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
спеціальності 122 “Комп’ютерні науки”
освітня програма “Комп’ютерний моніторинг та геометричне моделювання процесів і систем”
на тему: “Програмна платформа для організації конференцій з використанням веб-технологій”

Виконала:
студентка IV курсу, групи ТР-72
Федорова Юлія Євгенівна _____

Керівник:
доцент, кандидат технічних
наукКублій Лариса Іванівна _____

Рецензент:
старший науковий співробітник, кандидат технічних наук,
професор кафедри інформаційних технологій та програмування
Інституту комп’ютерних технологій Університету “Україна”
Самарай Валерій Петрович _____

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 (бакалаврський)

Спеціальності 122 “Комп’ютерні науки”

Освітня програма “Комп’ютерний моніторинг та геометричне моделювання процесів і систем”

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____Олександр КОВАЛЬ
(підпис)
”___”_____2021р.

ЗАВДАННЯ

на дипломну роботу студенту

Федоровій Юлії Євгенівні

1. Тема роботи “Програмна платформа для організації конференцій з використанням веб-технологій”

керівник роботи Кублій Лариса Іванівна, к.т.н., доцент
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від “24” травня 2021 р. **№ 1267-с**

2. Строк подання студентом роботи 09 червня 2021 р.

3. Вихідні дані до роботи: персональний комп’ютер під керуванням операційної системи Microsoft Windows, мова програмування JavaScript, мова програмування NodeJs.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити): проаналізувати існуючі програмні рішення та можливі засоби реалізації взаємодії, обґрунтувати вибір програмних застосунків і шляхи розробки програмних додатків, розробити програмне забезпечення, розробити користувацький інтерфейс

5. Перелік ілюстраційного матеріалу (з точним зазначенням обов’язкових креслень) 1. Завдання розробки програмної платформи для організації конференцій з використанням веб-технологій. 2. Існуючі веб-додатки для організації конференцій. 3. Засоби розробки програмної платформи. 4. Розробка платформи. 5. Опис роботи користувача з системою.

6. Публікації: Матеріали XIV Міжнародної науково-практичної конференції молодих вчених і студентів 2021 року “Сучасні проблеми наукового забезпечення енергетики”, м. Київ, КПІ ім. Ігоря Сікорського.

7. Дата видачі завдання “15” жовтня 2020 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Затвердження теми роботи	03.12.20	
2.	Вивчення та аналіз задачі	13-19.04.21	
3.	Розробка архітектури та загальної структури системи	20-26.04.21	
4.	Розробка структур окремих підсистем	27.04-03.05.21	
5.	Програмна реалізація системи	04-13.05.21	
6.	Оформлення пояснювальної записки	14-16.05.21	
7.	Захист програмного продукту	11.05.21	
8.	Передзахист	25.05.21	
9.	Захист	17.06.21	

Студент _____ Федорова Ю. Є.

(підпис) (прізвище та ініціали)

Керівник роботи _____ Кублій Л. І.

(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Метою дипломної роботи є розробка програмної платформи для організації наукових конференцій з використанням веб-технологій, основними складовими якої є збір даних про потенційних учасників, робота з архівом, взаємодія та обмін даними між учасниками та організатором конференції. Платформа написана за допомогою редактора вихідного коду Visual Studio Code. Для розробки веб-додатку використано технології: Javascript, React, Redux, PostgreSQL, Node.js, Express.js, HTML, CSS, JWT.

Записка містить 78 сторінок, 22 рисунки, 4 додатки і 22 посилання.

Ключові слова: веб-додаток, конференція, захід, клієнт-серверна архітектура, Node.js, PostgreSQL.

ABSTRACT

The purpose of the thesis is to develop a software platform for organizing scientific conferences using web technologies, the main components of which are data collection on potential participants, work with the archive, interaction and data exchange between participants and the conference organizer. The platform was developed using the Visual Studio Code source editor. The following technologies were used to develop the web application: Javascript, React, Redux, PostgreSQL, Node.js, Express.js, HTML, CSS, JWT.

The note contains 78 pages, 22 figures, 4 attachments and 22 links.

Keywords: web application, conference, event, client-server architecture, Node.js, PostgreSQL.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ І ТЕРМІНІВ

БД — база даних.

СКБД — система керування базами даних.

JSON (JavaScript Object Notation) — текстовий структурований формат даних.

API (Application Programming Interface) — стандартизований інтерфейс для взаємодії клієнтів з сервером за стандартним протоколом.

CSS (Cascading Style Sheets) — мова опису зовнішнього вигляду документа, написаного з використанням HTML.

HTML (HyperText Markup Language) — мова гіпертекстової розмітки.

HTTP (HyperText Transfer Protocol) — протокол передачі даних.

JWT (JSON Web Tokens) — стандарт для створення токенів доступу.

JavaScript — мова програмування.

React — бібліотека JavaScript для розробки користувацького інтерфейсу.

Redux — бібліотека JavaScript для керування станом веб-додатку.

Node — платформа для розробки високопродуктивних мережевих веб-додатків, написаних мовою JavaScript.

Express — веб-фреймворк для Node.js

PostgreSQL — система керування базами даних

Email (email) — адреса електронної пошти.

Фреймворк — програмна платформа, що забезпечує високорівневу або загальну функціональність програми. Призначено для прискореної розробки додатків.

Клієнт — веб-браузер, у роботі іменується браузер, якщо не вказано щось конкретне.

ВСТУП

Інформаційні технології стають дедалі важливішими і впливають на суспільний розвиток. Інформація стала одним з основних стратегічних ресурсів, а інформаційні технології — інструментом використання цього ресурсу. Еволюція інформаційних технологій є фактором, який впливає на зміни у всіх сферах життя і діяльності людей.

Важливим завданням є створення інформаційних ресурсів для спрощення і прискорення доступу до інформації і її використання. Університети та науково-дослідні установи регулярно проводять наукові конференції, тому для забезпечення успіху цієї діяльності потрібне ретельне планування організації конференції, керування учасниками [1].

Зважаючи на те, що у вільному доступі програмне забезпечення для організації конференцій відсутнє, актуальним завданням є розробка і впровадження програмної платформи для організації наукових конференцій з використанням веб-технологій. Така система зможе надати засоби для планування, швидкої і якісної підготовки і проведення конференцій, забезпечить потенційних учасників і всіх зацікавлених осіб інформаційними матеріалами, надасть доступ до поточних і архівних матеріалів конференції.

Для реалізації серверної частини веб-застосунку найкраще використовувати веб-сервер Node.js та фреймфорк Express.js. Його головними перевагами є асинхронність, швидке підняття сервера, мінімальне використання оперативної пам’яті і велика кількість інструментів для створення швидких і функціональних мікросервісів. Для роботи з даними — реляційну систему керування базами даних PostgreSQL. Вона є однією з найкращих СКБД з відкритим вихідним кодом для легкого масштабування і для легкої інтеграції в мікросервісну архітектуру. Для швидкої реалізації клієнтської частини з можливістю створювати гнучкий інтерфейс — бібліотеки React та Redux, які базуються на мові програмування JavaScript. Для аутентифікації користувача системи між сайтом та сервером —

стандарт JWT. Токени створюються сервером, підписуються секретним ключем і передаються клієнту, який в подальшому використовує даний токен для підтвердження своєї особи.

Впровадження програмної платформи надасть можливість організувати власними силами конференцію, усуне розрізнений підхід до планування заходів за рахунок централізації інформації.

У першому розділі пояснювальної записки сформульовано постановку задачі, описано вхідні дані і компоненти веб-ресурсу. У другому — проаналізовано існуючі підходи і системи для організації заходів. У третьому розділі обґрунтовано вибір засобів для створення веб-ресурсу. Програмну реалізацію описано в четвертому розділі. У п’ятому розділі описано процес розгортання ресурсу і подано приклади взаємодії з ним.

1. ЗАДАЧА РОЗРОБКИ ПРОГРАМНОЇ ПЛАТФОРМИДЛЯ ОРГАНІЗАЦІЇ КОНФЕРЕНЦІЙ

У теперішній час невід'ємною частиною діяльності вищих навчальних закладів є регулярне проведення наукових конференцій, семінарів, нарад та ін. Основна мета організації подібних наукових заходів в навчальних закладах — це розвиток і підтримка наукових досліджень як основи фундаменталізації освіти, як бази підготовки кваліфікованих фахівців відповідно до потреб суспільства. Масштаби наукових заходів можуть досягати як міжвузівських і регіональних, так і міжнародних рівнів. Такі зібрання вимагають інтенсивної роботи передусім з боку організаторів. Для проведення наукового заходу на високому рівні необхідна ґрунтовна підготовка, оскільки високий рівень організації є підставою для зміцнення статусу конференції, збільшення кількості учасників, залучення цікавих доповідачів. Ефективність підготовки і проведення заходу значною мірою підвищується за рахунок використання оргкомітетом допоміжних програмних засобів.

1.1. Задача створення програмної веб-платформи для організації наукових конференцій

Велику роль у підготовці сучасних фахівців відіграє така форма організації педагогічного процесу, як науково-практична конференція.

Науково-практична конференція — це захід, який проводиться з метою обговорення різних питань науки, методики і практики, вироблення рекомендацій щодо їхнього розв'язання.

Науково-практичну конференцію можна розглядати як одну з ефективних форм взаємодії викладачів і студентів, як необхідний етап в організації їхньої дослідницької діяльності.

Набуття вмінь щодо підготовки доповідей, повідомлень про проведені дослідження, навички громадської думки отриманих результатів і їхнього обгрунтування на науково-практичній конференції сприяють формуванню комунікативних здібностей студентів. Керівництво дослідницькою діяльністю і підготовкою студентів до науково-практичної конференції зміцнює наукове співробітництво викладачів і студентів.

На жаль, педагогічний потенціал науково-практичної конференції недостатньо затребуваний в педагогічній практиці освітніх установ. Однією з причин є недостатнє володіння технологією її організації. Як наслідок цього виникає протиріччя між необхідністю використовувати науково-практичну конференцію як форму організації освітнього процесу і небажанням або невмінням проводити цей захід.

Таким чином, виникає проблема оволодіння викладачами освітніх установ технологією організації та проведення науково-практичної конференції як однією з форм взаємодії і як невід'ємної складової дослідницької діяльності студентів і самих викладачів

1.2. Вибір програмного забезпечення для організації заходу

Якщо не використовувати спеціальне програмне забезпечення для підготовки та організації наукових конференцій, можна зіштовхнутися з дуже складною ситуацією: багато людей використовують велику кількість документів Word або Excel для керування інформацією, яка повторюється між собою. Якщо учасники змінюють імена або скасовують події, може виникати плутанина при оновленні кількох документів. Тому організація заходу без спеціального програмного забезпечення може створити багато додаткової роботи.

Організаторам при підготовці і проведенні різноманітних заходів без використання спеціалізованих програмних платформ для керування заходами необхідно було застосовувати для власноручного проектування веб-сайтів такі сторонні інструменти, як, наприклад, WordPress або Squarespace [2].

На даний час існують десятки додатків для планування заходів, серед яких найпопулярнішими є Monday, Eventzilla, Explara, Eventbrite, Bizzabo. Вони дають можливість зробити все — від залучення відвідувачів до цифрової візуалізації карт місцевості і планів приміщень. І хоч усі вони пропонують абсолютно різні функції, у них є одна спільна риса: вони виключають підхід “зроби сам” і оптимізують процес керування подіями [3].

При розробці програмної платформи для організації конференцій було враховано функціонал вище вказаних додатків для керування заходами.

1.3. Планування конференції за допомогою веб-додатку

Програмне забезпечення для організації конференцій або інших заходів — це рішення, яке розв’язує безліч задач: від реєстрації і планування заходу до створення списків електронних пошт. Платформа допомагає організаторам досягнути успіху в усіх аспектах їхніх заходів — від реєстрації і створення заходів до створення звітів. Оскільки програмне забезпечення для керування заходами об’єднує всі інструменти, потрібні організатору заходів будь-якого виду, в одну платформу, це робить програму легкою для розуміння та з великою кількістю готових ідей.

Система повинна відповідати всім вимогам дизайну, використовувати загальнодоступні значки та шаблони інтерфейсу користувача.

Веб-портал розрахований на 3 категорії користувачів: незареєстрований користувач, зареєстрований користувач та адміністратор сайту (організатор конференції). Відповідно до категорій розподіляється функціонал:

Незареєстрований користувач — має можливість зареєструватися, переглядати поточну інформацію про конференцію, переглядати архів доповідей з минулих конференцій.

Зареєстрований користувач — після авторизації має можливість переглядати поточну інформацію про конференцію, переглядати архів доповідей з минулих конференцій, а також може зайти у власний кабінет і подати свою доповідь.

Адміністратор — за допомогою створеної програмної платформи має можливість розмістити інформаційний лист конференції, приймати реєстрації учасників і доповідачів на конференцію, розмістити і редагувати програму конференції, створювати списки електронних пошт для кожної конференції для подальшого користування (e-mail-розсилки за адресами колишніх і нових учасників конференції). Також організатор має можливість створити нову конференцію, вказати час і дату проведення заходу, оперативно сповістити учасників і звичайних слухачів про певні зміни, розмістити на сайті файли різного типу: фотографії, відео-матеріали. Вказуючи місце проведення конференції, можна додати посилання для відео-зв'язку.

Однією з головних задач розробленої веб-платформи є збір даних про потенційних учасників конференції. При реєстрації користувач деяку інформацію вносить сам (ім'я, прізвище, статус, вчений ступінь, вчене звання, місце роботи, e-mail), а деяку вибирає з форми (наукова секція, форма участі).

Після реєстрації або авторизації система перенаправляє користувача до його особистого віртуального кабінету, де він вже може подати твою доповідь. Коли організатор створює нову конференцію, всі матеріали минулої конференції переносяться в архів.

Незареєстровані користувачі можуть лише переглядати поточну інформацію на сайті, а можливість відправити доповідь є тільки у зареєстрованих користувачів.

1.4. Задачі для сервера веб-додатку

В основі розробленої програмної платформи для організації конференцій лежить так звана модель взаємодії "клієнт-сервер", яка дає можливість розділяти функціонал і обчислювальне навантаження між клієнтськими і серверними додатками. Клієнтом в розробленій системі є браузер, а сервером — веб-сервер.

Клієнт і сервер взаємодіють за допомогою HTTP-запитів. У випадках, коли дані для передачі є специфічними об'єктами, вони конвертуються у рядковий

формат JSON. Його використання зумовлено легкістю серіалізації та десеріалізації складних структур даних.

У випадку відправлення запитів GET відбувається динамічне визначення типів. Коли на клієнтську частину приходить відповідь, то також відбувається динамічна типізація завдяки особливостям мови JavaScript. Клієнти й сервер створені за допомогою різних технологій, але це не впливає на основні принципи їхньої взаємодії.

Сервер надає усі необхідні дані порталу. Він відповідальний за авторизацію і реєстрацію користувача в системі. Сервер надає доступ до бази даних, що містить персональні дані користувачів, інформацію про надіслані доповіді та конференцію в цілому.

2. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ДЛЯ ОРГАНІЗАЦІЇ КОНФЕРЕНЦІЙ

Університети і науково-дослідні інститути регулярно проводять наукові конференції. У даний час існує досить багато програмного забезпечення для допомоги оргкомітету щодо збору і систематизації інформації. До них можна віднести, наприклад, портал Science-Community [4], його іноземний аналог EasyChair Conference System [5] та інші. Ці ресурси зручні і дають можливість швидко і якісно донести до користувачів інформацію про конференцію. Проте для організації конференції наукових закладів цього функціоналу не достатньо.

Наприклад, перед оргкомітетом можуть стояти такі завдання: швидко і якісно зібрати інформацію про учасників конференції, завантажити і провести рецензування їхніх доповідей, створити звіти або повторну конференцію тощо. Вказані інтернет-сервіси не дають можливості розв’язати всю сукупність завдань. Також велика частина досліджених сайтів конференцій є “статичними”, написані за допомогою стандартної мови розмітки веб-сторінок в Інтернеті HTML, тобто представляють собою набір розміщених на Інтернет-сервері файлів, що не містять виконуваних на стороні сервера програм. На таких сайтах неможливо динамічно генерувати вміст, повноцінно підтримувати відвідувачів, наповнювати сайт інформацією без допомоги FTP або сторонніх веб-скриптів, які дають можливість редагувати сторінки.

Також на даний час існують десятки додатків для планування заходів, які мають досить широкі можливості. Розглянемо деякі з них.

2.1. Веб-додаток Explara

Для організаторів подій веб-додаток Explara [6] (рисунок 2.1) — один з найкращих варіантів. Інтегрована платформа об’єднує інструменти, які десять інших

програм можуть запропонувати окремо, допомагаючи полегшити реєстрацію, участь у заході та звітування після події в одному місці. Менеджери подій можуть створити абсолютно фірмовий захід для відвідувачів, який можна використовувати.

Рисунок 2.1 — Веб-додаток Explara [6]

Увійшовши до Explara, можна побачити інформаційну панель з 15 блоками, кожен з яких дає можливість зробити щось (наприклад, створити опитування, створити електронну пошту або керувати квитками). Першим кроком є — клацання по кнопці “Покупка квитків / реєстрація подій”, щоб створити свою подію. Як тільки подія буде додана до Explara, одразу з’являється можливість використати всі інші блоки.

Щоб створити власний додаток для подій, потрібно натиснути на кнопку “Підключити”, ввести назву та опис свого додатка, додати політику конфіденційності, а також завантажити піктограму програми і зображення функції для iOS та Android.

Мінімалістичний інтерфейс полегшує додавання нових модулів до додатку, наприклад, включення гри для заохочення взаємодії під час сесії або створення порядку денного. Також можна додавати зображення й відео, надсилати електронні листи і сповіщення або нагороджувати користувачів балами за виконання певних дій.

