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

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

«На правах рукопису»
УДК 004.415.2

«До захисту допущено»
В.о. завідувача кафедри
_____ Едуард ЖАРІКОВ
«___»_____2021 р.

Магістерська дисертація

на здобуття ступеня магістра

за освітньо-професійною програмою «Інженерія програмного забезпечення інформаційних систем»

зі спеціальності 121 «Інженерія програмного забезпечення»
на тему: «Архітектура програмного забезпечення для підтримки ефективної комунікації між учасниками навчального процесу»

Виконав:
студент ІІ курсу, групи ІТ-04мп
Догмош Амір Важіхович _____

Керівник:
Старший викладач
Головченко Максим Миколайович _____

Рецензент:
доцент кафедри ІСТ, к.т.н.,
Жураковська Оксана Сергіївна _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2021 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Рівень вищої освіти – другий (магістерський)

Спеціальність – 121 «Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення інформаційних систем»

ЗАТВЕРДЖУЮ

В.о. завідувача кафедри

_____ Едуард ЖАРІКОВ

«___»_____2021р.

ЗАВДАННЯ

на магістерську дисертацію студенту

Догмошу Аміру Важіховичу

1. Тема дисертації «Архітектура програмного забезпечення для підтримки ефективної комунікації між учасниками навчального процесу», науковий керівник дисертації Головченко Максим Миколайович, старший викладач, затверджені наказом по університету від «27» жовтня 2021 р. № 3587-с
2. Термін подання студентом дисертації «6» грудня 2021 р.
3. Об’єкт дослідження – архітектура програмного забезпечення для підтримки ефективної комунікації між учасниками навчального процесу.
4. Предмет дослідження – технології програмного забезпечення для підтримки комунікації у навчальному процесі, які б забезпечили належну ефективність.
5. Перелік завдань, які потрібно розробити – аналіз існуючих рішень; побудова архітектури для програмного забезпечення; реалізація програмного забезпечення; оцінка ефективності та тестування запропонованого рішення.
6. Орієнтовний перелік графічного (ілюстративного) матеріалу – 2 плакати.
7. Орієнтовний перелік публікацій – одна публікація.

8. Консультанти розділів дисертації

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

9. Дата видачі завдання «30» вересня 2020 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання	Примітка
1	Аналіз предметної області	01.09.2021	
2	Пошук та аналіз існуючих рішень з виділенням переваг та недоліків	07.09.2021	
3	Постановка задачі та визначення вимог до програмного забезпечення	18.09.2021	
4	Дослідження та порівняння актуальних технологій та визначення типу ПЗ	02.10.2021	
5	Вибір та обґрунтування засобів програмної реалізації, які будуть використовуватись для вирішення поставленого завдання	12.10.2021	
6	Побудова архітектури програмного забезпечення	20.10.2021	
7	Розробка програмного забезпечення	15.11.2021	
8	Оформлення пояснювальної записки	21.11.2021	
9	Подання дисертації на попередній захист	22.11.2021	
10	Подання дисертації на захист	06.12.2021	

Студент Амір ДОГМОШ

Науковий керівник Максим ГОЛОВЧЕНКО

РЕФЕРАТ

Розмір пояснювальної записки – 139 аркушів, містить 42 ілюстрацій, 23 таблиці, 4 додатки.

Актуальність теми. У роботі розглянуто проблему підтримки комунікації між учасниками навчального процесу, показано основні особливості існуючих програмних засобів, що використовуються у навчальному процесі та наведено їх переваги та недоліки. Виявлено потребу в удосконаленні програмних засобів.

Мета дослідження. Метою є побудова архітектури програмного забезпечення для підвищення ефективності комунікації між учасниками навчального процесу.

Для реалізації поставленої мети були сформульовані **наступні завдання**:

– проаналізувати аналоги, визначити їх основні переваги та недоліки, а також причини, які сповільнюють їх роботу;

– визначити технології для побудови архітектури програмного забезпечення;

– визначити загальну концепцію архітектури програмного забезпечення та її основні модулі;

– побудувати архітектурне рішення та розробити програмне забезпечення для підтримки ефективної комунікації між учасниками навчального процесу;

– оптимізувати роботу програмного забезпечення та забезпечити його належну швидкодію;

– провести оцінку ефективності роботи розробленого програмного забезпечення.

Об’єкт дослідження – архітектура програмного забезпечення для підтримки ефективної комунікації між учасниками навчального процесу.

Предмет дослідження – технології програмного забезпечення для підтримки комунікації у навчальному процесі, які б забезпечили належну ефективність.

Наукова новизна. Наукова новизна полягає в удосконаленні рівня бізнес логіки багаторівневої архітектури програмного забезпечення, що полягає в прискоренні процесу поширення інформації для підтримки ефективної комунікації шляхом розширення функціональних можливостей цього рівня.

Практичне значення визначається тим, що запропоновано програмне забезпечення на базі багаторівневої архітектури, яке дасть змогу зручно для учасників навчального процесу вести облік, підтримувати комунікацію навчального процесу та отримувати своєчасні сповіщення про події навчання.

Зв'язок з науковими програмами, планами, темами. Робота виконувалась на кафедрі інформатики та програмної інженерії Національного технічного університету України "Київський політехнічний інститут імені Ігоря Сікорського".

Апробація. Наукові положення дисертації пройшли апробацію на Першій Всеукраїнській науково-практичній конференції молодих вчених та студентів «Інженерія програмного забезпечення і передові інформаційні технології» (SoftTech-2021) – м. Київ, 2021р.

Публікації. Наукові положення дисертації опубліковані в:

Догмош А.В., Головченко М.М. Архітектурне рішення для підтримки ефективної комунікації між учасниками навчального процесу. // Матеріали Першої Всеукраїнської науково-практичної конференції молодих вчених та студентів «Інженерія програмного забезпечення і передові інформаційні технології» (SoftTech-2021) – м. Київ: НТУУ "КПІ ім. Ігоря Сікорського", 22-26 листопада 2021 р.

Ключові слова: ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, БАГАТОРІВНЕВА АРХІТЕКТУРА, ПРИКЛАДНИЙ ПРОГРАМНИЙ ІНТЕРФЕЙС.

ABSTRACT

Explanatory note size – 139 pages, contains 42 illustrations, 23 tables, 4 applications.

The relevance of the topic. Examines the problem of maintaining communication between participants in the teaching-learning process, shows the main features of existing software used in the teaching-learning process and presents their advantages and disadvantages. The need to improve software has been identified.

The purpose of the study. The main target is to build a software architecture to increase the effectiveness of communication between participants in the teaching-learning process.

To achieve this goal, the **following tasks** were set:

- analysis of existing solutions, identify their main advantages and disadvantages, as well as the reasons that slow down their performance;
- identify technologies for building software architecture;
- define the general concept of software architecture and its main modules;
- build architectural solutions and develop software to support effective communication between participants in the teaching-learning process;
- optimize the operation of the software and ensure its proper performance;
- evaluate the effectiveness of the developed software.

The object of research: software architecture to support effective communication between participants in the teaching-learning process.

The subject of research: software technologies to support communication in the teaching-learning process, which would ensure proper efficiency.

Scientific novelty. Scientific novelty is to improve the level of business logic of multitier software architecture, which is to accelerate the process of dissemination of information to maintain effective communication by expanding the functionality of this level.

The practical significance is determined by the fact that the proposed software based on multitier architecture, which will allow convenient for participants in the teaching-learning process to keep records, maintain communication of the educational process and receive timely notifications of learning events.

Relationship with working with scientific programs, plans, topics. Work was performed at the Department of Informatics and Software Engineering of the National Technical University of Ukraine «Kyiv Polytechnic Institute. Igor Sikorsky».

Approbation. The scientific provisions of the dissertation were tested at the First All-Ukrainian scientific-practical conference of young scientists and students "Software Engineering and Advanced Information Technologies" (SoftTech-2021) - Kyiv, 2021.

Publications. The scientific provisions of the dissertation are published in: Dohmosh A.V., Holovchenko M.M., Architectural solution to support effective communication between participants in the teaching-learning process. // Proceedings of the First All-Ukrainian Scientific and Practical Conference of Young Scientists and Students "Software Engineering and Advanced Information Technologies" (SoftTech-2021) - Kyiv: NTUU "Kyiv Polytechnic Institute. Igor Sikorsky", November 22-26, 2021.

Keywords: SOFTWARE, MULTITIER ARCHITECTURE, APPLICATION PROGRAMMING INTERFACE.

ЗМІСТ

СПИСОК УМОВНИХ СКОРОЧЕНЬ

ACID – (Atomicity, Consistency, Isolation, Durability) – набір властивостей, що гарантують надійну роботу транзакцій бази даних: атомарність, узгодженість, ізольованість, довговічність.

AOP – (Aspect-oriented programming) – парадигма програмування, яка дозволяє виокремити перехресну функціональність.

API – (Application programming interface) – прикладний програмний інтерфейс.

DOM – (Document Object Model) – специфікація прикладного програмного інтерфейсу для роботи зі структурованими документами.

HTML – (HyperText Markup Language) – це мова тегів, засобами якої здійснюється розмічання вебсторінок для мережі Інтернет.

HTTP – (HyperText Transfer Protocol) – це формат протоколу передачі даних в Глобальної Мережі, в основі якого лежить технологія відносин клієнт-сервер.

HTTPS – (HyperText Transfer Protocol Secure) – це розширення протоколу HTTP, необхідне для підтримки шифрування з метою підвищення рівня безпеки даних.

JDBC – (Java Database Connectivity) – це програмний компонент, який дозволяє додаткам Java взаємодіяти з базами даних.

JS – (JavaScript) - мова програмування, що дозволяє зробити веб-сторінку інтерактивною, тобто такою що реагує на дії користувача.

JSON – (JavaScript Object Notation) – текстовий формат обміну даними.

JVM – (Java Virtual Machine) – віртуальна машина для виконання байт-коду Java.

LMS – (Learning management system) – система управління навчальною діяльністю, яка використовується для розробки, управління та поширення навчальних онлайн-матеріалів із забезпеченням спільного доступу.

RDBMS – (Relational Database Management System) – система управління базами даних.

REST – (Representational State Transfer) – архітектурний підхід до побудови веб сервісів.

SQL – (Structured Query Language) – мова програмування структурованих запитів, яка використовується у якості ефективного засобу зберігання даних, пошуку їх частин, отримання з БД та видалення.

SSL – (Secure Sockets Layer) – це цифровий сертифікат, який підтверджує справжність веб-сайту та зашифровує інформацію, що надсилається на сервер.

TDLib – (Telegram Database Library) – це кросплатформений Telegram-клієнт для розробників, призначений для створення застосунків для роботи на платформі Telegram.

WebRTC – (Web real-time communications) – це технологія, яка дозволяє передавати аудіо і відео потокові дані між браузерами і мобільними додатками.

XML – (Extensible Markup Language) - стандарт побудови мов розмітки ієрархічно структурованих даних для обміну між різними інтернет застосунками.

БД – (База даних) – це організована структура, призначена для зберігання, зміни та обробки взаємопов’язаної інформації, переважно великих розмірів.

СКБД – (Система керування базами даних) – це комплекс програмних і мовних засобів, необхідних для створення баз даних, підтримання їх в актуальному стані та організації пошуку в них необхідної інформації.

ПЗ – (Програмне забезпечення) – це одна із складових інформаційної системи – сукупність програм і програмних документів, необхідних для експлуатації цих програм.

ПК – (Персональний комп’ютер) обчислювальна машина, яка в сучасному світі використовується для роботи, пошуку інформації, читання та написання книг, малювання, ігор, прослуховування музики і т. д.

ОС – (Операційна система) – це набір комп’ютерних програм, що забезпечують працездатність комп'ютера та його зв’язок (інтерфейс) із користувачем.

ВСТУП

На сьогоднішній день освіта у навчальних закладах базується на використанні сучасних технологій, які досить стрімко розвиваються та утворюють нову форму комунікації між учасниками навчального процесу. Саме інформаційно-комунікаційні технології стають необхідним елементом в освітньому процесі. Можливості викладання лекційного матеріалу та сповіщення про навчальні події вже не обмежуються застарілими традиційними засобами, а поширюються в необмеженому, віртуальному та технологічному світі.

Особливість освіти нового типу спрямована на слідування інноваціям, всебічному вивченні та ефективному застосуванні нових форм спілкування і змісту навчальних програм. Впровадження цифрових технологій у теорію та практику педагогіки допомагає підвищити ефективність навчального процесу. Виникає активне впровадження засобів програмного забезпечення в освітній процес на всіх ступенях навчання.

Навчальні заклади намагаються автоматизувати науково-освітній процес шляхом використання різноманітних онлайн застосунків, які дозволяють полегшити комунікацію між здобувачем та закладом освіти. Головна особливість таких застосунків – це прискорення процесу поширення інформації. Також їх використання у навчальному процесі дає змогу полегшити викладання лекційного матеріалу та поширення навчальних мультимедійних матеріалів. Кожен застосунок використовує власні програмні інструменти та складові, які налаштовані на вирішення конкретної задачі та мають обмежену функціональність. Таким чином число застосунків для автоматизації навчального процесу може сягати великої кількості, а викладачі в свою чергу можуть застосовувати різноманітні програмні засоби для вирішення одних і тих же задач у зв’язку із тим, що не існує стандартних підходів до використання певних програмних засобів для комунікації. Виникає відсутність злагодженої роботи, оскільки студенти можуть несвоєчасно реагувати на отримані сповіщення, а викладачі в свою чергу змушені дублювати інформацію на

різноманітних платформах. Більш того, використання різноманітних ресурсів є не досить зручним і доволі часто інформація може бути неактуальною і не відповідати одна одній у різних джерелах у зв’язку з її застарілістю.

Тому на сьогоднішній час для навчальних закладів ключовим і актуальним завданням є підвищення швидкодії сповіщення студентів про навчальний процес, а саме зменшення кількості використання різноманітних застосунків за рахунок створення універсального програмного забезпечення, яке дасть змогу інтегрувати сучасні програмні засоби використати основні їх переваги для своєчасного сповіщення про навчальні заходи та оновлення складових елементів навчального процесу.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис предметної області

У наш час є багато інструментів, які дозволяють підтримувати комунікацію між здобувачем освіти та освітнім закладом. Такі інструменти допомагають ефективно вирішувати труднощі з проведенням комунікації між учасниками навчального процесу, але також вони можуть бути у деякій мірі складними у використанні та не досить зручними.

Завдяки інтеграції сучасних програмних засобів у навчальний процес забезпечується постійний зв’язок між його учасниками та будь-яка інформація може інтенсивно поширюватися між ними, хоча фізично вони можуть знаходитись на відстані один від одного за тисячі кілометрів. Тобто перевагою такої інтеграції є гнучкість, що дозволяє учасникам навчального процесу бути присутнім на занятті незалежно від місця проживання, стану здоров'я. Як показали останні роки, стан здоров'я суттєво впливає на можливість відвідування навчального закладу, тому щоб не наражати себе й оточення на небезпеку, можна перебувати вдома на лікарняному, стежити за своїм здоров'ям і продовжувати навчання.

До основних елементів традиційної освіти можна віднести семінари, лекції, практичні та лабораторні заняття, іспити тощо. З використанням програмних засобів особливість форми освіти набуває іншого характеру. Спілкування між учасниками навчального процесу не є безпосереднім у разі проведення онлайн лекцій, на відміну від типових лекцій у класі. Завдяки застосуванню сучасних інформаційних технологій лекції можуть бути наповнені додатковим візуальним та виразним контентом, завдяки гіпертексту або мультимедійним технологіям.

Лекційний матеріал можна слухати в будь-який момент і в будь-якому місці. Саме завдяки сучасним програмним засобам здобувачу освіти не потрібно робити конспектування та фізично зберігати навчальні матеріали.

Існує величезна кількість застосунків та сервісів для підтримки комунікації у навчанні, але переважна більшість із них мають суттєві недоліки пов’язані із перевантаженістю інформацією, відсутністю додаткових функціональних можливостей, неналежною оптимізацією, незручністю в користуванні та іншими факторами. Можна стверджувати, що більшість таких інструментів мають досить обмежену кількість можливостей, що призводить до того, що необхідно використовувати додаткові засоби програмного забезпечення. Таким чином виникає питання в належній ефективності комунікації, оскільки доводиться використовувати одночасно декілька застосунків, щоб мати актуальну інформацію щодо подій навчання. До того ж, певні програмні засоби мають застарілий дизайн користувацького інтерфейсу, що в свою чергу відштовхує від користування. Інші, маючи сучасний дизайн, відображають одночасно велику кількість інформації, що приводить до ускладнення перегляду.

В такому випадку виникає необхідність у реалізації програмного забезпечення, яке б мало сучасний дизайн, було зручним в користуванні та забезпечувало можливість підтримувати ефективну комунікацію між учасниками навчального процесу онлайн, що є одним із найголовніших критеріїв такого застосунку. Як правило, в такому програмному забезпеченні може розміщуватися велика кількість матеріалів текстового та відео формату, а також значна кількість посилань на інші ресурси та онлайн сервіси. Але з використанням багатьох онлайн сервісів процес сповіщення про навчальні події усіх учасників може бути незручним. Потрібно враховувати, щоб всі студенти відвідали певний ресурс для отримання термінового повідомлення або важливої інформації. Таким чином, використання і дублювання інформації на різних платформах, що використовуються у процесі навчання, буде спричиняти велику кількість неоднозначності та дублювання інформації для сповіщення усіх учасників навчального процесу. Тому можна сказати, що використання програмних засобів для підтримки комунікації у навчальному процесі не є в повному обсязі ефективним і потребує додаткового дослідження для усунення існуючих недоліків.

Для реалізації ефективної комунікації із використанням сучасного програмного забезпечення необхідно провести огляд вже існуючих аналогів та оцінити їх за такими критеріями як:

– підтримка передачі інформації у текстовому, графічному, звуковому та відео форматі;
– наявність групового та приватного чату;
– проведення відео-конференції;
– можливість онлайн демонстрації навчального матеріалу в мультимедійній формі;
– ідентифікація учасників навчального процесу;
– наявність форуму для обговорення навчальних заходів;
– наявність дошки оголошень;
– можливість залишати коментарі до оголошень;
– наявність опитування.

1.2 Огляд наявних рішень

Розглянемо існуючі аналоги, які є найбільш поширеними та використовуються на даний момент для підтримки комунікації у навчальних закладах, та приведемо їх переваги та недоліки за вищезазначеними критеріями.

1.2.1 Google Classroom

Google Classroom – онлайн платформа компанії Google (рис. 1.1), яка було розроблена спеціально для студентів та викладачів з метою підтримки комунікації, поширенні навчального матеріалу та організації контрольних завдань [3]. Сам застосунок складається із навчальних класів, в яких проводиться основна взаємодія між учасниками навчального процесу. Створення та розповсюдження завдань здійснюється через Google диск. Студенти можуть приєднатися до класу завдяки спеціальному коду для доступу, який може надати викладач.

Google Classroom інтегрується з календарями Google студентів і викладачів. Також студенти мають можливість надіслати роботу для оцінювання викладачу у класі, в якому створюється на Google диску окрема папка.

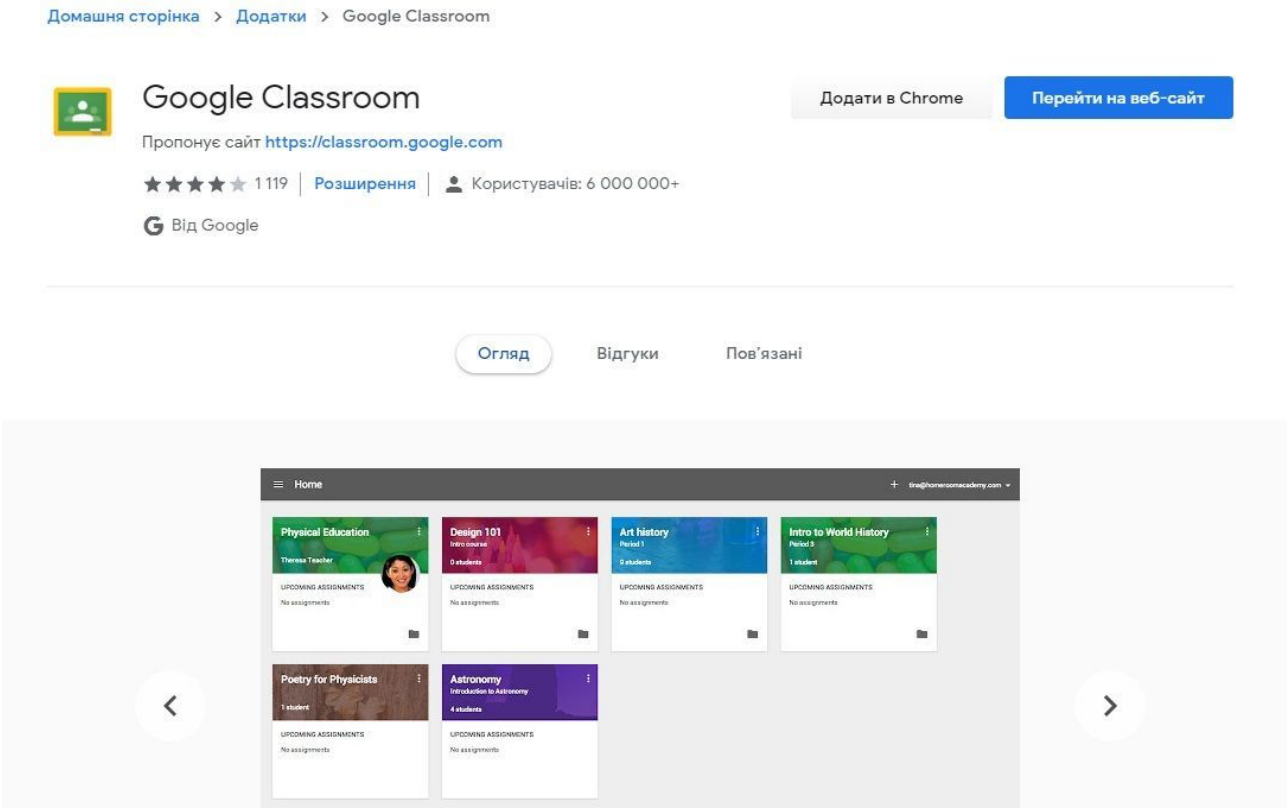


Рисунок 1.1 – Відображення навчальних курсів “Google Classroom”

З точки зору підтримки комунікації Google Classroom дає змогу викладачам спілкуватися зі студентами, використовуючи пошту Gmail. Даний застосунок дозволяє зручно надсилати інформацію як і одному учаснику курсу, так і цілій групі через пошту (рис. 1.2). Комунікація через Gmail дозволяє викладачам створювати оголошення (рис. 1.3), ставити питання до студентів та надсилати завдання в кожному класі (рис. 1.4). Слід зауважити, що Google Classroom надає можливість обмежувати виконання завдань у часі. Також ця платформа підтримує коментування завдань у класі, що дає змогу проводити більш детальні обговорення.

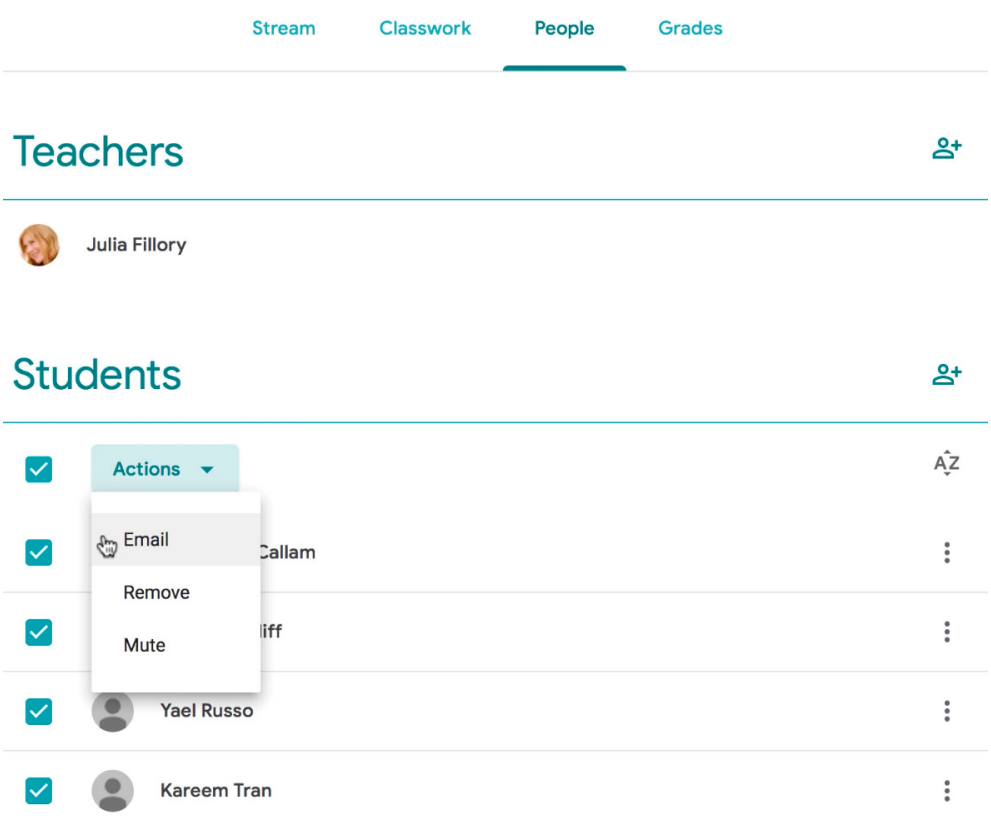


Рисунок 1.2 – Прилад надсилання електронної пошти всьому класу

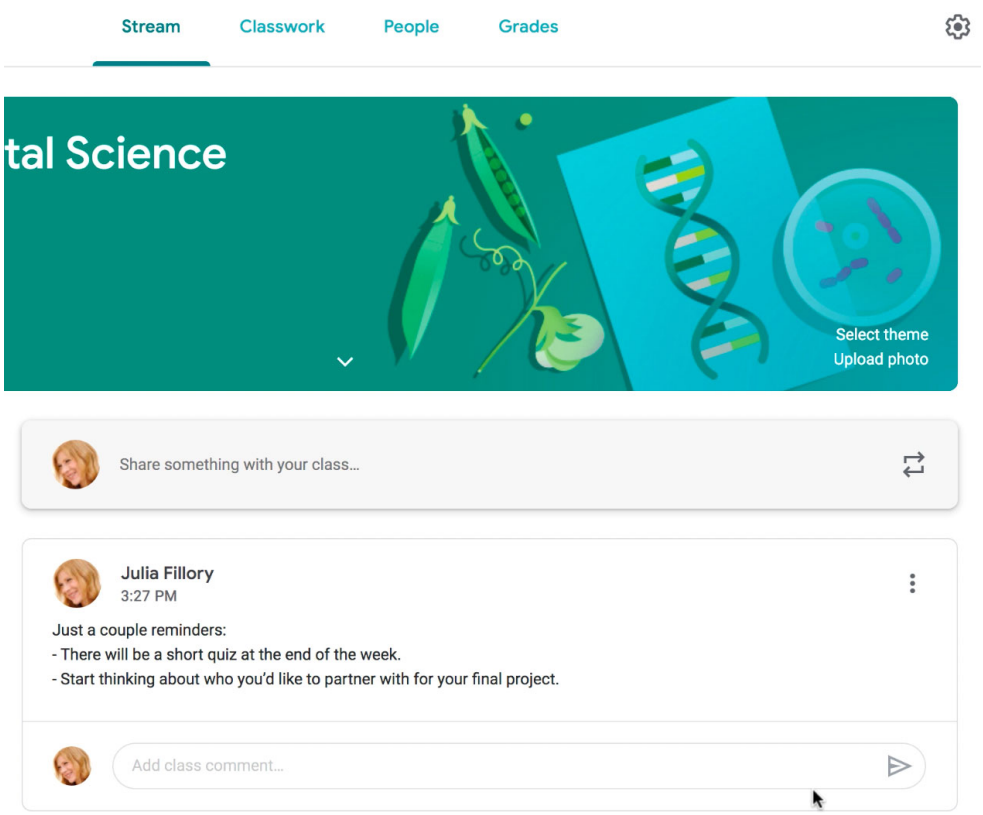


Рисунок 1.3 – Прилад опублікування оголошення всьому класу

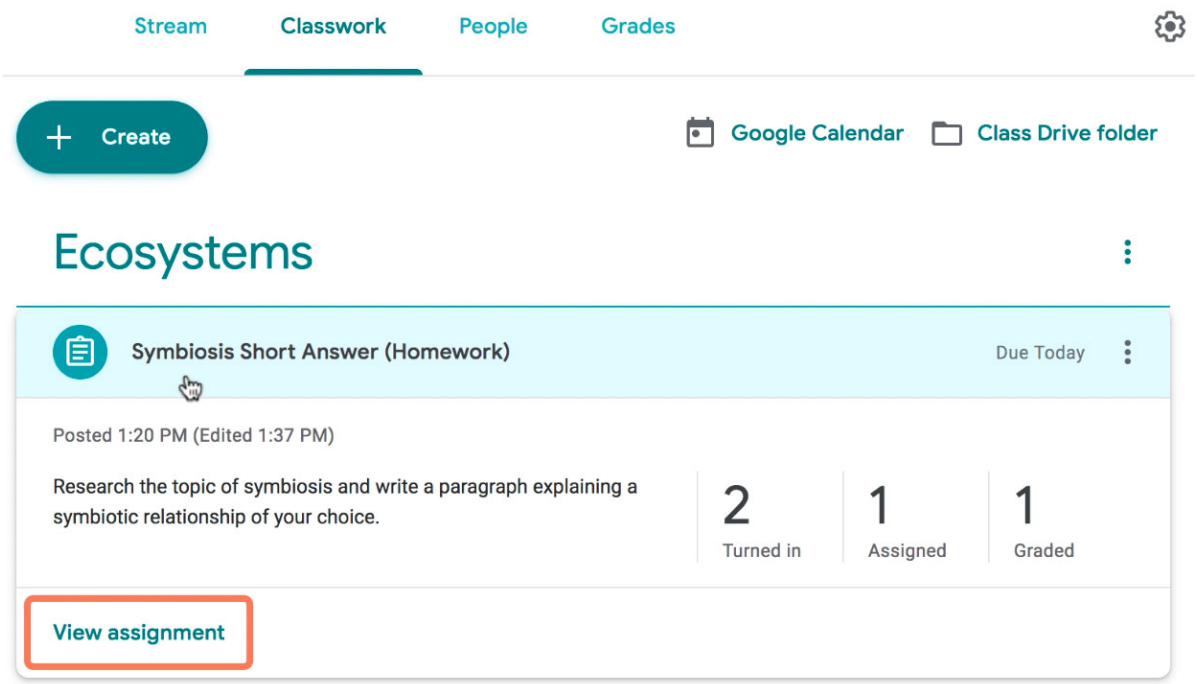


Рисунок 1.4 – Приклад створеного завдання

Наведемо основні переваги згідно тих критеріїв, які є доцільними для ефективної підтримки комунікації, що варто враховувати при роботі з Google Classroom:

– наявність дошки оголошень, що дозволяє публікувати інформацію до навчального курсу або новини про навчальні заходи;

– коментування оголошень, що дає змогу використовувати коментарі як деякий форум для спілкування між учасниками курсу для обговорення якихось питань;

– підтримка коментарів до завдань. Застосунок дозволяє створити і поширити копії, створених викладачем завдань, серед усіх студентів, які знаходяться у класі. Більш того, викладач має змогу слідкувати за тим, як виконуються завдання, а також залишати коментарі до перевірених робіт та надісланих студентами завдань;

– створення та модифікація документу у реальному часі. Можна ділитися документом між всіма учасниками курсу та виконувати його модифікацію у реальному часі, що дасть можливість спостерігати за цим кожному учаснику.

– інтеграція застосунку з гугл-диском. Коли створюється навчальний курс у застосунку, то папка з матеріалами, що відповідають даному курсу, автоматично створюється у всіх учасників класу на гугл-диску.

Нижче визначимо недоліки програмного застосунку, беручи до уваги відповідні критерії для підтримки комунікації:

– відсутність можливості запису голосових та відео повідомлень. У зв'язку з неможливістю транслювати та зберігати свої уроки в Google Classroom, викладачі змушені використовувати інші застосунки та надавати посилання на них, що не завжди є зручним;

– відсутність відео-конференції та демонстрації навчального мультимедійного матеріалу;

– можливість виникнення проблеми з ідентифікацією учасників курсу. Викладач завжди просить перейменовувати себе так, щоб була можливість ідентифікувати учасника курсу. В свою чергу, студент потрібен заходити в додаткові налаштування облікового запису і для деяких виникає питання в якому місці це потрібно зробити. Часто виникає ситуація коли викладач змушений проводити інструктаж в якому місці потрібно змінити дані налаштування, що призводить до зайвої витрати часу;

– відсутність вбудованих тестів та опитувань. Натомість, для проходження опитувань чи тестів використовується Google Forms або інші сервіси, які не є вбудованими у сам веб-застосунок. Користувач перед відправленням виконаного тесту повинен завжди ідентифікувати себе шляхом вказання своєї поштової адреси та інших даних (прізвище, група і т.д.);

– відсутність приватного чату, не існує можливості зв'язатися з кимось безпосередньо, а не ділитися в групі. Більш того, може виникнути ситуація коли необхідно обговорити деталі поставленого завдання надаючи певні матеріали у вигляді знімків. Але ця можливість відсутня окрім коментарів до завдання, де можна лише відправити текстове повідомлення;

– відсутність традиційного форуму за певними темами та відкритого обговорення. Натомість, даний застосунок підтримує лише дошку оголошень, в якому не досить зручно проводити дискусії та обговорення.

1.2.2 “Електронний кампус НТУУ КПІ”

Система електронного кампусу КПІ (рис. 1.5) представляє собою інформаційний ресурс університету, яке неможливо було не взяти до уваги, оскільки воно є невід’ємною частиною навчального процесу та є досить поширеним серед викладацького складу. Система використовується як онлайн платформа для зосередження інформаційної діяльності учасників навчального процесу та працівників університету. Електронний кампус формує єдиний інформаційний ресурс, яке дає змогу зобразити загальний стан науково-освітнього процесу [4].

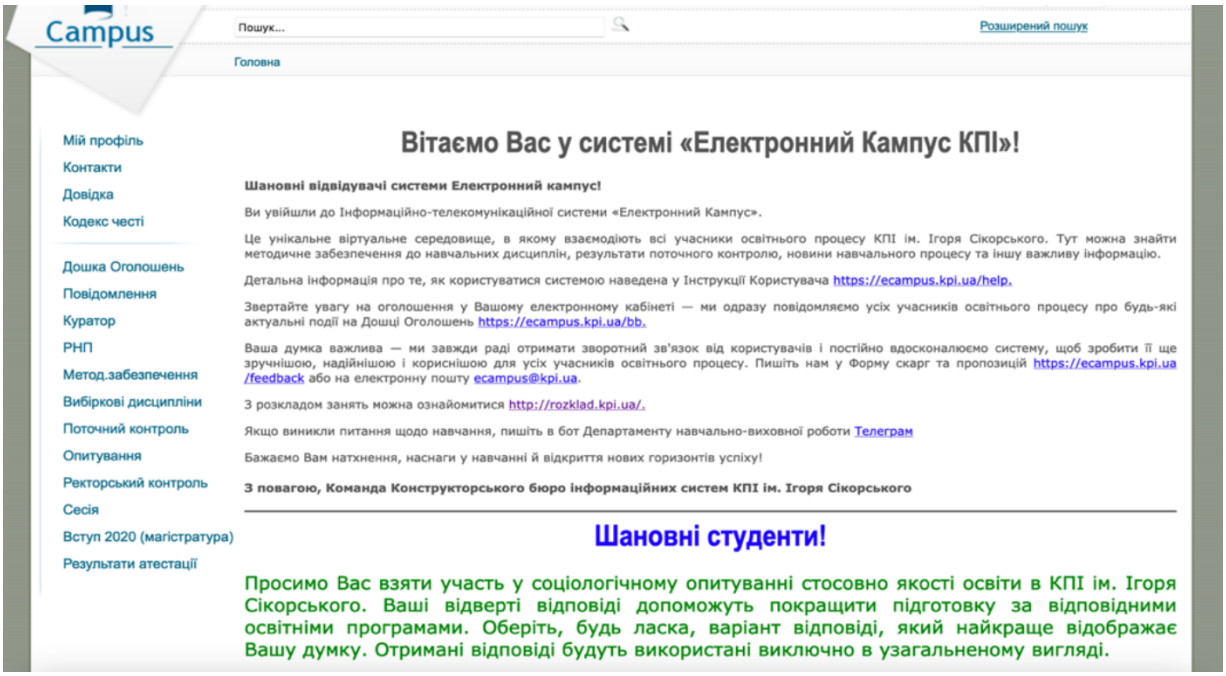


Рисунок 1.5 – Електронний кампус КПІ

Якщо розглядати розроблену платформу лише як засіб комунікації, то вона забезпечує наступні функції:

– розповсюдження інформації про поточні події та заходи навчального процесу;

– забезпечення зворотного зв’язку викладач-студент;

– організація єдиного інформаційного простору для університету;

– забезпечення багатобічної комунікації між учасниками навчального процесу.

Серед переваг комунікації, які варто віднести до даної платформи це змога чітко ідентифікувати студента для підтримання з ним подальшого зв’язку. Цей критерій забезпечується тим, що реєстрація даних студента відбувається не самими студентами, а співробітниками університету, і вже сам студент отримує від них необхідну інформацію для подальшої змоги користування своїм обліковим записом.

