосунок для роботи з інформаційним ресурсом. Продукт можна використовувати для запису даних в базу даних.

Загальний обсяг роботи: 54 сторінки, 29 ілюстрацій та 25 бібліографічних найменувань.

Ключові слова: SQL, Node.js, JavaScript, React.js, бази даних, інформаційні ресурси, галузевий реєстр, клієнт-серверна система.

ABSTRACT

The purpose of the thesis is to create a branch register of information resources of the department and the development of software that allows you to enter information resources into the database.

The object of research is the sectoral register of information resources. An overview of existing information resources was performed and conclusions were drawn about their advantages and disadvantages. A client application has been created to work with the information resource. The product can be used to write data to a database.

Total volume of work: 54 pages, 29 illustrations and 25 bibliographic titles.

Keywords: SQL, Node.js, JavaScript, React.js, databases, information resources, industry register, client-server system.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ І ТЕРМІНІВ

1) JavaScript (JS) — динамічна, об'єктно-орієнтована прототипна мова програмування.
2) Node.js — платформа з відкритим кодом для виконання високопродуктивних мережевих застосунків, написаних мовою JavaScript.
3) HTML — це мова тегів, якою пишуться гіпертекстові документи для мережі Інтернет
4) React — відкрита JavaScript бібліотека для створення інтерфейсів користувача, яка покликана вирішувати проблеми часткового оновлення вмісту веб-сторінки.
5) SQL — декларативна мова програмування для взаємодії користувача з базами даних, що застосовується для формування запитів, оновлення і керування реляційними БД.
6) GraphQL — це мова запитів і маніпуляції даними з відкритим кодом для API і середовище виконання для обслуговування запитів з наявних даних.
7) База даних – сукупність даних, організованих відповідно до концепції, яка описує характеристику цих даних і взаємозв'язки між їх елементами.

ВСТУП

Сьогодні інформаційні технології розвиваються так швидко, як ніколи раніше, що приводить до зростання обсягів інформації. Відповідно виникають задачі, з тим як їх зберігати та обробляти, з метою зручного та швидкого доступу до користувача.

З розвитком наукоємних сфер людської функціональності в сучасному світі зростає попит на комп'ютерні технології. Зростання обсягу інформації змушує знаходити нові способи її зберігання, уявлення, формалізації і систематизації, а також автоматичної обробки. Таким чином, зростає інтерес до всеосяжних баз знань, які можливо використовувати для різних практичних цілей. Такою ціллю є обробка та зберігання інформаційних ресурсів кафедри. Моя програма використовує для обробки семантичні зв’язки між документами одного виду. Адже документ потрібно не просто зберігати в базі даних, а й до якого саме сегменту впорядкованих даних внести той чи інший документ. В цьому і полягає основна проблема дипломної роботи, як саме правильно парсити вхідні документи та до якої частини бази даних їх занести.

Основною метою дипломної роботи є створення та заповнення інформаційних ресурсів кафедри та розробка програмного забезпечення, що було б доступним та легким у використанні для всіх користувачів і не потребувало б специфічних знань від користувача.

Для здійснення поставленої мети були висунуті такі завдання:

- проаналізувати інформаційні ресурси кафедри;
- описати систему зберігання ресурсів на кафедрі;
- структурувати методичні ресурси;
- створити базу даних;
- розробити програмний продукт.

1. ЗАДАЧА ЗАПОВНЕННЯ ГАЛУЗЕВОГО РЕЄСТРУ ІНФОРМАЦІЙНИХ РЕСУРСІВ КАФЕДРИ

Оскільки за роки існування кафедри кількість документів, методичних матеріалів та інших ресурсів зростає. Більшість матеріалі знаходяться в цифровому вигляді. Для їх зберігання створюються бази даних, але вони достатньо великі. І якщо при розробці бази даних не було правильно передбачено її майбутній розвиток – то через декілька років пошук та обробка даних стає досить складною. Знаходити дані у великих базах даних стає все важче, бо з року в рік цієї інформації стає все більше.

Саме тому була створена задача створити систему заповнення галузевого реєстру інформаційних ресурсів кафедри, для того щоб в подальшому простіше виконувалася організація, класифікація і керування цією великою кількістю даних. Для обробки інформації на кафедрі в дипломній роботі пропонується використати інформаційну систему ресурсів кафедри. Це певна структура, яка містить основні класи інформаційних ресурсів кафедри, їх взаємозв’язки, метадані та типи даних.

Дана задача включає в себе:

- аналіз предметної області;
- створення інформаційної системи кафедри;
- написання запитів до бази даних;
- можливість виводити інформацію у зручному вигляді;
- можливість переглядати документи.

До компонентів розробленої системи входить:

- JavaScript
- CSS
- HTML
- Node.js
- React.js
- MySQL

- Chrome DevTools

Важливо також виділити основні задачі програмного продукту та визначити, в яких цілях він буде використовуватися користувачами. Дану систему можна використовувати:

- для дослідження структури інформаційної системи;
- для отримання даних з інформаційної системи;
- для зручного та швидкого пошуку та перегляду даних;
- для розуміння основ заповнення інформаційної системи;

2. АНАЛІЗ ГАЛУЗЕВОГО РЕЄСТРУ ІНФОРМАЦІЙНИХ РЕСУРСІВ КАФЕДРИ

Спираючись на сукупність всіх перерахованих факторів у минулому розділі, виникає необхідність у реєстрі інформаційних ресурсів кафедри.

2.1 Поняття інформаційного ресурсу

Інформаційні ресурси — документи і масиви документів в інформаційних системах.

Інформаційна система — сукупність організаційних і технічних засобів для збереження та обробки інформації з метою забезпечення інформаційних потреб користувачів, комунікаційна система, що забезпечує збирання, пошук, оброблення та пересилання інформації.

2.2 Поняття галузевого реєстру інформаційних ресурсів кафедри

Галузевий реєстр інформаційних ресурсів кафедри — це інформаційна система, результати інтелектуальної та практичної діяльності, що сформовані в усіх сфері роботи кафедри, зафіксовані і систематизовані на відповідних матеріальних носіях інформації, як окремі документи і масиви документів, банки і бази даних та знань, усі види архівів і бібліотек, музейні фонди, інформаційні ресурси які обробляються й передаються в інформаційних системах загального призначення, інші ресурси, що містять дані, відомості і знання які є об’єктом права власності кафедри незалежно від форми власності на час їх створення і мають споживчу цінність, а також такі, що призначені для розвитку і задоволення потреб студентів та викладачів та підлягають захисту згідно з визначеною політикою безпеки й чинним законодавством.

Створення галузевого реєстру інформаційних ресурсів кафедри вимагає неперервного системного аналізу інформаційних ресурсів, їх взаємозв'язків та захищеності. При цьому реєстр слід розглядати як розподілену складну інформаційну систему, оскільки він наділений усіма властивостями складних систем, а саме: великою кількістю складових компонент, взаємодією з навколишнім середовищем, ієрархічною структурою та мінливістю у часі.

Реєстр має забезпечувати облік інформаційних ресурсів, згрупованих за певною спільною ознакою. Наприклад: за формою власності – реєстр інформаційних ресурсів кафедри; за тематикою – галузеві стандарти тощо.

До інформаційних ресурсів кафедри можна віднести всю належну інформацію включаючи окремі документи і масиви документів, які належать до наукового, навчально-методичного та організаційного напрямків роботи кафедри:

- навчальних планів; робочих навчальних планів;
- програм навчальних дисциплін, робочих програм кредитних модулів, плани практичних і семінарських занять, критерії оцінювання рейтингу студентів;
- підручники, навчальні посібники, конспекти лекцій;
- навчальні та методичні посібники до лабораторних робіт, курсових проектів,
- варіанти індивідуальних семестрових завдань, теми курсових проектів/робіт
- засоби діагностики для поточного та семестрового контролю результатів навчання та критеріїв оцінювання;
- завдання для проведення комплексних контрольних робіт з навчальних дисциплін та критеріїв оцінювання рівня підготовки студентів для проведення акредитації освітньої програми, моніторингу залишкових знань і вмінь;

2.3 Існуючі галузеві реєстри інформаційних ресурсів кафедри

Для роботи я за приклад взяв сайт який є галузевим реєстром інформаційних ресурсів кафедри – це https://ela.kpi.ua/ (Рис 2.1). Даний сайт підходить під всі визначення

галузевого реєстру і напряму відноситься до галузевих реєстрів кафедри. Але для того щоб занести до нього дані, потрібно мати деякий дозвіл, права на який можуть затягнутися і як показує досвід - на важливу інформацію на одному ресурсі краще не зберігати, потрібна більша розподіленість. Це і має вирішити моя програма.

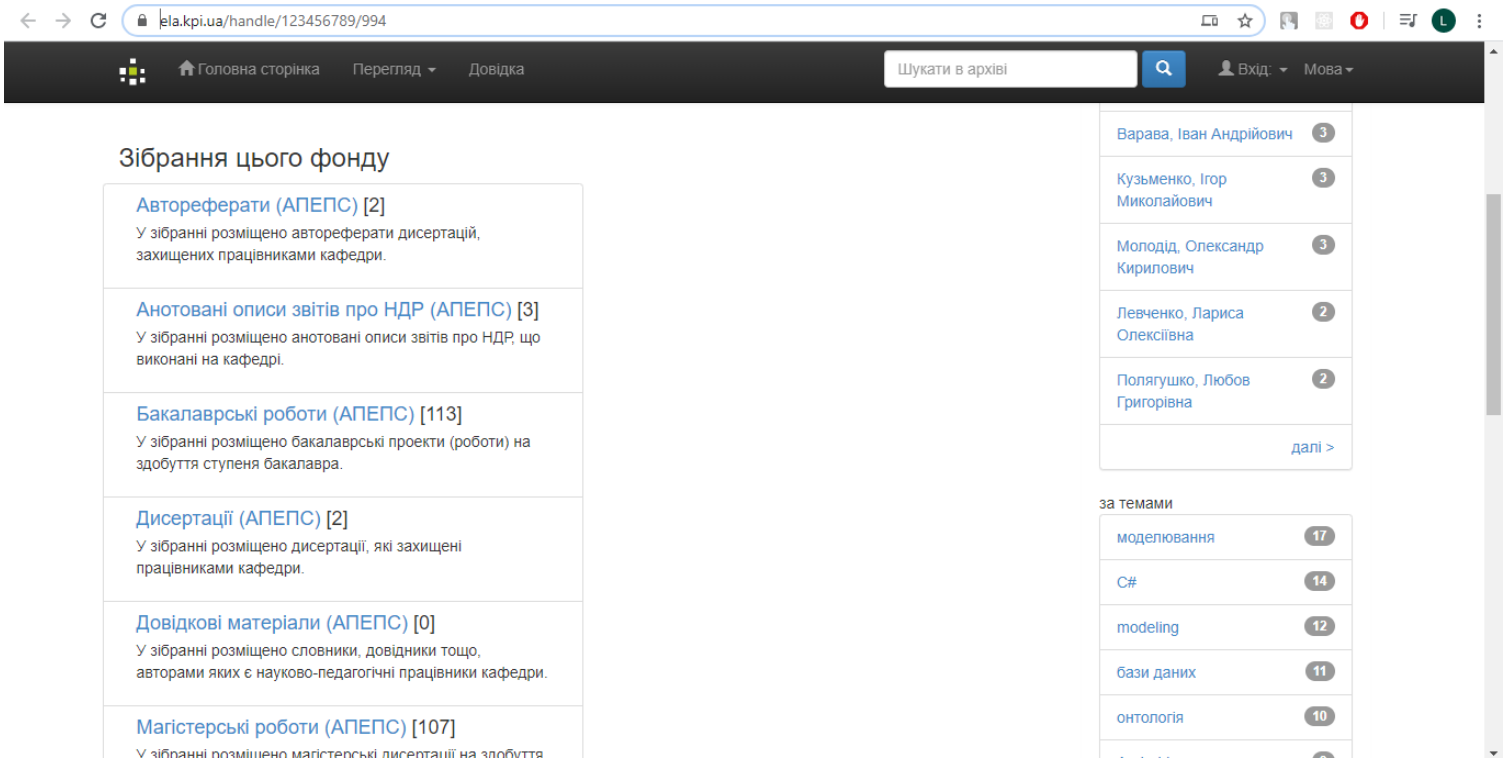

Рис. 2.1 – галузевий реєстр інформаційних ресурсів ela.kpi.ua

3. ЗАСОБИ РОЗРОБКИ ПРОГРАМНОГО ПРОДУКТУ

Для розробки програмного продукту було використано наступні засоби:

- середовище Visual Studio Code;
- програмна платформа Node.js
- система MySQL Workbench для створення БД.
- система GitKraken для керування репозиторіями
- набір інструментів для веб-розробників Chrome DevTools

Перш за все для доступу до методичних ресурсів необхідно створити БД. Саме для цього і використовується MySQL Workbench.

За допомогою середовища Visual Studio Code мовою JavaScript та з використанням бібліотеки React.js було спроектовано та розроблено веб додаток. Додаток запускається завдяки програмній платформі Node.js, керування версіями коду програми забезпечувалося системою контролю репозиторіїв GitKraken. Додаток дає змогу переглядати та зберігати інформаційні ресурси кафедри зручним для користувача способом.

3.1 Середовище розробки Visual Studio Code

Інтегроване середовище розробки — комплексне програмне рішення для розробки програмного забезпечення. Зазвичай, складається з редактора початкового коду, інструментів для автоматизації складання та відлагодження програм. Більшість сучасних середовищ розробки мають можливість автодоповнення коду.

Деякі середовища розробки містять компілятор, інтерпретатор або ж обидва, інші не містять жодного з них. Деякі інтегровані середовища розробки містять систему керування версіями або інструменти для полегшення розробки графічного інтерфейсу користувача. Багато сучасних IDE містять інспектор класів, інспектор

об'єктів, схему ієрархії класів для полегшення об'єктно-орієнтованої розробки програмного забезпечення.

