МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Кафедра комп'ютеризованих систем управління

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри

_____Литвиненко О.Є.
"_____"_____ 2022 р.

ДИПЛОМНИЙ ПРОЕКТ
(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ
"БАКАЛАВР"

Тема: Онлайн-сервіс організації конференцій

_____

_____

Виконавець: _____ Волошко Д.В.

Керівник: _____ к.т.н., доцент Халімон Н. Ф.

Нормоконтролер: _____Тупота Є.В.

Київ 2022

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, комп'ютерної та програмної інженерії
Кафедра комп’ютеризованих систем управління
Спеціальність 123 «Комп'ютерна інженерія»
(шифр, найменування)

Освітньо-професійна програма «Системне програмування»
Форма навчання денна

ЗАТВЕРЖУЮ
Завідувач кафедри

Литвиненко О. Є.

« ____ »_____2022 р.

З А В Д А Н Н Я
на виконання дипломної роботи (проекту)

Волошко Дмитро Валерійович
(прізвище, ім’я, по батькові випускника в родовому відмінку)

1. Тема дипломної роботи (проекту): Онлайн-сервіс організації конференцій

затверджена наказом ректора від "04" лютого 2022 року № 135/ст.

2. Термін виконання роботи (проекту): з 17.05.2022 до 20.06.2022

3. Вихідні дані до роботи (проекту): інтегроване середовище розробки *Visual Studio Code*, мова *JavaScript*, фреймворк *Vue.js, система управління базами даних Firebase*.

4. Зміст пояснювальної записки:
1) Задачі створення онлайн-сервісу конференцій.
2) Проектування онлайн-сервісу організації конференцій.
3) Розробка онлайн-сервісу організації конференцій.

5. Перелік обов’язкового графічного (ілюстративного) матеріалу:

1) Архітектура односторінокового та мультисторінкового додатку.

2) Структура файлів проекту.

3) Головна сторінка сторінка онлайн-сервісу організації конференцій.

4) Вікно сторінки конференції.

5) Вікно сторінки для створення конференції.

6. Календарний план-графік

№ пор.	Завдання	Термін виконання	Відмітка про виконання
1	Ознайомитись з постановкою задачі дипломного проектування	18.05.22	
2	Вивчити спеціальну літературу і технічну документацію	14.05.22	
3	Проаналізувати системи управління базами даних та утиліти для адміністрування	16.05.22	
4	Написати розділ 1.	18.05.22	
5	Дослідити параметри системних баз даних, подання, оператори, функції, змінні мови *Transact-SQL*.	19.05.22	
6	Написати розділ 1.	21.05.22	
7	Провести проектування програмного засобу моніторингу параметрів відновлення бази даних кадрів підприємства	25.05.22	
8	Написати розділ 2.	27.05.22	
9	Провести розробку програмного засобу моніторингу параметрів відновлення бази даних кадрів підприємства	07.06.22	
10	Написати розділ 3.	12.06.22	
11	Оформити пояснювальну записку	13.06.22	
12	Підготувати графічний демонстраційний матеріал та доповідь	14.06.22	

7. Дата видачі завдання: “17” травня 2022 р.

Керівник дипломної роботи (проекту) _____ Халімон Н.Ф.
(підпис керівника) (П.І.Б.)

Завдання прийняв до виконання _____ Волошко Д.В.
(підпис випускника) (П.І.Б.)

РЕФЕРАТ

Пояснювальна записка до дипломного проекту «Онлайн-сервіс організації конференцій»: 61 с., 5 рис., 22 літературних джерел.

ВЕБ-ДОДАТОК, ОНЛАЙН-СЕРВІС КОНФЕРЕНЦІЙ, *HTML*, *CSS*, *JAVASCRIPT*.

Об’єкт проектування – програмне забезпечення для організації он-лайн заходів.

Предмет проектування – онлайн-сервіс організації конференцій.

Метод проектування – застосування засобів для розробки онлайн-сервісу організації конференцій.

Дипломний проект присвячено тематиці створення онлайн-сервісу організації конференцій.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

WEB – World Wide Web (всесвітня павутина, всесвітня мережа, інтернет)

HTTP – Hypertext Transport Protocol (протокол передачі гіпертексту)

URI – Uniform Resource Identifier

URL – Uniform Resource Locator

DNS – Domain Name System

HTML – Hypertext Markup Language (мова розмітки гіпертексту)

XHTML – Extensible Hypertext Markup Language

XML – Extensible Markup Language

CSS – Cascading Style Sheets

UI – User Interface

DOM – Document Object Model

ВСТУП

Дізнатися про інноваційні проекти, дослідження і розробки можна на різних наукових форумах, зокрема на наукових конференціях, які проводяться з метою узагальнення досвіду розробників, отримання інформації про нові дослідження у своїх зарубіжних колег.

Всесвітню мережу (*WEB*) утворюють мільйони веб-серверів мережі Інтернет, розташованих по всьому світу. Всесвітня павутина нерозривно пов'язана з поняттями гіпертексту і гіперпосилання. Більша частина інформації у всесвітній павутині являє собою саме гіпертекст.

Термін гіпертекст був введений Тедом Нельсоном в 1965 році для позначення «тексту, що розгалужується або виконується за запитом». Зазвичай гіпертекст − це набір текстів, що містять вузли переходу від одного тексту до іншого. Для перегляду інформації, отриманої від веб-сервера на клієнтському комп'ютері застосовується спеціальна програма − веб-браузер. Основна функція веб-браузера − відображення гіпертексту.

Онлайн-сервіс (веб-сервіс) – це сукупність програмно-технічних засобів, інтегрованих з метою ефективного опрацювання веб-ресурсів, які знаходяться у веб-просторі. Трактування поняття веб-технологій пов’язано з засобами створення веб-сторінок з підтримкою мультимедіа, які поєднують у собі різні види інформації: текст, графіку, звук, анімацію й відео. Таке трактування веб-технологій охоплює базові сервіси Інтернету і не втрачає свого змісту і сьогодні. Поняття веб-технологій пов’язано з засобами створення веб-сторінок з підтримкою мультимедіа, які поєднують у собі різні види інформації: текст, графіку, звук, анімацію й відео. Таке трактування веб-технологій охоплює базові сервіси Інтернету і не втрачає свого змісту і сьогодні. Проте в сучасних умовах, коли Інтернет маніпулює не тільки статичними даними у веб-просторі, а є потужним засобом комунікації, інтеграції, пошуку веб-ресурсів, надання різноманітних сервісів, поняття веб-технологій трактується ширше – як комплекс технічних, комунікаційних, програмних методів розв’язання завдань організації спільної діяльності користувачів із застосування мережі Інтернет.

Кожен сервіс має свої переваги та недоліки, тому що їх створює людина, а не комп'ютер. Існуючі комплексні програмні продукти для автоматизованого вирішення проблеми організації конференцій, наприклад, популярні сервіси під час пандемії *Covid-19* такі як *Zoom, MS Teams* надають змогу проводити конференції у онлайн-форматі, але зі спадом захворювань пандемією з'явилася потреба у проведенні конференцій, самітів, подій у оффлайн-форматі, тому що багато прикладних сфер потребують взаємодії з учасниками у реальному житті.

Невід'ємною частиною створення онлайн-сервісу являється чіткий опис задачі та проектування відповідно до вимог, без цього неможливо створити якісний продукт. Проаналізувавши розглянуті онлайн-сервіси у попередньому розділі, був спроектований онлайн-сервіс, який має мету максимально спрощувати і автоматизувати процес організації та проведення конференцій.

Сучасний *WEB* дедалі частіше використовує технології *Single Page Application* (*SPA*) для *front-End* розробки за допомогою якого був спроектований онлайн-сервіс. *SPA* – це збірна назва набору технологій, що дозволяють реалізувати *WEB*-додаток, який виконується *WEB*-браузером як одна *WEB*-сторінка. Прикладом технології *SPA* є реалізований сервіс *Gmail* від *Google*. З точки зору користувача, дана технологія забезпечує високу швидкість відгуку на дії в інтерфейсі. При цьому не потрібно повного або навіть часткового перезавантаження *WEB*-сторінки з сервера – всі візуальні елементи конструюються прямо в браузері за допомогою технології *JavaScript* маніпуляцій з *DOM*-структурою документа.

Відображення – це найважливіша і найбільш складна частина сучасних *SPA*. Також відображення оновлює отриманий *HTML* при зміні моделі і навпаки – повідомляє модель про дії користувача з представленням. При натисненні клавіші мишки чи введенні з клавіатури модель може виконати маніпуляції з даними, а після цього повідомити відображення про зміну даних для того, щоб відображення оновило або згенерувало новий *HTML*.

Робота класичного *WEB*-додатка (або *WEB*-сайту) повністю будується за допомогою кешування даних: на сервері, на проксі-сервері і на клієнті. Якщо дані

і стан додатку оновлюються дуже часто, перевага від використання кешування практично нівелюється.

Односторінкові додатки стають все більш популярнішими за мультисторінкові (класичні) додатки. Тому є актуальними дослідження структури, поведінки, швидкодії односторінкових веб-додатків для того, щоб підвищувати їх якість та надійність. Односторінкові веб-додатки слід використовувати тоді, коли необхідно розробити кросплатформовий додаток. Як показує практика, вони добре функціонують як на персональних ПК, так і мобільних пристроях.

Переважно бібліотеки та фреймворки *JavaScript* забезпечують прозору структуру програми та реалізуються на одному з двох найпопулярніших шаблонів дизайну розробки веб-додатків *MVC* (*Model-ViewController*) та *Model-View-ViewModel* (*MVVM*).

На даному етапі розвитку *WEB*, односторінкові додатки швидко витісняють класичні додатки і є великим внеском у розробку масштабних, швидких, динамічних веб-систем, що було враховано при проектуванні додатку організації конференцій. Однак технології реалізації *front-End* частини *web*-додатків потребують додаткового дослідження в контексті вибору кращих альтернатив для реалізації систем під конкретні вимоги користувачів чи замовників продукту.

Динаміка розвитку сайтів вимагає готовності швидко реагувати на зміни і впроваджувати їх з максимальною оперативністю. Це також стосується медійних порталів, зокрема онлайн-сервісів організації конференцій, тому тема дипломного проекту “Онлайн-сервіс організації конференцій” є актуальним завданням.

РОЗДІЛ 1
ЗАДАЧІ СТВОРЕННЯ ОНЛАЙН-СЕРВІСУ КОНФЕРЕНЦІЙ

1.1. Поняття онлайн-сервісу та *WEB*

Всесвітню мережу (*WEB*) утворюють мільйони веб-серверів мережі Інтернет, розташованих по всьому світу. Веб-сервер є програмою, що запускається на підключеному до мережі комп'ютері і використовує протокол *HTTP* для передачі даних [1]. Зазвичай така програма отримує по мережі *HTTP*-запит на певний ресурс, знаходить відповідний файл на локальному жорсткому диску і відправляє його по мережі комп'ютеру, що надіслав запитав. Веб-сервери здатні динамічно формувати ресурси у відповідь на *HTTP*-запит. Для ідентифікації ресурсів (часто файлів або їх частин) у Всесвітній павутині використовуються уніфіковані ідентифікатори ресурсів *URI*. Для визначення місцезнаходження ресурсів в мережі використовуються уніфіковані локатори ресурсів *URL*. Такі *URL*-локатори поєднують в собі технологію ідентифікації *URI* і систему доменних імен *DNS* − доменне ім'я (або безпосередньо *IP*-адрес в числовому записі) входить до складу *URL* для позначення комп'ютера (точніше − одного з його мережевих інтерфейсів), який виконує код потрібного веб-сервера [2].

Для перегляду інформації, отриманої від веб-сервера на клієнтському комп'ютері застосовується спеціальна програма − веб-браузер. Основна функція веб-браузера − відображення гіпертексту.

Всесвітня павутина нерозривно пов'язана з поняттями гіпертексту і гіперпосилання. Більша частина інформації у всесвітній павутині являє собою саме гіпертекст.

Термін гіпертекст був введений Тедом Нельсоном в 1965 році для позначення «тексту, що розгалужується або виконується за запитом». Зазвичай гіпертекст − це набір текстів, що містять вузли переходу від одного тексту до іншого. Це дозволяє обирати об'єкт або послідовність читання. Загальновідомим прикладом гіпертексту є веб-сторінки − документи на *HTML* (на гіпертекстовій мові розмітки), які розміщені у Всесвітній павутині.

Для полегшення створення, зберігання і відображення гіпертексту у Всесвітній павутині традиційно використовується мова *HTML*, мова розмітки гіпертексту. Програмування гіпертексту називається верстkою, майстра розмітки називають веб-майстром (верстальником). Після програмування *HTML*-розмітки гіпертексту, він зберігається у файлі; такий *HTML*-файл є найпоширенішим ресурсом Всесвітньої мережі. Після того, як *HTML*-файл стає доступним веб-серверу, його починають називати «веб-сторінкою». Набір веб-сторінок утворює веб-сайт [3]. До тексту веб-сторінок додаються гіперпосилання. Гіперпосилання допомагають користувачам Всесвітньої павутини легко переміщатися між ресурсами (файлами) незалежно від того, знаходяться ресурси на локальному комп'ютері або на віддаленому сервері. Гіперпосилання Всесвітньої павутини засновані на технології *URL*.