Система також підтримує комунікацію шляхом надсилання текстових повідомлень як персонально, так і певній групі осіб (рис 1.5). Наявність дошки оголошень (рис. 1.6) дає змогу розповсюджувати інформацію про новини університету та навчальні події, що є беззаперечним критерієм інформування учасників навчального процесу.

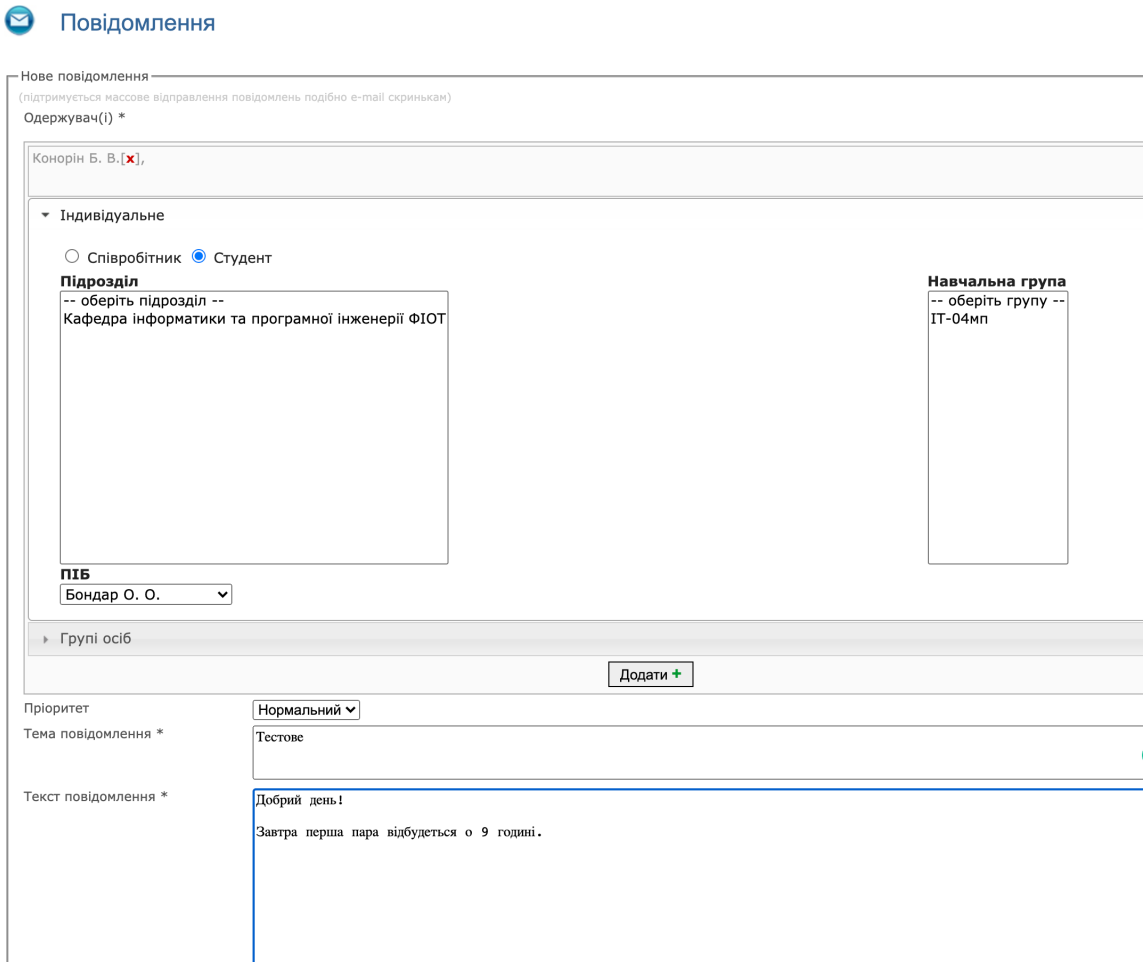


Рисунок 1.5 – Приклад надсилання повідомлення в електронному кампусі

Однак, програмним застосунком користується досить невелика кількість студентів через певну низку недоліків. Причиною цього є ряд суттєвих функціональних обмежень, що не дозволяють називати електронний кампус ефективним засобом комунікації. Серед можна назвати:

– відсутність форуму. Таке обмеження не може задовільнити студентів, оскільки у них завжди виникають питання з приводу навчального процесу на будь-яку тематику і саме форум може надати їм зручну відповідь на поставлені питання;

– відсутність відео-конференції та надання мультимедійного матеріалу у реальному часі. Можна вважати одним із основних недоліків цієї платформи, оскільки в період дистанційного навчання за останні роки саме відео-конференція почала відігравати важливу роль у забезпеченні ефективності у комунікації між учасниками навчального процесу;

– обмежена функціональність повідомлень. Окрім текстової інформації, в повідомленні неможливо прикріпити файлові документи, не говорячи вже про підтримку аудіо повідомлень.

– відсутність групового чату. Хоча завдяки повідомленням можна відправляти текст одночасно декілька особам, це не дає змогу спілкуватися усім учасникам навчального процесу одночасно між собою в одному чаті.

Ще одним недоліком, що містить даний ресурс, треба виділити й відсутність оновлення інформації. На рис. 1.6. можна побачити, що оголошення не є актуальними на сьогоднішній час, оскільки відстають на цілий рік. Таким чином, у зв’язку з вищезазначеними недоліками, кампус не можна вважати ефективним засобом для комунікації, тому що розповсюдження інформації відбувається на інших платформах, але ніяк не пов’язаних з електронним кампусом.

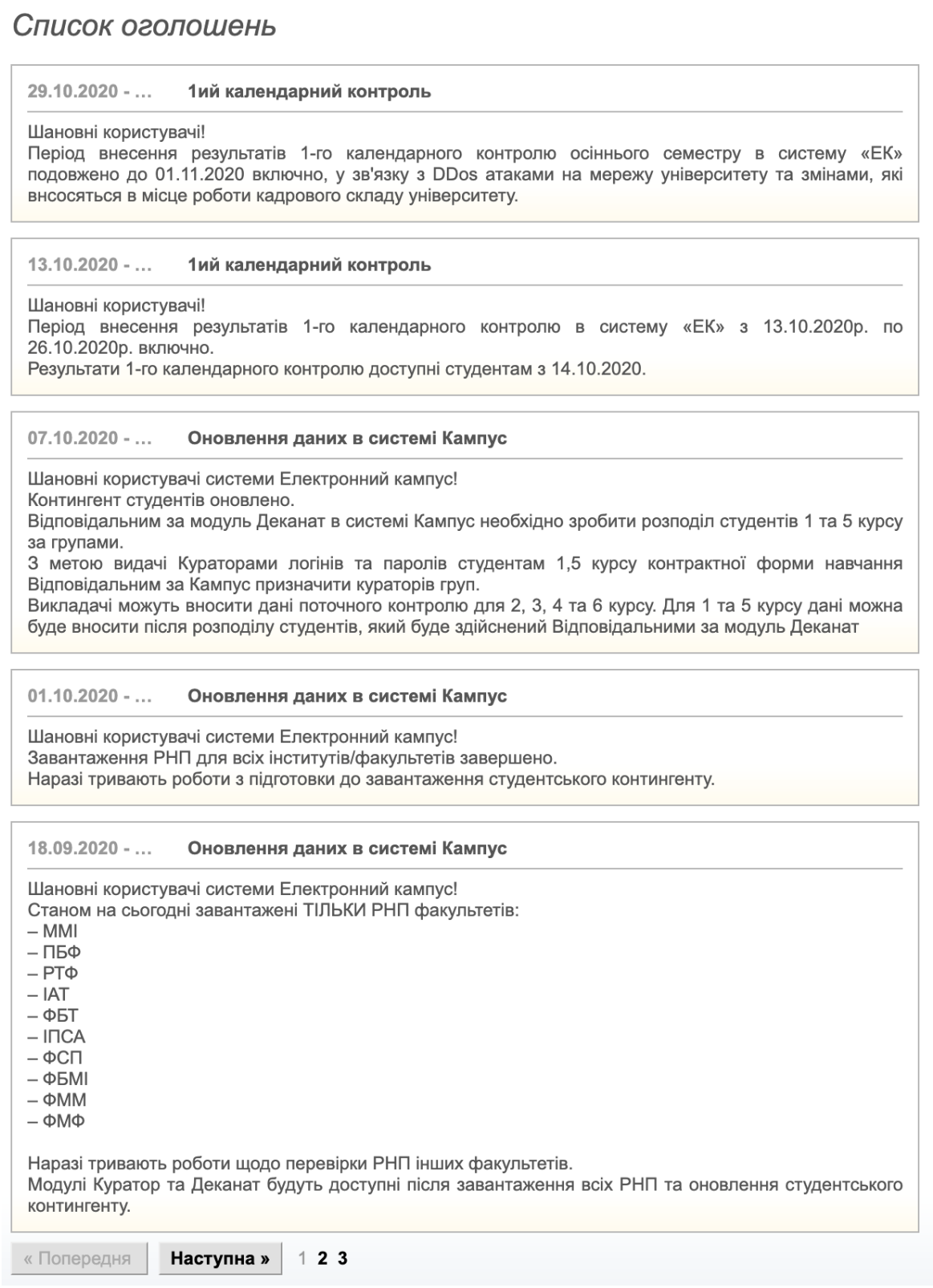


Рисунок 1.6 – Дошка оголошень електронного кампусу

Через вищезазначені особливості в структурі даного інформаційного ресурсу можна помітити, що є відмінність між описом його функціональних можливостей і тим, що знаходиться зараз на ресурсі.

1.2.3 Moodle

Moodle – онлайн платформа, яка допомагає створити повноцінне навчальне середовище шляхом створення власного веб-додатку, який може бути наповнений динамічними елементами. Платформа представляє собою інтуїтивно зрозуміле і сприятливе середовище для всіх рівнів системи освіти. З використанням системи Moodle можна надсилати електронні листи, працювати з PDF-файлами, переглядати онлайн-відео тощо. Більш того, Moodle має можливість включати ігри в навчальні курси (в Moodle це називається гейміфікацією). Ці ігри мають навчальний характер, вони допомагають залучити молодих студентів і заохочують їх до участі [5]. На рис. 1.7. можна побачити головну сторінку Moodle з основними її компонентами.

Moodle вважається ідеальним LMS [30] для вищої освіти, оскільки він забезпечує постійну комунікацію, що є ключовим моментом у дистанційній освіті. Ця система має досить широкий спектр функціональних можливостей та складових для підтримки комунікації, основними з яких є форуми (рис. 1.8), обмін повідомленнями, підтримка чату, наявність коментування та дописів в блозі, доступність для спілкування між учасниками класу за його межами. Таким же чином, Moodle дозволяє співпрацювати в команді – перераховані вище функції комунікації в ньому дозволяють студентам працювати разом, ділитися ідеями та ставити/відповідати на запитання у разі потреби. У навчальних закладах матеріали до курсу та додаткові ресурси так само важливі, як і лекції в класі. Також можна легко завантажувати та обмінюватися ресурсами, статтями, відео, зображеннями та всім іншим, що може знадобитися студентам для виконання курсової роботи та завдань.

У Moodle наявна система сторонніх плагінів, які дозволяють полегшити роботу викладачів або інших учасників навчального процесу. Існують безліч плагінів, що використовують бази даних, плани щодо навчальних заходів, організації програми навчання тощо. Застосовуючи різноманітні плагіни, Moodle набуває такої платформи, яка є повноцінною та розширюваною в плані функціональних можливостей.

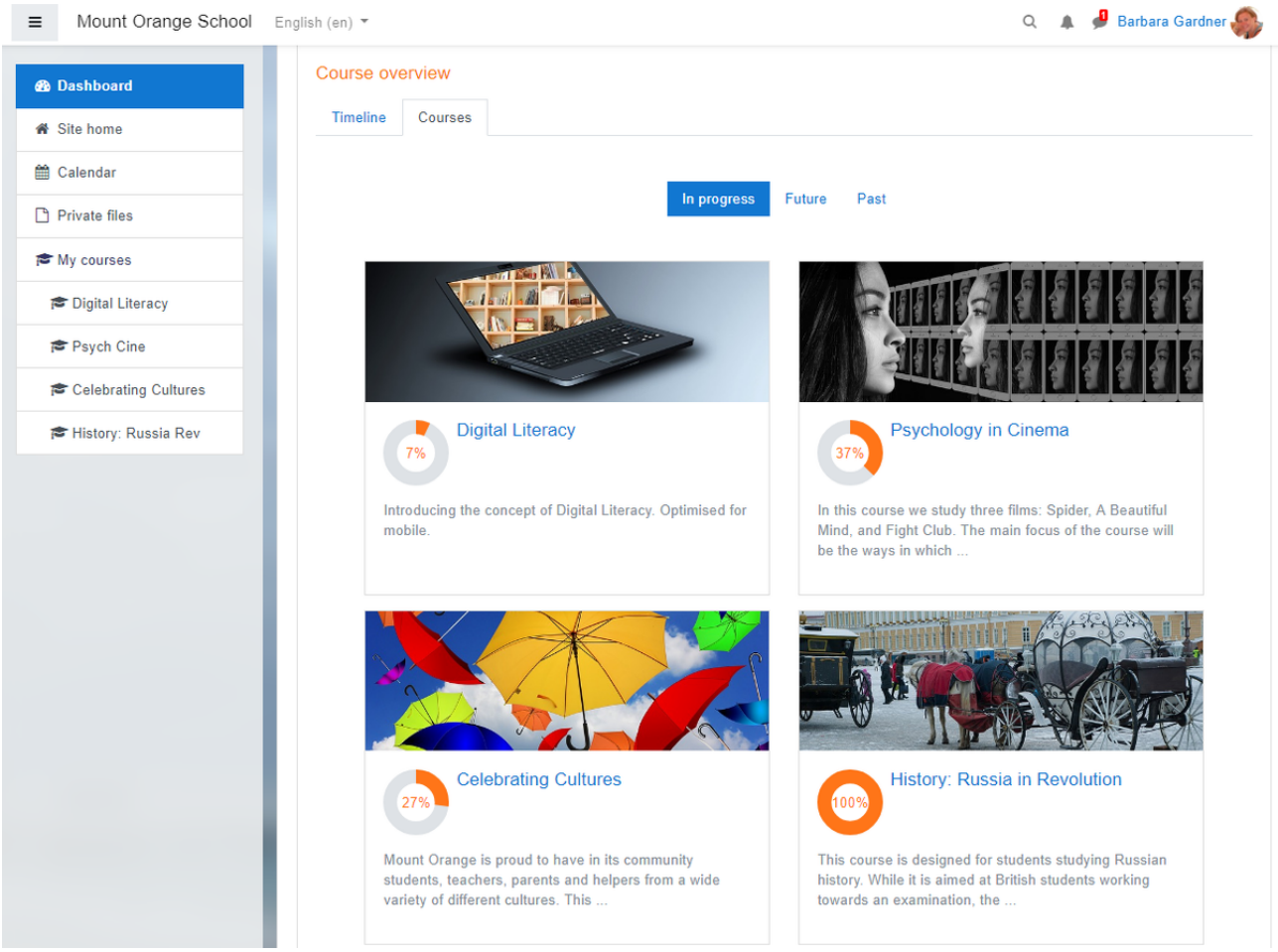


Рисунок 1.7 – Структура платформи Moodle

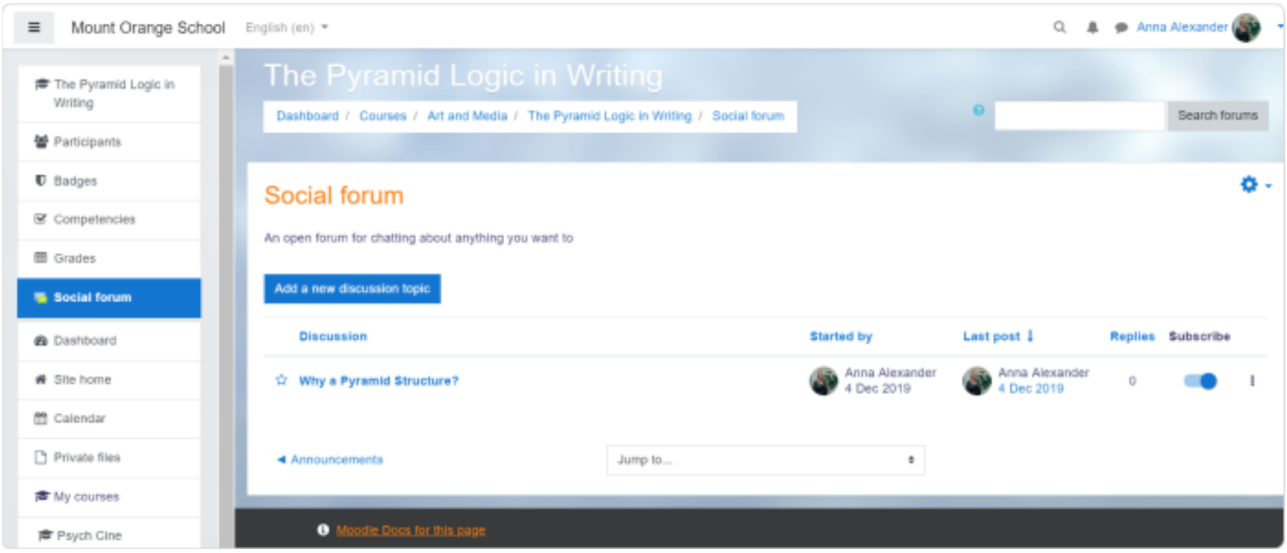


Рисунок 1.8 – Зображення соціального форуму у платформі Moodle

Якщо розглядати всі критерії, які були поставлені до визначення ефективності у комунікації учасників навчального процесу то можна сказати, що система Moodle задовольняє більшості таким критеріям. Але все ж, вона не є поширеною у закладах навчальної освіти. До основних недоліків, які

відштовхують студентів та викладачів від користування такої системи можна віднести:

– складність користування системою. Для того, щоб розібратися як працювати із системою необхідно детальніше ознайомитися з нею, використовуючи допоміжну літературу та інтернет-ресурси. Необхідно витратити додатковий час, щоб прочитати посібник для роботи із даною системою.

– витрачення ресурсів на розробку курсу. У разі необхідності розробки повноцінного курсу, що матиме матеріал лекцій, практичні заняття, матеріали для лабораторних робіт, електронні ресурси, контрольні та тестові опитування, розробка може зайняти досить довгий період часу – від декількох тижнів;

– відсутність аудіо повідомлень. Підтримка аудіо повідомлення може бути лише використано викладачем при публікації оголошення чи створенні завдання, але комунікації студентів між собою та з самим викладачем обмежується у використанні цієї функції. Більш того, для використання викладачем аудіо запису, необхідно завантажити додатковий плагін, який потребує допоміжних зусиль та часу на вивчення інструкції по налаштуванню;

– складність інтеграції із засобами відео конференції. У системі відсутня власна підтримка відео конференцій. Натомість, існує можливість використання додаткових плагінів, які дозволяють інтегрувати поширені програмні застосунки для проведення відео конференцій. Найбільш розповсюдженим застосунком для відео конференцій у системі Moodle є BigBlueButton, але повноцінна його робота може бути лише забезпечена у разі встановлення на окремому сервері, який повинен знаходитись окремо від того сервера, де встановлено Moodle. Але не всі навчальні заклади в змозі виділити певні ресурси для цього.

Таким чином, не дивлячись на величезну кількість функцій цієї системи, виникає складність у виділенні додаткових ресурсів та часу для розробки навчального курсу та створення його модулів та інтеграції додаткових плагінів.

1.2.4 Огляд додаткових застосунків

За останній період часу, в умовах дистанційного навчання, серед додаткових онлайн сервісів, які призначені для підтримки та покращення навчального процесу, у навчальних закладах також стали широко розповсюджені застосунки, які не обмежуються тільки для користування в навчальному процесі. Нижче будуть наведені найбільш розповсюджені програмні засоби.

Zoom або *Google Meet* – це аналогічні сервіси, які використовуються для проведення будь-яких відео-конференцій, а у випадку навчального процесу це онлайн-занять та веб-семінарів [6]. Для прикладу, приведемо інтерфейси програм Zoom та Google Meet, які зображені на рисунках 1.9 та 1.10 відповідно.

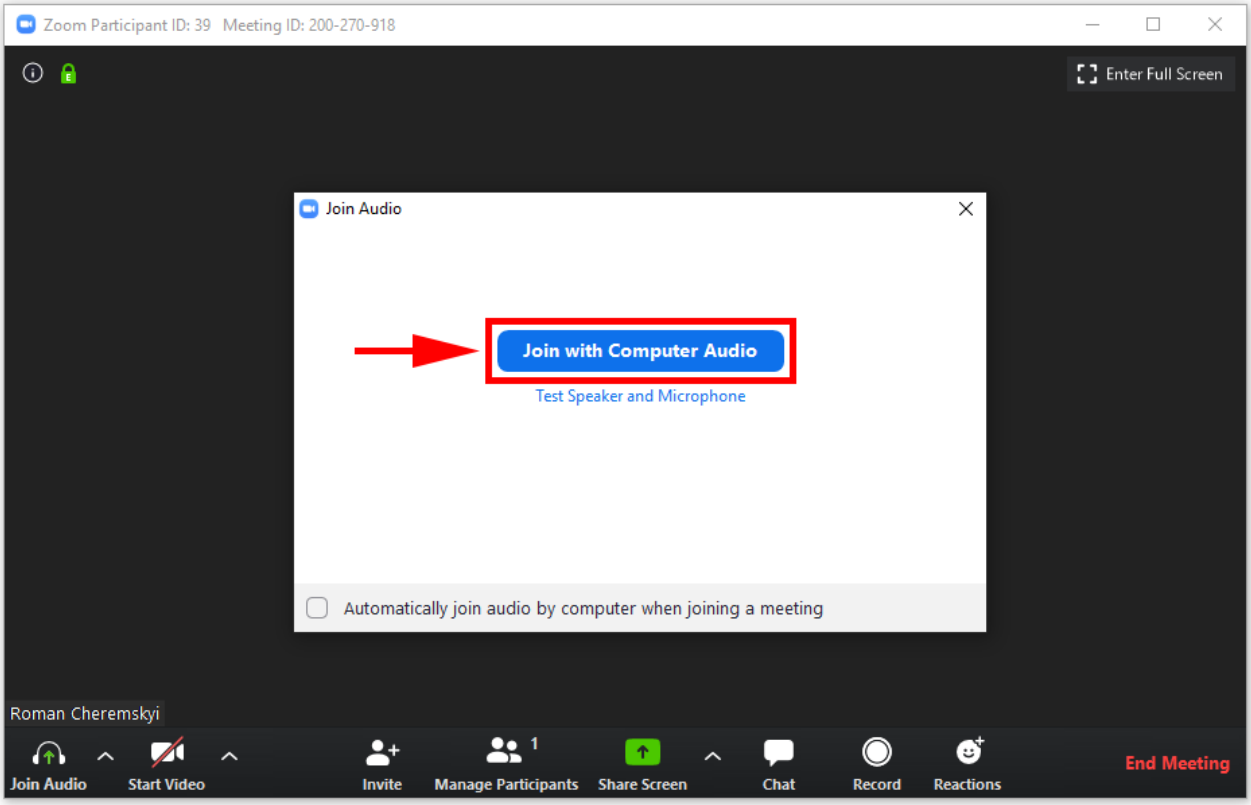


Рисунок 1.9 – Інтерфейс програмного застосунку Zoom

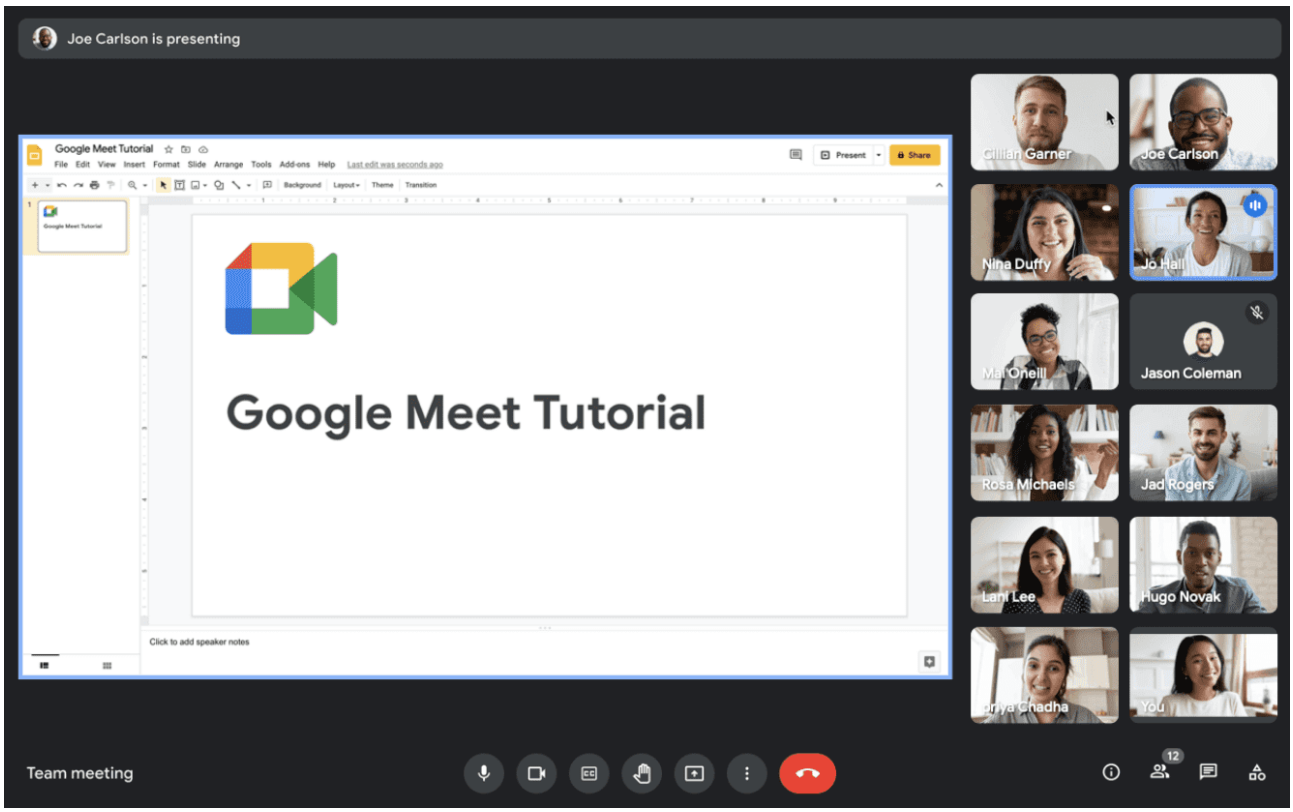


Рисунок 1.10 – Інтерфейс програмного застосунку Google Meet

Основними перевагами вищезазначених програм є:

– підтримка демонстрації навчального матеріалу в мультимедійній формі у реальному часі;

– підтримка чату, що дозволяє також проводити комунікацію як і в приватному чаті, так і в груповому з використанням текстових повідомлень;

– можливість запису екрану, що дозволяє поширити згодом файл цього запису на інших ресурсах;

– проведення опитувань;

– підтримка окремих кімнат для обговорення. Організатор відео конференції має змогу створювати окремі кімнати, в кожній з яких можуть брати участь інші учасники для обговорення будь-якої теми.

Серед недоліків даних програм є:

– обмеження у тривалості відео конференції та у загальної кількості учасників у безкоштовних версіях;

– можливість виникнення проблеми з ідентифікацією учасників конференції. Викладач завжди просить перейменовувати себе так, щоб була можливість ідентифікувати учасника конференції;

– на відміну від більшості інших платформ (Google Meet), Zoom не працює без завантаження програми на власний комп'ютер/телефон. Це створює додатковий бар'єр для входу для учасників, який може призвести до додаткових зусиль.

Але головним недоліком таких застосунків можна вважати те, що вони використовуються лише у реальному часі і учасники, які не мали змогу бути присутніми на той момент, не матимуть можливості задати додаткових питань з приводу якогось обговорення. Незважаючи на те, що запис екрану та поширення відео файлу серед всіх учасників дає змогу переглянути відео конференцію згодом, іноді виникає потреба в отриманні лише окремої частини інформації, яку необхідно шукати у цілому відеозаписі. Також відсутність форуму та аудіо повідомлень не дає змогу обговорювати якісь питання у будь-який час. Тому виникає необхідність статичної інформації, до якої мали б змогу доступу всі учасники у будь-який момент часу.

Telegram – багатоплатформовий месенджер, який дозволяє підтримувати комунікацію шляхом відправки повідомлень будь-якого формату (аудіо та відео повідомлення та конференції, текстові повідомлення, фотографії та файли багатьох форматів) [7]. Саме Telegram є найбільш розповсюдженим онлайн месенджером серед людей будь-якого віку в Україні, оскільки може використовуватись ефективно з будь-якого девайсу. На рисунку 1.11 можна побачити інтерфейс програми.

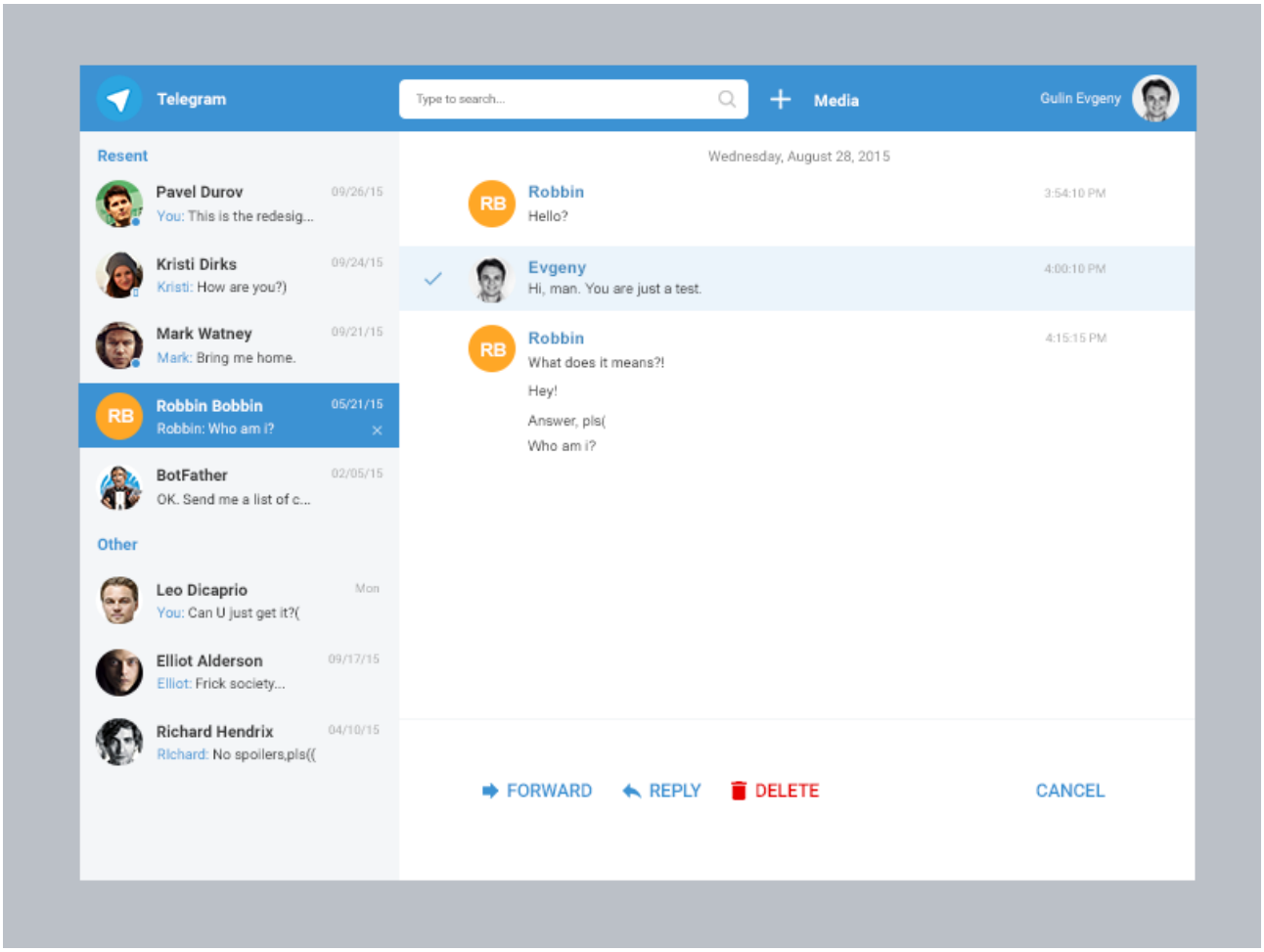


Рисунок 1.11 – Інтерфейс програмного застосунку Telegram

Головною особливістю використання Telegram у процесі навчання є:

– наявність групових чатів та можливість об’єднувати до 200 осіб в одному чаті;

– можливість створення каналів, які виконують функції дошки оголошень і можуть використовуватись для сповіщення всіх учасників навчального процесу про поточні події та ділитися додатковими ресурсами;

– підтримка голосових та відео повідомлень і конференцій;

– можливість передачі файлів будь-якого формату;

– проведення опитувань;

– можливість надсилання відкладених повідомлень, що дає змогу оголосити про певну подію у необхідний час;

До недоліків додатку Telegram з точки зору як застосунку для навчального процесу, можна віднести тільки те, що він застосовується не тільки у навчальних цілях, тому можуть виникнути додаткові незручності, такі як:

– відсутність форумів. Завдяки форумам інформація може бути чітко структурована та шанс знайти певне повідомлення є набагато вищим в порівнянні з груповим чатом у Telegram, де повідомлення можуть загубитися у величезному потоці тексту від інших учасників групи. Часто виникає проблема, коли доводиться дублювати раніше оголошену інформацію, так як не всі мали змогу переглянути її одразу.

– ідентифікація учасників групового чату. Оскільки застосунок Telegram використовується не тільки у навчальному процесі, то профілі учасників можуть мати довільну форму з довільними даними по імені та прізвищу, що ускладнює процес ідентифікації учасника у разі виникнення термінової передачі інформації безпосередньо учаснику.

Але, незважаючи на незначну кількість недоліків, в цілому додаток Telegram можна вважати одним із найкращих інструментів для ефективної та інтерактивної комунікації між учасниками навчального процесу, оскільки швидкість поширення інформації в порівнянні з іншими платформами буде значно більшою у зв’язку з тим, що в ньому наявна широка кількість корисних інструментів та ним користуються не тільки у навчальному процесі, а й повсякденно.

Нижче приведемо таблицю 1.1 для порівняння усіх описаних вище програмних застосунків та критеріїв, за якими вони оцінюються.

Таблиця 1.1 – Порівняння програмних застосунків за наявністю функціональних можливостей

	Google Classroom	Електронний кампус НТУУ КПІ	Moodle	Zoom/Google Meet	Telegram
Приватний чат	✗	✓	✓	✓	✓
Груповий чат	✗	✗	✓	✓	✓
Відео-конференція	✗	✗	✗	✓	✓
Аудіоповідомлення	✗	✗	✗	✗	✓
Дошка оголошень	✓	✗	✓	✗	✓
Опитування	✓	✓	✓	✓	✓
Коментарі	✓	✗	✓	✗	✓
Форум	✗	✗	✓	✗	✗

З таблиці можна зробити висновок, що в порівнянні з іншими застосунками, Telegram найбільше відповідає усім критеріям, які поставлені до вирішення задачі ефективної комунікації. Таким чином, буде доцільно застосувати даний програмний застосунок як додаткову платформу, що буде використовуватись у подальшій розробці програмного забезпечення.

1.3 Постановка завдання

Мета

Основною метою роботи є побудова архітектурного рішення для підтримки ефективної комунікації між учасниками навчального процесу, а також застосування побудованої багаторівневої архітектури при реалізації веб-застосунку.

Задачі

– на основі сучасних програмних засобів, що використовуються як засоби комунікації у навчальному процесі, побудувати багаторівневу архітектуру програмного забезпечення, націленого на підтримку ефективної комунікації між учасниками навчального закладу;

– побудувати даталогічну модель та на її основі створити фізичну модель бази даних;

– створити сутності для роботи з базою даних та оптимізувати запити шляхом додавання індексів до тих полів таблиць, які найчастіше використовуються у запитах;

– реалізувати логіку роботи основних компонентів програмного забезпечення використовуючи RESTful API;

– створити графічно-користувацький інтерфейс до веб-застосунку;

– реалізувати валідацію даних на рівні користувача та серверу;

– реалізувати засоби автентифікації та авторизації для розмежування доступу різних користувачів до відповідних ресурсів;

– написати автоматичні тести для перевірки роботи застосунку;

– додати можливість кешування;

– розробити виконання відкладених задач з інтеграцією Telegram-у;

– створити події та відповідних її слухачів для виконання асинхронних запитів до Telegram-у;

– забезпечити можливість масштабування;

– оцінити ефективність запропонованого рішення.

Вимоги

Можна висунути наступні вимоги до графічного інтерфейсу застосунку:

– інтуїтивно-зрозуміле розташування елементів навігації та управління;

– чітке та зрозуміле відображення інформації;

– наявність швидких переходів між сторінками;

– наявність сучасного дизайну;

– можливість адаптації інтерфейсу до різних електронних засобів;

– наявність динамічної інформації з використанням бази даних як сховища цієї інформації.

Необхідно уникнути будь-яких технічних навичок або знання якихось технологій, що можуть ускладнити роботу веб-застосунку для користувача. Надати таку можливість взаємодії застосунку з користувачем, щоб йому було достатньо лише базових навичок роботи з веб-браузером.

