

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»**

Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено:

Завідувач кафедри

_____ Олександр Коваль

«__» _____ 2021 р.

Дипломна робота

на здобуття ступеня бакалавра

**за освітньо-професійною програмою «Інформаційні технології моніторингу
довкілля»**

спеціальності 122 «Комп'ютерні науки та інформаційні технології»

**на тему: «Програмне забезпечення організації та супроводу міжнародних
конференцій»**

Виконав:

студент IV курсу, групи ТМ-71

Архипенко Дмитро Павлович

Керівник:

доцент

Сегеда Ірина Василівна.

Рецензент:

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

Студент _____

Київ – 2021 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

Напрямок підготовки 122 Комп'ютерні науки та інформаційні технології

Спеціалізація Інформаційні технології моніторингу довкілля

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Олександр Коваль

(підпис)

” ____ ” _____ 2021р.

ЗАВДАННЯ

на дипломну роботу студенту

Архипенку

Дмитру

Павловичу

(прізвище, ім'я, по батькові)

1. Тема роботи: Програмне забезпечення організації та супроводу міжнародних конференцій.

керівник роботи Сегеда Ірина Василівна, доцент.

(прізвище, ім'я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від

2. Строк подання студентом роботи _____

3. Вихідні дані до роботи

мова програмування JavaScript, бібліотека React.js, середовище розробки WebStorm, платформа Node.js

4. Зміст розрахунково-пояснювальної записки (перелік завдань, які потрібно розробити) ознайомитись з існуючими сайтами наукових конференцій, визначитись з засобами розробки сайту для реалізації БД, frontend частини та backend частини майбутнього сайту, алгоритм розробки програмного додатку, розробити програмне забезпечення, розробити інтерфейс користувача, зробити висновки за результатами роботи.

5. Перелік ілюстративного матеріалу Клієнт серверна структура, Тіло запиту на реєстрацію, Схема роботи драйвера, Колекція з ролями для користувачів сайту, Колекція з користувачами для сайту, Тіло запиту для авторизації в системі, Схема роботи авторизації на сайті, Тіло запиту на відправку листа поновлення

пароллю, Лист поновлення пароллю входу на сайт, Тіло запиту на зміну пароллю, Файлова архітектура ресурсів, Тіло запиту завантаження файлу, Відповідь сервера на отримання файлів типу pdf, Схема роботи сервісу з використанням Socket.io, Заголовок веб сайту, Нижній колонтитул сайту, Частина 1. контенту головної сторінки, Частина 2. контенту головної сторінки, Посилання Галерея, Сторінка з фотогалереєю, Посилання на сторінки Презентацій та Архівів робіт, Сторінка Презентацій на веб сайті, Контролер для гортання сторінок, Сторінка архівів з роботами, Меню навігації по завантаженим файлам, Форма подання заявки на участь, Демонстрація валідації даних Повідомлення про успішно подану заявку, Повідомлення іншого користувача, Повідомлення авторизованого користувача, Користувацький чат на веб сайті, Менеджер фото галереї, Активна відкрита файлова система для завантаження файлів, Видалення завантажених файлів на сервер, Таблиця з зареєстрованими користувачами, Файлова система ресурсів

6. Дата видачі завдання "10" жовтня 2021 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Затвердження теми роботи	9.10.20р.	
2.	Вивчення та аналіз задачі	12.04.21р. – 16.04.21р	
3.	Розробка архітектури та загальної структури системи	17.04.21р. – 22.04.21р.	
4.	Розробка структур окремих підсистем	23.04.21р – 25.04.21р.	
5.	Програмна реалізація системи	26.04.21р – 08.05.21р	
6.	Оформлення пояснювальної записки	09.05.21р – 20.05.2021	
7.	Захист програмного продукту	11.05.2021р.	
8.	Передзахист	22.05.2021р.	
9.	Захист	14.06.2021р	

Студент

(підпис)

Архипенко Д. П.

(прізвище та ініціали,)

Керівник роботи

(підпис)

Сегеда І.В.

(прізвище та ініціали,)

АНОТАЦІЯ

На сьогоднішній день наукові конференції є невід'ємною частиною процесу розвитку науки. З кожним роком кількість бажаючих прийняти участь тільки збільшується, з цим збільшується складність організації. Тому існує ряд проблем, які необхідно вирішувати для продуктивної і стабільної роботи з організації освітніх заходів.

Саме тому дана робота присвячена створенню сайту для організації наукової конференції.

Для реалізації поставленою мети поставлені наступні задачі:

- аналіз методів та засобів розробки сайту конференції;
- створення структури сайту;
- розробка серверу для сайту
- створення дизайну та користувацького інтерфейсу сайту;
- введення сайту в експлуатацію.

Обсяг пояснювальної записки складає 47 сторінок і містить у собі 45 ілюстрацій, та 21 посилань на літературу.

Ключові слова: наукові читання Ігоря Недіна, конференція, веб сайт, Javascript, Node.js, Soker.Io, SendGrid.

ABSTRACT

Today, science conferences are an integral part of the development process. Every year the number of interested participants decreases, as a result of which the volume of the organization increases. Therefore, there are a number of problems, allegedly not necessary for productive and stable work on the organization of educational activities.

According to words I said above this work is devoted to the creation of a site for the organization of a scientific conference.

- To achieve this goal it was decided to put the following tasks:
analysis of methods and means of developing the site Scientific Conference;
- implementation of site structure;
- create a site design and user interface of the website;
- creation of Web-site finished with the filling material
- deploy of created web service

The volume of the explanatory note is 47 pages and contains 45 illustrations, yes 21 references.

Keywords: Keywords: scientific readings of Igor Nedin, conference, web site,
Javascript, Node.js, SendGrid

Зміст

ВСТУП	9
1. ПОСТАНОВКА ЗАДАЧІ.....	Ошибка! Закладка не определена.
2. ОГЛЯД ІНТЕРНЕТ САЙТІВ ОСВІТНЬО-НАУКОВОГО НАПРЯМУ	11
2.1 Поняття веб сайту.....	11
2.2 Обґрунтування потреби в веб-сайті	11
2.3 Обґрунтування вибраного веб-редактора	12
3. ТЕОРЕТИЧНІ АСПЕКТИ РОЗРОБКИ ВЕБ-САЙТУ	13
3.1 Планування веб-сайту	13
3.2 Вибір засобів розробки веб-сайту.....	14
3.3 Розробка інтерфейсу	14
3.4 Основна мова програмування для розробки Javascript	15
3.5 Node.js – створення сервера на Javascript	15
3.6 Мова розмітки гіпертекстових документів HTML	16
3.7 Каскадні таблиці стилів CSS.....	16
3.8 Бібліотека для динамічного формування контенту React.js	17
3.9 MongoDB бази даних	17
4. РОЗРОБКА СЕРВЕРНОЇ ЧАСТИНИ ВЕБ САЙТУ	18
4.1 Процес авторизації користувачів у системі.....	19
4.2 Збереження даних у базі та отримання їх у середовищі сервера Node.js...	20

4.3 Визначення активної сесії для авторизованого користувача.....	22
4.4 Поновлення паролю для входу на сайт	24
5. УПРАВЛІННЯ КОНТЕНТОМ ТА ЗАВАНТАЖЕНИМИ ФАЙЛАМИ НА СЕРВЕР	27
5.1 Бібліотека Multer. Робота з файловою системою сервера	27
5.2 Запити файлових контролерів веб сайту.....	27
6. ІМПЛЕМЕНТАЦІЯ СЕРВЕРНОЇ ЧАСТИНИ ДЛЯ КОРИСТУВАЦЬКОГО ЧАТУ	29
7. РОЗРОБКА КОРИСТУВАЦЬКОГО ІНТЕРФЕЙСУ ДЛЯ ВЕБ САЙТУ.....	31
7.1 Аналіз серверної частини та визначення сторінок сайту	31
7.2 Розробка статичних секцій сайту	32
7.3 Формування контенту для головної сторінки	33
7.4 Інтерфейс сторінки фотогалереї	34
7.5 Інтерфейс сторінок презентацій та архівів робіт	35
7.6 Заявка на участь.....	38
7.7 Створення інтерфейсу для процесу авторизації на сайті	40
7.8 Користувацький чат	45
7.9 Адмін панель вебсайту	47
ВИСНОВКИ.....	50
ДОДАТОК 1.....	53
ДОДАТОК 2.....	55
ДОДАТОК 3.....	70

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Javascript - мова програмування.

Software architecture - архітектура програмного продукту. Це структура програми або обчислювальної системи, яка включає програмні компоненти, а також відносини між ними.

Model - це об'єкт, який надає абстрактний інтерфейс бази даних.

JWT token - згенерований рядок яка використовується для авторизації

Mongo - система управління базам даних яка базується на не реляційній парадигмі без явних зв'язків між колекціями об'єктів

Node.js - це середовище виконання із відкритим вихідним кодом, кросплатформерне та внутрішнє середовище виконання JavaScript, яке працює на движку V8 і виконує код JavaScript поза веб-браузером.

React.js - це відкрита бібліотека JavaScript для побудови користувацьких інтерфейсів або компонентів інтерфейсу.

SendGrid - це комунікаційна платформа для транзакційних та маркетингових електронних листів.

Express - це мінімальна та гнучка структура веб-додатків Node.js, яка забезпечує надійний набір функцій для веб- і мобільних додатків.

Socket.IO - Javascript бібліотека яка забезпечує двосторонню зв'язок у реальному часі та на основі подій. Він працює на будь-якій платформі, в браузері чи на пристрої, дедіє одне враження надійності та швидкості.

ВСТУП

На сьогоднішній день наукові конференції є невід’ємною частиною процесу розвитку науки. З кожним роком кількість бажаючих прийняти участь тільки збільшується, з цим збільшується складність організації. Коли в рамках конференції участь приймає відносно невелика кількість людей, то для організаційного комітету не є складно влаштувати захід однак коли кількість учасників перевищує поріг навіть в 50 чоловік то стає все складніше контролювати процес та доносити інформацію про останні події або плани на наступну конференцію особливо коли це відбувається в індивідуальному порядку з кожним з учасників.

Тому існує ряд проблем, які необхідно вирішувати для продуктивної і стабільної роботи з організації освітніх заходів.

Метою даної дипломної роботи є створення веб сайту для організації та супроводження конференції.

База даних в рамках дипломної роботи передбачає збереження даних про зареєстрованих користувачів та учасників поточної або минулих конференцій.

Також з одної з цілей є реалізація механізму автоматизованої відправки листів для імплементації функції поновлення паролю та отримання нових заявок на участь у наступній конференції використано

Для вирішення проблеми комунікації з учасниками та з метою уникнення дублювання інформації розроблено користувацький чат в якому будь хто з зареєстрованих користувачів має змогу прочитати останні новини або ж задати питання які його цікавлять.

Програмний комплекс також має передбачати можливість перегляд архівів робіт, презентацій а також фотографій з минулорічних конференцій.

1. ПОСТАНОВКА ЗАДАЧІ

Метою дипломної роботи є створення уніфікованого сервісу для організації та супроводу наукової конференції і дане завдання розробку веб сайту доступ до якого здійснюється через мережу інтернет.

Сайт наукової конференції — веб-сайт, який включає в себе систему управління контентом. Наповнення даного сайту повинно бути простим та зрозумілим, мати безліч функцій для редагування. Даний сайт розроблявся з метою донесення інформації про конференцію, її склад та наукові напрями які освітлюються на щорічних заходах. Сайт дозволяє виконувати такі задачі як:

- Додання та редагування новин конференції
- Інформування аудиторії
- Розміщення необхідних розділів сайту
- Реалізація можливості подання заявки на участь у конференції
- Розміщення інформації про організаційний та науковий комітети конференції
- Додавання матеріалів для бажаючих прийняти участь
- Створення користувацького чату для комунікації спільноти в рамках конференції
- Авторизація на сайті

Сайт повинен у повному обсязі надавати інформацію про конференцію, та допомагати керівникам в організації освітніх заходів.

2. ОГЛЯД ІНТЕРНЕТ САЙТІВ ОСВІТНЬО-НАУКОВОГО НАПРЯМУ

2.1 Поняття веб сайту

Веб-сайт — це сукупність логічно зв'язаної гіпертекстової інформації, оформленої у вигляді окремих сторінок і доступної в мережі Інтернет. [19]

Це визначення веб-сайту було правильним на початку формування інтернету. В 90-х роках були в основному статичними сторінками. Для розробки даних ресурсів потрібно було знати та використовувати лише мову гіпертекстової розмітки — HTML. Якщо ж сторінка надавала якісь програмні засоби — це були виключно засоби, що міг надати сервер, на якому розташований веб-сайт. Про зручність і красу тогочасних веб-сайтів взагалі особливо не доводилося говорити. Але з плином часу все у світі змінюється і веб теж не стоїть на місці.

2.2 Обґрунтування потреби в веб-сайті

В даний час розвиток інформаційних технологій є не тільки найважливішою областю знань, але є механізмом функціонування та розвитку соціальних відносин. На сьогоднішній день практично кожна організація має власний веб-сайт. Сайт є інструментом для ознайомлення з тою чи іншою організацією. [20]

Однак окрім цього наявність веб сайту також:

1. Розширює можливості та сприяє популяризації ідей та тематики закладеної конференцію;
2. Спрощує процес організації освітніх заходів;
3. Надає можливість висвітлювати останні події відносно організації;
4. Надає інструкції що до того як прийняти участь в подальших заходах
5. Може відображати результати роботи конференції
6. Містити звіти або ж архіви робіт за минулі роки діяльності конференції.

2.3 Обґрунтування вибраного веб-редактора

В наш час існує безліч програм для створення сайту та розміщення його в інтернеті. При цьому потрібно пам'ятати, що це лише інструмент для розробника. Але це не впливає на те чи сайт буде привертати увагу користувачів або чи буде він успішний у бізнесі. Це залежить тільки від маркетингу та фантазії дизайнера (розробника).

Для створення та редагування сайту було обрано IDE WebStorm.

Переваги IDE WebStorm:

1. ІСР надається безкоштовно для студентів та працівників ВНЗ.
2. Можливість скасування дії.
3. Можливість віддаленого підключення до FTP.
4. Підтримка плагінів.
5. Автодоповнення коду які фіналізують, класи, методи, теги і т.д.