В цілому можна зробити висновок, що Всесвітня павутина побудована з використанням: протоколу *HTTP*, мови *HTML* і уніфікованих локаторів ресурсів *URL*. Останнім часом *HTML* поступається сучаснішим технологіям розмітки: *XHTML* і *XML* тому, що *XHTML* широко використовуються при розробці веб-додатків на базі *Android*, що допомагає забезпечувати швидкий розвиток. А також застосовується в сценаріях на стороні сервера за допомогою інших мов, вбудованих в нього, що дозволяє робити інтерфейс користувача більш гнучким. Для поліпшення візуального сприйняття Всесвітньої павутини розробники почали широко застосовувати спеціальну мову стилю сторінок *CSS*,

яка дозволяє задавати єдині стилі оформлення для багатьох веб-сторінок одразу.

Онлайн-сервіс (веб-сервіс) – це сукупність програмно-технічних засобів, інтегрованих з метою ефективного опрацювання веб-ресурсів, які знаходяться у веб-просторі [4]. Трактування поняття веб-технологій пов'язано з засобами створення веб-сторінок з підтримкою мультимедіа, які поєднують у собі різні види інформації: текст, графіку, звук, анімацію й відео. Таке трактування веб-технологій охоплює базові сервіси Інтернету і не втрачає свого змісту і сьогодні. Проте в сучасних умовах, коли Інтернет маніпулює не тільки статичними даними у веб-просторі, а є потужним засобом комунікації, інтеграції, пошуку веб-ресурсів, надання різноманітних сервісів, поняття веб-технологій трактується ширше – як комплекс технічних, комунікаційних, програмних методів розв'язання завдань організації спільної діяльності користувачів із застосування мережі Інтернет.

Незважаючи на всю різноманітність сайтів і послуг, які вони надають, онлайн-сервіси можна розділити на наступні основні групи, в залежності від їх тематики і призначення: інформаційно-пошукові, соціальні, сервіси покупок, банківські сервіси та дорожні сервіси.

Інформаційно-пошукові ресурси призначені для пошуку необхідної інформації. До них відносяться не тільки пошукові бази, а й онлайн-бібліотеки з книгами, музикою або фільмами, різні каталоги і довідники, а також онлайн-перекладачі.

До групи соціальних відносяться соціальні мережі, онлайн-комунікатори, а також електронні поштові служби. За допомогою таких сервісів можна не тільки спілкуватися з людьми, незалежно від їх дислокації, а й вести ділову переписку, пересилати файли і багато іншого.

Веб-сайти групи сервіси для покупок призначені для придбання речей або замовлення послуг. До цієї категорії відносяться не тільки онлайн-магазини, але і сервіси на замовлення і бронювання квитків (починаючи від театральних і закінчуючи залізничними), готелів і багато чого іншого.

За допомогою банківських сервісів можна оплатити покупку або послугу, наприклад, замовлення в інтернет-магазині, або навіть заплатити за комунальні послуги, без необхідності особистого відвідування банку.

Дорожні сервіси дають можливість швидко прокласти маршрут незалежно від способу пересування, оцінити транспортну завантаженість на дорогах, отримати інформацію про тривалість поїздки тощо.

1.2. Задачі онлайн-сервісу для створення конференцій

Дізнатися про інноваційні проекти, дослідження і розробки можна на різних наукових форумах, зокрема на наукових конференціях, які проводяться з метою узагальнення досвіду розробників, отримання інформації про нові дослідження у своїх зарубіжних колег.

При організації проведення будь-якої конференції можна виділити наступні етапи:

1) формування інформаційного повідомлення про проведення наукового заходу, яке містить повну інформацію про конференцію (дата і місце проведення, правила оформлення матеріалів, відомості про організаторів тощо);

2) створення сайту або сторінки на сайті провідної організації з інформацією про конференцію, що значно спрощує поширення відомостей про неї;

3) розсилання інформації про конференцію потенційним учасникам за наявною базою адрес електронної пошти;

4) отримання, оброблення та приймання заявок до участі, матеріалів (тез або статей) та квитанцій про оплату;

5) інформування учасників про план проведення конференції, об'єкти проведення конференції, додаткові заходи та ін.;

6) реєстрація учасників конференції, які прийняли рішення особисто виступити з доповіддю (у разі очної форми проведення);

7) формування списку доповідачів для пленарних і секційних засідань;

8) безпосереднє проведення конференції.

Існуючі комплексні програмні продукти для автоматизованого вирішення проблеми організації конференцій, наприклад, популярні сервіси під час пандемії *Covid-19* такі як *Zoom, MS Teams* надають змогу проводити конференції у онлайн-форматі, але зі спадом захворювань пандемією з'явилася потреба у проведенні конференцій, самітів, подій у оффлайн-форматі, тому що багато прикладних сфер потребують взаємодії з учасниками у реальному житті.

1.3. Огляд існуючих онлайн-сервісів конференцій

Розглянемо декілька сервісів для того, щоб звернути увагу на недоліки сервісів та уникнути цих недоліків у своєму онлайн-сервісі.

Веб-сервіс *FWdays* – це онлайн-сервіс, який дозволяє великій кількості розробників поділитися своїм досвідом у сфері розробки програмного забезпечення, мобільної розробки, веб-розробки, аналітики, менеджменту, криптовалютних операцій, хмарних обчислень та інших прикладних сферах. Переважна кількість конференцій та форумів присвячені веб-розробці та веб-технологіям, які розвиваються над швидкими темпами у сучасному світі.

Онлайн-платформа *DOU* – це платформа, яка проводить велику кількість IT-конференцій, а також включає у собі сервіси: новини, пошук роботи, рейтинги компаній, ринок праці у сфері інформаційних технологій в Україні.

Онлайн-сервіс *Bizzabo* – це доволі відомий та масштабний зарубіжний веб-сервіс організації оф-лайн та онлайн конференцій, самітів, форумів, семінарів, шоу з кількістю від маленьких груп людей (від 5 до 30 людей) до гігантських форумів з тисячами учасників.

Кожен сервіс має свої переваги та недоліки, тому що їх створює людина, а не комп'ютер, розглянемо головні з них у вище перелічених онлайн-сервісах.

Серед переваг *FWdays* можу виділити: інтуїтивно зрозумілий та мінімалістичний дизайн (*UI user-friendly*), адаптивність під різні пристрої, доступність (*accessibility*), швидкість відклику веб-серверів та завантаження сторінок (*performance*), *SEO*-оптимізація, широка популярність серед української *IT*-спільноти та корисні вузько- спеціалізовані доповіді. Серед недоліків можна виділити використання застарілої бібліотеки *jQuery* та відсутність функції розміщення власних подій, а також зависока ціна подій та конференцій та лише проведення технічних подій.

Наступна платформа *DOU* відзначилась такими перевагами: швидкість відклику веб-серверів та завантаження сторінок (*performance*), сервіс має чудово серверну архітектуру через використання мови програмування *Python*, наявність функції фільтрації та сортування конференцій за різними параметрами, функції для створення конференції, велика кількість безкоштовних подій у різних містах України оф-лайн або он-лайн, наявність інших корисних сервісів для розробника. Серед недоліків *DOU* є: відсутність адаптивного дизайну, застарілий дизайн табличного типу, велика нагромадження елементів, використання застарілої бібліотеки *jQuery* та проведення лише технічних подій.

Онлайн-сервіс *Bizzabo* має велику перевагу у вигляді мобільного додатку, що значно збільшує популярність та обсяг користувачів на сервісі, сервіс написаний на новітній бібліотеці *React*, яка має широку підтримку та високу ефективність, функції надання аналітики та аналізу подій організатору, веб-сервіс має адаптивний та інтерактивний дизайн. Також перевагами *Bizzabo* вважаю різноплановість подій та можливість створення події на будь-яку тему. Недоліками *Bizzabo* є: недосконала система оплати та повернення коштів та складність реєстрації конференції або події.

1.4. Висновки до розділу

Веб-сервер є програмою, що запускається на підключеному до мережі комп'ютері і використовує протокол *HTTP* для передачі даних. Зазвичай така програма отримує по мережі *HTTP*-запит на певний ресурс, знаходить відповідний файл на локальному жорсткому диску і відправляє його по мережі комп'ютеру, що надіслав запитав. Загальновідомим прикладом гіпертексту є веб-сторінки − документи на *HTML* (на гіпертекстовій мові розмітки), які розміщені у Всесвітній павутині.

В цілому можна зробити висновок, що Всесвітня павутина побудована з використанням: протоколу *HTTP*, мови *HTML* і уніфікованих локаторів ресурсів *URL*.

Онлайн-сервіс (веб-сервіс) – це сукупність програмно-технічних засобів, інтегрованих з метою ефективного опрацювання веб-ресурсів, які знаходяться у веб-просторі.

Існуючі комплексні програмні продукти для автоматизованого вирішення проблеми організації конференцій, наприклад, популярні сервіси під час пандемії *Covid-19* такі як *Zoom, MS Teams* надають змогу проводити конференції у онлайн-форматі, але зі спадом захворювань пандемією з’явилася потреба у проведенні конференцій, самітів, подій у оф-лайн форматі, тому що багато прикладних сфер потребують взаємодії з учасниками у реальному житті. Розглянуті сервіси мають недоліки, які було розглянуто та усунуто у розробленому онлайн-сервісі для кращого оцінки користувачів.

Розглянуто декілька існуючих сервісів: веб-сервіс *FWdays*, онлайн-платформа *DOU*, онлайн-сервіс *Bizzabo*. Розглянуто їх недоліки та переваги.

РОЗДІЛ 2
ПРОЕКТУВАННЯ ОНЛАЙН-СЕРВІСУ ОРГАНІЗАЦІЇ КОНФЕРЕНЦІЙ

2.1. Засоби створення онлайн-сервісу організації конференцій в клієнт-серверній архітектурі

Головна мета була − проектування онлайн-сервісу, який включає в себе необхідні функції для проведення всіх етапів організації та проведення наукових конференцій з мінімальними витратами часу організаційного комітету конференції і використанням людського ресурсу. Також важливий аспект полягає у зручності використання даного сервісу організаторами і користувачами на всіх етапах процесу організації та проведення конференції.

Клієнт-сервер (англ. *Client-server*) – це архітектура комп'ютерної мережі, в якій багато клієнтів (*remote computers*) запитують та отримують інформацію з централізованого сервера (*host-computer*). Клієнтські комп'ютери надають інтерфейс, щоб дозволяє користувачеві комп'ютера отримати відповідь від сервера та відображати результати, які повертає сервер. (рис. 1.1) [4]

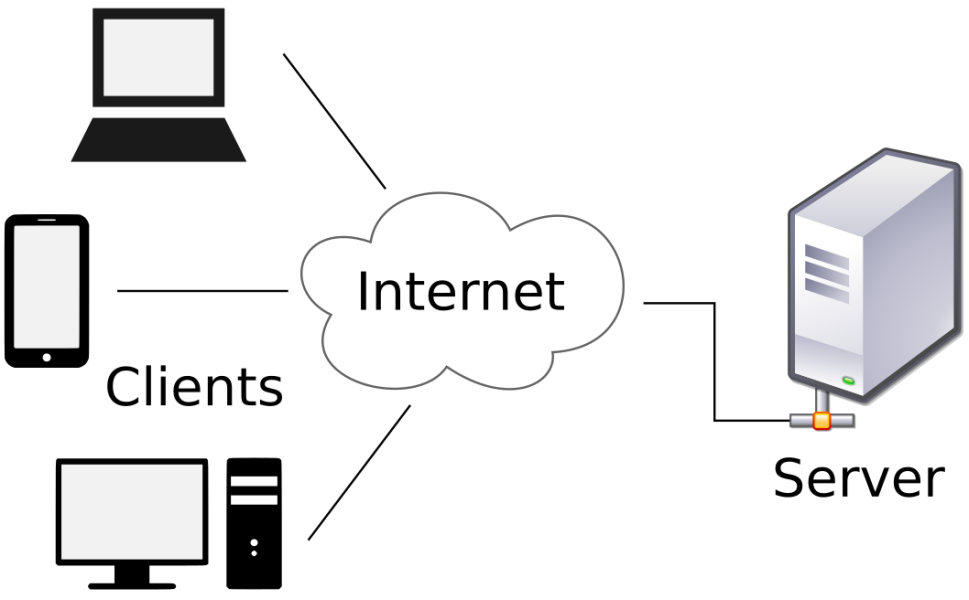

Рис. 2.1. Клієнт-сервер архітектура

Сервери чекають, коли запити прийдуть від клієнтів, а потім відповідають на них. В ідеалі сервер забезпечує стандартизований прозорий інтерфейс для клієнтів, щоб клієнтська частина сервісу не мала доступу до специфіки системи (тобто до апаратного забезпечення та програмного забезпечення), що надає послугу. Клієнти часто знаходяться на робочих станціях або на персональних комп'ютерах, тоді як сервери розташовані в іншому місці в мережі, як правило, на більш потужних машинах. Ця обчислювальна модель особливо ефективна, коли клієнти та сервер мають чіткі завдання, які вони регулярно виконують.

Наприклад, в обробці даних у лікарні, клієнтський комп'ютер може запускати прикладну програму для введення інформації про пацієнтів, поки серверний комп'ютер виконує іншу інструкцію, яка керує базою даних, в якій інформація постійно зберігається. Багато клієнтів можуть одночасно отримати доступ до інформації сервера, і, в той же час, клієнтський комп'ютер може виконувати інші завдання, такі як надсилання листів електронної пошти.

Оскільки як клієнтські, так і серверні комп'ютери, вважаються незалежними пристроями, модель клієнт-сервер абсолютно відрізняється від старої моделі *Mainframe*, в якій централізований комп'ютер (*Mainframe*) виконував усі завдання, пов'язані з простими (“*dumb*”) терміналами, які просто спілкувалися з центральним комп'ютером.