Щоб краще зрозуміти, що працює для учасників, є вбудована аналітика, яка відстежує використання додатків протягом події.

За допомогою веб-додатка Explara можна також:

— переглядати встановлення додатків на iOS та Android, веб-перегляди, перегляди зображень та відео, найбільш відвідувані сторінки, основні сесії порядку денного тощо;

— повернутися до головної сторінки інформаційної панелі та дослідити інші переходи;

— розробляти й друкувати значки подій, створювати сторінку товарів, щоб продавати товари, пов’язані з подією, або створювати “пакет відстеження” для обміну інформацією про житло для учасників подорожі.

2.2. Веб-додаток Splash

Веб-додаток Splash [7] (рисунок 2.2) допомагає створити індивідуальну цільову сторінку для будь-якої події, тоді як сервер надає все необхідне, щоб краще вивчити список учасників.

За допомогою веб-додатка Splash можна позначити учасників тегами на основі даних, зібраних на етапі реєстрації, та їхніх цифрових дій у день події або додати власні теги в будь-який момент.

Організатор події має змогу надсилати одноразові маркетингові електронні листи або автоматизувати власні та персоналізовані запрошення на події, нагадування, оновлення, подальші дії тощо.

Розширені можливості сегментації допомагають розділити аудиторію на основі інтересів одним клацанням миші і надсилати персоналізований вміст, який залежить від того, як учасники використовують програму під час заходу.

Наприклад, можна надіслати електронний лист усім, хто відвідав певну бесіду, і Splash це спростить.

Всередині інформаційної панелі використовується розділ Hub, щоб класифікувати подібні події та перевірити, чи не збігаються учасники, чи є певний гість, який регулярно купує VIP-квитки або інші доповнення.

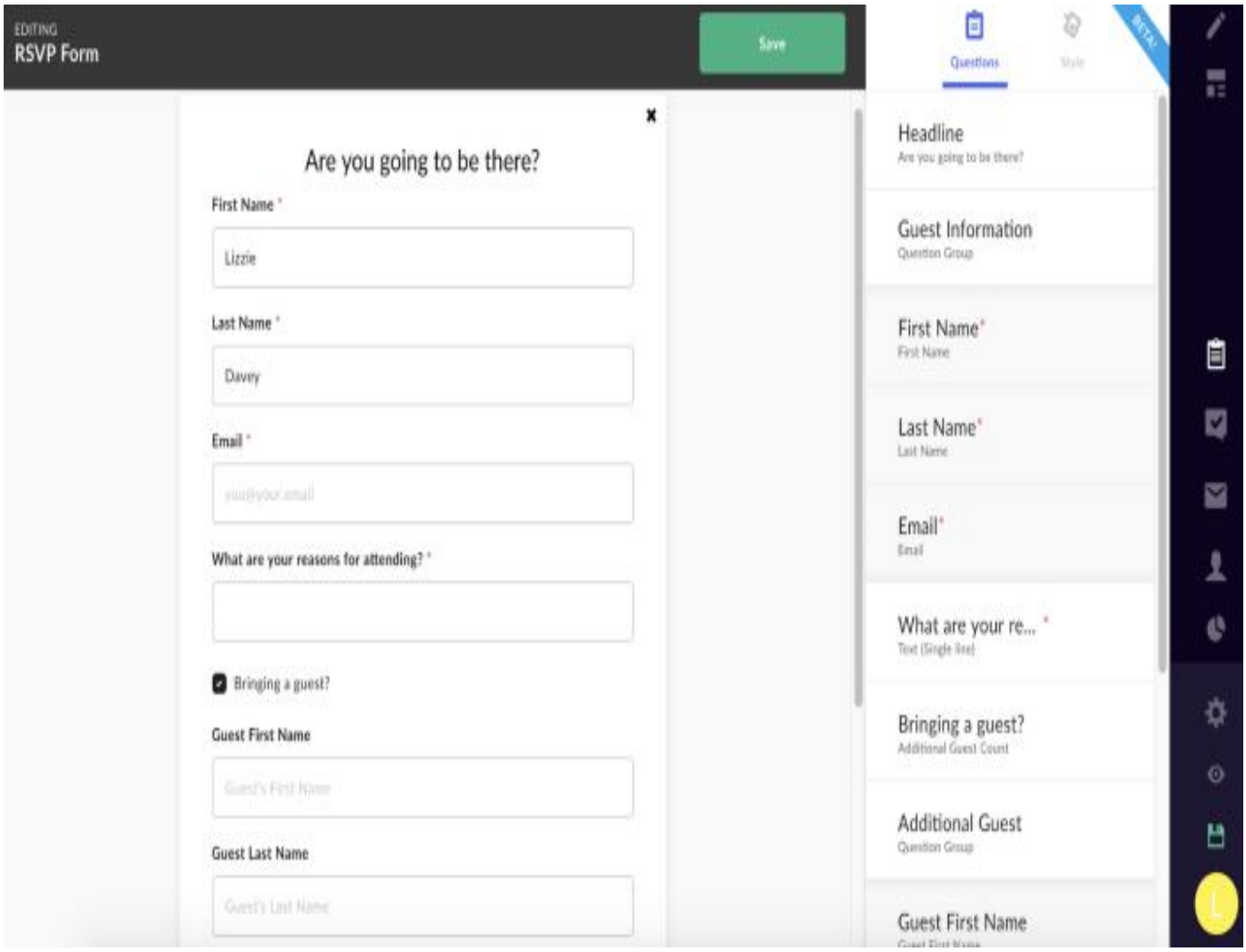

Рисунок 2.2 — Веб-додаток Splash [7]

Цей рівень сегментації та налаштування нагадує традиційний маркетинговий CRM-інструмент, але Splash представляє всі його функції в інтуїтивно зрозумілому та зручному для вивчення інтерфейсі. Таким чином, планувальники подій можуть використовувати потужність цифрового маркетингу без необхідності вивчати новий набір навичок або складний інструмент.

2.3. Платформа 4science

Сервіс 4science [8] (рисунок 2.3) організовує повний цикл онлайн-супроводу конкурсів і заходів, надає сильну консультаційну і технічну підтримку.

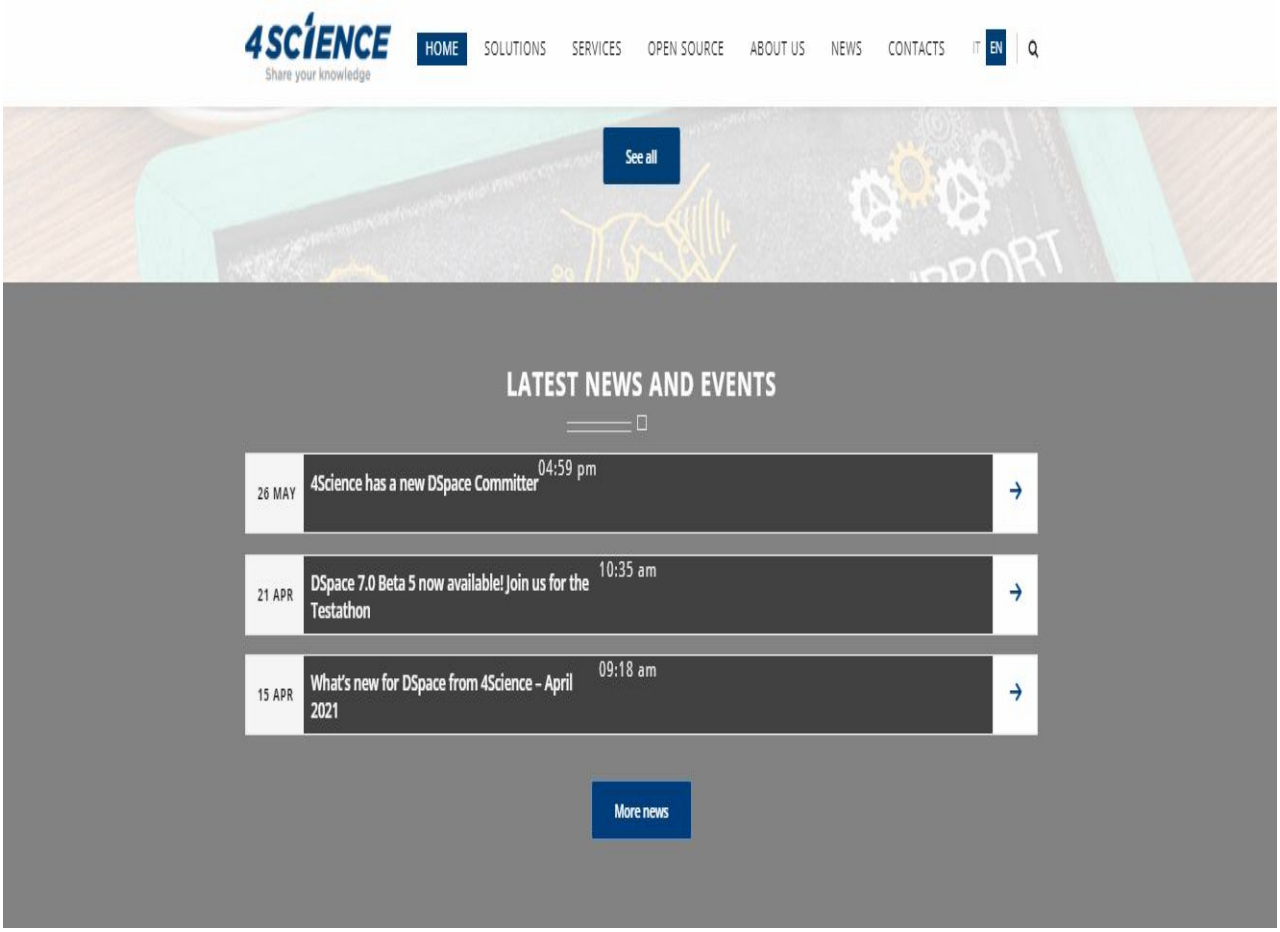

Рисунок 2.3 — Веб-додаток 4science [8]

Сторінка конференції створюється за допомогою спеціального конструктора. На ній можна розмістити кілька типів анкет для різних категорій учасників. Форми реєстрації супроводжуються інструкціями для заповнення або підказками. Сервіс 4science забезпечує не тільки збір, але й керування заявками: користувач приймає, відхиляє або відправляє їх на доопрацювання.

У системі легко обробляти заявки, оскільки всі учасники і їхні роботи видно на одному екрані, не потрібно щодня шукати їх у пошті. Ці дані зручно аналізувати: файли вивантажуються в потрібному для користувача форматі — xls, pdf або word.

Важлива перевага цієї платформи — функціонал для роботи з експертами або програмним комітетом. Щоб оцінити, чи надійшли заявки, і сформувати програму, в системі налаштовуються гнучкі правила розподілу заявок. Так експерти бачать роботи тільки за своєю тематикою.

Додаток 4science має важливу для наукових заходів функцію — оформляє всі тези в єдиний збірник. Його форматування і верстка відбуваються за пару клацань мишкою. За бажанням збірник може бути опублікований, і кожен учасник легко знайде свою роботу.

Сервіс 4science найбільш підходить для проведення наукових заходів, оскільки платформа розроблялася спеціально для них. Вона враховує необхідність роботи з експертами, публікації збірника тез та відсутність продажу квитків.

2.4. Застосунок для організації заходу Bizzabo

Застосунок Bizzabo (рисунок 2.4) — це універсальне рішення для створення орендованої події для учасників. Від створення веб-сайту для події до збору даних про учасників та створення звітів — Bizzabo має інструменти для кожного аспекту події.

Редактор веб-сайтів, що перетягує програмне забезпечення, дає можливість створювати фірмовий сайт або вбудовані віджети, які включають найважливіші особливості події. Є можливість додати віджети для порядку денного, спікерів та учасників, залежно від того, на чому має бути найбільший акцент. Потім це можна перетворити на спеціальний додаток для подій, до якого учасники можуть отримати доступ для створення персоналізованих розкладів і перегляду вбудованих мережевих спільнот.

Параметри списку гостей також досить розумні: можна легко сегментувати учасників за допомогою тегів, щоб створювати розумні списки спікерів, учасників та VIP-персон, а потім використовувати функцію маркетингу електронною поштою Bizzabo, щоб надсилати персоналізовані електронні листи кожному сегменту.

Застосунок Bizzabo пропонує унікальні інструменти, такі як функція Hot Leads, яка дає можливість запускати цілеспрямовані кампанії ремаркетингу для

учасників, які не завершили процес реєстрації, та функція Ticket Boost, яка допомагає учасникам поширювати інформацію в соціальних мережах за допомогою одного клацання мишкою.

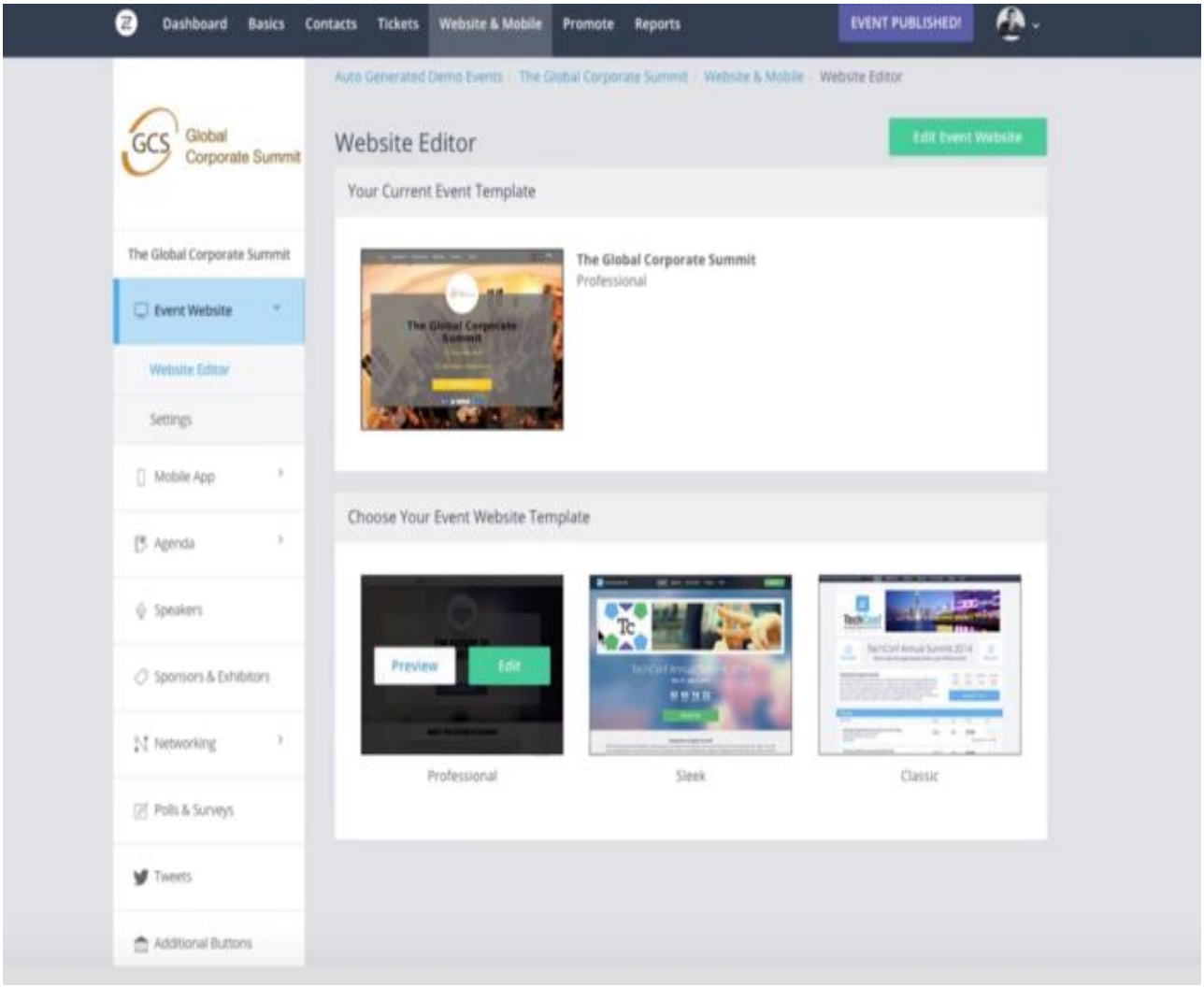

Рисунок 2.4 — Веб-додаток Bizzabo [9]

Усі вищевказані програмні платформи дають можливість зробити дуже багато для організації заходу — від залучення відвідувачів до цифрової візуалізації карт місцевості і планів приміщень. І хоч усі вони пропонують абсолютно різні функції, метою всіх цих сервісів є заробіток шляхом впровадження платних послуг ч продажу персональних даних. Як правило,такого типу сервіси не надають повного відчуття володіння своєю спільнотою, що є досить великим недоліком.

3. ЗАСОБИ РОЗРОБКИ ПРОГРАМНОЇ ПЛАТФОРМИ

Для забезпечення якісної розробки веб-додатку потрібно правильно вибрані засоби розробки програмного забезпечення. Програмна платформа для організації конференції створена за принципами триланкової архітектури побудови програм (рисунок 3.1).