3. ТЕОРЕТИЧНІ АСПЕКТИ РОЗРОБКИ ВЕБ-САЙТУ

3.1 Планування веб-сайту

На етапі планування, перш за все, слід визначити призначення майбутнього сайту. Тут доцільно визначити, буде сайт тематичним чи різні його сторінки будуть присвячені різним темам і яким саме. На етапі планування можна виділити наступні задачі:

- фіксація цілей і завдання сайту;
- максимально детальний опис майбутньої аудиторії;
- визначення того, що ми чекаємо від відвідувачів сайту;
- аналіз сайтів;
- планування того, чим наш майбутній сайт буде відрізнятися від інших;
- вибір сервісів та функцій;
- аналіз структури майбутнього сайту;
- планування наповнення для основних сторінок / розділів;
- створення візуального плану ключових сторінок;
- оцінка зручності використання сайту користувачем;
- планування способу отримання зворотного зв'язку;
- оцінка перспективи розвитку сайту;

3.2 Архітектурний шаблон розробки сайту

На сьогоднішній день при розробці веб сайту з нуля найбільш розповсюдженим та популярним підходом розробка згідно клієнт архітектурної структури і нижче представлена схема за якою працює сайт в рамках цього підходу розробки. (див рис.1)

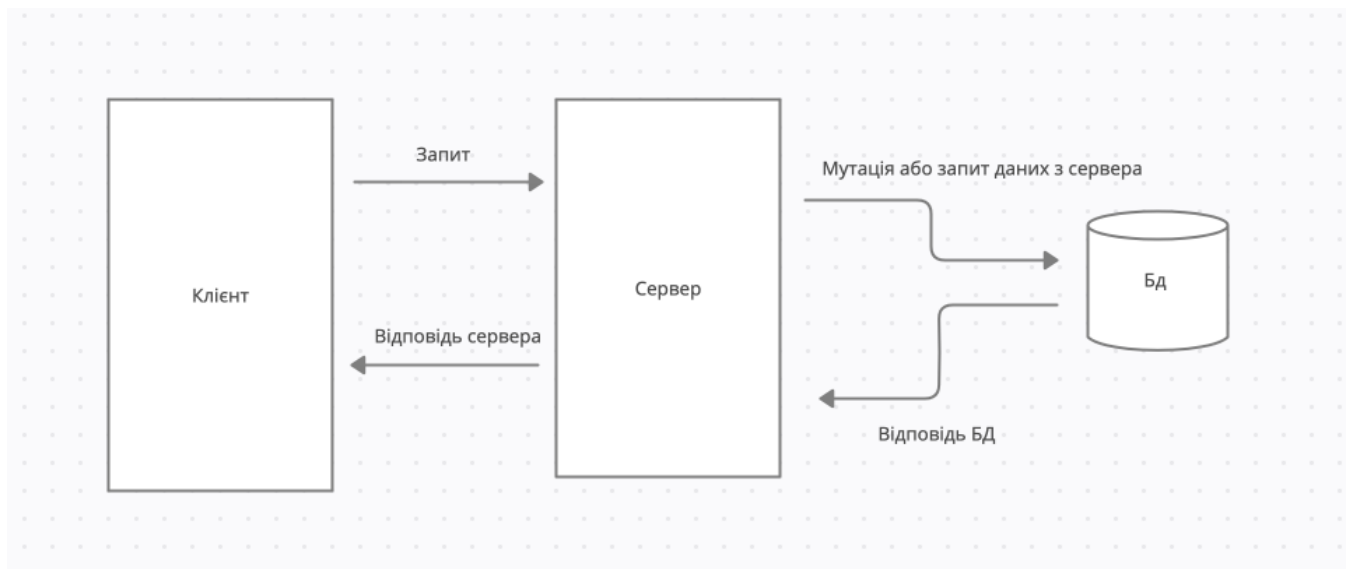


Рисунок. 1 Клієнт серверна структура

Умовна незалежність складових додатку дає можливість визначати технології однієї частини, наприклад клієнта, незалежно від стеку обраних технологій на сервері [16].

3.3 Вибір засобів розробки веб-сайту

Враховуючи поставлені вище завдання найбільш оптимальним засобом для розробки такого роду – застосування мови розмітки документів у мережі інтернет - HTML, формальної мови опису зовнішнього вигляду документа, написаного з використанням мови розмітки - CSS, прототипно-орієнтованого сценарного типу мови програмування JavaScript з використання баз даних MongoDB. Вибір JS є дуже вдалим оскільки браузер та фронтенд працює на JavaScript і це надає переваги з того боку що ми можемо гнучко переміщати логіку між сервером та клієнтом.

3.4 Розробка інтерфейсу

Титульна сторінка (головна) будь-якого сайту повинна максимально інформативно і в стислому об'ємі відображати необхідну користувачеві інформацію про сайт. На головній сторінці необхідно помістити логотип веб-сайту, основне меню сайту (для навігації з його структури), новини та загальну інформацію про

конференцію, організаційний склад, контакти а також інформацію що до участі в конференції

3.5 Основна мова програмування для розробки JavaScript

В якості основної мови програмування ми обрали JavaScript оскільки в останні декілька років він отримав не мало оновлень ядра швидкість виконання функцій на якому значно підвищилась, також використовуючи JS ми можемо використовувати як і на фронтенді нашого сайту у React.js так і на серверній частині у Node.js [1]. JavaScript - це динамічна мова комп'ютерного програмування. Він легкий і найчастіше використовується як частина веб-сторінок, реалізації яких дозволяють клієнтському сценарію взаємодіяти з користувачем та створювати динамічні сторінки. Це інтерпретована мова програмування з об'єктно-орієнтованими можливостями. JavaScript спочатку був відомий як LiveScript. Ядро мови загального призначення було вбудовано в Netscape, Internet Explorer та інші веб-браузери.

3.6 Node.js – створення сервера на Javascript

Вибір сервера на JS це дуже гарне рішення оскільки фронтенд який відображається на стороні клієнта також виконується на JS і це дозволяє гнучко мігрувати логіку з фронтенду на сервер поступово зменшуючи навантаження з клієнта або ж навпаки валідувати більшість даних на стороні клієнту перед тим як відправляти запит на сервер [4].

Node.js - це не мова програмування, а середовище виконання, що дозволяє виконувати JavaScript на стороні сервера. Тепер JavaScript, який майже так само старий, як Java, є інтерпретованою мовою, яка використовувалася на стороні клієнта протягом тривалого часу, поки не з'явився Node.js. Node.js з його концепцією "JavaScript скрізь" також дозволив використовувати JavaScript на серверній панелі.

3.7 Мова розмітки гіпертекстових документів HTML

HTML (англ. HyperText Markup Language — Мова розмітки гіпертекстових документів) — основана на SGML текстова мова розмітки, призначена для маркування документів, що містять текст, зображення, гіперпосилання, тощо. HTML-документи лежать в основі Веб, і відображаються із допомогою веб-браузерів. Разом із видимою інформацією, HTML-документи містять додаткові метадані, такі як, наприклад, мова тексту, автор документа, стислий підсумок. Мова розмітки розроблялась консорціумом W3C, остання версія — 4.01, очікується, що HTML буде замінена розширюваною мовою розмітки гіпертексту (XHTML). Найпростіша веб-сторінка складається з текстових блоків, декількох рисунків, горизонтальних розмежувальних ліній та гіперпосилань. Більш складні веб-сторінки містять фрейми, елементи керування, динамічні ефекти та анімовані об'єкти. Крім свого стандартного застосування таблиці в html-мові також використовують, якщо потрібно розташувати якийсь текст чи об'єкт у певному місті на веб-сторінці.

3.8 Каскадні таблиці стилів CSS

Для додавання стилів до елементів розмітки щоб реалізувати дизайн поверх сухої розмітки сторінки розробники використовують CSS [14].

Каскадні таблиці стилів (CSS) є потужним інструментом для керування зовнішній вигляд вашого веб-сайту, вплине на уявлення документа або набору документів.

Переваги CSS:

- CSS дозволяє значно багатший, ніж документ виступу HTML все дозволено, навіть у розпал презентаційних запал HTML;
- CSS дозволяє задати кольору на текст і на тлі будь-якого елементу;
- CSS дозволяє створювати рамки навколо будь-якого елемента, а також збільшення або зменшення простору навколо них;

Каскадні таблиці стилів являє собою набір інструкцій, який вказує веб-браузер, як представити або дисплей, різні HTML-елементи, такі як, який шрифт

використовувати, який розмір тексту абзацу повинен бути, який колір тексту заголовка має бути, або Не повинно бути колір фону стосовно до конкретних елементів, і так далі. Ви можете використовувати кілька таблиць стилів, щоб налаштувати дисплей вашого веб-сайту в різних браузерах, різних платформах і різних пристроїв.

3.9 Бібліотека для динамічного формування контенту React.js

ReactJS - це бібліотека JavaScript, яка використовується для побудови компонентів інтерфейсу, що використовуються багаторазово. Згідно з офіційною документацією React, таке визначення. [2]

Це надає можливість створення багаторазових компонентів інтерфейсу, які представляють дані, що змінюються з часом. React абстрагує від вас DOM, пропонуючи простішу модель програмування та кращу продуктивність. React реалізує односторонній реактивний потік даних, що зменшує шаблон і легше міркувати, ніж традиційне прив'язування даних.

React наполегливо рекомендує використовувати JSX при описуванні компонентів. JSX – розширення Javascript що дозволяє вставляти фрагменти JS які при компіляції дозволять динамічно відображати дані на сторінках сайту

Основні перевагами бібліотеки React.js:

- динамічна генерація Html;
- можливість використовувати JSX;
- незалежність від версії Javascript;
- легка масштабованість;
- можливість перевикористання існуючих компонентів;

3.10 MongoDB бази даних

MongoDB - це міжплатформна, орієнтована на документи база даних, яка забезпечує високу продуктивність, високу доступність та просту масштабованість. MongoDB працює над концепцією колекції та документа [5,6].

База даних - це фізичний контейнер для колекцій. Кожна база даних отримує власний набір файлів у файловій системі. Один сервер MongoDB зазвичай має кілька баз даних які в свою чергу складаються з колекцій.

Колекція - це група документів MongoDB. Це еквівалент таблиці RDBMS. Колекція існує в межах однієї бази даних. Колекції не застосовують схему. Документи в колекції можуть мати різні поля. Як правило, усі документи у колекції мають подібне або пов'язане призначення.

Документ - це набір пар ключ-значення. Документи мають динамічну схему. Динамічна схема означає, що документи в одній колекції не повинні мати однаковий набір полів або структури, а загальні поля в документах колекції можуть містити різні типи даних.

Опираючись до вище описаного стеку обраних технологій та цілей визначених технічним завданням найбільш вдалою ідеєю буде притримуватись клієнт серверної архітектури побудови веб сайту, що полягає у розробці трьох умовно незалежних компоненти, а саме:

- Клієнт з використанням HTML, CSS React.js
- Сервер на Node.js
- База даних MongoDB

4. РОЗРОБКА СЕРВЕРНОЇ ЧАСТИНИ ВЕБ САЙТУ

Основою будь якого сервера є маршрутизація. Маршрутизація відноситься до визначення того, як програма реагує на запит клієнта до певної кінцевої точки, яка є URI (або шляхом) та конкретним методом HTTP-запиту (GET, POST тощо).

Кожен маршрут може мати одну або кілька функцій обробника, які виконуються, коли маршрут узгоджується.

Тож для обробки будь якого запиту на сервер нам необхідно виконати 2 кроки:

1. Визначити route (маршрут по якому виконується запит)
2. Визначити контролер який буде вирішувати що зробити з вхідними даними та що віддати клієнту

4.1 Процес авторизації користувачів у системі

Для авторизації користувачів в системі перш за все нам необхідно визначити моделі, згідно яких ми створимо колекції в MongoDB для збереження даних про зареєстрованих користувачів.

В обов'язковому порядку ми передбачаємо наступну структуру для форми реєстрації на сайті, користувач має заповнити наступне:

- Прізвище Ім'я По-батькові
- Місце роботи/навчання
- Наукова ступінь
- Email
- Номер телефону
- Пароль доступу до особистого кабінету

Для адміністраторів сайту необхідно створити додаткові умови для можливості перегляду зареєстрованих користувачів та учасників наступної конференції.

Цю задачу ми можемо вирішити наступним чином, створити додаткову колекцію в базі даних з ролями яка буде містити 2 ролі:

- ROLE_USER
- ROLE_ADMIN

Таким чином при реєстрації ми можемо завчасно створити облікові записи для адміністраторів визначивши для них обидві ролі, а після запуску сервера в контролерах визначити що кожен наступний користувач який буде заповнювати форму отримає тільки роль користувача.

Опираючись на структуру даних описану вище перевіряючи інформацію авторизованого користувача ми матимемо змогу визначати роль користувача і в

залежності від цього обмежувати або відкривати доступ користувачу до певних сторінок сайту.

В ході реалізації створено маршрут з запитом POST на /api/registration для отримання даних про тільки зареєстрованого користувача. Тіло запиту представлено на скріншоті нижче. (див рис. 2) [17]

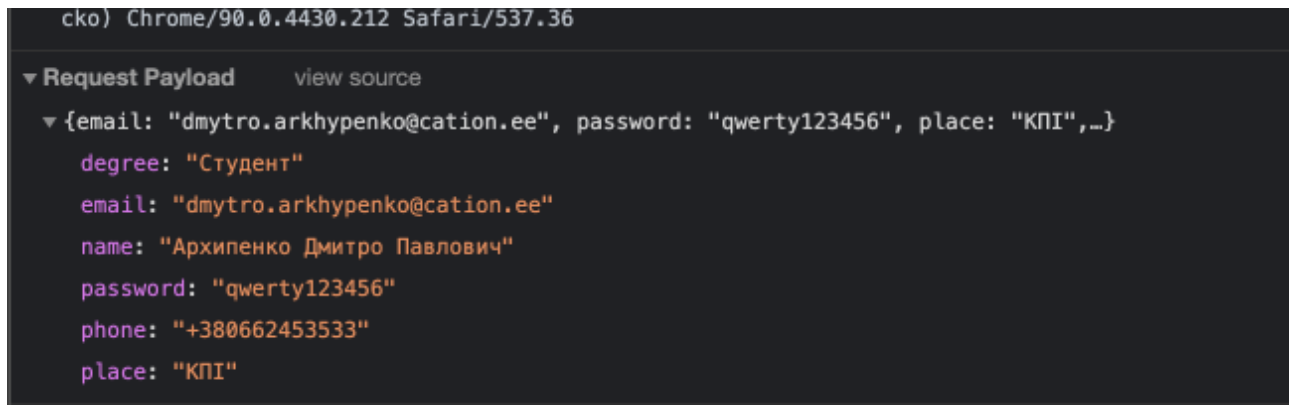


Рисунок 2. Тіло запиту на реєстрацію

4.2 Збереження даних у базі та отримання їх у середовищі сервера Node.js

Незважаючи на те, що MongoDB є безсхемовою базою даних і спирається на дисципліну розробника, щоб підтримувати структуру документа, структуровані дані потрібні для правильної роботи більшості додатків. Для вирішення цього питання була розроблена бібліотека Mongoose. Він застосовує стандартну структуру для всіх документів у колекції за допомогою схеми. Він також може перевірити дані, що зберігаються в документах, а також дозволити зберігати в базі даних лише дійсні дані.

Нижче представлені схема роботи драйвера Mongoose в рамках сервера. (див рис. 3)

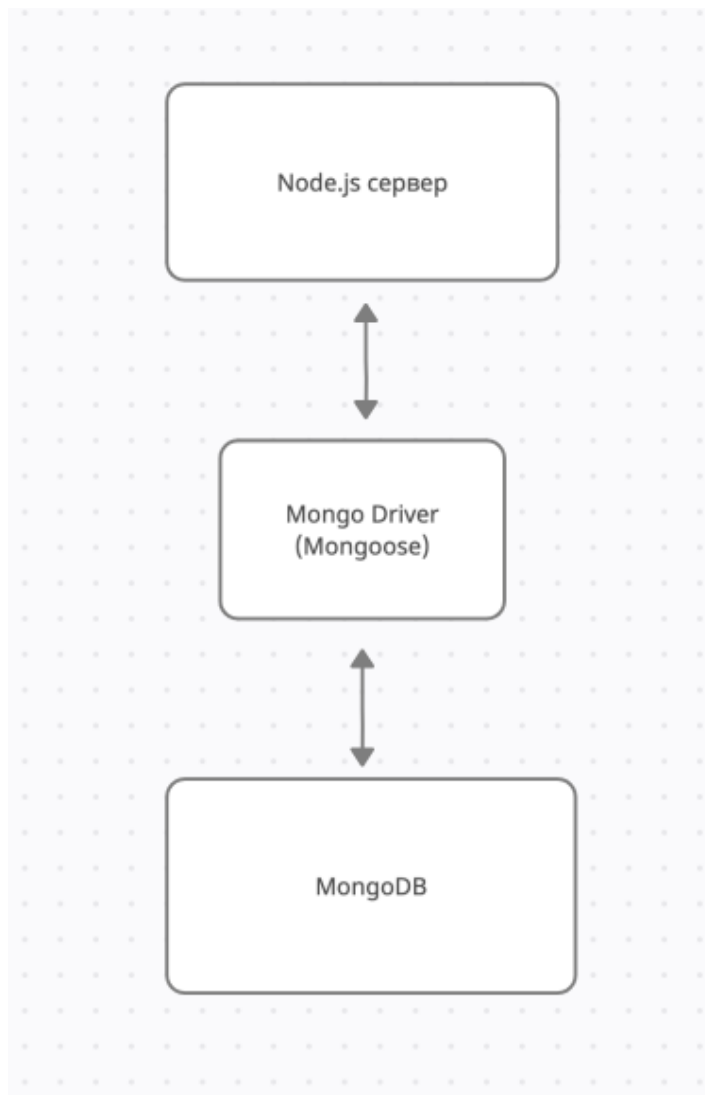


Рисунок 3. Схема роботи драйвера

Схема чітко відображає ідею роботи Mongoose як драйвера, він створює свого роду прокладку між сервером та базою даних і всі запити до БД проходять через дану прокладку. На додаток до цього, mongoose надає всі функції, що існують у MongoDB, із додаванням запитів функцій побудови запитів та бізнес-логіки в дані. Більше того, він встановлює зв'язок бази даних із сервером та реалізує операції для читання, запису, оновлення та видалення даних.

