МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ІНЖЕНЕРНИЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ім. Ю.М. Потебні
ЗАПОРІЗЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ

КАФЕДРА ЕЛЕКТРОНІКИ, ІНФОРМАЦІЙНИХ СИСТЕМ ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Кваліфікаційна робота

перший (бакалаврський)
(рівень вищої освіти)

на тему **Система управління та проведення відеоконференцій**

Виконав: студент 4 курсу, групи 6.1211-пзс
спеціальності 121 Інженерія програмного забезпечення
(код і назва спеціальності)
освітньої програми Програмне забезпечення систем
(код і назва освітньої програми)
О.О. Петренко
(ініціали та прізвище)
Керівник к.ф-м.н., доцент, доцент кафедри ЕІС та ПЗ
О.М. Міхайлуца
(посада, вчене звання, науковий ступінь, підпис, ініціали та прізвище)
Рецензент директор ТОВ «Дискус»
Р. О. Лютий
(посада, вчене звання, науковий ступінь, підпис, ініціали та прізвище)

Запоріжжя

2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ІНЖЕНЕРНИЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ім. Ю.М. Потебні
ЗАПОРІЗЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ

Кафедра електроніки, інформаційних систем та програмного забезпечення
Рівень вищої освіти _____перший (бакалавський)_____
Спеціальність ___121_Інженерія програмного забезпечення_____
(код та назва)
Освітня програма ___Програмне забезпечення систем____
(код та назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри_____Т. В. Критська
“_01_” __березня_____2025_року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Петренку Олександру Олександровичу
(прізвище, ім’я, по батькові)

1.Тема роботи Система управління та проведення відеоконференцій
керівник роботи Міхайлуца Олена Миколаївна, к.т.н., доцент
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом ЗНУ від 27.12.2024 № 2029-с

2. Строк подання студентом кваліфікаційної роботи 02.06.2025
3. Вихідні дані бакалаврської роботи
- комплект нормативних документів ;
- технічне завдання до роботи.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
- огляд та збір літератури стосовно теми кваліфікаційної роботи;
- огляд та аналіз існуючих рішень та аналогів;
- дослідження проблеми збору та аналізу даних з веб-сторінок;
- створення програмного продукту та його опис;
- перелік вимог для роботи програми;
- тестування програмної системи та розробка висновків.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
слайдів презентації

6. Консультанти розділів бакалаврської роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата Завдання прийняв

7. Дата видачі завдання 01.03.2025

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів кваліфікаційної роботи	Примітка
1	Аналіз предметної області	01.03 – 05.03.25	виконано
2	Огляд та збір матеріалу стосовно теми роботи	06.03 – 12.03.25	виконано
3	Формулювання основної задачі дипломної роботи та узгодження її з науковим керівником	13.03 – 15.03.25	виконано
4	Аналіз існуючих методів розв’язання задачі	16.03 – 22.03.25	виконано
5	Вибір реалізації застосунку	23.03 – 26.03.25	виконано
6	Проектування застосунку	27.03 – 04.03.25	виконано
7	Програмна реалізація застосунку	05.04 – 15.04.25	виконано
8	Реалізація користувацького інтерфейсу	16.04 – 21.04.25	виконано
9	Тестування застосунку	22.04 – 28.04.25	виконано
10	Оформлення звіту. Оформлення презентації. Отримання рецензії	29.04 – 29.05.25	виконано

Студент _____ О.О. Петренко
(підпис) (прізвище та ініціали)

Керівник роботи _____ О.М. Міхайлуца
(підпис) (прізвище та ініціали)

Нормоконтроль пройдено

Нормоконтролер _____ І. А. Скрипник
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Сторінок — 89

Рисунків — 23

Лістинги — 25

Джерел — 13

Петренко О.О. Система управління та проведення відеоконференцій: кваліфікаційна робота бакалавра спеціальності 121 «Інженерія програмного забезпечення» / наук. керівник О.М. Міхайлуца. Запоріжжя : ЗНУ, 2025. 89 с.

У даному проекті розробляється система управління для проведення відеоконференцій. А саме веб-застосунок з Rest Api, інтеграція з Jitsi Meet SDK та розробка мобільного додатку, за допомогою фреймворку Flutter, для зручності усіх користувачів. Веб-застосунок надає користувачам зручний інструмент для створення та приєднання до відеозустрічей, систему авторизації для різних типів ролей користувачів. Панель адміністратора для повного контролю над усім додатком та користувачами. Система дозволяє організовувати, аналізувати відеозустрічі, а також керувати всіма налаштуваннями системи. В мобільному додатку користувачі можуть створювати облікові записи, редагувати інформацію, бачити історію конференцій та брати в них участь.

Веб-застосунок та мобільний додаток мають приємний і інтуїтивний для користувачів дизайн.

Проведено повноцінне тестування працездатності розробленого мобільного додатка та веб-застосунку. Виконано порівняння з існуючими великими проектами.

Ключові слова: *відеоконференції, веб-застосунок, кросплатформність, мобільний додаток, авторизація, система, фреймворк*.

ABSTRACT

Pages — 89

Drawings — 23

Listings — 25

Source — 13

Petrenko O.O. System for Managing and Conducting Video Conferences: Bachelor's Qualification Work in the Specialty 121 “Software Engineering” / Scientific Supervisor O.M. Mikhailutsa. Zaporizhzhia : ZNU, 2025. 89 p.

In this project, a management system for video conferencing is being developed. This includes a web application with Rest Api, integration with Jitsi Meet SDK, and development of a mobile application using the Flutter framework for the convenience of all users. The web application provides users with a convenient tool for creating and joining video meetings, an authorisation system for different types of user roles. An administrator panel for full control over the entire application and users. The system allows organising and analysing video meetings, as well as managing all system settings. In the mobile application, users can create accounts, edit information, see the history of conferences and participate in them.

The web application and mobile app have a pleasant and intuitive design.

The developed mobile app and web application have been fully tested. A comparison with existing large projects is made.

Keywords: *video conferencing, web application, cross-platform, mobile application, authorisation, system, framework.*

ЗМІСТ

ВСТУП.....7
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ.....13
1.1 Аналіз існуючих платформ для проведення відеоконференцій.13
1.2 Аналіз характеристики Zoom 14
1.3 Аналіз характеристики Платформ 16
1.4 Постановка завдання.....18
2 ДОСЛІДЖЕННЯ ТЕХНОЛОГІЙ ТА ЗАСОБІВ РОЗРОБКИ 21
2.1 Огляд мов програмування для веб-застосунку.....21
2.2 Аналіз сучасних фреймворків для створення веб-застосунків 22
2.3 Архітектурні патерни 23
2.4 Управління залежностями.....27
2.5 Аналіз REST API.....29
2.6 Огляд технологій для розробки мобільного застосунку.....31
2.7 Висновок за розділом.....32
3 ПРОЕКТУВАННЯ ТА РОЗРОБКА СИСТЕМИ 35
3.1 Опис предметної області.....35
3.2 Архітектура системи 37
3.3 Фукціональні вимоги системи.....41
3.4 Вимоги до апаратного та програмного забезпечення 46
3.5 Опис модулів та алгоритмів веб-застосунку 49
3.6 Розробка мобільного застосунку 67
3.7 Дизайн інтерфейсу користувача 74
3.8 Тестування та аналіз ефективності.....83
3.9 Висновок за розділом 85
ВИСНОВКИ.....88
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....89

ВСТУП

Актуальність теми

В останні роки засоби організації відеоконференцій широко обговорюються науковою громадськістю з метою підвищення ефективності суспільної комунікації під час проведення наукових заходів. Враховуючи, що традиційні методи нездатні забезпечити досягнення цілей, обумовлених соціальним замовленням суспільства на сучасному етапі його розвитку, даний факт став причиною переходу від традиційної до особистісно-орієнтованої парадигми, стратегічними напрямками якої є інтелектуальний і моральний розвиток людини на основі залучення його в самостійну доцільну діяльність.

Системи управління та проведення відеоконференцій є надзвичайно важливим у сучасному інформаційному середовищі, особливо для організацій, які прагнуть забезпечити ефективну комунікацію, оперативне прийняття рішень та зниження витрат. Також за дослідженням Мельника та Гуменюка, використання відеоконференцій у дистанційному навчанні не лише збільшує рівень залученості студентів, але й дозволяє економити ресурси освітніх закладів [7]. Тому використання систем управлення відеоконференціями на базі веб-платформи з мобільним додатком, розробленим за допомогою фреймворку Flutter, який дозволяє створювати мобільні додатки за допомогою одного коду, з використанням мови програмування Dart, дозволяє забезпечити доступ до відеоспілкування користувачам різних операційних систем, а саме IOS та Android. Це надає максимальну зручність для всіх категорій користувачів. Інтеграція з SDK Jitsi Meet гарантує високу якість відео- та аудіозв’язку у мобільному та вебдодатку. Головна його перевага це відкритий код, тобто Open-Source проект, його можна використовувати і робити персональні модифікації без обмежень.

Веб-застосунок створено з використанням сучасних веб-технологій,

зокрема PHP-фреймворку CodeIgniter та REST API. А адміністративна панель надає широкий функціонал, для керуванням мобільним додатком. CodeIgniter – це дуже зручний та легкий PHP фреймворк для розробки веб-застосунків, він надає необхідний функціонал, але не перевантажує проект зайвими бібліотеками та компонентами, його перевага це швидкість, безпека і мінімалізм.

Загалом, розробка система управління та проведення відеоконференцій, завдяки мові програмування PHP, Dart та потужним фреймворкам з інструментами є актуальною темою. Ця технологія дозволяє здійснювати ефективну комунікацію, не виходячи з офісу або дому, що особливо важливо у сучасному світі з його швидким темпом життя та глобалізацією. Відеоконференцзв’язок є невід’ємною частиною сучасного бізнесу, адже він допомагає компаніям залишатися конкурентоспроможними та ефективними у швидкозмінному середовищі.

Мета дослідження

Метою дослідження є розробка ефективного та функціонального програмного забезпечення, а саме: веб-застосунок та мобільний додаток яке б задовольняло потреби користувачів. Дослідження також включає інтеграцію різних інструментів та сервісів для покращення функціональності та зручності користувачів. Важливий момент, на який звертається увага під час розроблення продукту, це безпека, швидкість роботи та можливість масштабування системи. Завдяки Jitsi Meet та їхній технології об'єднання серверів.

Завдання дослідження

Завдання цього дослідження полягають у детальному аналізі, проєктуванні та реалізації системи управління та проведення відеоконференцзв’язку та мобільного додатку. Для досягнення мети дослідження було сформульовано низку важливих завдань, які забезпечують комплексний підхід до розробки

системи відеоконференцій:

1. Виконати детальний аналіз існуючих платформ відеоконференцзв’язку (Zoom, Microsoft Teams, Google Meet, Cisco Webex, Jitsi Meet) з метою виявлення їх переваг та недоліків.

2. Запроектувати та створити веб-застосунок і мобільний додаток, які відповідають сучасним вимогам щодо безпеки, продуктивності, зручності користування та можливості масштабування.

3. Забезпечити інтеграцію різних технологій, зокрема REST API та Jitsi Meet SDK, для забезпечення високої якості аудіо- та відеозв’язку.

4. Виконати комплексне тестування розроблених додатків з метою їх оптимізації, забезпечення стабільності та швидкості роботи.

5. Розробити документацію для користувачів та розробників, що забезпечить простоту подальшої підтримки та розвитку системи.

Об’єкт дослідження

Об'єктом дослідження є процес створення програмного забезпечення для відеоконференцій, що включає розробку веб-застосунку та мобільного додатку. Особлива увага в дослідженні приділяється вибору оптимальних технологічних рішень, проектуванню архітектури, забезпеченню безпеки і продуктивності, а також зручності та ефективності роботи користувачів з системою.

Предмет дослідження

Предметом дослідження є методи, технології та інструменти, що використовуються для розробки веб-застосунків і мобільних додатків для відеоконференцзв’язку. Зокрема, досліджуються архітектурні підходи, способи інтеграції компонентів, методи забезпечення стабільності та швидкодії системи, а також підходи до тестування і документування програмного продукту.

Методи дослідження

У процесі дослідження були застосовані як теоретичні, так і експериментальні методи:

Теоретичні методи включають:

1. Аналіз та систематизація науково-технічної літератури та документації щодо веб-технологій, архітектури програмних систем та мобільних додатків.
2. Вибір та порівняльний аналіз технологій для розробки додатків та забезпечення їх ефективної інтеграції.
3. Моделювання архітектурних схем взаємодії компонентів веб- та мобільних застосунків.

Експериментальні методи дослідження охоплюють:

1. Реалізація прототипів веб-застосунку та мобільного додатку з подальшим аналізом їх продуктивності та ефективності.
2. Проведення інтеграційного тестування для перевірки сумісності компонентів системи та стабільності взаємодії.
3. Використання інструментів моніторингу продуктивності для виявлення потенційних проблем та оптимізації роботи додатків.

Практичне значення одержаних результатів

Практичне значення дослідження полягає у створенні комплексного програмного рішення для проведення відеоконференцій, яке може бути використане організаціями та компаніями для забезпечення ефективної комунікації між співробітниками та клієнтами. Запропонована архітектура, розроблені алгоритми та технологічні рішення можуть бути використані як типові шаблони при створенні аналогічних проєктів, що дозволить скоротити час на розробку, зменшити витрати та забезпечити високу якість кінцевого продукту.

Глосарій

HTML - HyperText Markup Language – мова розмітки, яка використовується для створення та структурування веб-сторінок шляхом визначення елементів і розташування інформації.

CSS - Cascading Style Sheets – мова стилів, яка використовується для оформлення зовнішнього вигляду веб-сторінок, створених за допомогою HTML.

PHP - Hypertext Preprocessor – серверна скриптова мова, яка використовується для генерації динамічних веб-сторінок та взаємодії з базами даних.

MySQL - вільна реляційна система управління базами даних, яка використовується для зберігання, організації та роботи з інформацією веб-застосунків.

БД (база даних) – організований набір даних, які зберігаються та використовуються програмним забезпеченням для зручності доступу, керування та оновлення інформації.

Flutter - відкритий фреймворк компанії Google для створення кросплатформних мобільних додатків з використанням мови програмування Dart, що підтримує розробку додатків для Android та iOS.

API (Application Programming Interface) - інтерфейс взаємодії між різними програмними додатками, що визначає правила та методи, за допомогою яких вони можуть обмінюватися даними.

JS (JavaScript) – це мова програмування, що використовується для створення інтерактивних та динамічних елементів веб-сторінок, таких як анімації, інтерактивні форми, інтеграція мультимедіа та робота з елементами інтерфейсу.

SDK (Software Development Kit) – набір інструментів та бібліотек, що дозволяють розробникам створювати програмне забезпечення для певної платформи або технології.

Jitsi Meet – відкрита платформа для відеоконференцзв’язку, яка надає SDK та інструменти для інтеграції відео- та аудіозв’язку в веб-застосунки та мобільні додатки.

CodeIgniter – легкий та продуктивний PHP-фреймворк для створення веб-додатків, який забезпечує швидку розробку, мінімалізм у використанні ресурсів та високий рівень безпеки.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз існуючих платформ для проведення відеоконференцій

В умовах сучасного цифрового суспільства відеоконференцзв’язок перетворився на невід’ємну складову комунікації, яка активно використовується у бізнесі, освітніх закладах, державних структурах і навіть у повсякденному спілкуванні. На сьогодні ринок пропонує широкий вибір платформ, серед яких найбільш відомими є Zoom, Microsoft Teams, Google Meet, Cisco Webex та Jitsi Meet.

Zoom є одним із найпопулярніших сервісів, відомий завдяки високій якості відео та простому інтерфейсу. Однак він має ряд обмежень щодо безпеки та конфіденційності, а також певні технічні обмеження у безкоштовній версії.

Microsoft Teams пропонує широкі можливості для корпоративного використання, інтегруючи роботу з документами, календарем та інструментами для командної співпраці. Проте, його недоліком є складний інтерфейс та висока ресурсомісткість, яка не підходить для малопотужних пристроїв.

Google Meet є простим у використанні завдяки інтеграції з екосистемою Google. Проте, його функціонал у безкоштовній версії також обмежений, що створює незручності для малих організацій та приватних користувачів.

Cisco Webex забезпечує високий рівень безпеки і стабільності зв’язку, але його інтерфейс складний для швидкого освоєння, а також платформа має високі вимоги до обладнання, що може бути перешкодою для широкого використання.