Дані, що надходять з клієнтської частини мають бути занесені до бази даних, яка реалізує всі вимоги підходу ACID. Для роботи з різними документами та зображеннями необхідно створити файлове сховище з використанням посилань для доступу до нього.

Висновки до розділу

У першому підрозділі описано предметне середовище та використання сучасних засобів програмного забезпечення, які дозволяють полегшити навчальний процес. Також виділено основні критерії, що будуть використовуватись для оцінки певного застосунку та які дають змогу ефективно підтримувати комунікацію між учасниками навчального процесу.

У другому підрозділі наведено актуальні програмні застосунки для підтримки комунікації та контролю навчального процесу, проаналізовано принцип їх роботи та головні проблеми під час використання. Також було розглянуто додаткові застосунки, якими користуються не тільки в освітніх цілях, але які відіграють важливу роль у покращенні навчального процесу. Визначено переваги та недоліки усіх існуючих платформ, котрі і є підґрунтям для усунення поточних недоліків шляхом розробки програмного забезпечення, що дозволить ефективно проводити комунікацію між учасниками навчального процесу.

У третій частині поточного розділу визначено загальну постановку завдання з описом мети та основних задач для її вирішення, а також сформовано вимоги до програмного забезпечення.

2 ВИБІР ЗАСОБІВ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

Для розробки програмного забезпечення спочатку необхідним етапом є проведення детального аналізу та порівняння засобів та актуальних технологій програмного забезпечення. Для цього потрібно обрати СКБД, мову програмування та допоміжні фреймворки, що будуть застосовуватись для основної розробки програмного забезпечення.

2.1 Обґрунтування вибору типу програмного забезпечення

2.1.1 Настільний застосунок

Настільний застосунок – прикладне програмне забезпечення, яке містить двійкові виконувані файли, що запускаються на автономному робочому столі [9].

Додатки для настільних комп’ютерів зазвичай мають можливість одночасного використання кількох функцій. Настільному додатку надано великий набір одночасних завдань або послідовність завдань, що виконують обов’язки. В ОС Windows настільні програми запускаються з основними обмеженими дозволами, які за замовчуванням необхідні для функціонування програми, але користувач може надати їм підвищені адміністративні привілеї. Наприклад, деякі програми можуть не працювати належним чином без таких підвищених дозволів, оскільки ці програми мають можливість змінювати файли на основі ОС. Прикладами такого є антивірусні програми.

Настільні програми можуть запускати кілька екземплярів паралельно. Програми для настільних комп’ютерів можуть працювати в будь-якій версії Windows. Деякі програми можуть бути несумісними зі старішими версіями Windows; основні функції програмування вирішують це у вихідному коді програми. На ці настільні програми необхідно встановити додаткові параметри конфігурації для служб на базі Windows, які надають їм доступ до використання різних системних ресурсів і дозволяють їм виконувати пов’язані завдання на основі вимог клієнта, наприклад антивірусні та VPN програми. Нативні десктоп-застосунки мають наступні переваги:

– можливість роботи офлайн. Використовуючи десктоп-застосунок можна отримати доступ до нього в будь-який час. Робочий стіл користувача дозволяє йому залишатися функціональними без підключення до інтернету;

– висока безпечність. Додатки для настільних комп'ютерів дають можливість працювати в автономному режимі, що автоматично надає їм перевагу над веб-програмами. Якщо необхідно працювати над чутливим даними, можна відключити локальну мережу або Wi-Fi і більше ніколи не турбуватися про загрозу. У деяких випадках можна навіть встановити диск із програмою, і не потрібно буде виходити в Інтернет, щоб вона працювала;

– залежність від швидкості ПК. Якщо користувач має дійсно швидкий комп'ютер і повільне підключення до Інтернету або даних, то настільна програма матиме значну перевагу. Він працює з пам'яттю та потужністю обробки локальних ресурсів, щоб забезпечити позитивний досвід користувача;

– дешева підтримка. Коли потрібно купити програму для настільних комп'ютерів, зазвичай виникають більші капітальні витрати, ніж якщо придбати веб-програму або підписку для її використання. Якщо порівняти витрати протягом усього терміну дії програми, одноразова вартість зазвичай дешевша, ніж поточні платежі, які потрібно оплачувати, щоб і надалі мати доступ до веб-програми. Рідко виникають будь-які витрати на технічне обслуговування, і користувач залишається під контролем будь-яких оновлень, які він вважає необхідними, вибираючи варіант настільного комп'ютера;

– підтримка різних версій. Якщо не оновлювати комп'ютер протягом 10 років, існує велика ймовірність того, що програми, якими користувач захоче користуватися, не працюватимуть у поточній системі. Якщо є стара версія Microsoft Word, яку користувач хочете використовувати, яка сумісна зі старою операційною системою, він все одно можете скористатися всіма її функціональними можливостями. Користувач може не отримати технічну підтримку для усунення несправностей у цій ситуації, але плюси, безумовно, переважають мінуси.

До недоліків, які мають десктоп-застосунки слід віднести:

– потреба у кількох оновлень, щоб продовжувати використовувати найкращі функції. Хоча є можливість часто використовувати опцію робочого столу в попередній формі без оновлення, вона в кінцевому підсумку не забезпечить повну функціональність, яка може знадобитися;

– настільні програми вимагають окремої установки для кожної станції. Хоча це, як правило, дає ліцензію на використання, існують додаткові витрати на управління вимогами IT, водночас урівноважуючи більші капітальні витрати. Якщо хтось не має програми на своєму робочому столі, у нього немає можливості приєднатися до спільної роботи.

– залежність десктоп-застосунок від операційної системи. Для кожної ОС необхідно мати ряд спеціалістів, які матимуть змогу розробити застосунок для відповідної платформи;

– довгий час розробки. Для того, щоб настільна програма могла працювати на різних ОС необхідно створити окремий застосунок на кожну таку платформу, що в свою чергу займає багато часу;

– необхідність періодичного оновлення такого типу застосунків.

2.1.2 Веб-застосунок

Веб-застосунок – це тип прикладного програмного забезпечення, яке зазвичай запускається за допомогою веб-браузера, а також використовує різні веб-технології для виконання різних завдань у глобальній мережі [8].

Веб-додаток може бути розроблений для кількох цілей, яким може користуватися будь-хто, окрім того, він може використовуватися як індивідуально конкретною особою або як цілою організацією з кількох причин. Загалом, веб-додаток може містити інтернет-магазини (так звані магазини електронної комерції), веб-пошту, калькулятори, платформи соціальних мереж тощо. Для роботи більшості веб-додатків зазвичай використовується спеціальний веб-браузер, який дозволяє отримувати до них доступ. Більшість веб-програм, доступних в Інтернеті, можна отримати за допомогою стандартного веб-браузера.

Якщо говорити про веб-додаток загалом, то він зазвичай використовує комбінацію серверних скриптів для обробки інформації, зберігання і пошуку даних. Деякі з них також використовують сценарії на стороні клієнта, такі як JavaScript, HTML для представлення даних та інформації перед користувачами, а деякі веб-додатки також використовують як серверну, так і клієнтську сторону одночасно. Це дозволяє користувачам спілкуватися з організацією або компаніями за допомогою онлайн-форм, онлайн-форумів, кошиків для покупок, системи керування вмістом та багато іншого. Крім того, веб-додатки також дозволяють своїм користувачам створювати документи, ділитися ними або обмінюватися даними/інформацією. Використовуючи веб-додаток, користувачі можуть співпрацювати над тими самими проектами, якщо вони знаходяться в різних географічних місцях.

Наведемо деякі переваги у використанні веб-застосунків:

– будь-яка типова веб-програма може працювати або доступна в будь-якій операційній системі, наприклад Windows, Mac, Linux, якщо браузер сумісний;

– веб-додаток зазвичай не потрібно встановлювати на жорсткий диск комп’ютерної системи, що усуває всі проблеми, пов’язані з обмеженням простору;

– існує єдина версія веб-застосунку для користувачів, що усуває всі проблеми, які можуть виникнути із сумісністю;

– зменшення програмного піратства у веб-додатках з використанням підписки, наприклад, SAAS (програмне забезпечення як послуга);

– зменшення допоміжних ресурсів та витрат для компаній, тому що обслуговування, необхідне бізнесу, значно менше в порівнянні з іншими типами застосунків;

– наявність гнучкості. Користувач веб-додатку може працювати з будь-якого місця, якщо у нього є з’єднання до інтернету;

– з використанням хмарних технологій простір для збереження є практично необмеженим;

– можливість запрограмувати веб-застосунок для роботи в широкому діапазоні операційних систем, на відміну від рідних програм, які можуть працювати на певній платформі.

Однак, використовуючи веб-застосунки можна виділити такі недоліки:

– необхідність підключення до Інтернету для доступу до будь-якої веб-програми, а без підключення до Інтернету ніхто не може використовувати жодну з веб-програм. Дуже легко можна отримати підключення до Інтернету в сучасних містах, а в сільській місцевості доступ до Інтернету не такий доступний;

– багато користувачів люблять використовувати різні веб-браузери відповідно до своїх потреб і вибору. Отже, створюючи веб-програму, необхідно пам’ятати, що програма повинна підтримувати кілька веб-браузерів, включаючи нові та старі версії браузерів;

– проблеми, пов’язані зі швидкістю, також впливають на продуктивність веб-програми, оскільки існує кілька факторів, від яких залежить продуктивність веб-програми, і всі ці фактори впливають на продуктивність веб-програми в цілому;

– необхідність виділення додаткових витратит, щоб підтримувати належний стан веб-додатка, надавати оновлення щоразу, коли виникає проблема, і створювати привабливий користувальницький інтерфейс, який зовсім не такий дешевий;

– для більш безпечної роботи веб-застосунку можуть використовуватись протоколи передачі даних HTTPS або SSL, які потребують додаткових зусиль для їх налаштування.

2.1.3 Результати порівняння

Провівши аналіз типів прикладного програмного забезпечення та перелічивши основні недоліки та переваги, було визначено, що доцільно використати веб-застосунок у свої роботі, тому що він дозволяє застосувати сучасні технології розробки, що є актуальними на сьогоднішній час, а також веб-

застосунок має відсутність великої затрати часу на його розробку. Використання веб-застосунків додає гнучкості та усуває проблему з його оновленням, оскільки при додаванні нової версії немає залежності від того, що користувачі встановлять оновлення тільки у той момент часу, коли вони самі побажають.

2.2 Аналіз та порівняння RDBMS

Реляційна база даних (RDB) — це сукупний набір даних, які упорядковуються з використанням таблиць, рядків та стовпців [10]. Головною концепцією у реляційних базах даних є визначення відносин між її таблицями. Саме обмін інформацією відбувається між таблицями, які дозволяють спростити вибірку даних.

Система управління реляційною базою даних (СКБД або RDBMS) — це програма, яка дозволяє виконувати різні операції над реляційною базою даних, а саме операції створення, оновлення та її адміністрування. Для доступу до бази даних в таких системах використовується мова структурованих запитів (SQL), що є стандартом для утворення взаємодії між користувачем та базою даних [11]. В залежності від використання певної СКБД, існує деяка відмінність між ними та синтаксисом SQL. Тому виникає необхідність у оптимальному виборі СКБД на базі якої буде розроблюватися програмне забезпечення. Для цього буде виконано детальний огляд популярних СКБД і порівняно їх переваги та недоліки.

2.2.1 MySQL

MySQL – це СКБД, що заснована на мові SQL, яка є популярною мовою для доступу до записів у базі даних і керування ними. MySQL представляє собою безкоштовне програмне забезпечення з відкритим вихідним кодом, а також підтримується компанією Oracle [12].

MySQL на даний момент є найпопулярнішою системою управління базами даних, яка використовується у сучасному світі у багатьох відомих компаніях. Вона зазвичай використовується разом з мовою програмування PHP для створення потужних і динамічних серверних або веб-додатків підприємства.

Сама СКБД розроблена, продається та підтримується компанією MySQL AB, і написано мовою програмування C і C++.

MySQL підтримується багатьма мовами програмування, які мають додаткові бібліотеки для роботи з нею. Наприклад, у мові Java використовується драйвер JDBC, а у C# - застосунок MySQL Connector. Загалом, MySQL можна вважати стабільною та потужною СКБД, що має ряд наступних переваг:

– відсутність фінансових витрат на використання;

– наявність портативності. Це дає можливість працювати на різних платформах;

– безпека та надійність. Дані захищаються паролем, і перевага цих паролів полягає в тому, що вони зберігаються в зашифрованому вигляді і зламати ці складні алгоритми шифрування практично неможливо;

– продуктивність за рахунок використання представлень, тригерів і збережених процедур;

– підтримка реплікації даних, що забезпечує безперебійну роботу навіть у разі збою. MySQL також використовує різноманітні стратегії резервного копіювання та відновлення, щоб гарантувати, що дані не будуть втрачені в разі збою системи або ненавмисного видалення;

– комплексна підтримка транзакцій. З такими функціями, як повна атомарність, послідовність, ізольованість, тривала підтримка транзакцій; підтримка багатоверсійних транзакцій; і необмежене блокування на рівні рядків, є найкращим рішенням для повної цілісності даних.

Однак, слід також виділити і недоліки MySQL, одними із таких є:

– версії MySQL, що є нижчими за 5.0 не дозволяють виконувати операції COMMIT та ROLE, а також не підтримують збережені процедури;

– не дуже ефективно обробляє транзакції і існує можливість пошкодження даних;

– відсутність підтримки обмежень перевірки SQL;

– відсутність якісного інструменту для розробки та налагодження порівняно з іншими базами даних;

– мала ефективність у випадку великих розмірів бази даних.

2.2.2 Microsoft SQL Server

Microsoft SQL Server – це СКБД, що розроблена та представлена компанією Microsoft. Вона містить мову SQL і Transact-SQL (T-SQL), що є власною мовою Microsoft з можливостями обробки винятків, оголошення змінної та збережених процедур. SQL Server Database Engine — це основний компонент SQL Server, який відповідає за контроль, обробку та захист зберігання даних. Механізм баз даних розділений на два сегменти, реляційний механізм, який використовується для обробки команд і запитів. Другий — це механізм зберігання даних, призначений для керування різними функціями бази даних, такими як таблиці, сторінки, файли, індекси та транзакції [13].

Нижче наведено переваги, що містить Microsoft SQL Server:

– простота у використанні, підтримка масштабованості для розподілених організацій;

– підтримка безпеки даних;

– достатньо проста інсталяція та налаштування;

– підтримка інтеграції з багатьма мовами програмування;

– підтримка швидкого і простого способу аналізу вихідних даних під час усунення несправностей;

– наявність тригерів;

– підтримка оптимізації запитів;

– підтримка відкладених задач;

– підтримка відновлення даних. У разі відключення живлення або вимкнення сервера дані можуть пошкодитися, що створює велику проблему для компаній, які майже не зберігають резервних копій. Microsoft SQL Server усуває ризик втрати даних завдяки функціям відновлення даних. Дані можуть бути захищені за допомогою кешування, файлів журналів і частого резервного копіювання, незалежно від того, що може статися з вашим сервером;

– включає професійне програмне забезпечення для керування базами даних корпоративного рівня. Програмне забезпечення, яке пропонує Microsoft, також забезпечує тісну інтеграцію з платформою .NET, чого немає у конкуруючих продуктах.

Хоча SQL Server має також деякі недоліки, представлені нижче:

– наявність платної ліцензії за користування версії Enterprise Edition, яка призначена для бізнесу;

– наявність проблем з ефективністю до пам’яті з таблицями великих розмірів;

– наявність апаратних обмежень. Для нових версій Microsoft SQL необхідно використовувати нові апарати, які будуть здатні підтримувати її, що приводить до додаткових витрат;

– обмежена сумісність. Microsoft SQL Server призначений лише для роботи на серверах на базі Windows. З різних причин, включаючи витрати на ліцензування та проблеми безпеки, розробники можуть вирішити розмістити свої веб-сайти на машинах на базі Unix. У цьому випадку вони не зможуть використовувати SQL Server. Окрім неможливості запуску на платформах, які не є Windows, також можуть виникнути проблеми з сумісністю щодо взаємодії з програмами, які працюють на інших платформах.

2.2.3 PostgreSQL

PostgreSQL — це СКБД корпоративного класу, яка працює на платформі Linux. PostgreSQL не контролюється жодною корпорацією чи іншою приватною особою, а вихідний код доступний знаходиться у вільному доступі. Вона підтримує як структуровану мову запитів і JSON для реляційних і нереляційних типів БД [14]. Також PostgreSQL підтримує розширені типи даних і функції оптимізації продуктивності, які доступні лише в дорогих комерційних базах даних, таких як Oracle і SQL Server.

Нижче наведено основні переваги PostgreSQL:

– безкоштовна та доступна для всіх, що дозволяє розширювати функціональні можливості в залежності від вимоги бізнесу;

– підтримка асинхронної реплікації даних;

– наявність табличних просторів;

– наявність вкладених транзакцій;

– підтримка резервного копіювання;

– розширений функціонал планувальника задач;

– відмовостійкість завдяки введенню журналу;

– підтримка всіх операційних систем;

– можливість визначати власні типи даних, створювати власні функції і навіть писати код іншою мовою програмування без перекомпіляції бази даних.

Нижче наведено недоліки вищезазначеної СКБД:

– є програмою баз даних з відкритим вихідним кодом і, отже, не належить одній конкретній організації. Незважаючи на багатофункціональні та видатні можливості, у неї були проблеми з поширенням свого імені в порівнянні з власним програмним забезпеченням, яке має повний контроль та авторські права на продукт. Тому на неї не поширюється гарантія і вона не несе жодної відповідальності чи захисту від відшкодування;

– зміни, внесені для підвищення швидкості, вимагають більше роботи, ніж MySQL, оскільки PostgreSQL зосереджується на сумісності;

– багато програм з відкритим кодом можуть не підтримувати PostgreSQL;

– за показниками продуктивності він повільніше, ніж MySQL.

2.2.4 Результати аналізу

Таким чином, виконавши аналіз та перелічення основних недоліків та переваг про RDBMS Microsoft SQL Server, MySQL та PostgreSQL, було виділено незначні переваги PostgreSQL над іншими, головними із яких є наявність всіх стандартних можливостей для створення і підтримки бази даних, її розширення згідно вимог бізнесу, а також безкоштовного програмного забезпечення з

відкритим кодом, що означає, що не доведеться виділяти додаткові витрати за використання додаткових послуг.

2.3 Обґрунтування вибору мови програмування для розробки серверної частини застосунку

Для побудови веб-застосунку необхідно приділити увагу і вибору мові програмування для реалізації серверної частини, яка буде основним фундаментом обробки та керування даними, матиме взаємодіє із зовнішніми серверами та відповідатиме за взаємодію із БД. Також доцільно буде використати ту мову програмування, яка є об’єктно-орієнтованою, матиме досить велику кількість бібліотек для роботи з будь-якими задачами, а також матиме фреймворк, який легко використати для створення веб-серверу.

2.3.1 C#

C# – це мова програмування, розроблена та запущена компанією Microsoft у 2001 році. Ця мова проста, сучасна й об’єктно-орієнтована, яка надає сучасним розробникам гнучкість та можливості для створення програмного забезпечення, яке не тільки працюватиме сьогодні, але й буде застосовуватися протягом багатьох років [15].

Метою C# була розробка мови програмування, яку не тільки легко вивчити, але й підтримувати сучасні функціональні можливості для всіх видів розробки програмного забезпечення. Мова C# була розроблена для компаній, щоб створювати всі види програмного забезпечення за допомогою однієї мови програмування. Ця мова програмування підтримує потреби в розробці веб та мобільних пристроїв, а також програмування серверної частини програм на базі операційної системи Windows.

Переваги:

– підтримка зворотної сумісності;

– інтеграція з іншими мовами. Додаток, написаний на .NET, матиме кращу інтеграцію та інтерпретацію порівняно з іншими технологіями.

Програмування на C# працює на C.L.R, що полегшує інтеграцію з компонентами, написаними іншими мовами;

– наявність великої кількості бібліотек. Мова C# має багатий клас бібліотек, які полегшують реалізацію багатьох функцій;

– наявність надійного резервного копіювання пам'яті. Мова програмування C# містить резервну копію великої кількості пам’яті, тому проблема витоку пам’яті та інших подібних проблем не виникає;

– наявність C-подібного синтаксису, що спрощує вивчення мови;

– наявність високої безпеки коду завдяки тому, що код компілюється у бінарний вигляд і його декомпіляція практично неможлива.

Недоліки:

– звільнення пам’яті здійснюється неявно за допомогою збирача сміття. Це ускладнює оцінку використання пам’яті та загальної продуктивності великих складних програм;

– необхідність платформи Windows для того, щоб запустити .NET;

– відсутність гнучкості у зв’язку із залежністю Microsoft .NET;

– відсутність підтримки старих версій;

– відсутність роботи на низькому рівні для безпосередньої взаємодії з обладнанням через драйвери та мікропрограм.

– перекомпіляція програми у випадку зміни будь-якої частини коду, що може призвести до додаткових витрат часу та помилок.

2.3.2 Java

Java – це широко використовувана об’єктно-орієнтована мова програмування та програмна платформа, яка працює на мільярдах пристроїв, включаючи ноутбуки, мобільні пристрої, ігрові консолі, медичні пристрої та багато інших. Правила та синтаксис Java засновані на мовах C і C++ [16].

Java складається як з мови програмування, так і з програмної платформи. Для створення програми за допомогою Java, потрібно завантажити Java Development Kit (JDK), доступний для Windows, macOS та Linux. Програмна

платформа Java складається з JVM, Java API та повного середовища розробки. JVM аналізує та запускає (інтерпретує) байт-код Java. Java API складається з великого набору бібліотек, включаючи основні об’єкти, мережеві функції та функції безпеки; генерація розширюваної мови розмітки (XML); веб-сервіси. У сукупності мова Java і програмна платформа Java створюють потужну, перевірену технологію для розробки корпоративного програмного забезпечення.

Наведено такі переваги даної мови програмування:

– підтримка портативності. Будь-який додаток, написаний на одній платформі, можна легко перенести на іншу платформу;

– захищеність програмного коду. Досягається тим, що весь код після компіляції перетворюється в байт-код, який не є читабельним для людини. Це дає змогу розробляти системи та програми без вірусів і несанкціонованого доступу;

– відсутність вказівників та перевантажень операторів, що ускладнює вивчення мови;

– підтримка динамічності. Має здатність адаптуватися до середовища, що розвивається, яке підтримує динамічне виділення пам’яті, завдяки якому зменшується втрата пам’яті та підвищується продуктивність програми;

– надійність забезпечується тим, що Java має потужну систему управління пам'яттю. Це допомагає усунути помилки, оскільки перевіряє код під час компіляції та виконання;

– висока продуктивність. Java досягає високої продуктивності завдяки використанню байт-коду, який можна легко перевести на рідний машинний код. Завдяки використанню компіляторів JIT (Just-In-Time) він забезпечує високу продуктивність;

– підтримка багатопоточності. Java підтримує кілька потоків виконання, включаючи набір примітивів синхронізації. Це значно полегшує програмування з потоками.

Серед недоліків Java можна виділити:

– відсутність резервного копіювання даних. Java в основному працює зі сховищем, а не зосереджується на резервному копіюванні даних;

– вимога до значного обсягу пам’яті в порівнянні з іншими мовами. Під час виконання збирання сміття ефективність пам’яті та продуктивність системи може бути негативною;

– погана читабельність. Java має багатослівний код, що означає, що в ньому багато слів і багато довгих і складних речень, які важко читати і розуміти;

– відсутність підтримки низькорівневого програмування.

2.3.3 Результати аналізу

Після перелічення всіх недоліків та переваг обох мов програмування було вирішено використати мову Java для програмування серверної частини програмного забезпечення. Завдяки своїй розповсюдженості, ця мова програмування має значну кількість бібліотек з детальною документацією для користувачів, що дозволяє зручно розроблювати веб-застосунок, який можна інтегрувати практично з усіма системами та сервісами за рахунок її кросплатформеності.

2.4 Обґрунтування вибору мови програмування для розробки клієнтської частини застосунку

Для реалізації клієнтської частини застосунку необхідно взяти до уваги мови програмування, які містять велику кількість бібліотек та компонентів, щоб мати можливість реалізувати будь-яку функціональність. Найбільш розповсюдженою мовою програмування для клієнтської частини є JavaScript, але також потрібно взяти до уваги мову Typescript, яка набирає зараз популярності і є не менш ефективною у користуванні. Розглянемо обидві мови та перелічимо їх переваги та недоліки.

2.4.1 JavaScript

JavaScript - це динамічна мова програмування. Вона легка і найчастіше використовується як частина веб-сторінок, чиї реалізації дозволяють

клієнтському сценарію взаємодіяти з користувачем та додавати до сторінок властивість динамічності, використовуючи різноманітні компоненти [17].

JavaScript на стороні клієнта є найпоширенішою мовою. Сценарій повинен бути включений в HTML-документ або посилання на нього, щоб код інтерпретував браузер. Це означає, що веб-сторінка не обов'язково має бути статичним HTML, але може включати програми, які взаємодіють з користувачем, керують браузером і динамічно створюють вміст HTML. Механізм на стороні клієнта JavaScript надає багато переваг перед традиційними серверними сценаріями. Наприклад, можна використовувати JavaScript, щоб перевірити, чи ввів користувач дійсну адресу електронної пошти в поле форми. Код JavaScript виконується, коли користувач надсилає форму, і лише якщо всі записи дійсні, вони будуть передані на веб-сервер. JavaScript можна використовувати для захоплення ініційованих користувачем подій, таких як натискання кнопок, навігація за посиланнями та інші дії, які користувач ініціює явно чи неявно.

Дана мова має такий список переваг:

– наявність величезної кількості бібліотек;

– швидкість;

– менша взаємодія з сервером. Наприклад, можна перевірити введення користувача перед відправкою сторінки на сервер. Це економить трафік сервера, що означає менше навантаження на ваш сервер;

– негайний зворотній зв'язок із відвідувачами сайту, тобто користувачам не потрібно чекати перезавантаження сторінки, щоб побачити, чи не забули вони щось ввести;

– підтримка інтерактивності. Можна створювати інтерфейси, які реагують, коли користувач наводить курсор на них за допомогою миші або активує їх за допомогою клавіатури;

– широка розповсюдженість. JavaScript використовується майже в усіх веб-застосунках.

До недоліків відносяться:

– при виникненні помилки, JavaScript може припинити відтворення всього веб-сайту. Браузери надзвичайно чутливі до помилок цієї мови;

– код може бути переглянутий будь-ким;

– недостатня безпека на стороні клієнта. Оскільки код JavaScript виконується на стороні клієнта, помилки іноді можуть використовуватися для зловмисних цілей;

– конфігурація часто потребує великих зусиль щодо кількості інструментів та бібліотек, які потрібно об’єднати, щоб створити середовище для великого проекту;

– відсутність багатопоточних або багатопроцесорних можливостей;

– можливість неправильної інтерпретації у браузерах. Різні браузери іноді по-різному інтерпретують код JavaScript.

2.4.2 Typescript

Створений і підтримуваний компанією Microsoft, TypeScript представляє собою мову програмування з відкритим вихідним кодом, яка є не тільки однією з найпопулярніших, але й неймовірно швидко розвивається. Ця мова дуже схожа на JavaScript. Вона була створена як верхня обгортка JavaScript, що означає, що вона пропонує повну сумісність з JavaScript, але будується на основі старої мови з новими можливостями. Іншими словами, усі існуючі програми JavaScript є автоматично дійсними програмами TypeScript. TypeScript виділяється тим, що він наповнений потужнішими функціями, які допомагають розробникам створювати великомасштабні програми. Крім того, TypeScript підтримується безкоштовним редактором коду Microsoft Visual Studio Code [18].

Переваги даної мови:

– підримка статичної типізації. TypeScript поставляється з додатковою системою статичного введення і виведення типів через TLS (TypeScript Language Service). Тип змінної, оголошеної без типу, може бути визначений TLS на основі її значення;

– підтримка концепції об’єктно-орієнтованого програмування, такої як класи, інтерфейси, успадкування тощо;

– наявність компіляції коду. TypeScript компілює код і генерує помилки компіляції, якщо знайде якісь синтаксичні помилки. Це допомагає виділити помилки перед запуском сценарію;

– підтримка визначення типів. Файл визначення TypeScript (з розширенням .d.ts) надає визначення для зовнішніх бібліотек JavaScript. Отже, код TypeScript може містити ці бібліотеки.

Недоліки:

– витрата часу на компіляцію коду;

– щоразу, коли TypeScript потрібно запустити в браузері, перед цим має бути крок компіляції для перетворення TS на JS;

– відсустність деяких функціональних можливостей в порівнянні з JavaScript;

– при використанні сторонньої бібліотеки має бути в ній наявний документ визначення, який буває не завжди недоступним.

2.4.3 Результати аналізу

У зв’язку з тим, що мова програмування Typescript має ряд деяких переваг, які витісняють недоліки мови JavaScript, то було обрано її у якості мови програмування для розробки клієнтської частини програмного забезпечення. Саме завдяки компіляції коду Typescript дозволить зекономити час на пошук помилок та їх вирішення, що в цілому забезпечить більше концентрацію на реалізацію задач, а ніж на відслідковування проблем.

2.5 Використання додаткових технологій

Окрім виділення мов програмування для клієнтської та серверної частини веб-застосунку, також бажано приділити увагу і ключовим технологіям, які плануються використати у подальшій розробці програмного забезпечення, і за допомогою яких можна ефективно досягти поставленої мети дослідження.

2.5.1 Spring Framework

Spring Framework — це фреймворк з відкритим вихідним кодом для створення серверної частини веб-додатків на мові програмування Java. Він потужний і легкий, а також простий у використанні і забезпечує підтримку для легкої розробки Java-додатків [19].

Архітектура Spring Framework надає безліч модулів, які можна використовувати на основі вимог програми. На рис. 2.1 наведені основні компоненти даного фреймворку.

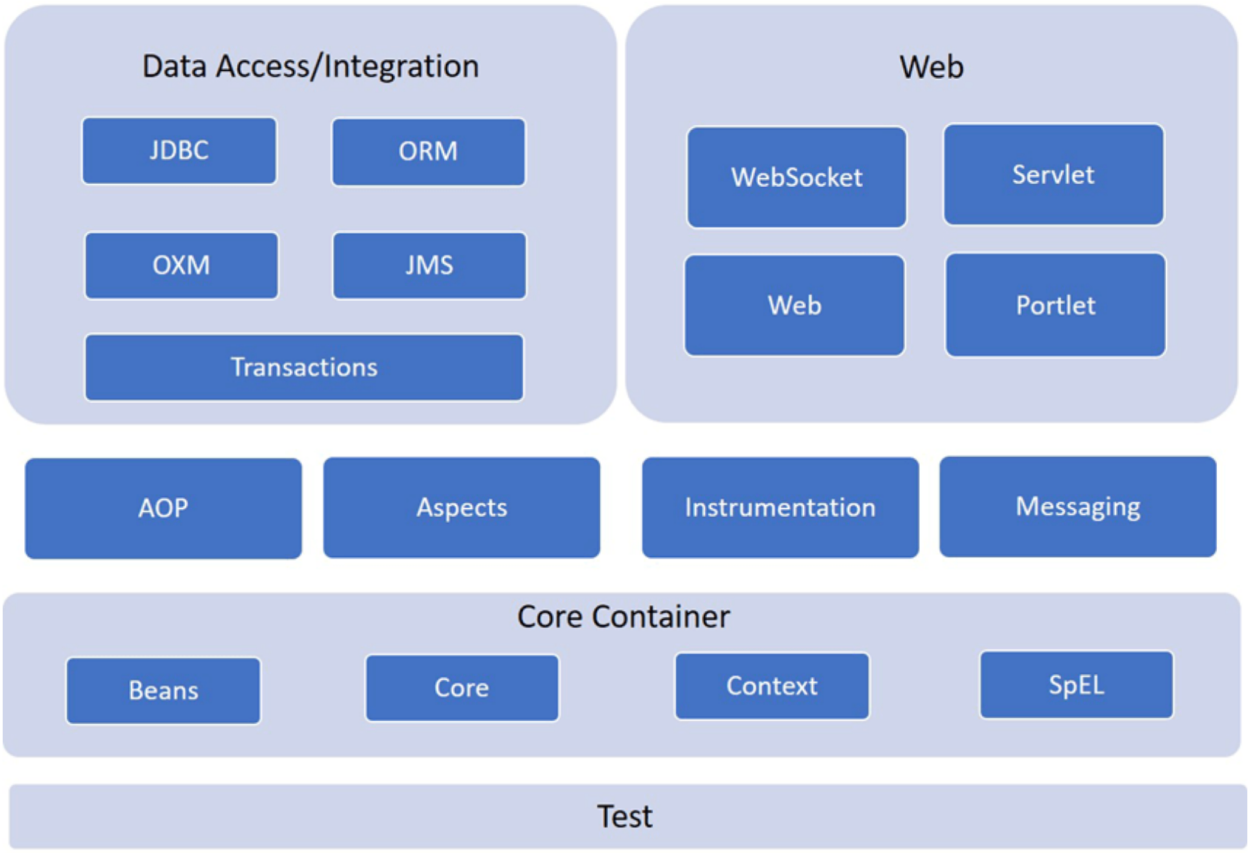


Рисунок 2.1 – Основні компоненти фреймворку Spring

Нижче наведемо ті модулі, які будуть використовуватись у розробці програмного забезпечення.

Основний модуль Spring. Модуль Spring Core, який є основним компонентом фреймворку Spring, забезпечує контейнер IoC. Є два типи реалізацій контейнера Spring, а саме, фабрика компонентів і контекст програми. Фабрика компонентів визначається за допомогою інтерфейсу BeanFactory і діє як контейнер для beans. Контейнер Bean дозволяє відокремити конфігурацію та

специфікацію залежностей від логіки програми. У фреймворку Spring фабрика Bean діє як центральний контейнер IoC, який відповідає за створення екземплярів об’єктів програми. Він також налаштовує та збирає залежності між цими об’єктами. Існує безліч реалізацій інтерфейсу BeanFactory. Клас XmlBeanFactory є найпоширенішою реалізацією інтерфейсу BeanFactory, що дозволяє усунути взаємозалежності між об’єктами програми.

Модуль Spring AOP. Подібно до ООП, яке розбиває програми на ієрархію об'єктів, АОП розбиває програми на аспекти або проблеми. Цей модуль дозволяє реалізувати проблеми або аспекти в додатку Spring. У Spring AOP аспектами є звичайні Spring bean або звичайні класи, анотовані анотацією @Aspect. Ці аспекти допомагають в управлінні транзакціями, веденні журналів і моніторингу збоїв програми. Наприклад, керування транзакціями потрібне для банківських операцій, таких як переказ суми з одного рахунку на інший. Модуль Spring AOP забезпечує рівень абстракції керування транзакціями, який можна застосувати до API транзакцій.

Модуль Spring ORM. Використовується для доступу до даних з БД у програмі. Він надає API для маніпулювання базами даних за допомогою JDO або Hibernate. Spring ORM підтримує DAO, що забезпечує зручний спосіб створення таких рішень ORM на основі DAO:

- підтримка простого декларативне управління транзакціями
- прозорість в обробці винятків
- наявність потокобезпечних, легких класів шаблонів
- наявність класів підтримки DAO
- управління ресурсами

Модуль Spring MVC. Реалізує архітектуру MVC для створення веб-додатків. Він розділяє код моделі і компонентів перегляду веб-додатка. У Spring MVC, коли запит генерується з браузера, він спочатку надходить до класу DispatcherServlet, який відправляє запит до контролера, використовуючи набір обробників. Контролер витягує та обробляє інформацію, вбудовану в запит, і надсилає результат до класу DispatcherServlet у вигляді об’єкта моделі. Тільки

тоді, клас DispatcherServlet використовує класи ViewResolver для надсилання результатів у представлення, яке відображає ці результати користувачам.

Контекстний модуль програми Spring. Заснований на модулі Core. Контекст програми ApplicationContext — це інтерфейс BeanFactory. Цей модуль підтримує такі функції, як інтернаціоналізація, валідація, поширення подій і завантаження ресурсів. Контекст програми реалізує інтерфейс MessageSource і надає програмі функціональні можливості обміну повідомленнями.

2.5.2 Angular

Angular – це фреймворк, написаний на TypeScript. Він використовується для розробки односторінкових додатків із використанням TypeScript та мови шаблонів HTML. У ньому практикуються теми необов'язкової та основної функціональності як набір бібліотек та ресурсів TypeScript, які можна імпортувати у свої програми. Архітектура спирається на конкретні основні концепції Angular. У мові програмування Angular елементарними будівельними блоками є NgModules, які забезпечують середовище компіляції для компонентів. Ці модулі збирають пов'язаний код у функціональний набір. Додаток завжди містить кореневий модуль, який надає можливість завантаження і класично має багато реалізацій модулів функцій[20].