В ході аналізу даних необхідних для збереження інформації про користувачів в нашій MongoDB ми створили 2 колекції для користувачів та можливих ролей (які представлені на рисунках 4 та 5) та виконали декілька тестових запитів.

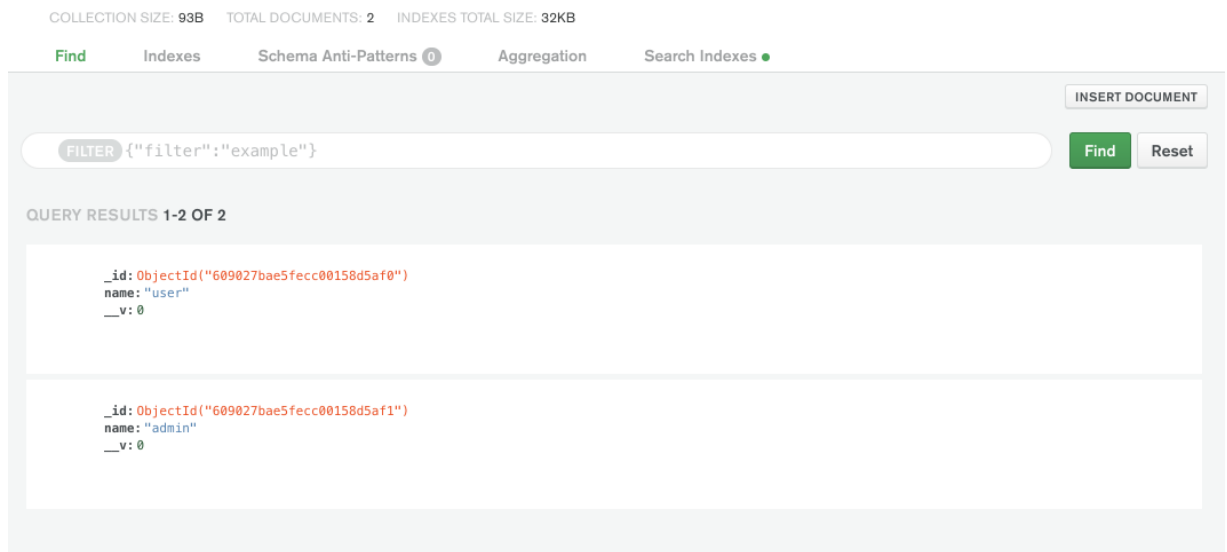


Рисунок 4. Колекція з ролями для користувачів сайту

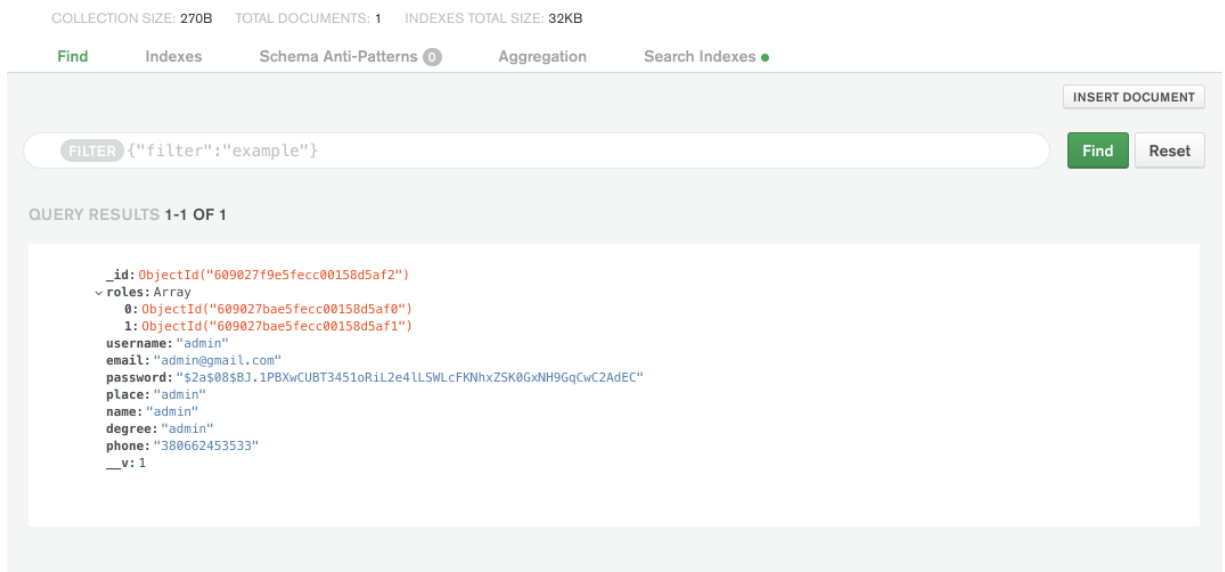


Рисунок 5. Колекція з користувачами для сайту

4.3 Визначення активної сесії для авторизованого користувача

Для авторизації в системі слід передбачити функцію входу до особистого кабінету. Для цього окремо створимо маршрут з POST запитом на `/api/login` де з інтерфейсу буде збиратись інформація про email та пароль користувача вхід якого передбачається. Тіло запиту представлено нижче (див рис. 6)

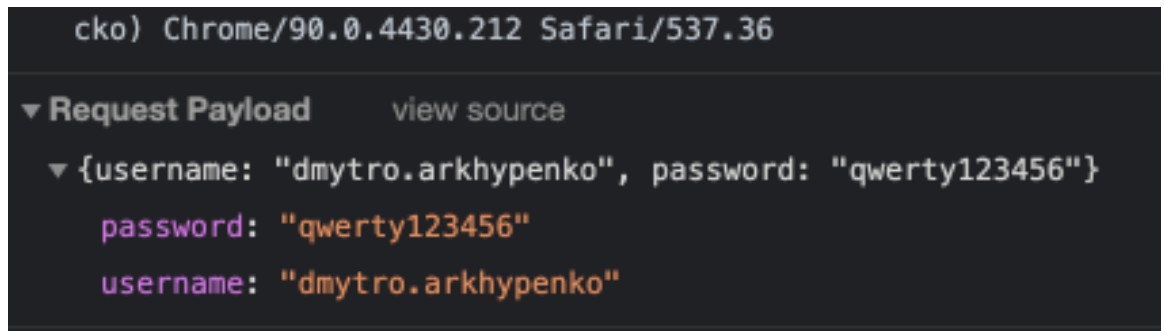


Рисунок 6. Тіло запиту для авторизації в системі

У разі якщо користувач ввів дані вірно то запит поверне код 200 про успішне виконання запиту а також поверне дані авторизованого користувача.

Найпоширенішою практикою серед розробників є використання JWT токена при реалізації механізму авторизації. Після авторизації нам обов'язково необхідно визначити активну сесію, дані повернені з запита ми збережемо в localStorage. localStorage - це об'єкт, який дозволяє сайтам та програмам JavaScript зберігати пари ключ-значення у веб-браузері без терміну дії. Це означає, що дані, що зберігаються у браузері, зберігатимуться навіть після закриття вікна браузера. Однак безстрокове зберігання даних в кеші браузера не є досить безпечно таким чином у відповіді сервера ми додатково генеруємо JWT токен та визначаємо час його експірації до 24 годин після яких користувачеві необхідно повторно авторизуватись в системі [12]. Нижче представлена схема роботи авторизації в рамках сайту (див рис. 7).



Рисунок 7. Схема роботи авторизації на сайті

4.4 Поновлення паролю для входу на сайт

У ході розробки сервера ми врахували можливий розвиток подій коли користувач забув пароль входу в систему та реалізували запит на поновлення паролю. [8]

Процес відбувається наступним чином:

1. Здійснюється POST запит на сервер за маршрутом /reset_password і в тілі запиту передається емейл користувача який забув. Нижче представлий скріншот тіла запиту (див рис. 8)

```
▼ Request Payload    view source
▼ {email: "dmytro.arkhypenko@cation.ee"}
  email: "dmytro.arkhypenko@cation.ee"
```

Рисунок 8. Тіло запиту на відправку листа поновлення паролю

2. Сервер визначає чи є зареєстрований користувач з таким емейлом
3. У разі якщо користувача знайдено ми генеруємо токен для цього користувача, і відправляємо посилання з токеном на пошту користувача.
4. У разі якщо користувача не знайдено сервер поверне повідомлення про помилку

Для відправки листів через сервер Node.js використали бібліотеку NodeMailer. Nodemailer - це модуль для додатків Node.js, який дозволяє легко надсилати електронні листи. Проект розпочався ще в 2010 році, коли не було розумної опції надсилання електронних повідомлень, сьогодні це рішення, до якого більшість користувачів Node.js звертаються за замовчуванням. – Нодмейлер.

Після додавання бібліотеки на сервер ми створили транспортер, який насправді є обліковим записом, що надсилає посилання електронної пошти для скидання пароля.

І в останню чергу коли користувача з поштою знайдено внутрішніми функціями Nodemailer-а нам вдалося відправити лист на пошту користувача. [10]

Лист виглядає наступним чином (див рис. 9)



mydev1488@gmail.com [через sendgrid.net](#)

кому: я ▾



Для поновлення пароля на сайті конференції перейдіть за посиланням та створіть новий пароль

Посилання для поновлення пароля: [Натисніть тут](#)



Ответить



Переслать

Рисунок 9. Лист поновлення паролю входу на сайт

Після реалізації запиту на відправку листа з посиланням на поновлення паролю ми також імплементували форму яка буде відправляти POST запит на маршрут `/api/new_password` з тілом запиту яке включає в себе значення нового паролю для оновлення в базі даних та токена користувача для якого необхідно оновити пароль. Тіло запиту оновлення паролю представлено нижче (див рис. 10)

```
▼ Request Payload    view source
▼ {password: "qwerty123", token: "e26a90e158b7c36d86e0727073538fecfa52983dd9e7d1464183c5ed82e65f98"}
  password: "qwerty123"
  token: "e26a90e158b7c36d86e0727073538fecfa52983dd9e7d1464183c5ed82e65f98"
```

Рисунок 10. Тіло запиту на зміну паролю

В цьому розділі ми розглянули повний цикл процесу авторизації з допоміжними запитамі для поновлення паролю в системі

5. УПРАВЛІННЯ КОНТЕНТОМ ТА ЗАВАНТАЖЕНИМИ НА СЕРВЕР ФАЙЛАМИ

5.1 Бібліотека Multer. Робота з файловою системою сервера

Сьогодні програми стають привабливішими, коли включають функціональність зберігання та використання таких файлів, як зображення, музика, файли документів тощо. Платформа Node.js має доступ до бібліотеки Multer, яка обробляє завантаження файлів. Multer - це проміжне програмне забезпечення, яке перехоплює запит POST завантаження файлу на сервер і працює з файлом різними способами, перш ніж дозволити серверу надіслати свою відповідь клієнту. Деякі способи роботи Multer з файлами - це збереження даних у файловій системі або навіть перевірка файлу, перевірка відповідності певним критеріям, таким як розмір файлу або тип файлу, а також багато інших.

Для початку, ми повинні встановити бібліотеку в проект.

Multer - це пакет npm, який полегшує обробку файлів, що завантажуються. Це робить це дуже ефективно, тому є досить популярним. Згідно ТЗ нам необхідно реалізувати окремі сторінки для перегляду матеріалів конференції, а саме:

1. Фотографії
2. Архіви робіт
3. Презентації з минулих років

Щоб реалізувати файлові менеджери в рамках веб сайту слід уніфікувати запити для цієї задачі та перевикористовувати їх в залежності від типу завантажуваного файлу.

5.2 Запити файлових контролерів веб сайту

Для завантаження файлів на сервер гарною практикою є визначення папки resources для розміщення завантажених або статичних фалів там.

В файловій системі проекту я створив папку та додатково створив ще папки під кожен з типів файлів (див рис. 11)



Рисунок 11. Файлова архітектура ресурсів

Для завантаження файлів ми створили запити POST для трьох маршрутів /api/upload/asset – для завантаження картинок, /api/upload/pdf – для завантаження архівів робіт та /api/upload/pptx для завантаження презентацій на сервер

В якості тіла запиту є об'єкт FormData що дозволяє цілком передати файл з клієнта на сервер на контролер сервера і вже там за допомогою бібліотеки Multer зберегти його до відповідної дочірньої папки ресурсів.

На зображенні приведено приклад тіла запиту на /api/upload/asset для завантаження картинки на сервер. Тіло запиту представлено нижче (див рис. 12)



Рисунок 12. Тіло запиту завантаження файла

Для отримання всіх файлів з сервера створимо відповідні GET запити для кожного з трьох типів файлів за маршрутами /api/files/assets – для отримання картинок, /api/files/pptx – для отримання презентацій, /api/files/pdf – для отримання картинок. [11]

Для демонстрації результату роботи запиту ми виконали тестові запити завантаження файлів типу pdf та далі протестували GET запит по маршруту /api/files/pdf в результаті отримали у відповіді запиту файли в форматі об'єктів з двома атрибутами: назва файлу, посилання на файл що знаходиться на сервері для подальшого відображення файлу при розробці користувацького інтерфейсу.

Нижче на скріншоті наведено приклад роботи GET запиту для отримання всіх завантажених файлів на сервер типу pdf (див рис 13)

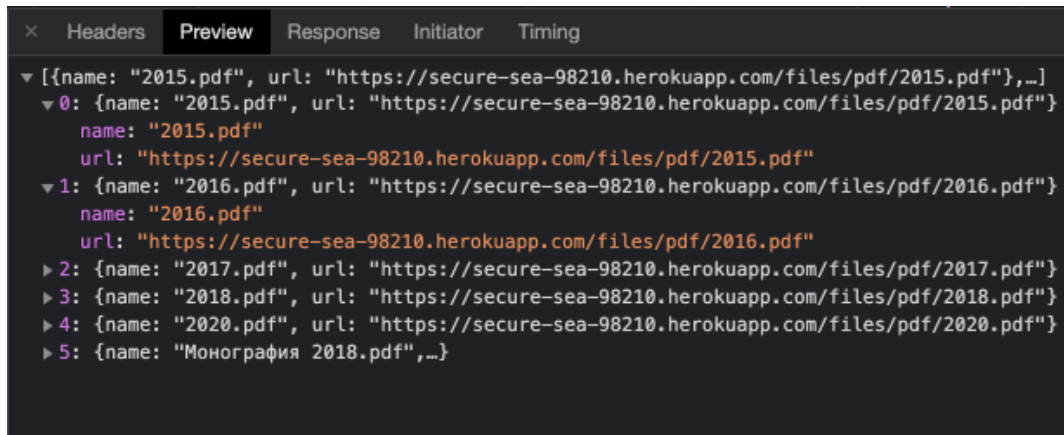


Рисунок 13. Відповідь сервера на отримання файлів типу pdf

6. ІМПЛЕМЕНТАЦІЯ СЕРВЕРНОЇ ЧАСТИНИ ДЛЯ КОРИСТУВАЦЬКОГО ЧАТУ

Розробка чату передбачає використання двонаправленого каналу між сервером та клієнтом. Двонаправлені канали зв'язку в реалізуються з використанням WebSockets. [3]

WebSocket – це протокол з'єднання між клієнтом та сервером. WebSockets забезпечують двоспрямований повнодуплексний канал зв'язку, який працює через HTTP через одне з'єднання TCP / IP. По суті, протокол WebSocket сприяє передачі повідомлень між клієнтом та сервером.