Jitsi Meet – платформа з відкритим кодом, яка дозволяє розробникам і спільнотам адаптувати і вдосконалювати функціонал під власні потреби. Це створює можливість забезпечити гнучкість і адаптивність, що особливо важливо для специфічних завдань і нестандартних сценаріїв використання.

Однак переважна більшість цих платформ є комерційними продуктами із

закритим кодом, через що ентузіасти і незалежні розробники не можуть вільно їх модифікувати чи вдосконалювати. Це пояснюється тим, що великим компаніям зручніше утримувати ринок завдяки обмеженій доступності інформації про внутрішню структуру їхніх сервісів.

Таким чином, незважаючи на великий вибір рішень, на ринку відсутнє універсальне рішення, яке б задовольнило усі категорії користувачів одночасно. Комерційні платформи орієнтовані на масові або великі корпоративні середовища і не завжди відповідають індивідуальним вимогам щодо безпеки, швидкодії, кросплатформності та простоти використання.

Враховуючи наведене вище, розробка власного програмного продукту, який би об'єднав у собі переваги популярних платформ і водночас усунув би їх недоліки, є актуальним та важливим завданням. Це дозволить створити більш адаптоване, безпечне і зручне рішення, доступне широкому колу користувачів, з урахуванням специфічних вимог конкретних організацій і завдань.

1.2 Аналіз характеристики Zoom

Zoom є однією з найбільш популярних платформ для організації відеоконференцій, яка отримала значний розвиток і популярність у період глобальних карантинних обмежень. Висока активність реклами та майже повна відсутність альтернативних продуктів на ринку в той період стали головними чинниками стрімкого зростання кількості користувачів Zoom. Однак згодом компанія почала активно вводити різноманітні ліміти у безкоштовній версії платформи, стимулюючи користувачів до переходу на платні тарифні плани.

Підвищення популярності платформи призвело до суттєвого збільшення навантаження на сервери, що стало причиною частих технічних збоїв та періодичних розривів зв’язку, що негативно впливає на досвід користувачів. Крім того, чимало користувачів відзначають складність та перевантаженість

інтерфейсу Zoom, що вимагає значних витрат часу на ознайомлення та адаптацію, особливо для нових користувачів.

Окремою критичною проблемою платформи Zoom є питання кібербезпеки. Зафіксовано багато випадків, коли хакери здійснювали атаки з використанням фейкових доменів компанії, через які поширювалися шкідливі програми та викрадалася особиста інформація користувачів. Внаслідок цих атак Google у травні 2020 року навіть заборонила своїм співробітникам використовувати Zoom для проведення службових нарад.

Також існують документовані випадки несанкціонованого підключення до приватних конференцій, що призводило до витоку відеозаписів, які потім публікувалися на відкритих відеосервісах, таких як YouTube та Vimeo. Такі інциденти були висвітлені у виданнях, зокрема, у The Washington Post, що посилило недовіру користувачів до Zoom.

Незважаючи на ці проблеми, платформа Zoom має кілька вагомих переваг, завдяки яким вона залишається одним з лідерів на ринку відеоконференцзв’язку:

Переваги:

стабільне з'єднання навіть за умови низької швидкості інтернету;

зручний функціонал для проведення масових заходів (запис, чат, демонстрація екрану, кімнати очікування);

широкі можливості для інтеграції зі сторонніми додатками та сервісами.

Недоліки:

обмежений час безкоштовних зустрічей (до 40 хвилин для групових конференцій);

значні питання безпеки та конфіденційності;

високе споживання системних ресурсів, що може бути критичним для малопотужних пристроїв.

Отже, аналіз платформи Zoom дозволяє зробити висновок, що для створення власного продукту необхідно враховувати її сильні сторони, водночас

приділяючи особливу увагу питанням безпеки, оптимізації системних ресурсів та зручності користувацького інтерфейсу.

1.3 Аналіз характеристики Платформ

Microsoft Teams

Microsoft Teams – це одна з провідних платформ для проведення відеоконференцій, яка тісно інтегрована з екосистемою Microsoft 365, що робить її ідеальною для корпоративного сегменту. Завдяки глибокій інтеграції з іншими продуктами компанії (Outlook, SharePoint, OneDrive), користувачі отримують зручні можливості для спільної роботи з документами, організації зустрічей та контролю прав доступу.

Переваги:

1. глибока інтеграція з Microsoft 365;
2. підходить для великих компаній і корпоративних потреб;
3. потужна система управління доступом і безпекою.

Недоліки:

1. складний і неінтуїтивний інтерфейс для нових користувачів;
2. менша гнучкість та адаптивність у порівнянні з альтернативами (Zoom, Google Meet).

Google Meet

Google Meet є популярним рішенням, що пропонує простоту використання завдяки інтуїтивному інтерфейсу та інтеграції з екосистемою Google Workspace. Важливою особливістю є те, що платформа працює без необхідності встановлення додаткового програмного забезпечення, що значно спрощує доступ.

Переваги:

1. простий і зручний інтерфейс;
2. інтеграція з іншими сервісами Google (Google Drive, Calendar);

3. високий рівень безпеки і шифрування.

Недоліки:

1. обмежений набір функцій персоналізації;

2. відсутність розширених функцій управління зустрічами (наприклад, кімнати для сесій або деталізоване управління).

Cisco Webex

Cisco Webex орієнтована насамперед на корпоративних клієнтів та великі організації, пропонуючи розширені функції безпеки, аналітики та управління. Платформа відома своєю високою продуктивністю та безпекою.

Переваги:

1. потужні інструменти для великих корпоративних структур;

2. високий рівень шифрування і безпеки;

3. аналітичні інструменти для моніторингу та оцінки ефективності зустрічей.

Недоліки:

1. складна процедура реєстрації та початкового налаштування;

2. високі вимоги до апаратного забезпечення та швидкості інтернету.

Підсумовуючи проведений аналіз, можна зазначити, що, незважаючи на широкий спектр можливостей, кожна з наведених платформ має суттєві обмеження, такі як високі системні вимоги, складний інтерфейс або відсутність гнучкості у безкоштовних версіях. Саме тому аналізуючи переваги та недоліки сучасних платформ відеоконференцій, які детально описані Бодруком О.І., було визначено необхідність створення більш безпечної та зручної системи [8], яка лягла в основу концепції розробки власної системи відеоконференцій, яка поєднувала б найкращі якості зазначених платформ, є обґрунтованою та перспективною. Основними критеріями такої системи можуть бути:

1. простий і зрозумілий користувацький інтерфейс;

2. низькі системні вимоги;

3. повна кросплатформність (Android, iOS, Web);

4. посилена безпека та захист даних;

5. гнучка адаптація під специфічні завдання (освітні, державні, бізнесові потреби).

Розробка такої платформи забезпечить конкурентні переваги на ринку, займаючи нішу між доступними, але обмеженими рішеннями та складними корпоративними сервісами.

1.4 Постановка завдання

Постановка завдання включає визначення основних цілей та етапів розробки системи управління відеоконференціями Confiero, що складається з веб-застосунку та мобільного додатку. Кожен етап направлений на створення високоякісного, функціонального, зручного та надійного продукту.

Основні завдання, які необхідно вирішити у процесі реалізації проекту, полягають у наступному:

1. Аналіз вимог до системи:

- визначити функціональні вимоги (можливість створення, приєднання до конференцій, управління користувачами та ролями, запис зустрічей);
- визначити нефункціональні вимоги (швидкодія, масштабованість, безпека та зручність інтерфейсу);
- дослідити цільову аудиторію, її потреби, сценарії використання веб- та мобільного застосунків;
- визначити процеси взаємодії користувачів з додатком.

2. Проєктування архітектури системи:

- розробити масштабовану архітектуру з використанням фреймворків PHP (CodeIgniter) для веб-частини та Flutter (Dart) для мобільного додатку;

- спроектувати структуру бази даних (MySQL) із оптимізованими зв’язками між елементами;
- створити інтуїтивний та адаптивний інтерфейс для зручної навігації у веб-застосунку та мобільному додатку.

3. Реалізація серверної частини (бекенд):

- розробити веб-застосунок за допомогою PHP-фреймворку CodeIgniter;
- інтегрувати REST API для взаємодії між веб-додатком та мобільним застосунком;
- реалізувати адміністративну панель для ефективного управління користувачами, налаштуваннями та відеоконференціями.

4. Реалізація клієнтської частини (фронтенд):

- створити веб-інтерфейс за допомогою HTML, CSS, JavaScript та додаткових технологій для динамічного оновлення;
- розробити мобільний додаток на базі Flutter для забезпечення роботи на платформах Android та iOS;
- реалізувати основні функціональні компоненти, такі як приєднання до конференцій, чат, запис відео та аудіо, демонстрація екрану.

5. Тестування та оптимізація додатків:

- провести комплексне тестування системи для виявлення та усунення недоліків;
- виконати стрес-тестування для перевірки стабільності роботи системи під навантаженням;
- оптимізувати продуктивність веб-застосунку та мобільного додатку для забезпечення стабільної роботи на різних пристроях і умовах використання.

Виконання зазначених завдань дозволить створити конкурентоспроможний, надійний та безпечний програмний продукт, який

відповідатиме сучасним вимогам користувачів до систем управління та проведення відеоконференцій.

2 ДОСЛІДЖЕННЯ ТЕХНОЛОГІЙ ТА ЗАСОБІВ РОЗРОБКИ

2.1 Огляд мов програмування для веб-застосунку

Вибір правильної мови програмування має критичне значення для успішної реалізації веб-застосунку системи управління та проведення відеоконференцій Confiero. Веб-застосунок повинен бути зручним, ефективним, безпечним та здатним легко інтегруватися з іншими сервісами. Для реалізації проєкту Confiero було проведено аналіз і вибір мов програмування, які найкраще відповідають цим критеріям.

PHP (Hypertext Preprocessor) – це одна з найпопулярніших серверних мов, яка широко застосовується для створення динамічних веб-додатків. PHP володіє значною спільнотою розробників, відкритим вихідним кодом та великою кількістю доступних фреймворків і бібліотек. В контексті розробки Confiero, PHP вибрано через його стабільність, простоту інтеграції з базами даних MySQL, а також можливість швидкого налаштування і підтримки REST API. Фреймворк CodeIgniter був обраний завдяки своїй мінімалістичності, високій продуктивності і зрозумілій структурі.

JavaScript – це мова програмування, яка відповідає за динамічні аспекти фронтенду веб-застосунку. Використання JavaScript дозволяє створювати інтуїтивно зрозумілі інтерфейси та забезпечувати інтерактивність без перезавантаження сторінок, що суттєво покращує користувацький досвід. Завдяки великій кількості доступних бібліотек та інструментів (наприклад, React, Angular, Vue.js), JavaScript став незамінним при розробці веб-інтерфейсів сучасних застосунків.

HTML та CSS також є важливими технологіями у створенні інтерфейсу користувача. HTML використовується для створення базової структури веб-сторінок, тоді як CSS відповідає за їх зовнішній вигляд, адаптивність та

відображення на різних пристроях.

Високий рівень сумісності, широка підтримка з боку спільноти та ефективна інтеграція стали основними причинами вибору зазначених мов програмування для розробки веб-застосунку Confiero. Таким чином, PHP разом з CodeIgniter були використані для створення надійної серверної частини системи, а HTML, CSS і JavaScript забезпечили якісний та інтерактивний інтерфейс.

2.2 Аналіз сучасних фреймворків для веб-застосункі

Розробка веб-застосунків потребує застосування сучасних технологій, які дозволяють створити продукт із високою продуктивністю, простотою підтримки та можливістю масштабування. Вибір відповідного фреймворку є важливим рішенням, що впливає на швидкість розробки, якість коду, а також легкість майбутньої підтримки. В контексті розробки системи відеоконференцій Confiero було проведено аналіз декількох сучасних веб-фреймворків, що активно використовуються у сфері веб-розробки.

Laravel є одним з найпопулярніших PHP-фреймворків, що надає широкий функціонал для швидкої розробки складних веб-застосунків. Серед переваг Laravel: потужна система маршрутизації, простота інтеграції з базами даних, велика кількість готових компонентів та інструментів. Однак його головним недоліком є значна ресурсомісткість і складність для простих та швидких рішень.

Symfony також є популярним PHP-фреймворком, який забезпечує високий рівень гнучкості та модульності. Він пропонує розширену архітектуру компонентів, що дозволяє детально налаштовувати кожну частину програми. Але через свою складність та детальну конфігурацію цей фреймворк не завжди оптимальний для проєктів з обмеженими ресурсами та часом на розробку.

CodeIgniter був обраний для реалізації веб-застосунку Confiero через низку своїх специфічних переваг [1]. Він характеризується мінімалізмом, простотою у

використанні, невеликим розміром та низьким рівнем споживання ресурсів. Фреймворк забезпечує необхідні функції для розробки REST API та інтеграції з різноманітними сервісами, що ідеально відповідає задачам системи Confiero. Також важливою перевагою CodeIgniter є чітка і зрозуміла структура, що значно полегшує підтримку та розвиток застосунку у майбутньому.

Vue.js, React.js та Angular – це популярні JavaScript-фреймворки для розробки фронтенду веб-застосунків. React забезпечує швидкий рендеринг і високу продуктивність, але потребує значного досвіду роботи з екосистемою. Angular пропонує широкий набір вбудованих інструментів та чітку структуру додатка, але має відносно високий поріг входження та значні системні вимоги. Vue.js, натомість, надає оптимальний баланс між простотою і продуктивністю, забезпечуючи легкий старт та гнучкість у налаштуванні.

Для проєкту Confiero оптимальним вибором для фронтенд-частини стало поєднання JavaScript з традиційними технологіями HTML і CSS, що забезпечує високу продуктивність, простоту розробки і підтримки, а також ефективну інтеграцію з бекендом, побудованим на CodeIgniter, бо питання кібербезпеки особливо гостро стоять у системах відеоконференцій, як зазначають Tan та Lee у своєму дослідженні, що вимагало додаткових заходів безпеки в розробці [4].

Таким чином, аналіз сучасних фреймворків показав, що найкращими для задач і умов проєкту Confiero є:

- CodeIgniter для серверної частини;
- JavaScript у комбінації з HTML і CSS для створення інтерактивного фронтенду.

2.3 Архітектурні патерни

Архітектурні патерни є важливою складовою при проектуванні веб-застосунків, оскільки вони визначають структуру взаємодії різних компонентів

системи, забезпечують зручність підтримки, розширення та тестування. У процесі розробки веб-застосунку системи управління та проведення відеоконференцій Confiero було обрано та реалізовано патерн **MVC (Model-View-Controller)**.

MVC (Model-View-Controller) є одним з найбільш поширених архітектурних патернів у веб-розробці. Він базується на поділі застосунку на три основні компоненти (див. Рис. 1):

- **Model (Модель)** – відповідає за роботу з даними, логіку їх обробки та зберігання в базі даних (в даному випадку – MySQL). Модель забезпечує коректну взаємодію з інформацією та реалізує бізнес-логіку програми.
- **View (Представлення)** – відповідає за відображення інтерфейсу користувача. Це компоненти, які формують зовнішній вигляд сторінок та взаємодіють з користувачем через веб-інтерфейс.
- **Controller (Контролер)** – відповідає за управління потоками даних між моделями та представленнями. Контролер обробляє запити користувачів, викликає методи моделі та формує відповідь у вигляді представлення.



Рисунок 1 — *Схема архітектури MVC*

Фреймворк CodeIgniter, використаний у проекті Confiero, чітко реалізує

патерн MVC. Він надає просту, прозору та зрозумілу структуру, що дозволяє розробникам легко розмежовувати різні компоненти програми, спрощує процес розробки, тестування та підтримки коду.

Конкретна реалізація MVC у Confiero передбачає наступне:

- Моделі розташовані у директорії application/models та відповідають за логіку роботи з базою даних та бізнес-процеси, наприклад, управління користувачами та конференціями.
- Представлення знаходяться у директорії application/views, і вони формують сторінки та компоненти, які бачить користувач. Це, наприклад, сторінки авторизації, панелі адміністрування, форма редагування профілю.
- Контролери, що розміщені у директорії application/controllers, керують логікою обробки запитів користувачів, взаємодією з моделями та відображенням відповідних представлень. Наприклад, обробка запитів на створення нових конференцій, авторизація користувачів та керування налаштуваннями.