Сучасні технології розробки програмного забезпечення характеризуються великою кількістю інструментальних засобів і методів проектування, які, в залежності від сфери застосування, формують окремі напрями інженерії програмного забезпечення. Зокрема, на сьогодні сформовано напрям – *web*-інженерії (*web-engineering*). Даний напрям характеризується високою технологічністю та розвинутими інструментами підтримки і супроводу.

При проектуванні додатку для організації конференцій було використано наступну архітектуру. Оскільки, більшість програмних продуктів під *WEB* використовують на концептуальному рівні клієнт-серверну архітектуру, то це призвело до розділення логіки роботи *web*-орієнтованих систем на дві частини – *front-End* та *back-End*. В програмній інженерії терміни «*front-End*» та «*back-*

End» розрізняють за принципом розділення відповідальності між рівнем представлення та рівнем доступу до даних відповідно. *Front-End* відповідає за взаємодію користувача з інтерфейсом *web*-системи, *back-End* – за обробку подій на стороні сервера.

Сучасний *WEB* дедалі частіше використовує технології *Single Page Application* (*SPA*) для *front-End* розробки за допомогою якого був спроектований онлайн-сервіс. *SPA* – це збірна назва набору технологій, що дозволяють реалізувати *WEB*-додаток, який виконується *WEB*-браузером як одна *WEB*-сторінка [5]. Прикладом технології *SPA* є реалізований сервіс *Gmail* від *Google*. З точки зору користувача, дана технологія забезпечує високу швидкість відгуку на дії в інтерфейсі. При цьому не потрібно повного або навіть часткового перезавантаження *WEB*-сторінки з сервера – всі візуальні елементи конструюються прямо в браузері за допомогою технології *JavaScript* маніпуляцій з *DOM*-структурою документа.

Таким чином, *WEB*-додатки стають дуже схожі на звичайні програми для робочих станцій, що завантажують інформацію з мережі Інтернет. Середовищем виконання для них є не операційна система, а браузер, який в результаті змушений нести на собі все навантаження, пов'язане з виконанням стороннього коду.

Центральне місце в *SPA*-архітектурі при проектуванні додатку організації конференцій займає відображення (*View*) – те, що бачить і з чим взаємодіє користувач. Результатом роботи відображення є звичайний *HTML*, що відображається браузером. На відміну від «перехідних» «*WEB* 2.0»-додатків, що активно працюють з *DOM*-структурою документа, наприклад, за допомогою *jQuery* або *underscore, SPA*-додаток використовує *DOM* тільки для запису змін, але не для читання, тобто не для зберігання даних [6]. Для зберігання даних тепер використовується ще один компонент *SPA*-архітектури – модель (*Model*), що було використано при проектуванні додатку організації конференцій.

SPA-модель додатку організації конференцій представляє собою сукупність даних, функцій для маніпуляції з даними і подіями. Всі дані моделі

повністю зберігаються в пам'яті. Для того, щоб дані, що знаходяться в моделі, і дані, які відображаються представленням, зберігали цілісність, відображення підписується на події моделі, відстежуючи таким чином зміни даних в моделі. У свою чергу, модель також реагує на повідомлення відображення і забезпечує нерозривний зв'язок *WEB*-додатка з сервером, виконуючи запити для отримання або відправки даних (зокрема із застосуванням методології *REST*).

Відображення – це найважливіша і найбільш складна частина сучасних *SPA*. Також відображення оновлює отриманий *HTML* при зміні моделі і навпаки – повідомляє модель про дії користувача з представленням. При натисненні клавіші мишки чи введенні з клавіатури модель може виконати маніпуляції з даними, а після цього повідомити відображення про зміну даних для того, щоб відображення оновило або згенерувало новий *HTML* [7].

Робота класичного *WEB*-додатка (або *WEB*-сайту) повністю будується за допомогою кешування даних: на сервері, на проксі-сервері і на клієнті. Якщо дані і стан додатку оновлюються дуже часто, перевага від використання кешування практично нівелюється.

Односторінкові додатки стають все більш популярнішими за мультисторінкові (класичні) додатки. Тому є актуальними дослідження структури, поведінки, швидкодії односторінкових веб-додатків для того, щоб підвищувати їх якість та надійність. Односторінкові веб-додатки слід використовувати тоді, коли необхідно розробити кросплатформовий додаток. Як показує практика, вони добре функціонують як на персональних ПК, так і мобільних пристроях. На рисунку 1.2 зображено структуру односторінкового та мультисорінкового додатка.

Рис. 2.2. Архітектура односторінокового та мультисторінкового додатку

Порівнявши вище наведені структури, можна зробити висновок, що справді односторінкові додатки є високопродуктивними завдяки тому, що не навантажують серверну частину частими запитами як це роблять класичні додатки. Односторінкові додатки завантажують усі дані при зверненні до сторінки. При виконанні певних дій користувачем *SPA* завантажує дані по мірі необхідності без повного перезавантаження. Це є однією із основних переваг при виборі структури веб-додатку.

При проектуванні додатку організації конференцій був використаний на модульний підхід [8]. Модулі в середині веб-додатку повинні бути мало зв'язними для того, щоб можна було в будь-який момент масштабувати програму. Найкращим вирішенням даної проблеми є використання саме односторінкових додатків через те, що вони будуються на основі модулів, які незалежні один від одного і можуть бути повторно використані. Клієнтська та серверна частини в *SPA* є повністю незалежними, що дає змогу ізолювати їх один від одного. Зміни у серверній частині після заміни ніяким чином не вплине на клієнтську частину.

На даному етапі розвитку *WEB*, односторінкові додатки швидко витісняють класичні додатки і є великим внеском у розробку масштабних, швидких, динамічних веб-систем, що було враховано при проектуванні додатку організації конференцій. Однак технології реалізації *front-End* частини *web*-додатків потребують додаткового дослідження в контексті вибору кращих альтернатив для реалізації систем під конкретні вимоги користувачів чи замовників продукту.

При проектуванні додатку організації конференцій було розглянуто наступне. *JavaScript* – це динамічна мова програмування, яка використовується для веб-розробки, у веб-додатках, для розробки ігор та інших завдань. Мова дозволяє впроваджувати динамічні функції на веб-сторінках, які неможливо виконати лише за допомогою *HTML* та *CSS*.

Багато браузерів використовують *JavaScript* як мову сценаріїв для того, щоб додати динамічних елементів на сторінках веб-сайтів у Всесвітній мережі. Щоразу, коли онлайн-сервіс має випадаюче меню, випливаюче вікно, слайдер та інші інтерактивні елементи на веб-сайті це все – *JavaScript*.

За допомогою *JavaScript* можуть бути розроблені як веб-сайти, так і серверні модулі. Клієнтські фреймворки та бібліотеки більше підходять для першого завдання, а для роботи на стороні сервера використовується серверна платформа *NodeJS*.

Переважно бібліотеки та фреймворки *JavaScript* забезпечують прозору структуру програми та реалізуються на одному з двох найпопулярніших шаблонів дизайну розробки веб-додатків *MVC* (*Model-ViewController*) та *Model-View-ViewModel* (*MVVM*).

Основні особливості цієї мови – динамічність, гнучкість роботи з функціями та універсальність. Вона підтримується всіма сучасними браузерами, легко інтегрується з верстkою (*HTML*) та дає змогу налаштувати комунікацію з сервером. Серед інших переваг:

− *JavaScript* є дуже швидкою мовою, оскільки він часто запускається відразу в браузері клієнта. Поки він не вимагає зовнішніх ресурсів, *JavaScript*

не сповільнюється через виклики до серверного сервера. Крім того, всі основні браузери підтримують компіляцію *JIT* (вчасно) для *JavaScript*, що означає, що немає необхідності компілювати код перед його запуском.

− Простий синтаксис *JavaScript* був подібний до *Java* і зрозуміліший порівняно з іншими популярними мовами, такими як *C++*.

− Тип даних визначається, коли змінній або константі присвоюється значення [9];

− В *JavaScript* функції можна як виконувати, так і повертати, передавати їх як параметри іншим функціями і привласнювати як значення змінних [10].

− Методологія об'єктно-орієнтованого програмування доступна у *JavaScript* та дає змогу представити програму у вигляді сукупності об'єктів [11];

− Дозволяє частково перенести бізнес-логіку із сервера на сторону користувача, тобто виконувати код в браузері, що своєю чергою зменшує навантаження на сервери.

− *JavaScript* має розвинену інфраструктуру та активну спільноту. Так, веб-розробники можуть працювати з великою кількістю бібліотек і фреймворків як *React*, *Angular* і *Vue*, декількома пакувальниками, як *Webpack*, *Gulp*, та допоміжними бібліотеками як *Lodash*, *axios*, та іншими [12].

Також мова має декілька свої недоліків, які веб-розробники намагаються запобігати:

− Захист на стороні клієнта тому, що код *JavaScript* виконується на стороні клієнта, помилки іноді можуть бути використані для зловмисних цілей. Через це деякі програмісти вирішують повністю вимкнути *JavaScript*.

− Підтримка браузера тому, що сценарії на стороні сервера завжди дають однакові результати, різні браузери іноді інтерпретують код *JavaScript* по-різному. У наш час відмінності мінімальні, і не стосуються нових браузерів, якими користується переважна більшість.

При проектуванні додатку організації конференцій було розглянуто головні бібліотеки та фреймворки *JavaScript*, на яких побудована переважна більшість онлайн-сервісів у Всесвітній павутині:

React – це ефективна і гнучка декларативна бібліотека *JavaScript* для збірки *UI* від команди *Facebook*. Вона дозволяє створювати інтерактивний інтерфейс користувача. У *React* використовується компонентний підхід. Бібліотека не має контролерів, моделей, шаблонів – лише є компонент. Компоненти можна повторно використовувати, успадковувати один від одного, об'єднувати. Компонент – це свого роду будівельна одиниця, з якої збирається інтерфейс. Перевагою є можливість порівняння *React*'ом віртуального *DOM*'а з реальним браузерним деревом вузлів *DOM* і виконання мінімальних змін для їх синхронізації. Віртуальний *DOM* вирішує проблему з обробкою подій в різних браузерах, за рахунок цього *React* надає сумісну модель подій в будь-якому браузері. *React* дозволяє використовувати будь-який інструмент при розробці, він добре поєднується з іншими фреймворками. Дизайн *React* поєднується з асинхронними серверними архітектурами для адаптації до майбутніх технологій [13].

Таким чином, *React* варто використовувати, якщо необхідно створити якісний і в найкоротші терміни швидкий, легкий, зручний односторінковий додаток.

Angular – це кросплатформовий (*cross-platform*) фреймворк розроблений *Google*, який дотримується дизайн-шаблону конструкції веб-додатків *MVC* і запроваджує відокремлений зв'язок між представленням веб-сайту у браузері, даними та логікою компонентів програми.

Angular переносить частину клієнтського коду на сторону сервера. Тому навантаження на веб-додаток значно зменшується. Завдяки *TypeScript* (строго типізований аналог *JavaScript*), код проекту стає чистим, зручним для розуміння розробників і містить менше помилок.

Можна відзначити наступні переваги кросплатформового фреймворку *Angular*, що було враховано при проектуванні додатку організації конференцій:

- відмінна документація;
- підтримка *Google*;
- величезний набір інструментів розробки (*Material UI, CLI* тощо) [14].

Однією з слабких сторін є високий поріг для введення проектів для розробників, оскільки, наприклад, потрібно знати *TypeScript*. Тому це ускладнює розвиток проектів, особливо якщо проект передається від однієї команди до іншої.

Друга проблема *Angular* – це дуже часте оновлення версій. У червні 2019 року вже було опубліковано восьму версію фреймворку. Цей факт також говорить про те, що *Angular* проекти складніше підтримувати.

Vue.js – це прогресивний фреймворк для створення інтерфейсів користувачів. *Vue.js* був створений для поступової реалізації у проект, на відміну від *Angular* або *React*. Це означає, що цей фреймворк можна реалізовувати на етапах створення *HTML*-сторінок, що значно спрощує розробку на *Vue.js*.

Vue.js широко використовується серед китайських компаній, наприклад, *Alibaba, Baidu, Xiaomi* та інші. Нещодавно система управління репозиторіями *Gitlab* також перейшла на *Vue.js*. *Vue.js* в першу чергу вирішує завдання рівня представленням веб-сайту у браузері (*view*), що спрощує інтеграцію з іншими бібліотеками та існуючими проектами.

Складова частина *Vue.js* побудована на найкращих сторонах *Angular* та *React*: швидкість, легкість, можливість підтримки таких технологій, як *TypeScript* та *JSX*. Але в той же час *Vue.js* залишається вірним стандартам написання коду на *HTML* та *CSS*, що сприяє легшому процесу розробки та підтримки проекту [15].

З слабких місць можна виділити ще не дуже велику спільноту, тому що фреймворки поки не підтримується великими корпораціями. Але популярність фреймворку щороку зростає.

Онлайн-сервіс організації конференцій спроектовано на технологіях клієнтської частини, але веб-сервіс повинен зберігати дані, тому було прийняте рішення використати систему керування базою даних *Firebase*.

На конференції *Google I / O* 2015 була представлена хмарна база даних на основі *NoSQL* з назвою *Firebase*. *Firebase* – це платформа для мобільних і веб-додатків. *Firebase* характерна наявністю:

1) Бази даних реального часу;
2) Засобів автентифікації.