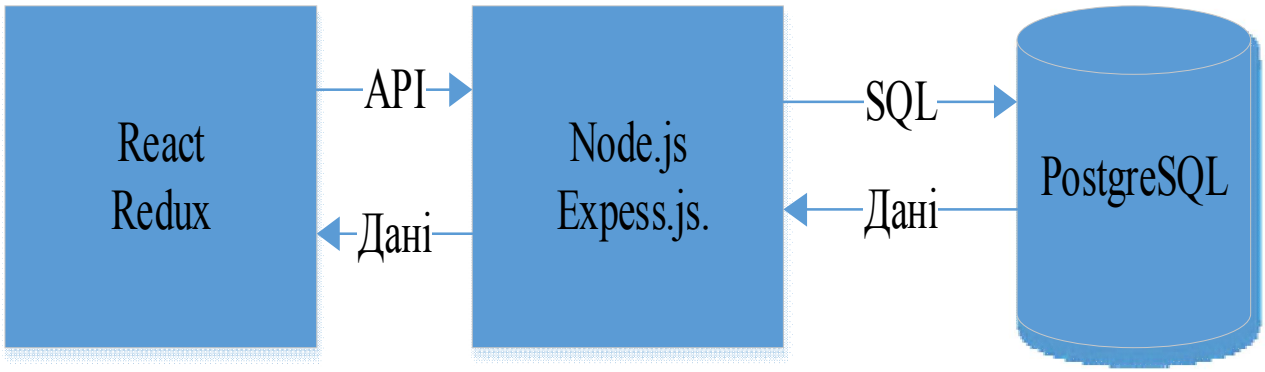

Рисунок 3.1 — Діаграма архітектури веб-платформи

Кожен рівень цієї архітектури реалізовано з використанням різних технологій. Для серверної частини веб застосунку було вирішено використовувати веб-сервер Node.js та фреймфорк Express.js. Для зберігання даних використовується реляційна система керування базами даних PostgreSQL. Для швидкої реалізації клієнтської частини з можливістю створювати гнучкий інтерфейс використано бібліотеки React та Redux, які базуються на мові програмування JavaScript. Для аутентифікації користувача системи між сайтом та сервером використовується стандарт JWT.

Реалізація веб-застосунку проводилася в середовищі Visual Studio Code.

3.1 Веб-сервер Node.js та фреймворк Express.js

Платформа Node — це платформа для розробки веб-додатків, серверів додатків і для програмування. Вона спроектована так, щоб забезпечити найвищу

масштабованість мережевих додатків — за рахунок поєднання асинхронного введення / виведення, використання JavaScript на стороні сервера та використання анонімних функцій JavaScript.

Прийнята в Node модель принципово відрізняється від поширених платформ для побудови серверних додатків, в яких масштабованість досягається за рахунок багатопоточності. Завдяки обставинно-оріентірованній архітектурі знижується споживання пам'яті, підвищується пропускна здатність і спрощується модель програмування. Наразі платформа Node швидко розвивається та приваблює альтернативою традиційного підходу до розробки веб-додатків — на базі Apache, PHP, Python і т. д.

В основі Node лежить автономна віртуальна машина JavaScript з розширеннями, що роблять її придатною для програмування загального призначення з акцентом на розробку серверів додатків. Платформу Node не має сенсу прямо порівнювати ні з мовами програмування, які, як правило, використовують для створення веб-додатків (PHP / Python / Ruby / Java та інші), ні з контейнерами, що реалізують протокол HTTP (Apache / Tomcat / Glassfish і т. д.), але потенційно вона може замінити традиційні стеки веб-додатків.

В основі реалізації лежить цикл обробки подій неблокуючого введення / виведення і бібліотеки файлового і мережевого введення / виведення, причому все це побудовано поверх движка V8 JavaScript (запозиченого з веб-браузера Chrome). Бібліотека введення / виведення має достатню подібність для реалізації будь-якого протоколу на базі TCP або UDP: DNS, HTTP, IRC, FTP та ін. Але хоча вона підтримує розробку серверів і клієнтів довільного протоколу, найчастіше застосовується для створення звичайних веб-сайтів, де замінює Apache / PHP або Rails.

Фреймворк Express є найпопулярнішим веб-фреймворком Node.js і є базовою бібліотекою поряд з іншими популярними веб-платформами Node. Він забезпечує механізми для:

— написання запитів HTTP за різними шляхами URL (маршрутами);

— інтеграції з механізмами візуалізації "view", щоб генерувати відповіді,

вставляючи дані в шаблони;

— встановлення загальних налаштувань веб-додатків, таких як порт для підключення та розташування шаблонів, які використовуються для надання відповіді;

— додавання додаткової обробки запитів “проміжне програмне забезпечення” в будь-яку точку конвеєра для обробки запитів.

Хоч сам фреймворк Express є досить мінімалістичним, розробники створили сумісні проміжні пакети для розв’язання практично будь-якої проблеми веб-розробки. Існують бібліотеки для роботи з файлами cookie, сеансами, логінами користувачів, параметрами URL, даними POST, заголовками безпеки та багатьма іншими.

3.2. Система керування базами даних PostgreSQL

Система керування базами даних PostgreSQL є вільною об’єктно-реляційною СКБД. Система PostgreSQL є гарною альтернативою комерційним СКБД, таким як Oracle Database або Microsoft SQL Server. Сьогодні система керування базами даних PostgreSQL існує в реалізаціях для різних платформ, включаючи Linux, Win32, Mac OS X, Solaris / OpenSolaris, FreeBSD, QNX 4.25, QNX 6.

Однією з найбільш сильних сторін СКБД PostgreSQL є архітектура. Як і у випадках з багатьма комерційними СКБД, PostgreSQL можна застосовувати в середовищі клієнт-сервер — це надає безліч переваг і користувачам, і розробникам.

В основі PostgreSQL — серверний процес бази даних, який виконується на одному сервері. Доступ з додатку до даних бази PostgreSQL проводиться за допомогою спеціального процесу бази даних. Тобто клієнтські програми не можуть отримувати самостійний доступ до даних навіть у тому випадку, якщо вони функціонують на тому ж ПК, на якому здійснюється серверний процес [10].

Таким чином досягається поділ клієнтів і сервера, що дає можливість створювати розподілені системи. Наприклад, можна відокремити клієнтів від

сервера за допомогою мережі, розробляючи клієнтські програми в середовищі, яке зручне для користувача.

Можна побачити, що кілька клієнтів приєднані до сервера по мережі. СКБД PostgreSQL орієнтована на протокол TCP / IP (локальна мережа або Інтернет), при цьому кожен клієнт з'єднаний з головним серверним процесом БД (на схемі цей процес називають Postmaster). Саме Postmaster створює новий серверний процес спеціально з метою обслуговування запитів на доступ до даних певного клієнта.

Оскільки маніпулювання з даними зосереджується на сервері, СКБД PostgreSQL не доводиться контролювати численних клієнтів, які отримують доступ в спільно використовуваний серверний каталог. У результаті база даних PostgreSQL здатна підтримувати цілісність даних навіть у разі одночасного доступу великої кількості користувачів.

Клієнт-серверна архітектура, реалізована в СКБД PostgreSQL, робить можливим розподіл праці. Тобто машина-сервер прекрасно підходить для зберігання і керування доступом до величезних обсягів даних, її можна використовувати як надійне сховище.

Нижче проаналізовано основні переваги і функціональні можливості СКБД PostgreSQL.

Надійність СКБД PostgreSQL перевірена і доведена. Вона забезпечується відповідністю принципам ACID (атомарність, ізольованість, несуперечливість, збереження даних), багатоверсійності, наявністю Write Ahead Logging (WAL) — загальноприйнятого механізму протоколювання всіх існуючих транзакцій.

Продуктивність у СКБД PostgreSQL заснована на застосуванні індексів, наявності гнучкої системи блокувань і інтелектуального планувальника запитів, використання системи керування буферами пам'яті і кешування.

Можливість розширення — для СКБД PostgreSQL це означає, що користувач може налаштувати систему за допомогою визначення нових функцій, типів, мов, агрегатів, індексів і операторів. А об'єктна орієнтованість СКБД PostgreSQL дає можливість переносити логіку додатка на рівень бази даних, а це, в свою чергу, помітно спрощує розробку клієнтів, оскільки вся бізнес-логіка знаходиться в БД.

При цьому функції в Postgres однозначно визначаються назвою, типами і кількістю аргументів.

Підтримка SQL — крім головних можливостей, які притаманні будь-якій SQL-базі, PostgreSQL підтримує схеми, підзапити, зовнішні зв'язки, правила, курсори, успадкування таблиць, тригери і багато вншого.

Підтримка численних типів даних — СКБД PostgreSQL підтримує числові (цілі, грошові, з фіксованою / плаваючою точкою), булеві, символьні, складові, мережеві типи даних, а також перерахування, типи "дата / час", геометричні примітиви, масиви, XML- і JSON-дані. Можна також створювати свої типи даних.

3.3. Бібліотеки React і Redux мови JavaScript

Бібліотека React [11] — це бібліотека JavaScript, яка використовується для створення інтерфейсу для користувача. Бібліотека React є ідеальним інструментом для створення масштабованих веб-додатків (мова йде про фронтенд).

Ця інтерфейсна бібліотека з відкритим початковим кодом заснована на компонентах. Бібліотека обробляє тільки користувацький інтерфейс.

Веб-додаток ReactJS складається з кількох компонентів, кожен компонент відповідає за виведення невеликого фрагменту HTML-коду, який багаторазово використовується. Компоненти — це основа всіх додатків React. Ці компоненти можуть бути вкладені в інші компоненти, що дає можливість будувати складні програми з простих будівельних блоків. Додаток ReactJS використовує механізм віртуального DOM для заповнення даних у HTML DOM. Віртуальний DOM працює швидко, оскільки змінює лише окремі елементи DOM, замість того, щоб кожен раз перезавантажувати повний DOM.

Усі веб-програми React базуються на компонентах [12-14]. Компонент — це окремий модуль, який є певним результатом. Можна використовувати компоненти для запису елементів інтерфейсу, таких як кнопка або поле введення. Компонент може містити один або кілька додаткових компонентів. Для створення програмного забезпечення потрібно писати компоненти React, які відповідають різним елементам

інтерфейсу. Потім потрібно розкласти ці компоненти на компоненти вищого рівня, які визначають структуру програми. Наприклад, якщо взяти форму. Форма може складатися з багатьох елементів інтерфейсу, таких як поля введення, мітки або кнопки. Кожен елемент всередині форми може бути записаний як компонент Resact. Потім з'являється можливість написати компонент вищого рівня, сам компонент форми. Компонент форми визначатиме структуру форми і міститиме кожен з цих елементів інтерфейсу.

Бібліотека Redux [15] представляє архітектуру додатків, які використовують React і є контейнером для керування станом додатку. Загальну схему взаємодії елементів архітектури Redux подано на рисунку 3.2.

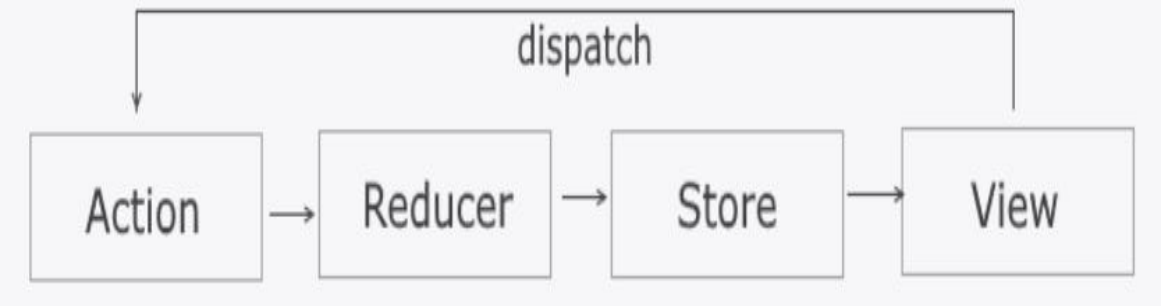

Рисунок 3.2 — Загальна схема взаємодії елементів архітектури Redux [15]

Ключовими моментами Redux є:

— сховище (store) — зберігає стан додатка;

— дії (actions) — деякий набір інформації, який виходить від додатка до сховища і який вказує, що саме потрібно зробити;

— творці дій (action creators) — функції, які створюють дії;

— reducer — функція (або кілька функцій), яка отримує дію і відповідно до цього дією змінює стан сховища.

3.4. Мова гіпертекстової розмітки HTML і мова стилів CSS

Мова HTML (HyperText Markup Language) — це мова розмітки гіпертексту, за допомогою якої створюються веб-сторінки. Використовується HTML виключно для

розмітки текстового документа.

Мова HTML не є мовою програмування, тобто вона не має можливості створювати динамічні функції. Замість цього вона дає можливість організовувати і форматувати документи, аналогічно до Microsoft Word. Мова HTML — це основа веб-сайтів, з її допомогою створюється каркас сторінки, яка відображається в браузері.

У загальному HTML — це мова розмітки, яка є дуже простою щодо освоєння навіть для початківців для створення сайтів і веб-додатків.

Технологія CSS — це мова стилів, яка визначає зовнішній вигляд HTML-документів. Мова стилів CSS відповідає за роботу з шрифтами, кольорами, полями, висотою, шириною тощо. Вона значно розширює можливості оформлення сайтів.

3.5. Протокол передачі даних HTTP

Протокол передачі гіпертексту HTTP (HyperText Transfer Protocol) — це прикладний протокол передачі даних в мережі. Він використовується для отримання інформації з веб-сайту. В основі протоколу HTTP лежить концепція клієнт-серверної архітектури: клієнт, найчастіше браузер, робить запит на сервер (рисунок 3.3) [16].

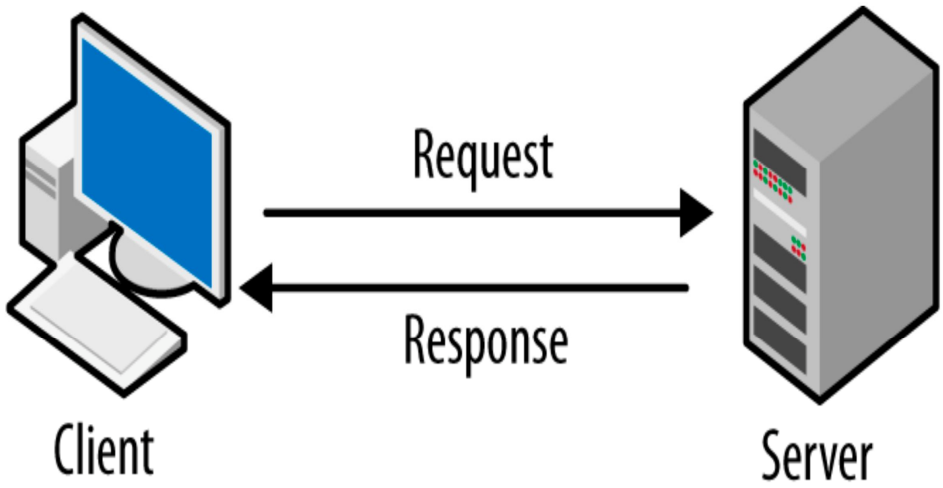

Рисунок 3.3 — Схема взаємодії клієнт-серверної архітектури [16]

Існує безліч видів запитів, найпоширеніші — це GET і POST: перший означає, що клієнт хоче отримати дані, а другий — що клієнт хоче послати дані на сервер. Таким чином, спілкування між клієнтом і сервером зводиться до обміну повідомленнями, причому завжди за принципом “клієнт надіслав запит — сервер надіслав відповідь”.

Протокол HTTP легко використовувати, він є розширюваним протоколом. Структура клієнт-сервера, разом із здатністю до простого додавання заголовків, дає можливість HTTP просуватися разом з розширювальними можливостями Мережі.

3.6. Веб-токен JSON (JWT)

Веб-маркер JSON (JWT) [17] — це відкритий стандарт, який визначає компактний та автономний спосіб безпечної передачі інформації між сторонами як об’єкт JSON. Використовувати веб-токени JSON можна при:

— авторизації — це найпоширеніший сценарій використання JWT. Після того, як користувач увійшов в систему, кожен наступний запит включатиме JWT, що дасть можливість користувачеві отримати доступ до маршрутів, послуг і ресурсів, дозволених цим маркером. Єдиний вхід — це функція, яка в наш час широко використовує JWT через невеликі накладні витрати та можливість легкого використання в різних доменах;

— обмін інформацією — веб-токени JSON — це хороший спосіб безпечної передачі інформації між сторонами, оскільки JWT можна підписувати, наприклад, використовуючи пари відкритого або приватного ключів. Крім того, оскільки підпис обчислюється за допомогою заголовка та корисного навантаження, можна бути впевненим, що вміст не зазнавав ніяких змін.

3.7. Редактор вихідного коду Microsoft Visual Code

Засіб Visual Studio Code [18] — безкоштовний і дуже популярний редактор вихідного коду, розроблений Microsoft для Windows, Linux і macOS. Редактор коду

допомагає в роботі веб-розробників всіх рівнів. З одного боку редактор підходить новачкам, тому що його інтерфейс інтуїтивно простий і зрозумілий. З іншого боку в VS Code вбудовано багато можливостей, цікавих досвідченим розробникам. Засіб Visual Studio Code включає вбудований Git і підтримку налагодження, виділення синтаксису, інтелектуальне завершення коду, фрагменти та рефакторинг коду. Він гнучко налаштовується, що дає можливість користувачам змінювати тему, комбінації клавіш, налаштування та встановлювати розширення, які додають додаткову функціональність. Вихідний код — вільний та відкритий, випущений згідно з ліцензією MIT.

4. ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБ-ПЛАТФОРМИ

Основною ціллю розробки є створення програмного продукту, який за допомогою простого та зручного інтерфейсу надасть можливість користувачу використовувати весь його функціонал.

До складу архітектури розробленої програмної платформи для організації конференцій входять такі основні компоненти:

— сервер бази даних PostgreSQL для зберігання і здійснення доступу до даних;

— сервер веб-додатку для розв’язання функціональних комплексів задач обробки даних;

— веб-клієнт користувача;

— веб-клієнт з інтерфейсом адміністратора.