Visual Studio Code - редактор вихідного коду, розроблений Microsoft для Windows, Linux і macOS. Позиціонується як «легкий» редактор коду для кросплатформеної розробки веб-і хмарних додатків. Включає в себе відладчик, інструменти для роботи з Git, підсвічування синтаксису, IntelliSense і засоби для рефакторинга. Має широкі можливості для кастомізації: призначені для користувача теми, поєднання клавіш і файли конфігурації. Розповсюджується безкоштовно, розробляється як програмне забезпечення з відкритим вихідним кодом.

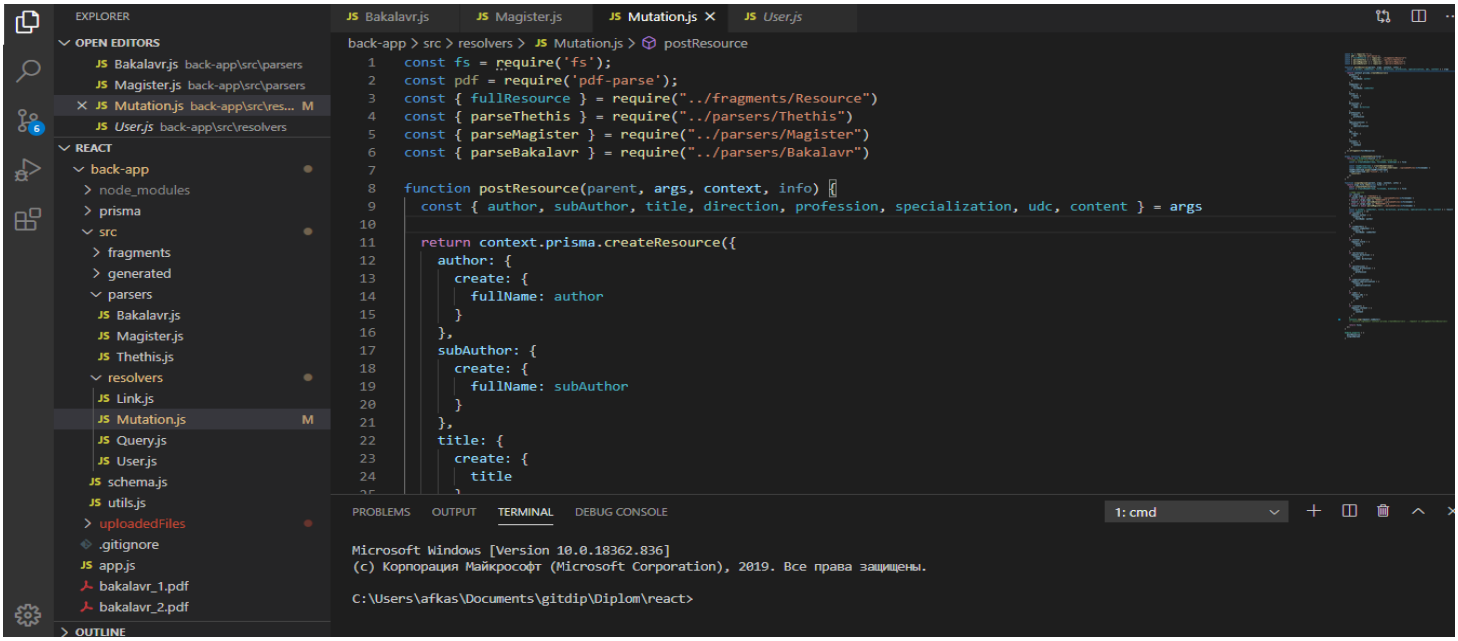

Рис. 3.1 – робоче вікно Visual Studio Code

Visual Studio Code заснований на Electron і реалізується через веб-редактор Monaco, розроблений для Visual Studio Online.

VS Code працює з будь-яким локальним або віддаленим сховищем Git і пропонує візуальне відображення для вирішення конфліктів до того, як код закомітиться.

– Налагодження: Однією з ключових особливостей Visual Studio Code є його велика підтримка налагодження. Вбудований налагоджувач VS Code допомагає

прискорити редагування, компіляцію та налагодження циклу. За замовчуванням він постачається з підтримкою NodeJS і може налагоджувати все, що транслюється в JavaScript, але для інших програм виконання, таких як C ++ або Python, потрібно буде встановити розширення.

– Деякі функції управління кодом: VS Code надає нам функції мовного обслуговування, такі як “Подивитися визначення”, “Перейти до визначення”, “Знайти всі посилання” та “Перейменувати символ”. Ці функції дуже корисні для кожного розробника. У VS Code ми можемо відформатовати код JavaScript, а також код інших мов. Ми можемо знайти ці функції, клацнувши правою кнопкою миші у файлі коду.

3.2 Модель БД MySQL Workbench

База даних – сукупність даних, організованих відповідно до концепції, яка описує характеристику цих даних і взаємозв'язки між їх елементами; ця сукупність підтримує щонайменше одну з областей застосування (за стандартом ISO/IEC 2382:2015). В загальному випадку база даних містить схеми, таблиці, подання, збережені процедури та інші об'єкти. Дані у базі організовують відповідно до моделі організації даних. Таким чином, сучасна база даних, крім саме даних, містить їх опис та може містити засоби для їх обробки.

В загальному випадку базою даних можна вважати будь-який впорядкований набір даних. Наприклад, паперову картотеку з формулярами про працівників підприємства у відділі кадрів. Але дана стаття зосереджена на використанні баз даних в інформаційних системах. На даний час застосунки для роботи з базами даних є одними з найпоширеніших прикладних програм.

У сучасних інформаційних системах для забезпечення роботи з базами даних використовують системи керування базами даних (СКБД). Система керування базами даних — це система, заснована на програмних та технічних засобах, яка забезпечує визначення, створення, маніпулювання, контроль, керування та використання баз даних (за стандартом ISO/IEC 2382:2015). Застосунки для роботи з базою даних

можуть бути частиною СКБД або автономними. Найпопулярнішими СКБД є MySQL, PostgreSQL, Microsoft SQL Server, Oracle, Sybase, Interbase, Firebird та IBM DB2. СКБД дозволяють ефективно працювати з базами даних, обсяг яких робить неможливим їх ручне опрацювання.

Сучасні СКБД забезпечують функції щодо керування даними, які можна поділити на такі групи:

Оголошення даних — створення, зміна та видалення визначень, які описують організацію даних.

Модифікація даних — додавання даних, їх редагування та видалення.

Отримання даних — надання даних за запитом застосунку у формі, яка дозволяє їх безпосереднє використання. Дані можуть надаватись або у формі, в якій вони зберігаються у базі даних, або в іншій формі (наприклад, через поєднання різних даних).

Адміністрування даних — реєстрування та відслідковування дій користувачів, дотримання безпеки роботи з даними, забезпечення надійності та цілісності даних, моніторинг продуктивності, резервне копіювання та відновлення даних тощо.

MySQL Workbench - інструмент для візуального проектування баз даних, що інтегрує проектування, моделювання, створення та експлуатацію БД в єдине безшовне оточення для системи баз даних MySQL. Є наступником DBDesigner 4 від FabForce.

- MySQL Workbench пропонується в двох редакціях:
- Community Edition - поширюється під вільною ліцензією GNU GPL;
- Standard Edition - доступна по щорічній оплачуваній підписці. Ця версія включає в себе додаткові функції, які підвищують продуктивність розробників і адміністраторів БД.
- Enterprise Edition

Можливості програми:

- Дозволяє наочно уявити модель бази даних в графічному вигляді.
- Наочний і функціональний механізм установки зв'язків між таблицями, в тому числі «багато до багатьох» зі створенням таблиці зв'язків.
- Reverse Engineering - відновлення структури таблиць з уже існуючою на сервері БД (зв'язку відновлюються в InnoDB, при використанні MyISAM - з цим необхідно встановлювати вручну).
- Зручний редактор SQL запитів, що дозволяє відразу ж відправляти їх із сервером і отримати відповідь у вигляді таблиці.
- Можливість редагування даних в таблиці у візуальному режимі.

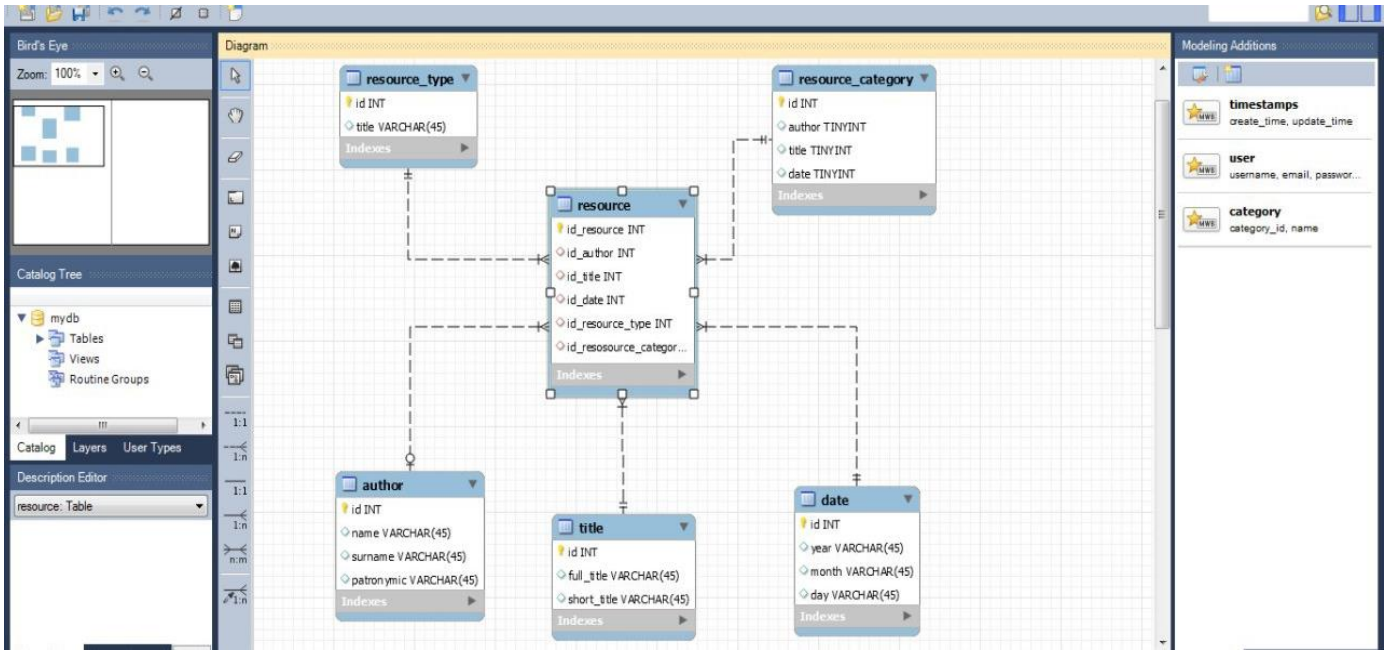

Рис. 3.2 – робоче вікно MySQL Workbench

3.3 Система керування реляційними базами даних MySQL

MySQL — вільна система керування реляційними базами даних.

MySQL, як і інші реляційні бази даних, зберігає дані в таблицях, стовпцях та рядках. Кожен запис визначається унікальним ідентифікатором. Основою MySQL завжди були продуктивність та надійність бази даних. MySQL був розроблений та

оптимізований для веб-розробок; це, мабуть, найпоширеніша база даних, яка використовується при розгортанні веб-серверів.

MySQL був створений шведською компанією під назвою MySQL AB в 1995 році. У перші роки розробки увага була зосереджена на швидкість та продуктивність, а не набори функцій. Однак нові функції згодом додавались із кожним головним випуском.

MySQL виявився успішним, оскільки він простий у використанні, простий в установці та використовує мову запиту, яку легко зрозуміти. (Рис. 4.1).

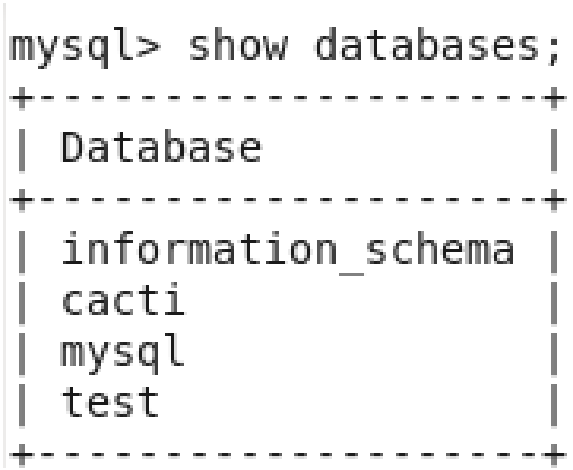

Рис. 4.1 – Приклад роботи MySQL

Sun Microsystems придбала MySQL AB у 2008 році; пізніше, у 2009 році, Oracle придбала компанію після придбання Sun Microsystems. Останні версії MySQL зараз дуже багаті на особливості продукту, зберігаючи при цьому високу продуктивність. Завдяки Oracle, MySQL нещодавно почали позиціонувати як платформу хмарних баз даних "Go-to Cloud" поряд із існуючими, більш традиційними платформами Oracle.

Існує кілька інструментів для управління та підтримки MySQL. У типових установках Linux управління БД здійснюється за допомогою командного рядка за допомогою клієнта MySQL. Клієнт MySQL може використовуватися для створення баз даних, налаштування дозволів та прав доступу. Крім того, існує кілька популярних інструментів графічного інтерфейсу, включаючи SQLyog, phpMyAdmin, браузери MySQL Query, Navicat, MySQL Administrator та MySQL Workbench.

Кожен інструмент виконує подібні завдання, включаючи створення нових таблиць, імпорт / експорт даних, управління користувачами та дозволами, резервне копіювання та відновлення, а також створення тригерів, переглядів та збереження процедур.

У MySQL є багато переваг:

–Зниження загальної вартості власності: MySQL - одна з найпопулярніших систем управління базами даних з відкритим кодом, яка дозволяє керувати реляційною базою даних. Оскільки MySQL з відкритим кодом, користувач може використовувати MySQL безкоштовно та має можливість налаштувати його вихідний код відповідно до потрібних вимог.

–Портативність: MySQL - це крос-платформенний сервер баз даних. Він може працювати на різних платформах, таких як Linux, Solaris та Windows. MySQL підтримує багато форм з різними мовами, такими як C, C ++, Node.js, PHP, PERL, JAVA, Python тощо.