Firebase Realtime Database – це база даних типу *NoSQL*, яка використовує веб-сокети, що дозволяє клієнту отримувати інформацію в реальному часі – без необхідності відправляти запити *GET* на сервер. Веб-сокети – це технологія, яка дозволяє створювати інтерактивне з'єднання між клієнтом (браузером) та сервером для обміну повідомленнями у режимі реального часу. Веб-сокети, на відміну від *HTTP*, дозволяють працювати з двонаправленим потоком даних, що робить цю технологію унікальною.

Переваги *Firebase Realtime Database*: синхронізація в реальному часі для даних *JSON*, створення без серверних додатків, оптимізація для автономного використання, безпека користувацьких даних,

База даних *Firebase Realtime* є хмарною базою даних *NoSQL*, яка дозволяє зберігати і синхронізувати дані між користувачами в режимі реального часу. *Firebase Realtime Database* поставляється з мобільними і веб-*SDK*, тому можна створювати додатки без необхідності серверів. Коли користувачі переходять в автономний режим, *SDK* бази даних *Realtime* використовує локальний кеш на пристрої для обслуговування і збереження змін. Коли пристрій підключається до мережі, локальні дані автоматично синхронізуються. Сервіс керування базою даних інтегрується з *Firebase Authentication* для забезпечення простої та інтуїтивної автентифікації для розробників.

Firebase Authentication дозволяє виконувати вхід в програму за допомогою системи, з якої користувачі добре знайомі, тобто *Facebook* або *Google*. Потім додаток може зберігати дані і персональні налаштування

користувача в захищеному хмарному сховище і забезпечувати доступ до них на всіх його пристроях. *Firebase* забезпечує служби серверної частини, прості пакети розробника і готові бібліотеки інтерфейсу для автентифікації користувачів різних додатків на будь-яких платформах. Автентифікацію можна виконувати за допомогою паролів і інтегрованих систем ідентифікації – *Google, Facebook, Twitter* та ін. Це значно спрощує процедуру входу в додаток і забезпечує надійний захист.

Firebase Authentication допомагає економити час при розробці методів для автентифікації: замість цього можна просто зберігати інформацію про користувача після автентифікації користувачем за допомогою *Firebase*. Також збережеться чимало часу, так як можна уникнути розробки методів на стороні сервера для різних видів перевірки токенів в разі, також є можливість додати соціальні логіни, такі як *Facebook* і *Google*. Все, що буде ефективно працювати з *Firebase*.

Переваги *Firebase Authentication* були враховані при проектуванні додатку організації конференцій та полягають в наступному: тісна інтеграція з іншими функціями *Firebase,* використання галузевих стандартів, таких *як OAuth* 2.0 і *OpenID Connect,* які спрощують інтеграцію з серверним кодом, два підходи використання, безпека автентифікації, безпечний доступ до сервісів *Google*

Забезпечується два варіанти для розробки: *FirebaseUI* – повністю універсальне рішення для виконання автентифікації або пакет *Firebase Authentication SDK*, який дозволяє вручну інтегрувати один або кілька методів входу в додаток.

Безпека автентифікації і зручність реалізуються за рахунок можливості виконувати вхід через обліковий запис *Google* та інші інтегровані системи ідентифікації – *Facebook, Twitter* і т.д. Також для входу можна використовувати пароль.

При проектуванні додатку організації конференцій були вибрані дані сервіси *Firebase* через їхню пристосованість до онлайн-сервісів. Це забезпечить

легкість роботи з ними. Дані сервіси мають хорошу захищеність. Завдяки захищеності цих сервісів необхідність в реалізації методів захисту в даному веб-додатку стає непотрібною. Даний сервіс бере на себе відповідальність за збереження даних. Завдяки широкому набору інструментів в даному сервісі, необхідність реалізації додаткових компонентів стає неактуальною [16].

2.2. Проектування інтерфейсу та функцій модулів онлайн-сервісу

Невід'ємною частиною створення онлайн-сервісу являється чіткий опис задачі та проектування відповідно до вимог, без цього неможливо створити якісний продукт. Проаналізувавши розглянуті онлайн-сервіси у попередньому розділі, був спроектований онлайн-сервіс, який має мету максимально спрощувати і автоматизувати процес організації та проведення конференцій. Кожен з модулів містить набір функцій для вирішення відповідних завдань. Умовно всі завдання сервісу можна розділити на шість модулів: авторизація, автентифікація, головна сторінка, сторінка конференції, організація конференції, профіль користувача.

Модуль авторизації – виконує функцію реєстрації на сервісі та повинен відображати такі поля реєстрації:

- текстове поле *email*;
- текстове поле ім’я;
- текстове поле пароль;
- текстове поле повтор паролю;
- кнопка типу “прапорець” для згоди з умовами та положеннями сервісу, кнопку типу “*submit*” для формування запиту та відправки даних користувача на сервер та подальшого запису в системі управління бази даних *Firebase*;
- текстове посилання на модуль автентифікації.

Якщо користувач намагається створити, видалити, відвідати певну конференцію або змінити зміст конференції у ролі гість (“*guest*”), він повинен бути переправлений на даний модуль.

Модуль автентифікації – виконує функцію входу користувача у систему. Модуль повинен відображати такі поля:

− текстове поле *email*;

− текстове поле пароль;

− кнопка типу “*submit*” для формування запиту та відправки даних користувача на сервер для подальшої перевірки введених даних з даними у базі даних, що дозволить вхід у сервіс або сповістить про невдалу спробу входу;

− текстове посилання на модуль авторизації.

Модуль головної сторінки – виконує функцію відображення списку конференцій та взаємодії з ними. Модуль складається з таких компонентів:

− шапка сторінки (*header*), яка відображає логотип-посилання на головну сторінку та текстові-посилання на модуль автентифікації, авторизації та створення конференції у ролі гість (“*guest*”). Авторизований користувач має дещо інші посилання на даний модуль з різними параметрами фільтрації відображення списку конференцій, на модуль створення конференції, на модуль профілю користувача та оброблювач події виходу з облікового запису користувача;

− компонент-фільтр списку конференцій за часовими параметрами: усі, минулі та майбутні;

− компонент текстове поле пошуку за назвою, описом та ім’ям організатора конференції;

− компонент перемикач відображення подій у вигляді списку або календаря;

− «підвал» сторінки (*footer*) з повідомленням про авторське право.

Модуль сторінки конференції – виконує функцію відображення розгорнутої інформації про певну конференцію та вміщує у собі такі компоненти:

– фото-банер з назвою конференції;

– дві текстові вкладки, які відображають опис конференції та докладний часовий графік програми події;

– інформативний компонент про організатора, місцезнаходження та дату конференції;

– кнопки “редагувати” та “видалити” у ролі організатора та кнопки “брати участь” та “скасувати участь” у ролі авторизованого користувача.

Модуль організації конференції – виконує функцію створення конференції та відображає такі поля:

– текстове поле назви конференції;

– поле календаря для встановлення дати проведення конференції;

– текстове поле місцезнаходження конференції;

– текстове поле опису події;

– поле файлового типу для завантаження фото-банера конференції;

– розширена форма програми з випадаючим списком різних типів подій на конференції, часовими полями для встановлення початку та кінця, заголовком й описом для кожної події конференції;

– кнопки “скасувати” та “створити” для відміни створення конференції та повернення до модуля головної сторінки або створення запиту на сервер та подальшого запису конференції у базі даних.

Модуль доступний лише у ролі авторизованого користувача, при спробі перейти на даний модуль у ролі гість (“*guest*”), користувач буде переправлений на модуль автентифікації.

Модуль профіль користувача – виконує функцію відображення розширеної інформацію про користувача та надає змогу редагування даних профіля. Модуль повинен відображати такі поля:

– текстове поле ім’я користувача;

– текстове поле *email* користувача;

– поле календаря для встановлення віку користувача;

– текстове поле місцезнаходження користувача;

− кнопка “редагувати” для формування запиту на сервер та редагування даних у базі даних певного користувача.

Чітко спроектований та описаний онлайн-сервіс є важливим кроком перед конструюванням та кодуванням веб-сервісу, далі розглянемо розробку, тобто яким чином створено онлайн-сервіс організації конференцій.

2.3. Висновки до розділу

Головна мета була − проектування онлайн-сервісу, який включає в себе необхідні функції для проведення всіх етапів організації та проведення наукових конференцій з мінімальними витратами часу організаційного комітету конференції і використанням людського ресурсу. Також важливий аспект полягає у зручності використання даного сервісу організаторами і користувачами на всіх етапах процесу організації та проведення конференції.

Клієнт-сервер (англ. *Client-server*) – це архітектура комп'ютерної мережі, в якій багато клієнтів (*remote computers*) запитують та отримують інформацію з централізованого сервера (*host-computer*). Клієнтські комп'ютери надають інтерфейс, щоб дозволяє користувачеві комп'ютера отримати відповідь від сервера та відображати результати, які повертає сервер.

Firebase Realtime Database – це база даних типу *NoSQL*, яка використовує веб-сокети, що дозволяє клієнту отримувати інформацію в реальному часі – без необхідності відправляти запити *GET* на сервер. Веб-сокети – це технологія, яка дозволяє створювати інтерактивне з'єднання між клієнтом (браузером) та сервером для обміну повідомленнями у режимі реального часу. Веб-сокети, на відміну від *HTTP*, дозволяють працювати з двонаправленим потоком даних, що робить цю технологію унікальною.

Firebase Authentication дозволяє виконувати вхід в програму за допомогою системи, з якої користувачі добре знайомі, тобто *Facebook* або *Google*. Потім додаток може зберігати дані і персональні налаштування

користувача в захищеному хмарному сховище і забезпечувати доступ до них на всіх його пристроях. *Firebase* забезпечує служби серверної частини, прості пакети розробника і готові бібліотеки інтерфейсу для автентифікації користувачів різних додатків на будь-яких платформах. Автентифікацію можна виконувати за допомогою паролів і інтегрованих систем ідентифікації – *Google, Facebook, Twitter* та ін. Це значно спрощує процедуру входу в додаток і забезпечує надійний захист.

При проектуванні додатку для організації конференцій було використано наступну архітектуру. Оскільки, більшість програмних продуктів під *WEB* використовують на концептуальному рівні клієнт-серверну архітектуру, то це призвело до розділення логіки роботи *web*-орієнтованих систем на дві частини – *front-End* та *back-End*. В програмній інженерії терміни «*front-End*» та «*back-End*» розрізняють за принципом розділення відповідальності між рівнем представлення та рівнем доступу до даних відповідно.

Проаналізувавши розглянуті онлайн-сервіси у попередньому розділі, був спроектований онлайн-сервіс, який має мету максимально спрощувати і автоматизувати процес організації та проведення конференцій. Умовно всі завдання сервісу можна розділити на шість модулів: авторизація, автентифікація, головна сторінка, сторінка конференції, організація конференції, профіль користувача.

Чітко спроектований та описаний онлайн-сервіс є важливим кроком перед конструюванням та кодуванням веб-сервісу, далі розглянемо розробку, тобто яким чином створено онлайн-сервіс організації конференцій.

РОЗДІЛ 3
РОЗРОБКА ОНЛАЙН-СЕРВІСУ ОРГАНІЗАЦІЇ КОНФЕРЕНЦІЙ

3.1. Створення проекту та структура файлів

Розробка онлайн-сервісу складається з наступних етапів: вибору та налаштування середовища розробки, створення початкового *Vue.js* середовища та завантаження основних програм для розробки.

Перший етап створення проекту завжди починається з вибору інтегрованого середовища, де відбувається конструювання та кодування проекту. Обрано було *Microsoft Visual Studio Code*, тому що цей редактор має такі переваги: вдосконалений для створення сучасних веб-сервісів та хмарних додатків, легка конфігурація у файлі *JSON*, гнучкість та велика кількість корисних розширень для написання більш надійного коду, доступний на головних операційних системах таких як *Windows, macOS* та *Linux*, має вбудовану підтримку *JavaScript, TypeScript, Node.js* та головна перевага *Microsoft Visual Studio Code* є те, що редактор безкоштовний.

Після встановлення редактору коду потрібно встановити розширення, які спростять велику кількість механічних дій, налагодження, збільшать стисліть та ергономічність коду.

Vetur – головний плагін для *Vue.js* розробника, якій підкреслює синтаксис *Vue.js* програм, що значно підвищує швидкість пошуку конкретної частини коду, шаблони *Vue.js* компонентів та багато іншого корисного для *Vue.js* розробника.

ESLint – це лінтер, один із найпопулярніших утиліт серед *FrontEnd* розробників, яке зберігає високу читабельність коду навіть у великих проектах. Також збільшує організованість коду, що дозволяє набагато легше налагоджувати та розширяти компоненти.

Vue VSCode Snippets – підіймає ергономічність коду та пришвидшує розробку завдяки синтаксичним шаблонам, які повсюди використовуються у *Vue.js* програмах.

Bracket Pair Colorizer – додає різні кольори дужок, що значно спрощує пошук об'єкта або потрібно пари дужок при глибокій вкладеності функцій або інших синтаксичних конструкцій.