Іншими поширеними архітектурними патернами у веб-розробці є:

1. MVVM (Model-View-ViewModel):

MVVM – патерн, що розширює MVC і надає додатковий компонент ViewModel. ViewModel відповідає за обробку логіки представлення та зв’язок між моделлю і представленням. Він часто використовується в SPA (Single Page Applications), наприклад, з фреймворками React або Vue.js, де необхідно забезпечити інтерактивність та швидке реагування на зміни стану застосунку (див. Рис. 2).

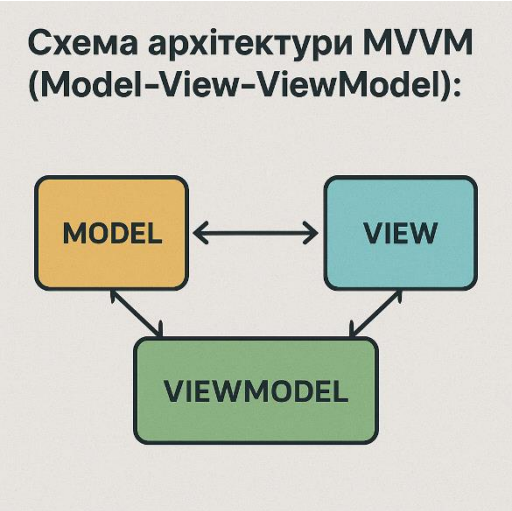


Рисунок 2 — *Схема архітектури MVVM*

2. MVP (Model-View-Presenter):

Патерн MVP має подібну до MVC структуру, проте Presenter бере на себе всю логіку обробки взаємодії з користувачем, що спрощує тестування інтерфейсу. Цей патерн широко застосовується в застосунках, де чітко відокремлена логіка інтерфейсу та бізнес-логіка, наприклад, корпоративні додатки (див. Рис. 3).

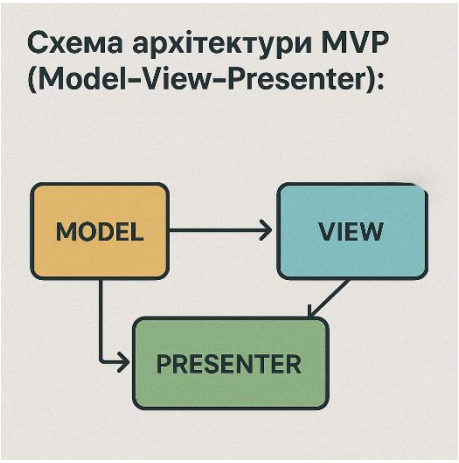


Рисунок 3 — *Схема архітектури MVP*

3. REST (Representational State Transfer):

REST – це архітектурний стиль, що використовується для розробки веб-сервісів та API. Він базується на бездержавній взаємодії між клієнтом та сервером, що робить його ідеальним для розподілених систем та інтеграції різних компонентів, зокрема у проєкті Confiero для взаємодії веб-застосунку та мобільного додатку (див. Рис. 4).



Рисунок 4 — *Схема архітектури REST*

Перевагами вибору архітектурного патерну MVC для веб-застосунку Confiero є:

- чітке розмежування логіки, представлення та управління;
- спрощення тестування та підтримки додатку;
- легкість масштабування та розширення функціоналу.

Таким чином, завдяки використанню патерну MVC було забезпечено структурованість коду, ефективне управління даними та можливість швидкого розвитку проекту Confiero у майбутньому.

2.4 Управління залежностями

Управління залежностями є важливим аспектом розробки сучасних веб-застосунків. Воно забезпечує чітку організацію зовнішніх бібліотек і модулів,

контроль версій, а також спрощує процес підтримки та оновлення компонентів системи.

У процесі розробки веб-застосунку Confiero було вибрано ручний підхід до управління залежностями, що визначалося специфікою проекту та вимогами до мінімалізму і швидкості роботи. Це дозволило зменшити кількість непотрібних залежностей, спростити структуру проєкту та полегшити його підтримку і оновлення.

Основними залежностями веб-застосунку Confiero є фреймворк CodeIgniter та бібліотеки для реалізації REST API та роботи з базами даних MySQL. CodeIgniter обраний за його легкість, мінімалізм і простоту у використанні, що дозволяє ефективно керувати компонентами системи вручну.

Зазвичай сучасні веб-додатки використовують пакетні менеджери, такі як Composer для PHP, npm для JavaScript або Yarn, які значно спрощують керування залежностями через автоматичну установку та оновлення пакетів. Проте, вибір ручного управління у випадку Confiero був обумовлений прагненням уникнути зайвих залежностей і контролювати кожен компонент окремо, що забезпечує додаткову стабільність і прозорість системи.

Щодо фронтенду веб-застосунку Confiero, залежності (CSS, JavaScript бібліотеки та інші ресурси) також були включені вручну. Такий підхід забезпечує повний контроль над версіями і станом компонентів, що гарантує стабільність роботи додатку та уникнення конфліктів версій або несумісності бібліотек.

Системи керування базами даних (СКБД)

Для зберігання та обробки даних у веб-застосунку Confiero використовується MySQL – одна з найпоширеніших систем керування базами даних (СКБД). Вибір MySQL був зроблений через його високу продуктивність, надійність, широку підтримку спільноти, легкість налаштування та інтеграції з PHP. MySQL дозволяє легко створювати складні запити до бази даних, забезпечує транзакційну підтримку та високий рівень безпеки.

Підхід до управління базою даних у проєкті Confiero базується на моделі ORM (Object-Relational Mapping), яка реалізована в рамках фреймворку CodeIgniter. Цей підхід дозволяє розробникам взаємодіяти з даними в базі за допомогою об’єктів та методів мови програмування PHP, а не за допомогою прямих SQL-запитів, що значно спрощує код, підвищує його зрозумілість, а також зменшує ризики помилок і покращує підтримку проєкту в майбутньому.

Таким чином, управління залежностями у проекті Confiero реалізоване у вигляді ручного підходу, що дозволяє досягти максимальної прозорості і простоти підтримки, уникнути непотрібних компонентів, забезпечити ефективну роботу з базою даних та високу загальну ефективність системи.

2.5 Аналіз REST API

REST (Representational State Transfer) є сучасним архітектурним стилем, що широко використовується для створення API (інтерфейсів програмування додатків). REST API забезпечує ефективну комунікацію між клієнтською і серверною частинами додатку, дозволяючи їм взаємодіяти за допомогою стандартизованих HTTP-запитів та відповідей.

У веб-системі управління та проведення відеоконференцій Confiero REST API відіграє важливу роль, забезпечуючи ефективну інтеграцію та комунікацію між серверною частиною (бекендом), яка реалізована на фреймворку CodeIgniter, та мобільним додатком, що розроблений за допомогою Flutter.

Основні принципи REST, що реалізовані у системі Confiero:

1. *Безстанова взаємодія (Stateless):* Кожен HTTP-запит містить всю необхідну інформацію для його обробки сервером без збереження стану між запитами. Це дозволяє забезпечити високий рівень масштабованості та надійності системи.

2. *Єдині інтерфейси (Uniform Interface):* Використовуються стандартні

методи HTTP (GET, POST, PUT, DELETE) для взаємодії з ресурсами (наприклад, створення, отримання, оновлення, видалення даних про конференції, користувачів тощо).

3. *Чітке визначення ресурсів***:** Кожен об'єкт у системі (наприклад, користувач, конференція, повідомлення) визначається як окремий ресурс з унікальним ідентифікатором (URL).

4. *Стандартизовані формати даних***:** Взаємодія між сервером та клієнтом здійснюється з використанням широко поширених форматів, таких як JSON, що спрощує процес інтеграції та підтримки API.

Функціональні можливості REST API Confiero включають:

- *Аутентифікація та авторизація користувачів*: REST API забезпечує реєстрацію користувачів, авторизацію та контроль прав доступу до певних функцій системи.
- *Керування конференціями*: створення, редагування, видалення конференцій, додавання учасників, отримання списку конференцій.
- *Керування користувачами*: перегляд, додавання, редагування та видалення профілів користувачів.
- *Інтеграція з мобільним додатком*: мобільний додаток Confiero отримує всю необхідну інформацію саме через REST API, забезпечуючи взаємодію з сервером у реальному часі.

Перевагами використання REST API у проєкті Confiero є:

- простота використання та інтеграції з різними платформами (веб, Android, iOS);
- легкість у масштабуванні системи завдяки безстановій природі REST;
- висока ефективність та швидкість роботи додатку завдяки мінімізації накладних витрат на комунікацію між клієнтом та сервером.

Отже, REST API є важливою складовою частиною веб-системи Confiero, бо

REST API застосунку було спроектовано відповідно до рекомендацій RESTful API Design [6], забезпечуючи надійну та ефективну взаємодію між серверною та клієнтською частинами системи, а також підтримку високої продуктивності та масштабованості додатку.

2.6 Огляд технологій для розробки мобільного застосунку

При створенні мобільних додатків важливим етапом є вибір технологій, які забезпечать високу продуктивність, гнучкість, надійність і простоту підтримки. У рамках проєкту Confiero, що передбачає розробку мобільного додатку для відеоконференцій з інтеграцією з веб-системою, проведено аналіз сучасних технологій, які використовуються для мобільної розробки.

Основними підходами у розробці мобільних застосунків є:

1. Нативна розробка (Swift/Objective-C для iOS, Java/Kotlin для Android):

- Переваги: висока продуктивність, глибока інтеграція з платформою, доступ до всіх функцій пристрою.
- Недоліки: високі витрати часу і ресурсів через необхідність розробки окремих версій для різних платформ.

2. Гібридна розробка (наприклад, Apache Cordova, Ionic):

- Переваги: одна кодова база для всіх платформ, швидкість розробки.
- Недоліки: нижча продуктивність у порівнянні з нативними додатками, обмежений доступ до платформових функцій.

3. Кросплатформна розробка (наприклад, React Native, Flutter):

- Переваги: поєднання переваг гібридної і нативної розробки, висока продуктивність, єдина кодова база.
- Недоліки: залежність від третіх сторін (спільноти, фреймворків), потенційні проблеми сумісності з платформовими оновленнями.

Для розробки мобільного застосунку Confiero обрано технологію *Flutter*, яка є сучасним кросплатформним рішенням, створеним компанією Google. Flutter використовує мову програмування **Dart**, що має зручний синтаксис, високу продуктивність і ефективну підтримку.

Переваги Flutter у контексті проекту Confiero:

- *Єдина кодова база* для Android та iOS, що значно скорочує час і витрати на розробку і підтримку додатку.
- *Висока продуктивність* завдяки компіляції Dart у рідний код платформ.
- *Гнучка та сучасна система віджетів*, яка дозволяє створювати красиві і адаптивні інтерфейси, що відповідають сучасним вимогам UX/UI.
- *Ефективна інтеграція з REST API*, що забезпечує плавну і стабільну взаємодію з серверною частиною веб-системи Confiero.

Таким чином, вибір Flutter для мобільного застосунку Confiero дозволив створити додаток, що відповідає сучасним вимогам до продуктивності, зручності користування, швидкості розробки і економії ресурсів [2]. Це технологічне рішення забезпечує довгострокову підтримку і можливість масштабування додатку в майбутньому.

2.7 Висновок за розділом

Для У результаті проведення детального дослідження технологій та засобів розробки для веб- та мобільного застосунку системи управління та проведення відеоконференцій Confiero було зроблено ряд важливих висновків.

Під час аналізу мов програмування було визначено, що для реалізації серверної частини веб-застосунку оптимальним рішенням є використання мови PHP. Ця мова забезпечує легкість розробки, високу швидкість роботи, інтеграцію з базами даних та підтримку REST API, що необхідно для ефективного

функціонування системи Confiero. Для створення інтерактивного фронтенду було обрано JavaScript у поєднанні з HTML і CSS, які дозволяють створювати зручні, адаптивні та інтуїтивні інтерфейси.

При аналізі сучасних фреймворків для веб-застосунків обрано PHP-фреймворк CodeIgniter. Головними перевагами цього фреймворку є його мінімалізм, простота, швидкість розробки, низькі системні вимоги та ефективна підтримка REST API. Це дозволило створити надійну, легко підтримувану та швидку у розробці серверну частину веб-додатку.

Використання архітектурного патерну MVC (Model-View-Controller), який реалізується фреймворком CodeIgniter, забезпечило чітке розмежування відповідальності між моделями, представленнями та контролерами. Це спростило процес розробки, тестування та подальшої підтримки системи Confiero.

Ручний підхід до управління залежностями було обрано для зменшення зайвих компонентів, прозорості структури додатку та полегшення процесу його підтримки. Використання системи керування базами даних MySQL з реалізацією підходу ORM забезпечило ефективне керування даними, легкість взаємодії з базою даних та покращення загальної продуктивності веб-застосунку.

Аналіз REST API підтвердив важливість цієї технології у проекті, забезпечуючи ефективну комунікацію між веб-застосунком та мобільним додатком, стабільність, гнучкість і високу продуктивність додатку завдяки використанню стандартизованих HTTP-запитів та JSON-формату.

Нарешті, дослідження технологій для мобільної розробки показало, що найкращим вибором для проекту Confiero є технологія Flutter із використанням мови Dart. Це рішення дозволяє забезпечити кросплатформність, високу продуктивність, єдину кодову базу для різних платформ, а також ефективну інтеграцію з серверною частиною через REST API.

Таким чином, обрані технології та підходи дозволили створити потужну,

надійну, ефективну та легко підтримувану систему для управління та проведення відеоконференцій Confiero, що відповідає сучасним вимогам користувачів та забезпечує високий рівень якості кінцевого продукту.

3 ПРОЕКТУВАННЯ ТА РОЗРОБКА СИСТЕМИ

3.1 Опис предметної області системи

Предметною областю даної роботи є процес створення та організації ефективного відеоконференцзв'язку для користувачів через спеціалізований веб-застосунок та мобільний додаток. Система розроблена під назвою Confiero, головна мета якої полягає в забезпеченні комфортної та надійної відеокомунікації між учасниками в онлайн-режимі.

Потреба у розробці подібного рішення продиктована активним переходом бізнесу, освітніх закладів, державних організацій та приватних користувачів до використання цифрових інструментів у щоденній комунікації. Як зазначає Скрипник І.В., ефективне впровадження систем відеоконференцій у навчальні заклади дозволяє значно покращити якість комунікації між викладачами та студентами, що було враховано при розробці системи для освітнього сегмента [6]. В умовах сучасного цифрового середовища надзвичайно важливо забезпечити не лише базовий відеозв’язок, а й ефективне управління зустрічами, захист персональних даних, а також доступність сервісу для різних платформ — веб-браузерів, Android та iOS мобільних пристроїв.

Система Confiero включає в себе дві основні компоненти:

1. Веб-застосунок Confiero

Даний компонент виступає ключовим елементом системи. Він побудований з використанням мови програмування PHP та фреймворку *CodeIgniter*, що дозволяє зручно організувати внутрішню структуру за допомогою MVC (Model-View-Controller) патерну. Веб-застосунок має такі функціональні можливості:

- авторизація та реєстрація користувачів;
- створення, управління та модерація відеоконференцій;

- інтеграція з платформою Jitsi Meet для забезпечення відеозв'язку;
- адміністрування користувачів та управління правами доступу;
- аналітика використання сервісу (статистика конференцій, активність учасників).

Основні файли, які відповідають за базову маршрутизацію та запуск застосунку:

- Головний файл ініціалізації: index.php
- Налаштування застосунку знаходяться в директорії: application/config/ (config.php, database.php, routes.php тощо)
- Контролери застосунку знаходяться в директорії: application/controllers/
- Моделі даних знаходяться в директорії: application/models/.

2. Мобільний застосунок Confiero

Мобільний додаток створений з використанням фреймворку Flutter, що дозволяє ефективно реалізувати один код для підтримки кількох платформ (iOS, Android). Завдяки цьому користувачі отримують єдиний зручний інтерфейс на всіх пристроях. Основний функціонал мобільного застосунку включає:

- створення та приєднання до конференцій;
- персональний кабінет користувача;
- інтеграцію Jitsi Meet SDK для високоякісного відео- та аудіозв'язку;
- перегляд історії конференцій.

Обґрунтування вибору технологій

Використання веб-фреймворку CodeIgniter виправдане завдяки його легкості, простоті розробки та високій швидкодії, що особливо важливо для забезпечення стабільної роботи великої кількості одночасних користувачів.

Flutter обрано для розробки мобільного додатку завдяки його

універсальності та високій продуктивності, що значно скорочує час і витрати на розробку порівняно з окремими нативними рішеннями.