–Швидкий розвиток і цілодобова робота: MySQL має гарантію тривалості роботи 24x7 і пропонує широкий спектр високодоступних рішень, включаючи спеціалізовані кластерні сервери. У MySQL є дуже велике співтовариство розробників, яке випускає регулярне оновлення.

–Безпека даних: MySQL визнається в усьому світі найбезпечнішою та надійною системою управління базами даних, яка використовується у популярних веб-додатках, включаючи WordPress, Drupal, Joomla, Facebook та Twitter. Дані, захищені паролем, який зберігається в зашифрованому вигляді.

3.4 Система керування версіями Git - GitKraken

GitKraken - багатоплатформовий візуальний клієнт системи управління версіями Git, який працює на Windows, Mac OS X і Linux.

Можливості:

- Інтеграція з GitHub, GitLab, BitBucket і OAuth авторизація

- Підтримка GitFlow
- Скасування і повторне виконання git операцій
- Розвинені інструменти для візуалізації та вирішення конфліктів
- Засоби нечіткого пошуку (Fuzzy finder)

Git - розподілена система керування версіями. Проект був створений Лінус Торвальдс для управління розробкою ядра Linux, перша версія випущена 7 квітня 2005 року. На сьогоднішній день його підтримує Джун Хама.

Серед проектів, які використовують Git - ядро Linux, Swift, Android, Drupal, Cairo, GNU Core Utilities, Mesa, Wine, Chromium, Compiz Fusion, FlightGear, jQuery, PHP, NASM, MediaWiki, DokuWiki, Qt, ряд дистрибутивів Linux.

Програма є вільною і випущена під ліцензією GNU GPL версії 2. Стандартною TCP порт 9418.

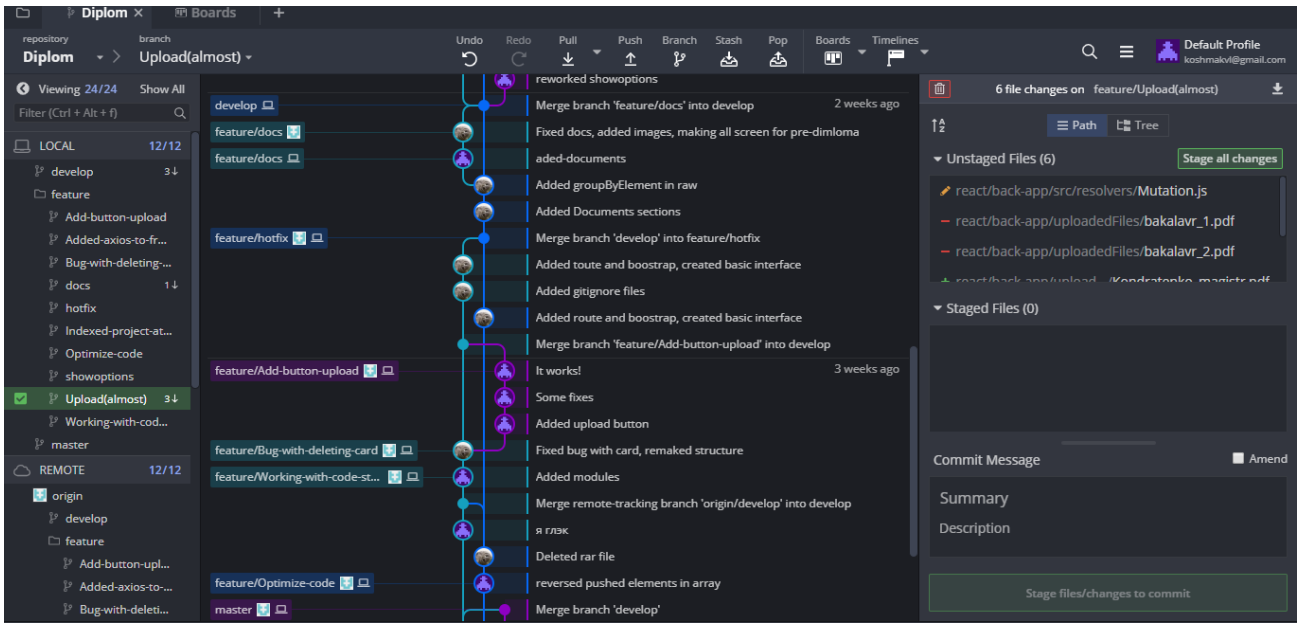

Рис. 3.4 – вікно керування репозиторіями GitKraken

3.5 Програмна платформа Node.js

Node або Node.js - програмна платформа, заснована на двигуні V8 (здійснює трансляцію JavaScript в машинний код), що перетворює JavaScript з вузькоспеціалізованого мови в мову загального призначення. Node.js додає можливість JavaScript взаємодіяти з пристроями введення-виведення через свій API (написаний на C ++), підключати інші зовнішні бібліотеки, написані на різних мовах, забезпечуючи виклики до них з JavaScript-коду. Node.js застосовується переважно на сервері, виконуючи роль веб-сервера, але є можливість розробляти на Node.js і десктопні віконні додатки (за допомогою NW.js, AppJS або Electron для Linux, Windows і macOS) і навіть програмувати мікроконтролери (наприклад, tessel, low.js і espruino). В основі Node.js лежить подієво-орієнтоване і асинхронне (або реактивне) програмування з неблокуючим введенням / висновком.

Node.js порівняно молодий проект: розробники дізналися про вихід ідейно нової технології розробки веб-ресурсів в 2009 році. З тих пір платформа розрослася і використовується багатьма програмістами по всьому світу. Чим визначена популярність Node.js?

Перше і саме корисна властивість платформи, особливо для новачків - простота і доступність. Для реалізації власних проектів вам не доведеться вручну по крупицях збирати бібліотеки, пакети, інформацію про даний продукт. Установка займає лічені хвилини, після яких Node вже готовий до роботи. Завантажити базовий комплект для установки можна на офіційному сайті платформи http://nodejs.org/ або в середовищі linux http://nodejs.org/en/download/package-manager/

Структурно Node.js є об'єднанням движка (V8), засобів введення-виведення (I / O) і набору бібліотек. При цьому розробка компонентів програми або сайту ведеться на одній мові програмування - JavaScript! Вам не буде потрібно вивчення додаткових ресурсів або залучення фахівця іншої мови, досить мати гарні навички програмування на JS, вивчення якого, в свою чергу, не важко, якщо ви будете уважно підходити до вивчення цього гнучкого мови. Як підсумок - сервер і клієнт написані однією мовою.

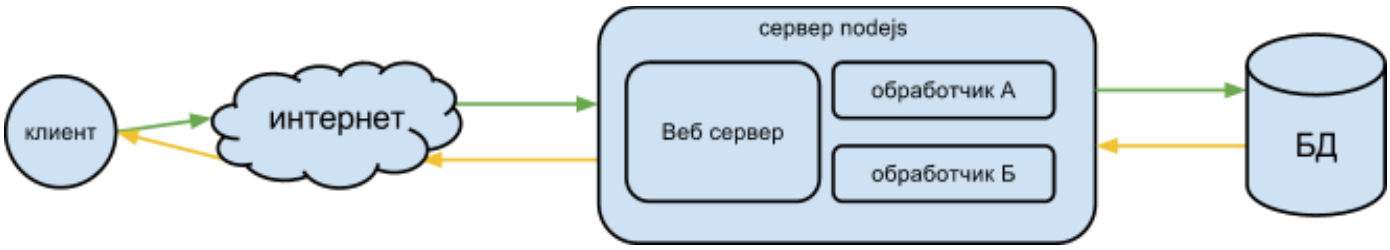

Рис. 3.5 – умовна схема взаємодії клієнта і сервера завдяки Node.js

Окремо відзначимо, що деякі виносять як перевага Node єдиний код на клієнті і на сервері. Теоретично ніхто цього не виключає, у багатьох проектах так і є, але для загальних бібліотек і загальних частин. У більшості випадків на практиці код на клієнті і сервері різниться. Однак це аж ніяк не ускладнює роботу з самої платформою. Також Node сповідує подієво-орієнтовану парадигму програмування відповідно до якої події визначають виконання програми, побудова додатки відбувається за допомогою програмування обробників безлічі подію, які можуть статися в процесі виконання.

Ще одна перевага даної платформи - розроблена для всіх типів ОС. Установка Node.js однаково проста і на Mac, і на Windows, і на Unix системах. Тим самим зміцнюється доступність і стабільність даного продукту.

Заглибимося в суть платформи. Раніше був згаданий двигун V8. На даний момент це одна з найбільш продуктивних програм для JS. Завдяки їй код виконується в рази швидше. V8 ефективно управляє пам'яттю:

- Переривання для збору сміття
- Мінімізація впливу цих переривань на час виконання коду
- Оптимальне зберігання покажчиків і об'єктів, як наслідок - ліквідація проблеми втрати пам'яті
- Швидкий доступ до інформації

Тандем з V8 і набору бібліотек Node.js реалізує асинхронне API по роботі з мережевими ресурсами. Тепер сервер не затримує процес в очікуванні відповіді для запиту певного процесу, наприклад, від бази даних, а продовжує роботу з іншими

запитами, поки даний знаходиться в процесі виконання. Це досить великий крок для інтерактивних систем, де важлива швидкість відповіді на певні дії на клієнта.

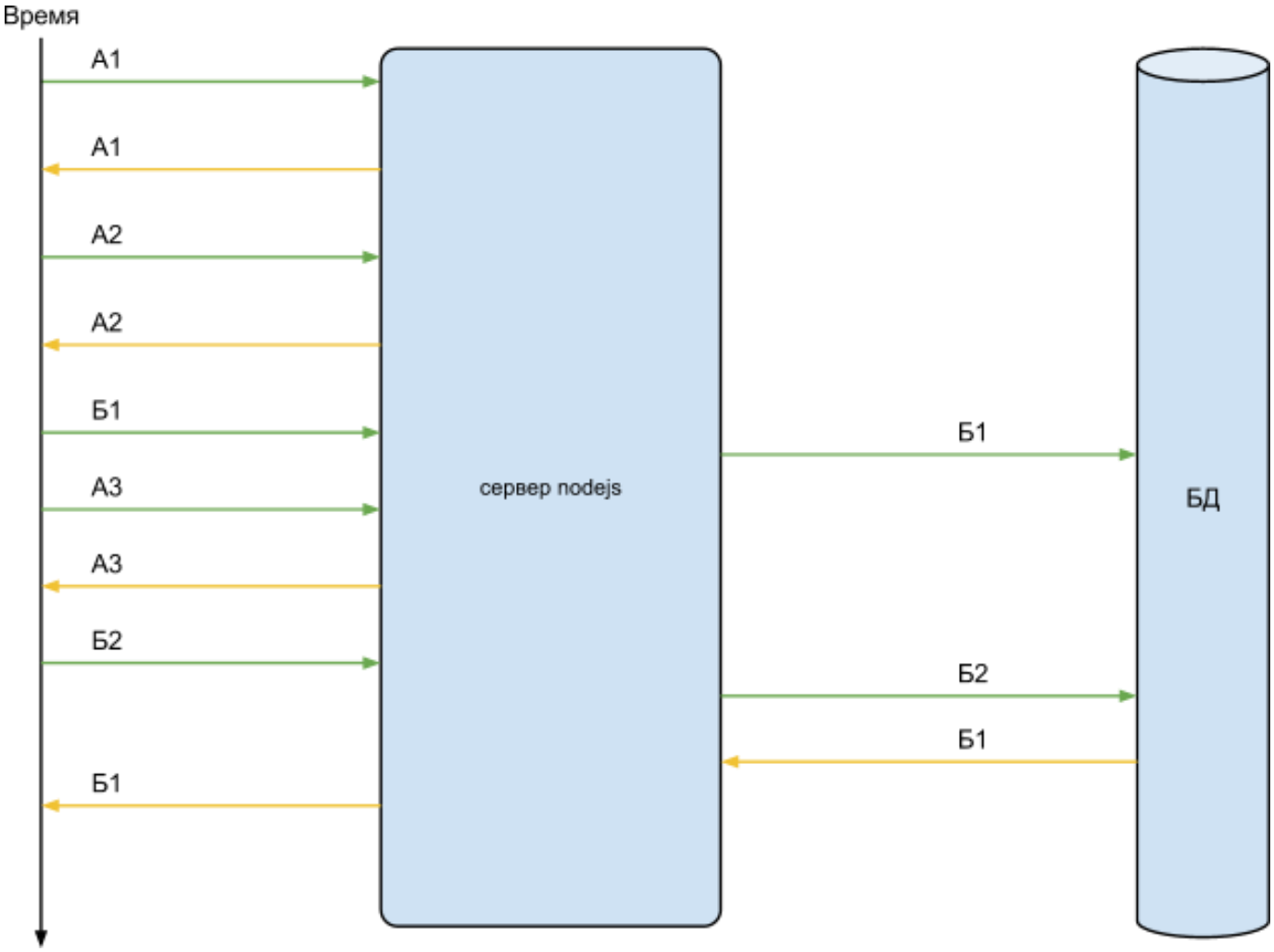

Рис. 3.6 – умовна схема запиту та відповіді між сервером та БД завдяки Node.js

Підсумок:

- легко працювати
- прототип
- Інтерактивна розробка
- Легко вибудовувати архітектуру (готові пакети)

3.6 Бібліотека React.js

React — відкрита JavaScript бібліотека для створення інтерфейсів користувача, яка покликана вирішувати проблеми часткового оновлення вмісту веб-сторінки, з якими стикаються в розробці односторінкових застосунків. Розробляється Facebook, Instagram і спільнотою індивідуальних розробників.

```
import React from 'react';

export default class Block extends React.Component {
    constructor(props) {
        super(props);
        this.state = {
            edit: false,
            text: this.props.text,
            show: true
        };
    };

    edit = () => {
        this.setState({ edit: true });
    };

    save = () => {
        this.setState({ edit: false, text: this.refs.txt.value })
    };

    remove = () => {
        this.setState({ show: false })
    };

    upload = () => {
        var filename = 'Uploaded-file';
        var element = document.createElement('a');
      element.setAttribute('href', 'data:text/plain;charset=utf-8,' + encodeURIComponent(this.state.text));
```

Рис. 3.7 – приклад частини коду на React.js

React дозволяє розробникам створювати великі веб-застосунки, які використовують дані, котрі змінюються з часом, без перезавантаження сторінки. Його мета полягає в тому, щоб бути швидким, простим, масштабованим. React обробляє тільки користувацький інтерфейс у застосунках. Це відповідає видові у шаблоні модель-вид-контролер (MVC), і може бути використане у поєднанні з іншими JavaScript бібліотеками або в великих фреймворках MVC, таких як AngularJS. Він також може бути використаний з React на основі надбудов, щоб піклуватися про частини без користувацького інтерфейсу побудови веб-застосунків. Як бібліотеку

інтерфейсу користувача React найчастіше використовують разом з іншими бібліотеками, такими як Redux.