Наступний крок у розробці онлайн-сервісу є створення початкового середовища *Vue.js* проекту за допомогою *Vue CLI.*

Vue CLI – це повноцінна система для швидкої розробки на *Vue.js*, яка дозволяє нам зосередитись на розробці логіки й зовнішнього вигляду проекту, та не витрачати час на налаштування конфігурації середовища розробки. Ця система забезпечує такі інструменти: інтерактивне створення проекту через *@vue/cli*, швидке прототипування через *@vue/cli* та *@vue/cli-service-global* без попередньої конфігурації, *runtime*-залежності для можливості оновлення й налаштування за допомогою конфігураційного файлу в проекті та підтримку велику кількість корисних розширень, повний графічний інтерфейс користувача для створення та управління проектами *Vue.js* [17].

Для створення початкової конфігурації проекту за допомогою *Vue CLI* потрібно встановити *Node.js* за посиланням *https://nodejs.org/uk/* для можливості користування стандартним менеджером пакетів *NPM* (*node package manager*), якій в свою чергу надає змогу встановити велику кількість корисних модулів та систем, а саме *Vue CLI* для початку розробки проекту.

Після встановлення вище зазначених інструментів потрібно перейти у командний рядок *cmd.exe* або створити термінал у інтегрованому середовищі розробки *Microsoft Visual Studio Code* та виконати команду "*npm install -g @vue/cli*" для завантаження *Vue CLI.* Відразу після установки буде отримано доступ до команд системи, перевірити вірність встановлення можна виконавши команду "*vue*", яка відображатиме довідкове повідомлення зі списком усіх доступних команд.

Отож, всі утиліти встановленні, тому приступимо до створення проекту за допомогою виконання команди “*vue create conferences-service*” у терміналі, що складається з: “*vue*” – ключового слова *Vue CLI*, “*create*” – команда створення проекту та “*conferences-service*” – назви проекту. Після виконання команди у командному рядку з’явиться можливість вибору налаштувань конфігурації проекту: стандартні (*default*) або ручні (*manual*) налаштування. Онлайн-сервіс включає у себе декілька сторінок та обробку значної кількості даних для таких вимог потрібно обрати ручні (*manual*) налаштування та зазначити у початковій конфігурації такі інструменти як:

1) *BabelJS* – це компілятор, який перетворює сучасний *JavaScript*-код нових стандартів у більш старіші стандарти для запуску нашого веб-сервісу в старих браузерах;

2) Vue-Router – офіційна бібліотека маршрутизації для *Vue.js*, яка допоможе побудувати *SPA* (*Single Page Application*);

3) Vuex – офіційне доповнення до фреймворку *Vue.js*, що легко інтегрується з ним і крім своєї основної функції управління станом (*state, storage*), також дає можливість створювати зліпки стану даних, а також інструменти для тестування та налагодження *web*-додатку;

4) Linter / Formatter – дозволяє відслідковувати синтаксичні помилки та виправляти їх, а також *Formatter* зберігає чистоту, ергономічність та лаконічність нашого коду.

Після вибору необхідних додаткових інструментів з’явиться запитання з вибором розташування конфігураційних налаштувань у окремих файлах або у одному файлі. Обриємо у одному файлі “*package.json*”, адже так зручніше у будь-який момент змінити версію будь-якого пакета або розширити його можливості також там будуть зберігатися основні дані про проект.

Детально розглянемо та розберемо структуру проекту зображену на рис. 3.1., яку створив *Vue CLI.*

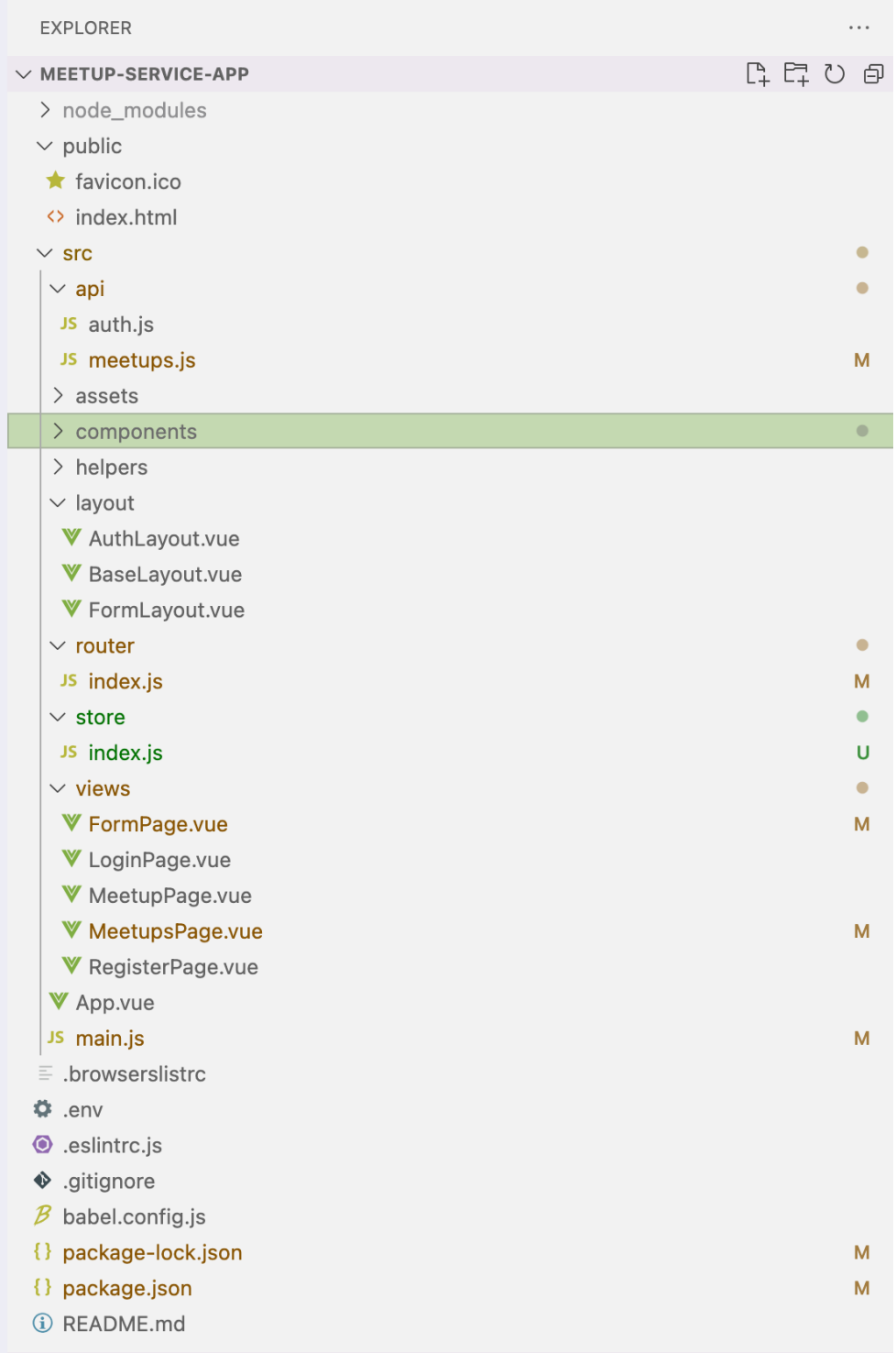

Рис. 3.1. Структура файлів проекту

Папка *node_modules* містить пакети, потрібні для інструментів додатків та розробки. Папка *public* містить статичні файли: корінний *HTML*-файл *index.html* та іконка вкладки онлайн-сервісу у веб-браузері *favicon.ico*, які не будуть включені в процес комплектування фінальної версії проекту. Файл *.gitignore* містить список файлів і папок, виключених з елемента керування версією

файлів *Git*. Файл *babel.config.js* містить налаштування конфігурації для компілятора *Babel*. Файл *package.json* містить список пакетів, необхідних для розробки *Vue.js*, а також команди, що використовуються для інструментів розробки. Файл *package-lock.json* містить повний список пакетів, необхідних для проекту та залежних файлів. Файл *README.md* містить загальну інформацію про проект. Файл *jsconfig.json* завжди лежить у корені *JavaScript* проекту та дозволяє конфігурувати певні функції та поведінку *JavaScript*. Конфігурація проекту заздалегідь містить гнучкі налаштування, але, якщо при необхідності ми можемо змінити їх, написавши певні команди у *vue.config.js*.

Головною папкою проекту є *src* – це директорія, де відбувається основна розробка онлайн-сервісу, також папка має заготовлену структуру: папка *assets*, що використовується для статичних ресурсів, переважно це графічні елементи – фотографії, логотипи, іконки та *CSS* стилі, які вимагає додаток, і які будуть включені в процес комплектування фінальної версії проекту; папка *components* використовується для компонентів програми; папка *router* містить конфігураційний файл для налаштування маршрутизації проекту; папка *store* містить конфігураційний файл для налаштування централізованого сховища (*Vuex*) проекту; папка *views* використовується для компонентів, які будуть відображатися за допомогою функцій *Vue-router* маршрутизації; файл *App.vue* є кореневим компонентом; main.js – це файл *JavaScript*, який створює поточні об'єкти *Vue.js* та є вхідною точкою проекту.

3.2. Верстка веб-сервісу

Отже, досконало розібравшись з завантаженням головних утиліт й інструментів та зрозумівши структуру файлів проекту, можна переходити до верстки шаблонів, сторінок та компонентів.

Верстка онлайн-сервісу складається з: верстки загальних шаблонів, верстки головних сторінок та верстки допоміжних компонентів.

Верстка сайту є одним з найголовніших етапів в процесі розробки веб-сервісу. Саме від того, наскільки правильно і якісно проходить даний етап, залежить подальша функціональність сайту. У разі допущених помилок онлайн-сервіс не буде коректно відображатися та функціонувати.

У верстку входить програмування скелету веб-сервісу на мові розмітки гіпертексту *HTML* та додавання візуальної складової за допомогою спеціальної мови стилю сторінок *CSS*.

Верстка поділяється на такі види: статична, таблична та блочна. А також верстка має ще декілька типів: резинова, семантична, адаптивна та гнучка [18].

Таблична верстка – це вид верстки, якій будується на основі однієї великої *HTML*-таблиці (*<table></table>*), розмір якої відповідає розміру браузерного вікна, в рядках та стовпчиках якої можуть бути інші вкладені таблиці для реалізації блоків верстки. Сьогодні цей підхід вважається застарілим, розробники не використовують його, тому що у пошукових систем браузерів виникають проблеми з індексацією веб-сторінки та код виходить занадто довгим, глибоко вкладеним та виглядає не лаконічно.

Статична верстка – це специфічний вид верстки, якій завжди будується за допомогою фіксованих розмірів ширини контенту незалежно від розмірів гаджета на якому відображається веб-сайт. Цей вид верстки підходить для створення одноразових сторінок з промо-акціями та відтворення лише на ПК, а ось великий недолік такого підходу є відтворення на мобільних приладах з полосою прокручування, що значно погіршує переміщення по сайту такого виду.

Блочна верстка – це найпопулярніший вид верстки серед розробників, якій будується на основі *HTML*-тегу * *, якій є звичайним блочним елементом та не є смисловим або заточеним лише під конкретну ціль верстки, сенсом його наділяє сам розробник, додаючи назву блоку через атрибут *class*.

Резинова верстка – це тип верстки, суть якого полягає у розтягуванні елементів на всю ширину браузера, задаючи ширину у відсотках від ширини браузерного вікна. Мінусом такої верстки є те, що вона не дозволяє відразу

підібрати хороший дизайн для будь-якої роздільної здатності екрану. Адже сторінки по-різному виглядають на кожному гаджеті, хоч по ширині розтягнуті однаково.

Семантична верстка – це підтип блочної верстки, коли з приходом нового стандарту *HTML5* з'явилися нові елементи, які назвою характеризують сенс та роль на сторінці, роблячи верстку структурованою та логічною, також це підтип чудово індексується пошукачами у браузерах, що робить цей підтип популярним та високоефективним.

Адаптивна верстка – це тип верстки, якій створює систему реагування на розмір екрану користувача та адаптує розмір сторінки під конкретний пристрій, що дозволяє коректно відображати онлайн-сервіс як на ПК, так й на смартфонах, планшетах.

Гнучка верстка – це тип верстки, якій дозволяє потрібні блоки перетворити на гнучкі контейнери за допомогою *CSS3* веб-модуля *Flexbox*. Однією з найважливіших особливостей *Flexbox* є його здатність формуватися на основі його середовища перегляду. Контейнери *Flex* можна регулювати за розміром, (як збільшувати, так і зменшувати) щоб уникнути надмірної монополізації простору. Більше того, цей шаблон краще обмежує потік контенту, ніж, наприклад, блокові та вбудовані типи дисплею, які, як правило, односпрямовані. Дійсно, можна не лише задати напрямок потоку флексу на рівні стилю, наприклад вправо, вліво, вгору або вниз; окремі елементи в такому контейнері також можуть бути автоматично перевпорядковані та перебудовані відповідно до наявного макета.

Детально розглянувши види та типи верстки, було обрано розробляти онлайн-сервіс використовуючи найкращі підходи та практики, а саме: гнучку верстку для зручного розташування елементів на сторінках, адаптивну верстку для коректного відображення на мобільних пристроях та семантичну верстку для покращення роботи браузерного пошукача.

Спочатку верстаються шаблони для винесення повторюваних компонентів верстки таких як: шапка сайту (*header*), підвал сайту (*footer*), навігаційне меню (*navigation*) та інші часто вживані елементи на сторінках.