Інтеграція з платформою Jitsi Meet

Jitsi Meet обраний через його відкритий код, гнучкість налаштувань і високу безпеку передачі даних. Інтеграція реалізована через REST API та спеціалізовані SDK, що дозволяє системі Confiero пропонувати користувачам надійне рішення, яке легко масштабувати та налаштовувати під специфічні вимоги.

Таким чином, система Confiero відповідає сучасним стандартам функціональності, безпеки та продуктивності й вирішує завдання, які ставляться перед сучасними системами відеоконференцзв'язку.

3.2 Архітектурні рішення та структурна побудова системи

Для реалізації проєкту Confiero було обрано класичний архітектурний патерн MVC (Model-View-Controller). Вибір цього підходу зумовлений необхідністю забезпечити зручну розробку, легке масштабування та ефективне розділення коду на окремі функціональні блоки. Це значно спрощує супровід, тестування та подальшу модернізацію системи.

Архітектура веб-застосунку Confiero базується на трьох основних компонентах:

1. **Моделі (Model)** — відповідають за управління логікою роботи з даними, взаємодіють з базою даних.

2. **Види (View)** — відповідають за візуалізацію інформації, яка надходить користувачеві.

3. **Контролери (Controller)** — відповідають за логіку додатку, координують взаємодію між моделями та видами.

Основна структура файлової системи проєкту виглядає наступним чином:

Контролери розташовані у директорії: application/controllers/.

Ключові контролери та їх призначення:

- AuthController.php – керування авторизацією та реєстрацією.
- DashboardController.php – відповідає за логіку головної панелі користувача.
- MeetingRoomController.php – обробляє функціонал створення та управління відеоконференціями (див. Лістинг 1).
- ApiConfiero_v1.php – реалізація REST API для взаємодії мобільного додатку з сервером.

Лістинг *1 Приклад фрагменту коду контролера (MeetingRoomController.php)*

```
    defined('BASEPATH') OR exit('No direct script access allowed');
    class MeetingRoomController extends CI_Controller {
        public function __construct(){
            parent::__construct();
            $this->load->model('ApiConfiero_v1_model');
        }
        public function create_meeting(){
            $data = $this->input->post();
            $meeting_id = $this->ApiConfiero_v1_model->createMeeting($data);

            if ($meeting_id) {
                $this->session->set_flashdata('success', 'Конференція створена успішно.');
                redirect('dashboard/meetings');
            } else {
                $this->session->set_flashdata('error',
```

```
'Помилка створення конференції.');
                redirect('dashboard/new_meeting');
            }
        }
    }
```

Моделі реалізують логіку доступу до бази даних, яка знаходиться в директорії: application/models/. Ключові моделі системи Confiero:

ApiConfiero_v1_model.php – основна логіка для API запитів (див. Лістинг 2).

MailService_model.php – сервіс для відправки електронних повідомлень.

AlertManager_model.php – управління нотифікаціями та повідомленнями.

Лістинг *2 Приклад методу моделі (ApiConfiero_v1_model.php):*

```
class ApiConfiero_v1_model extends CI_Model {
    public function createMeeting($data){
        $this->db->insert('meetings', $data);
        return $this->db->insert_id();
    }
    public function getMeetingById($meeting_id){
        return $this->db->get_where('meetings', ['id' =>
$meeting_id])->row_array();
    }
}
```

Представлення відповідають за відображення даних користувачу. Вони зберігаються у директорії: application/views/. Типова структура директорії представлень:

- admin/ – адміністративна панель системи.
- login.php – сторінка авторизації (див. Лістинг 3).
- togo_meeting.php – сторінка підключення до конференції.

Лістинг *3 Приклад простого представлення (login.php):*

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Вхід у Confiero</title>
</head>
<body>
    <form action="<?= site_url('authcontroller/login') ?>"
method="post">
        <input type="email" name="email"
placeholder="Email">
        <input type="password" name="password"
placeholder="Пароль">
        <button type="submit">Увійти</button>
    </form>
</body>
</html>
```

Для реалізації високоякісного відеозв'язку система Confiero інтегрована з Jitsi Meet. Веб-застосунок взаємодіє з Jitsi через REST API (див. Лістинг 4), а мобільний застосунок використовує Jitsi Meet SDK [3].

Лістинг *4 Типовий виклик API створення кімнати (приклад реалізації у PHP):*

```
public function createJitsiRoom($room_name){
    $url = "https://meet.jit.si/{$room_name}";
    return $url; // Повертаємо URL-адресу для підключення
до кімнати
}
```

Мобільний застосунок Confiero використовує API-ендпоінти веб-застосунку для отримання даних (див. Лістинг 5): GET /apiConfiero_v1/getMeeting/{meeting_id}.

Лістинг *5 Приклад JSON відповіді від API:*

```
{
    "status": "success",
    "meeting": {
        "id": "1234",
        "name": "Нарада проекту Confiero",
        "created_at": "2024-06-01T10:00:00",
        "url": "https://meet.jit.si/ConfieroMeeting1234"
    }
}
```

Отже, Архітектурні рішення в системі Confiero орієнтовані на чітке розмежування відповідальності між компонентами, що забезпечує легкість в підтримці, масштабуванні та розвитку додатку. Обраний MVC патерн підтвердив свою ефективність для реалізації складних веб-додатків, а використання відкритих технологій дозволяє легко адаптувати систему до нових потреб та технологічних вимог.

3.3 Фукціональні вимоги системи

Функціональні вимоги описують набір необхідних можливостей та функцій, які повинна реалізовувати система Confiero для виконання поставлених завдань. Вимоги поділяються на загальні (основні) функції веб-застосунку та специфічні функції мобільного додатку.

Загальні функціональні вимоги системи

1. Реєстрація та авторизація користувачів

- Реєстрація: користувач має можливість створити обліковий запис, використовуючи електронну пошту.
- Авторизація: система забезпечує доступ користувачів після успішної перевірки даних входу (логін, пароль), (див. Лістинг 6).

Лістинг *6* Приклад коду авторизації користувачів (файл: AuthController.php):

```
    public function login(){
        $email = $this->input->post('email');
        $password = $this->input->post('password');

        $user = $this->Auth_model->authenticate($email, $password);

        if($user){
            $this->session->set_userdata('user_id', $user->id);
            redirect('dashboard');
        } else {
            $this->session->set_flashdata('error', 'Невірні дані для входу.');
            redirect('auth/login');
        }
    }
```

2. Створення та управління конференціями

- *Створення конференції*: користувач може створювати відеоконференції з унікальним посиланням (див. Лістинг 7).
- *Редагування та видалення конференцій*: користувач має змогу змінювати або видаляти створені ним конференції.

Лістинг *7 Приклад створення конференції (файл: MeetingRoomController.php):*

```
public function create_meeting(){
    $data = [
        'name' => $this->input->post('name'),
        'date' => $this->input->post('date'),
        'time' => $this->input->post('time'),
        'host_user_id'          =>          $this->session-
>userdata('user_id'),
        'link' => $this->Meeting_model->generateLink(),
    ];

    $meeting_id = $this->Meeting_model->create($data);

    if ($meeting_id) {
        redirect('dashboard/meetings');
    } else {
        $this->session->set_flashdata('error',    'Помилка
створення конференції.');
        redirect('dashboard/new_meeting');
    }
}
```

3. Інтеграція з платформою Jitsi Meet

Підключення до конференції: здійснюється через створення посилання для Jitsi Meet, яке дозволяє легко долучитися до відеоконференції.

4. Панель адміністратора

Функції адміністратора включають:

- управління користувачами (створення, редагування, блокування), (див. Лістинг 8);

- управління конференціями (моніторинг активності, редагування параметрів);
- налаштування безпеки та доступу.

Лістинг *8 Приклад реалізації керування користувачами (файл: DashboardController.php):*

```
public function manage_users(){
    $data['users'] = $this->User_model->getAll();
    $this->load->view('admin/user_manage', $data);
}
```

Функціональні вимоги мобільного застосунку Confiero:

1. Реєстрація та авторизація

- Користувачі мобільного додатку також можуть створювати та авторизуватися у своїх акаунтах.

2. Перегляд та підключення до конференцій

- Перегляд списку доступних конференцій.
- Підключення до конференції за допомогою вбудованого Jitsi Meet SDK.

3. Перегляд історії конференцій

- Збереження та відображення інформації про минулі конференції.

Вимоги до роботи API між веб-застосунком та мобільним додатком

Мобільний застосунок використовує REST API для обміну даними з сервером. Основні ендпоінти API:

Отримання інформації про конференцію (див. Лістинг 9): *GET /apiConfiero_v1/getMeeting/{id}*

Створення конференції: *POST /apiConfiero_v1/createMeeting*

Лістинг *9 Приклад структури відповіді API:*

```
{
  "status": "success",
  "meeting": {
    "id": 5678,
    "name": "Вебінар з розробки ПЗ",
    "link": "https://meet.jit.si/confiero_webinar",
    "date": "2025-06-01",
    "time": "14:00"
  }
}
```

Додаткові функціональні вимоги:

1. Повідомлення та нотифікації

Система забезпечує надсилання електронних повідомлень користувачам щодо важливих подій (наприклад, підтвердження реєстрації, нагадування про конференцію), (див. Лістинг 10).

Лістинг 10: Приклад моделі надсилання листів (файл: MailService_model.php):

```
public function sendEmail($recipient, $subject, $message){
    $this->email->from('no-reply@confiero.com');
    $this->email->to($recipient);
    $this->email->subject($subject);
    $this->email->message($message);
    return $this->email->send();
}
```

2. Безпека системи

Реалізована перевірка доступу за ролями (адміністратор, звичайний користувач). Забезпечення безпеки передачі даних за допомогою HTTPS та

стандартів шифрування платформи Jitsi Meet.

Отже всі описані функціональні вимоги визначають ключові можливості системи Confiero, які є необхідними для ефективної роботи веб-застосунку та мобільного додатку. Реалізація цих вимог дозволяє створити гнучке, безпечне і зручне рішення для організації відеоконференцій, яке здатне задовольнити потреби широкого кола користувачів.

3.4 Вимоги до апаратного та програмного забезпечення

Для забезпечення стабільної роботи та високої продуктивності веб-застосунку та мобільного додатку Confiero необхідно ретельно спланувати та врахувати апаратні та програмні вимоги до серверних ресурсів, клієнтських пристроїв, а також інструментів розробки та підтримки проєкту системи. Апаратні вимоги до сервера: Сервер забезпечує основну логіку застосунку, взаємодію з базою даних, обробку HTTP-запитів від користувачів, генерацію відповідей та інтеграцію із сервісом Jitsi Meet.

Мінімальні вимоги до серверного обладнання:

- *Процесор:* багатоядерний серверний процесор Intel Xeon або AMD EPYC з мінімум 4 ядрами та частотою від 2.4 GHz для швидкої обробки численних запитів.
- *Оперативна пам'ять (RAM):* 8 ГБ оперативної пам'яті для базового функціонування веб-сервера, PHP-середовища та СКБД.
- *Накопичувач:* SSD-диск обсягом не менше 128 ГБ для швидкого доступу до даних та ефективного зберігання інформації.
- *Мережева карта:* Гігабітна мережева карта для забезпечення швидкого і стабільного доступу користувачів до веб-додатку.
- *Резервування живлення:* рекомендовано використовувати систему безперебійного живлення (UPS), щоб уникнути простоїв через перебої в

електроживленні.

Рекомендовані вимоги до серверного обладнання (для високого навантаження):

- Процесор: 8+ ядер Intel Xeon або AMD EPYC;
- Оперативна пам'ять: від 16 ГБ RAM;
- SSD-диск: 256 ГБ і більше.

Програмна інфраструктура, яка використовується сервером Confiero:

Операційна система:

- Ubuntu Server (версія 20.04 і новіше), CentOS (версія 8 і вище), Debian (версія 10 і вище).

Веб-сервер:

- Apache HTTP Server (версія 2.4 і вище) або Nginx (версія 1.18 і вище).

Мова програмування:

- PHP (версія 7.4 і вище).

Система керування базами даних (СКБД):

- MySQL (версія 8.0 і вище) або MariaDB (версія 10.3 і вище).

Розширення PHP:

- PDO, mysqli, mbstring, json, curl, openssl.

Є також чіткі вимоги до клієнтських пристроїв, а саме для забезпечення зручного та комфортного користування веб-інтерфейсом Confiero:

- *Операційна система*: Windows 10 і вище, macOS 10.15 і вище, Linux Ubuntu або аналогічні сучасні дистрибутиви.
- *Веб-браузер*: Google Chrome, Mozilla Firefox, Microsoft Edge, Safari останніх версій.
- *Оперативна пам'ять*: мінімум 4 ГБ оперативної пам'яті для коректної роботи браузера з відеоконференціями.
- *Процесор*: Intel Core i3 (і еквівалент) або потужніший.

- *Мережеве з’єднання*: стабільне інтернет-з'єднання зі швидкістю не менше 5 Мбіт/с для комфортного відеозв’язку.

Важливими також є вимоги до апаратного та програмного забезпечення мобільних пристроїв, а саме: мобільний додаток Confiero, створений на основі Flutter, має наступні вимоги:

Для Android-пристроїв:

- Операційна система: Android 6.0 (API рівень 23) або вище;
- Оперативна пам’ять: 2 ГБ або більше;
- Вільна пам'ять для встановлення: мінімум 100 МБ.

Для iOS-пристроїв:

- Операційна система: iOS 12 або вище;
- Оперативна пам’ять: 2 ГБ або більше;
- Вільна пам'ять для встановлення: мінімум 100 МБ.

Під час розробки, я використовував інструменти для розробки, тестування та підтримки системи, а саме для створення та підтримки веб-застосунку і мобільного додатку Confiero використовувалися наступні програмні інструменти:

- Інтегроване середовище розробки (IDE):
- PHPStorm (для веб-розробки на PHP);
- Android Studio та Visual Studio Code (для розробки мобільного додатку на Flutter).

Інструменти для тестування:

- Postman (тестування REST API);
- PHPUnit (юніт-тестування коду на PHP);
- Flutter Test Framework (для тестування Flutter-додатку).

Отже таким чином, правильно підібрані апаратні та програмні ресурси дозволяють системі Confiero стабільно функціонувати, забезпечувати високу

швидкість обробки запитів, стабільність відеоконференцій та зручний користувацький інтерфейс. Дотримання зазначених вимог гарантує високий рівень доступності сервісу та можливість легкої масштабованості і підтримки системи в подальшому.

3.5 Опис модулів та алгоритмів веб-застосунку

Розроблена поточна система Confiero є масштабованим веб-застосунком для організації, аудиту та управління відеоконференціями. Для забезпечення поставленої мети та ефективної взаємодії між користувачами, сервером і також сервером з базою даних, у проєкті реалізовано кілька функціональних модулів, кожен із яких виконує конкретне завдання. Усі модулі взаємодіють через REST API або безпосередньо через серверну частину, написану на PHP із використанням CodeIgniter.

Перший алгоритм 1, а саме: Аутентифікація та керування доступом, де в мене контролер AuthController.php (див. Лістинг 10), та модель SystemUtility_model.php. Цей алгоритм стійкий та відповідає за вхід користувачів у систему. В залежності від типу користувача (адміністратор або звичайний користувач), система надає доступ до відповідних функцій.

Основні етапи:

Отримання даних з форми входу.

Перевірка відповідності у базі даних.

Ініціалізація сесії у випадку успішної автентифікації.

Лістинг *10 Валідація авторизації (файл: AuthController.php):*

```
    function processAuthAttempt($userMail = '', $passKey = '')
{
        // Перевірка вхідних даних на наявність
        if (trim($userMail) === '' || trim($passKey) === '') {
```

```
            return 'invalid';
        }
        // Побудова умов фільтрації для пошуку облікового
запису
        $lookupCriteria = [
            'email' => $userMail,
            'password' => $passKey,
            'status' => '1'
        ];
        // Виконання запиту до БД та отримання даних
користувача
        $result = $this->db->get_where('user',
$lookupCriteria);
        $account = $result->row();
        // Перевірка, чи знайдено хоча б один збіг
        if ($result->num_rows() === 1 && !empty($account)) {
            // Заносимо основні ідентифікатори в сесію
            $sessionData = [
                '__auth'     => true,
                '__uid'      => $account->user_id,
                '__uname'    => $account->name,
                '__umail'    => $account->email,
                '__role'     => $account->role
            ];
            $this->session->set_userdata($sessionData);
            // Фіксуємо момент входу
            $this->db->where('user_id', $account->user_id)
                     ->update('user',    ['last_login'    =>
date('Y-m-d H:i:s')]);
            // Прив'язка сесійного токена до типу ролі
            $roleFlags = [
```