Властивості передаються в рендерер компоненту, як властивості html тегу. Компонент не може напряму змінювати властивості, що йому передані, але може їх змінювати через callback функції. Такий механізм називають «властивості донизу, події нагору».

React підтримує віртуальний DOM, а не покладається виключно на DOM браузера. Це дозволяє бібліотеці визначити, які частини DOM змінилися, порівняно зі збереженою версією віртуального DOM, і таким чином визначити, як найефективніше оновити DOM браузера. Таким чином програміст працює зі сторінкою, вважаючи що вона оновлюється вся, але бібліотека самостійно вирішує які компоненти сторінки треба оновити.

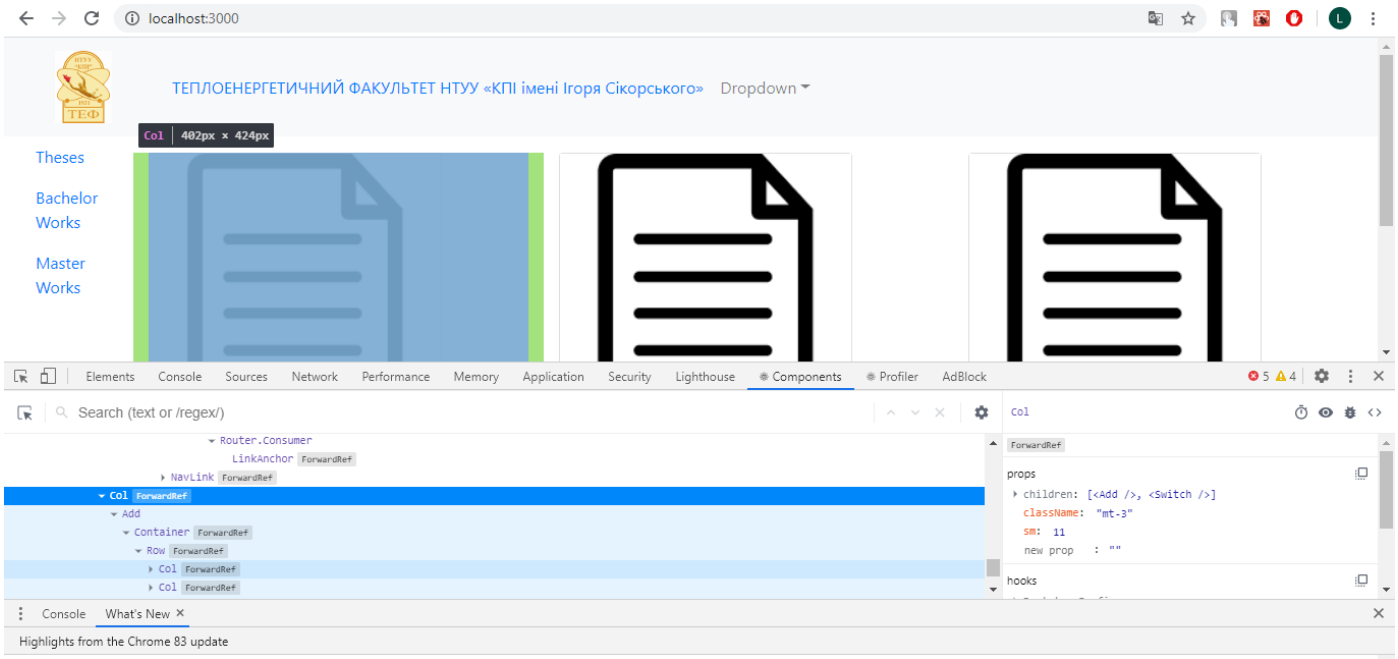

Рис. 3.8 – компоненти віртуального DOM що показані завдяки React Developer Tools

Компоненти React зазвичай написані на JSX. Код написаний на JSX компілюється у виклики методів бібліотеки React. Розробники можуть так само писати на чистому

JavaScript. JSX нагадує іншу мову, яку створили у компанії Фейсбук для розширення PHP, XHP.

React використовують не лише для рендерингу HTML в браузері. Наприклад, Facebook має динамічні графіки які рендеряться в теги <canvas>, Netflix та PayPal використовують ізоморфне завантаження для рендерингу ідентичного HTML на сервері та клієнті.

3.7 Chrome DevTools

Інструменти для розробників Google Chrome, також відомі як Chrome DevTools, - це інструменти для веб-розробки та налагодження, вбудовані прямо в браузер. Вони надають розробникам більш глибокий доступ до своїх веб-додатків та браузера: від тестування вікна та перегляду на мобільному пристрої до редагування веб-сайту на ходу та навіть вимірювання продуктивності всього веб-сайту чи окремих активів. Chrome DevTools має багато панелей. Нижче представлено опис функціональності більшості з них.

- Панель Elements

Використовується для вибору та редагування будь-яких HTML-елементів на сторінках. Дозволяє вільно маніпулювати DOM та CSS(Рис. 3.9). При виборі будь-якого DOM елементу на вкладці Styles буде відображатися всі CSS стилі, які застосовуються до нього, в тому числі і неактивні. Можна змінювати значення, деактивувати і дописувати нові стилі і дивитися як це впливає на відображення.

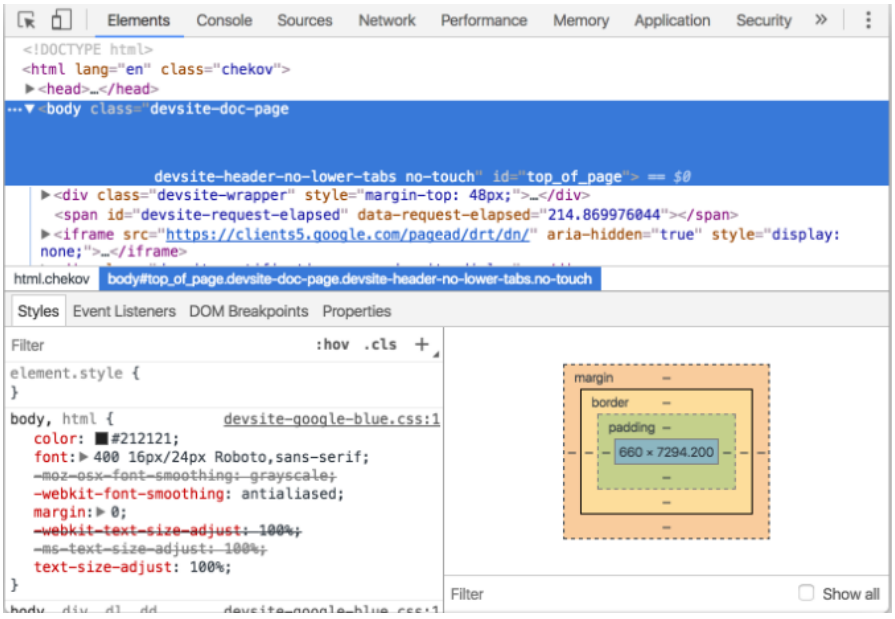

Рис 3.9 –Elements

- Console

Панель необхідна для логування діагностичної інформації в процесі розробки або взаємодії з JavaScript на сторінці. Також всі помилки в JavaScript коді, будуть виводиться тут із зазначенням файлу і конкретного місця в ньому де сталася помилка. Так само в консоль можна виводити XHR запити. Є можливість зберігати логи в окремий файл (Рис. 3.10).

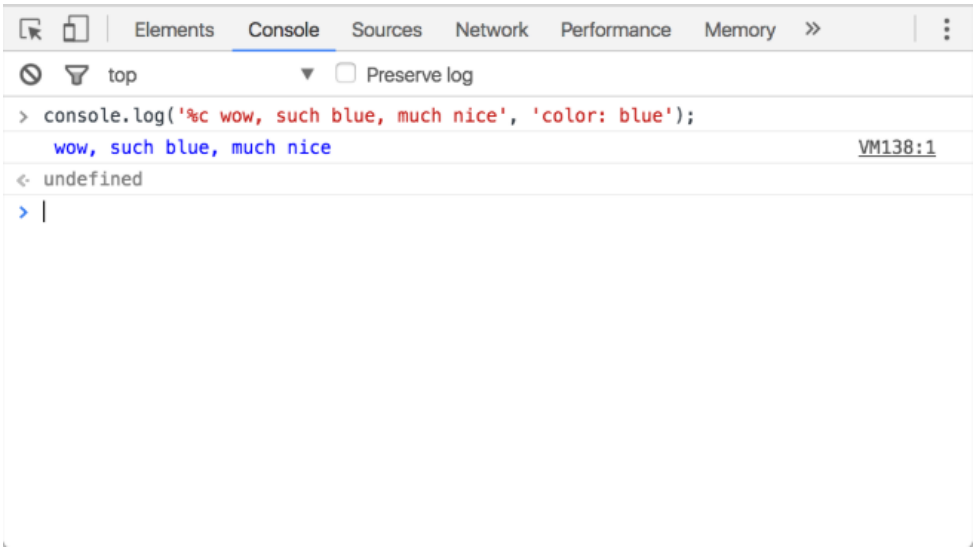

Рис. 3.10 –Console

- Sources

Інструмент Sources (Рис. 3.11) є середовищем, де ми можемо подивитися всі файли підключені на нашій сторінці. Ми можемо подивитися їх вміст, відредагувати код, скопіювати його або зберегти змінений файл, як новий файл. Дану вкладку можна використовувати і як повноцінний редактор коду.

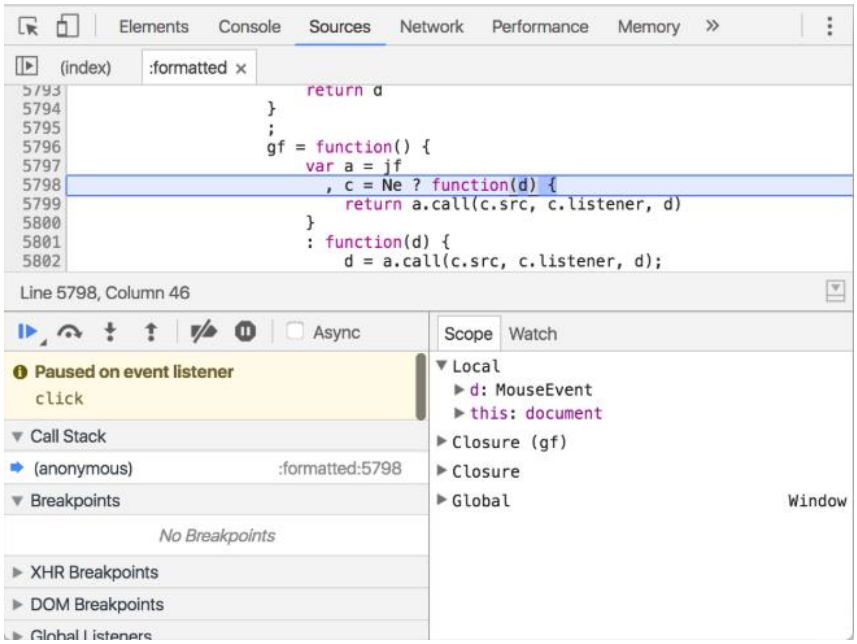

Рис. 3.11 –Sources

- Network

Ця вкладка дозволяє переглядати процес завантаження сторінки і всіх файлів які завантажуються. На панелі (Рис. 3.12) відображається таблиця всіх запитів до даних і файлів, над нею розташовуються кнопки для фільтрації потрібних користувачу запитів, очищення таблиці або включення / відключення запису запитів, кнопки управління відображенням таблиці.

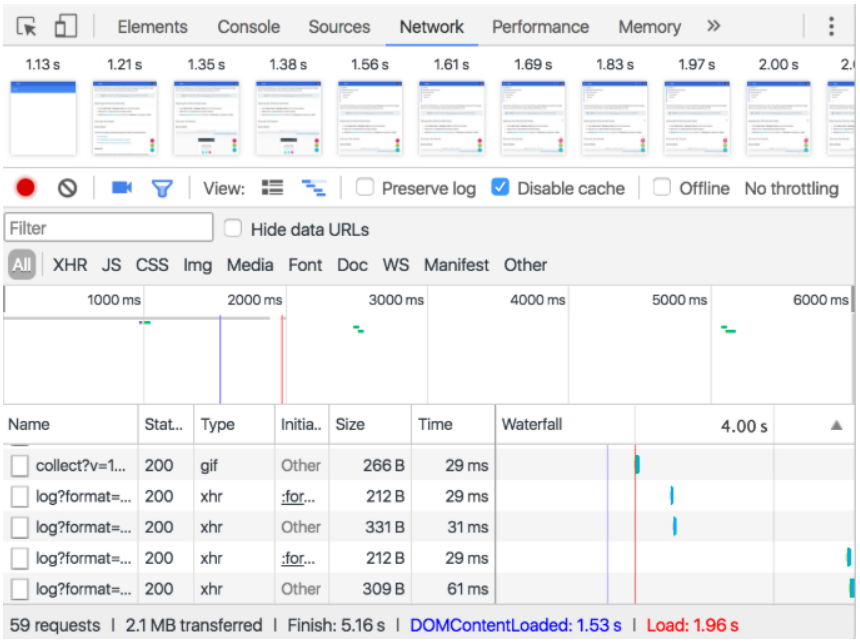

Рис. 3.12 –Network

- Performance

Панель (Рис. 3.13) відображає таймлайн використання мережі, виконання JavaScript коду і завантаження пам'яті. Після початкової побудови графіків таймлайн, будуть доступні дані про виконання коду. У вкладці є можливість ознайомлення з часом виконання окремих частин коду, вибору окремого проміжоку на часовій шкалі та ознайомлення з тим, які процеси відбувалися в цей момент. На вкладці Performance можливо:

- Зробити запис, щоб проаналізувати кожну подію, яка сталася після завантаження сторінки або взаємодії з користувачем.
- Переглянути FPS, завантаження CPU і мережеві запити в області Overview.
- Натиснути по події в діаграмі, щоб подивитися деталі про неї.
- Змінити масштаб таймлайну, щоб зробити аналіз більш ефективним.