Верстка головного шаблон (*BaseLayout*) має таку структуру:

```
<div class="wrapper bg-grey">
  <Header />
  <main class="main">
    <slot />
  </main>
  <Footer />
 </div>
```

Шаблон розроблений для обгортки модулів та складається з: обгортки (*wrapper*), модулів шапки (*Header*) та підвалу (*Footer*), а також основного змісту <*main*></*main*>, який містить у собі спеціальний *Vue.js*-компонент <*slot/*> для зазначення місця розташування контенту при використанні шаблону. Обгортка (*wrapper*) задає для <*div*> блока такі CSS-правила:

```
.wrapper {
  min-height: 100vh;
  display: flex;
  flex-direction: column;
}
.bg-grey {
  background-color: var(--grey-light);
}
```

Обгортка вказує мінімальну висоту контенту, позиціонує елементи шаблону за допомогою утиліти нового стандарту *CSS3* веб-модуль *Flexbox* та задає колір фону. Компонент шапки сторінки (*header*) містить таку верстку:

```
<header class="header">
  <div>
    <h1>
```

```
          <router-link
            :to="{ name: 'meetups' }"
            class="router-link-exact-active router-link-active"
            ><img src="../assets/logo.svg" alt="Meetups"
          /></router-link>
        </h1>
      </div>
      <nav>
        <router-link v-if="showReturnToMeetups" :to="{ name: 'meetups' }">
          &larr; Повернутися до списку
        </router-link>
        <router-link
          :to="{ name: 'meetups', query: { participation: 'attending' } }"
          >Мої конференції</router-link
        >
        <router-link
          :to="{ name: 'meetups', query: { participation: 'organizing' } }"
          >Організовані конференції</router-link
        >
        <router-link :to="{ name: 'login' }">Вхід</router-link>
        <router-link :to="{ name: 'register' }">Реєстрація</router-link>
        <router-link :to="{ name: 'form' }">Створити конференцію</router-
link>
        <button>Вихід</button>
      </nav>
    </header>
```

Шапка складається з: семантичного тегу <*header*>, логотип-посилання на головну сторінку сервісу реалізований за допомогою компонента маршрутизатора (*vue-router*) <*router-link*></*router-link*> та тегу <*img*/>, якій дозволяє відображати графічні ілюстрації на веб-сторінці; семантичне

навігаційне меню *<nav></nav>*, яке складається з посилань маршрутизатора на різні модулі онлайн-сервісу залежно від ролі користувача. Підвал сторінки (*footer*) задає нижній колонтитул сторінки та містить таку верстку:

```
<footer class="footer">
  <div class="container">
   © 2022. Dmytro Voloshko
     <a href="https://www.linkedin.com/in/dima-voloshko-660922130/">
          My   social profile
      </a>
      <a href="https://github.com/Tamagotchi9/meetup-service-app">
          My   project
      </a>
  </div>
 </footer>
```

Верстка підвалу складається з: ім'я автора, дати створення сервісу, посилання на базу коду онлайн-сервісу та профіль соціальної мережі розробника, які відтворені у *HTML*-посиланні *<a>*. Тег *<main></main>* розміщений в центрі шаблону, та відображає стилізовану інформацію передану при використанні шаблону.

Компонент шаблону автентифікації та авторизації (*AuthLayout*) містить таку верстку:

```
<div class="page page_onboarding">
  <div class="container">
   <h1 class="page__title text-center">{{ title }}</h1>
   <slot></slot>
  </div>
 </div>
```

Даний шаблон був розроблений як обгортка модулів входу та реєстрації, щоб сховати деяку частину логіки, яка є недоступною на етапі авторизації або автентифікації. Даний шаблон складається з: контейнеру, динамічного

заголовку та *Vue.js*-компоненту *<slot/>* для зазначення місця розташування полів входу або реєстрації. Контейнер – це блочний елемент * * з класом *"container"*, якій встановлює такі *CSS*-правила:

```
.container {
  max-width: 1008px;
  width: 100%;
  margin: 0 auto;
}
```

Контейнер задає максимальну допустиму ширину контенту (*max-width*) та позиціонує їх по центру документа за допомогою зовнішніх відступів (*margin*). Динамічний заголовок відтворений за допомогою *HTML*-тегу *<h1></h1>* та відображає заголовок залежно від модуля.

Шаблон-компонент *FormLayout* складається з такої верстки:

```
<div class="page page_meetup-forms">
   <div class="container">
    <h2 class="page__title">{{ title }}</h2>
    <slot></slot>
   </div>
  </div>
```

FormLayout був спроектований та розроблений для обгортання модулів створення і редагування конференції та винесення спільних стильових *CSS*-завдань *HTML*-полів. Шаблон складається з такої структури: контейнер, динамічний заголовок та *Vue.js*-компонент *<slot/>* для зазначення місця розташування форми полів створення та редагування конференції. Логіка даної структури однакова відповідно до минулого шаблону, відмінні лише стильові класи, які задаються внутрішні відступи контенту (*padding*).

Наступним етап – це верстка головних сторінок програми серед яких такі компоненти: *MeetupsPage, MeetupPage, LoginPage, RegisterPage, FormPage.*

Сторінка *MeetupsPage* був розроблений для того, щоб вміщувати у собі верстку відображення, пошуку, сортування та фільтрації конференцій. Дана

сторінка складається з таких тегів та елементів: контейнеру, фільтр-компоненту, текстового поле для пошуку конференцій, елементу перемикача для зміни відображення конференцій. Контейнер задає внутрішню ширину контенту, позиціонування та розміщення. Фільтр-компонент – внутрішній елемент, якій відображає текстові радіо-кнопки за допомогою тегу *☐*, щоб надати змогу *JavaScript* фільтрувати наші події. Пошук реалізований за допомогою текстового *HTML*-елементу *<input/>* та стилізований з іконкою "лупи" всередині. Перемикач вміщує у собі дві лаконічно стилізовані кнопки для зміни табличного виду конференцій у календарний та навпаки. Вікно результату верстки *MeetupsPage* зображено на рис. 3.2.

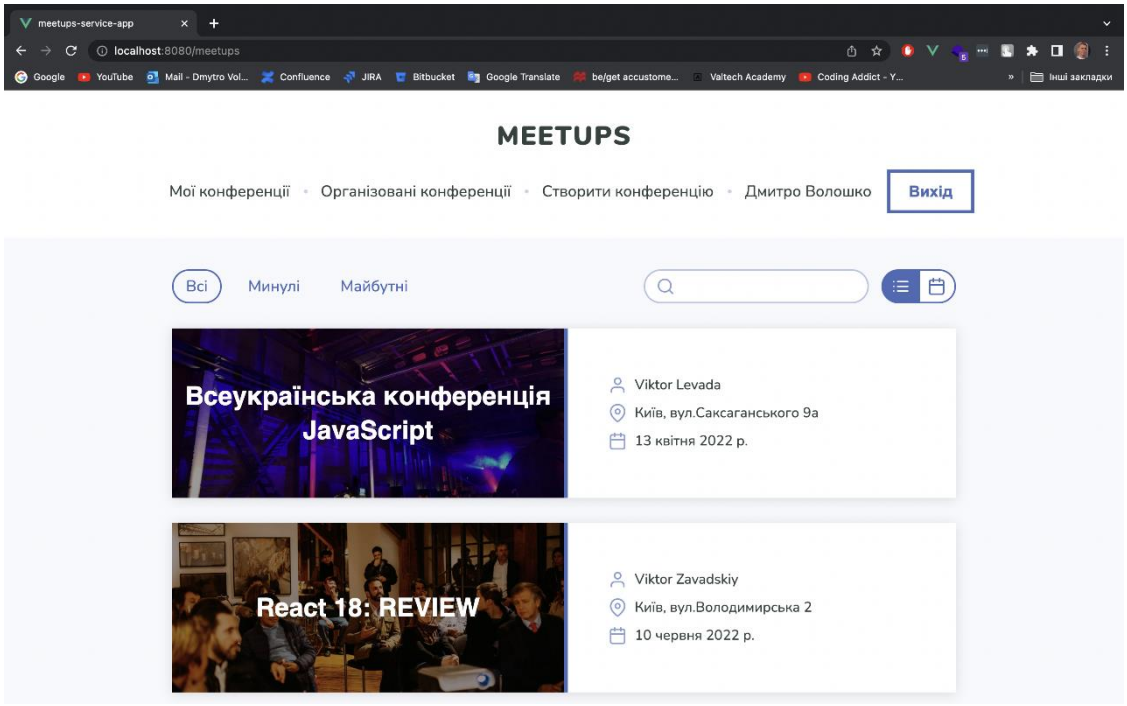

Рис. 3.2. Головна сторінка онлайн-сервісу організації конференцій

MeetupPage – це сторінка розроблена для детального відображення інформації конкретної конференції. Верстка даного компонента складається з таких елементів: компонент-обкладинка зображення конференції (*meetup-cover*), компонент детального опису події (*meetup-description*), програма (*meetup-agenda*) та загальна інформація про захід (*meetup-info*), автора, місце проведення, а також блок стилізованих обробників подій (*button-list*).

Компонент-обкладинка розміщує у собі фотографію, яка розтягується на весь блок * * за допомогою каскадних стилів *CSS* та заголовку *<h1></h1>* із назвою події. Компонент детального опису містить лише один тег * * у якому відображається опис події. Програма – це блок з стилізованих елементів програми події з такою верстkою: графічна іконка відповідна до кожного типу елемента програми події *<img/>*, початок та кінець події відображений у тегу *<time></time>*, підзаголовок назви елемента програми конференції *<h5></h5>*, ім'я спікера й мова промови відтворені у тегу **, а також короткий абзац опису елемента програми * *. Загальна інформація події відтворена за допомогою не нумерованого списку *
*, якій містить елементи списку *
* зі ім'я автору, місцем проведення й датою та часом проведення, а також з тематичною іконкою *<img/>* у кожному елементі. Набір елементів оброблювачів подій реалізований компонентами *PrimaryButton, SecondaryButton, DangerButton* в основі яких лежить *HTML*-елемент *<button></button>* з індивідуальними *CSS* стилями для наголошення сенсу оброблювача. Вікно результату верстки *MeetupPage* зображено на рис. 3.3.

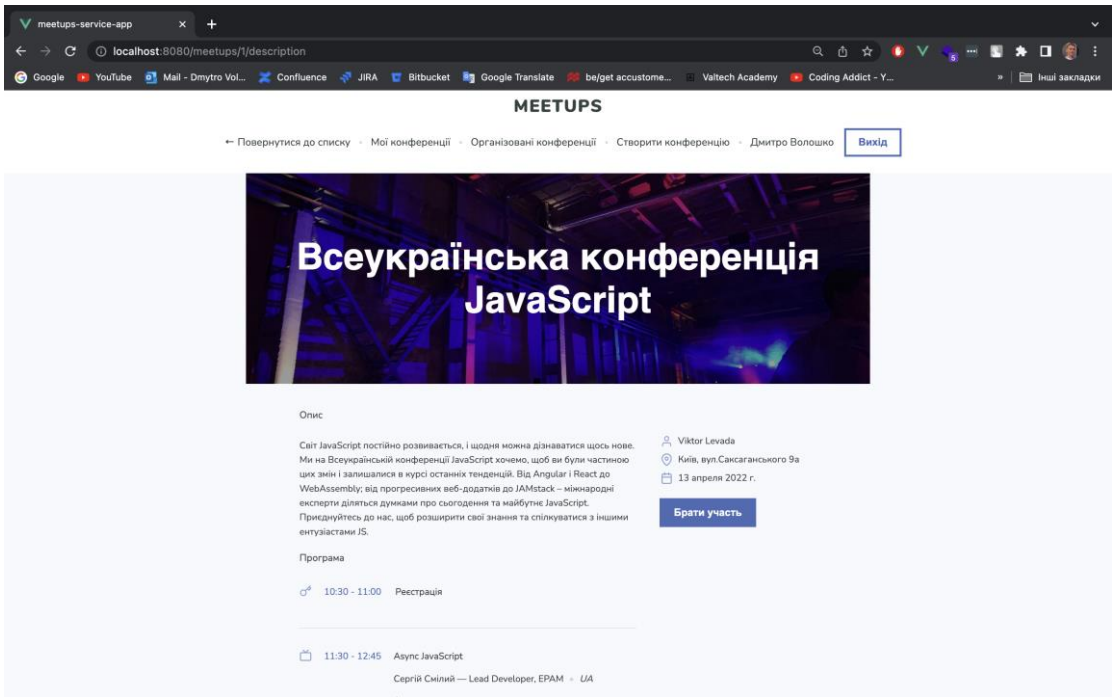

Рис. 3.3. Вікно сторінки конференції

LoginPage – це сторінка, яка відображає поля для входу в профіль користувача. Даний компонент використовує обгортку *AuthLayout* та містить таку верстку: форму, поле *email*, поле пароль, *submit*-кнопку входу та посилання маршрутизатора на сторінку реєстрації (*RegisterPage*). У *HTML* будь-які поля, які будуть якимось чином взаємодіяти з сервером верстаються у тегу *<form></form>*. Поле електронної адреси створено за допомогою тегу полів з типом email (*<input type="email"/>*), а поле паролю з типом password (*<input type="password"/>*). Кнопка входу реалізована завдяки змозі задавати тип кнопки у *HTML* (<button type="submit"></button>), що дозволить обробити певну функцію при імплементації *JavaScript*.