```
                'admin'      => ['_a_token' => true, 'mode' =>
'admin'],
                'teacher'    => ['_t_token' => true, 'mode' =>
'teacher'],
                'subscriber' => ['_s_token' => true, 'mode' =>
'subscriber']
            ];
            if (array_key_exists($account->role, $roleFlags))
{
                $this->session-
>set_userdata($roleFlags[$account->role]);
            }
            return 'success';
        }
        // Якщо не знайдено користувача
        return 'invalid';
    }
```

Ось також приклад це контролер відповідає за аутентифікацію користувачів (див. Лістинг 11). Він реалізує логіку авторизації, перевірки облікових даних та керування сесіями, а саме логіка:

- Отримання POST-даних з форми входу;
- Виклик SystemUtility_model для перевірки користувача;
- Запис ідентифікатора користувача в сесію;
- Переадресація в залежності від результату.

Лістинг *11 Аутентифікація користувачів (файл: AuthController.php):*

```
public function execute_login_flow() {
    // Отримання вхідних параметрів через POST, з очищенням
    $userLogin = $this->input->post('email', true);
    $userSecret = $this->input->post('password', true);
```

```
        // Підключення моделі для перевірки облікових даних
        $this->load->model('SystemUtility_model');
        $authResult = $this->SystemUtility_model->verify_user($userLogin, $userSecret);

        // Якщо вхід успішний — зберігаємо дані користувача
        if (!is_null($authResult) && property_exists($authResult, 'id')) {
            $this->session->set_userdata(['user_id' => $authResult->id]);
            redirect('dashboard', 'refresh');
            return;
        }

        // Якщо автентифікація провалилася — показуємо повідомлення
        $this->session->set_flashdata('error', 'Помилка входу: перевірте правильність даних.');
        redirect('login', 'refresh');
    }
```

Цей контролер напряму пов'язаний із представленням application/views/login.php, де формується інтерфейс авторизації.

Наступний 2 алгоритм, це створення відеоконференції, де контролер: MeetingRoomController.php (див. Лістинг 12), а модель: ApiConfiero_v1_model.php. Цей Алгоритм відповідає за створення нової відеозустрічі, генерування посилання Jitsi Meet та запис параметрів у базу даних. Представлення візуальні для користувача у: application/views/admin/meeting_add.php, meeting_history.php.

Лістинг *12 Підключення до конференції зі збереженням у БД (файл: MeetingRoomController.php):*

```
    public function accessVirtualRoom($code = '') {
        // Нормалізація коду — прибираємо усе, що не цифри, букви або дефіси
        $cleanCode = preg_replace('/[^a-zA-Z0-9\-]/', '', $code);

        // Якщо ввімкнено вимогу авторизації — перевірка сесії
        if (get_app_config('app_mandatory_login') === 'true') {
            $isLoggedIn = ($this->session->userdata('login_status') === 1);
            if (!$isLoggedIn) {
                $this->session->set_flashdata('error', 'Ви повинні спочатку увійти в систему.');
                $this->session->unset_userdata('login_redirect_url');

                $pendingRoomURL = base_url('room/' . $cleanCode);
                $this->session->set_userdata('login_redirect_url', $pendingRoomURL);
                redirect('login', 'refresh');
                return;
            }
        }

        // Якщо код порожній — відмова
        if (trim($cleanCode) === '') {
```

```
            $this->session->set_flashdata('error',
'Неприпустимий ідентифікатор сесії.');
            redirect('room', 'refresh');
            return;
        }

        // Обмеження: заборонено входити без авторизації?
        $openEntryAllowed                                    =
get_app_config('allow_unauthorized_meeting_code');
        if ($openEntryAllowed !== 'true') {
            $validCode      =      $this->SystemUtility_model-
>codeverif_meetings($cleanCode);
            if (!$validCode) {
                $this->session->set_flashdata('error',
'Доступ за неавторизованим кодом заборонено адміністратором.');
                redirect('room', 'refresh');
                return;
            }
        }

        // Формування запису і виклик моделі для збереження
        $entryLog = [
            'meeting_code' => $cleanCode,
            'nick_name'    => '',
            'user_id'                    =>    $this->session-
>userdata('user_id')
        ];

        $this->SystemUtility_model-
>record_meeting_join($entryLog);
```

```
        // Відображення кімнати
        $this->load->view('admin/room',   ['meeting_code'   =>
$cleanCode]);
    }
```

Дуже важливий 3 Алгоритм взаємодії з мобільним застосунком через REST API, де Контролер: application/controllers/api/ApiConfiero_v1.php, а Модель: application/models/ApiConfiero_v1_model.php. Для забезпечення безперервної інтеграції між веб-застосунком Confiero і мобільним додатком, написаним на Flutter, у системі реалізовано спеціалізований API-контролер ApiConfiero_v1.php. Головна функція цього контролера обслуговування запитів, що надходять від клієнтського застосунку на Flutter. Це дозволяє мобільному додатку працювати з сервером у форматі клієнт–сервер, обмінюючись даними через REST API за допомогою форматів JSON.

Основні функції API:

Авторизація користувача користувач мобільного додатку має можливість увійти у власний обліковий запис. Для цього з Flutter-застосунку надсилається спеціальний POST-запит, який містить email та пароль.

На сервері ці дані передаються в ApiConfiero_v1_model, яка здійснює запит до бази даних. Якщо користувача знайдено, у відповідь надсилається структура JSON з інформацією про обліковий запис. У протилежному випадку користувач отримує повідомлення про помилку (див. Лістинг 13).

Лістинг *13 Метод POST для авторизації користувача в мобільному додатку (файл: api/ApiConfiero_v1.php):*

```
public function processLoginRequest() {
    $inputEmail = trim($this->input->post('email'));
    $inputPass  = trim($this->input->post('password'));
    $hashedPass = md5($inputPass);
```

```
        $responsePayload = [];

        // Валідація email та пароля
        if (filter_var($inputEmail) && !empty($hashedPass)) {
            $isUserValid    =    $this->ApiConfiero_v1_model-
>validate_user($inputEmail, $hashedPass);

            if ($isUserValid) {
                $searchCriteria = [
                    'email'    => $inputEmail,
                    'password' => $hashedPass,
                    'status'   => '1'
                ];

                $userQuery   =   $this->db->get_where('user',
$searchCriteria);

                if ($userQuery->num_rows() > 0) {
                    $this->db->where($searchCriteria);
                    $this->db->update('user', ['last_login' =>
date('Y-m-d H:i:s')]);

                    $userDetails          =          $this-
>ApiConfiero_v1_model->get_user_info($inputEmail,
$hashedPass);

                    $responsePayload = [
                        'status'        => 'success',
                        'user_id'           =>  $userDetails-
>user_id,
                        'name'         => $userDetails->name,
```

```
                        'email'        => $userDetails->email,
                        'phone'        => $userDetails->phone,
                        'meeting_code'      =>   $userDetails-
>meeting_code,
                        'image_url'               =>   $this-
>SystemUtility_model->get_image_url('user',      $userDetails-
>user_id),
                        'gender'        => 'Unknown',
                        'role'         => $userDetails->role,
                        'join_date'          =>  $userDetails-
>join_date,
                        'last_login'         =>  $userDetails-
>last_login
                    ];

                    if ($userDetails->gender === '1') {
                        $responsePayload['gender']          =
'Чоловік';
                    } elseif ($userDetails->gender === '0') {
                        $responsePayload['gender'] = 'Жінка';
                    }

                } else {
                    $responsePayload = [
                        'status' => 'error',
                        'data'   => 'Увага! Ваш акаунт вимкнено
та деактивовано. Будь ласка, зверніться до адміністратора
системи за контактними даними на веб-сайті.'
                    ];
                }
```

```
            } else {
                $responsePayload = [
                    'status' => 'error',
                    'data'    => 'Увага! Ваш Email та Пароль
нажаль не співпадають. Спробуйте ще раз!'
                ];
            }

        } else {
            $responsePayload = [
                'status' => 'error',
                'data'   => 'Будь ласка, введіть коректні email
та пароль.'
            ];
        }

        $this->response($responsePayload, 200);
    }
```

Аналізуючи цей підхід, він дозволяє мобільному клієнту безпечно працювати з обліковими записами користувачів, отримувати доступ до закритих частин застосунку лише після проходження авторизації. Наступна функція в цьому контролері Апі не менш важлива, ця функція дозволяє створити нового користувача за допомогою мобільного додатка (див. Лістинг 14).

Лістинг *14 Метод POST для реєстрації користувача в мобільному додатку (файл: api/ApiConfiero_v1.php):*

```
    public function registerUser_post() {
        $userName  = trim($this->input->post('name'));
        $userEmail = trim($this->input->post('email'));
        $userPass  = trim($this->input->post('password'));
```

```
        $responsePayload = [];

        // Валідація email та пароля
        if (filter_var($userEmail) && !empty($userPass) &&
strlen($userPass) > 3) {
            $hashedPass = md5($userPass);
            $canSignup      =    $this->ApiConfiero_v1_model-
>check_signup_ability_by_email($userEmail);

            if ($canSignup) {
                $this->ApiConfiero_v1_model-
>create_user($userName, $userEmail, $hashedPass);

                $this->load->model('MailService_model');
                $this->MailService_model-
>account_opening_email($userEmail, $userPass);

                $userInfo    =    $this->ApiConfiero_v1_model-
>get_user_info($userEmail, $hashedPass);

                $responsePayload = [
                    'status'        => 'success',
                    'user_id'       => $userInfo->user_id,
                    'name'          => $userInfo->name,
                    'email'         => $userInfo->email,
                    'phone'         => $userInfo->phone,
                    'meeting_code'          =>     $userInfo-
>meeting_code,
                    'role'          => $userInfo->role,
                    'image_url'                  =>    $this-
```

```
>SystemUtility_model->get_image_url('user',         $userInfo-
>user_id),
                    'gender'        => 'Невідомо',
                    'join_date'     => $userInfo->join_date,
                    'last_login'    => $userInfo->last_login
                ];

                if ($userInfo->gender === '1') {
                    $responsePayload['gender'] = 'Чоловік';
                } elseif ($userInfo->gender === '0') {
                    $responsePayload['gender'] = 'Жінка';
                }

            } else {
                $responsePayload = [
                    'status' => 'error',
                    'data'   => 'Увага! Цей Email вже існує в
системі…'
                ];
            }

        } else {
            $responsePayload = [
                'status' => 'error',
                'data'      =>  'Увага!  Будь  ласка,  введіть
коректні email та пароль (вимоги: пароль має бути довжиною
більше 3 символів).'
            ];
        }

        $this->response($responsePayload, 200);
```

```
}
```

Якщо розглянути загальну схему роботи цього АПІ, то Flutter надсилає запит (login, meeting_list, getMeeting), далі Контролер ApiConfiero_v1 приймає запит, після цього Контролер передає дані до моделі ApiConfiero_v1_model. Ця Модель виконує запити до бази даних. Сервер повертає результати у форматі JSON, а Мобільний додаток парсить відповідь і показує інформацію на екрані. Завдяки модулю ApiConfiero_v1 мобільний додаток на Flutter отримав повноцінний інтерфейс для взаємодії з сервером у реальному часі, а API архітектура реалізована з дотриманням принципів REST де: чіткі ендпоінти, логічна структура запитів, повернення відповідей у форматі JSON. І це дійсно надає та забезпечує максимальну зручність як для розробників, так і для користувачів системи. Крім того, логіка рознесена між контролерами і моделями, що покращує читабельність і масштабованість коду, для майбутніх нових модифікацій системи.

4 алгоритм дозволяє працювати повноцінно з панеллю адміністратора, у ній можна аналізувати статус серверів, дивитися детальну статистику. Створювати, редагувати користувачів. Вносити критично важливі зміни до конфігурації системи, редагувати Апі шлюз для мобільного додатка. Налаштувати політику конфіденційності, яка важлива і має велике значення для публікації мобільного додатка в Google Play та AppStore. У цьому контролері також можна повноцінно редагувати проект, логотип, оформлення. Створювати резервну копію бази даних. Головний Контролер: DashboardController.php (див. Лістинг 15), а Представлення у views це admin/user_manage.php.

Лістинг *15 Функція створення нового ключа API для мобільного додатка, в панелі керування (файл: DashboardController.php):*

```
function configure_api($action = '', $key_id = '') {
    $this->session->unset_userdata('current_menu');
```

```
        $this->session->set_userdata('current_menu', '6');

        if ($action === "modify") {
            $updatedData = [
                'key' => $this->input->post("key")
            ];
            $this->db->where("id", $key_id);
            $this->db->update('keys', $updatedData);
            $this->session->set_flashdata('success',  'Успіх!
Оновлення ключа для застосунку пройшло успішно.');
            redirect($this->agent->referrer());
        }

        if ($action === "regen_key") {
            if (!empty($key_id)) {
                $existingKeyQuery        =        $this->db-
>get_where('keys', ['id' => $key_id]);
                if ($existingKeyQuery->num_rows() > 0) {
                    $newKey   =   $this->SystemUtility_model-
>genfunc_randstr(24);
                    $this->db->where('id', $key_id);
                    $this->db->update('keys',    ['key'    =>
$newKey]);
                    $this->session->set_flashdata('success',
'Новий ключ створено та збережено.');
                    redirect($this->agent->referrer());
                }
            }
            redirect($this->agent->referrer());
        }
```

```
        $pageData = [
            'page_name'  => 'api_setting',
            'page_title' => 'Налаштування API',
            'key'        => $this->db->get('keys')->first_row()
        ];
        $this->load->view('admin/index', $pageData);
    }
```

5 алгоритм це обробка електронної пошти, ця модель MailService_model.php використовує шаблони представлення views/email_templete/, які написані на HTML, для відправлення листів користувачам про скидання пароля або реєстрацію користувача в системі. Також у цій моделі відбувається налаштування функції для власного сервера надсилання пошти, а також функція для тестування налаштувань та надсилання тестового листа (див. Лістинг 16).

Лістинг *16 Ця функція потрібна для надсилання електронного листа (файл: MailService_model.php):*

```
    function dispatch_mail($content = '', $subject = null,
$recipient = null, $sender = null, $replyAddress = null)
    {
        if (empty($recipient) || empty($content)) {
            return false;
        }

        $this->load->library('email');

        $mailConfig = [
            'useragent' => 'CustomMailer',
            'mailtype'  => 'html',
            'charset'   => 'UTF-8',
```

```
            'newline'   => "\r\n",
            'wordwrap'  => true
        ];

        $mailProtocol = get_app_config('protocol');

        switch ($mailProtocol) {
            case 'smtp':
                $mailConfig['protocol']    = 'smtp';
                $mailConfig['smtp_host']                    =
get_app_config('smtp_host');
                $mailConfig['smtp_user']                    =
get_app_config('smtp_user');
                $mailConfig['smtp_pass']                    =
get_app_config('smtp_pass');
                $mailConfig['smtp_port']                    =
get_app_config('smtp_port');
                $mailConfig['smtp_crypto']                  =
get_app_config('smtp_crypto');
                $mailConfig['smtp_timeout']= 30;
                break;

            case 'sendmail':
                $mailConfig['protocol'] = 'sendmail';
                $mailConfig['mailpath']                     =
'/usr/sbin/sendmail';
                break;

            default:
                $mailConfig['protocol'] = 'mail';
                break;
```

```
        }

        $this->email->initialize($mailConfig);

        $fromAddress = $sender ?? get_app_config('system_email');
        $systemName  = get_app_config('app_name');
        $mailSubject = $subject ?: 'Без теми';

        $this->email->from($fromAddress, $systemName);
        $this->email->to($recipient);

        if (!empty($replyAddress)) {
            $this->email->reply_to($replyAddress);
        }