4.1. Архітектура веб-платформи

В основі розробленої програмної платформи для організації конференцій лежить так звана модель взаємодії “клієнт-сервер” (рисунок 4.1), яка дає можливість розділяти функціонал і обчислювальне навантаження між клієнтськими і серверними додатками.

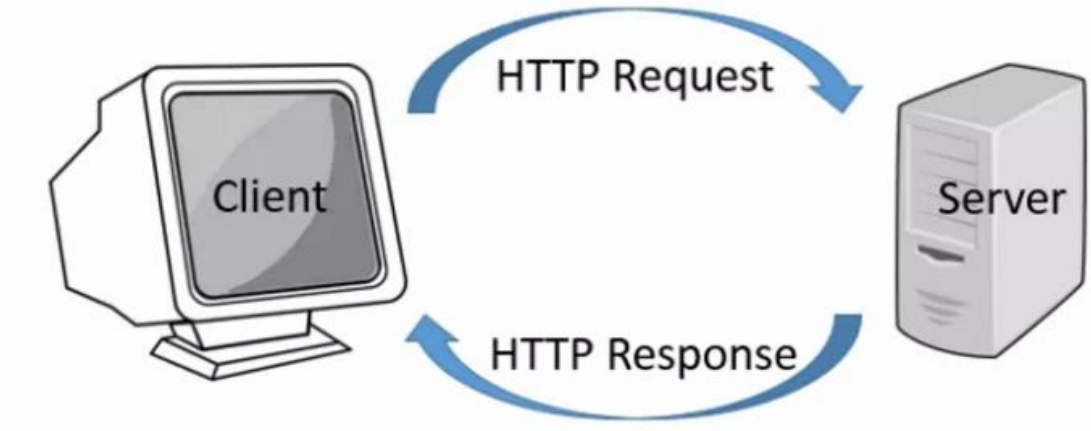

Рисунок 4.1 — Схема взаємодії клієнт-серверної архітектури [19]

“Клієнт-сервер” — концепція, яка дає можливість виконувати дії в мережі Інтернет.

Комп’ютер, з яким відбувається взаємодія через Інтернет, є веб-сервером. Клієнтський комп’ютер забезпечує інтерфейс веб-додатку, що дає можливість звичайному користувачеві комп’ютера робити запит послуги сервера і відобразити результати, які повертає сервер. Сервер чекає надходження запиту від клієнта, а потім відповідає на нього. Сервер надає стандартизований прозорий інтерфейс, так, що клієнти не знають про особливості програмного забезпечення [20].

Клієнтом у даній архітектурі є браузер, який знаходиться на персональному комп’ютері.

Ця архітектура програмної платформи дуже ефективна, тому, що клієнт і сервер мають різні завдання, які вони регулярно виконують. При обробці даних на веб-платформі на клієнтському комп’ютері запущено модуль для введення інформації про учасника конференції, у той час як на сервері працює інший програмний модуль, який керує базою даних, в якій інформація зберігається постійно.

Багато клієнтів можуть одночасно отримувати доступ до інформації сервера, а клієнтський комп’ютер може виконувати інші завдання, наприклад, створювати список електронних адрес учасників конференції.

4.2. База даних програмної платформи

Програмна веб-платформа для організації конференцій передбачає обробку даних (дані про конференції, учасників конференції або доповіді). Для зберігання і вилучення великої кількості інформації використовується реляційна база даних PostgreSQL. База даних зберігає всі записи в таблицях і має здатність зв’язувати їх.

Для реалізації веб-платформи для організації конференцій було спроектовано базу даних, структура якої показана на рисунку 4.2. На схемі зображено 6 створених таблиць, поля і типи даних полів.

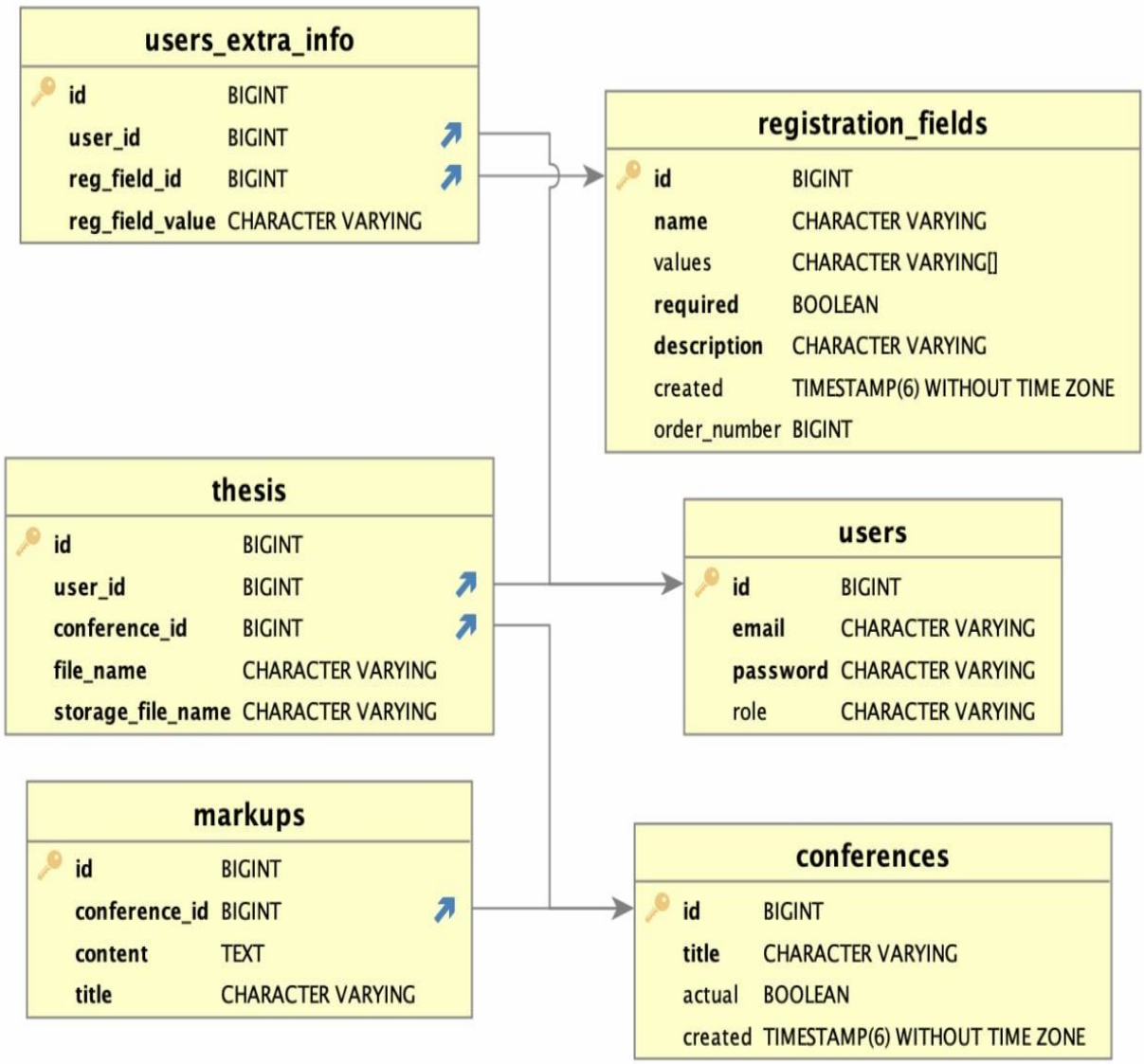

Рисунок 4.2 — Структура бази даних

Таблиця users містить основну інформацію про користувача системи: id, обов’язкові поля email та password, які вводять користувачі при реєстрації та role (адміністратор чи звичайний користувач).

Оскільки однією з головних цілей розробленої веб-платформи є збір даних про потенційних учасників конференції, використовується таблиця registration_fields, яка з’єднується з users ще однією таблицею users_extra_info. Вона призначена для того, щоб можна було додати нові поля реєстрації.

На розробленому сайті існує можливість при реєстрації користувачеві деяку інформацію вносити самому (ім’я, прізвище, номер телефона, вчений ступінь, вчене звання, місце роботи, статус), а деяку вибирати з форми (наукова секція, форма

участі). У базі даних організатор може змінити поля, видаливши або додавши в форму будь-яку кількість полів, оскільки, наприклад, з роками може змінюватися кількість і тематика секцій.

Таблиця Conferences (конференція) містить дані про назву конференції (title) і про її статус (actual). Поле actual має тип boolean і може набувати значення true тільки в одному випадку, оскільки актуальна конференція може бути тільки одна. Інші автоматично переходять в архівний режим.

Також у базі даних присутня таблиця thesis, яка містить інформацію про надіслані зареєстрованими учасниками конференції доповіді, і таблиця markup, в якій міститься шаблон і додаткові дані про Головну сторінку для кожної з конференцій.

4.3. Діаграма прецедентів системи

Веб-портал розрахований на три категорії користувачів:

— незареєстрований користувач;

— зареєстрований користувач;

— адміністратор сайту (організатор конференції).

Можливості всіх трьох категорій користувачів відображено на діаграмі прецедентів (рисунок 4.3), де в ролі акторів є зареєстрований, незареєстрований користувач та адміністратор сайту.

Діаграма прецедентів візуально зображає різноманітні сценарії взаємодії між акторами і прецедентами (випадками використання).

Відповідно до категорій розподіляється функціонал.

Незареєстрований користувач — може зареєструватися, переглядати поточну інформацію про конференцію, переглядати архів доповідей з минулих конференцій.

Зареєстрований користувач — може авторизуватися, переглядати поточну інформацію про конференцію, переглядати архів доповідей минулих конференцій, може зайти у власний кабінет і подати свою доповідь.

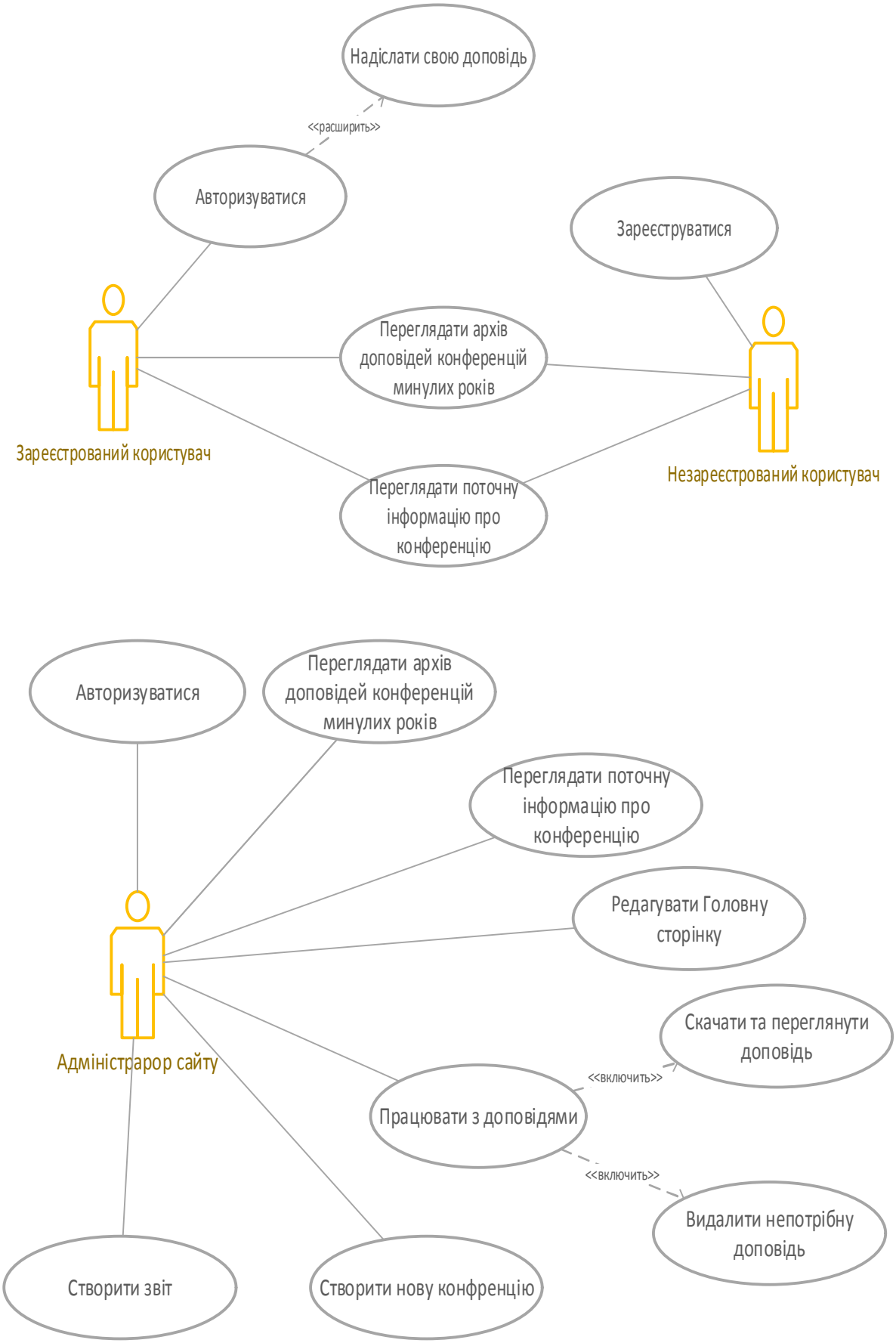

Рисунок 4.3 — Діаграма прецедентів

Адміністратор — за допомогою створеної програмної платформи може розмістити інформаційний лист конференції, приймати реєстрації учасників і доповідачів на конференцію, розмістити і редагувати програму конференції, створювати списки електронних пошт для кожної конференції для подальшого користування (e-mail-розсилки за адресами колишніх і нових учасників конференції). Також організатор має можливість створити нову конференцію, вказати час і дату проведення заходу, оперативно сповістити учасників і звичайних користувачів про певні зміни, розмістити на сайті файли різного типу: фотографії, відео-матеріали. Вказуючи місце проведення конференції, є можливість додати посилання для відео-зв’язку.

Діаграми прецедентів ідеально підходять для:

— подання цілей взаємодії системи та користувача;

— визначення та організації функціональних вимог у системі;

— визначення контексту та вимог системи;

— моделювання потоку подій у випадку використання [21].

Діаграми прецедентів відіграють важливу роль не тільки в комунікації між збирачами вимог до проекту і потенційними користувачами. Діаграми прецедентів дописані бізнес логікою і детальними специфікаціями прецедентів, як джерельна інформація, успішно використовують учасники розробки проекту на всіх його фазах (зародження, дизайн, програмування, тестування, документування).

4.4. Засіб аутентифікації користувачів

Процес взаємодії зареєстрованого користувача з системою починається з авторизації (для незареєстрованого користувача все починається з реєстрації, і ці два процеси дуже сильно взаємопов’язані).

Для перевірки автентичності користувача шляхом порівняння введеного ним пароля (для зазначеного логіна) з паролем, збереженим в базі даних, виконується процедура аутентифікації.

Програмна веб-платформа організації конференцій для виконання

аутентифікації використовує JWT-токен. Стандарт JWT — це JSON-об’єкт, який вважається одним з безпечних способів передачі інформації між двома учасниками (рисунок 4.4).

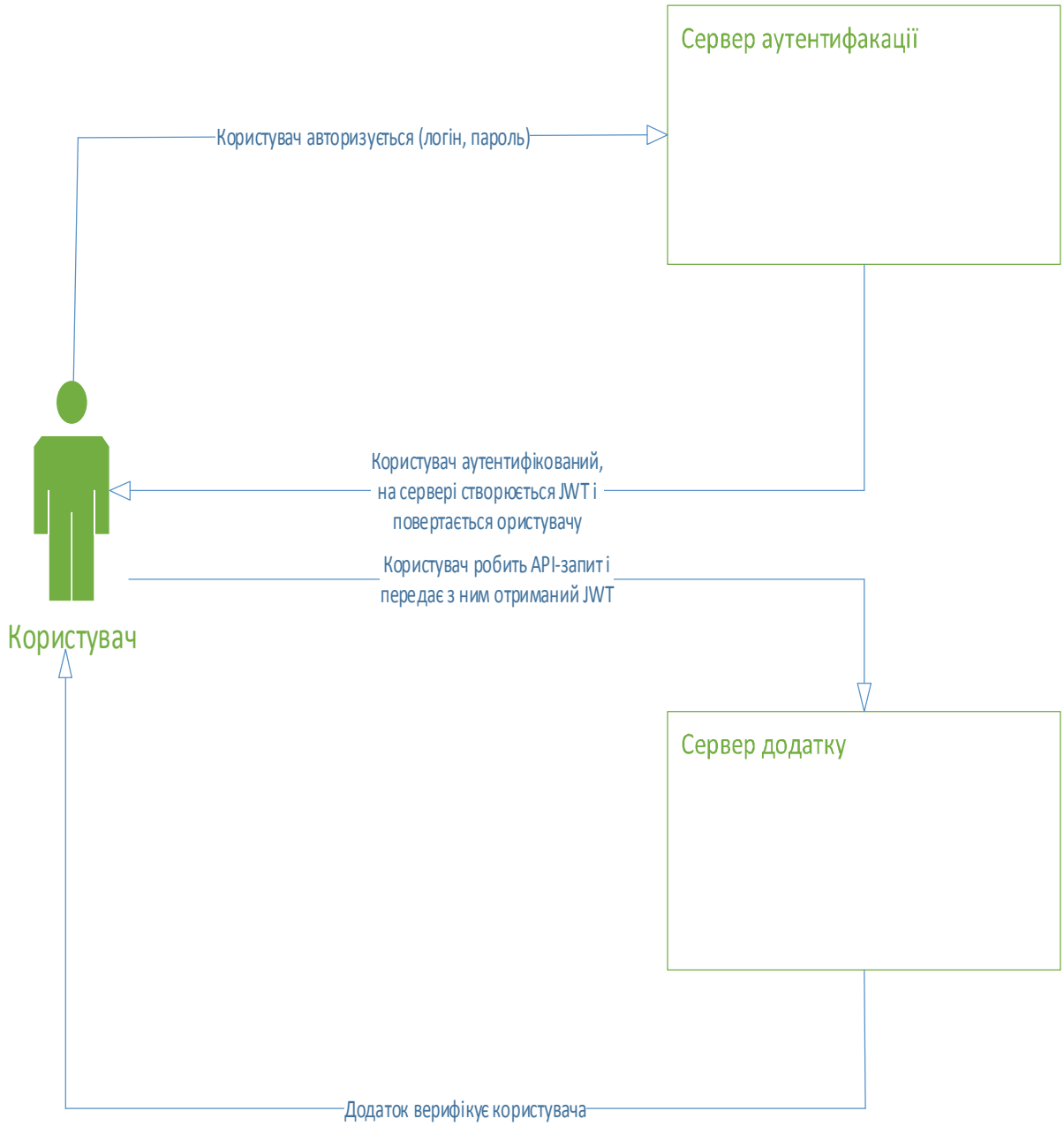

Рисунок 4.4 — Схема виконання аутентифікації користувачів

Засіб JWT відіграє важливу роль у безпеці систем аутентифікації користувачів, і представляє собою звичайний рядок коду.