Основними особливостями цього фреймворку є:

– кроссплатформенність. Розробник може оптимально використовувати цю сучасну платформу для відтворення динамічних додатків з високоякісною продуктивністю та підходом до її встановлення без виконання зайвих кроків. Інженери програмного забезпечення можуть легко створити програму AngularJS, залежну від робочого столу, для Mac, Windows або Linux за допомогою рідних API ОС;

– ефективність. Angular створює інтерфейс користувача з простим синтаксисом шаблонів. Angular CLI швидко утворює та додає компоненти і тести, а потім розгортає деталі. Екосистема Angular допомагає, наприклад, скористатися миттєвим виявленням помилок, інтелектуальним завершенням

коду та деякими іншими функціями в інтегрованому середовищі розробки та редакторах;

– повна підтримка розвитку застосунку. Використовуючи інтуїтивно зрозумілий API, будь-який розробник може створювати високопродуктивні графіки анімації, і це з мінімальним написанням коду. Angular служить для створення доступних додатків з посібниками для розробників і методом Accessible Rich Internet Applications;

– продуктивність і швидкість. Додатки Angular генерують частини шаблонів у код, який значною мірою реформований за допомогою сучасних інструментів віртуальної розробки JavaScript. Angular є універсальним, оскільки надає перший вигляд програми на .NET, Node.js або PHP для майже миттєвої і керованої поведінкою інтерпретації в HTML-документах.

Слід додати, що Angular побудовано на архітектурі MVC. Відповідно до нього запити від програми надходять до контролера, де він взаємодіє з моделлю для підготовки даних, необхідних для представлення. Архітектура дозволяє ізолювати логіку програми від інтерфейсу користувача та покращує зв’язування даних, а також весь процес розробки.

2.5.3 Telegram API

У першому розділі дисертації вже було наведено, що саме Telegram є найбільш розповсюдженим та гнучким застосунком, який має величезну кількість різноманітних функціональних можливостей. Тому доцільно буде використовувати офіційне API, яке дасть змогу полегшити поєднання користувачів месенджеру з програмним застосунком, який планується розробити у даній роботі. Саме використання цієї технології дозволить вирішити задачу ефективної комунікації між учасниками навчального процесу.

Telegram API дозволяє будь-кому створювати власні програми для обміну повідомленнями, які працюють у хмарі Telegram. Користувачі програм створених на базі Telegram API можуть спілкуватися з будь-ким у самому

застосунку Telegram. Дана технологія дозволить спростити створення швидких, безпечних і багатофункціональних додатків Telegram [21].

Використання бібліотеки TDLib вирішує питання про деталі реалізації мережі, шифрування та локальне зберігання даних, щоб розробники могли приділяти більше часу дизайну, адаптивним інтерфейсам та красивій анімації [22].

TDLib підтримує всі функції Telegram і робить розробку додатків Telegram легкою на будь-якій платформі. Його можна використовувати на Android, iOS, Windows, macOS, Linux і практично в будь-якій іншій системі. Бібліотека сумісна з будь-якою мовою програмування, яка може виконувати функції C; він також має власні прив'язки до Java та C#.

Більш того, TDLib залишатиметься стабільним при повільних і ненадійних підключеннях до Інтернету та гарантує, що всі оновлення будуть доставлені в правильному порядку. Усі локальні дані шифруються за допомогою ключа шифрування, наданого користувачем.

2.5.4 Технологія WebRTC

WebRTC — це браузерна технологія, яку можна використовувати для додавання медіа-зв'язку в реальному часі безпосередньо між браузером і різноманітними пристроями. Ця технологія дозволяє проводити голосові та відео зв’язки, працюючи на веб-сторінках [23].

WebRTC сьогодні доступний у всіх сучасних браузерах. Її також можна інтегрувати в програму або вбудований пристрій, не потребуючи браузера взагалі. Завдяки цій технології є можливість отримати доступ до мікрофона свого пристрою, камери, яка є на телефоні чи ноутбуці, або це може бути сам екран. Також можна зафіксувати відображення користувача, а потім поділитися цим екраном або записати його віддалено. Все відбувається в режимі реального часу, забезпечуючи живу взаємодію.

Саме використання відео-конференцій напряму у браузері без додаткових застосунків дозволить належним чином підтримувати ефективну комунікацію між учасниками навчального процесу.

Роботу технології представлено на прикладі дзвінка між двома абонентами за допомогою браузера (рис. 2.2).

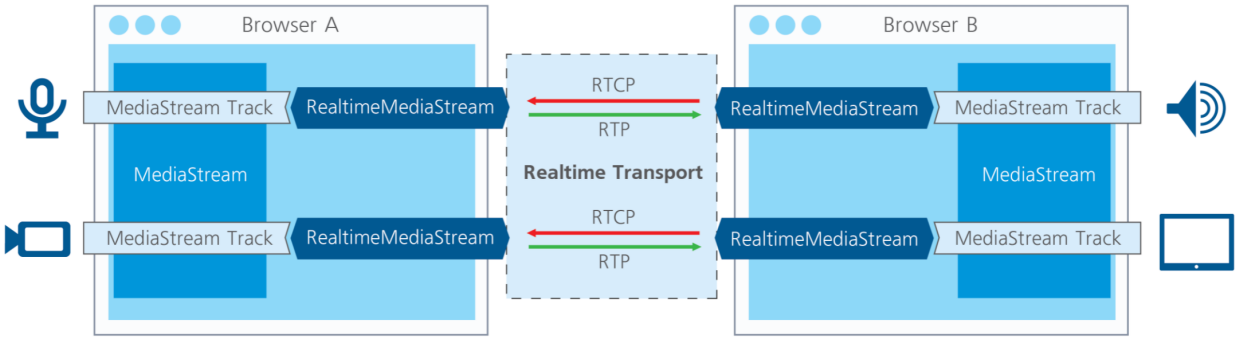


Рисунок 2.2 – Приклад використання технології WebRTC

Виділимо переваги використання даної технології:

– відсутність встановлення додаткового ПЗ;

– наявність відкритого вихідного коду, що дає змогу використовувати її безкоштовно для комерційних чи приватних цілей. Оскільки технологія постійно розвивається та вдосконалюється, то можна зрозуміти, що вона буде працювати протягом багатьох років;

– не обмежується лише браузерами, оскільки технологія також доступна для мобільних додатків Вихідний код є портативним і вже використовувався у багатьох мобільних додатках. Пакет SDK доступний як для мобільних, так і для вбудованих середовищ, тому можна використовувати WebRTC для запуску будь-де;

– додаткова можливість використовувати технологію для створення служби групових дзвінків, додавання запису або використання лише для доставки даних;

– висока якість зв’язку;

– високий рівень безпеки, завдяки підтримці протоколу HTTPS.

Висновки до розділу

У першій частині цього розділу наведено особливості типів програмного забезпечення та їх переваги з недоліками. Після проведеного аналізу, було обґрунтовано вибір типу програмного забезпечення, що буде використовуватись у подальшій розробці.

У другому підрозділі було приділено увагу реляційним базам даних та систем їх керування. До розгляду було взято декілька розповсюджених СУБД, де було наведено порівняльний аналіз з виділенням переваг та недоліків кожної та обґрунтовано вибір остаточної для подальшого використання у програмному забезпеченні.

Третя частина розділу містить опис мов програмування для розробки серверної частини застосунку та обґрунтування вибору остаточної на основі перелічених недоліків та переваг.

Для розробки клієнтської частини застосунку, у четвертому підрозділі був обґрунтований вибір мови програмування Typescript з переліченням основних її переваг над мовою Javascript.

Також в останньому підрозділі було наведено додаткові технології, які плануються використати при розробці програмного забезпечення і які є основним фундаментом для вирішення проблеми ефективності в комунікації учасників навчального процесу.

3 ПОБУДОВА АРХІТЕКТУРИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Побудова архітектури є важливим етапом, оскільки вона робить процес розробки програмного забезпечення більш простим та ефективним. Програму з гарно побудованою архітектурою легше розширювати та модифіковувати, а також тестувати, налагоджувати та розуміти.

3.1 Вибір архітектурного шаблону

На сьогодні одними з найбльш розповсюджених підходів до побудови архітектури є багаторівнева, яка користується популярністю вже багато років і на її основі побудовано тисячі застосунків, що працюють сьогодні, та мікросервісна, яка є переважно новою та користується широким попитом у повсякденні. Таким чином, доцільно провести огляд кожного архітектурного підходу та визначити його основні переваги і недоліки.

3.1.1 Багаторівнева архітектура

Даний архітектурний шаблон інакше відомий як n-рівневий шаблон архітектури. Це шаблон є де-факто стандартом для більшості програм Java ентерпрайзу і тому широко використовується архітекторами та розробниками. Шаблон багаторівневої архітектури є оптимальним вибором для реалізації більшості застосунків завдяки своїй структурі [24].

З точки зору архітектури, рівень є логічною одиницею, яка розділяє певну роль і відповідальність у програмі. Кожен рівень керує власними програмними залежностями. Вищий рівень може використовувати служби нижчого рівня, але не навпаки. Рівень – це фізична одиниця, на якій виконується код; наприклад, веб-сервер або база даних. Кожен рівень може бути розміщений на окремому рівні, але це не обов’язково. На одному рівні можна розмістити кілька шарів. Масштабованість, ремонтопридатність і стійкість підвищуються при фізичному розділенні рівнів, однак затримка збільшується через додатковий мережевий зв’язок. Традиційна багаторівнева архітектура програмного забезпечення має три і чотири рівні. Розглянемо наведену архітектурну схему (рис. 3.1).

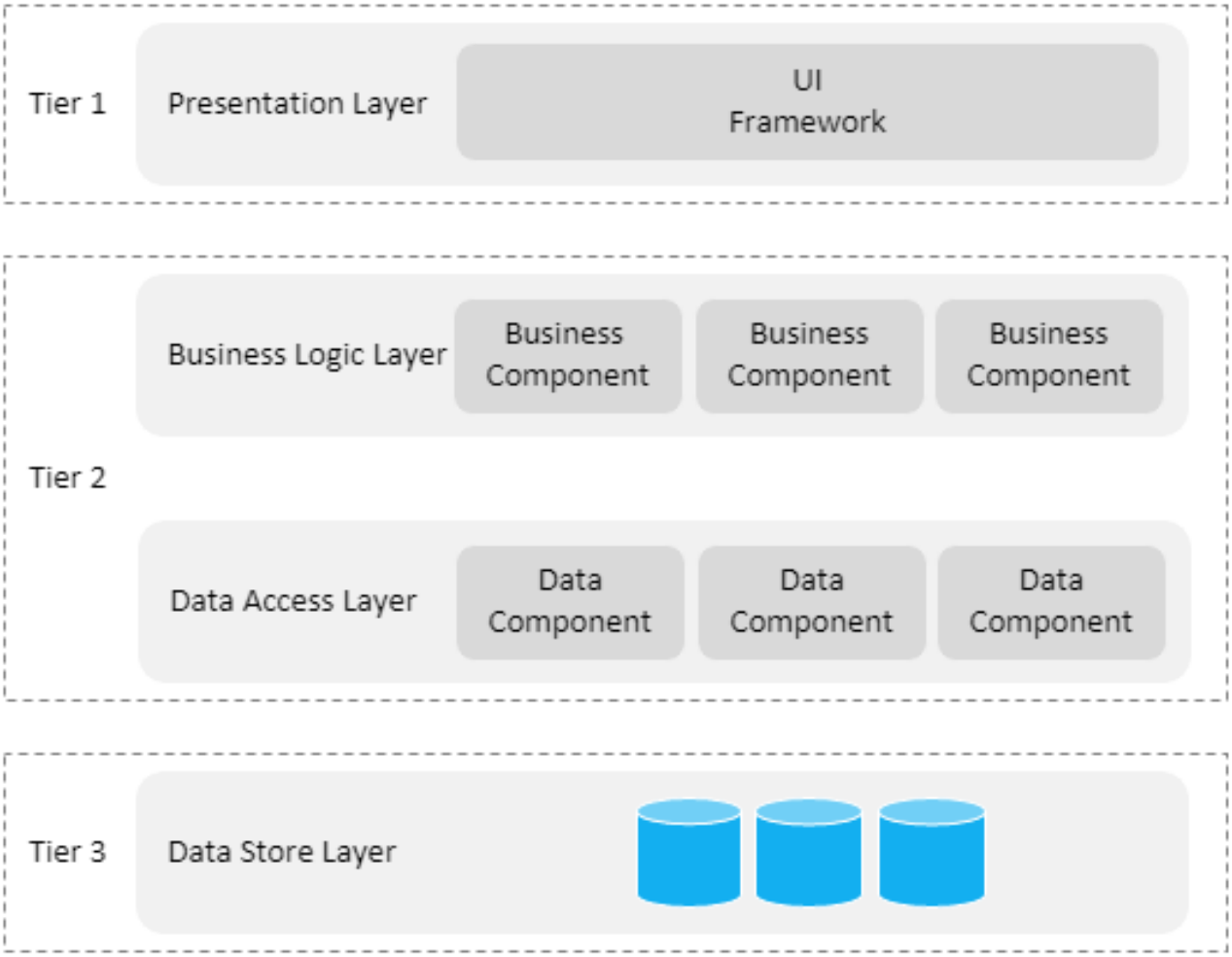


Рисунок 3.1 – Схема багаторівневої архітектури

Чотири стандартні рівні описані нижче:

– рівень представлення даних. Цей рівень містить усі користувацькі інтерфейси, доступні для користувача. Він може надавати різні типи інтерфейсів користувача, а саме веб-додатки, настільні та нативні мобільні додатки;

– рівень бізнес-логіки. Цей рівень обробляє всю бізнес-логіку, перевірки та бізнес-процеси;

– рівень доступу до даних. Цей рівень відповідає за взаємодію з базою даних. Він також відомий як рівень стійкості;

– рівень сховища даних. Цей рівень є фактичним сховищем даних для програми.

Завдяки введенню різних рівнів для програмних компонентів розділення проблем різко збільшується, що покращує простоту, ремонтопридатність і можливість тестування компонентів. Рівні описуються наступним чином.

Рівень 1. Рівень представлення даних. На цьому рівні розміщується базова кодова база, тобто рівень представлення. Це найвищий рівень програми, і, по суті, це рівень, до якого користувачі мають доступ безпосередньо.

Рівень 2. Рівень додатків. На цьому рівні розміщується базова кодова база, тобто рівень бізнес-логіки та рівень доступу до даних. Він також відомий як середній ярус.

Рівень 3. Рівень даних. На цьому рівні розміщується сховище даних, тобто рівень сховища даних. Бази даних, файлова система, сховище великих об’єктів, база даних документів тощо – це приклади ресурсів, знайдених у сховищі даних.

Існує два способи реалізації багаторівневої архітектури програмного забезпечення, а саме закрита і відкрита багаторівнева архітектура.

Закрита багаторівнева архітектура. У закритій багаторівневій архітектурі рівень може викликати лише наступний рівень безпосередньо під ним. Розглянемо наведену нижче схему закритої багаторівневої архітектури (рис. 3.2).

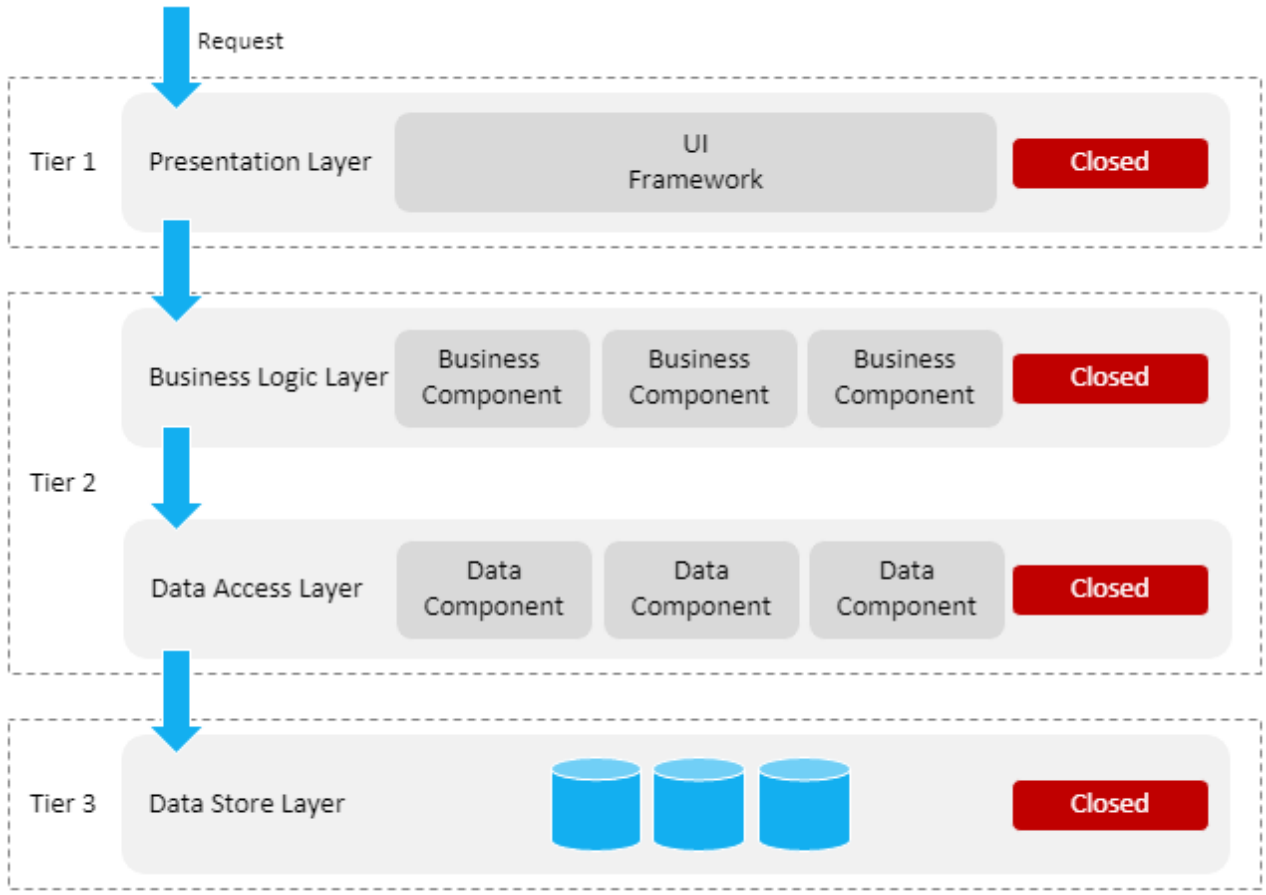


Рисунок 3.2 – Схема закритої багаторівневої архітектури

У наведеній вище діаграмі архітектури кожен рівень позначено як закритий. Це означає, що для того, щоб запит просунувся до самого нижнього рівня, запит повинен проходити через кожен рівень. Наприклад, коли запит ініціюється через рівень представлення даних, запит спочатку проходить через рівень бізнес-логіки, поруч із рівнем доступу до даних і, нарешті, до рівня зберігання даних.

Метою закритої багаторівневої архітектури є гарантувати, що рівні повністю ізольовані один від одного, тобто зміни, внесені в одному рівні архітектури, не впливають на інші рівні. Іншими словами, рівні слабо пов’язані, коли кожен рівень майже не знає інших рівнів. Зауважте, що при реалізації закритої багатошарової архітектури це обмежує залежності між рівнями, таким чином вводячи непотрібний мережевий трафік, оскільки один рівень просто передає запити наступному.

Відкрита багаторівнева архітектура. У відкритій багаторівневій архітектурі рівень може викликати будь-який рівень під ним, за умови, що рівень під ним позначено як відкритий. Розглянемо відкриту діаграму багаторівневої архітектури (рис. 3.3).

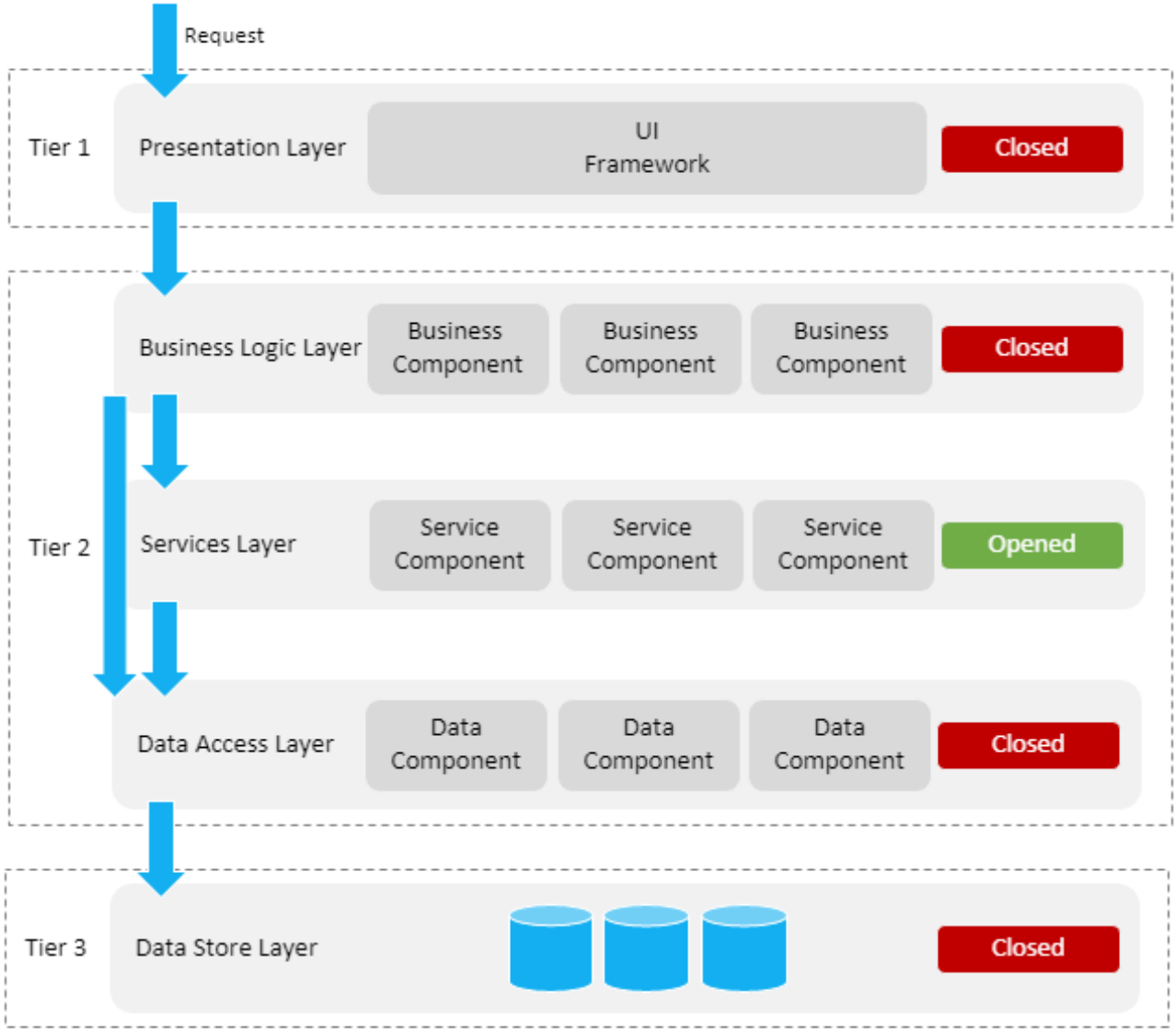


Рисунок 3.3 – Схема відкритої багаторівневої архітектури

На діаграмі вище було представлено новий рівень сервісів. Цей рівень забезпечує ряд компонентів обслуговування. Не всі компоненти служби на рівні сервісів взаємодіють з базою даних. З цієї причини шар позначено як відкритий. Це означає, що рівень бізнес-логіки має доступ як до рівня послуг, так і до рівня доступу до даних, тобто рівень бізнес-логіки не повинен проходити через рівень послуг, щоб використовувати рівень доступу до даних.

Виконавши огляд архітектурного шаблону, визначимо далі його переваги:

– легкість тестування. Цей шаблон дозволяє легко проводити тестування компоненти одного рівня, оскільки кожен рівень є ізольованим один від одного;

– простота реалізації;

– можливість розробки слабозв’язаних систем;

– гнучкість розгортання. Різні компоненти програми можна незалежно розгортати, підтримувати та оновлювати у різний час;

– можливість налаштувати різні рівні безпеки для різних компонентів, розгорнутих на різних коробках. Багаторівнева архітектура дозволяє захистити частини програми за брандмауером і зробити інші компоненти доступними з Інтернету;

– відсутність конфліктів між рівнями. Кожен рівень виконує обмежений набір функцій і не має уяву про те, як влаштовані інші рівні. Тому компонента будь-якого рівня може бути змінена без ризику виникнення конфліктів з іншими рівнями.

Недоліками багаторівневої архітектури є:

– низька продуктивність. Виконання будь-якого запиту є не досить ефективним у зв’язку з тим, що запит проходить усі рівні архітектури, де на кожен перехід витрачається додатковий час;

– складність масштабування;

– складність виявлення помилок. Перш ніж потрапити до БД, інформація проходить через усі рівні і у разі коли ця інформація пошкоджується на якомусь етапі, то для пошуку помилки доводиться аналізувати кожен рівень окремо.

3.1.2 Мікросервісна архітектура

На відміну від традиційних архітектурних стилів, які спрямовані на створення програмного забезпечення як єдиного блоку, архітектура мікросервісів заснована на модульному підході. Таким чином, програми розробляються як пакети, що складаються з кількох мікросервісів, які можна

незалежно розгортати та керувати. Незважаючи на те, що немає чіткого визначення мікросервісів як таких, є певні характеристики, які пов’язані з цим революційним підходом до розробки програмного забезпечення. Однією з головних характеристик є те, як архітектурний стиль поєднує складний додаток. Програмне забезпечення розбито на кілька компонентів, які створені як сервіси, які можна замінити та оновити. Ці послуги можна визначити як процеси, які часто спілкуються через мережу за допомогою протоколів, які не залежать від технологій. Мікросервіси також створені відповідно до можливостей бізнесу. А оскільки вони є децентралізованими та малими за розміром, вони пропонують велику гнучкість, коли справа доходить до мов програмування, обладнання та програмного середовища [25]. На рис. 3.4 наведена спрощена схема мікросервісної архітектури.

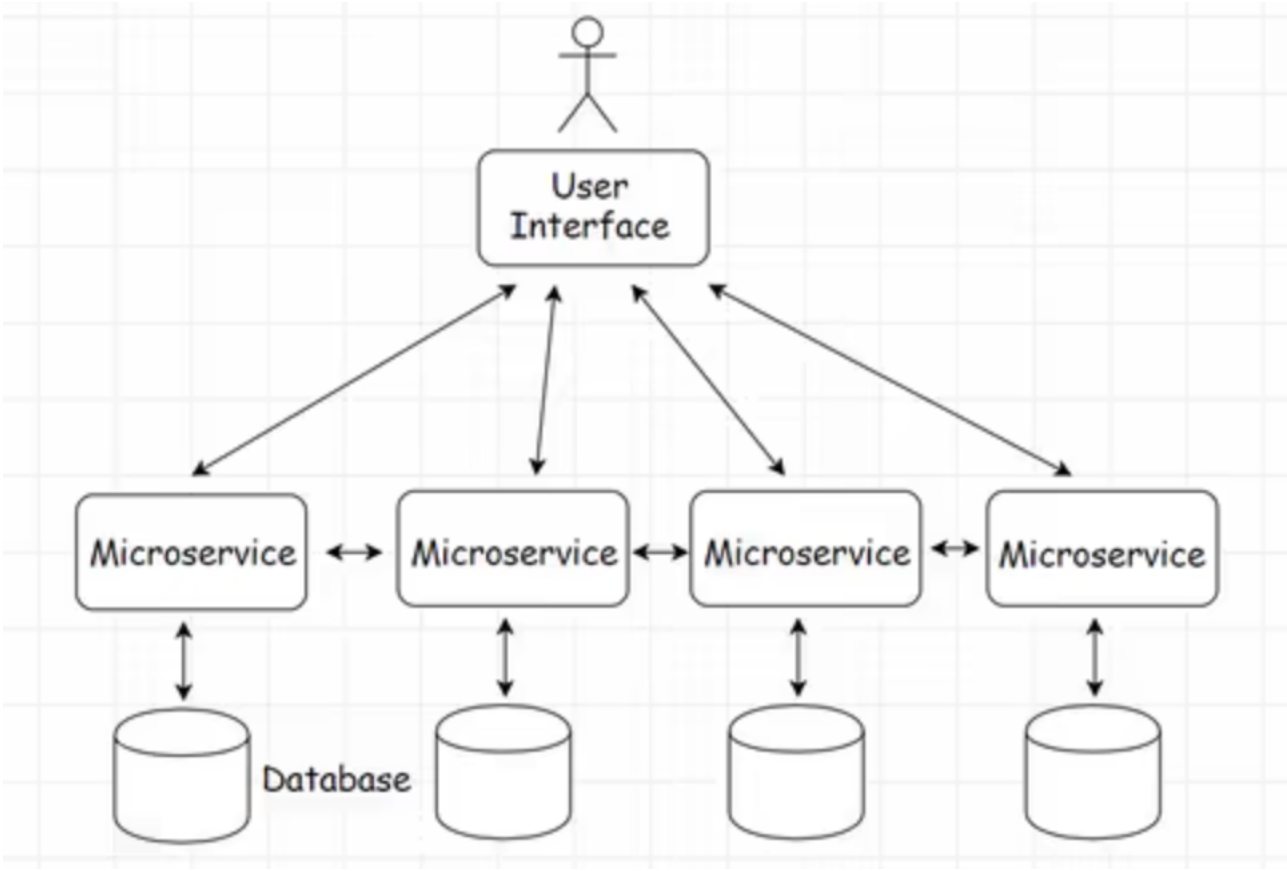


Рисунок 3.4 – Спрощена схема мікросервісної архітектури

Ідея слабо пов’язаних між собою служб, які зручно коригуються і піддаються тестуванню, звучить інтригуюче для компаній, які стикаються з проблемами IT через складні програми. Але перш ніж викорінювати систему,

важливо зрозуміти, як насправді працюють мікросервіси. Щоб створити функціонуючу модульну архітектуру, бізнес повинен спочатку визначити свої можливості та визначити послуги, які працюватимуть незалежно один від одного. Діаграма архітектури мікросервісів потрібна для виявлення залежностей, які заважають гнучкості кожної служби. Важливо переконатися, що помилка в окремому мікросервісі не вплине на всю систему. Але незважаючи на те, що мікросервіси розроблені для незалежної роботи, вони іноді все ще залежать від інших мікросервісів і потребують зв’язку один з одним. Цей зв'язок відбувається за допомогою інтерфейсів прикладного програмування; синхронні або асинхронні протоколи, такі як HTTP/REST або AMQP. В ідеалі кожен мікросервіс має власну базу даних, щоб децентралізувати відповідальність та керувати оновленнями окремо. При використанні шаблону «База даних на послугу» для підтримки узгодженості даних створюється сага, яка являє собою транзакцію, яка охоплює кілька служб. Однак нерідко кожен мікросервіс з часом розвиває все більше і більше сервісних залежностей. Це нормальне явище, коли послуги починають рости та впливають на кількість екземплярів, місць розташування та протоколів. Шлюзи API допомагають уникнути проблем, які виникають внаслідок цих змін. І ось як це працює: шлюз API приймає всі запити API, а потім направляє кожен до найбільш відповідного мікросервісу. Він також викликає серверні служби та агрегує результати, щоб визначити найкращий шлях. Він служить точкою входу для кожного запиту, організовує весь вхід і створює спрощену роботу користувача.

Існує ряд переваг, які мікросервіси можуть запропонувати. Нижче наведено основні:

– наявність доступності. Оскільки мікросервіси розгортаються окремо і спілкуються через API, код і програми стають доступнішими для співробітників поза розробкою. Підхід модульного проектування також допомагає зрозуміти різні послуги, які пропонує компанія, включаючи їхні конкретні функції, не турбуючись про інші частини;

– швидке відслідковування помилок. Якщо архітектура мікросервісів реалізована правильно, знаходити та ізолювати помилки стає набагато легше. Щоразу, коли виникає помилка, розробники точно знають, де шукати, що економить дорогоцінний час і зусилля. А виправлення лише однієї незалежно запущеної служби означає, що інший код не буде порушено, і немає необхідності розгортати весь додаток;

– адаптивність. Мікросервіси слабо пов’язані і мають стандартизований спосіб зв’язку один з одним. Але оскільки вони створені окремо, вони можуть не використовувати одну мову програмування чи технологію. Це означає, що кожен мікросервіс може бути розроблений таким чином, щоб він найкраще відповідав функціям, які він повинен виконувати;

– легко масштабувати та інтегрувати зі сторонніми службами. Вимоги бізнесу можуть змінюватися, що створює потребу додавати нові функції до пропонованих послуг. У традиційних архітектурних підходах, які розглядають програмне забезпечення як єдине ціле, це означає втручання в всю бізнес-логіку та потенційне порушення існуючих функцій. Завдяки модульному підходу архітектура мікросервісів дозволяє уникнути цієї проблеми.

Незважаючи на значну низку переваг, така архітектура має ряд недоліків:

– складність тестування. Через розподілене розгортання тестування може стати складним і затратним по часу;

– архітектура створює додаткову складність, оскільки розробникам доводиться зменшувати відмовостійкість, затримку мережі та працювати з різними форматами повідомлень, а також з балансуванням навантаження;

– складність інтеграції та управління у разі збільшення кількості сервісів;

– наявність додаткового налаштування комунікації, що ускладнює процес розробки. Розробникам доводиться докладати додаткових зусиль для впровадження механізму зв’язку між сервісами;

– складність підтримки розподілених транзакцій. Обробляти випадки використання, які охоплюють більше одного мікросервісу без використання

розподілених транзакцій, не тільки складно, але також вимагає спілкування та співпраці між різними командами.

3.1.3 Результати аналізу

Провівши детальний огляд обох архітектурних підходів до розробки програмного забезпечення було вирішено обрати багаторівневу архітектуру, яка вже за багато років довела свою актуальність і показала свої переваги, незважаючи на новий тренд побудови застосунків з використанням мікросервісної архітектури та її переваг.

Основними завадами, які стали на шляху вибору мікросервісної архітектури, були ті, що не дозволяли сконцентруватися на реалізації лише функціональної частини застосунку, потрібно було також приділяти увагу налаштуванню взаємозв'язку та комунікації між сервісами, що не входить до головного пріоритету для розробки програмного забезпечення. Також ще однією причиною стало те, що при передачі даних між мікросервісами по мережі затрати часу збільшуються у рази, що можуть призвести до проблем у застосунку, оскільки він має обробляти дані у реальному часі при використанні технології WebRTC.

І на останок, у програмному забезпеченні не планується створювати занадто велику кількість бізнес логіки, що дозволяє ефективно застосувати багаторівневу архітектуру, не беручи до уваги деякі її недоліки.

3.2 Опис архітектурного рішення

На рис. 3.5 наведена схема побудованої багаторівневої архітектури, яка містить основні компоненти програмного забезпечення. Як видно з рисунку, в архітектурі присутні зв'язані між собою рівні, на кожному з яких виконується певна задача.

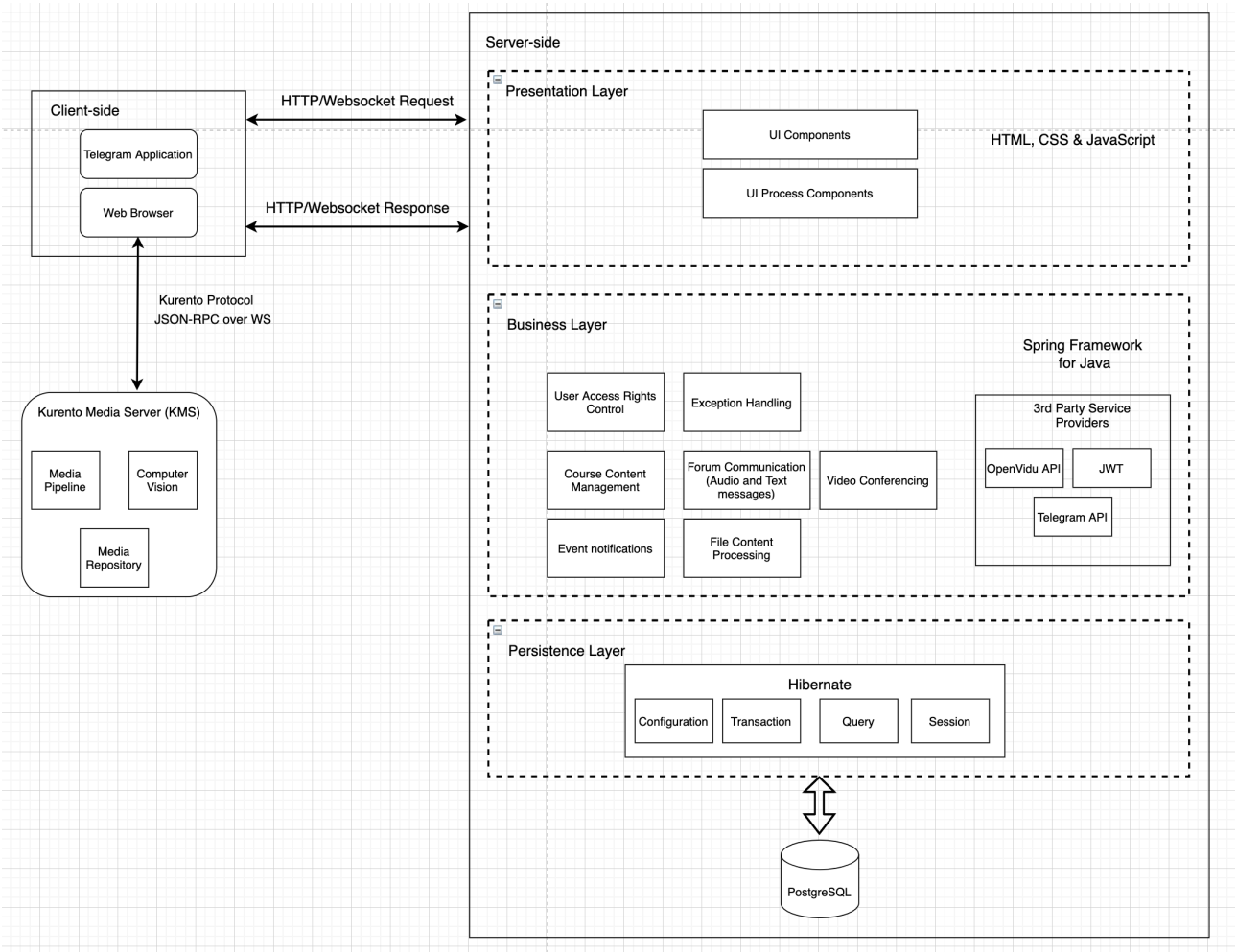


Рисунок 3.5 – Схема побудованої багаторівневої архітектури з використанням всіх компонентів

Оскільки на основі побудованої архітектури буде створено веб-застосунок, то для його розробки можна застосувати архітектурний стиль REST, який буде відповідати за взаємодію компонентів застосунку через мережу. Для розробки REST API буде використано формат даних JSON, який буде передаватися між компонентами [26]. Для того, щоб застосунок був продуктивним та легко масштабувався необхідно притримуватися принципам проектування REST, які наведені нижче:

– наявність єдиного інтерфейсу. Ресурс у системі повинен мати лише один логічний URI, який повинен забезпечувати спосіб отримання відповідних даних. Будь-який окремий ресурс не повинен бути занадто великим і містити все у своєму представленні. За потреби ресурс повинен містити посилання, що вказують на відносні URI для отримання пов’язаної інформації. Крім того,

представлення ресурсів у всій системі має відповідати певним інструкціям, таким як конвенції щодо імен, формати посилань або формат даних;

– розділення клієнта та сервера. Цей принцип означає, що клієнтські програми та серверні програми повинні працювати окремо без будь-якої залежності один від одного. Клієнт повинен знати лише URI ресурсу, при цьому сервер не повинен модифікувати клієнтський застосунок, а лише передавати дані через HTTP;

– відсутність збереження стану. Сервер не повинен зберігати нічого про останній HTTP-запит, зроблений клієнтом, а повинен лише розглядати кожен запит як новий. Якщо клієнтська програма має визначати стан кінцевого користувача, де користувач входить один раз і після цього виконує інші авторизовані операції, тоді кожен запит від клієнта повинен містити всю інформацію, необхідну для обслуговування запиту, включаючи автентифікацію та деталі авторизації;

– підтримка кешування. Кешування забезпечує підвищення продуктивності на стороні клієнта та кращий простір для масштабованості сервера, оскільки зменшується навантаження. Кешування може бути реалізовано на стороні сервера або клієнта;

– наявність багаторівневої архітектури. REST дозволяє використовувати багаторівневу архітектуру системи, де, наприклад, API розгортається на сервері A, дані зберігаються на сервері B і запити аутентифікуються на сервері C. Зазвичай клієнт не може визначити, чи підключений він безпосередньо до кінцевого сервера чи до посередника на цьому шляху.