RegisterPage – це сторінка реєстрації користувача, яка має схожу будову до минулої сторінки, відрізняється лише декілька додатковими полями необхідними для автентифікації: поле для вводу ім'я, поле для перевірки вірності паролю та *checkbox* елемент для створення логічного параметра погодження з умовами сервісу. Вище перелічені елементи побудовані за допомогою *<input/>* тегу, різниця полягає у тому, що ім'я має тип за замовчуванням *type="text"*, повтор паролю *type="password", checkbox* елемент *– type="checkbox".*

FormPage – це сторінка була розроблена для створення та редагування конференції. Верстка має таку структуру: шаблон *FormLayout*, форму, яка містить пусті поля для запису даних про конференцію або заповнені для редагування певної конференції, а також спеціальний компонент *MeetupAgendaItemForm* для поетапного створення програми конференції. Форма реалізована *HTML*-елементом *<form></form>*, якій обгортає поля тегом *<fieldset></fieldset>* для групування великої кількості полів. Компонент має два текстових поля (*<input/ type="text">*) – "назва конференції" та "місце проведення", поле типу дата (*<input/ type="date">*) – "дата проведення", поле для вводу декількох рядків тексту *(<textarea><.textarea>*) та поле файлового типу для додавання/змінити/видалення зображення конференції *(<input/ type="file">*). У компоненті *MeetupAgendaItemForm* головна верстка це:

випадаючий список *DropdownButton*, в основі якого лежить *HTML*-елемент *<select></select>*, для вибору типу події конференції, два часових поля *(<input/ type=”time”>* для встановлення початку та кінця події, а також ще декілька другорядних полів залежних від вибору типу події. Завершальна верста даної сторінки – це секція кнопок *<button></button>* для збереження або відміни змін. Вікно результату верстки *FormPage* зображено на рис. 3.4.

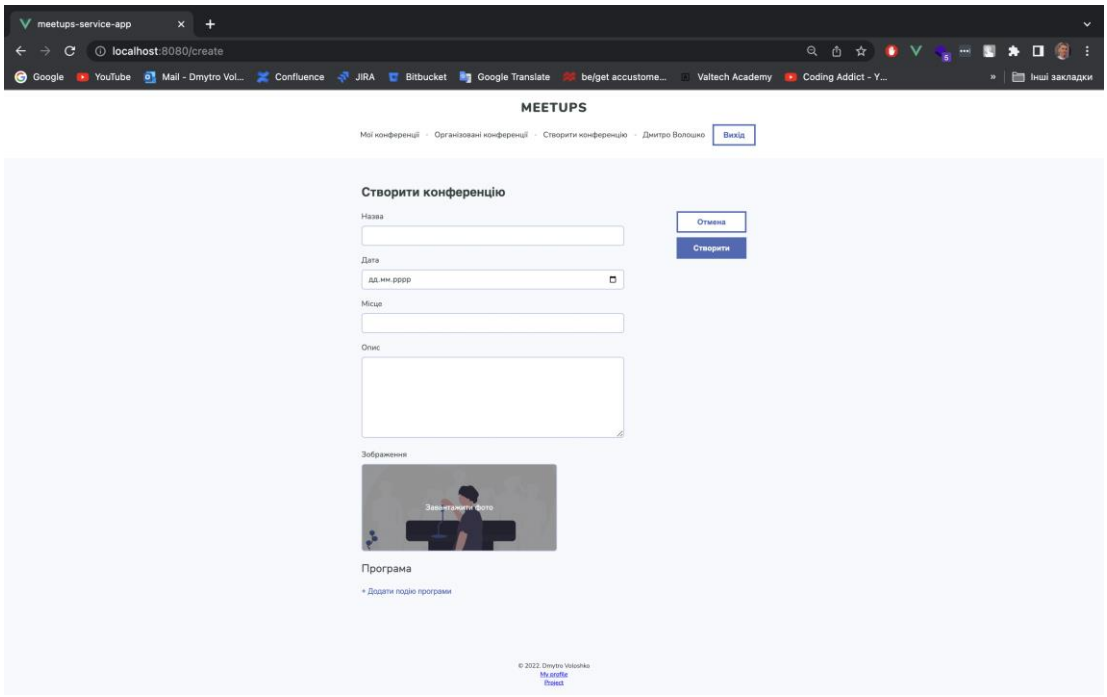

Рис. 3.4. Вікно сторінки для створення конференції

3.3. Модулі веб-сервісу

Після етапу розробки верстки онлайн-сервісу, відбувається імплементація головних функцій для встановлення взаємозв'язку з базою даних *Firebase Realtime Database*, обробки подій, налаштування авторизація й автентифікації *Firebase Authentication* та реалізація допоміжних функцій.

Ознайомившись з клієнт-серверною архітектурою у другому розділі, саме час розповісти якій був використаний архітектурний підхід, за допомогою якого інтерфейсу був відтворений зв’язок з сервером та які інструменти використовувалися на клієнті для виконання цих завдань.

API (*Application Programming Interface*) – це набір способів і правил взаємодії та обміну даними між різними програмами. Спілкування відбувається за допомогою класів, методів, функцій і структур. Іноді комунікації можливі за допомогою констант однієї програми, до яких звертаються інші.

REST (*Representational State Transfer*) – це архітектурний стиль, якій використовує *HTTP* у ролі протоколу передачі даних для запитів й відповідей. В архітектурному стилі немає обмежень *XML* як формат повідомлення. *API REST* можуть використовувати будь-який формат повідомлень, який бажають використовувати розробники *API*, включаючи *XML*, *JSON*, *Atom*, *RSS*, *CSV*, *HTML* та інші. Незважаючи на різноманітність параметрів формату повідомлень, *API REST* онлайн-сервісу використовує *JSON* (нотацію об'єктів *JavaScript*) як формат повідомлень за замовчуванням. JSON використовується, тому що забезпечує легкий, простий та більш гнучкий формат повідомлень, який збільшує швидкість зв'язку.

Ресурс – це представлення віртуального об'єкту (такого як зображення), реального об'єкту (клієнт) чи колекції об'єктів. Взагалі, ресурс може бути чим завгодно, розробник *API* вирішує що в нього буде ресурсом.

API-інтерфейси *REST* фокусуються на ресурсах (тобто речах, а не діях) та способах доступу до ресурсів. Ресурси, зазвичай, є різними типами інформації. Ви отримуєте доступ до ресурсів через *URL*, так як перехід до *URL*-адреси у вашому браузері дозволяє підключитися до інформаційного ресурсу. *URL*-адреси супроводжуються методом, який вказує на те, як ви хочете взаємодіяти з ресурсом.

Загальні методи включають *GET* (читання), *POST* (створення), *PUT* (оновлення) та *DELETE* (видалення). Кінцева точка зазвичай включає параметри запиту, які визначають докладнішу інформацію про представлення ресурсу, який потрібно побачити. Ось приклад кінцевої точки для отримання наявних конференцій з серверу: “*https://meetups-service-default-rtdb.europe-west1.firebasedatabase.app/meetups.json*” [19].

Fetch API – це інтерфейс для отримання ресурсів побудований на основі *Rest API*. *Fetch API* забезпечує узагальнене визначення об'єктів *Request* та *Response* (та інших речей, пов'язаних із мережними запитами).

При розробці функцій для запитів з клієнту було обрано використовувати *Axios* – це клієнт *HTTP* на основі *Promise* для *node.js* та браузера для зручності використання.

На стороні клієнта використовується *Vuex* (бібліотека керування станом) – це глобальне сховище для *Vue.js* проектів, де будуть зберігатися необхідні дані з серверу, а також будуть описані головні функції у роботі з сервером [20].

Функція *signUp* – це асинхронна функція (*action*) складається з такого коду:

```
import {getAuth, createUserWithEmailAndPassword } from 'firebase/auth';
async register(context, email, password) {
  createUserWithEmailAndPassword(getAuth(), email, password)
    .then(() => {
      SuccessMessage('Користувач успішно зареєстрований!')
      this.$router.push({ name: 'login'})
    })
    .catch((error) => ErrorMessage(error))
}
```

Попередньо потрібно імпортувати функції: *getAuth*, *createUserWithEmailAndPassword* з модуля *firebase/auth* для того, щоб зареєструвати користувача у базі даних. Функція має три параметри: *context*, *email*, *password*. Параметр *context* – це об’єкт *Vuex* для взаємодії з іншими частинами сховища, *email* та *password* – це поля поштової адреси та паролю. Функція спрацьовує при натисканні на оброблювач подій під назвою “Зареєструватись” на сторінці автентифікації (*RegisterPage*). Реалізація функції побудована на двох інших методах, які ми отримуємо після встановлення *npm*-пакету “*firebase*” та його налаштування, стають доступні такі методи як: *getAuth* та *createUserWithEmailAndPassword*. Метод *createUserWithEmailAndPassword*

приймає в себе три параметри: *callback*-функцію *getAuth, email* та пароль, що дозволяє додати користувача до бази даних при умові проходження валідації.

Функція signIn – це асинхронна функція (*action*), яка складається з такого коду:

```
import {getAuth, signInWithEmailAndPassword } from 'firebase/auth';
async register(context, email, password) {
 signInWithEmailAndPassword(getAuth(), email, password)
  .then(() => {
   SuccessMessage('Користувач успішно авторизувався!')
   this.$router.push({ name: 'meetups'})
  })
```

Функція створена для входу в обліковий запис користувача. Функція спрацьовує, коли користувач натискає на оброблювач подій під назвою “Увійти” на сторінці автентифікації (*LoginPage*) та відпрацьовує функція signIn з схожими методами як у *signUp*, але замість метода *createUserWithEmailAndPassword,* потрібно імпортувати метод signInWithEmailAndPassword та передати такі ж аргументи як у *createUserWithEmailAndPassword* при реєстації, після чого відправиться запит до серверу, якій перевірить чи є такий користувач у базі даних.

Функція *getMeetups* – це асинхронна функція (*action*) з таким кодом:

```
async getMeetups(context) {
state.isLoadingMeetups = true;
 this.$axios(“https://meetups-service-default-rtdb.europe-
west1.firebasedatabase.app/meetups.json”)
 .then(meetups => {
  context.commit('SET_MEETUPS', meetups);
  state.isLoadingMeetups = false;
 })
 .catch(error => {
  ErrorMessage(error);
```

```
  })
}
```

Функція має один параметр *context*, якій ми отримали завдяки тому, що описуємо функції у сховищі *Vuex*, цей параметр дозволяє нам спілкуватися з іншими структурними частинами *Vuex*. Функція спілкується з сервером для отримання конференцій з бази даних, щоб *HTTP*-клієнт *Axios* за допомогою *GET*-метода зі вказаним шляхом *URL:* "*https://meetups-service-default-rtdb.europe-west1.firebasedatabase.app/meetups.json*" отримав усі наявні конференції, якщо запит успішний, то конференції збережуться у *Vuex*-сховище за допомогою *Vuex*-мутації *SET_MEETUPS* та вимкне прапорець *isLoadingMeetups*, якій відповідає за відображення стану завантаження конференцій. Якщо запит невдалий, тоді з'явиться графічний компонент *ErrorMessage*, якій сповістить про невдалу спробу. Модуль *MeetupsPage* за допомогою спеціального *Vuex*-метода *dispatch* буде викликати функцію *getMeetups* в хуку життєвого циклу компонента *created* для того, щоб отримати конференції до повного створення браузерного *DOM*-дерева та відтворити їх, відразу коли дерево буде готове та сторінка відображена.

Функція *getMeetup* – це асинхронна функція (*action*) з таким кодом:

```
async getMeetup(context, meetupId) {
state.isLoadingMeetup = true;
  this.$axios("https://meetups-service-default-rtdb.europe-
west1.firebasedatabase.app/meetups/meetupId(ідентифікатор_конференції).j
son")
  .then(meetup => {
    context.commit('SET_MEETUP', meetup);
    state.isLoadingMeetup = false;
  })
  .catch(error => {
    ErrorMessage(error);
  })
```

}

Функція має два параметри: *context* та *id* (ідентифікатор конференції). Роль *context* описано у функції *getMeetups*, а *id* допомагає отримати одну конкретну конференцію. *Axios* відправляє *GET*-запит на адресу "*https://meetups-service-default-rtdb.europe-west1.firebasedatabase.app/meetups/meetupId(*ідентифікатор_конференції*).json*", якщо запит успішний, то отриманий об'єкт конференції (*meetup)* збережеться у *Vuex*-сховище за допомогою *Vuex*-мутації *SET_MEETUP* та вимкне прапорець *isLoadingMeetup*, якій відповідає за відображення стану завантаження конференції. Якщо запит невдалий, то компонент *ErrorMessage* відобразить помилку. Дана функція запускається під час кліку на конкретну конференцію у списку на сторінці *MeetupsPage*, тоді модуль *MeetupPage* отримає конкретну конференцію зі *Vuex*-сховища та відмалює його.

Функція *createMeetup* – це асинхронна операція (*action*) з таким кодом:

```
async createMeetup(context, newMeetup) {
  this.$axios.post("https://meetups-service-default-rtdb.europe-west1.firebasedatabase.app/meetups.json", newMeetup)
    .then(successMsg => {
      SuccessMessage(successMsg.text)
      state.isLoadingMeetup = false;
    })
    .catch(error => {
      ErrorMessage(error);
    })
}
```

Функція створює конференцію, вона має два параметри: *context* та *newMeetup*. Параметр *context* відомий з минулих функцій, а *newMeetup* – це об'єкт нової конференції, яка буде передана на сервер та записана у базу даних. На сторінці *FormPage*, користувач створює конференцію та натискає оброблювач подій (кнопку) "Створити" для виклику функції *createMeetup* з

Vuex-сховища, яка приймає аргументом об’єкт нової конференції та посилає *POST*-запит на адресу “*https://meetups-service-default-rtdb.europe-west1.firebasedatabase.app/meetups.json*” за допомогою *Axios*. У разі успішного виконання запиту, ми отримає відповідь від сервера, що конференція була успішно додана у базу даних за допомогою компонента *SuccessMessage*, а у разі помилки – *ErrorMessage*. Після даної операції, при переході на головну сторінку (*MeetupsPage*) з’явиться додана конференція у списку, тому що функція *getMeetups* відпрацює знову, так як вхідні дані зміняться, а *Vuex*-сховища є реактивним та реагує на будь-які зміни.