Стандарт JWT для аутентифікації використовується так:

— спочатку користувач проходить звичайну аутентифікацію. Вводить пару логін-пароль;

— якщо все пройшло вдало, сервер аутентифікації створює JWT і відправляє його користувачеві назад;

— тепер користувач може взаємодіяти з додатком через API. Роблячи запит, передається також отриманий токен;

— сервер програми перевіряє, чи справді цей токен був виданий сервером аутентифікації;

— якщо JWT справжній, сервер просто виконує отриманий запит і користувач отримує потрібний йому результат.

Процес авторизації можна подати у вигляді схеми (рисунок 4.5).

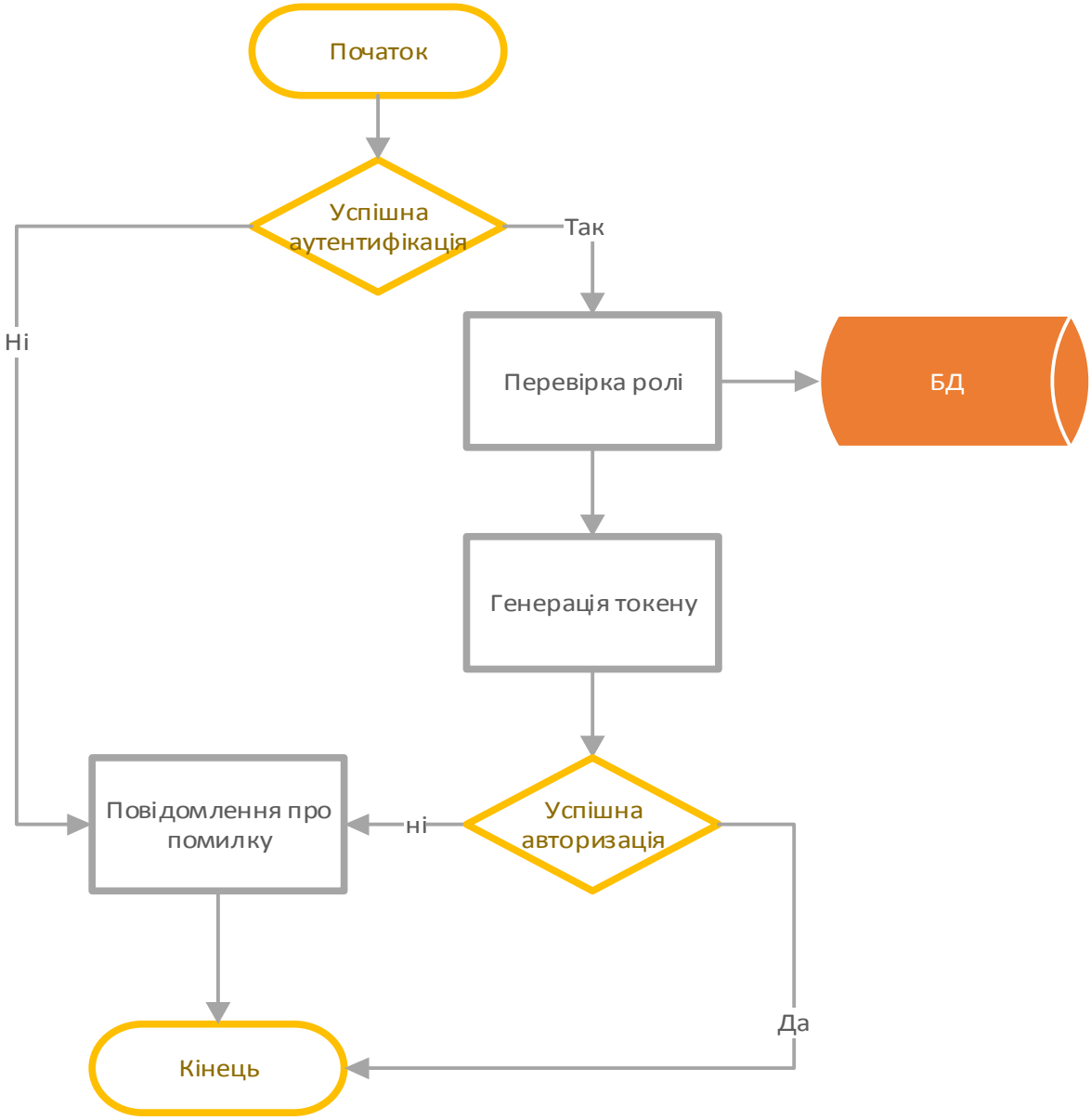

Рисунок 4.5 — Схема процесу авторизації

Схемі відповідає такий алгоритм:

— користувач з логіном та паролем успішно проходить аутентифікацію;

— сервер дізнається в базі даних про роль у користувача;

— сервер генерує користувачеві токен з вказаною роллю;

— користувач заходить на програмну платформу, використовуючи отриманий токен;

— сервер перевіряє права (роль) користувача в токені і відповідно пропускає або відхиляє запит.

4.5. Розробка засобу створення нової конференції

Метою наукових конференцій є обговорення актуальних наукових і практичних проблем, обмін досвідом і досягненнями, розвиток вчених і фахівців в науково-іноваційній діяльності, тому такі заходи потребують періодичного проведення та організації.

На створеному програмному порталі є можливість створювати нову конференцію для керування заходом.

Створювати нову конференцію може тільки адміністратор сайту. Алгоритм подано на рисунку 4.6.

Після створення нової конференції актуальна конференція набуває статусу архівної і всі доповіді, які надсилали користувачі на цю конференцію, автоматично переносяться в архів. Користувачі більше не мають можливості надсилати файли на минулу (архівну) конференцію.

Після створення нової конференції, змінюється також Головна сторінка додатку. З Головної сторінки видаляється весь контент і вона стає повністю порожньою, без тексту, новин, фотографій.

Для наповнення Головної сторінки інформацією адміністратор може не мати навичок в програмуванні, що значно полегшує використання програмної платформи.

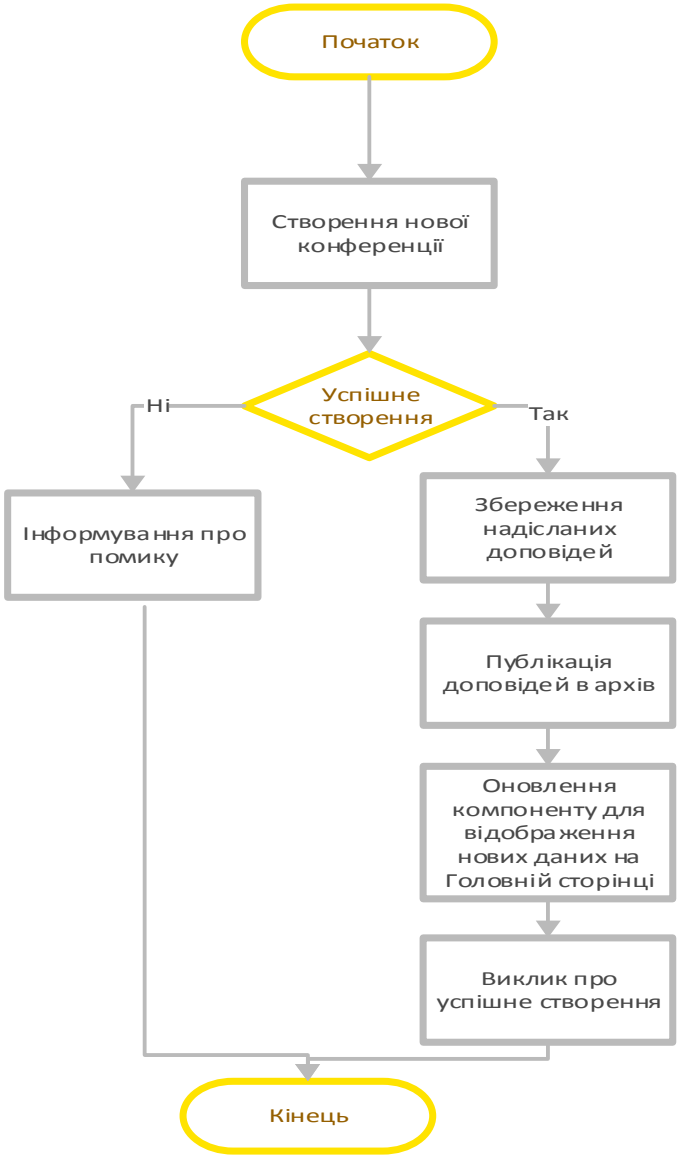

Рисунок 4.5 — Алгоритм створення конференції

При створенні нової конференції програмна платформа інформує організатора заходу про всі успішні дії або виводить повідомлення про помилку на екран комп’ютера.

5. РОБОТА КОРИСТУВАЧА З ВЕБ-ПЛАТФОРМОЮ

Для забезпечення стабільної роботи створеної веб-платформи потрібно дотримуватися вимог при її інсталяції, а також рекомендацій щодо роботи з нею. Зручний інтерфейс, створений за допомогою бібліотеки React, використовується для взаємодії користувача з платформою.

5.1. Системні вимоги

Для того, щоб почати працювати з програмною платформою необхідно встановити додаткові програми: NodeJS, PostgreSQL безспосередньо на пристрої. Після встановлення додаткового програмного забезпечення для запуску платформи треба виконати команди, прописані у файлі README.md. У результаті виконання команд на пристрої запуститься серверна частина (Back End) і клієнтська частина (Front End).

Для роботи системи на пристрої мають виконуватися такі умови:

— вільне місце на жорсткомудиску ~ 45 Гб;

— оперативна пам’ять (RAM) 2 Гб.

5.2. Використання платформи

При вході на веб-додаток користувачеві одразу надається можливість переглянути Головну сторінку програмної платформи (рисунок 5.1).

Головна сторінка веб-додатку містить основну інформацію про майбутню конференцію (мета та задачі конференції, програма заходу або останні новини).

Інформація про “Умови участі”, “Вимоги до оформлення”, “Архів”, “Комітет” та “Контакти” мають окремі вкладки на веб-сайті, для зручності користування. Приклади вигляду таких сторінок подано на рисунках 5.2-5.4.

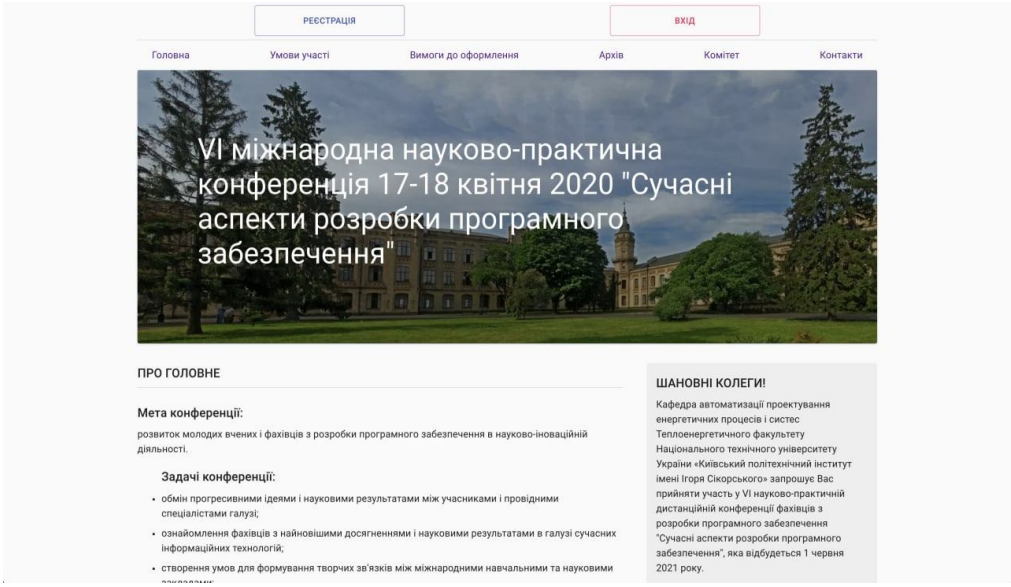

Рисунок 5.1 — Головна сторінка

Головна сторінка веб-додатку містить основну інформацію про майбутню конференцію (мета і завдання конференції, програма заходу або поточні повідомлення).

Інформація про “Умови участі”, “Вимоги до оформлення”, “Архів”, “Комітет” і “Контакти” для зручності користування розміщується на окремих вкладках веб-сайту. Приклади вигляду таких сторінок подано на рисунках 5.2-5.4.

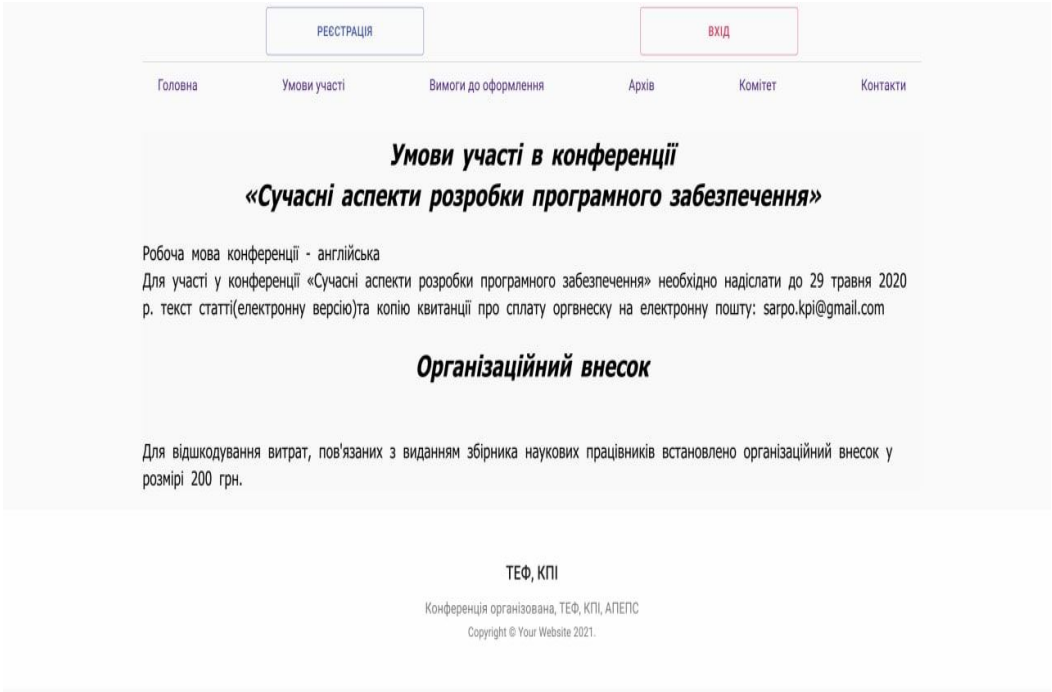

Рисунок 5.2 — Сторінка “Умови участі”

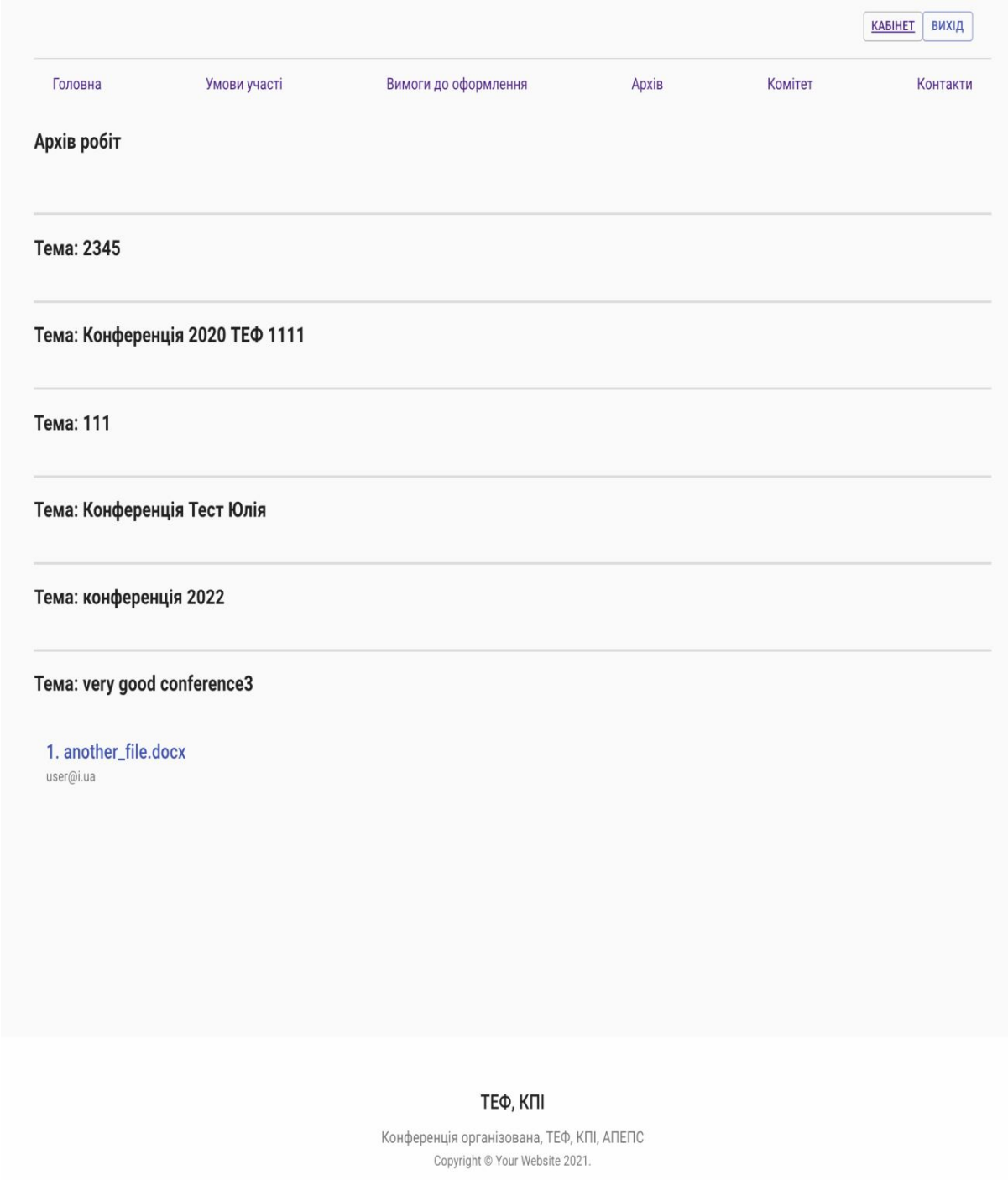

Рисунок 5.3 — Сторінка “Архів”

Якщо користувач, після перегляду детальної інформації про захід, захоче взятии участь у конференції, він повинен авторизуватися в системі. На рисунку 5.5 зображена форма авторизації.