На основі WebSockets розробники створили бібліотеку Socket.io яка надає можливість швидко та легко створювати двоспрямовані сервіси які на клієнті відображаються як окремі компоненти і вони автоматично оновлюються у кожного

підключеного до клієнта користувача коли в рамках сокета виконується будь яка подія.

Динамічного оновлення контейнера сервісу ми на сервері визначили основні події які будуть відображатись після того як хоча б у 1 з декількох на екрані користувача (див рис. 14)

Вікно контейнера чату буде оновлюватись та засобами розробки інтерфейсу ми зможемо відобразити на екрані кожного з підключених користувачів наступні події:

1. Підключення користувача до чату
2. Відправка нового повідомлення
3. Користувач почав писати повідомлення
4. Користувач перестав писати повідомлення
5. Від'єднання користувача від чату

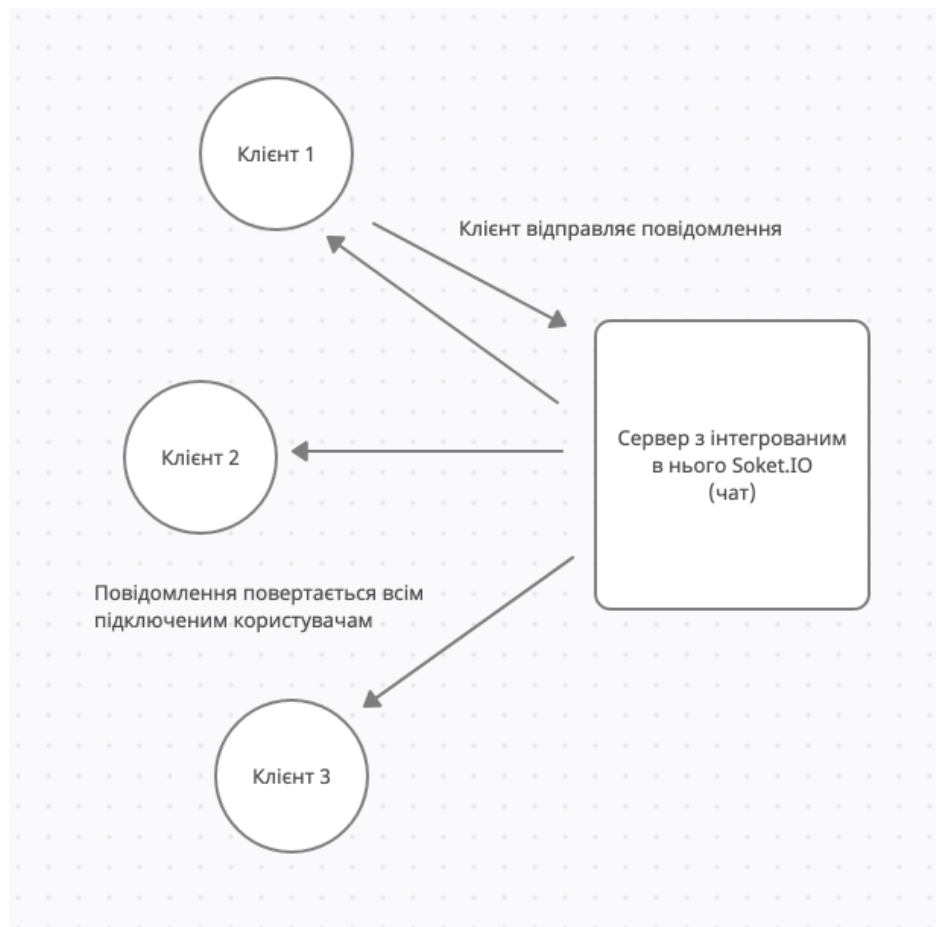


Рисунок 14. Схема роботи сервісу з використанням Socket.io

7. РОЗРОБКА КОРИСТУВАЦЬКОГО ІНТЕРФЕЙСУ ДЛЯ ВЕБ САЙТУ

7.1 Аналіз серверної частини та визначення сторінок сайту

Якщо повторно ознайомитись з усіма можливостями сервера розробленого з використанням платформи Node.js та додаткових бібліотек які нам знадобилися у ході розробки нам варто визначитись з необхідними сторінками та їх наповненням для гарного користувацького досвіду як звичайних користувачів так і адміністраторів веб сайту [15].

Перш за все визначимо контент який доступний для всіх включаючи незареєстрованих користувачів:

1. Головна
2. Архіви робіт
3. Презентації
4. Галерея
5. Реєстрація для участі в конференції
6. Сторінка реєстрації на сайті

Для зареєстрованих користувачів необхідно додатково створити такі сторінки:

1. Авторизація в системі
2. Профіль
3. Забув пароль
4. Поновлення паролю за посиланням у листі
5. Користувацький чат

Для адміністратора також необхідно додати адмін панель використовуючи яку він матиме можливість додавати файли на сайт та переглядати інформацію про зареєстрованих користувачів

Слід також зазначити що на сайті є статичні повторювані блоки такі як заголовок з картинкою та меню і нижній колонтитул з контактами та меню для

швидкої навігації на сайті.

7.2 Розробка статичних секцій сайту

Заголовок сайту складається з 2 основних компонентів HomeHero – картинка з назвою та Menu яке містить посилання на інші сторінки вебсайту.

Нижче на картинці представлена реалізація даної секції (див рис 15)

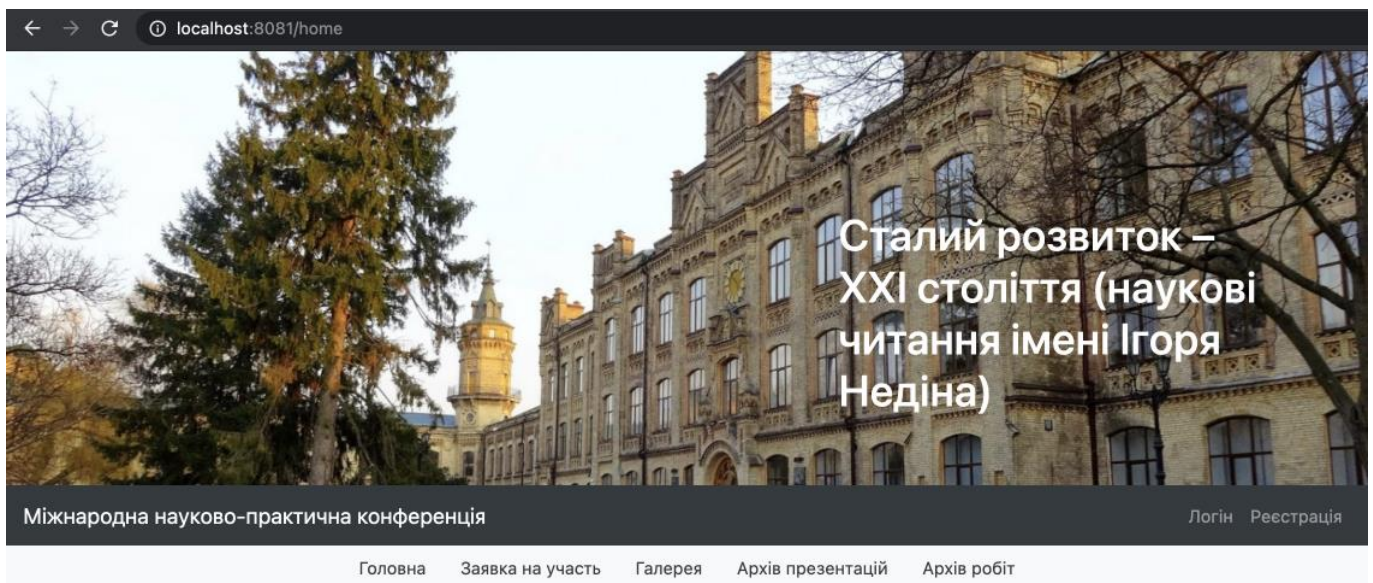


Рис. 15

Рисунок 15. Заголовок веб сайту

Нижній колонтитул має містити також 2 компоненти Menu та Contacts – розділ з контактними особами конференції

Нижче представлена реалізація секції нижнього колонтитула (див рис. 16)

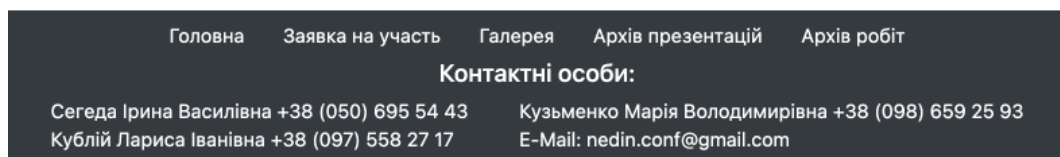


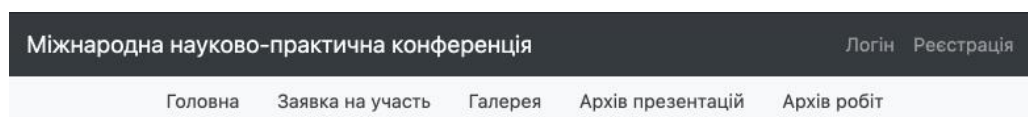
Рисунок 16. Нижній колонтитул сайту

7.3 Формування контенту для головної сторінки

На головній сторінці ми виділю наступні секції:

1. Про головне
2. Напрями роботи конференції
3. Об'яви
4. Умови
5. Організаційний внесок
6. Вимоги
7. Програма конференції
8. Організаційний комітет
9. Науковий комітет

Деякі з них освітлені на скріншотах нижче (див рис 17-18)



Про головне



Недін Ігор Валентинович (1944 – 2008) Доктор економічних наук, професор "Викладач-дослідник", "Заслужений енергетик України". Нагороджений урядовим нагрудним знаком "Петро Могила", медалями ВДНГ СРСР і УРСР. Під керівництвом І.В. Недіна було створено наукову школу з проблем аналізу умов забезпечення сталого розвитку й економічної безпеки територіально-виробничих систем.

Шановні колеги!

Національний університет «**Києво-Могилянська академія**» та Теплоенергетичний факультет Національного технічного університету України «**Київський політехнічний інститут імені Ігоря Сікорського**» запрошують Вас взяти участь у роботі

Мета конференції: об'єднати науковців та широке коло фахівців для обговорення актуальних проблем сталого розвитку в XXI столітті, встановлення контактів між вченими, обмін практичними результатами наукових досліджень.

Напрями роботи конференції:

- Сталий розвиток держави: теорія, методологія, механізми забезпечення
- Реалізація принципів сталого розвитку в забезпеченні економічної та екологічної безпеки держави

Рисунок 17 Частина 1. контенту головної сторінки

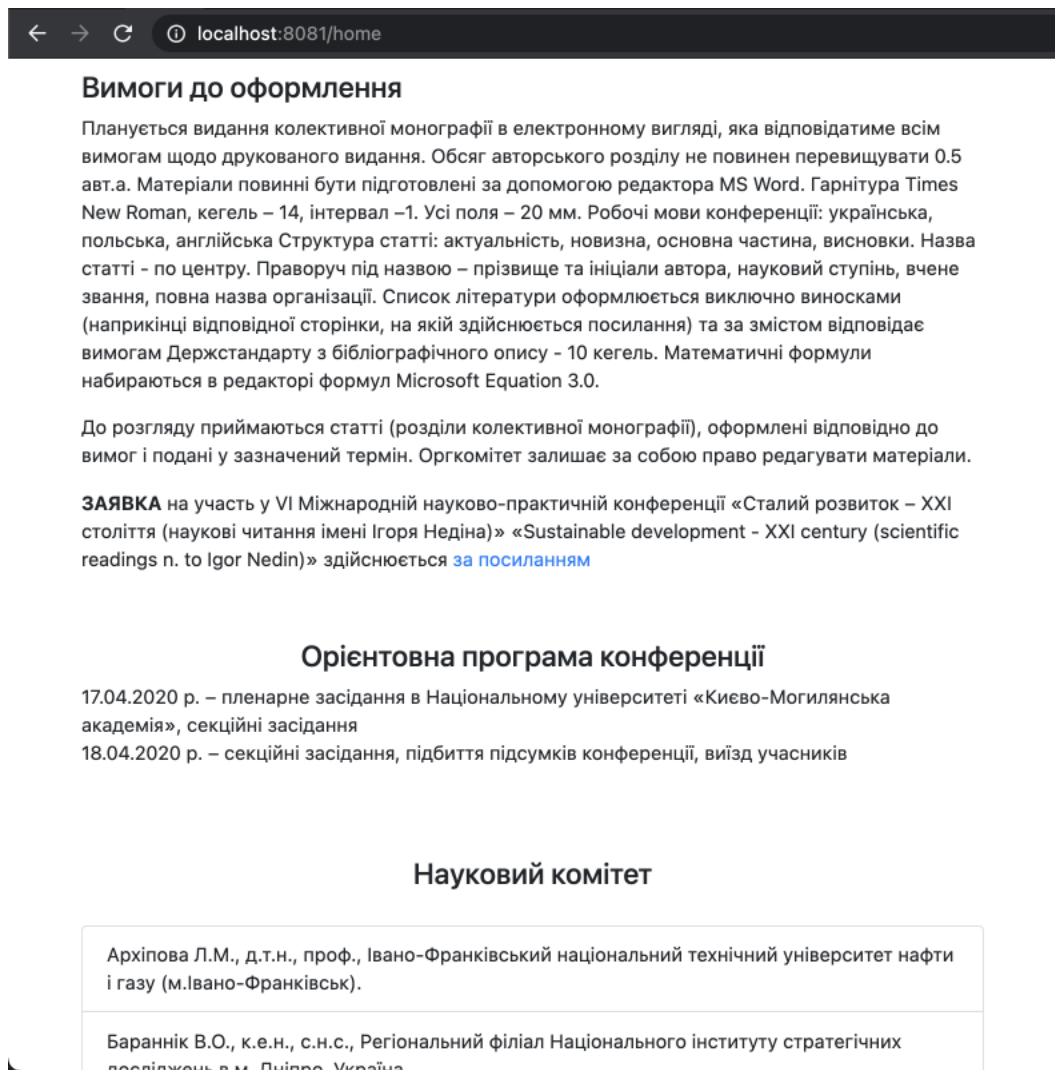
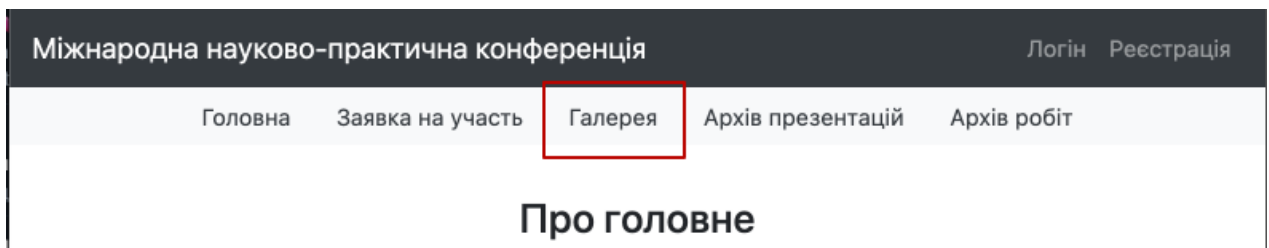


Рисунок 17. Частина 2. контенту головної сторінки

7.4 Інтерфейс сторінки фотогалереї

За посиланням в меню галерея ми можемо переглядати фотозвіти з минулорічних конференцій які є завантаженими на сервер та за допомогою React.js. Посилання на сайті представлено на скріншоти (див рис. 19)



Рисунко 18. Посилання Галерея

При переході за посиланням клієнт здійснює запит на отримання картинок і динамічно рендеримо отримані з сервера картинки на нашій сторінці (див рис 20)