За побудованою архітектурою виконаємо детальний опис кожного рівня.

3.2.1 Рівень представлення даних

Даний рівень виконує взаємодію з користувачем застосунку. Основною метою цього рівня є відображення інформації та отримання її від користувача. На цьому рівні наявні компоненти користувацького інтерфейсу, такі як файли

стилів та html, код javascript-у. Рівень представлення даних містить контролери, сервіси обробки динамічних сторінок та моделі представлень.

На рівні представлення даних використовується фреймворк Angular, який дозволяє розширити веб-застосунки на основі шаблону MVC, який розділяє програмну логіку на три пов'язані між собою елементи [27]. На рис. 3.6 зображено схему взаємодії компонентів MVC.

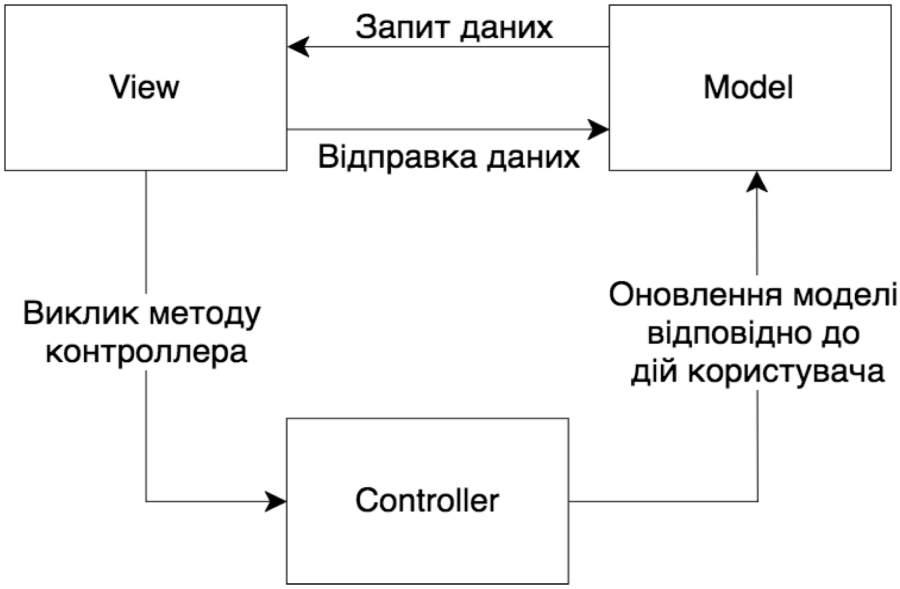


Рисунок 3.6 – Схема взаємодії компонентів MVC

Реалізація шаблону MVC на стороні клієнта отримує дані від компонентів на сервері через веб-сервіс RESTful. Мета контролера і представлення – оперувати даними в моделі для виконання маніпуляцій DOM, щоб створювати елементи HTML і керувати ними, з якими може взаємодіяти користувач. Ці взаємодії повертаються в контролер, замикаючи цикл для формування інтерактивної програми.

Моделі в MVC містять дані, з якими працюють користувачі. Наприклад, об'єкт Controller буде отримувати інформацію про користувача з бази даних. Він маніпулює даними та надсилає назад до бази даних або використовує її для відтворення тих самих даних. Моделі відповідають на запит від представлень, а також відповідають контролерам щодо оновлення відповідно до дій користувача. Моделі Angular ефективно використовують логіку на стороні сервера і

запускають її через веб-сервіс RESTful, оскільки браузер не підтримує збереження даних, і легше отримати потрібні дані через технологію Ajax.

Переваги забезпечення ізольованості моделі від контролера та представлень полягають у тому, що можна легше виконувати тестування логіки, а також простіше і легше покращити й підтримувати програму. Моделі домену містять логіку отримання та зберігання даних, а також логіку для операцій створення, читання, оновлення та видалення. Використовуються також окремі моделі для запитів і модифікації даних, відомі з використанням шаблону розділення відповідальності команд і запитів (CQRS).

Контролери, які є відомими як компоненти в Angular, є ланкою між моделлю даних і представленнями. Компоненти додають бізнес-логіку, необхідну для представлення деяких аспектів моделі та виконання операцій над нею. Компонент, який слідує за шаблоном MVC, повинен містити:

– логіку, необхідну для налаштування початкового стану шаблону;

– логіку, необхідну шаблону для представлення даних з моделі;

– логіку, необхідну для оновлення моделі на основі взаємодії з користувачем.

Компонент не повинен містити:

– логіку, яка маніпулює DOM;

– логіку, яка керує збереженням даних (це робота моделі).

Модель домену не є єдиними даними в додатку Angular. Компоненти можуть створювати дані перегляду (також відомі як дані моделі представлення або моделі представлення), щоб спростити шаблони та їх взаємодію з компонентом.

Подання, відомі як шаблони в Angular, визначаються за допомогою елементів HTML та прив’язками даних. Саме прив’язки даних робить Angular настільки гнучким, що перетворює елементи HTML на основу для динамічних веб-додатків.

Шаблони не повинні:

– містить логіку та розмітку, які необхідні для представлення даних користувачеві;

– містити складну логіку (складну логіку краще помістити в компонент або один з інших будівельних блоків Angular, таких як директиви, служби або канали);

– містити логіку, яка створює, зберігає або маніпулює моделлю домену.

Шаблони можуть містити логіку, але вона повинна бути простою і не використовуватися часто. Поміщення в шаблон будь-чого, крім найпростіших викликів методів або виразів, ускладнює тестування та обслуговування загальної програми.

3.2.2 Рівень бізнес-логіки

Задачею рівня бізнес-логіки є забезпечення взаємодії з рівнем доступу до даних та опрацьовування основних маніпуляцій з даними, виконуючи відповідні бізнес-правила. Рівень бізнес-логіки містить основну логіку роботи програми, де також реалізується оброблення даних будь-якого типу, наприклад обробка файлів та зображень будь-якого формату, робота з різними API, забезпечення механізмів автентифікації і авторизації користувача тощо.

На цьому рівні міститься серверна частина застосунку, в якому створюються класи сервіси, що отримують HTTP-запити від рівня представлення даних, перепаковуючи отримані значення, обробляють їх за відповідними бізнес-правилами та передають далі до відповідної веб-служби на рівні доступу.

З точки зору функціональності, на цьому рівні реалізована обробка файлових документів та можливих помилок, які виникають у всій програмі, підтримка відео конференцій завдяки роботі з відповідними API, надсилання нотифікацій про новостворені події, керування навчальними курсами шляхом виконання стандартних CRUD операцій, а також підтримка комунікації завдяки аудіо та текстовим повідомленням шляхом використання протоколу WebSocket [29].

Також проведена автоматизація процесу передачі інформації на платформу Telegram для розширення можливостей рівня бізнес-логіки. Інтеграція з Telegram API дозволяє автоматично дублювати новостворену інформацію у веб-застосунку на платформу Telegram.

Для підвищення ефективності роботи веб-застосунку з роботою Telegram застосовано шаблон проектування “Спостерігач”, який дозволяє об’єктам слідкувати і реагувати на події, які відбуваються в інших об’єктах, створюючи можливість підписки. При використанні даного шаблону видавцями називають ті об’єкти, які мають певний стан. Інші об’єкти, які займаються відслідковуванням цього стану називають підписниками. Як приклад, у якості видавця виступає дошка оголошень у веб-застосунку, а підписниками є студенти, які приєдналися до навчального курсу. Коли на дошці оголошень з’являється нова публікація, то всім учасникам курсу, які прив’язали свої дані згідно даним з платформи Telegram, буде надсилатися додаткове повідомлення у Telegram чат. Використовуючи даний шаблон, взаємодія з платформою Telegram відбувається в асинхронному потоці програми, що дозволяє уникнути блокування інтерфейсу веб-застосунку.

Використання бібліотеки з відкритим вихідним кодом OpenVidu на рівні бізнес-логіки дало змогу проводити відео конференцію у реальному часі. Бібліотека використовує API на основі технології WebRTC, що дозволяє виконувати передачу потокових відео та аудіо даних між браузерами.

3.2.3 Рівень доступу до даних

На рівні доступу до даних використовуються компоненти, які спілкуються з базою даних, виконуючи базові CRUD операції за допомогою фреймворку Hibernate. Також цей рівень містить сутності, які є відображенням таблиць бази даних. Класи репозиторії, що знаходяться на цьому рівні використовуються у рівні бізнес-логіки для комунікації з базою даних.

Рівень доступу до даних базується на використанні технології Hibernate, яка забезпечує сумісність із будь-якою БД. Ця технологія дозволяє працювати з

базою даних за допомогою представлення таблиць баз даних у вигляді класів Java. Hibernate абстрагує базу даних і дозволяє визначити властивості підключення до неї у файлі під назвою hibernate.properties [28]. Як частина архітектури програми, рівень ORM відповідає за управління перетворенням програмних об’єктів для взаємодії з таблицями і стовпцями в реляційній базі даних. На даному рівні ORM перетворює класи та об’єкти Java, щоб їх можна було зберігати та керувати ними в реляційній базі даних.

Використовуючи Java анотації, виконується зіставлення сутностей Java класів з таблицями БД. У класах налаштовані обмеження та відношення між один одним, що утворюють реляційні зв’язки таблиць бази даних.

Для виконання CRUD операцій над базою даних з допомогою Hibernate використовується мова HQL, яка дозволяє формувати SQL запити, використовуючи лише класи та його атрибути. Таким чином, досягається об’єкто-орієнтованість мови і не потрібно приділяти увагу синтаксису SQL, що може відрізнятися в різних базах даних.

3.2.4 Рівень бази даних

Рівень даних контролює сервери, на яких зберігається інформація; на цьому рівні запускається система керування реляційною базою даних на сервері баз даних або мейнфреймі та містить логіку зберігання даних. Рівень даних зберігає дані незалежними від серверів додатків та покращує масштабованість і продуктивність.

На цьому рівні використовується система керування базою даних PostgreSQL. Завдяки використанню цієї СКБД можна керувати даними, а саме зберігати та отримувати їх, а також керувати оновленнями, надавати одночасний доступ для кількох процесів для рівня доступу до даних, забезпечувати безпеку та цілісність даних та надавати послуги підтримки, такої як резервне копіювання даних.

На рис. 3.7 наведена схема таблиць бази даних, які використовуються у застосунку.

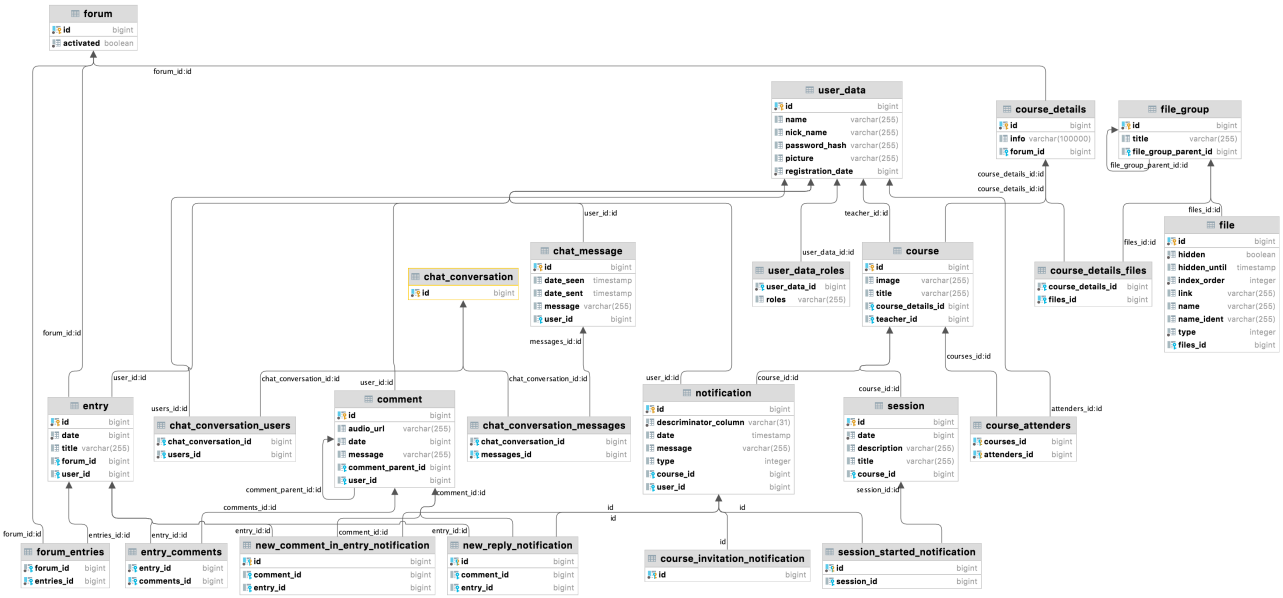


Рисунок 3.7 – Схема таблиць бази даних

3.2.5 Опис архітектури Kurento

Традиційні програми WebRTC є одноранговими, тому завдяки браузерам вони можуть працювати без посередництва будь-якої інфраструктури. Цього достатньо для створення базової програми, але такі функції, як групове спілкування, запис медіа-потоку, медіа-мовлення або перекодування медіа складно реалізувати поверх цього. З цієї причини було доцільним використання медіа-сервера.

Kurento Media Server (KMS) представляє собою медіа-сервер WebRTC з відкритим вихідним кодом і набором клієнтських API, що спрощують розробку відео додатків для веб-платформ. Модульна архітектура Kurento дотримується цілісного підходу, яка надає можливість інтеграції усіх типів медіа-можливостей на стороні сервера. Ця архітектура відповідає 5 основним характеристикам модульності: ізоляції, абстракції, можливості повторного використання, компонування та масштабованості.

Щоб забезпечити ізоляцію, абстракцію, можливість повторного використання, Kurento вводить концепцію Media Element. Медіа-елемент — це модуль, який містить певну медіа-можливість. Наприклад, медіа-елемент під назвою WebRtcEndpoint має здатність надсилати та отримувати медіапотоки

WebRTC, медіа-елемент під назвою RecorderEndpoint має можливість запису у файлову систему будь-якого носія потоків, які він отримує. На рис. 3.8 відображено набір інструментів, що надає Kurento як частину API.

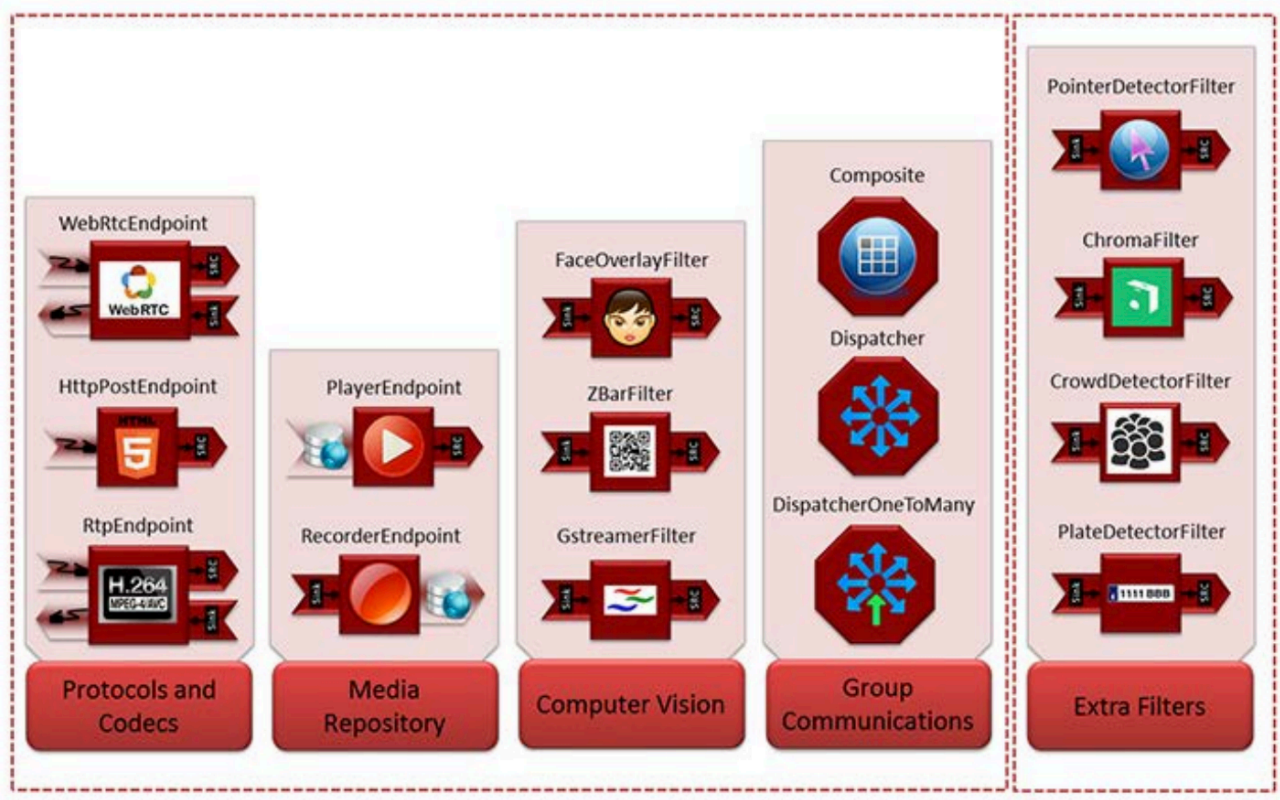


Рисунок 3.8 – Набір інструментів Kurento Media Elements

Для оптимізації Kurento Media Server реалізовано в технологіях низького рівня на основі GStreamer споживання ресурсів. Ключовим інгредієнтом для забезпечення модульності Kurento є концепція Media Element. Щоб зрозуміти, як ці компоненти працюють внутрішньо, варто звернути увагу на низькорівневі деталі їх архітектури, які зображено на рис. 3.9.

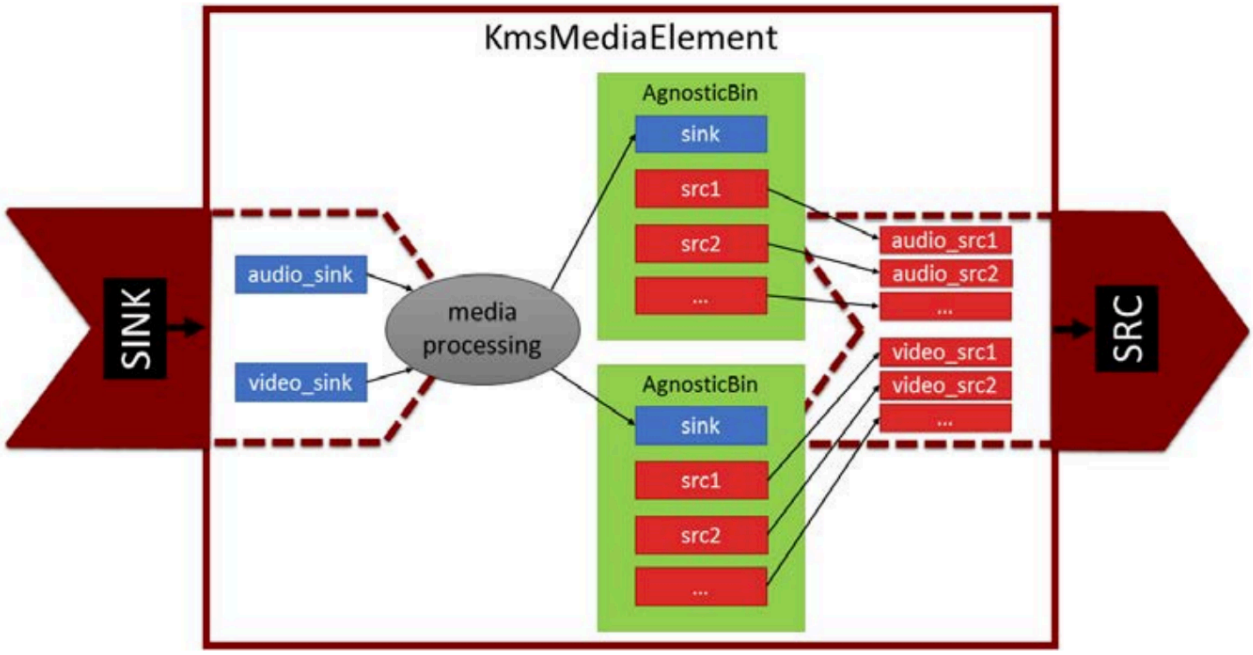


Рисунок 3.9 – Внутрішня структура медіа-елемента Kurento

Всередині медіа-елемента знаходиться компонент під назвою KmsMediaElement, який є базовим класом усіх реалізацій Media Element., KmsMediaElement забезпечує вміння керувати колодками та бункерами. Також цей елемент можна підключити до іншого KmsMediaElement, створюючи таким чином шлях, по якому протікають медіа. Кошик — це об’єкт GStreamer, що представляє ланцюжок компоненти обробки медіа низького рівня. Можливості обробки медіа KmsMediaElement (наприклад, виявлення натовпу, накладання обличчя тощо). AgnosticBin відповідає за надання агностичних носіїв, тобто керування трансформації та адаптації, необхідних для забезпечення безперебійного взаємозв’язку KmsMediaElements.

Використовуючи API Kurent, можна застосовувати всі можливості та функції Kurento медіа серверу. Цей API реалізується за допомогою бібліотек під назвою Kurento Clients, які обмінюються за допомогою JSON-RPC через WebSocket для керування медіа-сервером Kurento. Формат цих повідомлень реалізує протокол Kurento.

Висновки до розділу

У даному розділі було приділено увагу до побудови архітектури програмного забезпечення.

У першому підрозділі обґрунтовано вибір архітектурного шаблону. В ньому було описано 2 архітектурних підходи, а саме багаторівневу та мікросервісну архітектуру, які є найбільш розповсюдженими на даний момент. Для кожного типу архітектури було проведено аналіз та виявлено їх основні переваги та недоліки. Результатом вибору стала багаторівнева архітектура, яка має ряд переваг саме для реалізації поставлених задач, висунутих до програмного забезпечення.

У другому підрозділі було описано безпосередньо архітектурне рішення з використанням усіх його складових та компонент. За взаємодією між різними компонентами через мережу було застосовано архітектурний стиль REST з наведенням його переваг. До того ж, наведений детальний опис кожного рівня багаторівневої архітектури. На кожному рівні було описано внутрішню складову програмної реалізації, в якій застосовуються обрані технології, зосереджена основна логіка роботи програми та наведена схема побудованої бази даних.

В останньому підрозділі наведено опис архітектури медіа-серверу Kurento, для реалізації обробки аудіо та відео потоків у реальному часі.

4 ОПИС РОБОТИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Програмне забезпечення реалізовано з використанням веб-технологій, яке надає можливість працювати у будь-яких браузерах.

4.1 Огляд сторінок авторизації та реєстрації

Для того, щоб використовувати можливості програмного застосунку, користувач повинен пройти авторизацію. На рисунку 4.1 наведена форма авторизації.

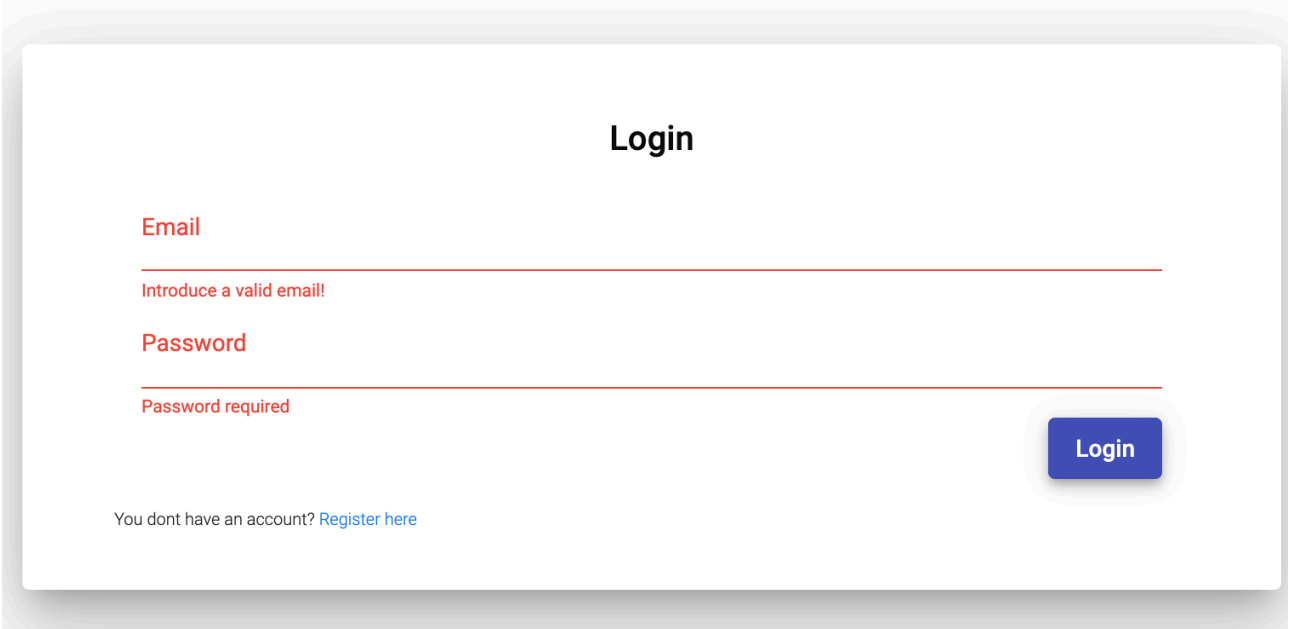


Рисунок 4.1 – Форма авторизації

Також у застосунку існує форма реєстрації, якщо користувач вже був зареєстрований або бажає створити новий профіль. При реєстрації користувача наявна валідація введеного паролю, який повинен мати певну довжину та містити спеціальні символи, та поштової адреси. У разі невалідності якогось поля, користувач отримує попередження про неможливість реєстрації у вигляді повідомлення. Після того, як користувач увів усі дані у полях, він має доступ до натискання кнопки “Sign Up”. На рисунку 4.2 зображена форма реєстрації.

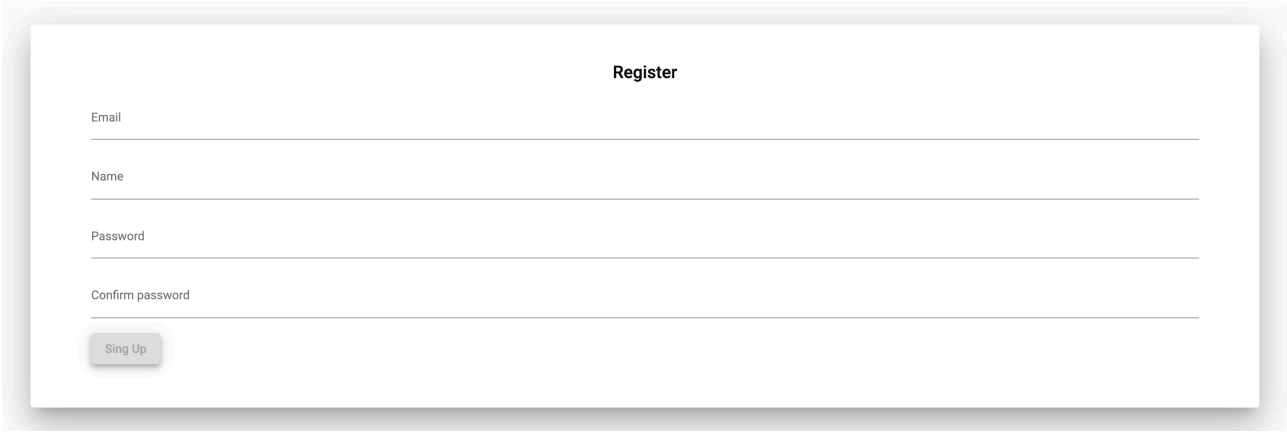


Рисунок 4.2 – Форма реєстрації

Більш того, на пошту користувача додатково надходить повідомлення з підтвердженням реєстрації після того як він відправив свою форму реєстрації.

4.2 Огляд головної сторінки

Сторінка з відображенням усіх навчальних курсів відображається одразу, коли користувач пройшов авторизацію. На даній сторінці є можливість підтримувати комунікацію за допомогою чату з усіма учасниками платформи (рис. 4.3).

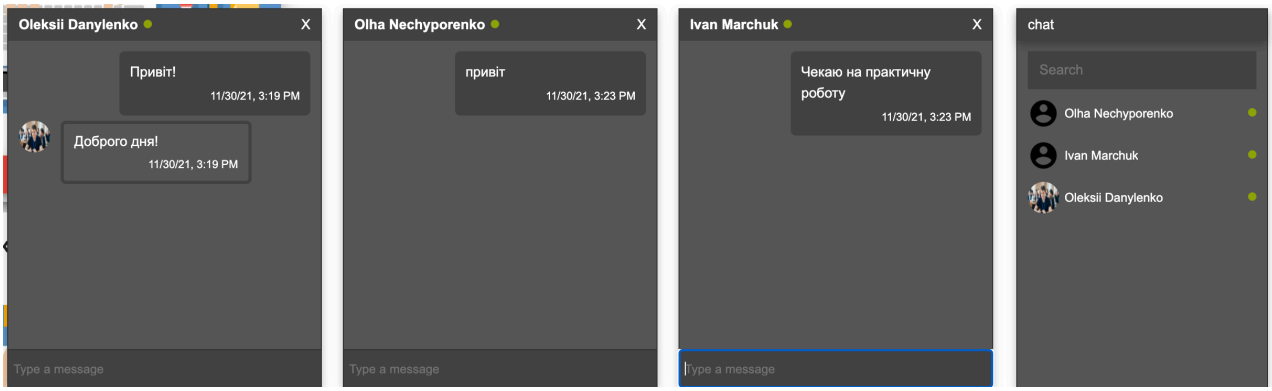


Рисунок 4.3 – Відображення чатів

Перейшовши до пункту меню “Календар”, можна подивитись, що на ньому виділяються спеціальним кольором дні, в які проводяться веб-конференції. Таким чином, користувач може переглянути дні, в яких будуть вони проводитися та одразу перейти на них, натиснувши відповідну кнопку. На рис. 4.4. зображено сторінку календаря з відображенням усіх веб-конференцій у певний день.

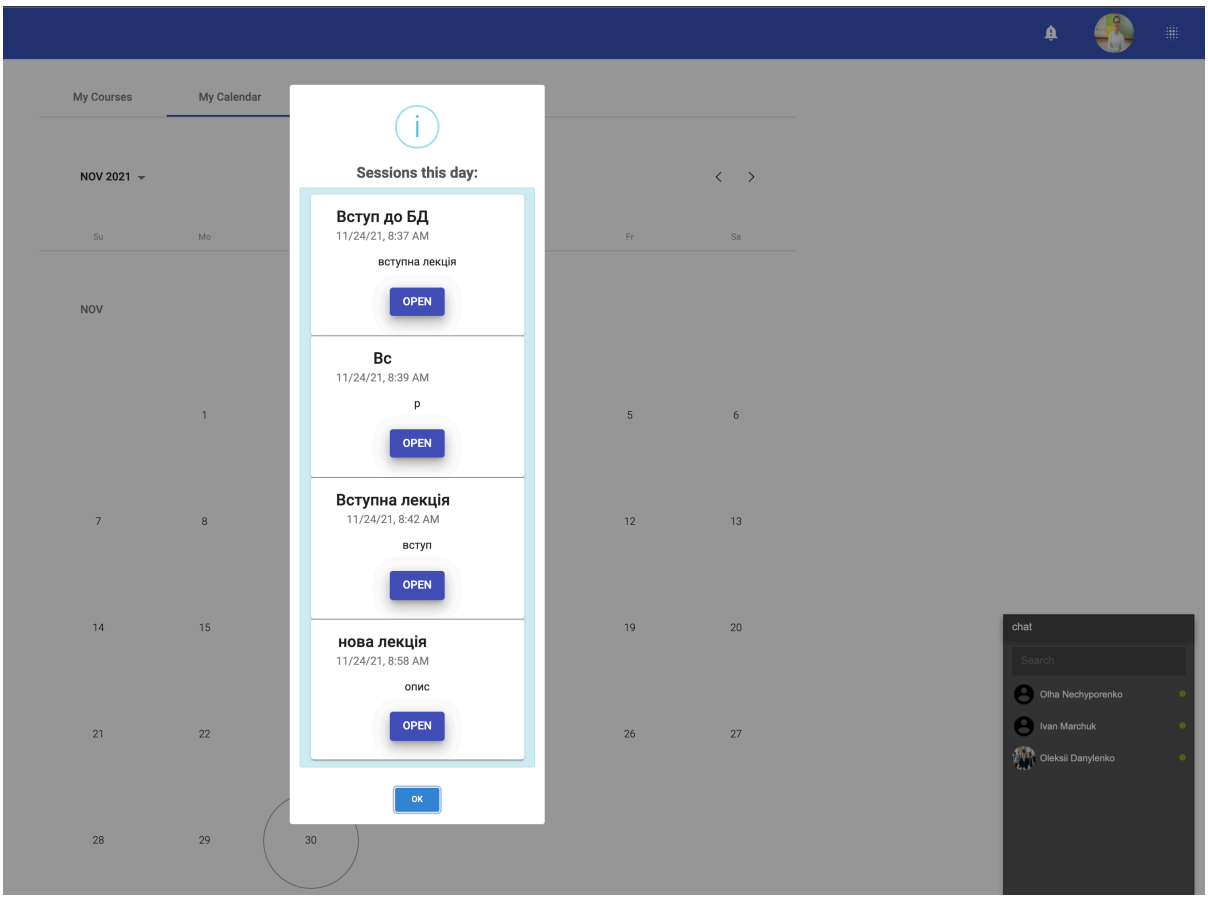


Рисунок 4.4 – Пункт меню “Календар”

З головної сторінки користувач має змогу переглядати свій профіль та оновлювати його. Приклад зображення профілю користувача наведений на рис. 4.5.