У випадку, якщо користувач ще ніколи не використовував систему або хоче створити новий профіль, то йому необхідно зареєструватися на платформі. На рисунку 5.6 зображена форма реєстрації.

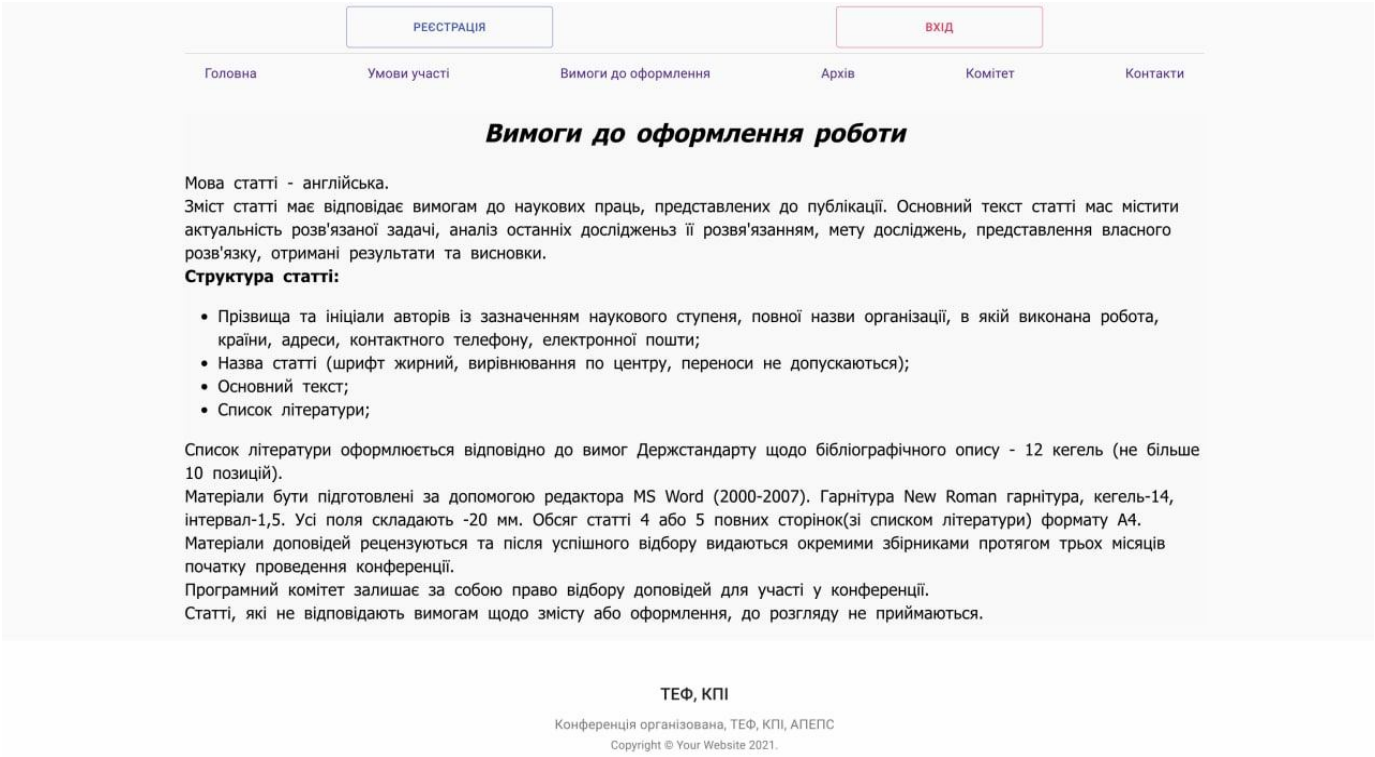

Рисунок 5.4 — Сторінка “Вимоги до оформлення роботи”

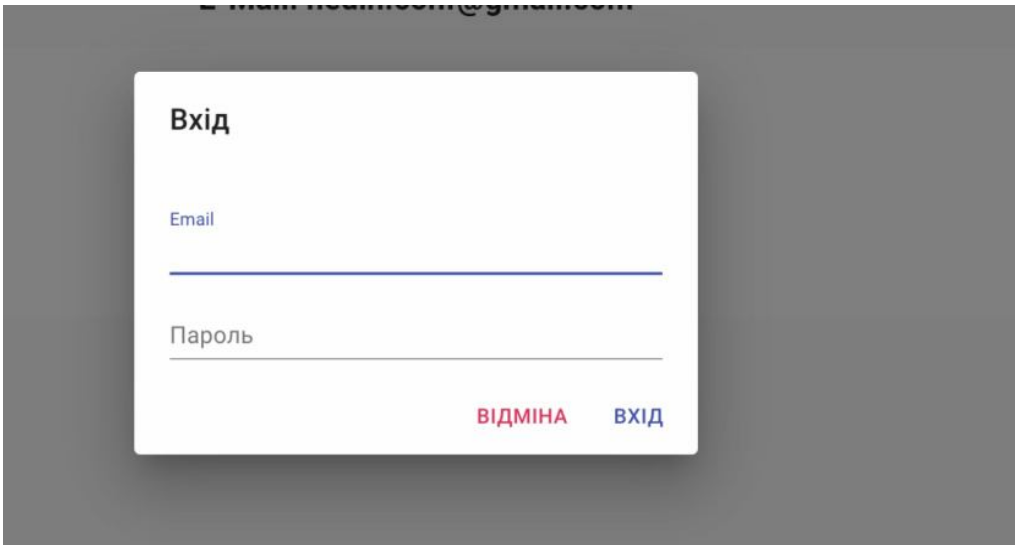

Рисунок 5.5 — Сторінка авторизації

Однією з головних задач розробленої веб-платформи є збір даних про потенційних учасників конференції. При реєстрації користувач деяку інформацію вносить сам (ім’я, прізвище, номер телефону, статус, вчений ступінь, вчене звання, місце роботи, e-mail), а деяку вибирає з форми (наукова секція, форма участі).

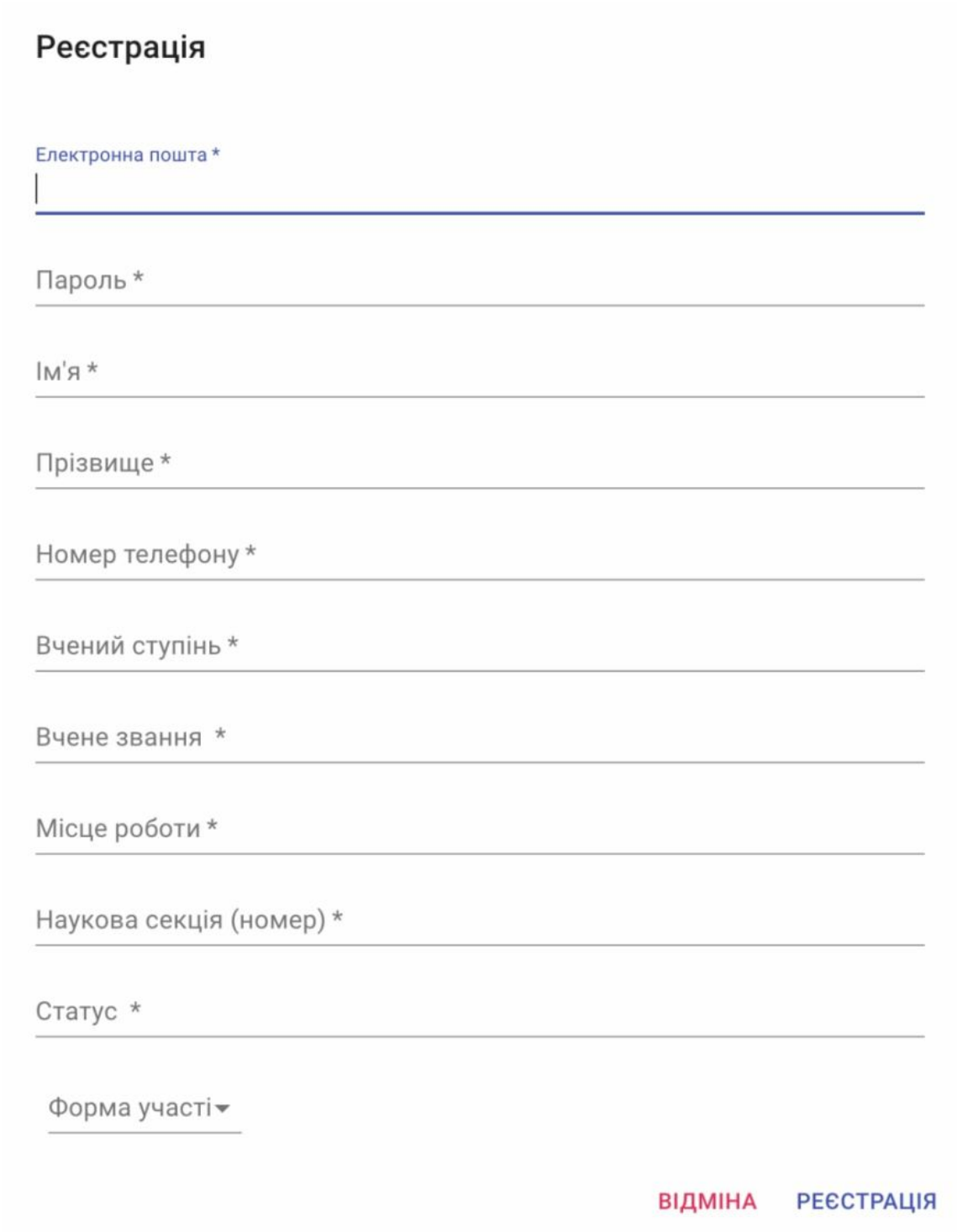

Рисунок 5.6 — Сторінка реєстрації нового користувача

Після реєстрації або авторизації система перенаправляє користувача до його особистого віртуального кабінету, де він вже може розмістити свою доповідь або тези на конференцію (рисунок 5.7).

На рисунках 5.8-5.10 подано вигляд можливостей програмної платформи зі сторони організатора (адміністратора) конференції.

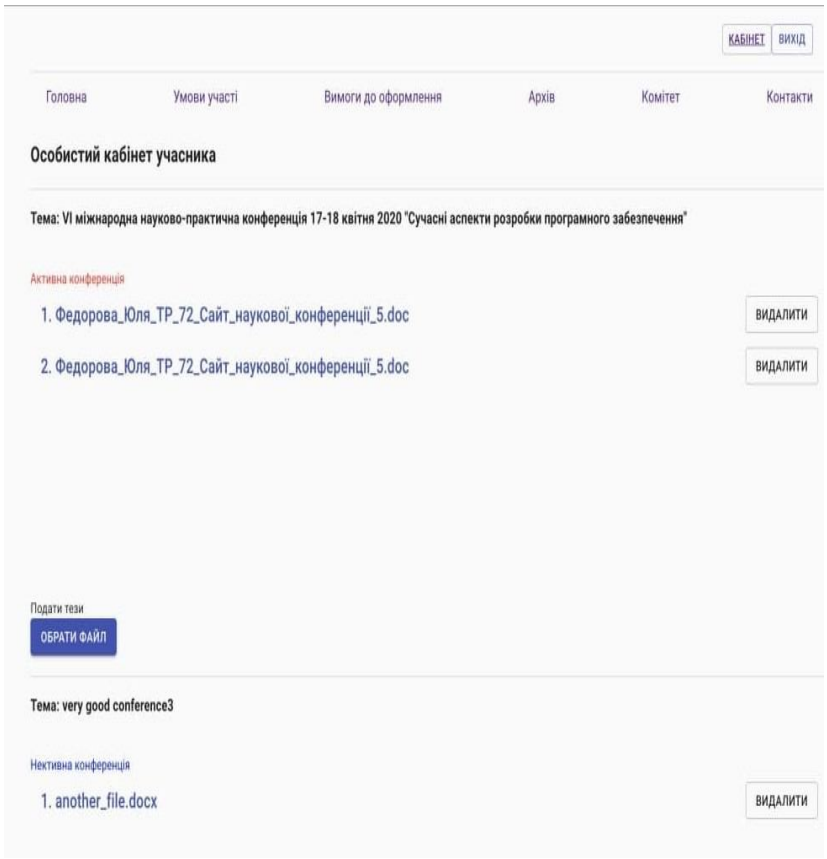

Рисунок 5.7 — Сторінка особистого кабінету користувача

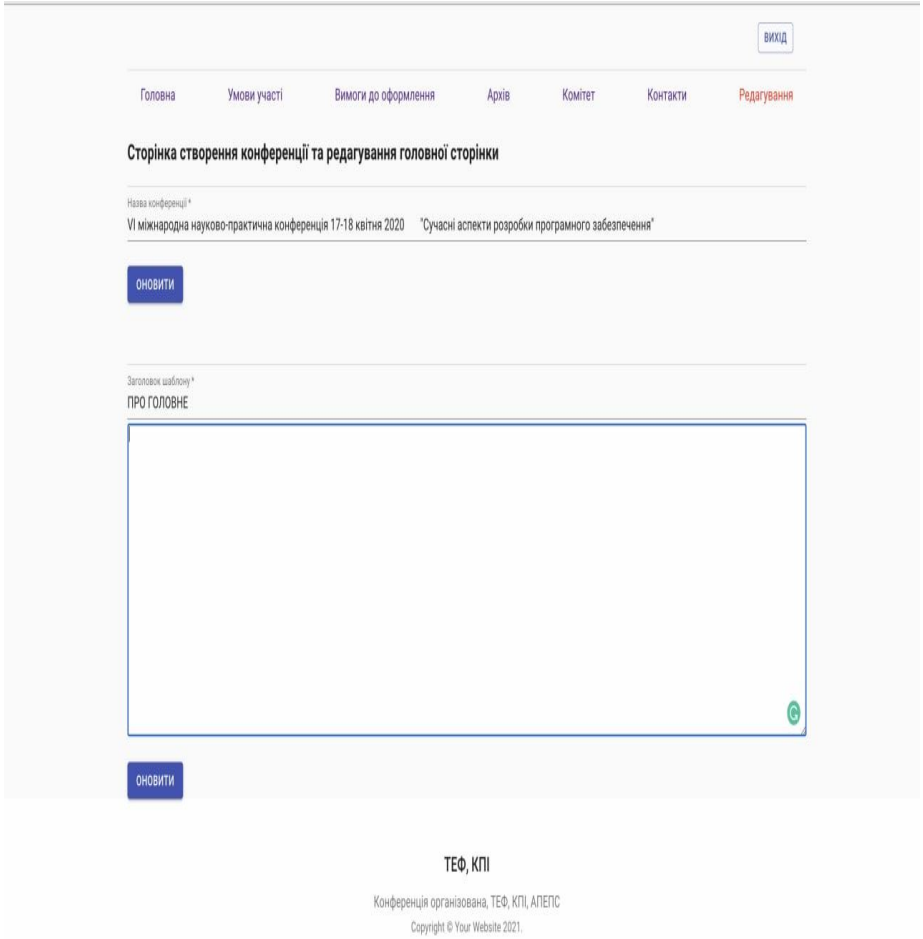

Рисунок 5.8 — Сторінка редагування Головної сторінки

Адміністратор веб-додатку авторизується за допомогою даних (email, password), які зберігаються в БД і має можливість не тільки переглядати Головну сторінку, а й редагувати її (рисунок 5.8). Наприклад, вказати час і дату проведення заходу. Вказуючи місце проведення конференції, можна додати посилання для відео-зв’язку.