        $this->email->subject($mailSubject);
        $this->email->message($content);

        return $this->email->send();
    }
```

6 алгоритм — це обробка помилок, де є чіткий контролер і представлення відображення помилки наприклад 404: application/views/errors/html/error_404.php. Ця функція та алгоритм забезпечує зручний висновок повідомлень про помилки: application/views/errors/html/error_general.php.

Далі розглядатимемо моделі 1 Модель системи SystemUtility_model — це дуже важлива модель, тому що вона виконує загальні запити до бази даних. Виконує валідацію користувача. Виводить дані для статистики та утворення діаграм, відображає загальну інформацію з бази даних наприклад кількість

користувачів. Створює запит на перевірку системи на працездатність та відображає затримку між серверами.

2 Модель AlertManager_mode, відповідає за створення та управління внутрішніми сповіщеннями. Та може використовуватись у спливаючих повідомленнях для адміністратора або повідомлень користувачам.

Також є helpers, а саме головне це допоміжна функція trans() бо вона використовується для генерації зрозумілих заголовків на основі ключів. Це дозволяє динамічно формувати заголовки сторінок, замінюючи snake_case на людський вигляд як приклад: user_profile → User Profile.

Важливим також є взаємозв'язок із представленнями views, де основна частина інтерфейсу реалізована у: /application/views/ для загальних сторінок (вхід, забутий пароль); та /application/views/admin/ інтерфейс адміністратора (конференції, користувачі, налаштування, email-шаблони тощо). Кожен контролер використовує відповідне представлення, передаючи в нього дані через $data (див. Лістинг 17).

Лістинг *17 Приклад зв'язку:*

```
$data['users'] = $this->SystemUtility_model->getAllUsers();
$this->load->view('admin/user_manage', $data);
```

Отже система побудована на модульному принципі, де кожен компонент відповідає за конкретну функціональність. Завдяки використанню MVC, REST API, шаблонів листів, розмежуванням адміністративного і користувацького інтерфейсів, система стала більш зрозумілою, зручною та придатною для масштабування. А алгоритми реалізовані з урахуванням практичних потреб кінцевих користувачів та технічних вимог до веб-сервісів у сфері відеоконференцій.

3.6 Розробка мобільного застосунку

У сучасному світі мобільні застосунки стають невід'ємною частиною цифрового середовища і забезпечують зручність доступу до різноманітних сервісів, платформ і функцій. У рамках даного проекту був розроблений застосунок що дозволяє користувачам приєднуватися до відеоконференцій без необхідності реєстрації або через авторизацію. Застосунок побудований на основі технології Flutter, що дозволяє створювати крос-платформені додатки для Android та iOS.

Архітектура застосунку:

Застосунок складається з кількох основних екранів: екран привітання (WelcomeScreen), екран для входу та реєстрації (LoginScreen та RegisterScreen), екран для проведення зустрічей (MeetingScreen) та профіль користувача (ProfileScreen). Кожен з цих екранів виконує чітко визначену функцію у процесі взаємодії з користувачем.

Дуже важливий момент при проектуванні системи мобільного застосування відеоконференцій. Система і SDK вимагають, щоб були грамотно встановлені дозволи для операційних систем Android та IOS [12]. А саме встановлення дозволів на доступ до: камери, локального сховища, доступу до мікрофону, доступу до технології Bluetooth, можливість змінювати і модифікувати аудіо. Всі ці дозволи, необхідні для коректної роботи Jitsi Meet SDK (див. Лістинг 18).

Лістинг *18 Список дозволів для операційної системи Android (файл: android\app\src\main\AndroidManifest.xml):*

```
    <uses-permission
android:name="android.permission.INTERNET" />
    <uses-permission  android:name="android.permission.CAMERA"
/>
```

```
    <uses-permission
android:name="android.permission.RECORD_AUDIO" />
    <uses-permission
android:name="android.permission.MODIFY_AUDIO_SETTINGS" />
    <uses-permission
android:name="android.permission.BLUETOOTH" />
    <uses-permission
android:name="android.permission.BLUETOOTH_CONNECT" />
```

Також необхідно встановити ці дозволи в налаштуваннях під операційну систему IOS [13], (див. Лістинг 19).

Лістинг *19 Список дозволів для операційної системи IOS (файл: ios\Runner\Info.plist):*

```
    <key>NSCameraUsageDescription</key>
    <string>Додаток потребує доступу до камери
відеоконференцій.</string>
    <key>NSMicrophoneUsageDescription</key>
    <string>Додаток потребує доступу до мікрофону для
відеоконференцій.</string>
```

Оскільки наш мобільний додаток працює разом з Веб-сервером по Rest Api, нам необхідно вказати в константах значення нашого URL веб-сервера, обов'язково з використанням протоколу HTTPS. А також ключ до API, який доступний в адмін панелі, (див. Лістинг 20).

Лістинг *20 Налаштування констант (файл: lib\main.dart):*

```
    const apiBase = "https://petral-
dev.com/api/ApiConfiero_v1";
    const apiKey = "aoqj4f0uki1foidhgjyq7v15";
    const configUrl = "$apiBase/config?API-KEY=$apiKey";
```

Далі у нас опис ключових функцій мобільного додатку, а саме Конфігурація сервера. Бо при запуску застосунку користувач отримує налаштування сервера для відеоконференцій через API запит. Якщо запит до сервера не вдається, застосунок використовує сервер за замовчуванням. Це гарантує коректну роботу навіть у випадку збоїв у зовнішніх системах (див. Лістинг 21).

Лістинг *21 Конфігурація сервера (файл: lib\main.dart):*

```
Future<void> loadConfig() async {
  try {
    final response = await http.get(Uri.parse(configUrl));
    final json = jsonDecode(response.body);
    setState(() {
      jitsiServer = json['app_config']?['jitsi_server'] ??
"https://jitsi.belnet.be";
      loading = false;
    });
  } catch (e) {
    setState(() {
      jitsiServer = "https://jitsi.belnet.be";
      loading = false;
    });
  }
}
```

Наступна важлива функціональна частина залучення користувачів до відеоконференцій, застосунок надає можливість користувачам приєднуватися до відеоконференцій як гість або після авторизації. Для цього потрібно ввести назву кімнати та, за бажанням, ім’я (див. Лістинг 22).

Лістинг *22 Залучення користувачів до відеоконференцій (файл:*

lib\main.dart):

```
    void _joinMeeting() async {
      if (_roomController.text.isEmpty) {
        ScaffoldMessenger.of(context).showSnackBar(
          const SnackBar(content: Text("Будь ласка, введіть назву кімнати")),
        );
        return;
      }
      final options = JitsiMeetConferenceOptions(
        room: _roomController.text,
        serverURL: widget.jitsiServer,
        userInfo: JitsiMeetUserInfo(
          displayName: _nickController.text.isNotEmpty ? _nickController.text : " Гість ",
        ),
        featureFlags: {
          "welcomepage.enabled": false,
          "add-people.enabled": false,
          "call-integration.enabled": false,
          "pip.enabled": true,
        },
      );
      final listener = JitsiMeetEventListener(
        conferenceJoined: (url) => print("Conference joined: $url"),
        conferenceTerminated: (url, error) => print("Conference terminated: $url, error: $error"),
      );
      await _jitsiMeet.join(options, listener);
```

```
}
```

Авторизація та реєстрація користувачів, користувач може увійти до застосунку за допомогою електронної пошти та пароля або зареєструватися, створивши новий обліковий запис. У разі успішної авторизації зберігаються дані користувача в локальному сховищі (SharedPreferences), що дозволяє зберігати сесію між запуском додатку, (див. Лістинг 23).

Лістинг *23 Авторизація користувачів (файл: lib\main.dart):*

```
Future<void> login() async {
  setState(() { isLoading = true; errorText = null; });
  final response = await http.post(
    Uri.parse("$apiBase/login"),
    headers: {
      'API-KEY': apiKey,
      'Content-Type': 'application/x-www-form-urlencoded',
    },
    body: {
      'email': _emailController.text.trim(),
      'password': _passwordController.text.trim(),
    },
  );
  final json = jsonDecode(response.body);
  if (json['status'] == 'success') {
    final prefs = await SharedPreferences.getInstance();
    prefs.setString('user', jsonEncode(json));
    Navigator.pushReplacement(context, MaterialPageRoute(
      builder: (_) => ProfileScreen(jitsiServer:
widget.jitsiServer, userData: json),
    ));
  } else {
```

```
        setState(() { errorText = json['data'] ?? "Помилка
авторизації"; });
      }
      setState(() { isLoading = false; });
    }
```

Лістинг *24: Реєстрація користувачів (файл: lib\main.dart):*

```
    Future<void> register() async {
        setState(() { isLoading = true; errorText = null; });
        final response = await http.post(
          Uri.parse("$apiBase/signup"),
          headers: {
            'API-KEY': apiKey,
            'Content-Type':           'application/x-www-form-
urlencoded',
          },
          body: {
            'name': _nameController.text.trim(),
            'email': _emailController.text.trim(),
            'password': _passwordController.text.trim(),
          },
        );
        final json = jsonDecode(response.body);
        if (json['status'] == 'success') {
          final prefs = await SharedPreferences.getInstance();
          prefs.setString('user', jsonEncode(json));
          Navigator.pushReplacement(context,
MaterialPageRoute(
            builder:    (_)    =>    ProfileScreen(jitsiServer:
widget.jitsiServer, userData: json),
```

```
        ));
      } else {
        setState(() { errorText = json['data'] ?? "Помилка
реєстрації"; });
      }
      setState(() { isLoading = false; });
    }
```

Оновлення профілю та історія зустрічей, користувачі застосунку мають можливість оновлювати свої персональні дані, а також переглядати історію своїх відеозустрічей. Дані профілю зберігаються на сервері, що дозволяє зберігати інформацію між сесіями (див. Лістинг 25).

Лістинг *25 Оновлення профілю (файл: lib\main.dart):*

```
Future<void> updateProfile() async {
  setState(() { isLoading = true; info = null; });
  final response = await http.post(
    Uri.parse("$apiBase/update_profile"),
    headers: {
      'API-KEY': apiKey,
      'Content-Type': 'application/x-www-form-urlencoded',
    },
    body: {
      'id': widget.userData['user_id'].toString(),
      'name': nameController.text.trim(),
      'email': emailController.text.trim(),
      'phone': phoneController.text.trim(),
    },
  );
  final json = jsonDecode(response.body);
  setState(() {
```