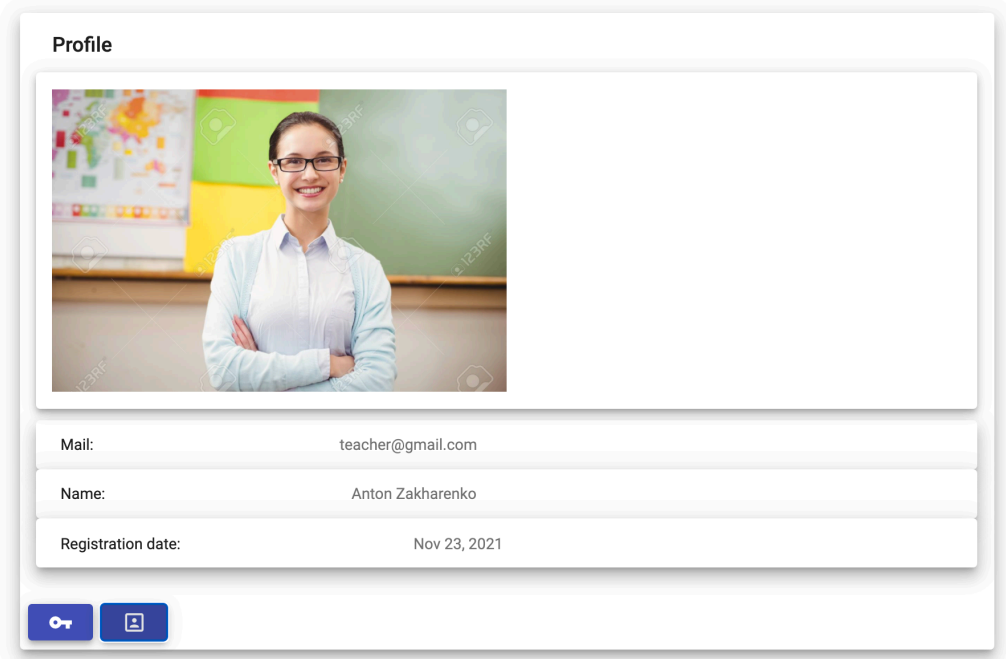


Рисунок 4.5 – Сторінка профілю користувача

Головна сторінка містить також перелік навчальних курсів, на які користувач був доданий. Якщо зайти на будь-який курс, відкриється головна сторінка навчального курсу, що є основною навігаційною ланкою між іншими сторінками програмного застосунку, такими як дошки оголошень, сторінки форуму, відео-конференцій, завдань та переліку учасників курсу. На рисунку 4.6 зображено перелік навчальних курсів головної сторінки.

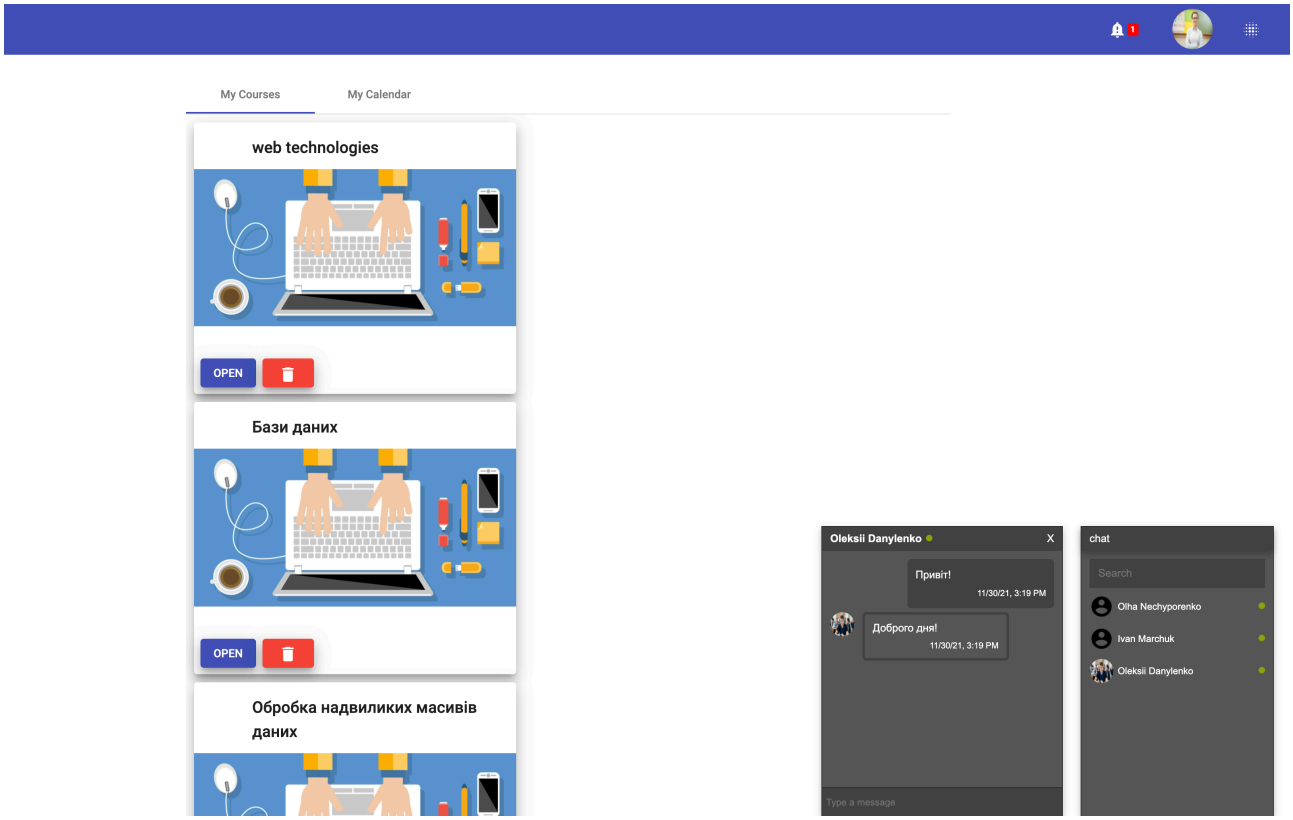


Рисунок 4.6 – Відображення навчальних курсів

4.3 Огляд сторінок навчального курсу

Сторінка навчального курсу містить декілька пунктів меню, які використовуються для підтримки комунікації між учасниками цього курсу.

Завдяки пункту меню “Форум”, учасники навчального курсу мають можливість створювати форуми на відповідні теми для обговорення якихось питань або подій навчального процесу. Окрім текстових повідомлень у форумі, існує можливість голосового запису. На рисунку 4.7 зображено приклад створених форумів.

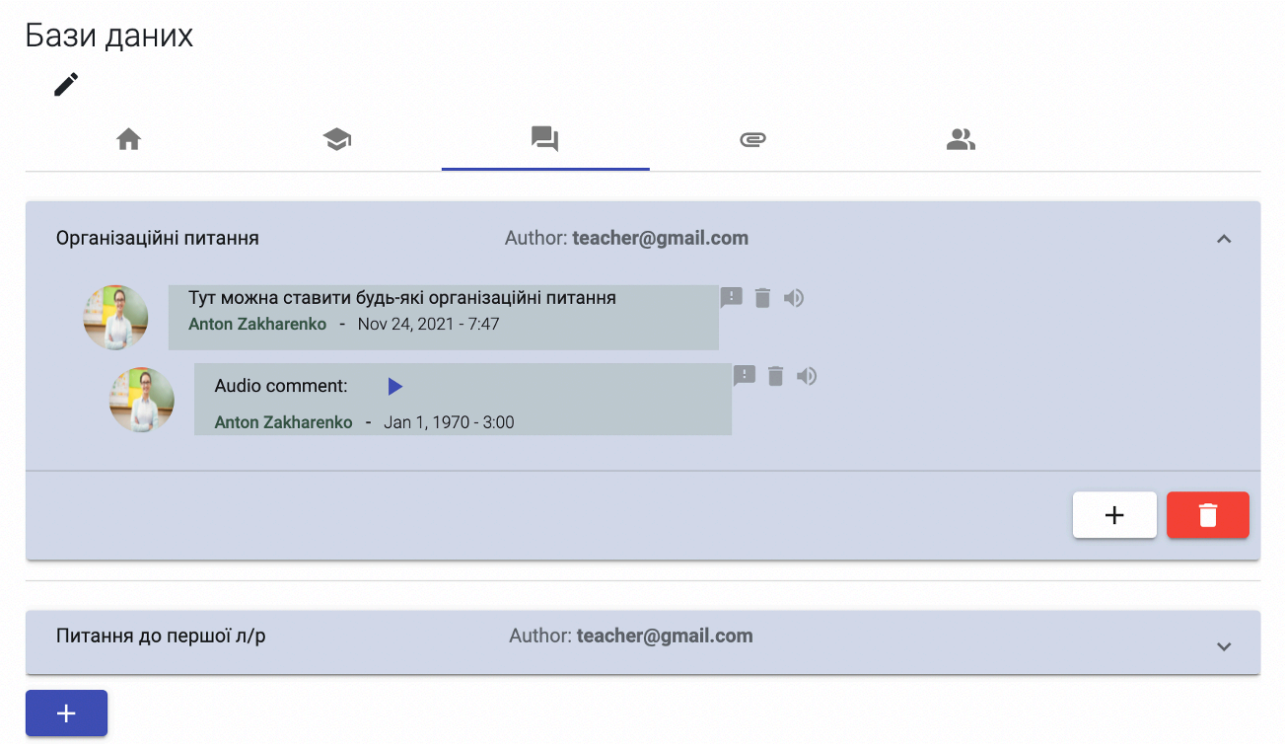


Рисунок 4.7 – Відображення форумів навчального курсу

У навчальному курсі також наявний пункт меню “Сесії”, в якому містяться створені викладачем веб-конференції, на яких можна проводити навчальні лекції або практичні заняття. У веб-конференції є підтримка чату між учасниками, що зображено на рис. 4.8.

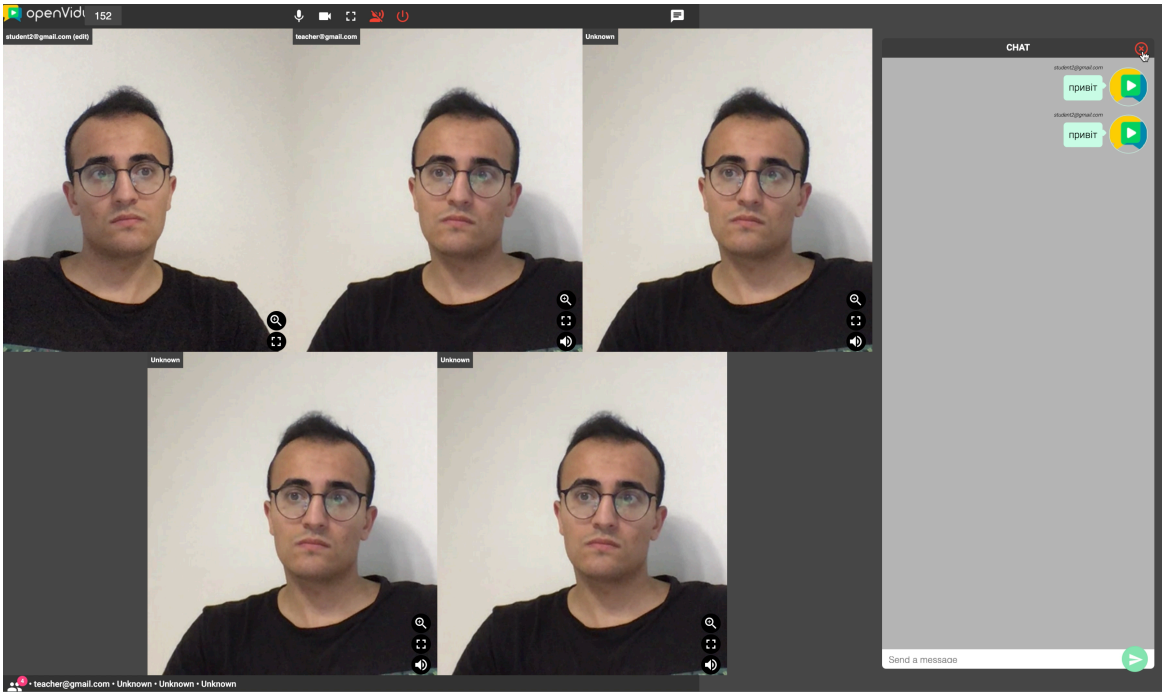


Рисунок 4.8 – Приклад веб-конференції між учасниками курсу

Ще одним пунктом меню навчального курсу є дошка оголошень, яка може оновлюватися тільки викладачем, який створив цей курс. Використовуючи дошку оголошень, викладач має змогу створювати текстові оголошення, додавати відео файли та посилання на інші ресурси.

Слід також відмітити, що кожен курс прив’язаний до каналу в застосунку Telegram. Саме при створенні навчального курсу викладачем, створюється автоматично канал у Telegram, якщо викладач підв’язав свій профіль телеграму з веб-застосунком. При додаванні нових публікацій до дошки оголошень, інформація дублюється автоматично у прив’язаному каналі телеграму, що додає можливість більш ефективного сповіщення учасників про навчальні події. При створенні веб-конференції до телеграму також автоматично надсилаються повідомлення з посиланням до неї, що значно прискорює процес сповіщення та усуває витрату часу на пошук посилання. Приклад відображення каналів телеграму та відповідних до них навчальних курсів у браузері зображені на рис. 4.9.

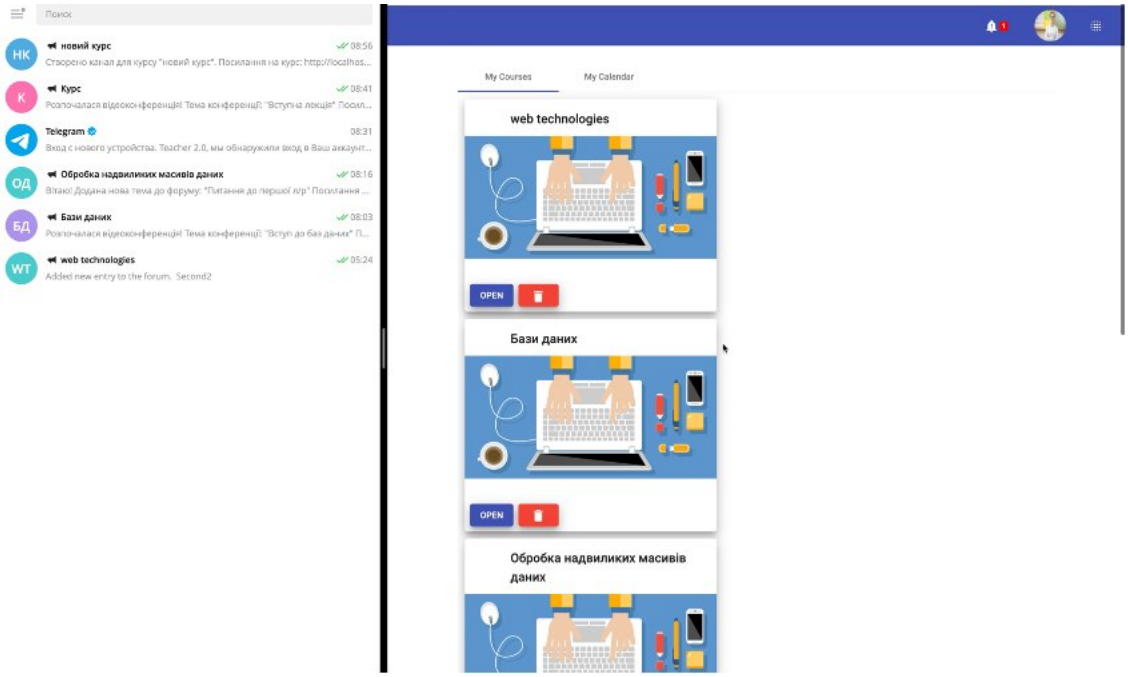


Рисунок 4.9 – Відображення навчальних курсів та відповідних телеграм каналів

Також навчальний курс містить пункт меню про учасників, що дає змогу переглянути всіх учасників навчального курсу. Викладач в свою чергу, може

додавати нових учасників курсу шляхом вказання необхідної інформації про студента.

Висновки до розділу

У даному розділі описано взаємодію користувача і запропонованого рішення з виділенням основної функціональності. Наведено основні сценарії роботи для користувачів застосунку та описано їх можливості у ньому.

У першому підрозділі описано детально сторінки для авторизації та реєстрації. Також було взято до уваги, що для кожного поля форми існує валідація на введення коректних даних.

Другий підрозділ містить взаємодію користувача із головною сторінкою веб-застосунку, в якому зосереджена основна навігація по іншим його місцям, а саме сторінка профілю користувача, календаря, відображення чатів з іншими користувачами та безпосередньо навчальних курсів.

В останній частині розділу проведено огляд сторінок навчального курсу, де відбувається основна комунікація між учасниками навчального процесу за рахунок використання форумів, дошки оголошень та проведення веб-конференцій.

5 ОЦІНКА ЕФЕКТИВНОСТІ ТА ТЕСТУВАННЯ ЗАПРОПОНОВАНОГО РІШЕННЯ

5.1 Визначення метрик ефективності

У результаті розробки програмного забезпечення важливим етапом також є перевірка його якості за рахунок проведення оцінки ефективності. Якість програмного забезпечення визначається ступенем відповідності явним або неявним вимогам та очікуванням. Ці так звані явні та неявні очікування відповідають двом базовим рівням якості програмного забезпечення.

Функціональний – відповідність продукту функціональним (явним) вимогам та технічним умовам. Цей аспект орієнтований на практичне використання програмного забезпечення з точки зору користувача, таким як його особливості, продуктивність, зручність у використанні, відсутність дефектів.

Нефункціональні – внутрішні характеристики та архітектура системи, тобто структурні (неявні) вимоги. Сюди входить ремонтопридатність, зрозумілість, ефективність, продуктивність та безпека.

Оцінка якості ПЗ – сукупність операцій, які включають вибір номенклатури показників якості оцінюваного ПЗ, визначення значень цих показників і порівняння їх з базовими значеннями [36].

Таблиця 5.1 – Метрики ефективності

Параметр	**Одиниці виміру**	**Опис**
Час відповіді	Час (мілісекунди)	Час очікування, поки клієнт отримає відповідь сервера після надсилання запиту.
Пропускна здатність	запит/секунда	Кількість запитів, які обробляються додатком за одиницю часу
Використання ресурсів	відсоток	Використання ресурсів, таких як ЦП (центральний процесор), пам’ять, пропускна здатність мережі

У дослідженні будуть проводитися оцінка вищезазначених метрик запропонованого рішення.

5.2 Написання unit тестів

Для перевірки програми, а точніше її компонентів, на наявність помилок використовують unit тестування. Його особливість полягає в тому, що вибирається певний компонент або метод класу і для нього ми знаємо, які повинні бути вхідні та вихідні дані. Далі, використовуючи бібліотеку Junit, ми обираємо потрібний нам метод і пишемо, що на вхід такого-то компоненту повинні прийти такі дані, а в результаті його роботи повинні вийти такі результати. Потім ми запускаємо написаний unit тест на виконання і отримуємо результат, в якому відображається чи співпадає очікуваний результат з дійсним. Цей метод тестування є досить зручним, адже дозволяє знайти всі помилки в програмі при роботі, наприклад, в тестовому оточенні та при перенесенні системи на робочий сервер.

Для перевірки правильності виконання алгоритму створення нового користувача у застосунку були написані відповідні unit тести, один з яких зображений на рисунку 5.1.

```
public class UserUnitaryTest extends AbstractUnitTest {

    /*Test user data*/
    String name = "TestUser";
    String password = "blablaba";
    String nickName = "testi";
    String picture = "picture/test.jpg";
    String[] roles = {"STUDENT"};


    /**
     * Test method for {@link User#User(java.lang.String, java.lang.String, java.lang.String, java.lang.String, java.lang.String[])}
     * and {@link User#User()}.
     */
    @Test
    public void newUserTest() {

        //Empty user
        User emptyUser = new User();
        Assert.notNull(emptyUser, message: "User failed to be created");

        //User with picture
        User u = new User(name, password, nickName, picture,roles);
        Assert.notNull(u, message: "User failed to be created");
        Assert.isTrue(name.equals(u.getName()), message: "User failed to be created");
        Assert.isTrue((new BCryptPasswordEncoder()).matches(password, u.getPasswordHash()), message: "User failed to be created");
        Assert.isTrue(nickName.equals(u.getNickName()), message: "User failed to be created");
        Assert.isTrue(picture.equals(u.getPicture()), message: "User failed to be created");
        Assert.isTrue( expression: roles.length == u.getRoles().size(), message: "User failed to be created");

        //user witout picture
        u = new User(name, password, nickName, picture: null,roles);
        Assert.notNull(u, message: "User failed to be created");
        Assert.isTrue(name.equals(u.getName()), message: "User failed to be created");
        Assert.isTrue((new BCryptPasswordEncoder()).matches(password, u.getPasswordHash()), message: "User failed to be created");
        Assert.isTrue(nickName.equals(u.getNickName()), message: "User failed to be created");
        Assert.isTrue( expression: roles.length == u.getRoles().size(), message: "User failed to be created");
    }
```

Рисунок 5.1 – Приклад unit-тесту для створення нового користувача у застосунку

На даному рисунку ми можемо побачити 1 unit тест. Назва кожного методу unit тесту є унікальною. Прочитавши назву, можна зрозуміти, що перевіряється в даному тесті. Так, в даному тесті ми виконується створення нового користувача із застосуванням змінних класу для заповнення всіх необхідних атрибутів користувача. В даному методі застосовується клас Assert бібліотеки Junit.

Наприклад, метод notNull даного класу приймає на вхід два параметри: фактичне значення поля, яке початково проініціалізовано в даному класі та текст помилки. Текст помилки призначений у разі не завершення роботи метода, а саме якщо перший параметр цього методу буде пустим, то цей текст буде відображатися у помилці. Метод isTrue має перший вхідний параметр логічного типу, де перевіряється на відповідність значення, яке очікується отримати та значення, яке є на даний момент, другий параметр цього методу представляє собою строковий тип, у який передається текст у разі помилки.

Результат виконання тестів можна переглянути на рисунку 5.2.

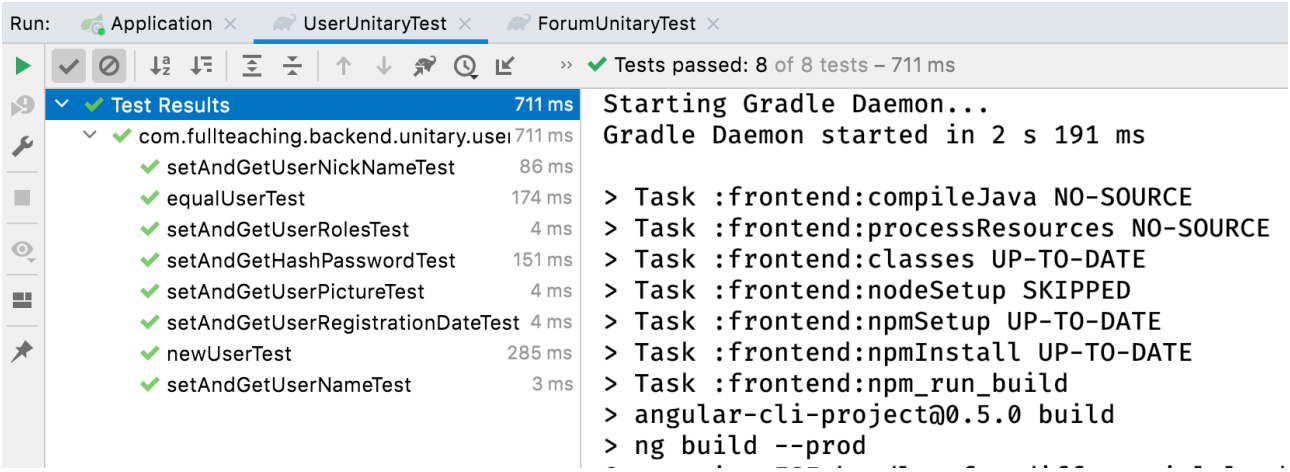


Рисунок 5.2 – Результат виконання unit тестів

Всі тести успішно виконалися. В лівому вікні відображається перелік тестів та статус їх виконання. В правому вікні після виконання тестів відображається загальна інформація, така як: час виконання, об’єм використаної пам’яті, кількість виконаних тестів. В тому разі, коли ми отримуємо у роботі методу помилку і тест вважається невиконаним, в лівому вікні поряд з назвою тесту з’являється знак оклику у красному колі. При натисненні на невиконаний тест в лівому вікні з’явиться інформація про конкретний тест. Як правило, там пишеться причина невиконання тесту, значення, яке очікувалось отримати та

значення, яке отримали в результаті виконання методу певного класу, а також відображається посилання на отримання відмінностей. Таким чином, якщо ми змінимо певне значення в написаних тестах на неправильне, ми отримаємо наступний результат (рис. 5.3).

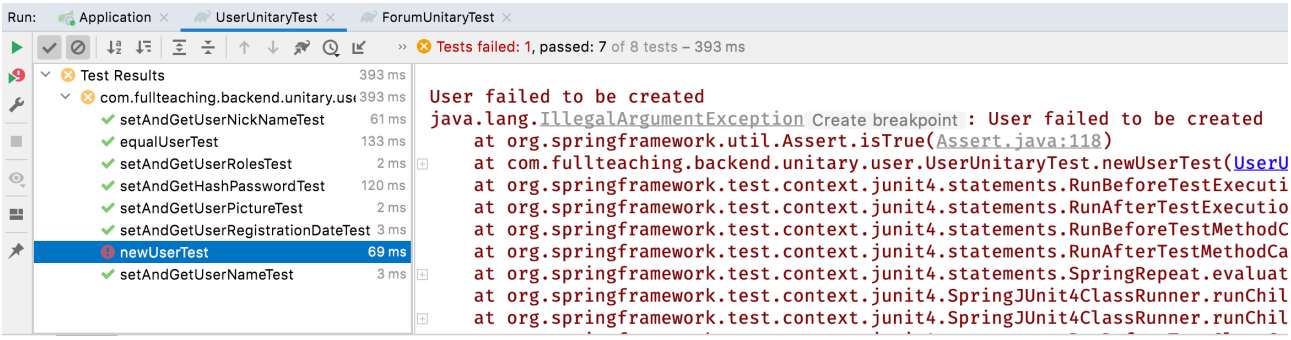


Рисунок 5.3 – Відображення інформації про невиконаний тест

Аналогічні тести були написані та виконані для всіх інших методів веб-орієнтованого застосунку для підтримки комунікації між учасниками навчального процесу. Методи, для яких тести не виконались, були перевірені на наявність помилок та було здійснене їх виправлення та повторне виконання тестів. Така процедура виконувалася для всіх методів, поки не було досягнено потрібних результатів.

5.3 Аналіз швидкодії завантаження сторінок

Проведемо аналіз швидкодії веб-застосунку, проаналізувавши час завантаження різних його сторінок, використовуючи консоль розробника браузеру. Час завантаження головної сторінки становить 1-2 секунди, що я досить хорошим показником (рис. 5.4).

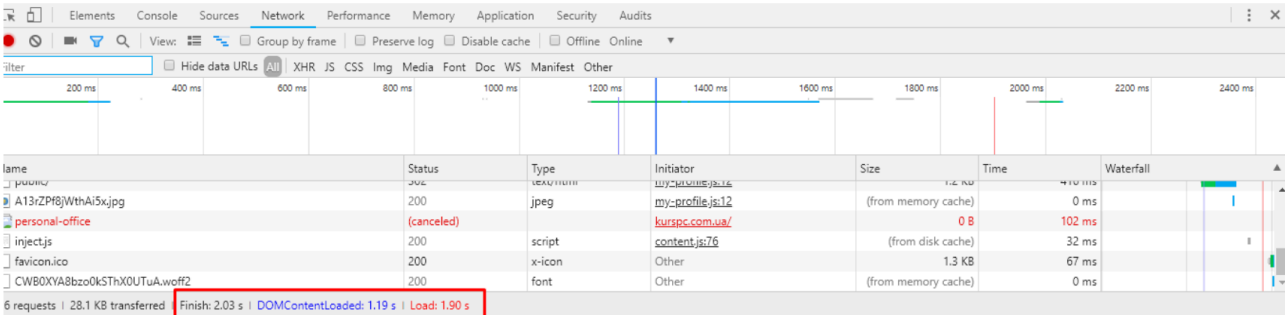


Рисунок 5.4 – Час завантаження головної сторінки

Час завантаження сторінки, яка містить матеріали навчального курсу для студентів, які зареєстровані у ньому (рис. 5.5).

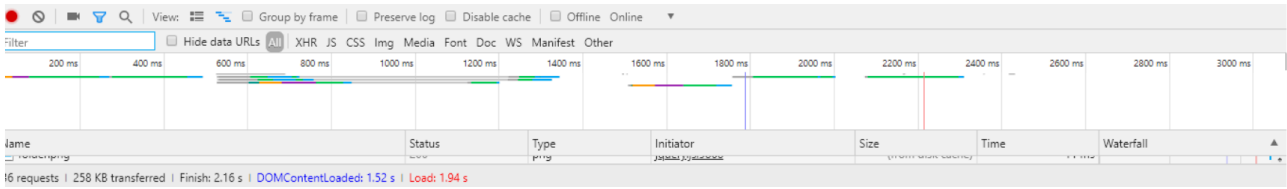


Рисунок 5.5 – Час завантаження сторінки із навчальними матеріалами

Маємо аналогічний час завантаження для сторінок із переліком всіх студентів, яких на момент тестування програми налічувало 245 чоловік (рис. 5.6).

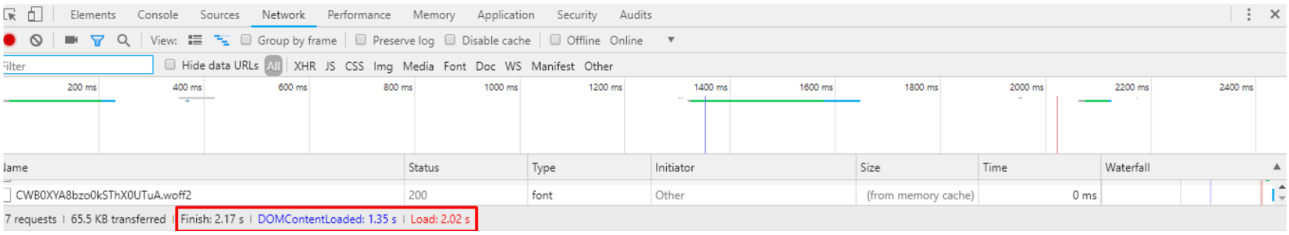


Рисунок 5.6 – Час завантаження сторінки зі списком всіх студентів

Час завантаження сторінки, де відображається форум навчального курсу з переліком всіх коментарів обраної тематики (рис. 5.7).

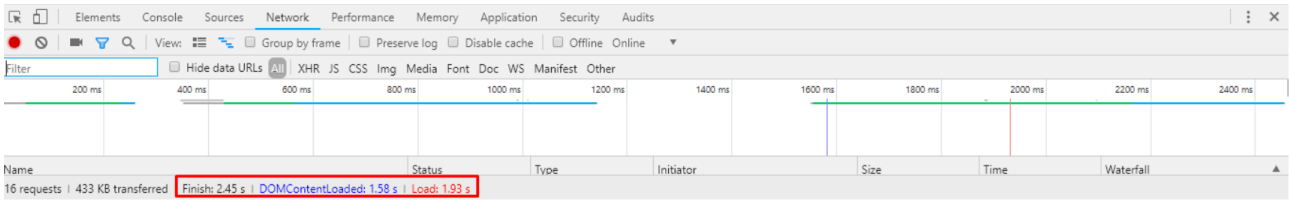


Рисунок 5.7 – Час завантаження сторінки форуму

Отже, проаналізувавши всі отримані результати, можна зробити висновок, що програмний застосунок має належну швидкодію та завдяки цьому є зручним у користуванні.

5.4 Аналіз роботи протоколів WebSocket та HTTP

У даному підрозділі буде проведено навантажене тестування для порівняння роботи 2 протоколів, які застосовуються при комунікації клієнта зі створеним веб-сервером.

Протокол WebSocket використовується при роботі застосунку у реальному часі, а саме при проведенні веб-конференції для обробки аудіо та відео потоків.

Також цей протокол використовується для обміну, публікації та трансляції повідомлення у чаті веб-застосунку. Він повторно використовує те саме з'єднання WebSocket-у для надсилання та отримання повідомлення.

Для того, щоб довести ефективність роботи цього протоколу, виконаємо порівняння його роботи із протоколом HTTP за декілька сценаріями. Кожен тестовий сценарій виконувався десять разів для заданої кількості користувачів і в середньому з них було розраховано десять пробігів. Кількість користувачів застосунку коливається від 1 до 500 із кроком 50.

На рисунку 5.8 зображено графік залежності завантаження процесора від кількості користувачів під час трансляції.

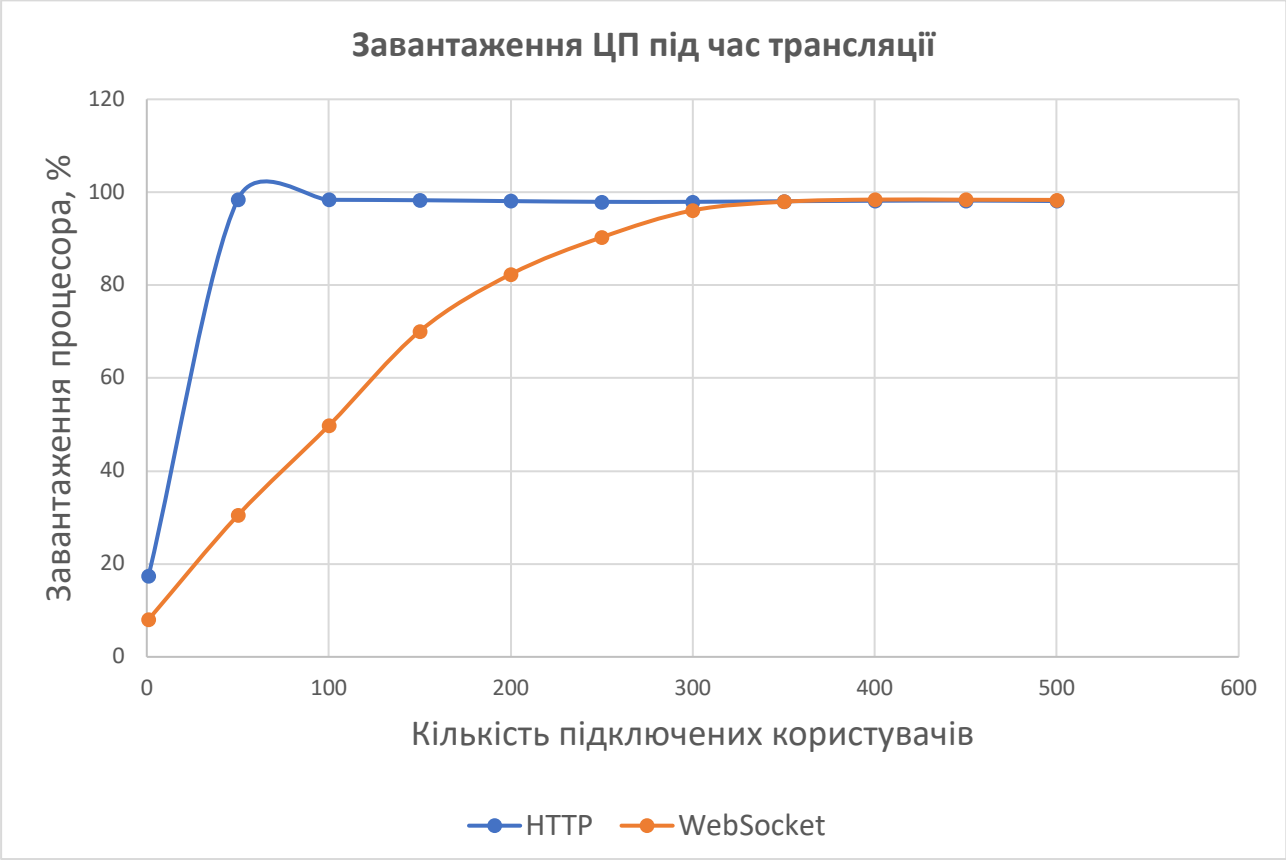


Рисунок 5.8 – Графік завантаження ЦП під час трансляції

На рисунку 5.8 показано, що сервер з використанням HTTP досягає максимального використання ЦП вже при 50 користувачів. З іншого боку, при використанні протоколу WebSocket досягається максимальне навантаження на ЦП при наявності 350 користувачів. Таким чином, пікове навантаження ЦП при використанні HTTP досягається у рази швидше при збільшенні користувачів у

застосунку в порівнянні із WebSocket, тому саме використання WebSocket більше підходить для обміну повідомленнями між сервером у реальному часі, ніж довге опитування HTTP.

На рисунку 5.9 зображено графік залежності часу відповіді сервера від кількості користувачів під час трансляції.