Також є можливість розмістити на сайті файли різного типу: фотографії, відео-матеріали, карти місцезнаходження тощо.

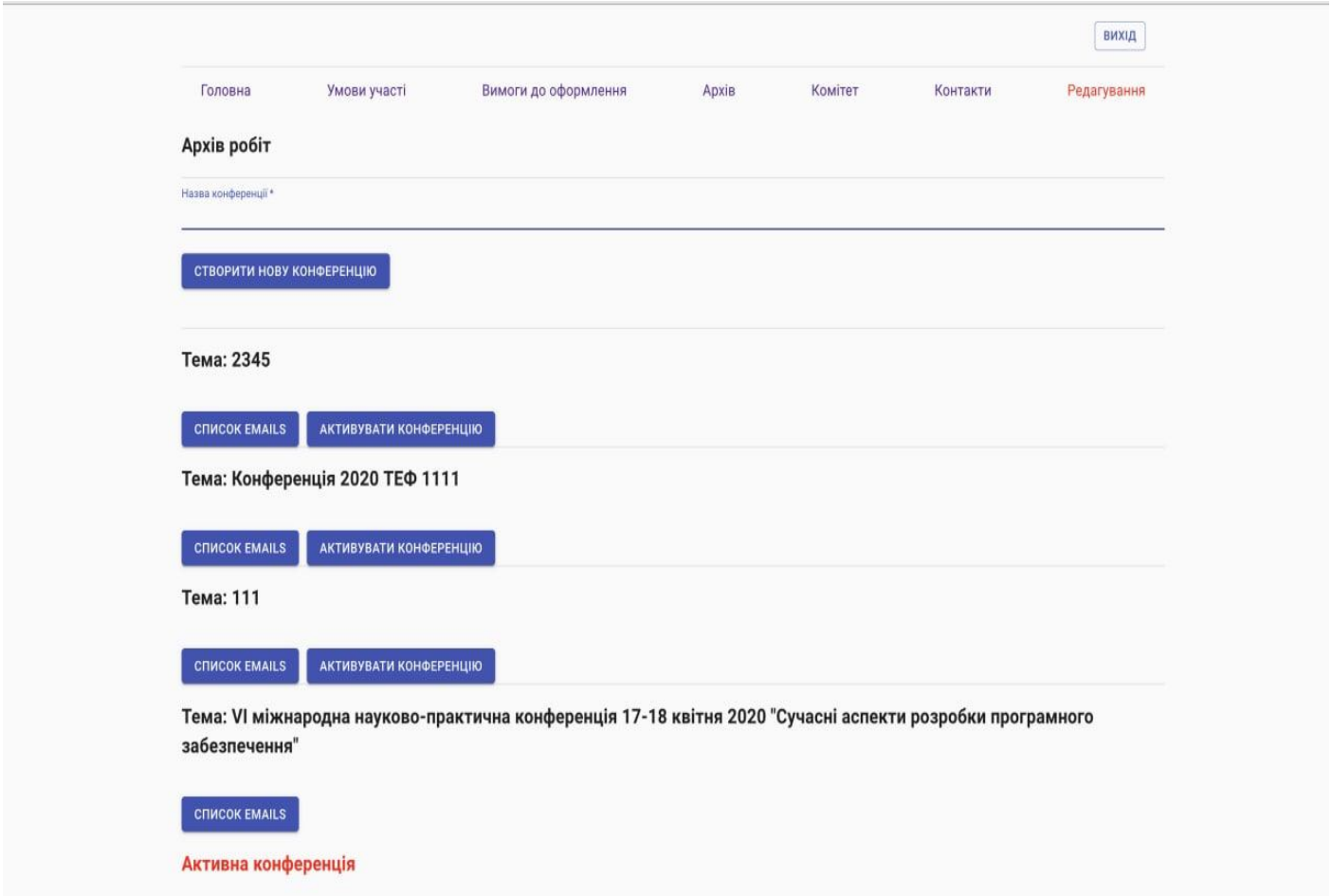

Рисунок 5.9 — Сторінка можливостей адміністратора

Використовуючи веб-додаток, адміністратор також має змогу:

— працювати (проглядати, завантажувати або видаляти) з файлами, надісланими учасниками конференції;

— створювати нову конференцію. Коли організатор створює нову конференцію, всі матеріали минулої конференції переносяться в архів;

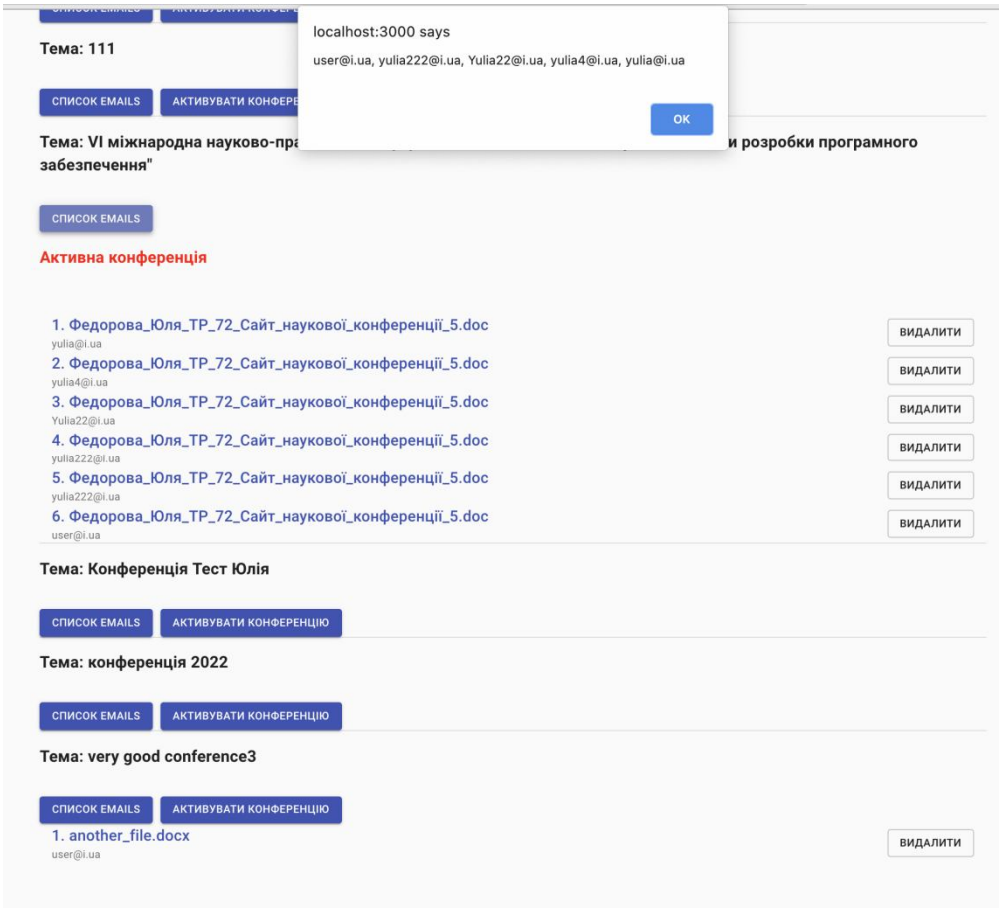

Рисунок 5.10 — Відображення списку електронних пошт

— створювати списки електронних пошт учасників поточної конференції для подальшої розсилки листів.

Розроблений інтуїтивно зрозумілий, простий і зручний інтерфейс для взаємодії користувача з веб-платформою дає можливість використовувати весь функціонал платформи.

ВИСНОВКИ

При виконанні дипломної роботи проведено аналіз і визначено переваги й недоліки існуючих сайтів науково-практичних конференцій. Розроблено програмну платформу для організації конференцій з використанням веб-технологій для організатора та учасників конференції. Функціонал платформи розрахований на три категорії користувачів: незареєстрований користувач, зареєстрований користувач, адміністратор сайту.

У розробленій платформі враховано функціонал існуючих веб-додатків, складовими якого є авторизація, реєстрація, можливість збирати різноманітні матеріали, переглядати матеріали поточної і попередніх конференцій. Також є можливість організатору створювати звіти для подальшого використання, виконувати редагування Головної сторінки, завантажувати або видаляти доповіді, надіслані користувачами — учасниками конференції. Адміністратор сайту при створенні або редагуванні заходу може прикріпити файли різного типу: зображення, опис, карту місцезнаходження, вказати час і дату проведення заходу тощо.

Впровадження розробленої програмної платформи усуває розрізнений підхід до планування заходів за рахунок централізації інформації в одному місці, дає можливість спростити організацію конференції, забезпечує передачу файлів, архівування різноманітних даних. Програмне забезпечення для керування та організації конференцій робить більше, ніж просто відображає захід на сайті, воно дає можливість організаторам розкрити всі деталі спланованих подій і в підсумку забезпечити успіх заходу.

Результати роботи доповідалися на XIX Міжнародній науково-практичній конференції молодих вчених і студентів “Сучасні проблеми наукового забезпечення енергетики” [22].

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1 Goldblatt J. Special Events: A New Generation and the Next Frontier / Joe Goldblatt // Wiley; 6-th edition, October, 5, 2010. — P. 474-475.

2 Why you need event management software [Електронний ресурс] — Режим: https://blog.bizzabo.com/why-you-need-event-management-software

3 Best event management software [Електронний ресурс] — Режим доступу: https://zapier.com/blog/best-event-management-software

4 Science-Community [Електронний ресурс] — Режим доступу: https://www.science-community.org/ru

5 EasyChair Conference System [Електронний ресурс] — Режим доступу: https://easychair.org/

6 Explara [Електронний ресурс] — Режим доступу: https://www.explara.com/

7 Splash [Електронний ресурс] — Режим доступу: https://splashthat.com/

8 4science [Електронний ресурс] — Режим доступу: https://www.4science.it/en/

9 Bizzabo [Електронний ресурс] — Режим доступу: https://www.bizzabo.com/

10 СУБД PostgreSQL. Особенности и архитектура Postgres [Електронний ресурс] — Режим доступу: https://otus.ru/nest/post/1584/

11 Banks A. Learning React: Functional Web Development with React and Redux / Alex Banks // O'Reilly Media; 1-st edition, May, 18, 2017. — P. 127-128.

12 Wieruch R. The Road to React: Your journey to master plain yet pragmatic React.js Robin Wieruch // O'Reilly Media; 1-st edition, August, 9, 2016. — P. 45.

13 Stefanov S. React: Up & Running / Stoyan Stefanov // O'Reilly Media; 1-st edition,August, 9, 2016. — P. 74-76.

14 What is React? [Електронний ресурс]/ — Режим доступу: https://www.simplilearn.com/what-is-react-article

15 Введение в Redux [Електронний ресурс] — Режим доступу: https://metanit.com/web/react/5.3.php

16 Client-Server Application [Електронний ресурс] — Режим доступу: https://madooei.github.io/cs421_sp20_homepage/client-server-app/

17 What is JSON Web Token? [Електронний ресурс]. — Режим доступу: https://jwt.io/introduction

18 Visual Studio Code [Електронний ресурс] — Режим доступу: https://code.visualstudio.com/docs

19 Client-Server Application [Електронний ресурс] — Режим доступу: https://madooei.github.io/cs421_sp20_homepage/client-server-app/

20 Software Architecture: 13th European Conference, ECSA 2019, Paris, France, September 9-13, 2019, Proceedings (Lecture Notes in Computer Science, 11681) — P. 48.

21 Use case driven object modeling with UML theory and practices Don Rosenberg and Matt Stephens Jan 11, 2007 — P. 116.

22 Федорова Ю.Є. Програмна платформа для організації конференцій з використанням веб-технологій / Ю.Є. Федорова, Л.І. Кублій // Сучасні проблеми наукового забезпечення енергетики: Матеріали XIX Міжнародної науково-практичної конференції молодих вчених і студентів, м. Київ, 20-23 квітня 2021 року. — К.: КПІ ім. Ігоря Сікорського, Вид-во “Політехніка”, 2021. — Т. 2. — С. 190-191.

ДОДАТОК А

Програмна платформа для організації конференцій з використанням веб-технологій

Специфікація

УКР.НТУУ “КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕПС_ТР72275_21Б 13-1

Аркушів 2

Київ 2021

Позначення	Найменування	Примітки
Документація		
УКР.НТУУ “КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕПС_ТР72275_21Б	Записка.docx	Текстова частина дипломної роботи
Компоненти		
УКР.НТУУ “КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕПС_ТР72275_21Б 12-1	Sarpi.api	Основний модуль серверної частини
УКР.НТУУ«КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕПС_ТР72275_21Б 12-2	Sarpo.web	Основний модуль клієнтської частини
УКР.НТУУ«КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕПС_ТР72275_21Б 13-1	Опис.docx	Опис програми

ДОДАТОК Б

Програмна платформа для організації конференцій з використанням веб-технологій

Текст програми

УКР.НТУУ “КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕПС_ТР72275_21Б 12