```
          info = json['status'] == 'success' ? 'Профіль оновлено'
: (json['data'] ?? 'Помилка оновлення');
          isLoading = false;
        });
      }
```

Отже створений мобільний застосунок демонструє ефективне використання технології Flutter для створення крос платформеного рішення з підтримкою відеоконференцій. А використання сторонніх бібліотек, таких як jitsi_meet_flutter_sdk для інтеграції з сервером Jitsi, дозволяє забезпечити високу якість відео та аудіозв'язку. Розробка також включає механізми авторизації, реєстрації та оновлення профілю користувачів, що робить застосунок зручним для широкого кола користувачів.

Застосунок підтримує гнучкість в налаштуваннях серверів і забезпечує зручний інтерфейс для користувачів, що не потребує додаткових зусиль для підключення до конференцій.

3.7 Дизайн інтерфейсу користувача

Веб-застосунок містить зручний та зрозумілий інтерфейс для швидкого використання всіх можливостей системи. На головній сторінці веб-порталу користувач може легко приєднатися до конференції за допомогою введення спеціального ідентифікатора (рисунок 5). Крім цього, доступна можливість організувати власну відеоконференцію, задавши її назву та унікальний ідентифікатор (рисунок 6). Система підтримує авторизацію користувачів за допомогою введення електронної пошти та пароля (рисунок 7). У разі втрати пароля передбачена сторінка для його відновлення (рисунок 8).

Адміністратор має доступ до важливих функцій керування через окрему адміністративну панель. На сторінці адміністратора відображається основна

системна інформація, яка дозволяє оперативно оцінити стан роботи платформи (рисунок 8). Також адміністратор може легко керувати зустрічами, створювати, редагувати або видаляти їх (рисунок 10). Для швидкого пошуку інформації про проведені зустрічі існує окремий розділ з історією відеоконференцій (рисунок 11).

Важливим інструментом адміністративної панелі є управління користувачами, що дозволяє переглядати список зареєстрованих користувачів та редагувати їхні облікові записи (рисунок 12). Всі основні налаштування системи доступні в зручному інтерфейсі адміністративної панелі, де можна задавати критично важливі параметри (рисунок 13). Для інтеграції з мобільним додатком передбачені спеціальні налаштування API, включаючи створення та управління ключами доступу (рисунок 14). Також реалізована функція створення резервної копії бази даних безпосередньо з адмін-панелі, що забезпечує додаткову безпеку і можливість швидкого відновлення даних (рисунок 15):

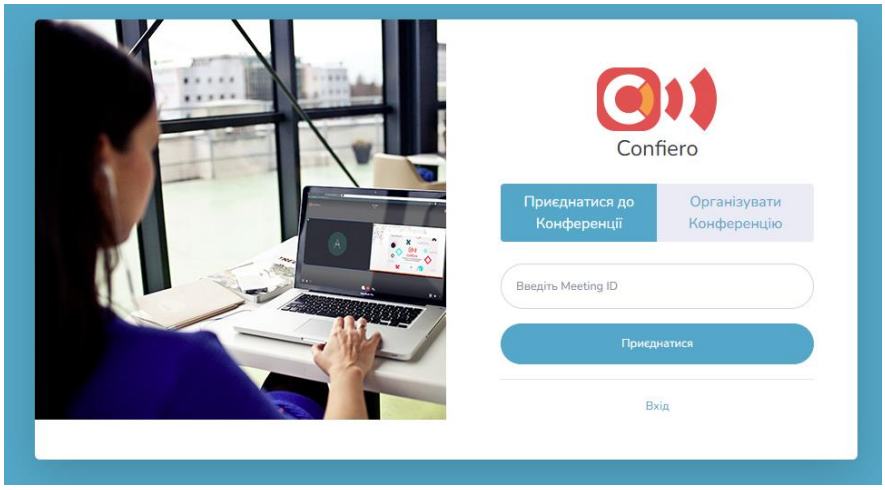


Рисунок 5 — *Головна сторінка веб порталу, на якому можна приєднатися до конференції з ідентифікатора.*

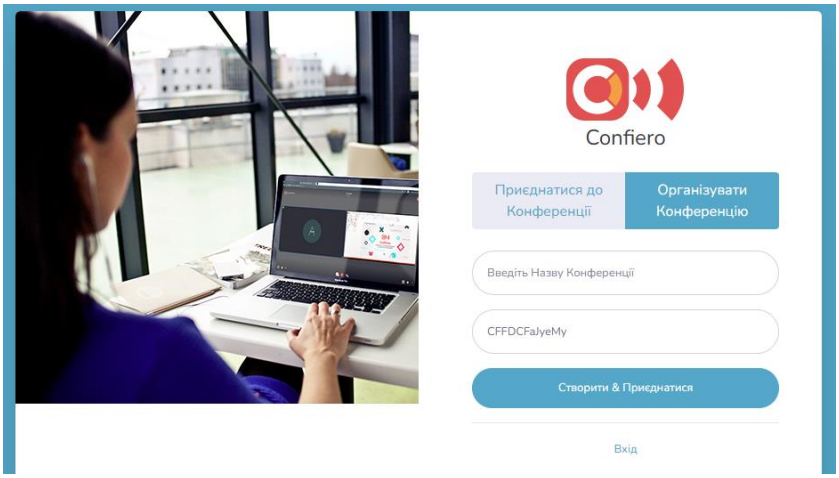


Рисунок 6 — *Можливість організації конференції із зазначенням назви та спеціального ідентифікатора.*

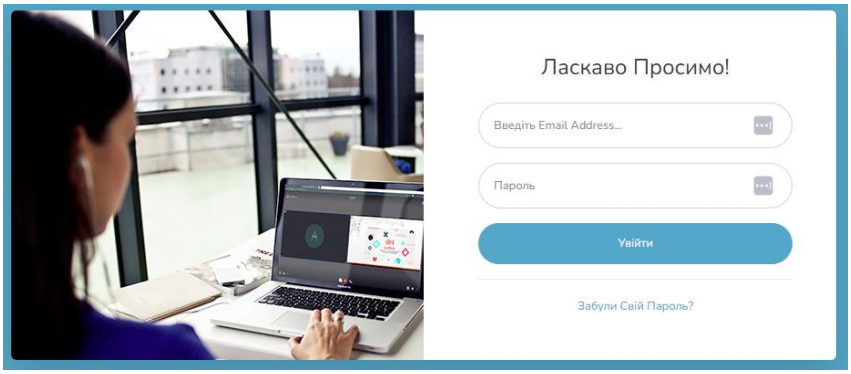


Рисунок 7 — *Авторизація в системі проекту на веб-сайті.*

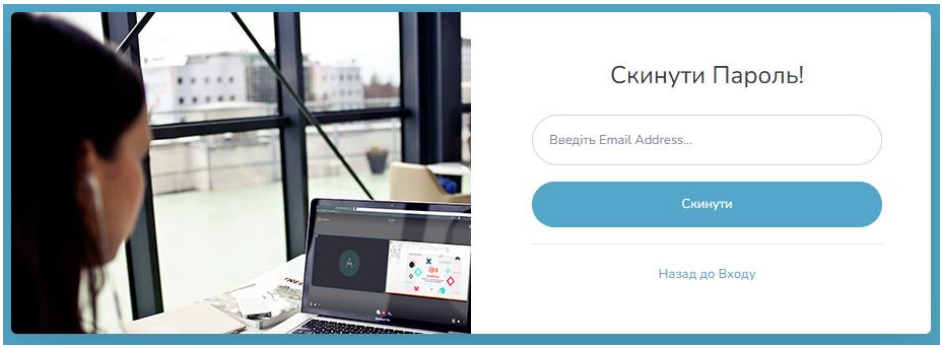


Рисунок 8 — *Сторінка скидання пароля користувача.адміністратора та відображення важливої системної інформації та інформації по платформі.*

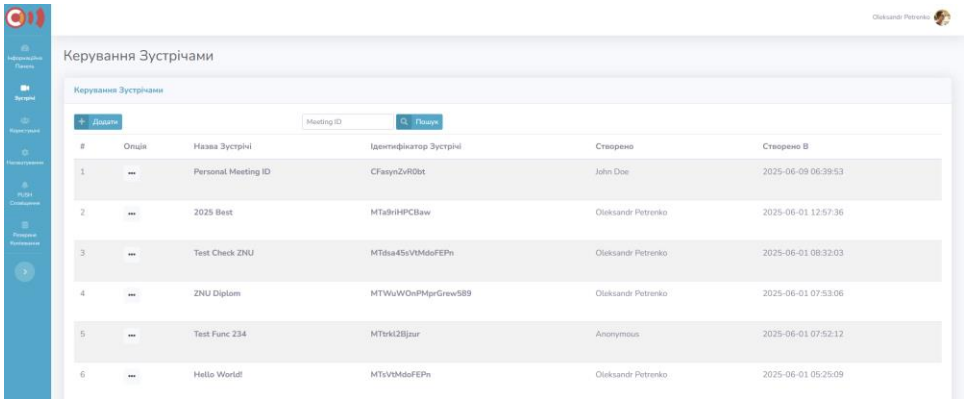


Рисунок 10 — *Управління зустрічами в адмін-панелі.*

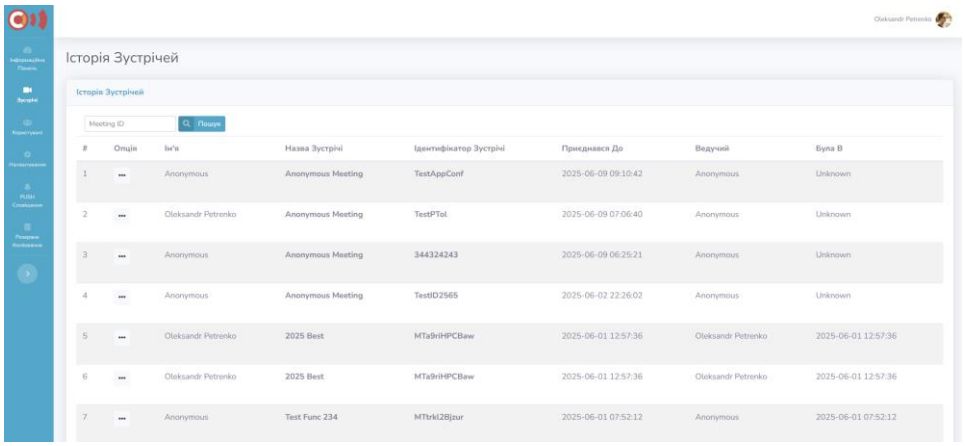


Рисунок 11 — *Історія зустрічей, пошук зустрічей та редагування інформації про зустрічі у панелі адміністратора.*

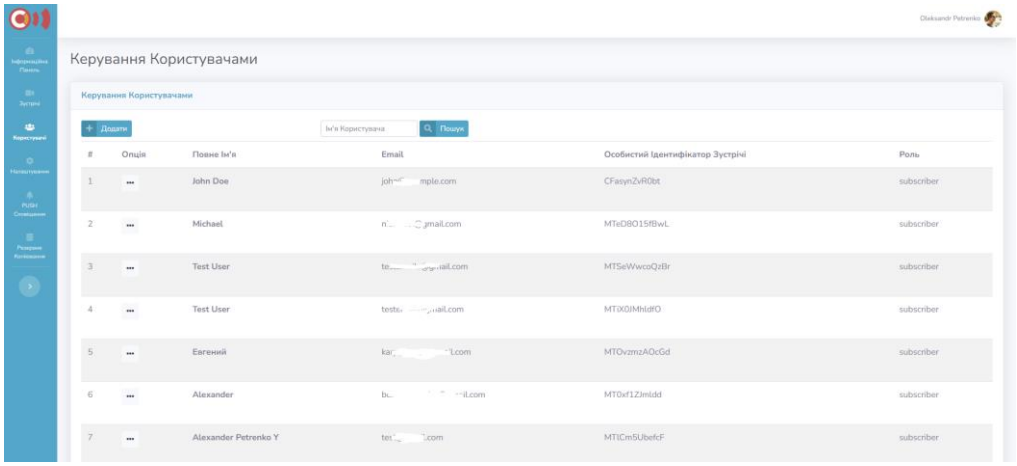


Рисунок 12 — *Список користувачів системи та можливість редагувати облікові записи користувачів.*

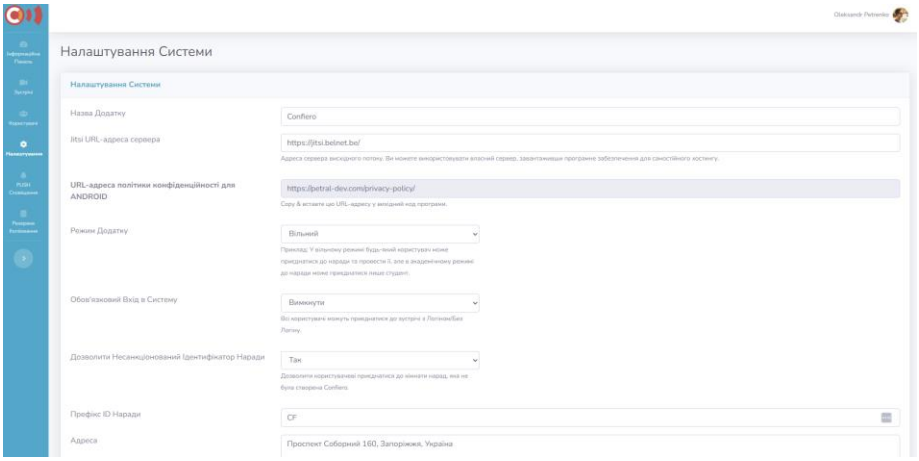


Рисунок 13 — *Налаштування всієї системи та встановлення важливих параметрів усередині системи за допомогою панелі адміністратора.*

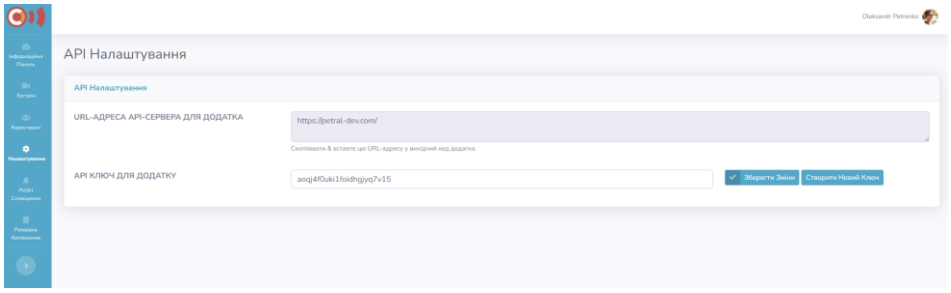


Рисунок 14 — *Налаштування API системи для інтеграції в мобільному додатку та ключ доступу.*

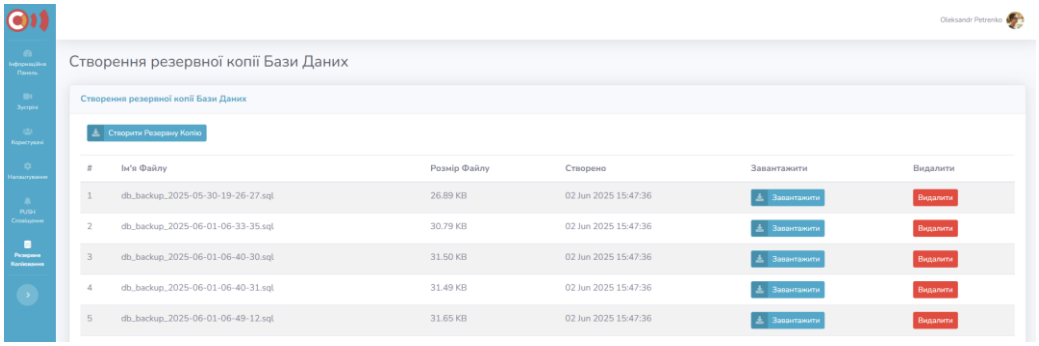


Рисунок 15 — *Адмін Панель, створення резервної копії бази даних системи.*

Мобільний додаток включає архітектуру для демонстрації можливостей системи

та API. На головному екрані мобільного додатка є дві кнопки, які можна використовувати для використання мобільного додатка (рисунок 16). Можна приєднатися до відеоконференції без авторизації або реєстрації (рисунок 17). Після попереднього етапу починає працювати мобільне SDK для проведення конференції, яке запитує інформацію про користувача від клієнта через API або з попереднього етапу (рисунок 18).

Демонстрація роботи системи та проведення конференції в мобільному додатку весь інструментарій доступний для користувачів (рисунок 19). Авторизація в програмі та реєстрація доступні після вибору спеціальної кнопки на головному екрані. Авторизація та Реєстрація (рисунок 20 та 21). Оновлення профілю користувача, зміна його фотографії, відображення історії користувача (рис. 22):

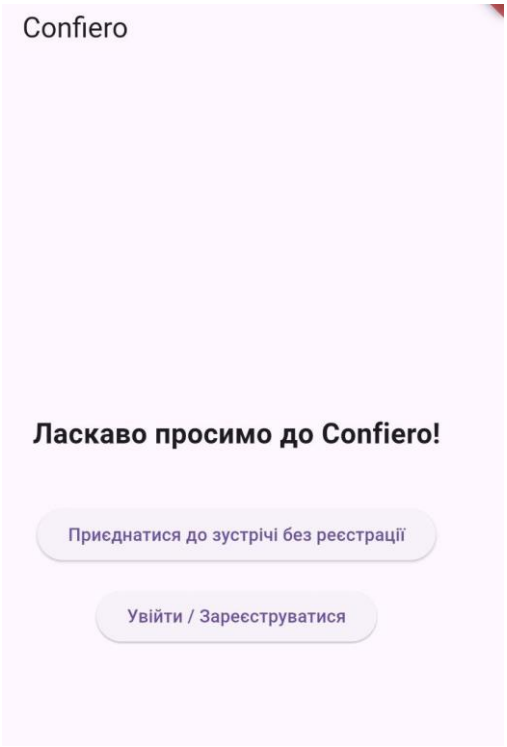


Рисунок 16 — *Головний екран мобільного додатка*

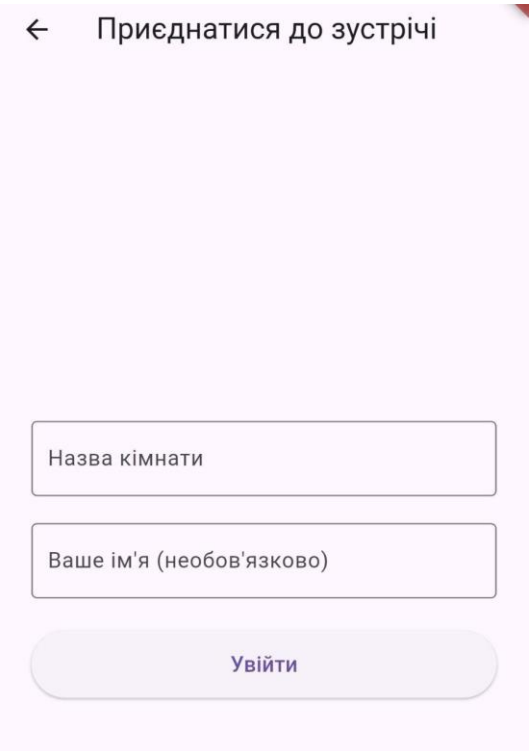


Рисунок 17 — *Вказати важливу інформацію, якщо користувач не увійшов до системи* .

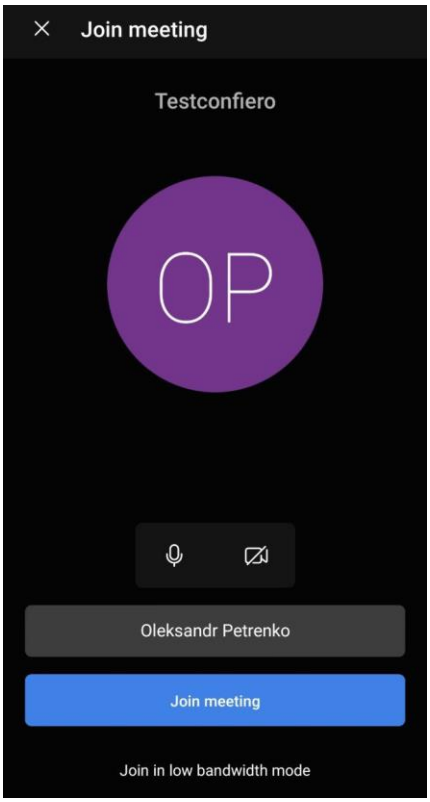


Рисунок 18 — *Авторизація користувача у конференції та лобі для перевірки обладнання (є можливість вибрати режим низької пропускної спроможності, коли якість зв'язку погана, а на конференції необхідно перебувати).*

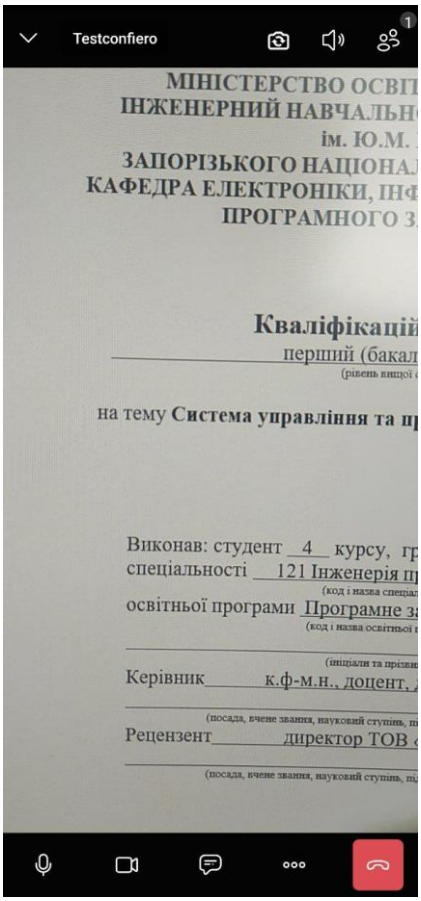


Рисунок 19 — *Демонстрація можливості проведення відеоконференції у мобільному додатку.*

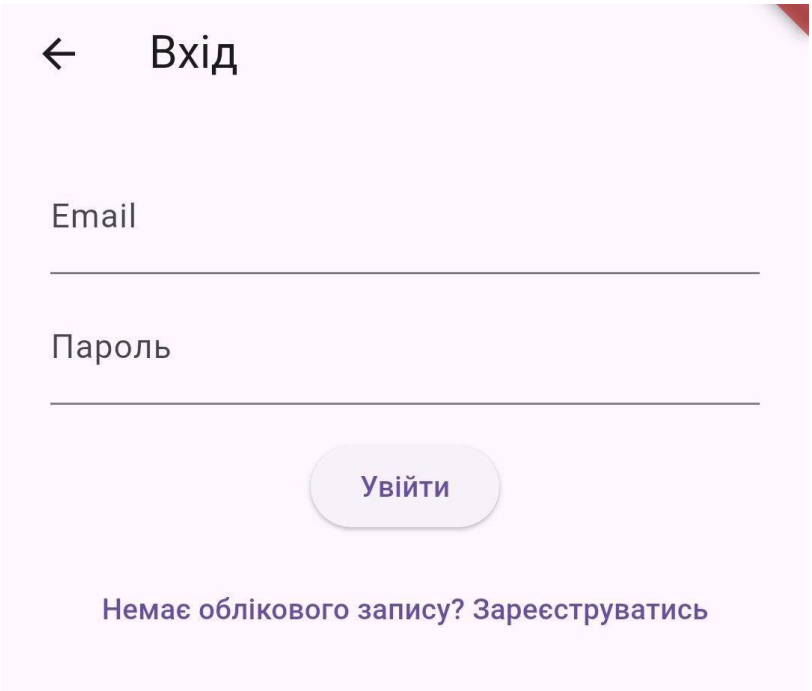


Рисунок 20 — *Авторизація користувача в мобільному додатку*.

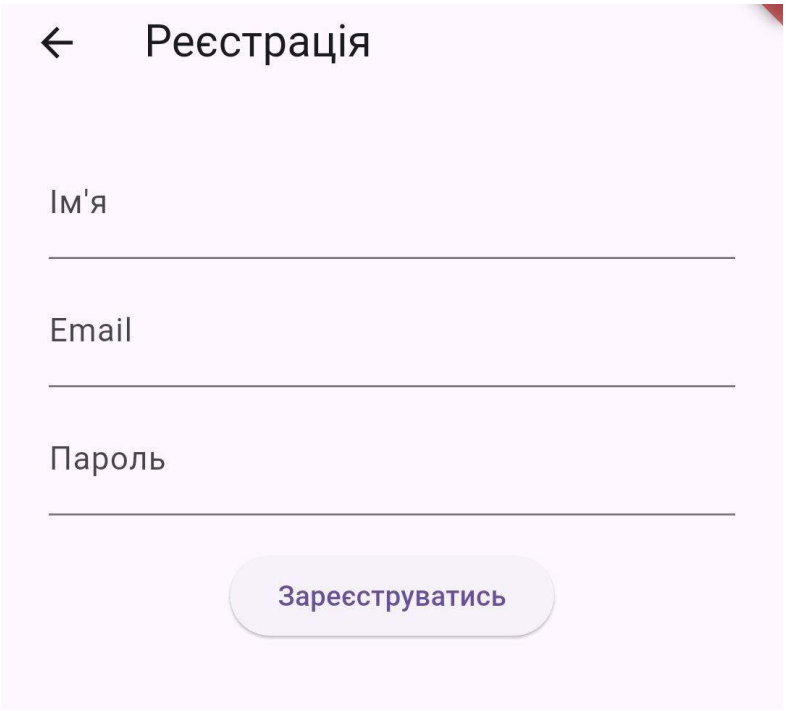


Рисунок 21 — *Реєстрація користувача у мобільному додатку.*

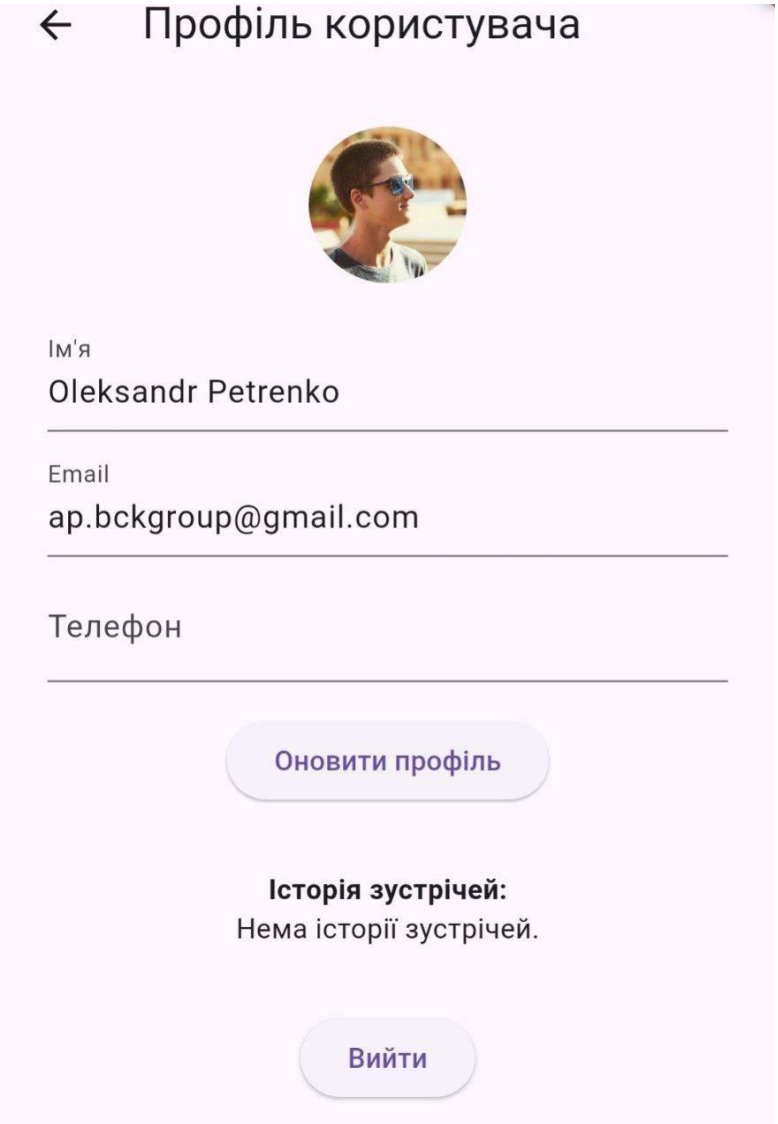


Рисунок 22 — *Редагування профілю користувача в мобільному додатку.*

3.8 Тестування та аналіз ефективності

Завершальним етапом розробки системи Confiero стало її тестування та аналіз ефективності функціонування. Основна мета тестування готового продукту: виявити потенційні помилки, перевірити стабільність роботи при різних навантаженнях та оцінити загальний користувацький досвід як у веб-застосунку, так і в мобільному додатку.

Підходи до тестування які були використані

- У процесі перевірки були застосовані декілька видів тестування:
- Функціональне тестування: перевірка працездатності всіх модулів (реєстрація, вхід, створення конференцій, API).
- Інтеграційне тестування: перевірка взаємодії між окремими компонентами (контролери, моделі, представлення, API).
- Тестування API: використано Postman для перевірки правильності відповідей на запити від мобільного застосунку (див. Рис.: 23).

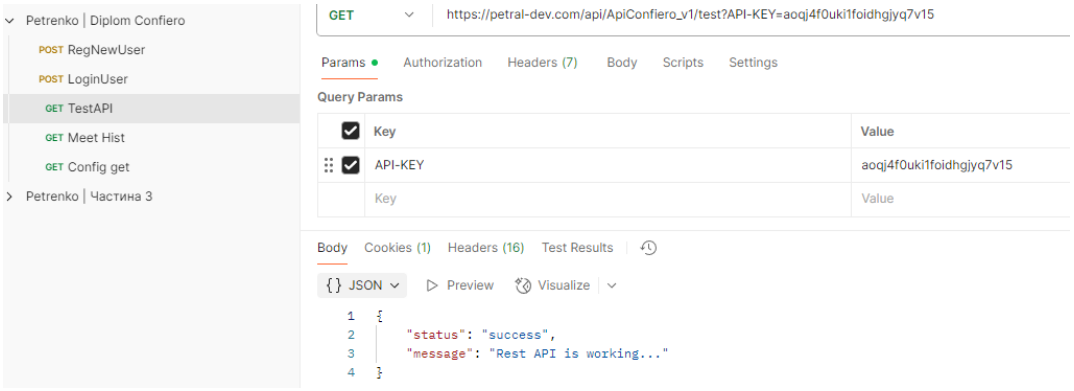


Рисунок 23 — *PostMan*

- Юзабіліті-тестування: перевірка зручності інтерфейсу, навігації, дизайну веб та мобільного додатків.
- Тестування навантаження: емуляція паралельних з'єднань та створення кількох сесій одночасно.

Для тестування використовувалися такі інструменти:

- Postman - перевірка REST API-запитів;
- Google Chrome DevTools - моніторинг мережевих запитів, продуктивності;
- Android Studio (Flutter Emulator) - для тестування мобільного клієнта.

Результати тестування:

- Функціональні тести показали, що основна логіка працює коректно. Усі основні сценарії — реєстрація, вхід, створення відеоконференції, авторизація через API — були пройдені успішно.

- Інтеграційні тести підтвердили, що контролери та моделі коректно обмінюються даними. Зокрема, API-контролер ApiConfiero_v1 успішно обробляє запити на створення/отримання конференцій, а також коректно повертає JSON-відповіді.