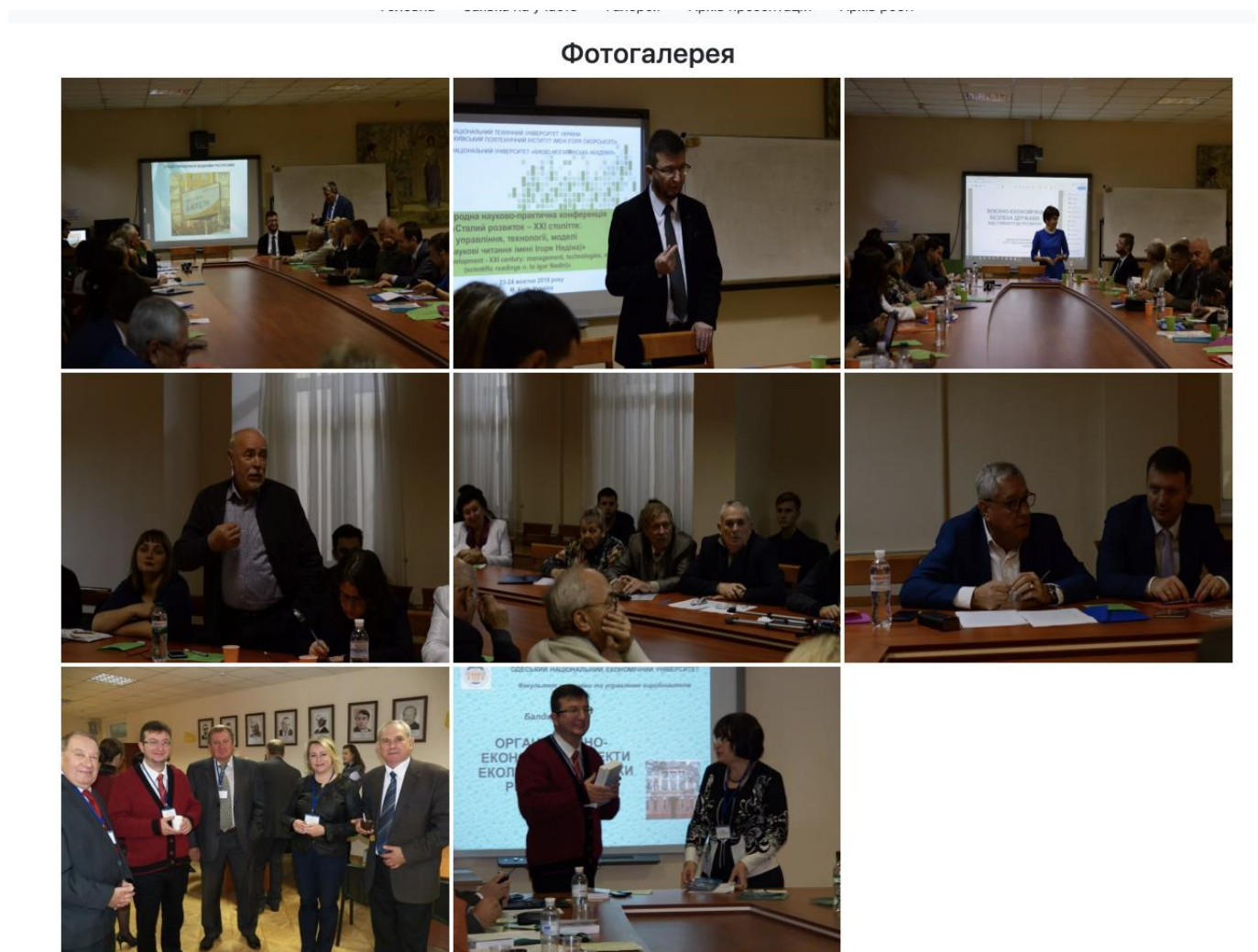


Рисунок 20. Сторінка з фотогалереєю

7.5 Інтерфейс сторінок презентацій та архівів робіт

Презентації та архіви робіт завантажуються таким же чином як і картинки, при переході за посиланнями Архівів або Презентацій. Посилання презентовані на ілюстрації нижче (див рис. 21)

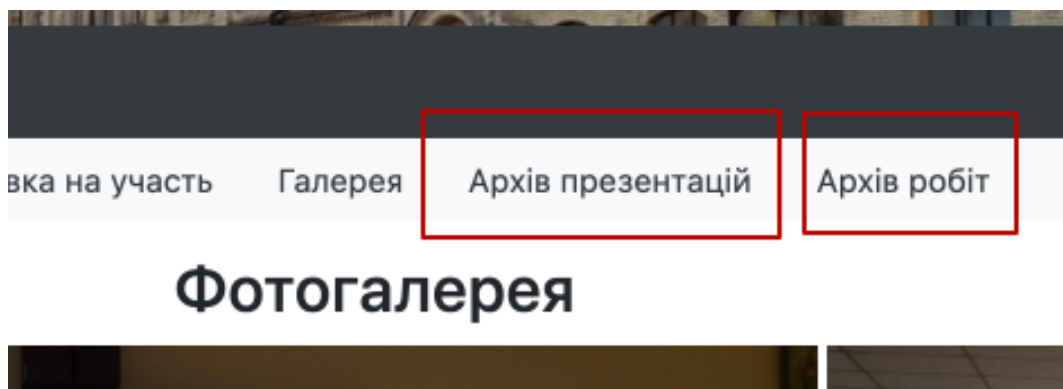


Рисунок 21. Посилання на сторінки Презентацій та Архівів робіт

Виконується запит на сервер на отримання файлів і після завантаження, сторінка має такий вигляд (див рис. 22, 24)

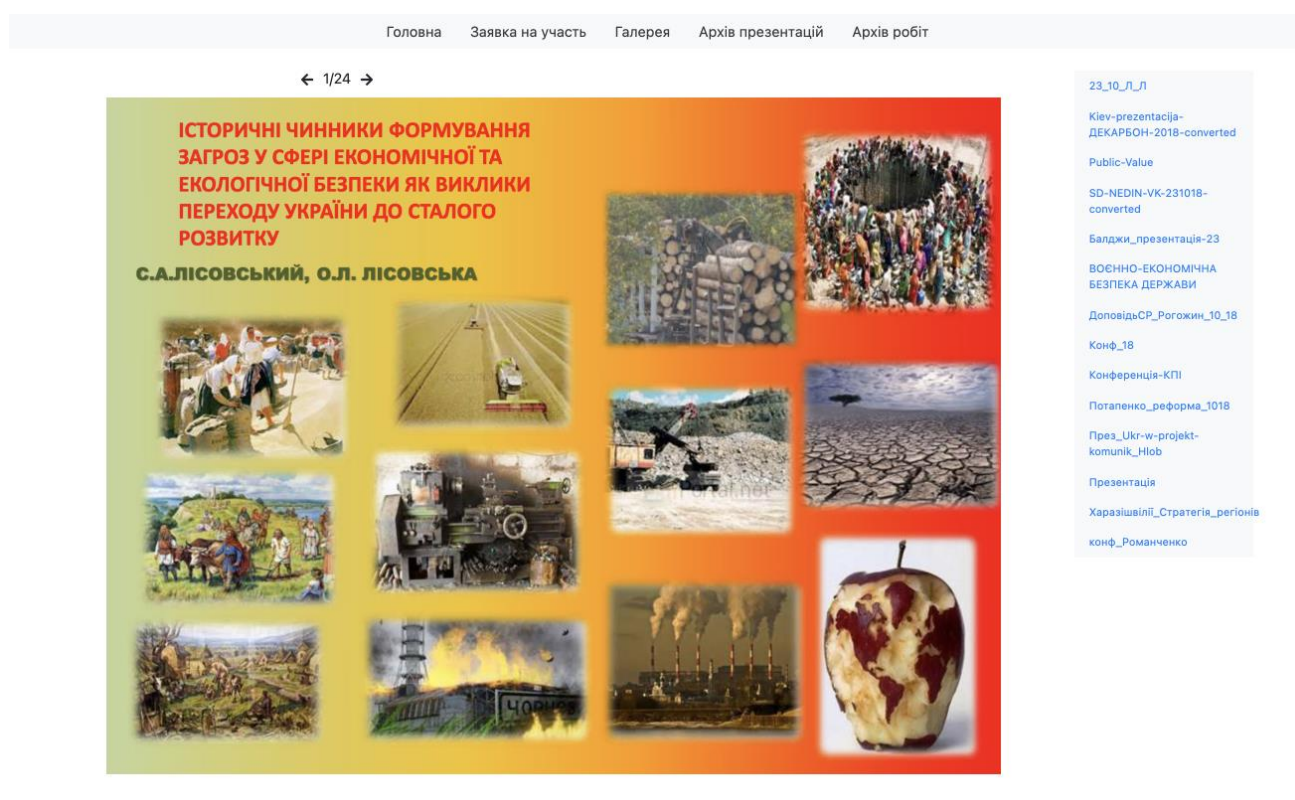


Рисунок 22. Сторінка Презентацій на веб сайті

Окрім сторінки з файлу ми можемо користуватись контролером для гортання файлу (див рис. 23)

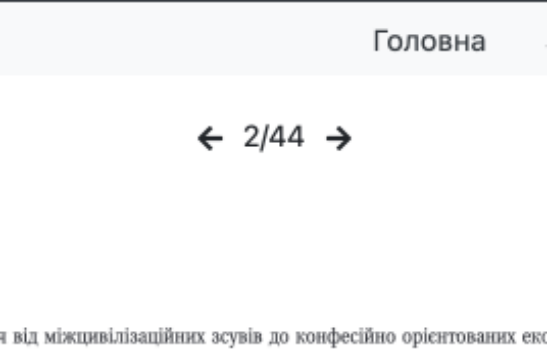


Рисунок 23. Контролер для гортання сторінок

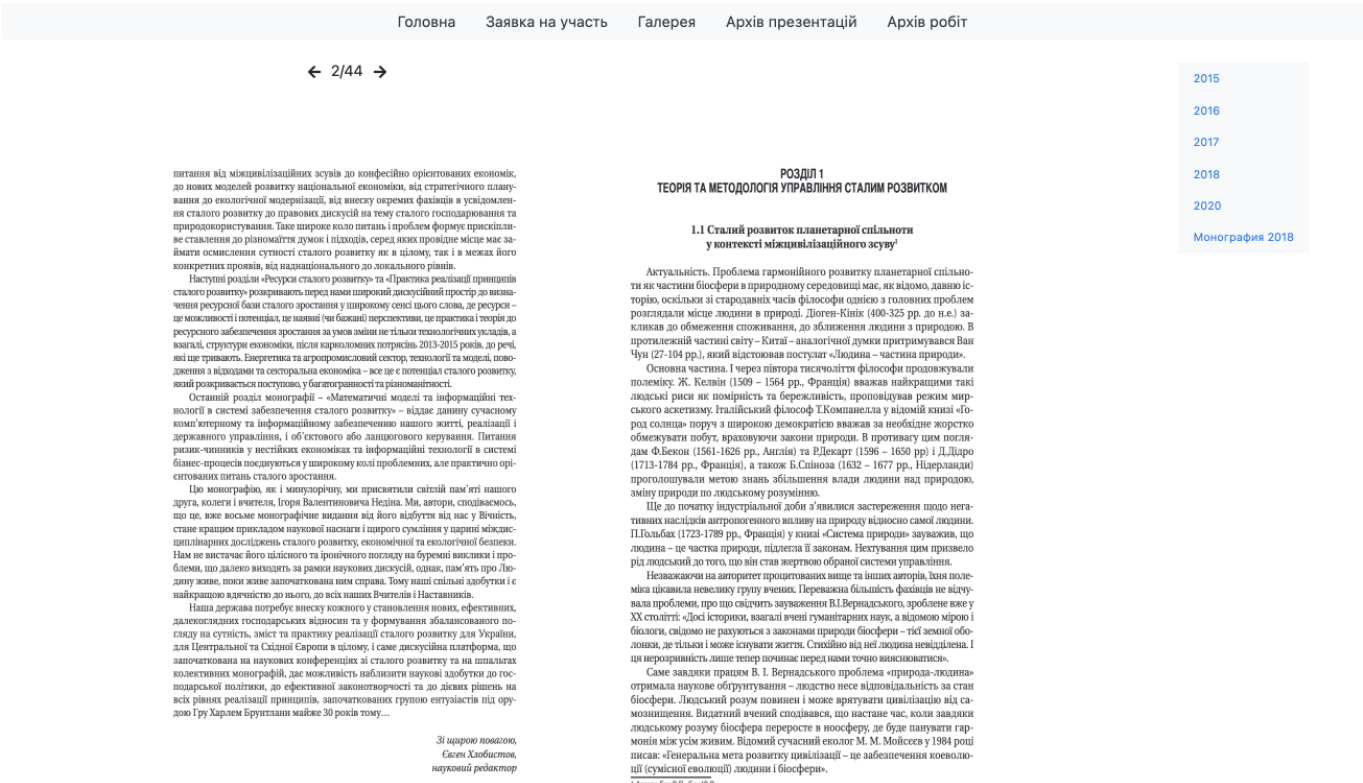


Рисунок 24. Сторінка архівів з роботами

А також за допомогою бокового меню можна перемикатись між файлами для перегляду. При натисканні на файл до перегляду встановиться обраний користувачем файл (див рис. 25)

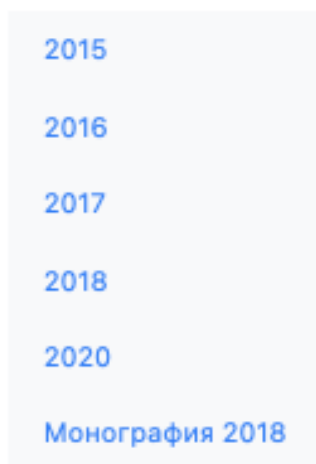


Рисунок 25. Меню навігації по завантаженим файлам

7.6 Заявка на участь

Для виконання запиту на реєстрацію ми на сервері додали ще 1 POST запит який на пряму взаємодіє з транспортером, котрий ми створювали для відправки листів з посиланням на поновлення паролю до власного кабінету.

На сайті ми додали сторінку “Заявка на участь” при переході на яку ми створили форму подання заявки (див рис. 26)

Головна

Заявка на участь

Галерея

Архів презентацій

Архів робіт

Прізвище, ім'я, по-батькові (повністю) авторів (усіх авторів вписувати в це поле)

Наукова ступінь (напр. бакалавр)

Посада

Місце роботи/навчання (напр. Київський національний університет імені Тарса Шевченка)

Контактний номер телефону

Email

Тема наукової роботи

Напрями роботи конференції

Реалізація принципів сталого розвитку в забезпеченні економічної та екологічної безпеки держави

Подати заявку

Рисунок 26. Форма подання заявки на участь

До форми додано валідацію відповідних даних а також всі поля з форми є обов'язковими і користувач не зможе відправити недозаповнену форму. Демонстрація валідації представлена нижче (див рис. 27)

Email

dmytro.arkhyenko1@gmail.com

Тема наукової роботи

Поле є обов'язковим!