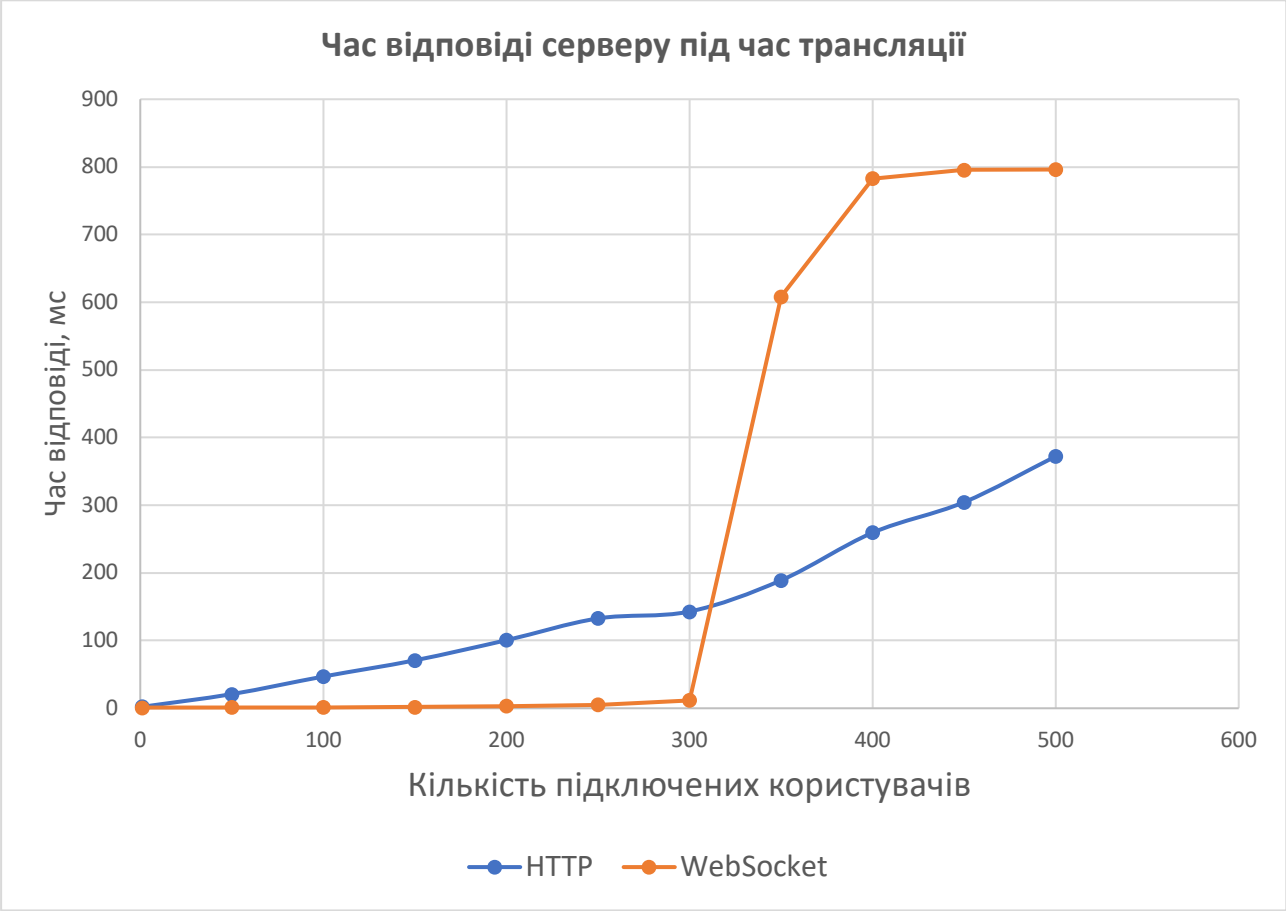


Рисунок 5.9 – Графік часу відповіді серверу під час трансляції

Коли сервер при використанні HTTP досягає повного завантаження ЦП при 50 користувачах, ми бачимо початок стабільного, майже лінійного підйому в часі відгуку сервера, оскільки кількість користувачів зростає. При 200 користувачів, сервер починає використовувати більше 100 мілісекунд для відповіді на запит. При максимальній кількості 500 користувачів сервер відповідає аж 372 мілісекунди.

При використанні WebSocket, завантаження ЦП є помірним при 1 до 300 користувачів, час відгуку залишається дуже низьким. Але вже при 350 користувачів центральний процесор сервера навантажується повністю. Час

відгуку сервера значно перевищує межу – в 0,1 секунди. Зростання часу відгуку сервера припиняється на 800 мілісекундах при 400 користувачів.

Далі виконаємо тестування чату веб-застосунку. На рисунку 5.10 зображено графік залежності завантаження процесора від кількості користувачів під час роботи чату.

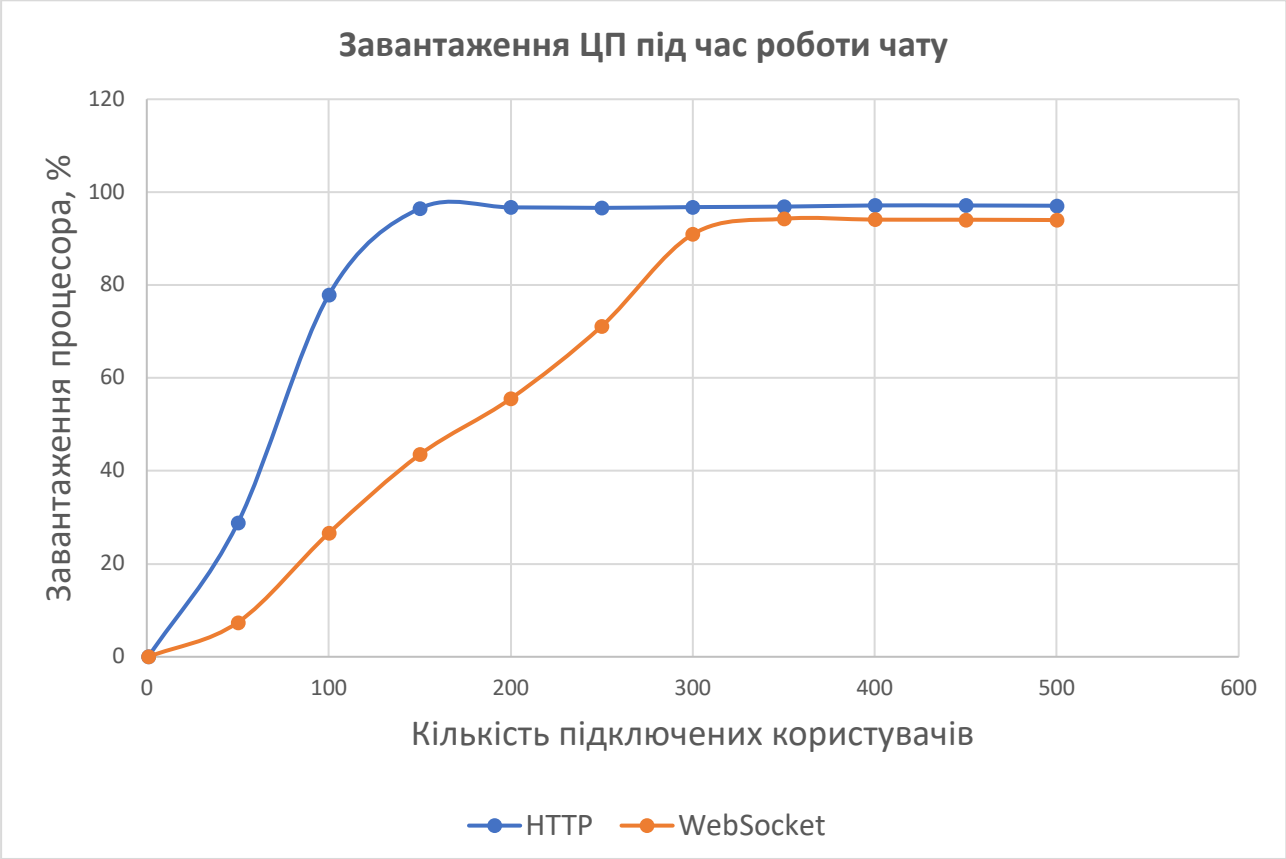


Рисунок 5.10 – Графік завантаження ЦП під час роботи часу

На рисунку 5.10 показано, що сервер при використанні HTTP досягає максимального використання ЦП при 150 користувачах, тоді як при використанні WebSocket досягається максимум 350 користувачів.

На рисунку 5.11 зображено графік залежності часу відповіді сервера чату від кількості користувачів під час роботи чату.

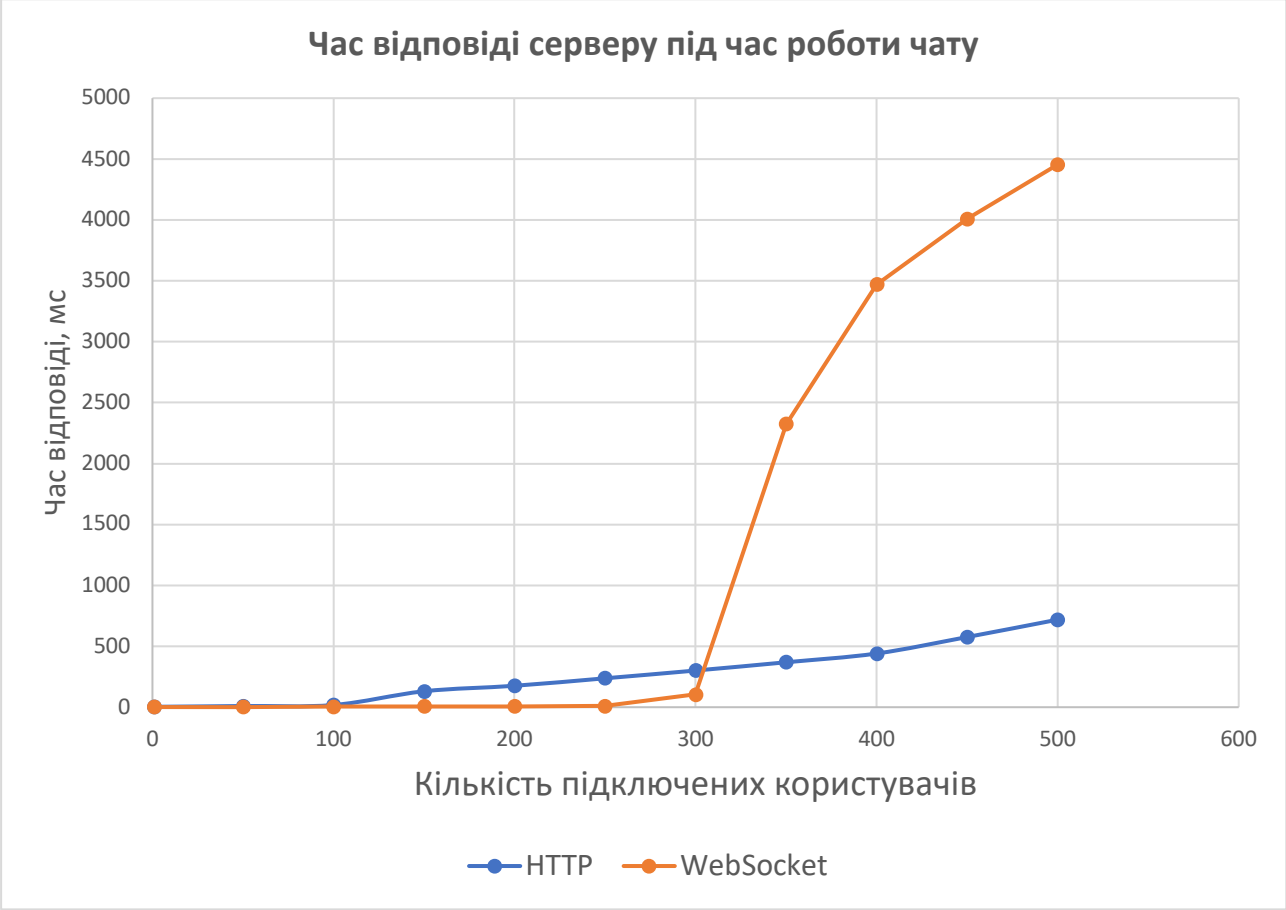


Рисунок 5.11 – Графік часу відповіді серверу під час роботи чату

Коли сервер при використанні HTTP досягає повного завантаження ЦП при 150 користувачах, ми бачимо майже лінійне збільшення часу відповіді в міру збільшення кількості користувачів. Сервер веде себе належним чином, і поки завантаження ЦП помірне, він ніколи не витрачає більше 0,1 секунди на відповідь. Коли тестування досягається пікового рівня, можна побачити, що час відповіді перевищує обмеження в 0,1 секунди, але ніколи не перевищує обмеження в 1 секунду.

Коли навантаження досягає пікового рівня при використанні WebSocket, то відбувається раптове зростання часу відгуку сервера. З максимальним числом 500 користувачів, сервер має час відповіді 4,5 секунди.

Отже, робота з протоколом HTTP є не такою ефективною порівняно з WebSocket при виконанні навантажувальних тестів, як з точки зору використання ЦП, так і часу відгуку сервера. Тому можна сказати, що WebSocket

слід вважати кращим вибором, якщо очікується використання низького та середнього рівня навантаження.

Висновки до розділу

У даному розділі проведено оцінку ефективності та тестування запропонованого рішення.

У першому підрозділі наведено визначення якості програмного забезпечення та функціональних і нефункціональних вимог. Також було визначено метрики ефективності програмного забезпечення, такі як час відгуку, пропускна здатність та використання ресурсів.

Другий підрозділ містить інформацію про застосування бібліотеки Junit для написання та проведення unit тестів. Наведено приклади unit тестів та використаних методів бібліотеки для перевірки певних значень під час тесту.

У третьому підрозділі проведено аналіз швидкодії завантаження сторінок веб-застосунку. Наведено декілька прикладів, при яких можна побачити час завантаження сторінки.

В останній частині розділу проведено дослідження ефективності роботи протоколів HTTP та WebSocket, які використовуються при роботі передачі відео та аудіо потоків у реальному часі та роботі чату. Було наведено графіки залежностей часу відгуку та завантаження ЦП сервера від кількості користувачів, в результаті чого було виявлено, що використання WebSocket є більш оптимальним для низького та середнього навантаження застосунку.

6 МАРКЕТИНГОВИЙ АНАЛІЗ СТАРТАП-ПРОЄКТУ

Розділ має на меті проведення маркетингового аналізу стартап проекту задля визначення принципової можливості його ринкового впровадження та можливих напрямів реалізації цього впровадження.

6.1 Опис ідеї проєкту

Проаналізуємо зміст ідеї, її можливі напрямки застосування, чим запропонована ідея відрізняється від існуючих аналогів, а також основні вигоди, які може отримати користувач продукту. Результати аналізу представлені у таблиці 6.1.

Таблиця 6.1 – Опис ідеї стартап-проєкту

Зміст ідеї	Напрямки застосування	Вигоди для користучача
Розробка ПЗ для ефективної комунікації між учасниками навчального процесу	Канал зв’язку між студентами та викладачами, платформа для обміну інформацією.	Користь для учасників навчального процесу, яка надає необхідну та актуальну інформацію про розклад, графік викладання, успішність студентів, можливість комунікації.
	Джерело актуальної інформації про навчання для аспірантів.	Забезпечення зручності у користуванні та наданні студентам доступу до необхідних навчальних матеріалів.

У таблиці 6.2 представлено сильні та слабкі сторони проекту у порівнянні із конкурентами.

Таблиця 6.2 – Опис ідеї стартап-проєкту

№ п/п	Техніко-економічні характеристики ідеї	Продукція конкурентів			W (слабка сторона)	N (нейтральна сторона)	S (сильна сторона)
		Мій проєкт	Google Classroom	Moodle			
1	Гнучке налаштування	Так	Так	Ні	-	-	+
2	Безкоштовне використання	Так	Так	Так	-	+	-
3	Підтримка широкого спектру функцій	Так	Ні	Так	-	-	+
4	Наявність великої функціональності	Так	Ні	Так	-	-	+
5	Кросплатформеність	Ні	Так	Так	+	-	-
6	Необхідність вивчення документації з користування	Ні	Ні	Так	-	-	+

6.2 Технологічний аудит ідеї проєкту

У даному підрозділі буде проведено технологічний аудит проекту та зроблено висновок щодо технологічної здійсненності проекту. Технології, які необхідно реалізувати при розробці проекту, наведені у таблиці 6.3.

Таблиця 6.3 – Технологічна здійсненність ідеї проєкту

№ п/п	Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
1	Автентифікація та реєстрація користувача за допомогою електронної пошти	JWT	Наявна	Доступна
2	Створення та редагування навчальних курсів викладачем	Angular, Typescript, Java	Наявна	Доступна
3	Створення медіа-сховища для збереження навчальних матеріалів	PostgreSQL	Наявна	Загально-недоступна
4	Інтеграція із платформою Telegram	Telegram API	Наявна	Доступна
5	Обробка аудіо та відео потоків у реальному часі	WebRTC	Наявна	Доступна
Обрана технологія реалізації ідеї проекту: для роботи з Telegram API було обрано мову програмування Java. Для реалізації веб-застосунку було обрано фреймворки Spring та Angular.				

Отже, нижче узагальнено обрані технології реалізації ідей проекту:

– середовище розробки: IntelliJ IDEA, Visual Studio;
– мови програмування (об’єктно-орієнтована): TypeScript, Java;
– frameworks: Spring, Angular;
– бібліотеки: Guava, OpenVidu, Telegram API, Hibernate, JWT;
– база даних: PostgreSQL.

6.3 Аналіз ринкових можливостей запуску стартап-проєкту

У даному підрозділі буде проведено аналіз ринку та його можливостей при вході проекту. Також буде наведено фактори загроз та можливостей після запуску проекту. Дане дослідження дозволить спланувати напрямок розвитку проекту у разі виникнення перешкод.

Характеристика потенційного ринку та потенційних клієнтів стартап-проекту наведені у таблицях 6.4 та 6.5 відповідно.

Таблиця 6.4 – Попередня характеристика потенційного ринку

№	Показники стану ринку	Характеристика
1	Кількість головних гравців, од	4
2	Загальний обсяг продаж, грн./ум.од	Немає точної публічної статистики
3	Динаміка ринку	Зростає
4	Наявність обмежень для входу	Затвердження ліцензійних умов впровадження, створення маркетингової стратегії для проведення ефективної рекламної діяльності стосовно ПЗ
5	Специфічні вимоги до стандартизації та сертифікації	Немає
6	Середня норма рентабельності в галузі або по ринку, %	Невідома

Враховуючи кількість головних гравців по ринку, зростаючу динаміку ринку та середню норму рентабельності можна зробити висновок, що на даний момент, ринок для входження стартап-продукту є привабливим.

Таблиця 6.5 – Характеристика потенційних клієнтів стартап-проєкту

№	Потреба, що формує ринок	Цільова аудиторія	Відмінності у поведінці цільових груп клієнтів	Вимоги споживачів до товару
1	Створення програмного забезпечення для підтримки комунікації між учасниками навчального процесу	Викладачі та співробітники університету, студенти	Викладачі потребують універсального застосунку для підтримки ефективної комунікації, у той час як студенти потребують зручного засобу для відстеження навчальних подій.	- Наявність зручного програмного інтерфейсу - Можливість використання на різних платформах - Відсутність що разового налаштування - Легкий доступ

Після визначення потенційних груп клієнтів було проведено аналіз ринкового середовища: було складено таблиці факторів, що сприяють ринковому впровадженню проекту, та факторів, що йому перешкоджають (таблиці 6.6 та 6.7).

Таблиця 6.6 – Фактори загроз

№	Фактор	Зміст загрози	Можлива реакція компанії
1	Конкуренти	Наявність конкурентів котрі надають схожі рішення	Зменшення ціни на поставлену послугу; Розробка унікальних характеристик товару; Надання ліцензій на обслуговування
2	Кошти на розробку та підтримку продукту	Закінчення грошей та недостатнє фінансування	Залучення додаткових інвесторів, мотивація роботи на перспективу; Ітеративна розробка продукту задля покрокового виведення продукту на ринок та отримання відповіді користувачів
3	Вихід аналогу	Вихід аналогу даного товару може призвести до знецінення та безідейності даного товару	Вихід товару на ринок в коротші строки з не повною, але достатньою, функціональністю для зацікавлення усіх цільових аудиторій; Проведення рекламної компанії

Таблиця 6.7 – Фактори можливостей

№	Фактор	Зміст можливості	Можлива реакція компанії
1	Новий продукт	Вихід на ринок, Зменшення монополії, надання нових рішень у сфері	Розробка нової функціональності; Вихід нової продукції на ринок; Надання різноманітних типів ліцензій в залежності від потреб користувача\замовника.
2	Вихід аналогу	Надати продукт з певними характеристиками та можливостями що відсутні у компаній конкурентів	Аналіз ринку та користувачів задля задоволення їх потреб та надання функціональності у найкоротші строки за ціну, котра є дешевшою ніж у продуктів-замінників.
3	Зворотній зв’язок від користувачів	Можливість отримання необхідної інформації для вдосконалення продукту	Наявність вхідних даних та реакція на них з боку команди розробників задля задоволення потреб та бажань кінцевих користувачів.
4	Грошова винагорода за рекламу	При попиті на ПЗ можлива комерціалізація на основі реклами задля отримання грошової винагороди.	Точкова комерціалізація продукту; Введення реклами; Ведення додаткових коштів у проєкт задля його подальшого розвитку.

Як видно із таблиці 6.7 фактори можливості спираються на факт з появлення конкурентів, рекламної кампанії та зворотного зв’язку. У таблиці 6.8 наведено риси конкуренції на ринку.

Таблиця 6.8 – Ступеневий аналіз конкуренції на ринку

№	Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
1	Тип конкуренції: монополістична	Товар від кожної компанії на ринку, являється недосконалим замінником товару, реалізованого іншими фірмами; На ринку є умови для входу та виходу; Ціна корелює між суперниками;	Розробка продукту з характеристиками, які покривають сфери вживання що не покривають інші товари-замінники; Кореляція цін у відповідності до товарів замінників; Різни типи ліцензій.
2	Рівень конкурентної боротьби: світовий	Всі продукти замінники розроблялись інтернаціональними командами з різних куточків світу, продукти не належать до певної держави.	Вихід на ринок збуту продукту з клієнто-необхідною функціональністю; Налагодження маркетингу на основних Інтернет ресурсах задля охоплення великої кількості потенційних користувачів; Надання бета-версій продукту.

Продовження таблиці 6.8

№	Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
3	Галузева ознака: внутрішньогалузева	Даний тип продукту може використовуватися тільки у сфері розробки IT додатків \ продуктів	Надання зручного, інтуїтивно зрозумілого інтерфейсу; Підтримка всім відомих методів взаємодії з середовищем розробки; Наявність документації та он-лайн підтримки.
4	Конкуренція за видами товарів: товарно-видова	Дана конкуренція – конкуренція між товарами одного виду.	Впровадження функціональності яка відсутня у товарів-замінників; Спрощення інтерфейсів; Надання підтримки.
5	Характер конкурентних переваг: цінова та не цінова	Цінові переваги – точкова комерціалізація; Не цінова – надання функціональності, що відсутня у товарах-замінниках.	Надання платних ліцензій лише на критично важливу функціональність для клієнта з певним строком підтримки, що зазначена у відповідній ліцензії; Впровадження унікальної функціональності.
6	За інтенсивністю: марочна	Наявність унікального знаку що відрізняє даний продукт від продуктів-замінників	Впровадження власної назви та власного знаку.

Після аналізу конкуренції проводиться більш детальний аналіз умов конкуренції в галузі за моделлю М. Портера, що наведено у таблиці 6.9.

Таблиця 6.9 – Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
Висновки	– Google Classroom – Moodle	Можливості виходу на ринок є оскільки аналогічні рішення мають значно гірший функціонал	Основним постачальником можна визначити інтернет-ресурси та література з програмування, які не суттєво впливають на проект.	Викладача співробітники навчального закладу, студенти	На ринку наявні лише готові універсальні рішення, які для наших цілей мають бути додатково дороблені та кастомізовані. Рішення, яке б на 100% задовольняло потреби навчального закладу

Проаналізувавши можливості роботи на ринку з огляду на конкурентну ситуацію можна зробити висновок, що кожний з існуючих продуктів не впливає у великій мірі на поточну ситуацію на ринку в цілому, кожний з існуючих продуктів має свою специфічну сферу використання та свої позитивні та негативні сторони щодо рішення певних типів задач, то робота та вихід на даний ринок є можливою і реалізованою задачею.

Для виходу на ринок продукт повинен мати функціонал що відсутній у продуктів-аналогів, повинен задовольняти потреби користувачів, мати

необхідний та достатній функціонал з конфігурування, підтримку зі сторони розробників та можливість розробки спеціального функціоналу за відповідною ліцензією.

У таблиці 6.10 наводяться фактори конкурентоспроможності та їх обґрунтування.

Таблиця 6.10 – Обґрунтування факторів конкурентоспроможності

№	Фактор конкурентоспроможності	Обґрунтування
1	Технічне обслуговування	Випуск нових версій продукту, оновлення технічної бази
2	Потреби споживачів	Потреби споживачів впливають на вдосконалення застосунку
3	Результативність	Кінцевий результат
4	Ціна та собівартість продукції	Конкурентоспроможна ціна

За визначеними факторів конкурентоспроможності у таблиці 6.10 проведено аналіз сильних та слабких сторін стартап-проекту (таблиця 6.11).

Таблиця 6.11 – Порівняльний аналіз сильних та слабких сторін стартап-проекту

№	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні з запропонованим						
			-3	-2	-1	0	+1	+2	+3
1	Технічне обслуговування	15					+		
2	Потреби споживачів	18							+
3	Результативність	18							+
4	Ціна та собівартість продукції	13				+			

Фінальним етапом ринкового аналізу можливостей впровадження проекту є складання SWOT-аналізу (матриці аналізу сильних (Strength) та слабких (Weak) сторін, загроз (Troubles) та можливостей (Opportunities) на основі виділених ринкових загроз та можливостей, та сильних і слабких сторін (таблиця 6.12).

Таблиця 6.12 – SWOT аналіз стартап-проєкту

Сильні сторони (S):	Слабкі сторони (W):
– Простота та зручність у використанні, привабливий інтерфейс користувача	– Відсутність досвіду використання програмного продукту, можлива присутність багів
Можливості (O):	Загрози (T):
– Вихід на національний ринок; розширення функціональних можливостей та доопрацювання поточних.	– Присутність недоробок на ранньому етапі може спричинити відтік клієнтів.

На основі SWOT-аналізу розробляються альтернативи ринкової поведінки для виведення стартап-проекту на ринок та орієнтовний оптимальний час їх ринкової реалізації з огляду на потенційні проекти конкурентів, що можуть бути виведені на ринок. Визначені альтернативи аналізуються з точки зору строків та ймовірності отримання ресурсів (таблиця 6.13).

Таблиця 6.13 – Альтернативи ринкового впровадження стартап-проєкту

№ п/п	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Імплементація ПЗ у рамках іншої платформи	Середня	1-3 місяці

Продовження таблиці 6.13

№ п/п	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
2	Рекламна кампанія та розповсюдження інформації	Достатня	1-4 місяця
3	Розширення функціоналу	Достатня	3-6 місяця

Як видно із таблиці 6.13, найбільш сприятливою альтернативою є імплементація програмного забезпечення у рамках іншої платформи. Проте існують ризики пов’язані із специфікою реалізації системи у невідомій платформі, тому строки реалізації можуть збільшитися. Тому можна вважати, що найкращими альтернативами є розширення функціоналу та розповсюдження інформації про стартап-проект, а саме про його переваги, унікальність та успішний досвід інтеграції з існуючими каналами.

6.4 Розроблення ринкової стратегії проекту

У таблиці 6.14 наведені основні цільові групи потенційних споживачів.

Таблиця 6.14 – Вибір цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту) за рік	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1.	Вищі навчальні заклади	Готові	середній	Невелика конкуренція	Легка
2.	Університет КПІ	Готові	високий	Мала конкуренція	Легка

Які цільові групи обрано: обрано цільову групу - університет КПІ, так як саме для даної групи є можливість задоволення 100% інтересів клієнтів завдяки створенню програмного забезпечення для ефективної комунікації. У цієї групи також є високий попит на дане програмне забезпечення, та відносно мала конкуренція, звідки простота входу у сегмент – легка.

За результатами аналізу потенційних груп споживачів необхідно обрати цільові групи, для яких буде запропоновано продукт, також необхідно визначити стратегію охоплення ринку, базову стратегію розвитку (таблиця 6.15).

Таблиця 6.15 – Визначення базової стратегії розвитку

Обрана альтернатива розвитку проєкту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
Надання функціональності що відсутня у товарів-замінників, підтримка клієнтів	Проведення реклами, освітлення унікальної функціональності через інтернет ресурси, контакт напряму з споживачами;	Зниження ступеню замінності товару; Прихильність клієнтів; Відмітні властивості товару; Відмітні характеристики товару;	Стратегія диференціації

На наступному етапі необхідно обрати стратегії конкурентної поведінки. На основі базової стратегії розвитку розробимо стратегію конкурентної поведінки (таблиця 6.16).

Таблиця 6.16 – Визначення базової стратегії конкурентної поведінки

Чи є проєкт «першопрохідцем» на ринку	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, які?	Стратегія конкурентної поведінки
Ні, оскільки є товари-заміннники, але дані товари замінники не мають деякого необхідного функціоналу	Так, ціль компанії знайти нових споживачів та, частково, забрати існуючих у конкурентів задля задоволення потреб останніх	Компанія частково копіює характеристики товару конкурента, основна ціль компанії розробка нового унікального функціоналу, з підтримкою основного функціоналу конкурентів	Стратегія заняття конкурентної ніші

За стратегію конкурентної поведінки було обрано стратегію заняття конкурентної ніші. За допомогою проведених досліджень визначена стратегія позиціонування з урахуванням основних вимог до проекту та стратегій його розвитку, які наведені у таблиці 6.17.

Таблиця 6.17 – Визначення стратегії позиціонування

№	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проєкту	Вибір асоціацій, які мають сформувати комплексну позицію власного проєкту
1	Зручність у використанні	Стратегія диференціації	Великі замовлення від масового виробника	– Швидкий вхід в систему – Доступність для будь-яких користувачів
2	Якість програмного забезпечення	Стратегія диференціації	Фізичні властивості продукту	– Зрозуміле використання – Адаптованість

Відповідно до проведеного аналізу можна зробити висновок, що стартап-компанія вибирає як базову стратегію розвитку – стратегію диференціації, як базову стратегію конкурентної поведінки – стратегію заняття конкурентної ніші.

6.5 Розроблення маркетингової програми стартап-проєкту

У даному підрозділі буде проведено розроблення маркетингової програми проекту. Спочатку необхідно сформувати маркетингову концепцію товару, що наведена у таблиці 6.18.

Таблиця 6.18 – Визначення ключових переваг концепції потенційного товару

№	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)
1	Зручне використання	Довготривалість	Розроблено за допомогою Java
2	Адаптованість	Можливість багатофункціонування	Багатофункціонування продукції

Надалі розробляється тривіальна маркетингова модель товару: уточняється ідея продукту та/або послуги, його фізичні складові, особливості процесу його надання (таблиця 6.19).

Таблиця 6.19 – Опис трьох рівнів моделі товару

Рівні товару	Сутність та складові		
1. Товар за задумом	Програмне забезпечення для підтримки ефективної комунікації		
2. Товар у реальному виконанні	Властивості/характеристики	М/Нм	Вр/Тх/Тл/Е/Ор
	– Надання зрозумілого інтерфейсу для керування; – Можливість створення та перегляду навчальних матеріалів; – Наявність проведення веб-конференцій; – Можливість створення навчальних курсів.	М М Нм М	Тх/Ор Тх/Ор Тх/Вр Тх/Вр

Продовження таблиці 6.19

	Пакування: Електронна документація, що містить таку інформацію: – загальна назва продукту, власна назва; – найменування та адреса виробника і місце виготовлення; – товарний знак; – інструкція щодо користування; – необхідні технічні вимоги.
	Марка: ClassGram.
3. Товар із підкріпленням	До продажу: презентація програмного продукту
	Після продажу: підтримка клієнтів, розширення існуючого функціоналу
За рахунок чого потенційний товар буде захищено від копіювання: користувачі можуть впливати на розвиток та набір функціоналу проекту.	

Наступним кроком є визначення оптимальної системи збуту, в межах якої приймається рішення (таблиця 6.20):

– проводити збут власними силами або залучати сторонніх посередників (власна або залучена система збуту);

– вибір та обґрунтування оптимальної глибини каналу збуту; вибір та обґрунтування виду посередників.

Таблиця 6.20 – Формування системи збуту

Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
Клієнти купують програмне забезпечення безпосередньо у компанії-розробника	Налаштування програмного забезпечення на матеріально-технічній базі замовника	Канал нульового рівня (виробник безпосередньо продає товар клієнту)	Через сайт виробника

Сформована система збуту показує, що групи цільових клієнтів в основному виконують закупівлю товарів через Інтернет, тому немає необхідності у налагодженні додаткових однорівневих або дворівневих систем збуту. Організація збуту власними силами має зменшити додаткові витрати пов’язані із збутом за допомогою третіх сторін.

Останньою складовою маркетингової програми є розроблення концепції маркетингових комунікацій, що спирається на попередньо обрану основу для позиціонування, визначену специфіку поведінки клієнтів (таблиця 6.21).

Таблиця 6.21 – Концепція маркетингових комунікацій

№	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуютьс я цільові клієнти	Ключові позиції, обрані для позиціонуван ня	Завдання рекламного повідомлення	Концепція рекламного звернення
1	Клієнти дізнаються про нові модулі програмного забезпечення за допомогою поштових розсилок	Інтернет, соціальні мережі, форуми	Низька ціна, зручність у використанн і	Поширення знань про нові програмні модулі/оно влення.	Перелік основних правдивих даних про оновлення програмног о забезпечен ня, створення нових модулів системи

Як результат, було визначено ринкову (маркетингову) програму, що включає в себе концепції товару, збуту, просування, що спирається на цінності та потреби потенційних клієнтів, конкурентні переваги ідеї, стан та динаміку ринкового середовища, в межах якого буде впроваджено проект, та відповідну обрану альтернативу ринкової поведінки.

Висновки до розділу

У даному розділі було проаналізовано ринок проектів, які надають засоби ефективної комунікації та процесу поширення інформації серед учасників навчального процесу. У результаті дослідження не було виявлено прямих конкурентів, проте були наведені потенційні конкуренти. Дані аналоги мають схожі концепції, проте було визначено переваги розроблюваного програмного забезпечення перед вже існуючими.

Також було проведено технологічну експертизу втілення стартап-проекту. Вона показала, що реалізація програмного забезпечення є актуальною та рентабельною. Проте для того, щоб залишатися на лідируючих позиціях на ринку, необхідно постійно удосконалювати програмний продукт.

ВИСНОВКИ

У процесі роботи над магістерською дисертацією було проведено аналіз предметної області та дослідження сучасних програмних засобів, що використовуються як засоби для комунікації між учасниками навчального процесу. Наведено існуючі аналоги, проаналізовано принцип їх роботи та головні недоліки під час їх використання. Також було виділено основні критерії, що використовувались для оцінки існуючих застосунків, на основі яких було виявлено актуальність реалізації програмного забезпечення.

На основі проведеного дослідження було визначено вимоги та задачі до створення програмного забезпечення. Обґрунтовано тип програмного забезпечення і вибір технологій для його реалізації, а також підхід до вибору архітектурного рішення та його побудову з подальшою реалізацією програмного забезпечення для підтримки ефективної комунікації.

Наведено та описано роботу програмного забезпечення. Для використання запропонованого рішення було наведено відповідні приклади з відображанням усіх його функціональних можливостей.

Проведено оцінку ефективності та тестування програмного забезпечення шляхом аналізу швидкодії завантаження сторінок, написання та виконання unit тестів, визначення метрик ефективності та проведення навантажувального тестування серверу при роботі з різними протоколами.

У роботі було також розглянуто комерційну частину з проведенням маркетингового аналізу стартап-проекту, де було визначено ідею проекту та наведено список потенційних клієнтів. Було визначено конкурентоспроможність стартап-проекту та його поведінку на ринку.

Результати дослідження та роботи над магістерською дисертацією були опубліковані у вигляді тезів на науковій конференції.

Наукова новизна роботи полягає в удосконаленні рівня бізнес логіки багаторівневої архітектури програмного забезпечення, що полягає в прискоренні процесу поширення інформації для підтримки ефективної комунікації шляхом розширення функціональних можливостей цього рівня.

Мету дослідження було досягнуто завдяки таким критеріям ефективності як:

– оперативність або швидкість охоплення, тобто збільшення швидкості повідомлення учасників навчального процесу про будь-які події;

– підвищення активності здобувачів освіти у взаємодії з викладачем на платформі (наприклад, кількість уточнюючих питань) за рахунок інтеграції з розповсюдженим застосунком.

У результаті створення програмного забезпечення усі учасники мають змогу підтримувати комунікацію та обмінюватись інформацією про події та заходи, що спрощує навчальний процес.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) Методика використання інформаційно-комунікаційних технологій у навчальному процесі. Ч.4. Проектування методів управління навчальною діяльністю:начальний посібник / Стариченко Б.Е., Коротаєва Е.В., Сардак Л.В., Єгоров А.Н. , Під ред. Стариченко Б.Е. Єкатеринбург: 2013. 141 с.

2) Биков В.Ю., Лапінський В.В. Методологічні та методичні основи створення і використовування електронних засобів навчального призначення // Комп'ютер у школі та сім'ї. – №3 . – 2012. С. 3-6.. 3.

3) Остапчук Н., Полюхович Н. Використання Google Classroom для організації уроків інформатики: структура віртуального класу //New pedagogical thought. – 2020. – Т. 101. – №. 1. – С. 27-32.

4) Система електронний кампус НТУУ «КПІ ім. Ігоря Сікорського». Єдине інформаційне середовище національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського » // Інструкція користувача – 2015.

5) Moodle [Електронний ресурс] // Вікіпедія. – Режим доступу: https://uk.wikipedia.org/wiki/Moodle (дата звернення: 14.09.2021).

6) Google Meet vs Zoom [Електронний ресурс]. – Режим доступу: https://www.businessinsider.com/google-meet-vs-zoom (дата звернення: 11.09.2021).

7) Telegram [Електронний ресурс] // Вікіпедія. – Режим доступу: https://uk.wikipedia.org/wiki/Telegram (дата звернення: 15.10.2021).

8) Веб-застосунок [Електронний ресурс] // Вікіпедія. – Режим доступу: http://wiki.kubg.edu.ua/Веб-застосунок (дата звернення: 12.10.2021).