- Тестування мобільного додатку показало, що додаток стабільно працює на Android 8.0 і вище, не вимагає великого обсягу ресурсів, а інтеграція з Jitsi Meet відбувається без збоїв. Усі REST API-запити обробляються швидко, із середнім часом відповіді < 200 мс.
- Під час навантаження (емуляція одночасного входу 15 користувачів) система не втратила стабільність. Завдяки використанню CodeIgniter та оптимізованим SQL-запитам затримка відповіді зросла незначно (~20%).

Аналіз ефективності всього розробленого проекту:

Результати тестування показали, що веб-застосунок Confiero має високу продуктивність та стабільність. Це стало можливим завдяки дотриманню принципів розділення логіки (MVC), використанню REST API та обмеженню зайвого навантаження на сервер. Крім того, легкий за обсягом фронтенд без зайвих залежностей дозволив мінімізувати час завантаження сторінок.

Система демонструє низький поріг входу для користувачів: інтерфейс інтуїтивний, мобільна версія має логічно побудовану навігацію, а зворотний зв'язок на дії користувача (наприклад, повідомлення про успішну реєстрацію або помилки входу) реалізований чітко та зрозуміло.

3.9 Висновок за розділом

У третьому розділі дипломної роботи було здійснено проектні та розробницькі заходи щодо створення системи відеоконференцій Confiero, яка містить веб-застосунок і мобільний додаток з інтеграцією технології Jitsi Meet.

У рамках технічного проектування:

Виконано аналіз наявних рішень на ринку, виділено основні проблеми: обмежені можливості безкоштовного використання, закритість коду, недостатню гнучкість налаштувань, складність інтерфейсу. Це підтвердило необхідність створення власного рішення.

Обґрунтовано вибір технологічного стека: мови програмування PHP і Dart, фреймворки CodeIgniter і Flutter, технологія REST API та СУБД MySQL, на основі детального аналізу їх переваг та можливостей.

Спроектовано архітектуру системи із застосуванням шаблону MVC, що дозволило чітко структурувати програмний код, розмежувати компоненти і зробити систему гнучкою та масштабованою.

У процесі розробки було створено:

- веб-додаток, що містить адміністративну панель, модуль авторизації користувачів та управління відеоконференціями;
- мобільний додаток, реалізований на фреймворку Flutter, що дозволяє користувачам створювати, переглядати та підключатися до конференцій;
- REST API, що забезпечує взаємодію клієнтських додатків із серверною частиною;
- механізми захисту інформації, перевірки введених даних та управління сесіями користувачів;
- інтеграцію SDK Jitsi Meet як основний модуль для організації відеоконференцій.

Основні функціональні модулі (авторизація, створення конференцій, поштові сервіси, система сповіщень та API) реалізовані у вигляді незалежних компонентів, що значно спрощує подальший розвиток та обслуговування системи.

Тестування підтвердило високу стабільність, завершеність функціоналу та готовність системи до експлуатації. Інтерфейси веб-додатку та мобільного застосунку характеризуються швидкістю роботи, зручністю використання та

низьким рівнем використання апаратних ресурсів.

Практичне застосування застосунку передбачає організацію онлайн-заходів в освітніх закладах та бізнес-організаціях, внутрішню комунікацію в командних проектах, а також розвиток рішення у комерційний продукт із додатковими можливостями і гнучким налаштуванням під різні потреби користувачів.

ВИСНОВКИ

У відповідності до мети дипломної роботи було виконано наступне:

1. Досліджено існуючі рішення для проведення відеоконференцій (Zoom, Microsoft Teams, Google Meet, Cisco Webex), виявлено основні їх недоліки, такі як обмежений безкоштовний функціонал, закритий вихідний код, обмежена кастомізація та перевантаженість інтерфейсу, що стало підґрунтям для створення власної системи.

2. Вивчено сучасні технології та інструменти для реалізації веб-застосунків і мобільних додатків, зокрема мови програмування PHP та Dart, фреймворки CodeIgniter і Flutter, а також технології REST API і баз даних (MySQL).

3. Спроєктовано архітектуру програмного рішення відповідно до патерну MVC, що забезпечило структурованість, логічність і масштабованість системи.

4. Розроблено повноцінний веб-застосунок з адміністративною панеллю, системою авторизації та модулем управління відеоконференціями, а також мобільний додаток із застосуванням Flutter для платформ Android та iOS. Реалізовано REST API для зв’язку клієнт-сервер та інтегровано Jitsi Meet SDK для організації відеозв’язку. Проведено багаторівневе тестування, що підтвердило стабільність і функціональну готовність системи до використання.

5. Результати дипломної роботи мають практичну цінність і можуть бути застосовані для онлайн-нарад в освітніх або комерційних установах, як внутрішній інструмент для команд, або як основа для створення SaaS-рішення з подальшою комерціалізацією.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. CodeIgniter 4 User Guide. URL: https://codeigniter.com/user_guide (дата звернення: 17.04.2025).

2. Flutter Documentation. URL: https://flutter.dev/docs (дата звернення: 12.05.2025).

3. Jitsi Meet Handbook. URL: https://jitsi.org/user-guide (дата звернення: 07.03.2025).

4. Tan W., Lee L. Security and Privacy Issues in Video Conferencing Platforms: A Review // ACM Computing Surveys. 2022. Vol. 55, No. 4. Article 72. DOI: 10.1145/3507919.

5. Huang Y., Huang R. Managing Large-Scale Video Conferencing Systems: Architecture and Best Practices // IEEE Communications Magazine. – 2021. – Vol. 59, No. 6. – С. 92–98. – DOI: 10.1109/MCOM.001.2000609.

6. Скрипник І. В. Системи відеоконференцзв’язку в сучасному освітньому процесі // Вісник ЖДТУ. Серія: Технічні науки. – 2021. – № 1(97). – С. 112–117.

7. Мельник А. В., Гуменюк С. П. Використання технологій відеоконференцзв’язку в дистанційному навчанні // Інформаційні технології і засоби навчання. – 2020. – № 3(77). – С. 45–52.

8. Бодрук О. І. Засоби відеоконференцзв’язку: переваги, недоліки, безпека використання // Науковий вісник НЛТУ України. – 2020. – № 30.3. – С. 144–149.

9. RESTful API Design. URL: https://restfulapi.net (дата звернення: 10.04.2025).

10. PHP: Hypertext Preprocessor Documentation. URL: https://www.php.net/docs.php (дата звернення: 02.03.2025).

11. MySQL 8.0 Reference Manual. URL: https://dev.mysql.com/doc (дата звернення: 28.04.2025).

12. Android Developers Guide. URL: https://developer.android.com/docs (дата звернення: 16.04.2025).

13. Apple Developer Documentation. URL: https://developer.apple.com/documentation (дата звернення: 28.04.2025).

Декларація

академічної доброчесності

здобувача ступеня вищої освіти ЗНУ

Я, Петренко Олександр Олександрович, студент 4 курсу, форми навчання денної, Інженерного навчально-наукового інституту, спеціальність 121 Інженерія програмного забезпечення, адреса електронної пошти se121-11@stu.zsea.edu.ua, — підтверджую, що написана мною кваліфікаційна робота на тему «**Система управління та проведення відеоконференцій**» відповідає вимогам академічної доброчесності та не містить порушень, що визначені у ст.42 Закону України «Про освіту», зі змістом яких ознайомлений.

- заявляю, що надана мною для перевірки електронна версія роботи є ідентичною її друкованій версії;

- згоден на перевірку моєї роботи на відповідність критеріям академічної доброчесності у будь-який спосіб, у тому числі за допомогою інтернет- системи, а також на архівування моєї роботи в базі даних цієї системи.

Дата 01.06.2025 Підпис Петренко Олександр Олександрович
(студент)

Дата 01.06.2025 Підпис Міхайлуца Олена Миколаївна
(науковий керівник)