Напрями роботи конференції

Реалізація принципів сталого розвитку в забезпеченні економічної та екологічної безпеки держави

Подати заявку

Рисунок 27. Демонстрація валідації даних

У разі правильності вводу даних та успішної відправки форми користувач отримає повідомлення (див рис. 30)

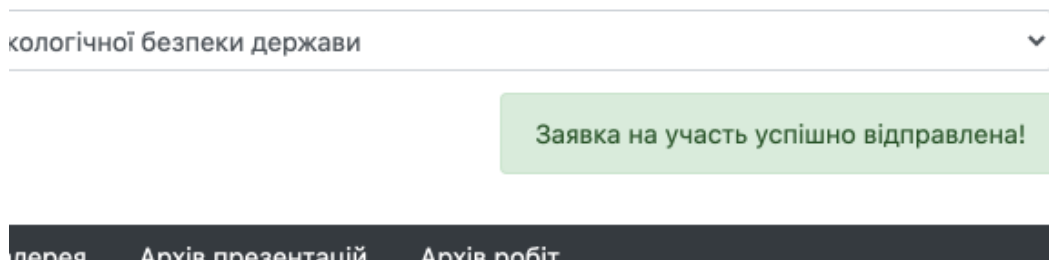


Рисунок 30. Повідомлення про успішно подану заявку

7.7 Створення інтерфейсу для процесу авторизації на сайті

Спочатку створимо форму реєстрації нового користувача. В меню додали посилання для реєстрації. Посилання представлено нижче (див рис. 31)

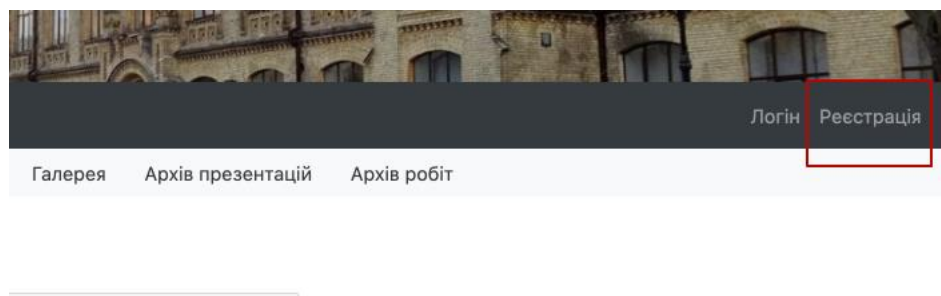
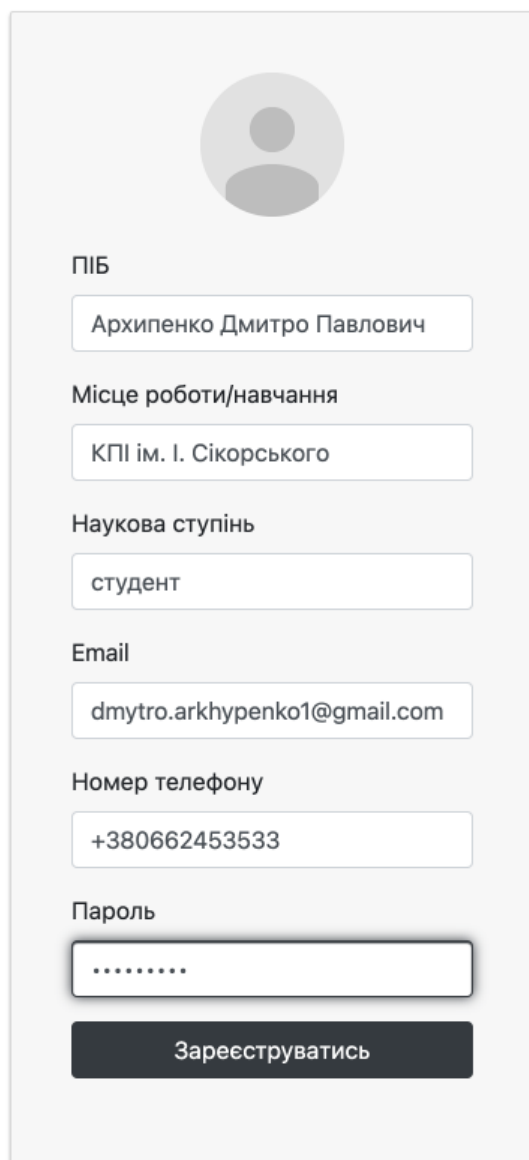


Рисунок 31. Посилання для реєстрації

І на сторінці створили форму під запит сервера описаний у 4 розділі. Нижче на картинці ілюстрована форма створена на веб сайті (див рис. 32)



Registration form on a website. At the top is a circular placeholder for a profile picture. Below it are several input fields with labels in Ukrainian: "ПІБ" (Full Name), "Місце роботи/навчання" (Place of work/study), "Наукова ступінь" (Degree), "Email", "Номер телефону" (Phone number), and "Пароль" (Password). Each field contains a sample value. At the bottom is a dark button labeled "Зареєструватись" (Register).

ПІБ

Архипенко Дмитро Павлович

Місце роботи/навчання

КПІ ім. І. Сікорського

Наукова ступінь

студент

Email

dmytro.arkhypenko1@gmail.com

Номер телефону

+380662453533

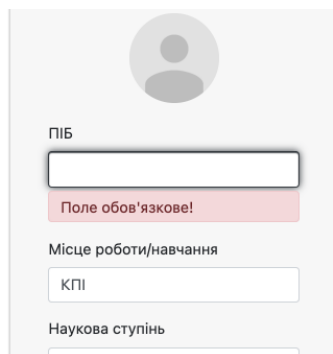
Пароль

.....

Зареєструватись

Рисунок 32. Форма реєстрація на сайті

До полів додано відповідну валідацію а також всі поля форми є обов'язковим (див.рис 33)



Close-up of the registration form showing validation. The "ПІБ" field is empty and has a red error message "Поле обов'язкове!" (Field is required!) below it. The "Місце роботи/навчання" field contains "КПІ" and the "Наукова ступінь" field is partially visible.

ПІБ

Поле обов'язкове!

Місце роботи/навчання

КПІ

Наукова ступінь

Рисунок 33. Валідація даних форми реєстрації

Для входу до особистого кабінету зареєстрованому користувачу потрібно перейти за посиланням Логін в меню та ввести свої дані від аккаунту. Посилання ілюстроване нижче (див рис 34)

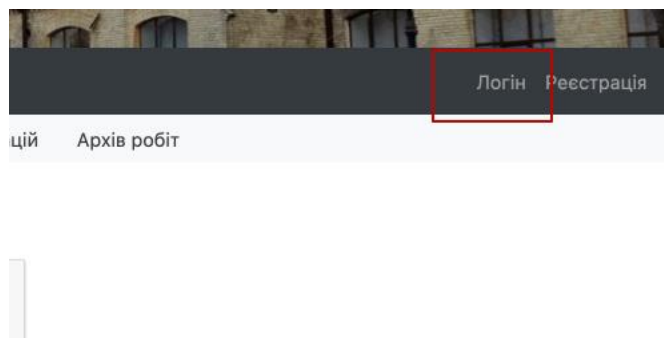
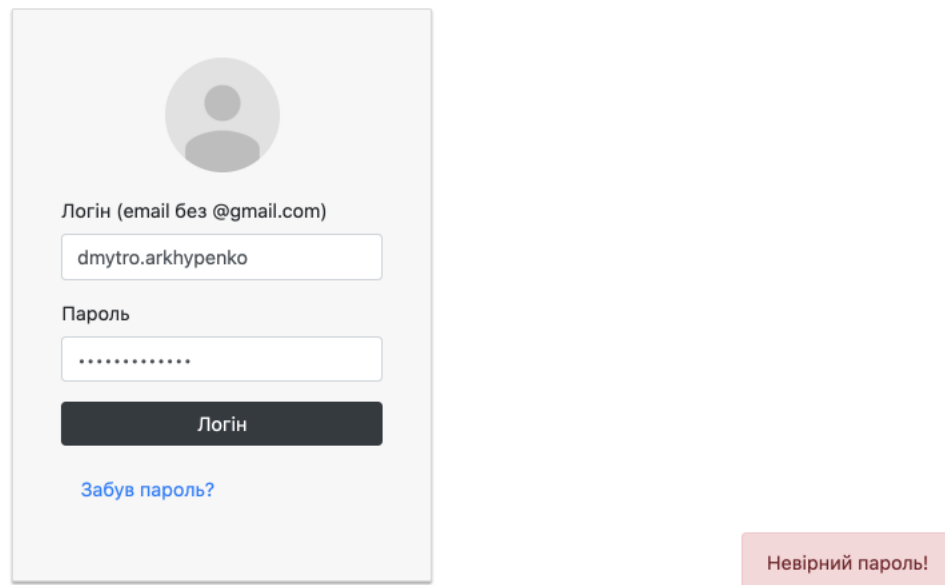


Рисунок 34. Посилання на вхід до особистого кабінету
Реалізація форми авторизації в системі виглядає наступним чином (див рис. 35)

A login form for a website. It features a light grey background. At the top, there is a circular placeholder for a user profile picture. Below this, the text 'Логін (email без @gmail.com)' (Login (email without @gmail.com)) is displayed. Underneath, there is a text input field containing the email address 'dmytro.arkhypenko1'. Below the email field, the text 'Пароль' (Password) is displayed. Underneath, there is a password input field with masked characters '.....'. Below the password field, there is a dark grey button with the text 'Логін' (Login). At the bottom, there is a blue link that says 'Забув пароль?' (Forgot password?).

Рисунок 35. Форма логіну для сайту

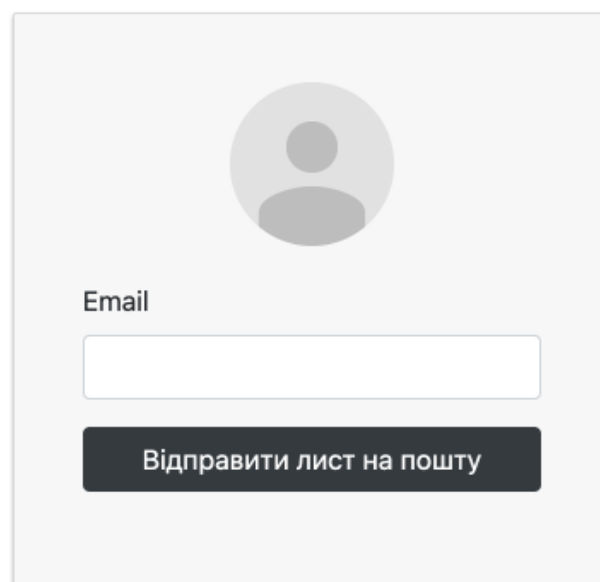
При введенні невірних даних сервер поверне помилку та користувач отримає повідомлення на дисплеї (див рис. 36)



The image shows a login form with a light gray background. At the top is a circular placeholder for a profile picture. Below it, the text 'Логін (email без @gmail.com)' is followed by a text input field containing 'dmytro.arkhypenko'. Underneath is the label 'Пароль' followed by a password input field with masked characters. A dark gray button labeled 'Логін' is positioned below the password field. A blue link 'Забув пароль?' is located at the bottom left of the form. To the right of the form, a pink rectangular box contains the text 'Невірний пароль!'.

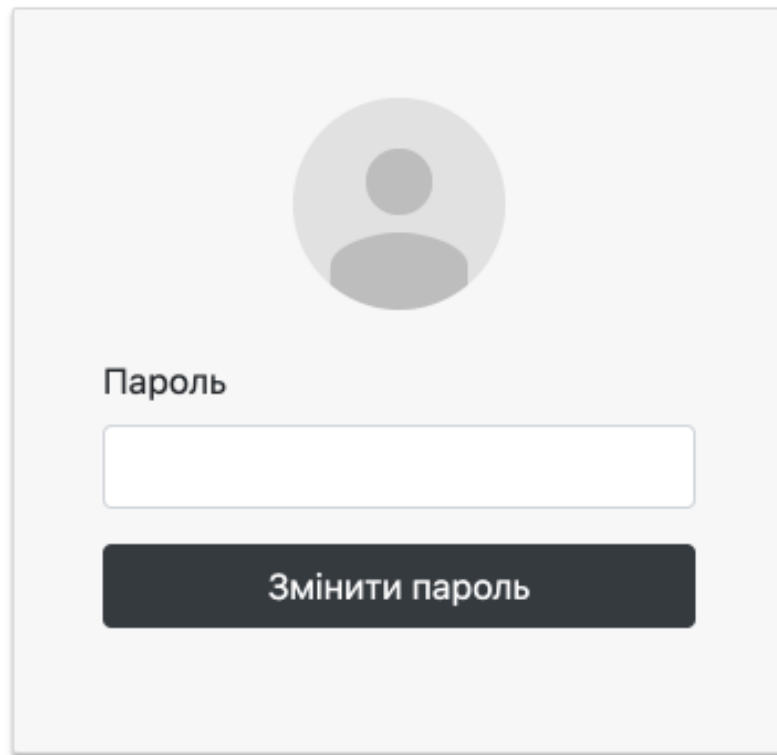
Рисунок 36. Помилка авторизації

Нижче представлені 2 форми для поновлення (див рис. 37) та для створення нового пароля (див рис. 38) у разі якщо користувач забув його



The image shows a form for password reset. It has a light gray background and a circular profile picture placeholder at the top. Below the placeholder is the label 'Email' followed by an empty text input field. At the bottom of the form is a dark gray button with the text 'Відправити лист на пошту'.

Рисунок 37. Форма відправки листа для поновлення паролю



Пароль

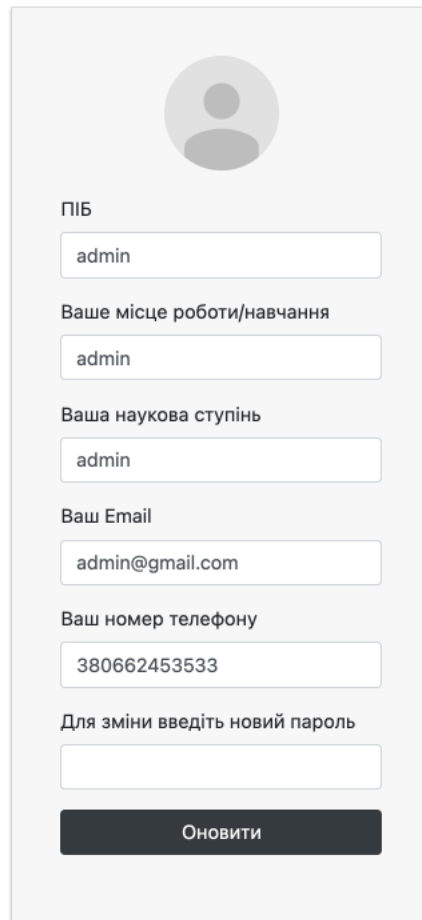
Змінити пароль

Рисунок 38. Форма створення нового паролю

Після оновлення паролю необхідно повторно увійти в систему через посилання “Логін”.

Якщо вхід до особистого кабінету успішний, то в залежності від ролі користувача для адміністратора відкриється доступ до Сторінки адміністратора, Профілю та Користувацького чату, а для звичайного користувача тільки до Профілю та Користувацького чату.

Сторінка профілю має призначення відображати інформацію про користувача та надавати змогу редагувати дані. Таким чином ми реалізували ще 1 форму, однак вже з актуальними даними однак при редагуванні та натисканні на кнопку “Оновити” користувач може змінити дані введені при реєстрації (див рис. 39)



A user profile form with a light gray background. At the top is a circular placeholder for a profile picture. Below it are several input fields, each preceded by a label: 'ПІБ' (Name) with 'admin', 'Ваше місце роботи/навчання' (Work/Study place) with 'admin', 'Ваша наукова ступінь' (Degree) with 'admin', 'Ваш Email' (Email) with 'admin@gmail.com', and 'Ваш номер телефону' (Phone number) with '380662453533'. There is also an empty input field for a new password with the label 'Для зміни введіть новий пароль'. At the bottom is a dark gray button with the text 'Оновити' (Update).

Рисунок 39. Форма профілю користувача

7.8 Користувацький чат

На сервері ми вже створили real-time service з функціями оновлення контенту для підключених користувачів, тепер на сторінці “Користувацького чату” створили контейнер для повідомлень та зверстали засобами HTML & CSS вигляд для різних повідомлень, повідомлення авторизованого користувача (див рис. 41) та повідомлення іншого користувача (див рис. 40)



Рисунок 40. Повідомлення іншого користувача

16:12 0.4

Як справи?

Рисунок 41. Повідомлення авторизованого користувача

Додавши форми вводу та відправки повідомлень наш користувацький чат виглядає наступним чином (див рис. 42)

admin

Всім привіт!

16:10 0.4

16:12 0.4

Як справи?