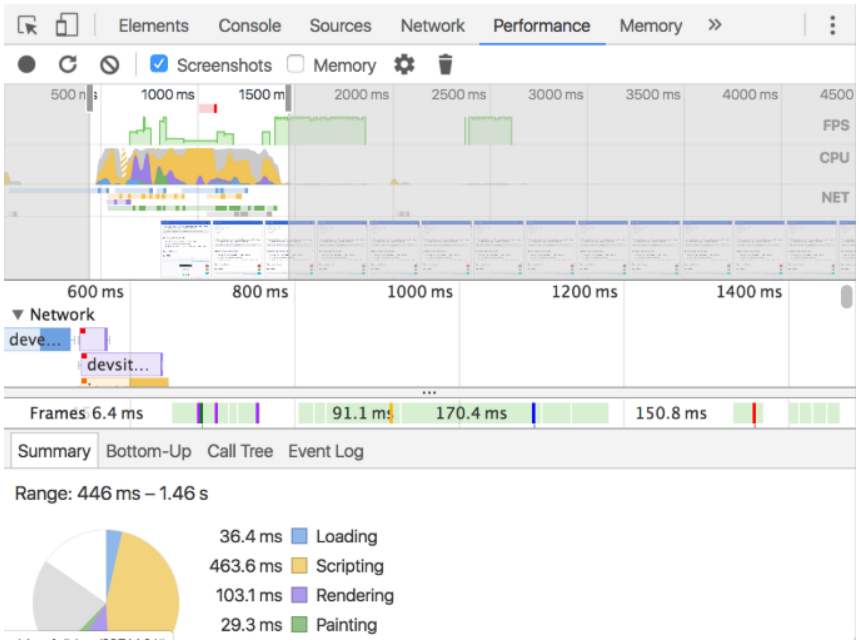

Рис. 3.13 –Performance

- Memory

Ця панель потрібна для виправлення проблем з пам'яттю та профілювання CPU при роботі з JavaScript, що дозволяє дуже зручно розрахувати навантаження клієнтського додатку на машину користувача та в разі чого оптимізувати чи прибрати проблеми, які перевантажують систему (Рис. 3.14).

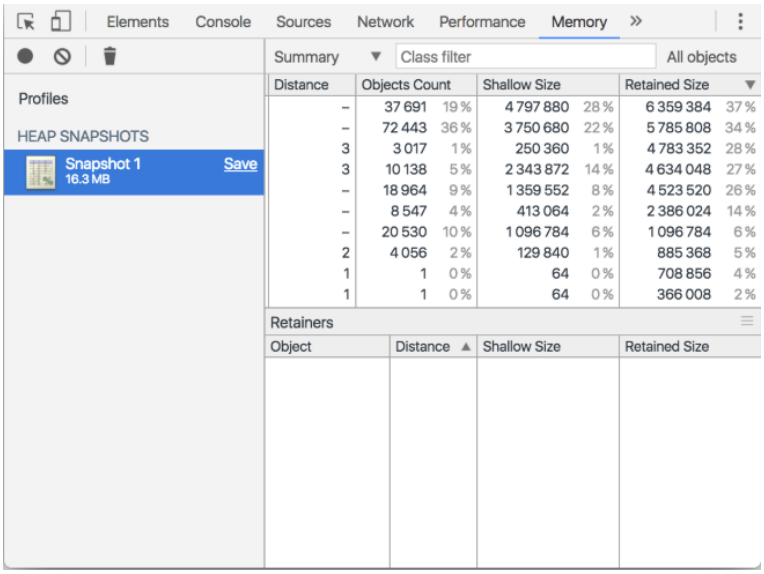

Рис. 3.14 –Memory

- Application

Вкладка призначена для інспектування та очищення всіх завантажених ресурсів, включаючи IndexedDB або Web SQL бази даних, куків, кеша додатків, зображень, шрифтів і таблиць стилів (Рис. 3.15). Можливості вкладки:

- Швидке очищення сховищ і кеша.
- Інспектування і управління сховищами, базами даних і кешем.
- Інспектування і видалення файлів cookie.

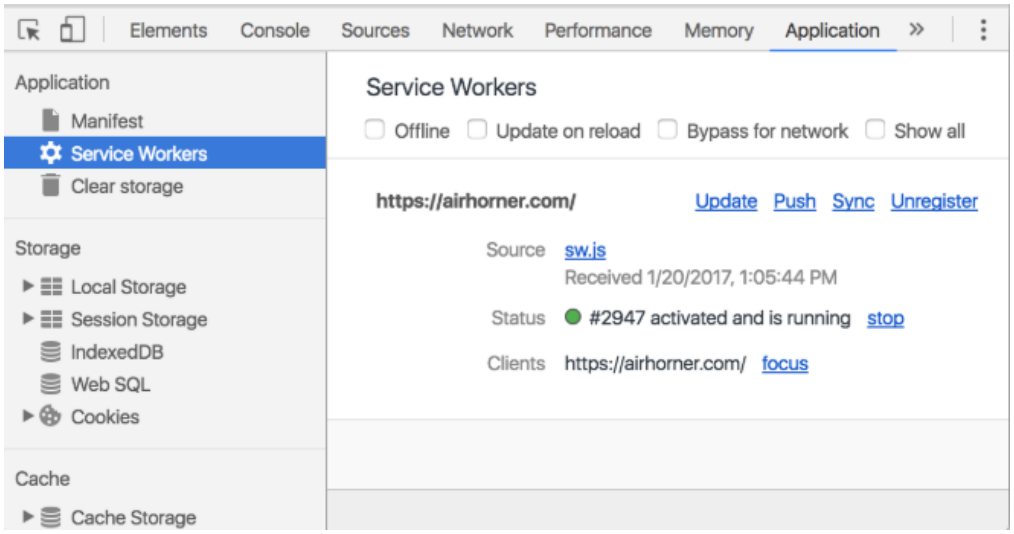

Рис. 3.15 –Application

- Security

На вкладці можна ознайомиться з протоколом безпеки при його наявності і переглянути дані про сертифікат безпеки, якщо він є. Цей інструмент використовується для налагодження проблем змішаного контенту, проблем сертифікатів та інше (Рис. 3.16).

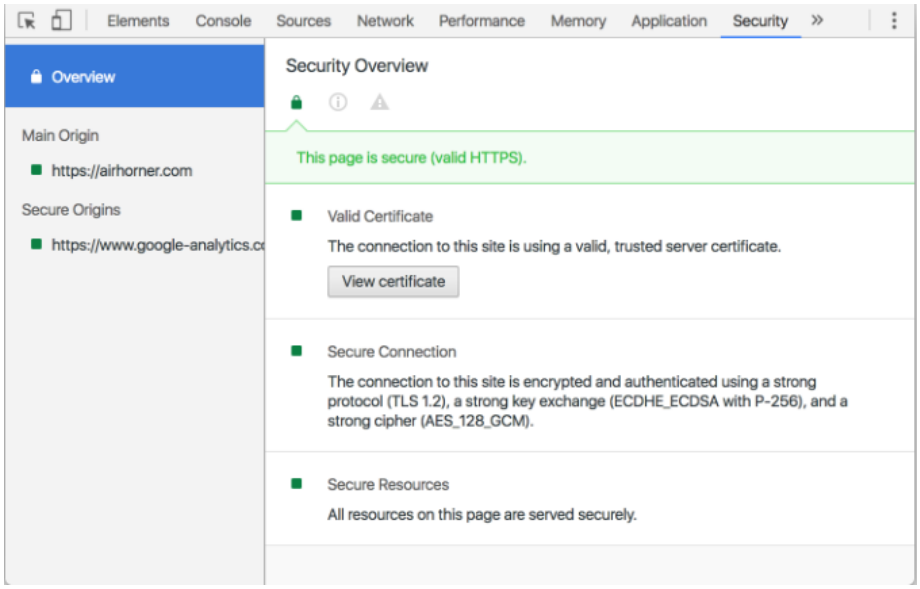

Рис. 3.16 –Security

3.8 GraphQL

GraphQL — це мова запитів і маніпуляції даними з відкритим кодом для API і середовище виконання для обслуговування запитів з наявних даних.

GraphQL надає підхід розробки веб API і його можна порівнювати і протиставляти REST та іншим архітектурам веб-сервісів. Він дозволяє клієнтам визначати структуру потрібних даних і таку саму структуру повертає сервер, таким чином запобігаючи передачі надлишкових даних, але це впливає на дієвість веб-кешування результатів запитів. Гнучкість і багатість мови запитів, що може бути не потрібна для простих API. Він складається з системи типів, мови запитів і семантики виконання, статичної валідації і інтроспекції.

GraphQL підтримує читання, писання (змінювання) і підписування на зміни даних (оновлення в реальному часі - зазвичай втілені за допомогою Webhook).

Основні поняття:

- Схема - визначає систему типів. Вона описує множину можливих даних (об'єктів, полів, зв'язків та ін.) до чого клієнт може доступитись. Виклики від клієнта валідуються і виконуються згідно зі схемою. Клієнт може знайти інформацію про схему через інтроспекцію. Схему зберігають на GraphQL API сервері.
- Поле - це одиниця даних, що її можна отримати з об'єкта. З офіційної документації Мова запитів GraphQL це, по суті, вибирання полів з об'єктів.
- Аргумент - це множина пар ключ-значення припасованих до певного поля. Кожне поле на типі об'єкта може мати нуль або більше аргументів.

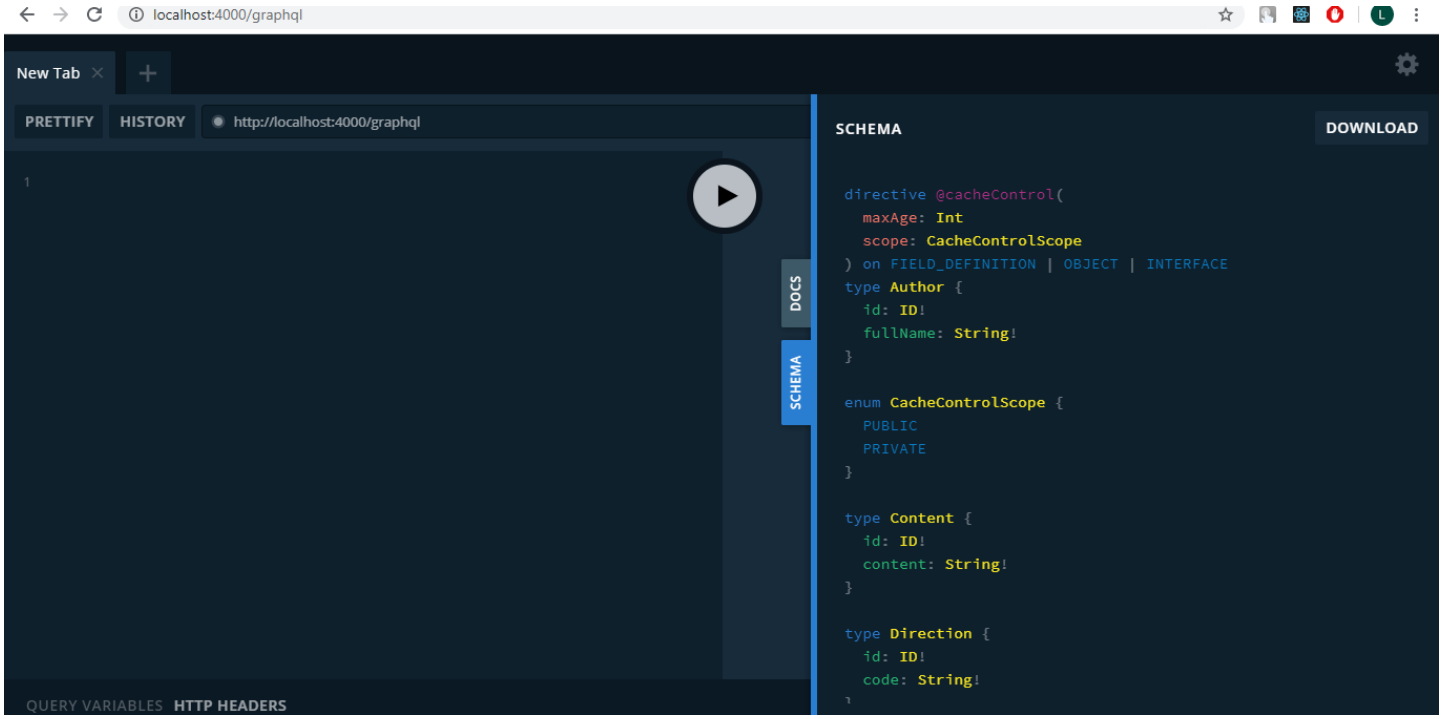

Рис. 3.18 – схема GraphQL

4. ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

Для створення системи було використано два середовища: MySQL Worckbench та Visual Studio Code.

Основна частина дипломної роботи – заповнення реєстру інформаційних ресурсів кафедри. Ця частина розроблялась в середовищі MySQL Worckbench. Оскільки SQL є надійною та популярною мовою запитів до баз даних а це середовище чудово підходить для написання БД цією мовою. База даних зберігається віддалено завдяки сервісам Prisma що забезпечує надійність системи. База даних зв'язується з клієнтським додатком завдяки мові GraphQL. GraphQL виступає посередником між сервером та базою даних, бо він забезпечує швидкодійність системи, є зручним у використанні та кросплатформеним.