Функція *editMeetup* – це асинхронна операція (*action*) складається з таких операторів:

```
async editMeetup(context, editedMeetup) {
  this.$axios.put(“https://meetups-service-default-rtdb.europe-
west1.firebasedatabase.app/meetups/id(ідентифікатор_конференції).json”,
editedMeetup)
    .then(successMsg => {
      SuccessMessage(successMsg.text)
      state.isLoadingMeetup = false;
    })
    .catch(error => {
      ErrorMessage(error);
    })
}
```

Функція була створена для редагування конференції та має два параметри: *context* та *editedMeetup*. Параметр *context* відомий з минулих функцій, а *editedMeetup* – це об’єкт відредагованої конференції. На сторінці конференції (*MeetupPage*), коли користувач є організатором конференції, у нього з’являється оброблювач подій під назвою “Редагувати” за допомогою якого маршрутизатор (*vue-router*) переходить на сторінку *FormPage* із заповненими полями інформацією з об’єкту конференції, звідки відбувся

перехід. При зміні даних у полях та кліку по оброблювачу подій під назвою “Редагувати”, *Axios* буде відправлятися *PUT*-запит за адресою “*https://meetups-service-default-rtdb.europe-west1.firebasedatabase.app/meetups/id(ідентифікатор_конференції).json*” та у разі успішної відповіді отримаємо повідомлення, що дані конкретної конференції були змінені за допомогою компонента *SuccessMessage*, а в противному випадку – *ErrorMessage*.

Функція *deleteMeetup* – це асинхронна функція (*action*), яка має таку будову:

```
async deleteMeetup(context, deletedMeetup) {
  this.$axios.delete(“https://meetups-service-default-rtdb.europe-west1.firebasedatabase.app/meetups/id(ідентифікатор_конференції).json”, deletedMeetup)
    .then(successMsg => {
      SuccessMessage(successMsg.text)
      state.isLoadingMeetup = false;
    })
    .catch(error => {
      ErrorMessage(error);
    })
}
```

Даний метод видаляє конференцію та доступний лише організатору події, якій має право видалити конференцію за допомогою оброблювача подій під назвою “Видалити”, що дозволяє *Axios* виконати *DELETE*-запит за адресою “*https://meetups-service-default-rtdb.europe-west1.firebasedatabase.app/meetups/id(ідентифікатор_конференції).json*” та видалити конкретну конференцію з бази даних, що призведе до повторного виконання функції *getMeetups* та оновлення списку на головній сторінці.

3.4. Висновки до розділу

Було вибрано та налаштоване інтегроване середовище розробки, встановлено необхідні програми та розширення, а також було встановлено утиліту для створення *Vue.js*-проектів *Vue CLI* та налаштовано під потреби розробки.

Верстка сайту є одним з найголовніших етапів в процесі розробки веб-сервісу. Саме від того, наскільки правильно і якісно проходить даний етап, залежить подальша функціональність сайту. У разі допущених помилок онлайн-сервіс не буде коректно відображатися та функціонувати. Був виконаний один із головних етапів створення онлайн-сервісу – верстка. Суть якої полягала у верстці головних шаблонів та компонентів з додаванням *CSS*-стилів.

Було виконано верстку головних сторінок програми: *MeetupsPage* для відображення, пошуку, сортування та фільтрації конференцій; *MeetupPage* для детального відображення інформації конкретної конференції; *LoginPage* − сторінка, яка відображає поля для входу в профіль користувача; *RegisterPage* – це сторінка реєстрації користувача; *FormPage* − сторінка для створення та редагування конференції.

Після етапу розробки верстки онлайн-сервісу, відбулася імплементація головних функцій для встановлення взаємозв'язку з базою даних *Firebase Realtime Database*, обробки подій, налаштування авторизація й автентифікації *Firebase Authentication* та реалізація допоміжних функцій. Проект було виконано з дотриманням діючих стандартів та положень [21], [22].

ВИСНОВКИ

Веб-сервер є програмою, що запускається на підключеному до мережі комп'ютері і використовує протокол *HTTP* для передачі даних. Зазвичай така програма отримує по мережі *HTTP*-запит на певний ресурс, знаходить відповідний файл на локальному жорсткому диску і відправляє його по мережі комп'ютеру, що надіслав запитав. Загальновідомим прикладом гіпертексту є веб-сторінки − документи на *HTML* (на гіпертекстовій мові розмітки), які розміщені у Всесвітній павутині.

В цілому можна зробити висновок, що Всесвітня павутина побудована з використанням: протоколу *HTTP*, мови *HTML* і уніфікованих локаторів ресурсів *URL*.

Онлайн-сервіс (веб-сервіс) – це сукупність програмно-технічних засобів, інтегрованих з метою ефективного опрацювання веб-ресурсів, які знаходяться у веб-просторі.

Існуючі комплексні програмні продукти для автоматизованого вирішення проблеми організації конференцій, наприклад, популярні сервіси під час пандемії *Covid-19* такі як *Zoom, MS Teams* надають змогу проводити конференції у онлайн-форматі, але зі спадом захворювань пандемією з’явилася потреба у проведенні конференцій, самітів, подій у оф-лайн форматі, тому що багато прикладних сфер потребують взаємодії з учасниками у реальному житті. Розглянуті сервіси мають недоліки, які було розглянуто та усунуто у розробленому онлайн-сервісі для кращого оцінки користувачів.

Розглянуто декілька існуючих сервісів: веб-сервіс *FWdays*, онлайн-платформа *DOU*, онлайн-сервіс *Bizzabo*. Розглянуто їх недоліки та переваги.

Головна мета була − проектування онлайн-сервісу, який включає в себе необхідні функції для проведення всіх етапів організації та проведення

наукових конференцій з мінімальними витратами часу організаційного комітету конференції і використанням людського ресурсу. Також важливий аспект полягає у зручності використання даного сервісу організаторами і користувачами на всіх етапах процесу організації та проведення конференції.

Клієнт-сервер (англ. *Client-server*) – це архітектура комп'ютерної мережі, в якій багато клієнтів (*remote computers*) запитують та отримують інформацію з централізованого сервера (*host-computer*). Клієнтські комп'ютери надають інтерфейс, щоб дозволяє користувачеві комп'ютера отримати відповідь від сервера та відображати результати, які повертає сервер.

При проектуванні додатку для організації конференцій було використано наступну архітектуру. Оскільки, більшість програмних продуктів під *WEB* використовують на концептуальному рівні клієнт-серверну архітектуру, то це призвело до розділення логіки роботи *web*-орієнтованих систем на дві частини – *front-End* та *back-End*. В програмній інженерії терміни «*front-End*» та «*back-End*» розрізняють за принципом розділення відповідальності між рівнем представлення та рівнем доступу до даних відповідно.

Firebase Realtime Database – це база даних типу *NoSQL*, яка використовує веб-сокети, що дозволяє клієнту отримувати інформацію в реальному часі – без необхідності відправляти запити *GET* на сервер. Веб-сокети – це технологія, яка дозволяє створювати інтерактивне з'єднання між клієнтом (браузером) та сервером для обміну повідомленнями у режимі реального часу. Веб-сокети, на відміну від *HTTP*, дозволяють працювати з двонаправленим потоком даних, що робить цю технологію унікальною.

Firebase Authentication дозволяє виконувати вхід в програму за допомогою системи, з якої користувачі добре знайомі, тобто *Facebook* або *Google*. Потім додаток може зберігати дані і персональні налаштування користувача в захищеному хмарному сховище і забезпечувати доступ до них на всіх його пристроях. *Firebase* забезпечує служби серверної частини, прості пакети розробника і готові бібліотеки інтерфейсу для автентифікації користувачів різних додатків на будь-яких платформах. Автентифікацію можна

виконувати за допомогою паролів і інтегрованих систем ідентифікації – *Google*, *Facebook*, *Twitter* та ін. Це значно спрощує процедуру входу в додаток і забезпечує надійний захист.

Проаналізувавши розглянуті онлайн-сервіси у попередньому розділі, був спроектований онлайн-сервіс, який має мету максимально спрощувати і автоматизувати процес організації та проведення конференцій. Умовно всі завдання сервісу можна розділити на шість модулів: авторизація, автентифікація, головна сторінка, сторінка конференції, організація конференції, профіль користувача.

Чітко спроектований та описаний онлайн-сервіс є важливим кроком перед конструюванням та кодуванням веб-сервісу, далі розглянемо розробку, тобто яким чином створено онлайн-сервіс організації конференцій.

Було вибрано та налаштоване інтегроване середовище розробки, встановлено необхідні програми та розширення, а також було встановлено утиліту для створення *Vue.js*-проектів *Vue CLI* та налаштовано під потреби розробки.

Верстка сайту є одним з найголовніших етапів в процесі розробки веб-сервісу. Саме від того, наскільки правильно і якісно проходить даний етап, залежить подальша функціональність сайту. У разі допущених помилок онлайн-сервіс не буде коректно відображатися та функціонувати. Був виконаний один із головних етапів створення онлайн-сервісу – верстка. Суть якої полягала у верстці головних шаблонів та компонентів з додаванням *CSS*-стилів.

Було виконано верстку головних сторінок програми: *MeetupsPage* для відображення, пошуку, сортування та фільтрації конференцій; *MeetupPage* для детального відображення інформації конкретної конференції; *LoginPage* − сторінка, яка відображає поля для входу в профіль користувача; *RegisterPage* – це сторінка реєстрації користувача; *FormPage* − сторінка для створення та редагування конференції.

Після етапу розробки верстки онлайн-сервісу, відбулася імплементація головних функцій для встановлення взаємозв'язку з базою даних *Firebase Realtime Database*, обробки подій, налаштування авторизація й автентифікації *Firebase Authentication* та реалізація допоміжних функцій. Проект було виконано з дотриманням діючих стандартів та положень [21], [22].

СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАННИХ ДЖЕРЕЛ

1. Мэтью Макдональд Веб-разработка. Исчерпывающее руководство. – СПб.: «Питер», 2017. – 640с.
2. Барри Поллард *HTTP*/2 в действии – С.: «ДМК Пресс», 2021. – 424с.
3. Элизабет Фримен, Эрик Фриман. Изучаем программирование на *HTML5* – С.: «Питер Пресс», 2018. – 592с.
4. Мельник Р.А. Програмування веб-застосувань (фронт-енд та бек-енд) – М.: «Видавництво Львівської політехніки», 2018. – 248с.
5. Мартин Фаулер Шаблоны корпоративных приложений – М.: «Диалектика-Вильямс», 2019. – 544с.
6. Р. Мартин Чистая архитектура. Искусство разработки программного обеспечения. – СПб.: «Питер Пресс», 2018. – 352 с.
7. Эрик Фримен, Элизабет Робсон, Берт Бейтс, Кэти Сиерра. Head First. Патерни проєктування – М. : «Фабула», 2020. – 672с.
8. С. Макконнелл Совершенный код. Мастер-класс / Пер. с англ. – СПб.: «БХВ», 2022. – 896 с.
9. Р. Мартин Чистый код: создание, анализ и рефакторинг. – СПб.: «Питер», 2019. – 464 с.
10. *Kyle Simpson Up&Going. Published by O'Reilly Media, Inc, 2015*. – 73 с.
11. *Kyle Simpson Types&Grammar. Published by O'Reilly Media, Inc, 2015*. – 198 с.
12. *Kyle Simpson ES6&Beyond. Published by O'Reilly Media, Inc, 2015*. – 278 с.
13. Алекс Бэнкс, Ева Порселло. *React*: современные шаблоны для разработки приложений. – СПб.: «Питер», 2022. – 320 с.
14. Файн Я., Моисеев А. *Angular* и *TypeScript*. Сайтостроение для профессионалов – С.: «Питер Пресс», 2018. – 464с.
15. *Vue.js Documentation* [Електронний ресурс] – Режим доступу: *https://vuejs.org/v2/guide/* (дата звернення 25.03.2022)

16. *Firebase Documentation* [Електронний ресурс] – Режим доступу: *https://firebase.google.com/docs* (дата звернення 25.03.2022)
17. *Vuex Documentation* [Електронний ресурс] – Режим доступу: *https://vuex.vuejs.org/* (дата звернення 25.03.2022)
18. *Learn.javascript* [Електронний ресурс] – Режим доступу: https://learn.javascript.ru/ (дата звернення 25.03.2022)
19. *w3schools* [Електронний ресурс] – Режим доступу: https://www.w3schools.com/ (дата звернення 22.03.2022)
20. *Developer Mozilla documentation* [Електронний ресурс] – Режим доступу: *https://developer.mozilla.org/* (дата звернення 22.03.2022)
21. ДСТУ 3008-95. Документація. Звіти у сфері науки і техніки. Структура і правила оформлення / Держстандарт України. – Вид. офіц. – [Чинний від 1995-02-23]. – Київ, 2007. – 86с.
22. Слободян О. Положення про дипломні роботи (проекти) випускників Національного авіаційного університету. – К.: Видавництво НАУ, 2017. – 63с.