Надіслати

Рисунок 42. Користувацький чат на веб сайті

7.9 Адмін панель вебсайту

Якщо авторизований користувач має роль адміністратора тоді він матиме доступ до посилання “Сторінка адміністратора” де передбачені наступні функції:

1. Перегляд завантажених файлів на сайт
2. Перегляд інформації про зареєстрованих користувачів
3. Можливість завантажувати нові файли
4. Можливість видаляти старі файли

В рамках управління файлами створено 3 однакові за структурою та механізмом роботи файлові менеджери які виглядають наступним чином (див рис. 43)

Менеджер фото галереї

Выберите файл

Файл не выбран

Завантажити

Список завантажених фото

09012019tef1.jpg	Видалити
09012019tef2.jpg	Видалити
09012019tef3.jpg	Видалити
09012019tef4.jpg	Видалити
09012019tef5.jpg	Видалити
09012019tef6.jpg	Видалити
b93fb2_5eee2731174643ddbd9e40590e6ad6c7_mv2.jpg	Видалити
b93fb2_bb036948ae16477f8d3f68514f21c0c5_mv2.jpg	Видалити

Менеджер архівів з роботами

Выберите файл

Файл не выбран

Завантажити

List of Files

Рисунок. Менеджер фото галереї

При натисканні на Вибір файлу відкривається файлова система комп'ютера та адміністратор може обрати файл з власного ПК для завантаження (див рис. 44)

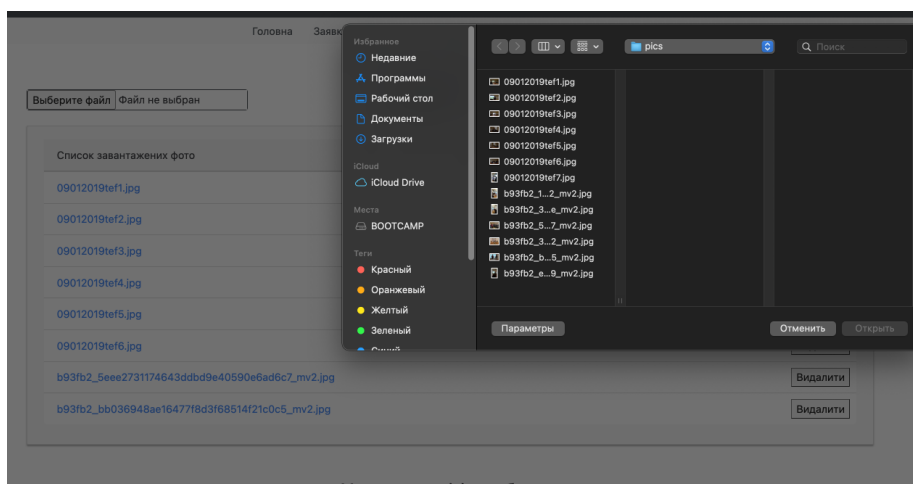


Рисунок 44. Активна відкрита файлова система для завантаження файлів

Навпроти кожного файлу є кнопка “Видалити” для видалення файлу з сервера. (див рис. 45)

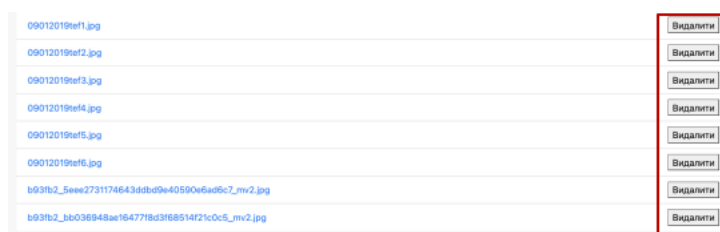


Рисунок 45. Видалення завантажених файлів на сервер

Також в нижній частині сторінки передбачено таблицю з зареєстрованими користувачами для того щоб адміністратор міг переглядати та відстежувати контакти зареєстрованих користувачів (див рис 46)

Інформація про зареєстрованих користувачів сайту				
ПІБ	Навчальний заклад	Наукова ступінь	Номер телефону	Емейл
admin	admin	admin	380662453533	admin@gmail.com
user	user	user	0662453533	dmitry.arkhipenko58@gmail.com
Архипенко Дмитро Павлович	КПІ ім. І. Сікорського	студент	+380662453533	dmytro.arkhypenko1@gmail.com
Архипенко Дмитро Павлович	КПІ	Студент	+380662453533	dmytro.arkhypenko@cation.ee

Рисунок 46. Таблиця з зареєстрованими користувачами

На стороні сервера у файловій системі ми маємо наступну структуру по завантаженим файлам (див рис. 47)

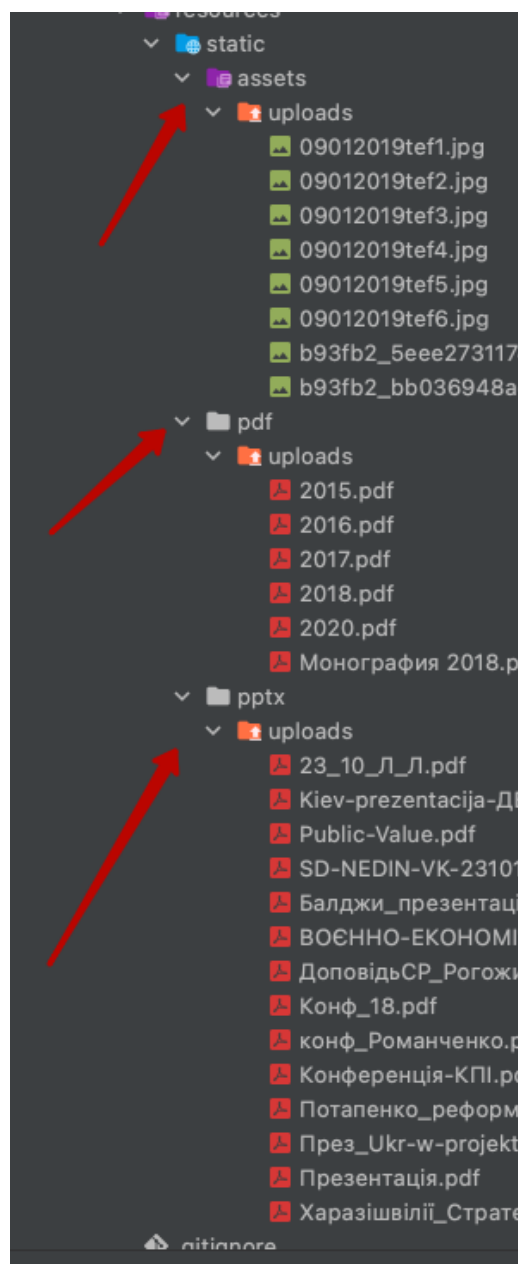


Рисунок 47. Файлова система ресурсів

ВИСНОВКИ

Метою даної дипломної роботи було створення веб сайту для організації та супроводження конференції.

Розроблено програмне забезпечення на платформі WebStorm 2021 з використанням JavaScript, React.js Node.js та підтримкою системи керування базами даних MongoDB.

База даних в рамках дипломної роботи передбачає збереження даних про зареєстрованих користувачів та учасників поточної або минулих конференцій.

З використанням веб сервіса SendGrid для автоматизованої відправки листів реалізовано функції поновлення паролю та отримання нових заявок на участь у наступній конференції.

Для вирішення проблеми комунікації з учасниками та з метою уникнення дублювання інформації розроблено користувацький чат з використанням бібліотеки Soket.Io в якому будь хто з зареєстрованих користувачів має змогу прочитати останні новини або ж задати питання які його цікавлять.

Також програмний комплекс пропонує перегляд архівів робіт, презентацій а також фотографій з минулорічних конференцій.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Кайл Сімпсон You don't know JS 2014 [Електронний ресурс] — Режим доступу: <https://github.com/azat-io/you-dont-know-js-ru>
2. React Javascript бібліотека для створення користувацьких інтерфейсів [Електронний ресурс] — Режим доступу: <https://uk.reactjs.org/>
3. Socket.io офіційна документація бібліотеки [Електронний ресурс] — Режим доступу: <https://socket.io/>
4. Node.js офіційна документація платформи [Електронний ресурс] — Режим доступу: <https://nodejs.org/en/>
5. MongoDB офіційна документація СУБД [Електронний ресурс] — Режим доступу: <https://docs.mongodb.com/>
6. MongoDB atlas хмарне сховище для деплою баз даних [Електронний ресурс] — Режим доступу: <https://www.mongodb.com/cloud/atlas>
7. Heroku хмарне сховище для експлуатації серверних ресурсів [Електронний ресурс] — Режим доступу: <https://id.heroku.com/login>
8. Сервіс для надсилання листів Sendgrid [Електронний ресурс] — Режим доступу: <https://sendgrid.com/>
- 9 NPM сховище великої кількості бібліотек для Javascript-a [Електронний ресурс] — Режим доступу: <https://www.npmjs.com/>
10. Nodemailer бібліотека JS для створення транспортних каналів листування [Електронний ресурс] — Режим доступу: <https://nodemailer.com/about/>
11. Multer бібліотека для реалізації навігації по файловій системі сервера на node.js [Електронний ресурс] — Режим доступу: <https://github.com/expressjs/multer>
12. JWT офіційна документація по токенам [Електронний ресурс] — Режим доступу: <https://jwt.io/>
13. Auth0 найпоширеніша бібліотека авторизації для клієнт серверних додатків [Електронний ресурс] — Режим доступу: <https://auth0.com/>
14. CSS офіційна документація по веб розробці з використання HTML CSS JAVASCRIPT [Електронний ресурс] — Режим доступу: <https://www.w3schools.com/css/>

15. Форум науково-технічних статей по ІТ технологіях [Електронний ресурс] — Режим доступу: <https://medium.com/>
16. Клієнт серверна архітектура [Електронний ресурс] — Режим доступу: https://uk.wikipedia.org/wiki/%D0%9A%D0%BB%D1%96%D1%94%D0%BD%D1%8-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80%D0%BD%D0%B0_%D0%B0%D1%80%D1%85%D1%96%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0
17. Типи http запитів [Електронний ресурс] — Режим доступу: <https://developer.mozilla.org/ru/docs/Web/HTTP/Methods>
18. Загальні відомості про вебдизайн [Електронний ресурс] — Режим доступу: <https://uk.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1%D0%B4%D0%B8%D0%B7%D0%B0%D0%B9%D0%BD>
19. Создание сайта, web-дизайн. — Режим доступу: <http://www.artus.ru/>. — Дата доступу: 20.05.2016
20. Итан Маркотт Отзывчивый веб-сайт / Итан Маркотт — М:Манн, Иванов и Фербер, 2012. — С. 176
21. Прохоров Н. Системы управления контентом / Прохоров Н. — КомпьютерПресс, 2007. — С. 134-137
22. Спикльмайр С. Разработка Web-приложений и управление контентом для программистов / Спикльмайр С. — М.: ДМК, 2003 — С. 464

ДОДАТОК

Вебсайт для організації та супроводу наукових конференцій

СПЕЦИФІКАЦІЯ

УКР.КПІм.ІгоряСікорського_ТЕФ_АПЕПС_ТМ7188_21Б

Аркушів 2

Київ 2021

Позначення	Найменування	Примітки
Документація		
УКР.КПІм.ІгоряСікорського_ТЕ Ф_АПЕПС_ТМ71180_21Б 81-1	Записка.docx	Текстова частина дипломної роботи
Компоненти		
УКР.КПІм.ІгоряСікорського_ТЕ Ф_АПЕПС_ТМ71180_21Б 81-2	Бекенд	Основний компонент додатку
УКР.КПІм.ІгоряСікорського_ТЕ Ф_АПЕПС_ТМ71180_21Б 81-3	Фронтент	Основний компонент додатку
УКР.КПІм.ІгоряСікорського_ТЕ Ф_АПЕПС_ТМ71180_21Б 81-4	Опис програми	

ДОДАТОК

Вебсайт для організації та супроводу наукових конференцій

ТЕКСТ ПРОГРАМНОГО МОДУЛЮ

УКР.КПІм.ІгоряСікорського_Ф_АПЕПС_ТМ71180_21Б 81-2

Аркушів 10

Київ 2021

Сервер основний файл server.js

```
const express = require("express");
const http = require('http');
const socketIo = require('socket.io');
const cors = require("cors");
const router = require('./router');

const bodyParser = require("body-parser");
const dbConfig = require("./app/config/db.config");
global.__basedir = __dirname;

const app = express();
app.use(cors({
  origin: "http://localhost:8081",
  methods: ["GET", "POST"],
  optionsSuccessStatus: 200,
  credentials: true,
}));
app.use(router);
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({extended: true}));

const server = http.createServer(app);
const io = socketIo(server, {
  cors: {
    origin: "http://localhost:8081",
    methods: ["GET", "POST"],
    credentials: true,
  },
});

const PORT = 5000;
const USER_JOIN_CHAT_EVENT = "USER_JOIN_CHAT_EVENT";
const USER_LEAVE_CHAT_EVENT = "USER_LEAVE_CHAT_EVENT";
const NEW_CHAT_MESSAGE_EVENT = "NEW_CHAT_MESSAGE_EVENT";
const START_TYPING_MESSAGE_EVENT = "START_TYPING_MESSAGE_EVENT";
const STOP_TYPING_MESSAGE_EVENT = "STOP_TYPING_MESSAGE_EVENT";

io.on("connection", (socket) => {
  console.log(`${socket.id} connected`);
```

```

const {roomId, name, picture} = socket.handshake.query;
socket.join(roomId);

const user = addUser(socket.id, roomId, name, picture);
io.in(roomId).emit(USER_JOIN_CHAT_EVENT, user);

socket.on(NEW_CHAT_MESSAGE_EVENT, (data) => {
  const message = addMessage(roomId, data);
  io.in(roomId).emit(NEW_CHAT_MESSAGE_EVENT, message);
});

socket.on(START_TYPING_MESSAGE_EVENT, (data) => {
  io.in(roomId).emit(START_TYPING_MESSAGE_EVENT, data);
});

socket.on(STOP_TYPING_MESSAGE_EVENT, (data) => {
  io.in(roomId).emit(STOP_TYPING_MESSAGE_EVENT, data);
});

socket.on("disconnect", () => {
  removeUser(socket.id);
  io.in(roomId).emit(USER_LEAVE_CHAT_EVENT, user);
  socket.leave(roomId);
});
});

app.get("/rooms/:roomId/users", (req, res) => {
  const users = getUsersInRoom(req.params.roomId);
  return res.json({users});
});

app.get("/rooms/:roomId/messages", (req, res) => {
  const messages = getMessagesInRoom(req.params.roomId);
  return res.json({messages});
});

const db = require("./app/models");
const {removeUser} = require("./users");
const {addMessage} = require("./message");
const {addUser} = require("./users");
const {getMessagesInRoom} = require("./message");
const {getUsersInRoom} = require("./users");

```

```
const Role = db.role;

const MONGODB_URI =
"mongodb+srv://dmytroarkhyenko:admin@mydb.yebdw.mongodb.net/myFirstDatabase?retryWrites=true&w=majority"
const connectionParams = {
  useNewUrlParser: true,
  useCreateIndex: true,
  useUnifiedTopology: true
}

db.mongoose
  .connect(MONGODB_URI, connectionParams)
  .then(() => {
    console.log("MongoDB підключено");

    function initial() {
      Role.estimatedDocumentCount((err, count) => {
        if (!err && count === 0) {
          new Role({
            name: "user"
          }).save(err => {
            if (err) {
              console.log("error", err);
            }

            console.log("added 'user' to roles collection");
          });

          new Role({
            name: "admin"
          }).save(err => {
            if (err) {
              console.log("error", err);
            }

            console.log("added 'admin' to roles collection");
          });
        }
      });
    }

    initial();
  })
```

```

        console.log("initialized");
    })
    .catch(err => {
        console.error("MongoDB не підключено", err);
        process.exit();
    });

require("../app/routes/auth.routes")(app);
require("../app/routes/user.routes")(app);
require("../app/routes/application.routes")(app);
require("../app/routes/upload.routes")(app);

server.listen(process.env.PORT || 5000, () => {
    console.log(`Server is running on port ${PORT}.`);
});