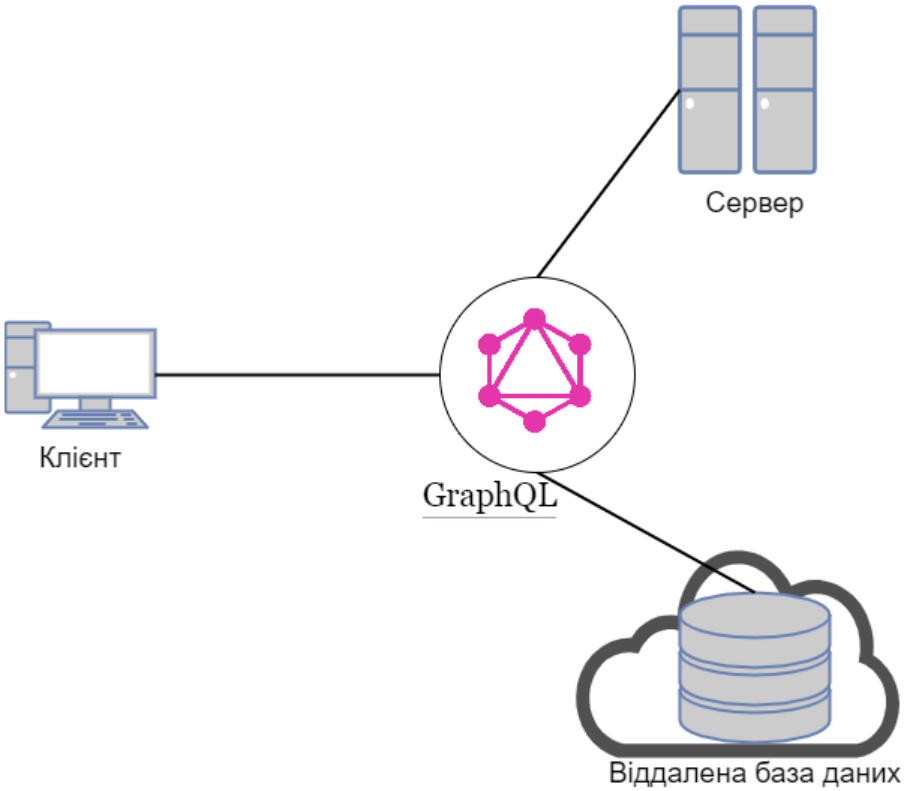

Рис. 4.1 – схема GraphQL

Клієнтський додаток написаний мовою JavaScript на платформі Node.js у середовищі Visual Studio Code з використанням бібліотеки React.js.

В якості графічного інтерфейсу може використовуватися будь-який браузер. Це є дуже зручним крос-платформеним рішенням, оскільки сучасні браузери працюють майже однаково з html-розміткою та каскадними стилями CSS.

На сьогодні існує безліч систем для зберігання даних. Вони можуть працювати не тільки локально ай віддалено. Моя програма є однією з таких програм, що зберігають інформацію, але призначена вона перш за все для кафедри та зберігає лише інформаційні ресурси. Основною перевагою перед іншими подібними системами є її вузьконаправленість. Вона створена спеціально для зберігання документів які частіше всього зустрічаються на кафедрі. Також вона включає в себе такі важливі характеристики як зручність, надійність та швидкодійність.

Дана система призначена перш за все для викладачів чи відповідальних осіб які відповідають за інформаційні ресурси кафедри. Вона допомагає внести та впорядкувати і структурувати документи, що призначені для використання на кафедрі.

Отже ця система має лише одного актора і це по суті є адміністратор, оскільки актор здатен не лише переглядати результат ай заносити до регістру інфомаційні ресурси.

Доступні функції системи:

- Внесення інформаційних ресурсів
- Пошук інформаційних ресурсів
- Перегляд інформаційних ресурсів

Для того щоб користуватися системою не потрібно якихось особливих знань від користувача, оскільки інтерфейс інтуітивно зрозумілий.

4.1 Створення бази даних

База даних, як було сказано вище, створена, як і більшість, на реляційній системі MySQL. Обрана вона була через те, що це найпопулярніша та найрозвиненіша система для побудови баз даних. Ця система, як і всі реляційні системи, основується на тому що дані зберігаються в ролях таблиць. При використанні таких систем важливо не лише створити правильні таблиці ай розставити між ними зв'язки.

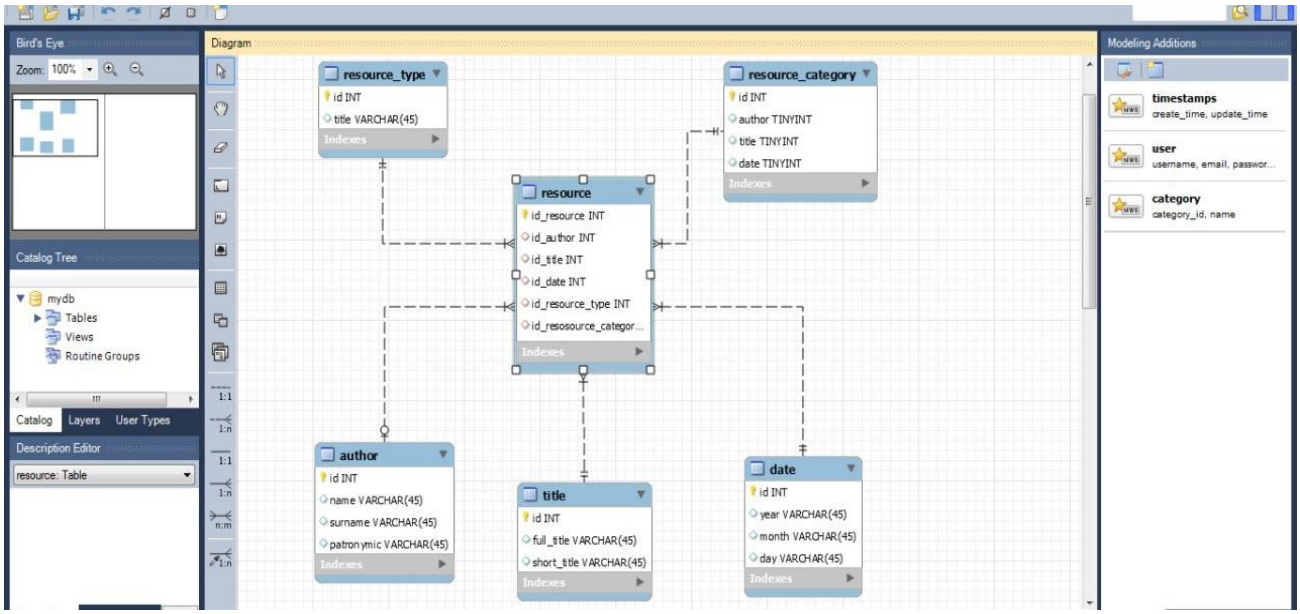

Рис. 4.2 – представлення реляційної БД в MySQL

База даних зберігається на віддаленому сервісі Prisma. Це рівень даних, який замінює традиційні інструменти об'єктно-реляційного відображення (ORM) в вашому додатку. Пропонуючи підтримку як для побудови серверів GraphQL, так і для API REST, Prisma спрощує доступ до бази даних з акцентом на type safety і забезпечує декларативну міграцію бази даних. Безпека типів допомагає зменшити потенційні помилки і невідповідності коду, а декларативні міграції баз даних дозволяють зберігати вашу модель даних в системі управління версіями. Ці функції допомагають розробникам скоротити час, що витрачається на налаштування доступу до баз даних, міграцій та робочих процесів управління даними.

Ви можете декількома способами розгорнути сервер Prisma, який виступає в якості проксі-сервера для вашої бази даних, і розмістити його віддалено або локально. Через сервіс Prisma ви можете отримати доступ до своїх даних і підключитися до своєї бази даних за допомогою API-інтерфейсу GraphQL, який дозволяє виконувати операції в реальному часі і створювати, оновлювати і видаляти дані. GraphQL - це мова запитів для API, який дозволяє користувачам відправляти запити для доступу до точних даними, які їм потрібні на їх сервері. Сервер Prisma - це автономний компонент, який розташовується поверх вашої бази даних.

4.2 Створення клієнта і сервера

Клієнт моєї системи (та частина з якою користувач взаємодіє більше всього) написана на мові JavaScript за допомогою бібліотеки React.js на платформі Node.js.

JavaScript -це основна та загально прийнята мова для написання будь-яких веб додатків, тому саме ця мова була обрана для реалізації мого проекту.

Node.js - платформа на основі JavaScript, що розширяє цю мову і надає їй нові можливості як асинхронність, багатопоточність, доступ до файлів користувача. В моїй системі дана платформа потрібна саме для того щоб розширити можливості JS до перелічених вище та імітувати роботу сервера, що надає змогу для використання сторонніх бібліотек завдяки пакетному менеджеру NPM та можливість легко працювати з базами даних.

React.js – бібліотека для мови JavaScript, яка використовується для написання клієнтських інтерфейсів та вирішує проблему часткового оновлення контенту на сторінках браузерів. Я використав саме цю бібліотеку завдяки її швидкодійності та зручності, оскільки в проекті часто змінюється контент що знаходиться на сторінці (як на Рис. 4.3-4.5), тому потрібно було вирішити проблему оновлення цією бібліотекою.

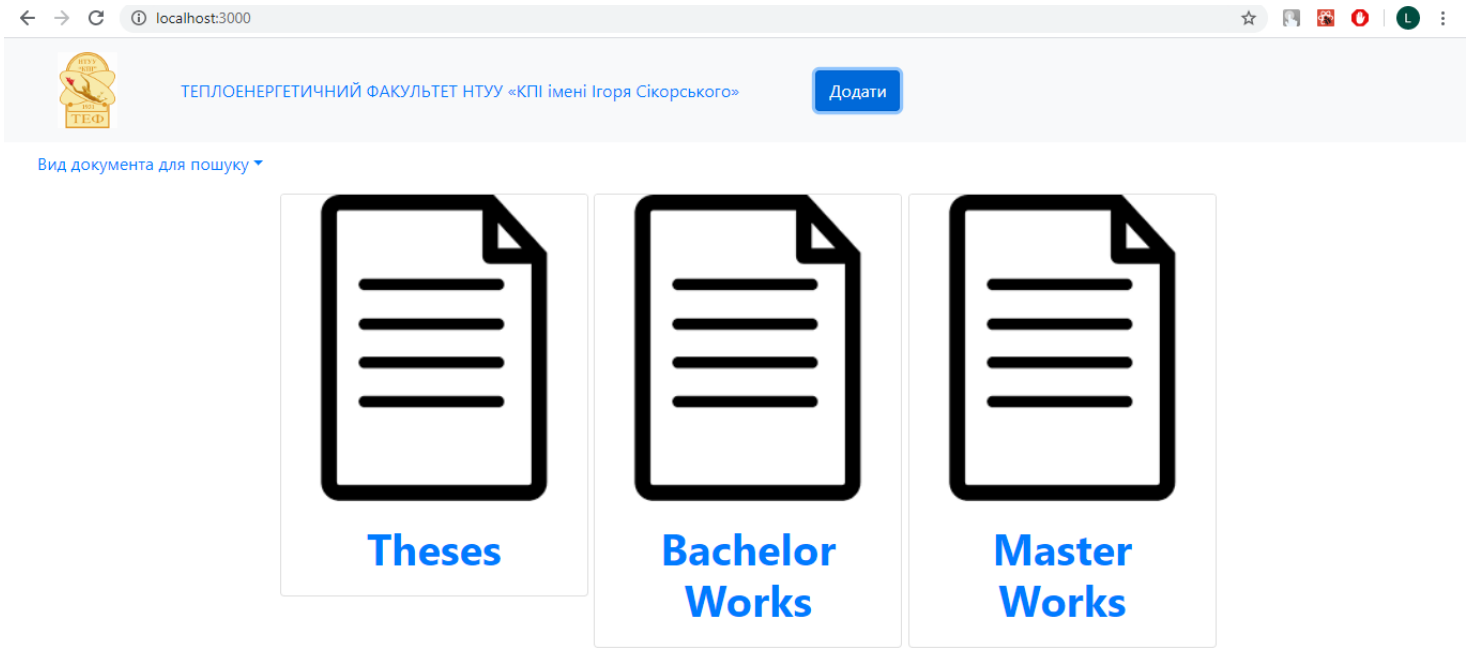

Рис. 4.3 – стартова сторінка додавання

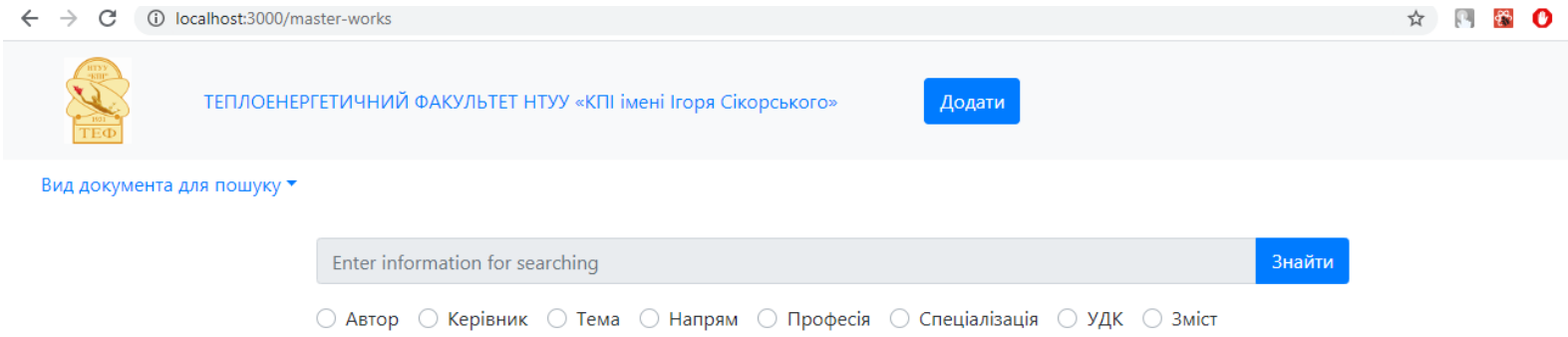

Рис. 4.4 – елемент пошуку

4.3 Взаємодія клієнта з базою даних

Отже тепер питання стоїть лише в тому як пов’язати всі компоненти. Клієнтський інтерфейс доволі зручно взаємодіє з сервером оскільки вони написані на однаковій мові. Потрібно щоб сервер давав правильні запити на базу даних та правильні відповіді до клієнтського інтерфейсу, цю проблему легко вирішує GraphQL.