9) Desktop Application [Електронний ресурс] – Режим доступу: https://v2cloud.com/glossary/what-is-a-desktop-app (дата звернення: 12.10.2021).

10) RDB [Електронний ресурс] // Вікіпедія. – Режим доступу: https://ru.wikipedia.org/wiki/Rdb (дата звернення: 01.11.2021).

11) Система управління базами даних [Електронний ресурс] // Вікіпедія. – Режим доступу:

https://uk.wikipedia.org/wiki/Система_управління_базами_даних (дата звернення: 01.11.2021).

12) Kristina Chodorow // MySQL. The Definitive Guide Second Edition. Sebastopol: O'Reilly Media, 2013. – 432 p.

13) Baron Schwartz, Peter Zaitsev, Vadim Tkachenko // High Performance Microsoft SQL Server, 3rd Edition. – Sebastopol: O'Reilly Media, 2012. – 864 c.

14) PostgreSQL [Електронний ресурс]. – Режим доступу: http://www.postgresqltutorial.com/what-is-postgresql/ (дата звернення: 01.11.2021).

15) Троелсен Э. Язык программирования C# и платформа .NET 4.5 / Эндрю Троелсен. – Киев: вильямс, 2014. – 1312 с. – (6).

16) Java Reference Manual: [Електронний ресурс]. – Режим доступу: https://docs.oracle.com/en/java/ (дата звернення: 08.11.2021).

17) Мова програмування Javascript [Електронний ресурс] // Вікіпедія. – Режим доступу:

https://uk.wikipedia.org/wiki/JavaScript (дата звернення: 03.11.2021).

18) Мова програмування Typescript [Електронний ресурс] // Вікіпедія. – Режим доступу:

https://uk.wikipedia.org/wiki/TypeScript (дата звернення: 03.11.2021).

19) Документація Spring Framework [Електронний ресурс]. – Режим доступу:

https://docs.spring.io/spring-framework/docs/current/reference/html/ (дата звернення: 04.11.2021).

20) Introduction to Angular concepts : [Електронний ресурс]. – Режим доступу:

https://angular.io/guide/architecture (дата звернення: 03.11.2021).

21) Офіційний сайт Telegram API [Електронний ресурс]. – Режим доступу:

https://core.telegram.org/ (дата звернення: 22.10.2021).

22) TDLib [Електронний ресурс] // Вікіпедія. – Режим доступу: https://core.telegram.org/tdlib (дата звернення: 22.10.2021).

23) Bergkvist, D.C. Burnett, C. Jennings, and A. Narayanan. WebRTC 1.0: Real-time Communication between Browsers

24) Richards, M. Software Architecture Patterns [Text] / M. Richards. – USA: O'Reilly Media, 2015. – ISBN: 9781491924242.

25) Martin Fowler, Microservices [Електронний ресурс]. – Режим доступу: https://martinfowler.com/articles/microservices.html (дата звернення: 07.11.2021).

26) Архитектура REST [Електронний ресурс]. – Режим доступу: https://habr.com/ru/post/38730/ (дата звернення: 09.11.2021).

27) Обобщенный Model-View-Controller [Електронний ресурс]. – Режим доступу:
http://rsdn.org/article/patterns/generic-mvc.xml (дата звернення: 09.11.2021).

28) Офіційний сайт Hibernate Framework [Електронний ресурс]. – Режим доступу:
http://hibernate.org/orm/ (дата звернення: 10.11.2021).

29) Websocket [Електронний ресурс] // Вікіпедія. – Режим доступу: https://uk.wikipedia.org/wiki/WebSocket (дата звернення: 05.11.2021).

30) What Is LMS? [Електронний ресурс]. – Режим доступу: https://www.ispringsolutions.com/blog/what-is-lms (дата звернення: 08.09.2021).

31) PYPL PopularitY of Programming Language [Електронний ресурс]. – Режим доступу:
http://pypl.github.io/PYPL.html (дата звернення: 02.11.2021).

32) TIOBE Index for May 2020 [Електронний ресурс]. – Режим доступу: https://www.tiobe.com/tiobe-index/ (дата звернення: 02.11.2021).

33) The RDBMS Rankings: January 2020 [Електронний ресурс]. – Режим доступу:
https://redmonk.com/sogrady/2020/02/28/languagerankings-1-20/ (дата звернення: 02.11.2021).

34) The Top Programming Languages 2019 [Електронний ресурс]. – Режим доступу:

https://spectrum.ieee.org/computing/software/the-topprogramming-languages-2019 (дата звернення: 02.11.2021).

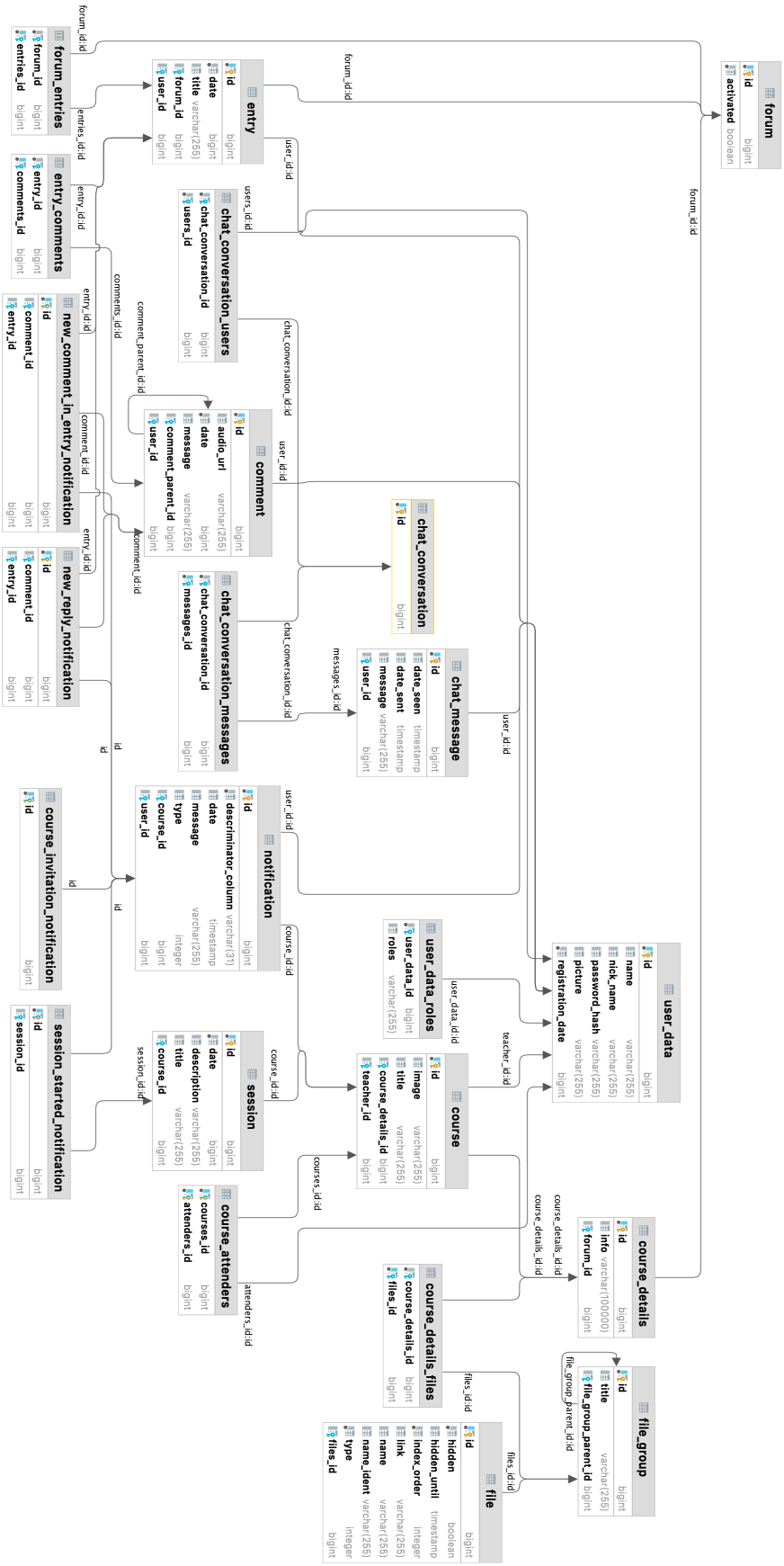
35) Розроблення стартап-проекту. Методичні рекомендації до виконання розділу магістерських дисертацій для студентів інженерних спеціальностей / За заг. ред. О.А. Гавриша. – Київ : НТУУ «КПІ», 2016. – 28 с.

36) Software Quality [Електронний ресурс]. – Режим доступу: http://softwaretestingfundamentals.com/software-quality.

ДОДАТКИ

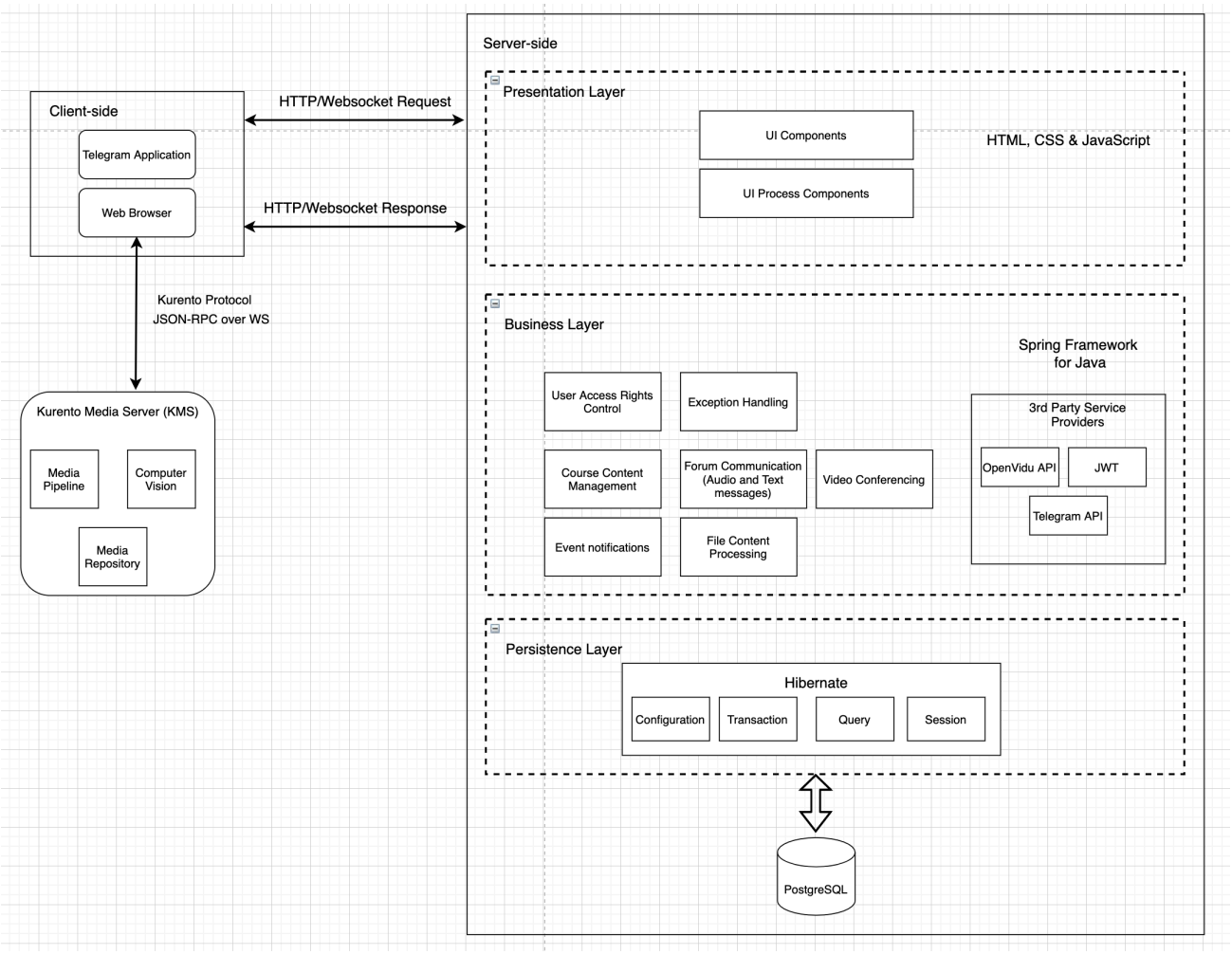
ДОДАТОК А

Схема баз даних



ДОДАТОК Б

Схема архітектури



ДОДАТОК В

Лістинг програмного коду

```
@RestController
@RequestMapping("/api/courses")
@CrossOrigin(origins = "*")
@Slf4j
public class CourseController {

    private final CourseService courseService;
    private final UserService userService;
    private final NotificationService notificationService;
    private final TelegeramChannelService tlChannelService;

    @Autowired
    public CourseController(CourseService courseService, UserService userService,
                                NotificationService notificationService, TelegramChannelService tlChannelService) {
        this.courseService = courseService;
        this.userService = userService;
        this.notificationService = notificationService;
        this.tlChannelService = tlChannelService;
    }

    private class AddParticipantsResponse {
        public Collection<User> participantsAdded;
        public Collection<User> participantsAlreadyAdded;
        public Collection<String> emailsInvalid;
        public Collection<String> emailsValidNotRegistered;
    }

    @RequestMapping(value = "/user/{id}", method = RequestMethod.GET)
    public ResponseEntity<Object> getCourses(@PathVariable(value = "id") long id) {
        log.info("Getting all courses");
        User user = userService.getUserById(id);
        Collection<Course> courses = courseService.getCourserOfUser(user);
        return new ResponseEntity<>(courses, HttpStatus.OK);
    }

    @RequestMapping(value = "/course/{id}", method = RequestMethod.GET)
    public ResponseEntity<Object> getCourse(@PathVariable(value = "id") long id) {
        log.info("Getting exact course");
        Course course = courseService.getCourseById(id);
        return new ResponseEntity<>(course, HttpStatus.OK);
    }

    @RolesAllowed(Role.TEACHER)
    @RequestMapping(value = "/new", method = RequestMethod.POST)
    public ResponseEntity<?> newCourse(@RequestBody Course course, @AuthenticationPrincipal User user) {

        log.info("Adding a new course");

        course.setTeacher(user);
```

```
    course.getParticipants().add(user);

    courseService.save(course);
    course = courseService.getCourseById(course.getId());

    tlChannelService.createChannel(course.getTitle(), course.getCourseDetails().getInfo());

    tlChannelService.sendMessageToChannel("Створено канал для курсу " + "\"" + course.getTitle() + "\"."
            + "\nПосилання на курс: http://localhost:4200/#/courses/" + course.getId());
    log.info("New course succesfully added: {}", course.toString());

    return new ResponseEntity<>(course, HttpStatus.CREATED);
}

@RolesAllowed(Role.TEACHER)
@RequestMapping(value = "/edit", method = RequestMethod.PUT)
public ResponseEntity<Object> modifyCourse(@RequestBody Course updatedCourse) {

    log.info("Updating a course");

    Course course = courseService.getCourseById(updatedCourse.getId());

    log.info("Updating course. Previous value: {}", c.toString());

    course.setImage(updatedCourse.getImage());
    course.setTitle(updatedCourse.getTitle());
    if (updatedCourse.getCourseInfo() != null) {
            course.setCourseInfo(updatedCourse.getCourseInfo());
    }
    courseService.save(course);

    log.info("Course was updated");

    tlChannelService.updateChannelDetails(tlChannelService.channel.getId(),
                                                    tlChannelService.channel.getAccessHash(), course.getCourseInfo());

    return new ResponseEntity<>(course, HttpStatus.OK);
}

@DeleteMapping(value = "/delete")
public ResponseEntity<Object> deleteCourse(@RequestParam(value = "course_id") long courseId) {

    log.info("Deleting a course by id {}", courseId);

    Course course = courseService.getCourseById(courseId);

    Set<Course> courses = new HashSet<>(course);
    List<User> users = userService.findByCourses(courses);
    for (User u : users) {
       u.getCourses().remove(course);
    }
    userService.saveUsers(users);
    courseService.delete(course);

    log.info("Course was deleted");
```

```
    return new ResponseEntity<>(course, HttpStatus.OK);

}


@RequestMapping(value = "/edit/participants", method = RequestMethod.PUT)
public ResponseEntity<Object> addparticipants(
        @RequestBody String[] participantEmails,
        @RequestParam(value = "course_id") long courseId) {

    log.info("Adding participants to the exact course");

    Course course = courseService.getFromId(courseId);

    EmailValidator emailValidator = EmailValidator.getInstance();
    for (int i = 0; i < participantEmails.length; i++) {
        if (emailValidator.isValid(participantEmails[i])) {
            participantEmailsValid.add(participantEmails[i]);
        } else {
            participantEmailsInvalid.add(participantEmails[i]);
        }
    }

    Collection<User> newParticipants = userService.findByNameIn(participantEmailsValid);

     for (String s : participantEmailsValid) {
        if (!this.userListContainsEmail(newParticipants, s)) {
            participantEmailsNotRegistered.add(s);
        }
    }

     for (User participant : newParticipants) {
        boolean newParticipant = true;
        if (!participant.getCourses().contains(course)) participant.getCourses().add(course);
        else newParticipant = false;
        if (!c.getParticipants().contains(participant)) c.getParticipants().add(participant);
        else newParticipant = false;
        if (newParticipant) newAddedParticipant.add(participant);
        else alreadyAddedParticipants.add(participant);
    }

    userService.saveAll(newPossibleParticipants);
    courseService.save(c);

    AddParticipantsResponse response = new AddParticipantsResponse();
    response.participantAdded = newAddedParticipants;
    response.participantsAlreadyAdded = alreadyAddedParticipants;
        response.emailsInvalid = participantEmailsInvalid;
        response.emailsValidNotRegistered = participantEmailsNotRegistered;


    for (User participant : newAddedParticipants) {
        this.notificationDispatcher.notifyInvitedToCourse(participant, c);
    }
```

```
        return new ResponseEntity<>(response, HttpStatus.OK);
        }
    }

    @RequestMapping(value = "/edit/remove-participant", method = RequestMethod.PUT)
    public ResponseEntity<?> removeParticipant(@RequestParam(value = "course_id") long courseId,
                                               @RequestParam(value = "participant_id") long participantId) {
        log.info("Removing exact participant {} from the course {}", participantId, courseId);
        Course course = this.courseService.getCourseById(courseId);

        User participant = this.userService.getUserById(participantId);
        Collection participants = this.courseService.removeParticipant(participant, c);
        log.info("Participant {} was removed from the course {}", participantId, courseId);
        return new ResponseEntity<>(participants, HttpStatus.OK);

    }

}

@Service
public class TgLoginService {

    public TLSentCode sentCode = null;
    public int dcId = 2;
    public TLRequestAuthSendCode asc;
    public TelegramApi api;

    public void sendLoginRequest() {
        MemoryApiState apiState = new MemoryApiState("loggfiles.log");
        int apiId = SystemProperties.get("apiId");
        String apiHash = SystemProperties.get("apiHash");
        this.api = new TelegramApi(apiState, new AppInfo(apiId, "console", "dev", "1", "ua"),
                new ApiCallback() {
                    @Override
                    public void onAuthCancelled(TelegramApi telegramApi) {

                    }

                    @Override
                    public void onUpdatesInvalidated(TelegramApi telegramApi) {

                    }

                    @Override
                    public void onUpdate(TLAbsUpdates tlAbsUpdates) {
                    }
                });
        try {
            final TLConfig config = api.doRpcCallNonAuth(new TLRequestHelpGetConfig());
            apiState.updateSettings(config);

            this.asc = new TLRequestAuthSendCode();
            asc.setPhoneNumber(SystemProperties.get("phoneNumber"));
            asc.setApiId(apiId);
```

```
            asc.setApiHash(apiHash);
            this.sentCode = api.doRpcCallNonAuth(asc);

        } catch (RpcException e) {
            if (e.getErrorCode() == 303) {
                int destDC = Integer.parseInt(e.getErrorTag().substring("PHONE_MIGRATE_".length()));;
                this.dcId = destDC;
                api.switchToDc(destDC);
                try {
                    this.sentCode = (TLSentCode)api.doRpcCallNonAuth(this.asc);
                } catch (RpcException ex) {
                    ex.printStackTrace();
                } catch (TimeoutException ex) {
                    ex.printStackTrace();
                }
            }

        } catch (IOException | TimeoutException e) {

        }
        System.out.println("code was sent successfully " + this.sentCode);


        String hash = this.sentCode.getPhoneCodeHash();
        String smsCode = null;
        try {
            smsCode = reader.readLine();
        } catch (IOException e) {
            e.printStackTrace();
        }

        TLRequestAuthSignIn tlRequestAuthSignIn = new TLRequestAuthSignIn();
        tlRequestAuthSignIn.setPhoneCode(smsCode);
        tlRequestAuthSignIn.setPhoneCodeHash(hash);
        tlRequestAuthSignIn.setPhoneNumber(SystemProperties.get("phoneNumber));


        try {
            TLAuthorization auth = api.doRpcCallNonAuth(tlRequestAuthSignIn);
            api.getState().setAuthenticated(api.getState().getPrimaryDc(), true);


            TLRequestAuthExportAuthorization tlRequestAuthExportAuthorization =
                                                        new TLRequestAuthExportAuthorization();
            tlRequestAuthExportAuthorization.setDcId(api.getState().getPrimaryDc());

            TLUpdatesState state = api.doRpcCall(new TLRequestUpdatesGetState());

            TLAbsUser user = auth.getUser();
        } catch (TimeoutException | IOException e) {

        }
    }
}
```

```
@Service
public class TelegramChannelService {

    @Autowired
    TgLoginService loginService;

    public TLChannel channel;

    public String createChannel(String title, String about) {
        try {
            TLRequestChannelsCreateChannel chh = new TLRequestChannelsCreateChannel();
            chh.setAbout(about);
            chh.setTitle(title);
            this.channel = (TLChannel)((TLUpdates)loginService.api.doRpcCall(chh)).getChats().get(0);
            return "created";
        }
        catch (Exception ex) {
            return "timeOut";
        }
    }

    public void updateChannelDetails(int channelId, long accessHash,  String details) {
        try {
            TLRequestChannelsEditAbout channelsEditAbout = new TLRequestChannelsEditAbout();
            TLInputChannel tlInputChannel = new TLInputChannel();
            tlInputChannel.setChannelId(channelId);
            tlInputChannel.setAccessHash(accessHash);
            channelsEditAbout.setChannel(tlInputChannel);
            channelsEditAbout.setAbout(details);
            loginService.api.doRpcCall(channelsEditAbout);
        } catch (IOException | TimeoutException e) {
            e.printStackTrace();
        }

    }

    public String sendMessageToChannel(String message) {

        try {
            TLRequestMessagesSendMessage rms = new TLRequestMessagesSendMessage();
            TLInputPeerChannel pee = new TLInputPeerChannel();
            pee.setAccessHash(this.channel.getAccessHash());
            pee.setChannelId(this.channel.getId());
            rms.setPeer(pee);
            rms.setMessage(message);
            rms.setRandomId(new Random().nextInt());
            TLVector<TLAbsMessageEntity> ent = new TLVector<TLAbsMessageEntity>();
            TLMessageEntityBold bol = new TLMessageEntityBold();
            bol.setLength(2);
            bol.setOffset(2);
            rms.setEntities(ent);
            rms.enableBroadcast(true);
            loginService.api.doRpcCall(rms);
        }
        catch (Exception e) {
```

```
            e.printStackTrace();
            return e.getMessage();
        }
        return "Sent.";
    }

    public void sendAudioMessageToChannel(File audioFile) {

        try {
            TLSendMessageRecordAudioAction audioAction = new TLSendMessageRecordAudioAction();
            audioAction.serialize(audioFile.getBytes());
            loginService.api.doRpcCall();


            TLRequestUploadSaveFilePart saveFilePart = new TLRequestUploadSaveFilePart();
            saveFilePart.setBytes(new TLBytes(Files.readAllBytes(audioFile.toPath())));
            saveFilePart.setFileId(new Random().nextInt(1000));

            final TLBool res = loginService.api.doRpcCall(saveFilePart);

        }
        catch (Exception e) {
            if (e.getMessage().contains("FLOOD_WAIT") || e.getMessage().toLowerCase().contains("peer_flood")) {
                System.out.println("\tFLOOD_WAIT.");
            }
            e.printStackTrace();
        }
    }
}

@RestController
@RequestMapping("/video/live")
@Slf4j
public class LiveVideoController {

    public static final Path FOLDER_OF_VIDEOS = SystemProperties.get("user.dir");
    private final FileService fileService;
    private final CourseService courseService;

    @Autowired
    public LiveVideoController(FileService fileService, CourseService courseService) {
        this.fileService = fileService;
        this.courseService = courseService;
    }

    @GetMapping(value = "/start/{fileId}/{courseId}")
    public void getVideo(@PathVariable long fileId, @PathVariable long courseId,
                         HttpServletRequest request, HttpServletResponse response) throws Exception {

        Course course = this.courseService.getCourseById(courseId);

        File file = this.fileService.getFileById(fileId);
```

```
        if (file != null) {
            String videoPath = FOLDER_OF_VIDEOS + "/" + file.getFileName();
            log.info(videoPath);

            FileSender fileSender = new FileSender
            fileSender.setFromPath(Paths.get(videoPath);
            fileSender.setRequest(request)
            fileSender.setResponse(response)
            fileSender.serveResource();
        } else {
            log.info("File(id={}) was not found", fileId);
            response.sendError(404, "not found");
        }
    }
}

@RestController
@RequestMapping("/api/video-conference")
@Slf4j
public class VideoConferenceController {

    private final ConferenceService conferenceService;
    private final NotificationService notificationService;
    private final TLChannelService tlChannelService;

    private OpenVidu openVidu;
    private String openviduUrl = SystemProperties.get("openViduUrl");
    private String openviduPassword = SystemProperties.get("openViduPassword");

    @Autowired
    public VideoConferenceController(ConferenceService conferenceService, NotificationService
notificationService, TLChannelService tlChannelService) {
        this.conferenceService = conferenceService;
        this.notificationService = notificationService;
        this.tlChannelService = tlChannelService;
    }

    @PostConstruct
    public void initIt() throws Exception {
        this.openVidu = new OpenVidu(this.openviduUrl, this.openviduPassword);
    }



    @RequestMapping(value = "/get-conferenceid-token/{id}", method = RequestMethod.GET)
    public ResponseEntity<Object> getConferenceIdAndToken(@PathVariable(value = "id") String id,
@AuthenticationPrincipal User user)
                                                throws OpenViduJavaClientException, OpenViduHttpException {

        Conference conference = conferenceService.getConfereceById(id);
        if (conference != null) {
            String conferenceId;
            String token;
            JSONObject responseJson = new JSONObject();
```

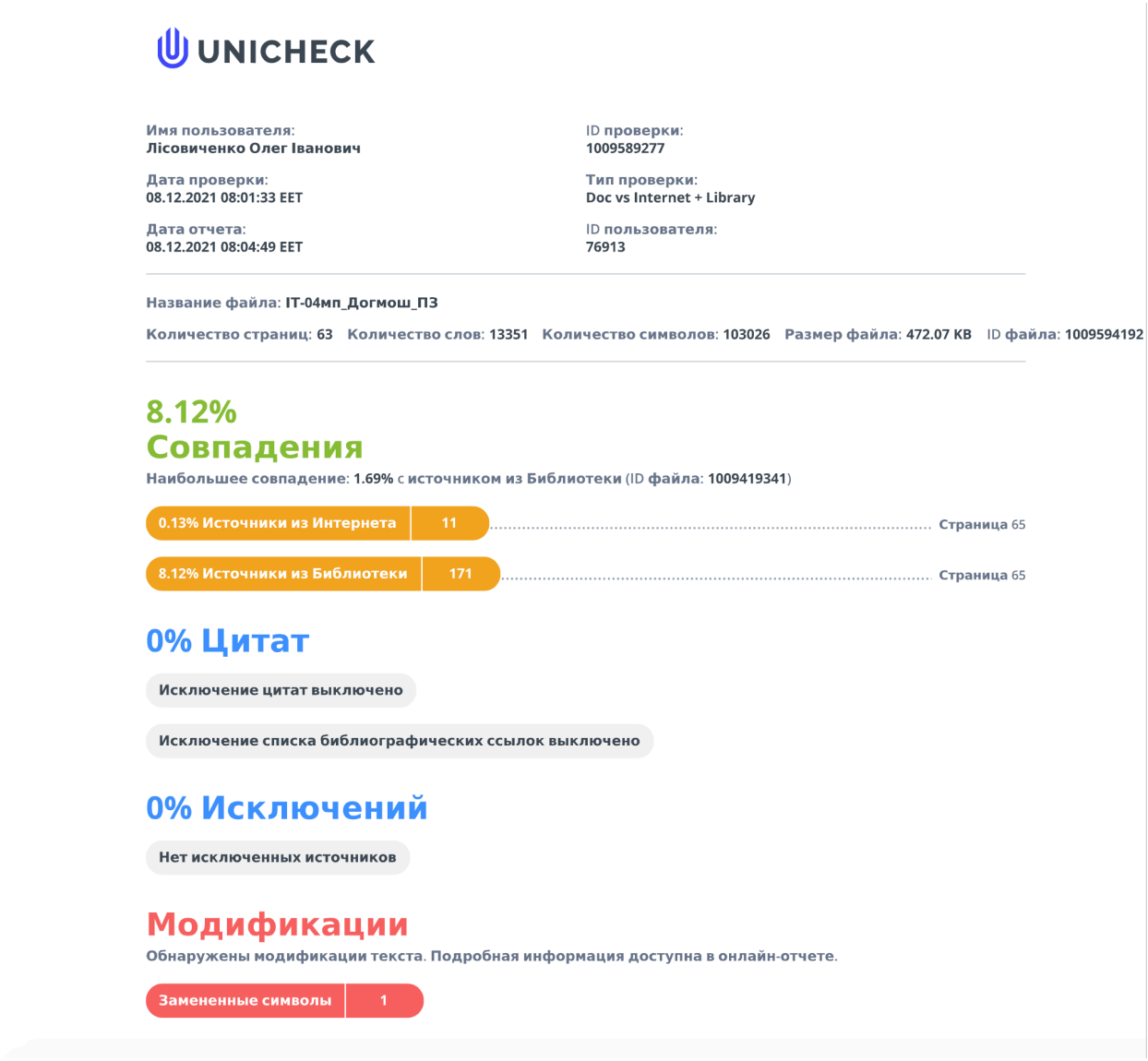
```
        this.notificationService.notifyConferenceStarted(conference);

        OpenViduConference openViduConference = this.openVidu.createConference();

        conferenceId = openViduConference.getId();
        token = openViduConference.generateToken(user.getName);

        log.info("token '{}' was succesfully retrieved from Media Server", token, conferenceId);

        responseJson.put(0, conferenceId);
        responseJson.put(1, token);

        tlChannelService.sendMessageToChannel("Розпочалася відеоконференція!\nТема конференції: \"" +
conference.getTitle() + "\"" +
                    "\nПосилання на конференцію: http://localhost:4200/#/");

        return new ResponseEntity<>(responseJson, HttpStatus.OK);
    }

    @RequestMapping(value = "/delete-user", method = RequestMethod.DELETE)
    public ResponseEntity<Object> removeUser(@RequestBody String conferenceName, @AuthenticationPrincipal
User user)
        throws Exception {

      log.info("Removing participant '{}' from the conference '{}'", this.user.getUsername(), conferenceName);

      JSONObject conferenceNameTokenJSON = (JSONObject) new JSONParser().parse(conferenceName);
      Long lessonId = (Long) conferenceNameTokenJSON.get("lessonId");

      if (this.lessonIdConference.get(lessonId) != null) {
        String conferenceId = this.lessonIdConference.get(lessonId).getConferenceId();

        if (this.conferenceIdUserIdToken.containsKey(conferenceId)) {
          if (this.conferenceIdUserIdToken.get(conferenceId).remove(this.user.getId()) != null) {
            log.info("User(username={}) was removed", this.user.getusername());
            if (this.conferenceIdUserIdToken.get(conferenceId).isEmpty()) {
              log.info("Last user removed from the conference. Conference '{}' empty and removed",
conferenceName);
              this.lessonIdConference.remove(lessonId);
            }
            return new ResponseEntity<>(HttpStatus.OK);
          } else {
            log.error("Conference TOKEN related to user(username={}) is not valid", this.user.getUsername());
            return new ResponseEntity<>(HttpStatus.INTERNAL_SERVER_ERROR);
          }
        } else {
          log.error("There was no conferenceId related to the lesson(id={})", lessonId);
          return new ResponseEntity<>(HttpStatus.INTERNAL_SERVER_ERROR);
        }
      } else {
        log.error("There was no conference for lesson(id={})", lessonId);
        return new ResponseEntity<>(HttpStatus.INTERNAL_SERVER_ERROR);
      }
    }
}
```

```
@RestController
@RequestMapping("/api/chat")
@Slf4j
public class ChatController {

   private final User currentUser;
   private final ChatService chatService;
   private final UserService userService;

   @Autowired
   public ChatController(User currentUser, ChatService chatService, UserService userService) {
      this.currentUser = currentUser;
      this.chatService = chatService;
      this.userService = userService;
   }

   @PutMapping("/messages/from/{from}/to/{to}")
   public ResponseEntity<Boolean> messagesAlreadyRead(@PathVariable long from, @PathVariable long to){
      log.info("messages was read from {} to {}", from, to);
      User userFrom = userService.getUserById(from);
      User userTo = userService.getUserById(to);
      boolean response = this.chatService.messageAlreadyRead(userFrom, userTo);
      return ResponseEntity.ok(response);
   }

   @GetMapping("/all/chats")
   public ResponseEntity<Page<UserChatDto>> getAllChats(@RequestParam int page, @RequestParam int size){
      Page<UserChatDto> usersChat = this.chatService.getAllChatUsers(this.currentUser, page, size);
      return ResponseEntity.ok(usersChat);
   }

   @GetMapping("/all/chats/user/{name}")
   public ResponseEntity<Page<UserChatDto>> getAllChatUsers(@PathVariable String name, @RequestParam int
page, @RequestParam int size){
      Page<UserChatDto> userChatDto = this.chatService.getAllChatUsersByName(this.currentUser, name, page,
size);
      return ResponseEntity.ok(userChatDto);
   }

   @GetMapping("/all")
   public ResponseEntity<Collection<ChatDto>> getAll() {
      Collection<ChatDto> allUserChats = this.chatService.getAllByUser(this.currentUser);
      return ResponseEntity.ok(allUserChats);
   }

   @PostMapping("/new/chat/user/{userId}")
   public ResponseEntity<ChatDto> newConversation(@PathVariable long userId, @RequestBody
ChatMessageDto firstMessage) {
      User recipient = this.userService.getUserById(userId);

      if (Objects.nonNull(recipient)) {
         ChatDto chatDto = this.chatService.newChat(this.currentUser, recipient, firstMessage);
         return ResponseEntity.ok(chatDto);
      } else {
         return ResponseEntity.notFound().build();
```

```
        }
    }

    @GetMapping("/chat/withuser/{userId}")
    public ResponseEntity<Collection<ChatDto>> getChatHistory(@PathVariable("userId") long userId) {
        User user = this.userService.getUserById(userId);
        Collection<ChatDto> userChats = this.chatService.getChatOfUser(this.currentUser, user);
        if (Objects.nonNull(userChats)) {
            return ResponseEntity.ok(userChats);
        } else {
            log.warn("chats with user(id={}) not found", userId);
            return new ResponseEntity<>(HttpStatus.NOT_FOUND);
        }
    }

    @PutMapping("/message/chat/{id}")
    public ResponseEntity<ChatDto> sendMessage(@PathVariable long id, @RequestBody ChatMessageDto
chatMessageDto) {
        ChatDto chatDto = this.chatService.getChatById(id);
        if (Objects.nonNull(chatDto)) {
            chatDto = this.chatService.sendMessage(chatMessageDto, chatDto);
            return ResponseEntity.ok(chatDto);
        } else {
            return ResponseEntity.notFound().build();
        }
    }

    @GetMapping("/unviewed/number/user/{userId}")
    public ResponseEntity<Long> getNumberOfNotViewedMessages(@PathVariable long userId){
        User other = this.userService.getUserById(userId);
        if(Objects.nonNull(other)){
            long unviewed = this.chatService.getUnviewedMessagesNumber(this.currentUser, other);
            return ResponseEntity.ok(unviewed);
        }
        else{
            return ResponseEntity.notFound().build();
        }
    }
}
```

ДОДАТОК Г

Результати перевірки роботи на співпадіння



UNICHECK

Имя пользователя:
Лісовиченко Олег Іванович

ID проверки:
1009589277

Дата проверки:
08.12.2021 08:01:33 EET

Тип проверки:
Doc vs Internet + Library

Дата отчета:
08.12.2021 08:04:49 EET

ID пользователя:
76913

Название файла: **IT-04мп_Догмош_ПЗ**

Количество страниц: **63** Количество слов: **13351** Количество символов: **103026** Размер файла: **472.07 KB** ID файла: **1009594192**

8.12%
Совпадения

Наибольшее совпадение: **1.69%** с источником из Библиотеки (ID файла: **1009419341**)

0.13% Источники из Интернета | 11 ..... Страница 65

8.12% Источники из Библиотеки | 171 ..... Страница 65

0% Цитат

Исключение цитат выключено

Исключение списка библиографических ссылок выключено

0% Исключений

Нет исключенных источников

Модификации

Обнаружены модификации текста. Подробная информация доступна в онлайн-отчете.

Замененные символы | 1