```

Роути авторизації

```

const {verifySignUp} = require("../middlewares");
const controller = require("../controllers/auth.controller");
const cors = require("cors");

module.exports = function (app) {
    app.use(cors());

    app.post(
        "/api/auth/signup",
        [
            verifySignUp.checkDuplicateUsernameOrEmail,
            verifySignUp.checkRolesExisted
        ],
        controller.signup
    );

    app.get("/api/auth/getall", controller.getall);

    app.post("/api/auth/signin", controller.signin);

    app.post("/api/auth/resetpassword", controller.resetpassword);

    app.post("/api/auth/newpassword", controller.newpassword);

```

```
app.post("/api/auth/updateuser", controller.updateuser);  
};
```

Контроллери авторизації

```
const config = require("../config/auth.config");  
const db = require("../models");  
const crypto = require("crypto");  
const sendgridTransport = require('nodemailer-sendgrid-transport')  
const nodemailer = require('nodemailer')  
const User = db.user;  
const Role = db.role;  
  
var jwt = require("jsonwebtoken");  
var bcrypt = require("bcryptjs");  
const {user} = require("../models");  
const {modelName} = require("mongoose");  
  
const transporter = nodemailer.createTransport(sendgridTransport({  
  auth: {  
    api_key: "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"  
  }  
}))  
  
exports.signup = (req, res) => {  
  const {email, password, place, name, degree, phone} = req.body;  
  const index = email.indexOf("@");  
  const login = email.substring(0, index);  
  const user = new User({  
    username: login,  
    email: email,  
    password: bcrypt.hashSync(password, 8),  
    place: place,  
    name: name,  
    degree: degree,  
    phone: phone,  
  });  
  
  console.log(user);  
  
  user.save((err, user) => {  
    if (err) {
```

```
res.status(500).send({message: err});
return;
}

if (req.body.roles) {
  Role.find(
    {
      name: {$in: req.body.roles}
    },
    (err, roles) => {
      if (err) {
        res.status(500).send({message: err});
        return;
      }

      user.roles = roles.map(role => role._id);
      user.save(err => {
        if (err) {
          res.status(500).send({message: err});
          return;
        }

        res.send({message: "Користувач зареєстрований успішно!"});
      });
    }
  );
} else {
  Role.findOne({name: "user"}, (err, role) => {
    if (err) {
      res.status(500).send({message: err});
      return;
    }

    user.roles = [role._id];
    user.save(err => {
      if (err) {
        res.status(500).send({message: err});
        return;
      }

      res.send({message: "Користувач зареєстрований!"});
    });
  });
}
```

```

    });
  }
});
};

exports.signin = (req, res) => {
  User.findOne({
    username: req.body.username
  })
    .populate("roles", "-__v")
    .exec((err, user) => {
      if (err) {
        res.status(500).send({message: err});
        return;
      }

      if (!user) {
        return res.status(404).send({message: "Користувача не знайдено!"});
      }

      var passwordIsValid = bcrypt.compareSync(
        req.body.password,
        user.password
      );

      if (!passwordIsValid) {
        return res.status(401).send({
          accessToken: null,
          message: "Невірний пароль!"
        });
      }

      var token = jwt.sign({id: user.id}, config.secret, {
        expiresIn: 86400 // 24 hours
      });

      var authorities = [];

      for (let i = 0; i < user.roles.length; i++) {
        authorities.push("ROLE_" + user.roles[i].name.toUpperCase());
      }

      res.status(200).send({

```

```

        id: user._id,
        username: user.username,
        email: user.email,
        roles: authorities,
        accessToken: token,
        phone: user.phone,
        degree: user.degree,
        place: user.place,
        name: user.name
    });
});
};

exports.resetpassword = (req, res) => {
    console.log(123)
    crypto.randomBytes(32, (err, buffer) => {
        if (err) {
            console.log(err)
        }
        const token = buffer.toString("hex")
        User.findOne({email: req.body.email})
        .then(user => {
            if (!user) {
                res.status(500).send({message: "Такого користувача не існує"});
                return;
            }
            user.resetToken = token
            user.expireToken = Date.now() + 3600000
            user.save().then((result) => {
                transporter.sendMail({
                    to: user.email,
                    from: "mydev1488@gmail.com",
                    subject: "Поновлення пароля для сайту конференції",
                    html: `
                        <p>Для поновлення пароля на сайті конференції перейдіть за посиланням
та створіть новий пароль</p>
                        <h5>Посилання для поновлення пароля: <a
href="http://localhost:8082/reset/${token}">Натисніть тут</a></h5>
                    `
                })
                res.send("Перевірте свою поштову скриню!");
            })
        })
    })
};

```

```

    })
  })
}

exports.getall = (req, res) => {
  User.find()
    .exec((err, result) => {
      if (err) {
        res.status(500).send({message: err});
        return;
      }

      if (!result.length) {
        return res.status(404).send({message: "Користувачів не знайдено!"});
      }

      res.status(200).send(result);
    });
}

exports.newpassword = (req, res) => {
  const newPassword = req.body.password
  const sentToken = req.body.token
  User.findOne({resetToken: sentToken, expireToken: {$gt: Date.now()}})
    .then(user => {
      bcrypt.hash(newPassword, 12).then(hashpassword => {
        user.password = hashpassword
        user.resetToken = undefined
        user.expireToken = undefined
        user.save().then(() => {
          res.send("Новий пароль встановлено, повторіть вхід до особистого кобінету!");
        })
      })
    })
    .catch(err => {
      console.log(err)
    })
}

exports.updateuser = (req, res) => {
  const {email, password, place, name, degree, phone, username} = req.body;
  const index = email.indexOf("@");

```

```

const login = email.substring(0, index);
User.findOne({username: username})
  .then(user => {
    if (password) {
      bcrypt.hash(password, 12).then((hashedpassword) => {
        user.email = email;
        user.username = login;
        user.password = hashedpassword;
        user.place = place;
        user.name = name;
        user.degree = degree;
        user.phone = phone;
        console.log(user);
        user.save().then(() => {
          res.json({message: "Дані користувача поновлено!"})
        })
      })
    } else {
      user.email = email;
      user.username = login;
      user.place = place;
      user.name = name;
      user.degree = degree;
      user.phone = phone;
      console.log(user);
      user.save().then(() => {
        res.json({message: "Дані користувача оновлено!"})
      })
    }
  })
  .catch(err => {
    console.log(err)
  })
}

```

Головний компонент фронтенду

```

import React, {useEffect, useState} from "react";
import {Link, Route, Switch} from "react-router-dom";
import "bootstrap/dist/css/bootstrap.min.css";
import "../App.css";

```

```

import AuthService from "../services/auth.service";

import {Login} from "../components/Auth/Login";
import {Register} from "../components/Auth/Register";
import {Home} from "../components/Pages/Home/Home";
import {Profile} from "../components/Pages/Profile";
import {ChatRoom} from "../components/Pages/Chat/ChatRoom/ChatRoom";
import {BoardAdmin} from "../components/Pages/AdminBoard/BoardAdmin";
import {ResetPassword} from "../components/Auth/ResetPassword";
import {NewPassword} from "../components/Auth/NewPassword";
import {Application} from "../components/Pages/Application";
import {Archives} from "../components/Pages/Archives";
import {ImageGallery} from "../components/Pages/Gallery";
import {Presentations} from "../components/Pages/Presentations";

const App = () => {
  const [showAdminBoard, setShowAdminBoard] = useState(false);
  const [currentUser, setCurrentUser] = useState(undefined);

  useEffect(() => {
    const user = AuthService.getCurrentUser();

    if (user) {
      setCurrentUser(user);
      setShowAdminBoard(user.roles.includes("ROLE_ADMIN"));
    }
  }, []);

  const logOut = () => {
    AuthService.logout();
  };

  return (
    <>
      <div className={"home-hero"}>
        <div className={"home-hero-text"}>
          <h1>Сталий розвиток – XXI століття
            (наукові читання імені Ігоря Недіна)</h1>
        </div>
      </div>
    </>
  );
};

```

```

</div>
<div>
  <nav className="navbar navbar-expand navbar-dark bg-dark">
    <Link to={"/"} className="navbar-brand">
      Міжнародна науково-практична конференція
    </Link>
    <div className="navbar-nav mr-auto">
      {currentUser && (
        <li className="nav-item">
          <Link to={` /chat/general`} className="nav-link">
            Користувацький чат
          </Link>
        </li>
      )}

      {showAdminBoard && (
        <li className="nav-item">
          <Link to={"/admin"} className="nav-link">
            Сторінка адміністратора
          </Link>
        </li>
      )}
    </div>

    {currentUser ? (
      <div className="navbar-nav ml-auto">
        <li className="nav-item">
          <Link to={"/profile"} className="nav-link">
            {currentUser.username}
          </Link>
        </li>
        <li className="nav-item">
          <a href="/login" className="nav-link"
onClick={logOut}>
            Вийти
          </a>
        </li>
      </div>
    ) : (
      <div className="navbar-nav ml-auto">
        <li className="nav-item">

```

```

        <Link to={"/login"} className="nav-link">
            Login
        </Link>
    </li>

    <li className="nav-item">
        <Link to={"/register"} className="nav-link">
            Реєстрація
        </Link>
    </li>
</div>

    )}
</nav>
<ul className="white-text nav justify-content-center navbar-light bg-
light">

    <li className="nav-item">
        <Link to={"/home"} className="nav-link">
            Головна
        </Link>
    </li>

    <li className="nav-item">
        <Link to={"/application"} className="nav-link">
            Заявка на участь
        </Link>
    </li>

    <li className="nav-item">
        <Link to={"/gallery"} className="nav-link">
            Галерея
        </Link>
    </li>

    <li className="nav-item">
        <Link to={"/presentations"} className="nav-link">
            Архів презентацій
        </Link>
    </li>

    <li className="nav-item">
        <Link to={"/archives"} className="nav-link">
            Архів робіт
        </Link>
    </li>

```

```

    </ul>
    <div className="container">
      <Switch>
        <Route exact path={['/', '/home']} component={Home}/>
        <Route exact path="/application" component={Application}/>
        <Route exact path="/login" component={Login}/>
        <Route exact path="/register" component={Register}/>
        <Route exact path="/profile" component={Profile}/>
        <Route exact path="/passwordreset"
component={ResetPassword}/>
        <Route path="/reset/:token" component={NewPassword}/>
        <Route path="/chat/:roomId" component={ChatRoom}/>
        <Route path="/admin" component={BoardAdmin}/>
        <Route path="/presentations" component={Presentations}/>
        <Route path="/archives" component={Archives}/>
        <Route path="/gallery" component={ImageGallery}/>
      </Switch>
    </div>
  </div>
  <div className={"footer"}>
    <ul className="white-text nav justify-content-center navbar-light bg-
dark">
      <li className="nav-item">
        <Link to={"/home"} className="nav-link">
          Головна
        </Link>
      </li>
      <li className="nav-item">
        <Link to={"/application"} className="nav-link">
          Заявка на участь
        </Link>
      </li>
      <li className="nav-item">
        <Link to={"/gallery"} className="nav-link">
          Галерея
        </Link>
      </li>
      <li className="nav-item">
        <Link to={"/presentations"} className="nav-link">
          Архів презентацій

```

```

        </Link>
    </li>
    <li className="nav-item">
        <Link to={"/archives"} className="nav-link">
            Архів робіт
        </Link>
    </li>
</ul>
<h5 className={"text-center"}>
    Контактні особи:
</h5>
<div className={"flex"}>
    <p className={"first"}>Сегеда Ірина Василівна +38 (050) 695 54 43
<br/>Кублій Лариса Іванівна +38
        (097) 558 27 17</p>
    <p>Кузьменко Марія Володимирівна +38 (098) 659 25 93 <br/>Е-Mail:
nedin.conf@gmail.com</p>
    </div>
</div>
</>
);
};

export default App;

```

ДОДАТОК 3

Вебсайт для організації та супроводу наукових конференцій

ОПИС ПРОГРАМНОГО МОДУЛЮ

УКР.КПІм.ІгоряСікорського_ТЕ Ф_АПЕПС_ТМ71180_21Б 81-4

Аркушів 9

Київ 2021

АНОТАЦІЯ

Додаток містить опис веб сайту який був розроблений для організації та супроводу наукових конференцій.

В вебсайт має такі функції:

- Реєстрація, авторизація;
- Користування чатом для зареєстрованих користувачів;
- Перегляд архівів робіт за минулі роки.
- Менеджмент завантажених файлів
- Подача заявки для участі в конференції.

Вхідні дані отримуються з форм введених на сайті.

Система створена в Webstorm 2021 з використанням мови програмування Javascript. Графічний інтерфейс користувача розроблений за допомогою HTML React.js, CSS Для збереження даних використана система управління базами даних (СУБД) MongoDB. Для доступу до бази даних та автоматичного створення сутностей та управління ними був застосований Express.js фреймворк Node.js з використанням драйвера для MongoDB mongoose.

ЗМІСТ

ЗАГАЛЬНІ ВІДОМОСТІ	68
ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ	69
ОПИС ЛОГІЧНОЇ СТРУКТУРИ	70
ВИКОРИСТОВУВАНІ ТЕХНІЧНІ ЗАСОБИ	71
ВИКЛИК І ЗАВАНТАЖЕННЯ	72
ВХІДНІ І ВИХІДНІ ДАНІ	73

ЗАГАЛЬНІ ВІДОМОСТІ

У цьому додатку описано сайт розроблений для організації та супроводу наукових конференцій

Модулі системи надано в ДОДАТКУ 2.

Передбачено що розроблений сайт працює в будь-якому з можливих браузерів

Компоненти необхідні для перегляду: будь який браузер (Chrome, Firefox, Internet Explores).

Використана мова програмування — JS.

ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Розроблений вебсайт відображає останні новини про конференцію, заплановані події на конференції. Зберігає архів робіт за минулі роки а також розроблений для організації та супроводу наукових конференцій дає можливість адміністратору сайту займатись оновленням наповнення сайту файлами, переглядом зареєстрованих користувачів, обробляти заявки на участь.

Також у разі якщо користувач не є зареєстрованим він може подати заявку на участь на відповідній сторінці а також ознайомитись з роботами за минулий рік.

Зареєстровані користувачі можуть спілкуватись у спільному чаті, а також поновлювати інформацію про себе за необхідності.

ОПИС ЛОГІЧНОЇ СТРУКТУРИ

Програмний додаток розроблений за схемою дворівневої архітектури «Клієнт-Сервер». Доступ до бази даних здійснюється за допомогою драйвера для СУБД MongoDB такого як mongoose.

На початку роботи з сайтом користувач може зразу визначити свої можливості, а саме можливість зареєструватись для участі у конференції або ж на сайті для спілкування з іншими учасниками конференції

Також у разі якщо користувач забув свій пароль то він може скористуватись можливістю поновити пароль шляхом відправки листа на пошту користувача.

ВИКОРИСТОВУВАНІ ТЕХІЧНІ ЗАСОБИ

Для розробки програмного продукту було обрано:

- операційну систему Mac OS;
- Javascript
- середовище Webstorm;
- фреймворк Express.js для Node.js;
- систему керування базами даних MongoDB.
- Веб сервіс для відправки листів Sendgrid
- Бібліотеку React.js для реалізації користувацького інтерфейсу

ВИКЛИК І ЗАВАНТАЖЕННЯ

Для завантаження сайту необхідно скористуватись браузером та просто перейти за посиланням яке буде визначене після розгортання на якомусь домені

ВХІДНІ І ВИХІДНІ ДАНІ

Для реєстрації вхідні отримуються при заповненні форми, форма містить строкові вхідні даних.

Для інших функцій пов'язаних з авторизацією та поновленням паролів вхідні дані також отримуються при заповненні форми, форма містить строкові вхідні даних.

Для менеджменту файлами на сторінці адміністратора створено 3 файлові менеджери які мають змогу зчитувати тільки img/png/jpeg для фото, pdf для презентацій та архівів робіт за минулі роки

Вихідними даними для даних менеджерів служить 3 сторінки на яких ми маємо змогу переглядати всі вище описані файли а також гортати завантажені презентації та архіви