GraphQL - мова запитів з відкритим вихідним кодом, розроблений Facebook. Він створювався як більш ефективна альтернатива REST для розробки і використання програмних інтерфейсів додатків.

GraphQL має безліч переваг, наприклад:

- Ви отримуєте інформацію саме в тому обсязі, в якому запитуєте. На відміну від REST, відповідь на запит не буде містити непотрібних даних.
- Вам буде необхідна всього одна кінцева точка, ніяких додаткових версій для єдиного API.
- GraphQL - сильно типізована мова, що дозволяє попередньо оцінити коректність запиту в рамках системи типів цього синтаксису, до виконання. Це дозволяє розробляти більш потужні API.

Так як GraphQL розроблявся на платформі Node.js (як і React.js) то взаємодія між всіма елементами системи дуже зручна і надійна. Це дозволяє описати всі потрібні типи даних для запиту і відповіді завдяки поняттю аргумента (ключ-значення) в GraphQL та швидко вивести значення.

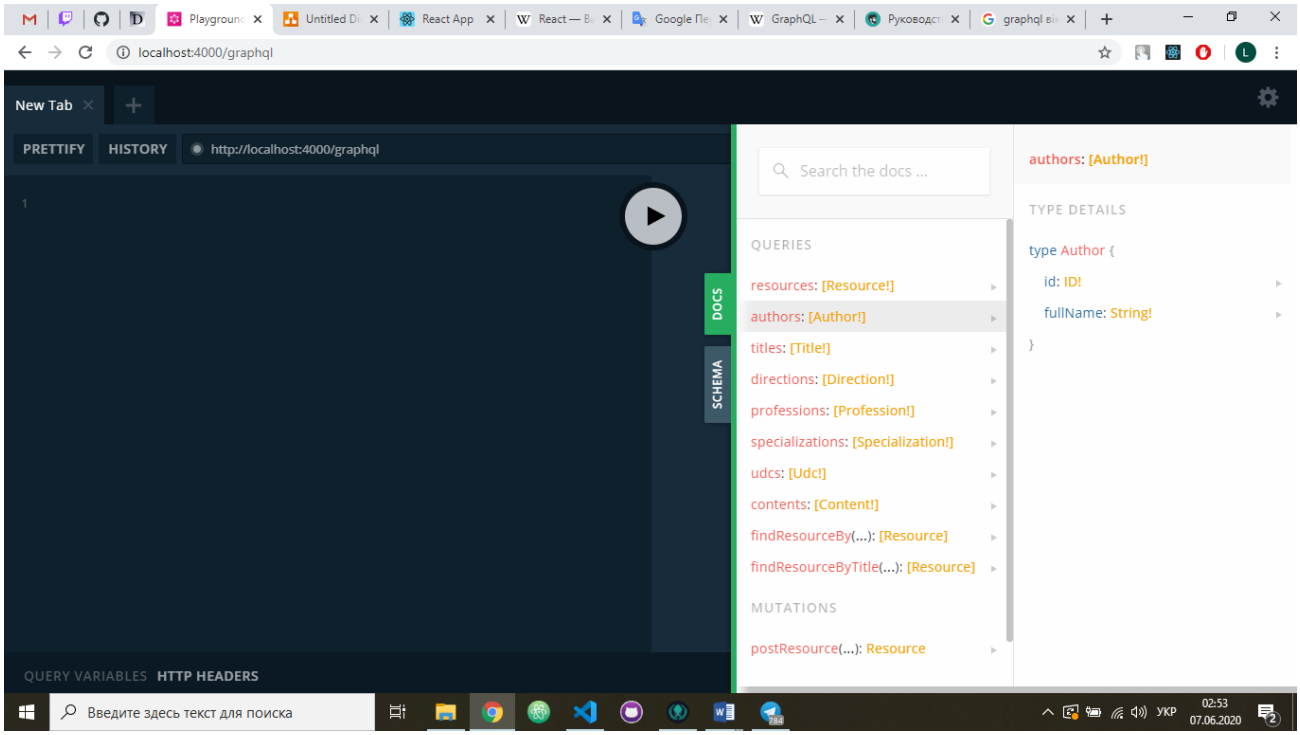

Рис. 4.5 – GraphQL дозволяє наглядно продемонструвати аргументи

Коли документ вноситься до БД відбувається розбір документа (парсинг) на сервері написаний на мові JavaScript. З документа виносяться головні поняття, які відрізняють його з поміж інших документів та котрі дозволять в подальшому не лише коректно відображати документи на сторінці, ай шукати по цим пунктам в системі потрібний клієнту документ. Оброблені значення передаються до бази даних як аргументи GraphQL.

5. РОБОТА КОРИСТУВАЧА З ПРОГРАМНОЮ СИСТЕМОЮ

Метою дипломної роботи було створення системи заповнення до галузевому реєстру інформаційних ресурсів кафедри було вирішення проблеми запису електронного документу до реєстру та його миттєве відображення. Для цього було розроблено систему яка працює як веб додаток. Для забезпечення безвідмовної роботи системи треба дотримуватися основних вимог при інсталяції та рекомендацій щодо її використання та слідкувати за системними вимогами.

5.1 Системні вимоги

Для продуктивного використання системи пошуку по галузевому реєстру інформаційних ресурсів кафедри потрібні такі системні характеристики:

- 64-розрядна архітектура.
- Ядро версії 3.10 або вище.
- Windows 7/10 чи Ubuntu 14.04 / 15.10
- 2 або більше процесорів / ядер.
- Принаймні 2 ГБ оперативної пам’яті.
- Принаймні 25 Гб місця на диску.
- Відкриті порти для вхідного трафіку TCP.
- Доступ до Інтернету безпосередньо чи через проксі.

Дані вимоги задовольняють більшість комп’ютерів що є в наявності на кафедрі тому дана програма чудово підходить для використання в учбовому закладі.

5.2 Робота користувача з програмним продуктом

Щоб користуватися даним продуктом необхідно перейти у браузері до ресурсу на якому буде розташована користувацька частина програми або відкрити директорію

в якій розташований програмний код та в консолі запустити виконання команди “npm start” та зачекати відкриття ресурсу у браузері за замовчуванням.

Виконання даної команди є процесом прослуховування порту під номером 3000, на якому працює локальний сервер, що відповідає за графічну частину представлення програмного продукту, яка в свою чергу обмінюється даними з серверною частиною продукту, відповідальною за обробку та зберігання даних.

При запуску програми на екрані з’явиться домашня сторінка, яка надає змогу продивитися всі занесені в систему документи та також здійснити пошук по обраним категоріям та записаним символам (Рис. 5.1).

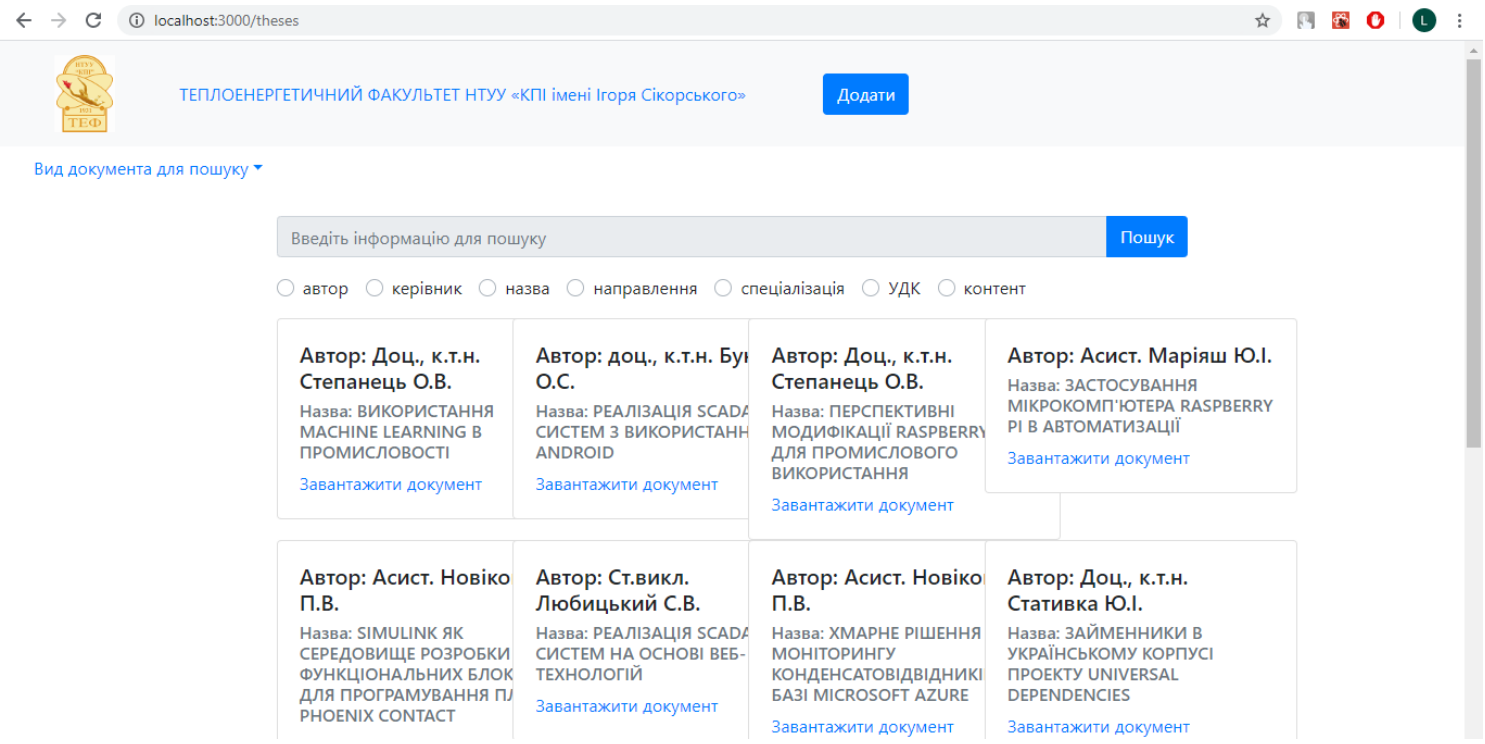

Рис. 5.1 – Вибір категорії документа

Після того щоб додати документ в регістр потрібно натиснути кнопку «Додати», після чого вид інтерфейсу миттєво зміниться (Рис. 5.2). Будуть виведені на екран список типів документів, які можна добавити до системи.

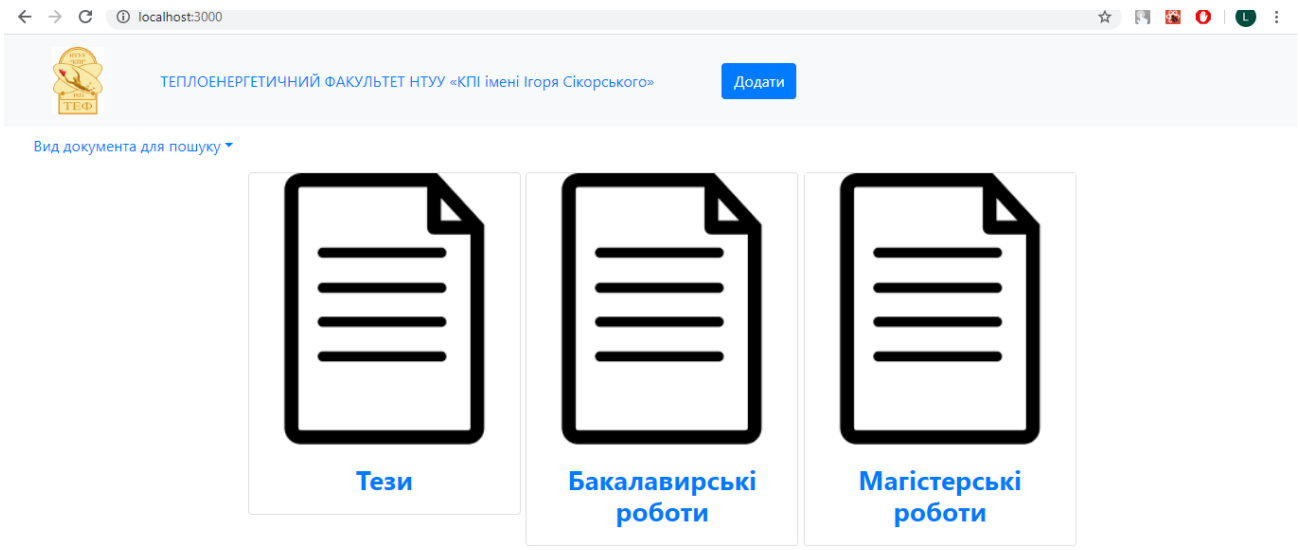

Рис. 5.2 – Вибір типу документа

Для того щоб завантажити документ до системи, потрібно натиснути на той тип документа який хоче внести користувач. Після натискання з'явиться стандартне меню завантаження файлів (Рис. 5.3) залежно від браузера який використовує клієнт. Після підтвердження запиту, від даної форми, обраний документ буде завантаженим до реєстру.