Аркушів 12

Київ 2021

```
var createError = require ('http-errors');
var express = require ('express');
var path = require ('path');
var cookieParser = require ('cookie-parser');
var logger = require ('morgan');
var cors = require ('cors');
var fileUpload = require ('express-fileupload');

//CRUD
var authRouter = require ('./routes/authRouter'); //TODO use this
var usersRouter = require ('./routes/usersRouter');
var usersExtraInfoRouter = require ('./routes/usersExtraInfoRouter');
var conferencesRouter = require ('./routes/conferencesRouter');
var extraInfoFieldsRouter = require ('./routes/registrationFieldsRouter');
var markupsRouter = require ('./routes/markupsRouter');
var thesisRouter = require ('./routes/thesisRouter');

//configuration
var storedFileLocation = require ('./config/storageConfig.json').location;

var app = express ();

// enable files upload
app.use (fileUpload ({
  // useTempFiles : true,
  // tempFileDir : '/tmp/',
  //safeFileNames: true,
  abortOnLimit: true,
  responseOnLimit: true,
  createParentPath: true,
  limits: {
    fileSize: 200 * 1024 * 1024 * 1024 //200MB max file (s) size
  },
}));

// view engine setup
app.set ('views', path.join (__dirname, 'views'));
app.set ('view engine', 'jade');

app.use (logger ('dev'));
app.use (cors ());
app.use (express.json ());
app.use (express.urlencoded ({ extended: false }));
```

```
app.use (cookieParser ());
app.use (express.static (path.join (__dirname, 'public')));
app.use ('/thesis', express.static (path.join (__dirname, storedFileLocation)));

// app.use ('/', indexRouter);
app.use ('/auth', authRouter)
app.use ('/users', usersRouter);
app.use ('/users/extra/info/', usersExtraInfoRouter);
app.use ('/conferences', conferencesRouter);
app.use ('/registration/fields', extraInfoFieldsRouter);
app.use ('/markups', markupsRouter);
app.use ('/thesis', thesisRouter);

// catch 404 and forward to error handler
app.use (function  (req, res, next) {
  next (createError (404));
});

// error handler
app.use (function  (err, req, res, next) {
  // set locals, only providing error in development
  res.locals.message = err.message;
  res.locals.error = req.app.get ('env') === 'development' ? err : {};

  // render the error page
  res.status (err.status || 500);
  res.render ('error');
});

app.listen (process.env.PORT || '9000');

const Pool = require ('pg').Pool
const dbConnectionConfig = require ('../config/dbConfig.json')
const pool = new Pool (dbConnectionConfig)

const getConferences =  (req, res) => {
    pool.query ('SELECT * FROM conferences ORDER BY created DESC',  (error, results) => {
        if  (error) {
            console.error (error);
            res.status (500).send (error);
            return;
        }
        res.status (200).json (results.rows)
```

```
    })
}

const getConferenceEmails =  (req, res) => {
    const id = parseInt (req.params.id)

    pool.query ('select distinct u.email from thesis t left join users u on t.user_id=u.id where
t.conference_id = $1',
        [id],  (error, results) => {
            if  (error) {
                console.error (error);
                res.status (500).send (error);
                return;
            }
            res.status (200).json (results.rows)
        })
}

const getActiveConference =  (req, res) => {
    pool.query ('SELECT * FROM conferences WHERE actual LIMIT 1',  (error, results) => {
        if  (error) {
            console.error (error);
            res.status (500).send (error);
            return;
        }
        res.status (200).json (results.rows[0])
    })
}

const getConferencesArchivalData =  (req, res) => {
    pool.query (`SELECT conf.id,
                max (title) as conf_title,
                bool_or (actual) as actual,
                json_agg (json_build_object ('id', t.id,
                                    'file_name', t.file_name,
                                    'storage_file_name', t.storage_file_name,
                                    'user_id', u.id,
                                    'user_email', u.email
                                ))
                as thesis
            FROM conferences conf
                left join thesis t on  conf.id=t.conference_id
                left join users u on t.user_id = u.id
```

```
            GROUP BY conf.id
            ORDER BY conf.created DESC`,
        (error, results) => {
          if (error) {
            console.error (error);
            res.status (500).send (error);
            return;
          }
          res.status (200).json (results.rows)
        })
}

const getConferenceWithThesisByUserId =  (req, res) => {
   const userId = parseInt (req.params.userId)


   pool.query ("SELECT conf.id, max (title) as conf_title, bool_or (actual) as actual, json_agg
(json_build_object ('id', t.id, 'file_name', t.file_name, 'storage_file_name', t.storage_file_name)) as
thesis"
      + " FROM conferences conf left join thesis t on  conf.id=t.conference_id where t.user_id=$1 group
by conf.id", [userId],  (error, results) => {
          if (error) {
            console.error (error);
            res.status (500).send (error);
            return;
          }
          res.status (200).json (results.rows)
        })
}

const getConferenceById =  (req, res) => {
   const id = parseInt (req.params.id)


   pool.query ('SELECT * FROM conferences WHERE id = $1', [id],  (error, results) => {
      if (error) {
        console.error (error);
        res.status (500).send (error);
        return;
      }
      res.status (200).json (results.rows)
   })
}

// const createConference =  (req, res) => {
//     const { title } = req.body
```

```
//    pool.query ('INSERT INTO conferences  (title) VALUES  ($1)', [title],  (error, results) => {
//       if  (error) {
//          console.error (error);
//          res.status (500).send (error);
//          return;
//       }
//       res.status (201).send (Created conference with title: ${title})
//    })
// }

const createConference =  (req, res) => {
   const { title } = req.body;
    (async  () => {
      // note: we don't try/catch this because if connecting throws an exception
      // we don't need to dispose of the client  (it will be undefined)
      const client = await pool.connect ()
      try {
         await client.query ('BEGIN')
         const result = await client.query ('INSERT INTO conferences  (title) VALUES  ($1)
RETURNING id', [title]);
         const newConfId = result.rows[0].id;
         await client.query ('INSERT INTO markups  (conference_id, title, content) VALUES  ($1, $2,
$3)',
            [newConfId, title, "Про конференцію"])
         await client.query ('COMMIT')
         res.status (200).send (Conference created with ID: ${newConfId});
      } catch  (error) {
         console.log (error)
         await client.query ('ROLLBACK')
         res.status (500).send (error);
      } finally {
         client.release ()
      }
   }) ().catch (error => {
      console.log (error)
      console.error (error.stack);
      res.status (500).send (error);
   })
}

const updateConference =  (req, res) => {
   const id = parseInt (req.params.id)
```

```
    const { title } = req.body

    pool.query (
        'UPDATE conferences SET title = $1 WHERE id = $2',
        [title, id],
        (error, results) => {
            if (error) {
                console.error (error);
                res.status (500).send (error);
                return;
            }
            res.status (200).send (Conference modified with ID: ${id})
        }
    )
}

const deleteConference =  (req, res) => {
    const id = parseInt (req.params.id)

    pool.query ('DELETE FROM conferences WHERE id = $1', [id],  (error, results) => {
        if (error) {
            console.error (error);
            res.status (500).send (error);
            return;
        }
        res.status (200).send (Conference deleted with ID: ${id})
    })
}

const activateConference =  (req, res) => {
    const id = parseInt (req.params.id);
    (async () => {
        // note: we don't try/catch this because if connecting throws an exception
        // we don't need to dispose of the client (it will be undefined)
        const client = await pool.connect ()
        try {
            await client.query ('BEGIN')
            await client.query ('UPDATE conferences SET actual = false WHERE true')
            await client.query ('UPDATE conferences SET actual = true WHERE id=$1', [id])
            await client.query ('COMMIT')
            res.status (200).send (Conference activated ted with ID: ${id});
        } catch (error) {
            await client.query ('ROLLBACK')
```

```
            res.status (500).send (error);
        } finally {
            client.release ()
        }
    }) ().catch (error => {
        console.error (error.stack);
        res.status (500).send (error);
    })
}


module.exports = {
    getConferences,
    getConferenceEmails,
    getActiveConference,
    getConferencesArchivalData,
    getConferenceWithThesisByUserId,
    getConferenceById,
    createConference,
    updateConference,
    deleteConference,
    activateConference
}
const Pool = require ('pg').Pool
const dbConnectionConfig = require ('../config/dbConfig.json')
const pool = new Pool (dbConnectionConfig)

const getMarkups =  (req, res) => {
    pool.query ('SELECT * FROM markups ORDER BY id ASC',  (error, results) => {
        if  (error) {
            console.error (error);
            res.status (500).send (error);
            return;
        }
        res.status (200).json (results.rows)
    })
}

const getMarkupById =  (req, res) => {
    const id = parseInt (req.params.id)

    pool.query ('SELECT * FROM markups WHERE id = $1', [id],  (error, results) => {
        if  (error) {
```

```
        console.error (error);
        res.status (500).send (error);
        return;
      }
      res.status (200).json (results.rows)
    })
}

const getActiveConferenceMarkup =  (req, res) =>{
    (async  () => {
      // note: we don't try/catch this because if connecting throws an exception
      // we don't need to dispose of the client  (it will be undefined)
      const client = await pool.connect ()
      try {
        let result = {};
        const confRows = await client.query ('SELECT * FROM conferences WHERE actual = true
LIMIT 1');
        const conf = confRows.rows[0];
        if (conf!== undefined){
          const confId = conf.id;
          const markup = await client.query ('SELECT * FROM markups WHERE conference_id = $1
LIMIT 1', [confId]);
          result = markup.rows[0];
          result.confTitle = conf.title;
        }
        res.status (200).json (result);
      } catch  (error) {
        console.error (error);
        res.status (500).send (error);
      } finally {
        client.release ()
      }
    }) ().catch (error => {
      console.error (error);
      res.status (500).send (error);
    })
}

const createMarkup =  (req, res) => {
    const { conference_id, title, content } = req.body

    pool.query ('INSERT INTO markups  (conference_id, title, content) VALUES  ($1, $2, $3)',
[conference_id, title, content],  (error, results) => {
      if  (error) {
```

```
            console.error (error);
            res.status (500).send (error);
            return;
        }
        res.status (201).send (Created user with title: ${title})
    })
}

const updateMarkup =  (req, res) => {
    const id = parseInt (req.params.id)
    const { title, content } = req.body

    pool.query (
        'UPDATE markups SET title = $1, content = $2 WHERE id = $3', [title, content, id],
        (error, results) => {
            if  (error) {
                console.error (error);
                res.status (500).send (error);
                return;
            }
            res.status (200).send (Markup modified with title: ${title})
        }
    )
}

const deleteMarkup =  (req, res) => {
    const id = parseInt (req.params.id)

    pool.query ('DELETE FROM markups WHERE id = $1', [id],  (error, results) => {
        if  (error) {
            console.error (error);
            res.status (500).send (error);
            return;
        }
        res.status (200).send (Markup deleted with ID: ${id})
    })
}


module.exports = {
    getMarkups,
    getMarkupById,
    getActiveConferenceMarkup,
```

```
    createMarkup,
    updateMarkup,
    deleteMarkup,
}
const Pool = require ('pg').Pool
const dbConnectionConfig = require ('../config/dbConfig.json')
const pool = new Pool (dbConnectionConfig)

const getRegistrationFields =  (req, res) => {
    pool.query ('SELECT * FROM registration_fields ORDER BY order_number ASC',  (error, results)
=> {
        if  (error) {
            console.error (error);
            res.status (500).send (error);
            return;
        }
        res.status (200).json (results.rows)
    })
}

const getRegistrationFieldById =  (req, res) => {
    const id = parseInt (req.params.id)

    pool.query ('SELECT * FROM registration_fields WHERE id = $1', [id],  (error, results) => {
        if  (error) {
            console.error (error);
            res.status (500).send (error);
            return;
        }
        res.status (200).json (results.rows)
    })
}

const createRegistrationField =  (req, res) => {
    let { name, values, required, description } = req.body
    if (required==undefined){
        required = false;
    }

    pool.query ('INSERT INTO registration_fields  (name, values, required, description) VALUES  ($1,
$2, $3, $4)',
    [name, values, required, description],  (error) => {
        if  (error) {
```

```
            console.error (error);
            res.status (500).send (error);
            return;
        }
        res.status (201).send (Created registration field with title: ${name})
    })
}
```

ДОДАТОК В

Програмна платформа для організації конференцій з використанням веб-технологій

Опис програми

УКР.НТУУ “КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕПС_ТР72275_21Б 13-1

Аркушів 8

Київ 2021

АНОТАЦІЯ

Даний додаток містить опис двох основних модулів розроблених для програмної платформи. Створені модулі показують взаємодію сервера та клієнта.

При розробці обох модулів платформи використовувалась мова JavaScript у середовищі написання коду — Microsoft Visual Studio Code.

ЗМІСТ

ЗАГАЛЬНІ ВІДОМОСТІ

У цьому додатку міститься опис програмної платформи для організації конференцій з використанням веб-технологій. У додатку Б міститься програмний код основних модулів розроблюваної програмної платформи.

Розроблений додаток може працювати кросс-платформенно але потребує додаткового встановлення на пристрій сервісу Docker.

При розробці обох модулів платформи використовувалась мова JavaScript у середовищі написання коду — Microsoft Visual Studio Code.

-5-

ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Розроблений додаток виконує завдання організації конференцій за допомогою веб-технологій. За допомогою розробленої платформи можна легко і просто створити нову конференцію, переглянути інформацію про поточну конференцію та архів минулих, відправити доповіді організатору конференції, створювати списки електронних пошт учасників конференції.

-6-

ОПИС ЛОГІЧНОЇ СТРУКТУРИ

Програмна платформа складається з клієнтської частини та серверної частини. Взаємодія клієнта та сервера відбувається звичайним HTTP протоколом.

Загальний принцип роботи додатку наступний:

1) Користувач відправляє дані для створення конференції на сервер методом POST.
2) Нова конференція створюється та повертається на клієнт.
3) Користувач натискає на елемент списку конференцій для детального перегляду.
4) На сервер відправляється запит методом GET.
5) PostgresSQL запитом знаходиться відповідний документ.
6) Результат запиту повертається на клієнт.
7) Користувач натискає на кнопку для створення списку електронних пошт учасників конференції.
8) На сервер відправляється запит методом POST.
9) Електронні пошти передаються на сервіс та результат повертається на клієнт.

-7-

ВИКОРИСТОВУВАНІ ТЕХНІЧНІ ЗАСОБИ

Для користування програмною платформою потрібен комп’ютер з оперативною пам’яттю не менше 2 Гб, вільне місце на жорсткому диску 45 Гб; веб-браузер останніх версій (на будь-якій доступній операційній системі) Google Chrome, Mozilla Firefox, Opera, Safari. Розробка і тестування під інші браузери не проводилась.

-8-

ВХІДНІ І ВИХІДНІ ДАНІ

Вхідні дані для веб-порталу: текстова інформація, файли.

Вихідні дані для веб-порталу: записи в таблиці бази даних.

ДОДАТОК Г

Програмна платформа для організації конференцій з використанням веб-технологій

АПРОБАЦІЯ

Міжнародна науково-практична конференція “Сучасні проблеми наукового забезпечення енергетики”: Матеріали XIX Міжнародної науково-практичноїконференції молодих вчених і студентів, 20-23 квітня 2021 року, м. Київ

УКР.НТУУ “КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕПС_ТР72275_21Б

Аркушів 5

2021

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

СУЧАСНІ ПРОБЛЕМИ НАУКОВОГО ЗАБЕЗПЕЧЕННЯ ЕНЕРГЕТИКИ

Матеріали XIX Міжнародної
науково-практичної конференції
молодих вчених і студентів
м. Київ, 20–23 квітня 2021 року

ТОМ 2

Київ- 2021

УДК 004.42

Студент 4 курсу, гр. ТР-72 Федорова Ю.Є.
Доц., к.т.н. Кублій Л.І.

ПРОГРАМНА ПЛАТФОРМА ДЛЯ ОРГАНІЗАЦІЇ КОНФЕРЕНЦІЙ З ВИКОРИСТАННЯМ ВЕБ-ТЕХНОЛОГІЙ

Програмне забезпечення для організації наукових конференцій з використанням веб-технологій — це засіб, який надає можливість спланувати всі заходя, швидко і якісно підготувати і провести конференцію, забезпечити потенційних учасників і всіх зацікавлених осіб інформаційними матеріалами, надати доступ до поточних і архівних матеріалів конференції, використовуючи можливості комп'ютера.

Університети і науково-дослідні інститути регулярно проводять наукові конференції, тому для успіху цих заходів потрібна добре спланована організація заходу, вибір місця проведення і керування учасниками [1].

Основне завдання даного рішення — не тільки надати зручний інструмент для підготовки конференцій, а й можливість зберегти і надати доступ до інформації з раніше проведених конференцій. При цьому як організатори, так і учасники конференції можуть оперативно взаємодіяти і обмінюватися даними. Наприклад, обговорювати процес підготовки до конференції, складати програму конференції, розміщувати на сайті збірники доповідей, презентації, фотографії, відео тощо.

Організаторам при підготовці і проведенні різноманітних заходів без використання спеціалізованих програмних платформ для керування заходами необхідно було застосовувати для класичного проектування веб-сайтів такі сторонні інструменти, як, наприклад, WordPress або Squarespace [2]. На даний час існують десятки додатків для планування заходів, серед яких найпопулярнішими є Monday, Eventzilla, Explara, Eventbrite, Bizzabo. Вони дають можливість зробити все — від залучення відвідувачів до цифрової візуалізації карт місцевості і планів приміщень. І хоч усі вони пропонують абсолютно різні функції, у них є одна спільна риса: вони виключають підхід "зроби сам" і оптимізують процес керування подіями [3].

При розробці програмної платформи для організації конференцій було враховано функціонал вказаних вище додатків для керування заходами.

Підготовка конференції потребує дуже багато часу і зусиль. За допомогою створеної програмної платформи організатор зможе розмістити інформаційний лист конференції, приймати реєстрації учасників і доповідачів на конференцію, розмістити і редагувати програму конференції, здійснювати e-mail-розсилки за адресами колишніх і нових учасників конференції. Також організатор має можливість вказати час і дату проведення заходу, оперативно сповістити учасників і звичайних слухачів про певні зміни, розмістити на сайті файли різного типу: фотографії, відео-матеріали, карти місцезнаходження тощо. Вказуючи місце проведення конференції, є можливість додати посилання для відео-зв'язку.

Однією з головних складових розробленої веб-платформи є збір даних про потенційних учасників конференції. При реєстрації користувач деяку інформацію вносить сам (ім'я, прізвище, статус, вчений ступінь, вчене звання, місце роботи, e-mail), а деяку вибирає з форми (наукова секція, форма участі). Починаючи зі стандартної форми реєстрації за замовчуванням, організатор може змінити її, видаливши або додавши в форму будь-яку кількість полів, оскільки, наприклад, з роками може змінюватися кількість і тематика секцій. Після реєстрації або авторизації система перенаправляє користувача до його особистого віртуального кабінету. Коли організатор завершує свій захід, всі матеріали конференції переносяться в архів і надається можливість створення нової конференції.

Розроблений програмний додаток має інтуїтивно зрозумілий інтерфейс, що надає можливість роботи з ним користувачам з різними рівнями підготовки. Інтерфейс розроблено у вигляді меню з вкладками: головна сторінка для розміщення різноманітної інформації організаційного характеру (інформаційного листа, програми конференції, поточних оголошень тощо), матеріали поточної конференції, архів.

Перед початком роботи з системою користувач має можливість обрати мову, з якою він буде працювати в подальшому — українську чи англійську. На вкладці головної сторінки є посилання на сторінку редагування персональних даних. На сайті наявний інструмент зворотного зв'язку — чат для забезпечення можливості учасникам письмово задавати питання, а організаторам відповідати. Це допомагає підвищити інтерес до заходу серед учасників і є зручною формою збору даних для організаторів. Кожна конференція умовно має два статуси: активний і архівний. Активний етап — це, по суті, сайт поточної конференції, опублікований в Інтернеті. Конференція переводиться в архів, коли дана подія відбулася. Архів фільтрує заходи за датою і містить інформацію про завершену конференцію, яку можуть переглядати всі відвідувачі.

При розробці модулів веб-платформи використано такі програмні засоби: для написання клієнтської частини — мову розробки веб-додатків Typescript (розширює можливості мови JavaScript, додаючи статичні визначення типів, і компілюється в JavaScript) і бібліотеку ReactJS (бібліотека JavaScript, призначена для створення швидких і інтерактивних користувацьких інтерфейсів для веб-додатків); для серверної частини — платформа Node.js (платформа з відкритим кодом для виконання високопродуктивних мережевих застосунків, написаних мовою JavaScript); для створення бази даних — документо-орієнтовану систему керування базами даних з відкритим вихідним кодом MongoDB (лежить в основі програмного забезпечення для організації конференції, оскільки вона об'єднує реєстрацію, списки розсилки тощо). У системі MongoDB застосовується модель даних, орієнтована на документи, і неструктурована мова запитів, яка не використовує звичайні рядки і колонки; це архітектура, заснована на колекціях і документах [4].

Для того, щоб користувач, який не має навичок програмування, зміг керувати сайтом, використано систему керування контентом Joomla. За допомогою панелі керування організатор конференції зможе працювати з сайтом: оновлювати, видаляти, додавати, прикріплювати контент. Модулі Joomla дають можливість створювати і змінювати блоки даних в потрібних місцях сторінки. Це може бути форма авторизації, форма зворотного зв'язку або поточне оголошення.

Впровадження розробленої програмної платформи усуває розрізнений підхід до планування заходів за рахунок централізації інформації в одному місці, дає можливість спростити організацію конференцій, забезпечує передачу текстових повідомлень і файлів, архівування різноманітних даних. Програмне забезпечення для керування та організації конференцій не просто відображає захід на сайті, воно дає можливість організаторам розкрити всі деталі спланованих подій і забезпечити оперативність надання інформації, а в результаті і успіх проведення заходу.

Перелік посилань:

1. Goldblatt J. Special Events: A New Generation and the Next Frontier / Joe Goldblatt // Wiley; 6-th edition, October, 5, 2010. — P. 474-475.

2. Why you need event management software [Електронний ресурс]. — Режим доступу до ресурсу: https://blog.bizzabo.com/why-you-need-event-management-software

3. Best event management software [Електронний ресурс]. — Режим доступу до ресурсу: https://zapier.com/blog/best-event-management-software

4. Banker K. MongoDB in Action / Kyle Banker // Manning Publications; 1-st edition, December, 27, 2011. — P. 182-183.