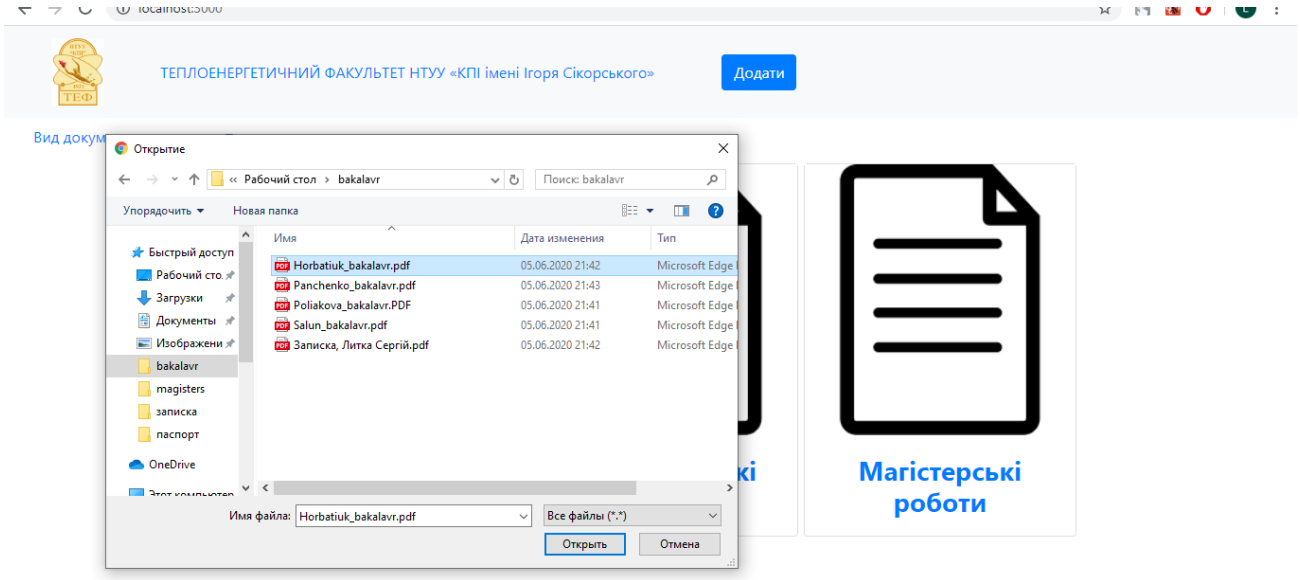

Рис. 5.2 – Вибір документа для завантаження в систему

ВИСНОВКИ

У ході даної роботи було створено систему заповнення до галузевого реєстру інформаційних ресурсів кафедри, яка може поліпшити, прискорити та оптимізувати процес роботи кафедри та допомогти у роботі користувачам, сфера зайнятості яких пов’язана з електронним документообігом. Створена система здатна зберігати вненсені до неї будь-які електронні документи. Її основними перевагами перед існуючими системами є автономність, швидкодійність та не вимагає авторизації.

Для створення системи заповнення галузевого реєстру інформаціних ресурсів кафедри були використані такі інструменти, як мова програмування JavaScript, мова запитів GraphQL, серверна платформа Node.js, фреймворк React.js, система управління базами даних MySQL, мова розмітки HTML, каскадна таблиця стилів CSS, система для керування версіями гіт GitKraken, інструменти розробника Chrome DevTools і віддалений сервіс Prisma.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Флэваган, Дэвид. JavaScrirt: карманный справочник, 3-е изд. — М., 2013. — 320 с. — ISBN 978-5-8459-1830-7.
2. Брэд Дейли, Брендан Дейли, Калеб Дейли. Разработка веб-приложений с помощью Node.js, MongoDB и Angular: исчерпывающее руководство по использованию стека MEAN = Web Development with Node and Express. — 2-е изд.. — Санкт-Петербург: «Диалектика-Вильямс», 2020. — 656 с. — ISBN 978-5-6040044-8-7.
3. Мардан Азат. React быстро. Веб-приложения на React, JSX, Redux и GraphQL. — СПб.: «Питер», 2019. — С. 560. — ISBN 978-5-4461-0952-4.
4. Когаловский М. Р. Энциклопедия технологий баз данных. — М.: Финансы и статистика, 2002. — 800 с. — ISBN 5-279-02276-4.
5. В. Васвани. MySQL: использование и администрирование = MySQL Database Usage & Administration. — М.: «Питер», 2011. — 368 с. — ISBN 978-5-459-00264-5.
6. Кузнецов Максим, Симдянов Игорь. MySQL на примерах. — Спб.: «БХВ-Петербург», 2008. — С. 952. — ISBN 978-5-9775-0066-1.
7. Поль Дюбуа. MySQL, 3-е издание = MySQL, 3ed. — М.: «Вильямс», 2006. — 1168 с. — ISBN 5-8459-1119-2.
8. Кузнецов Максим, Симдянов Игорь. MySQL 5. В подлиннике. — Спб.: «БХВ-Петербург», 2006. — С. 1024. — ISBN 5-94157-928-4.
9. Кузнецов Максим, Симдянов Игорь. Самоучитель MySQL 5. — Спб.: «БХВ-Петербург», 2006. — С. 560. — ISBN 5-94157-754-0.
10. Мардан Азат. React быстро. Веб-приложения на React, JSX, Redux и GraphQL. — СПб.: «Питер», 2019. — С. 560. — ISBN 978-5-4461-0952-4.
11. Бэнкс Алекс, Порселло Ева. GraphQL: язык запросов для современных веб-приложений. — СПб.: «Питер», 2019. — С. 240.—ISBN 978-5-4461-1143-5.

12. Бэнкс Алекс, Порселло Ева. React и Redux: функциональная веб-разработка. — СПб.: «Питер», 2018. — С. 336. — ISBN 978-5-4461-0668-4.
13. Томас Марк Тиленс. React в действии. — СПб.: «Питер», 2019. — С. 368. — ISBN 978-5-4461-0999-9.
14. Кирупа Чиннатамби. Изучаем React. — СПб.: «Питер», 2019. — С. 368. — ISBN 978-5-04-098028-4.
15. Брэд Дейли, Брендан Дейли, Калеб Дейли. Разработка веб-приложений с помощью Node.js, MongoDB и Angular: исчерпывающее руководство по использованию стека MEAN = Web Development with Node and Express. — 2-е изд.. — Санкт-Петербург: «Диалектика-Вильямс», 2020. — 656 с. — ISBN 978-5-6040044-8-7.
16. Итан Браун. Веб-разработка с применением Node и Express. Полноценное использование стека JavaScript = Web Development with Node and Express / Итан Браун. — Санкт-Петербург: Питер, 2017. — 336 с. — ISBN 978-1-491-94930-6.
17. Каскиаро М., Маммино Л. Шаблоны проектирования Node.js. — 2017. — С. 396. — ISBN 978-5-97060-485-4.
18. ECMA-262, ECMAScript Language Specification.
19. JavaScript домашня сторінка, JavaScript довідник на mozilla.org.
20. Зміни в нових версіях JavaScript: 1.7, 1.6
21. JScript та JScript .NET довідники на сайті MSDN Library
22. GraphQL June 2018 Release Notes

ДОДАТОК 1

Заповнення галузевого реєстру інформаційних ресурсів кафедри

Специфікація

УКР.НТУУ«КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТР61

Аркушів 8

Київ – 2020

Позначення	Найменування	Примітки
Документація		
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТР61_109 20Б 81-1	Кошмак В. Л._ТР61.docx	Пояснювальна записка
Компоненти		
УКР.НТУУ «КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТР61 20Б 12-1	my-app.js	Модулі клієнтської частини та інтерфейсу програмного продукту
УКР.НТУУ «КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТР61 20Б 12-2	back-app.js	Модуль серверної частини
УКР.НТУУ «КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТР61 20Б 13-2	Опис.docx	Опис модуля серверної частини програми

ДОДАТОК 2

Заповнення галузевого реєстру інформаційних ресурсів кафедри

Текст програми

УКР.НТУУ«КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТР61_20Б 12-2

Аркушів 8

Київ – 2020

```
const fs = require('fs');
const pdf = require('pdf-parse');
const { fullResource } = require("../fragments/Resource")
const { parseThethis } = require("../parsers/Thethis")
const { parseMagister } = require("../parsers/Magister")
const { parseBakalavr } = require("../parsers/Bakalavr")

function postResource(parent, args, context, info) {
  const { author, subAuthor, title, direction, profession, specialization, udc, content }
= args

  return context.prisma.createResource({
    author: {
      create: {
        fullName: author
      }
    },
    subAuthor: {
      create: {
        fullName: subAuthor
      }
    },
    title: {
      create: {
        title
      }
    },
    direction: {
      create: {
```

```
        code: direction
      }
    },
    profession: {
      create: {
        profession
      }
    },
    specialization: {
      create: {
        specialization
      }
    },
    udc: {
      create: {
        udc
      }
    },
    content: {
      create: {
        content
      }
    }
  }).$fragment(fullResource)
}

async function createTheFile(file) {
  return new Promise(resolve => {
    //ToDo remove file from this, understand why
```

```
    const { createReadStream, filename, mimetype } = file

    const readFileStream = createReadStream()
    const readWriteStream = fs.createWriteStream(`./uploadedFiles/${filename}`)
    readFileStream.pipe(readWriteStream)
    readWriteStream.on('finish', () => {
      resolve()
    })
  })
}

function singleUpload(parent, args, context, info) {
  return args.file.then(async file => {
    await createTheFile(file)
    const { createReadStream, filename, mimetype } = file

    //TODO rewrite
    let result
    if (args.type == "THETHIS") {
      result = await parseThethis(`./uploadedFiles/${filename}`)
    } else if (args.type == "BAKALAVR") {
      result = await parseBakalavr(`./uploadedFiles/${filename}`)
    } else if (args.type == "MAGISTER") {
      result = await parseMagister(`./uploadedFiles/${filename}`)
    }
    if (args.type == "THETHIS") {
      result.forEach(async res => {
        const { author, subAuthor, title, direction, profession, specialization, udc,
content } = res
```

```
const request = {}
if (author) {
  request.author = {
    create: {
      fullName: author
    }
  }
}
if (subAuthor) {
  request.subAuthor = {
    create: {
      fullName: subAuthor
    }
  }
}
if (title) {
  request.title = {
    create: {
      title
    }
  }
}
if (direction) {
  request.direction = {
    create: {
      code: direction
    }
  }
}
```

```
if (profession) {
  request.profession = {
    create: {
      profession
    }
  }
}
if (specialization) {
  request.specialization = {
    create: {
      specialization
    }
  }
}
if (udc) {
  request.udc = {
    create: {
      udc
    }
  }
}
if (content) {
  request.content = {
    create: {
      content
    }
  }
}
await context.prisma.createResource({ ...request }).$fragment(fullResource)
```

```
    })
  } else {
    const { author, subAuthor, title, direction, profession, specialization, udc,
content } = result
    const request = {}
    if (author) {
      request.author = {
        create: {
          fullName: author
        }
      }
    }
    if (subAuthor) {
      request.subAuthor = {
        create: {
          fullName: subAuthor
        }
      }
    }
    if (title) {
      request.title = {
        create: {
          title
        }
      }
    }
    if (direction) {
      request.direction = {
        create: {
```

```
      code: direction
    }
  }
}
if (profession) {
  request.profession = {
    create: {
      profession
    }
  }
}
if (specialization) {
  request.specialization = {
    create: {
      specialization
    }
  }
}
if (udc) {
  request.udc = {
    create: {
      udc
    }
  }
}
if (content) {
  request.content = {
    create: {
      content
```

```
          }
        }
      }
      await context.prisma.createResource({ ...request }).$fragment(fullResource)

    }

    return file;
  });
}

module.exports = {
  postResource,
  singleUpload
}
```

ДОДАТОК З

Заповнення галузевого реєстру інформаційних ресурсів кафедри

Опис програми

УКР.НТУУ«КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТМ52_19Б 13-2

Аркушів 8

Київ – 2020

АНОТАЦІЯ

Додаток надає інформацію про програмну систему заповнення галузевого реєстру інформаційних ресурсів кафедри.

Розроблене програмне забезпечення дозволяє вносити документи до бази даних та виводити результат.

Клієнтський застосунок було розроблено в середовищі Microsoft Visual Studio Code з використанням платформи Node.js та мови JavaScript.

ЗМІСТ

-4-

ЗАГАЛЬНІ ВІДОМОСТІ

Відповідно до теми дипломної роботи, програма має назву «Інформаційний галузевий реєстр кафедри АПЕПС».

Програма працює на віддалених сервісах Prisma та потребує доступу до мережі інтернет, в свою чергу це забезпечує надійність та швидкість системи.

Застосунок був написаний мовою JavaScript з використанням бібліотеки React.js.

-5-

ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Розроблений програмний засіб покликаний вирішити задачу заповнення до галузевих інформаційних ресурсів, щоб спростити роботу працівникам кафедри, що відповідають за документообіг. Це було реалізовано за допомогою кількох функцій системи, а саме:

– Наявності зручного інтерфейсу;
– Швидкого заповнення даних до БД;
– Можливості переглянути завантажені документи.

-6-

ОПИС ЛОГІЧНОЇ СТРУКТУРИ

Програмний продукт складається з клієнтської та серверної частини.

Загальний принцип роботи додатку такий:

1) користувач змінює стан сторінки (наприклад, тисне на кнопку);
2) відбувається звернення до пов’язаного обробника подій;
3) метод, що викликається в обробнику виконує запит;
4) результати виконання методу виводяться на сторінку.

Продемонструвати роботу програми можна на прикладі виконання будь-якого запиту, наприклад для занесення бакалаврської роботи.

Після натискання на картку «Бакалаврські роботи» викликається метод для завантаження файлу, що викликає вікно браузера для завантаження файлів, після обрання файлу, він завантажується до серверної частини де обробляється та перевіряється і вже після заноситься до бази даних.

-7-

ВИКОРИСТОВУВАНІ ТЕХНІЧНІ ЗАСОБИ

Клієнт-серверна система функціонує завдяки наступним використаним засобам. База данних MySQL дає відповідь на запит одержаний Express сервером й оброблений GraphQL, відповідь отримана клієнською частиною обробляєть Apollo-клієнтом, та передається до React.js додатку який генерує графічне представлення відповіді.

Для організації доступу до програмного продукту потрібно мати комп'ютер з доступом до інтернету, операційною системою Windows або Linux, встановленим браузером та платформа Node.js.

-8-

ВХІДНІ І ВИХІДНІ ДАНІ

Вхідними даними є:

— pdf файл;

Вихідними даними є:

— результати виконання запиту в MySQL;