МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХЕРСОНСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
Факультет комп'ютерних наук, фізики та математики
Кафедра комп'ютерних наук та програмної інженерії

РЕАЛІЗАЦІЯ ВЗАЄМОДІЇ ПЛАТФОРМИ ВІДЕОКОНФЕРЕНЦІЙ ZOOM ІЗ ПЛАТФОРМОЮ KSU24

Кваліфікаційна робота (проєкт)
на здобуття ступеня вищої освіти "магістр"

Виконав: студент 2 курсу 241М групи
Спеціальність: 121 Інженерія програмного забезпечення
Освітньо-професійна програма: Інженерія програмного забезпечення другого (магістерського) рівня віщої освіти
Новіков Михайло Миколайович
Керівники: д.пед.н., к.ф.-м.н., професор Співаковський О.В.;
к.ф-м.н., доцент Єрмолаєв В.А.
Рецензент: Кльонов Д. М. фріланс, full stack developer

Івано-Франківськ – 2024

ЗМІСТ

ВСТУП

На сьогодні більшість провідних закладів вищої освіти України прагнуть до повної цифровізації всіх процесів, що відбуваються в них, від навчання до документообігу. Це вимагає створення й розвитку власної університетської системи, що дозволить усім працівникам і здобувачам освіти взаємодіяти між собою, навіть не знаходячись за фактичною адресою закладу і навіть не перебуваючи в межах однієї країни.

Цифровізація стає ключовим елементом стратегії розвитку сучасних університетів, оскільки вона забезпечує гнучкість і доступність освітнього процесу для всіх учасників. Університети України, як і у всьому світі, поступово переходять на новий рівень взаємодії, коли фізична присутність на кампусі більше не є обов'язковою для участі в академічному житті. У цьому контексті важливим є не лише впровадження електронних ресурсів, а й інтеграція всіх освітніх і адміністративних процесів в єдину цифрову екосистему. Така екосистема має забезпечити безперебійний обмін інформацією, доступ до освітніх матеріалів, проведення лекцій, консультацій та зустрічей, незалежно від місця перебування користувачів.

Важливо зазначити, що для ефективного функціонування такої системи необхідно забезпечити її інтеграцію з сучасними платформами для відеоконференцій, такими як Zoom. Це дозволить університетам оптимізувати процес проведення дистанційних лекцій, семінарів та конференцій, одночасно забезпечуючи високий рівень комунікації між викладачами, студентами та адміністрацією. Крім того, використання таких інструментів створює нові можливості для змішаного навчання, коли поєднуються традиційні та онлайн-формати. Це є важливим кроком до реалізації концепції гнучкого навчального процесу, який здатний адаптуватися до сучасних викликів.

У Херсонському державному університеті, наприклад, було спроєктовано й представлено такий сервіс – онлайн платформу KSU24 [1], над розробкою якого працюють співробітники й студенти закладу освіти, а також певна кількість розробників зі сторонньої компанії. Серед її функціоналу присутні такі елементи, як електронні журнали та електронний розклад [2], що були спроєктовані й розроблені в працях [3, 4].

Оскільки більшість занять у Херсонському державному університеті проводяться за допомогою онлайн платформи проведення зустрічей Zoom, то було прийнято рішення розширити наявний функціонал електронних журналів у KSU24 таким чином, щоб викладачі мали можливість створити Zoom зустріч, запустити її у відведений час, а студенти – доєднатися до заняття, і все це за допомогою одного, максимум двох кліків мишею. Окрім того, після завершення заняття викладачеві (а також навчальному відділу) пропонується статистика, що відображає присутність здобувачів, а також проміжки часу, в які вони були присутніми на уроці.

Актуальність теми кваліфікаційної роботи полягає в тому, що подібні інтеграційні сервіси-адаптери не є розповсюдженим явищем, а ті, що існують, переважно представлені як вбудовані частини (модулі або мікросервіси) конкретних університетських систем, тому розробка такого сервісу як окремого самостійного додатку (або мікросервісу) може бути корисною не лише для одного закладу освіти, а й для інших, які можуть зацікавитися подібною розробкою.

Мета й завдання кваліфікаційної роботи полягає в тому, щоб спроєктувати й розробити сервіс-адаптер, який поєднує систему KSU24 та Zoom API задля впровадження бізнес-процесу організації та проведення конференцій, моніторингу та аналітики відповідного заходу в цифрових журналах.

Об’єктом дослідження кваліфікаційної роботи є програми-сервіси для сполучення систем проведення онлайн-конференцій та університетських середовищ.

Предметом дослідження кваліфікаційної роботи є сервіс інтеграції функціоналу Zoom API у систему KSU24 з модулем “Академічні журнали”.

РОЗДІЛ 1
Огляд аналогів сервісів інтеграції Zoom API до університетських систем

1.1. Приклади інтеграції Zoom API у відомих системах забезпечення онлайн освіти

Питання інтеграції платформ проведення відеоконференцій до університетських середовищ (або так званих систем управління навчанням, Learning Management System або скорочено LMS) постало в той момент, коли розробники програмного забезпечення намагалися зрозуміти, як спростити комунікацію між учасниками освітнього процесу, не покидаючи при цьому сайту або додатку LMS, та зменшити кількість дій, що мають постійно виконуватися задля організації онлайн зустрічі або заняття.

Система управління навчанням (Learning Management System, LMS) — це програмне забезпечення, зазвичай у хмарному середовищі, яке забезпечує створення освітніх матеріалів у цифровому форматі та організацію онлайн-навчання.

LMS-платформа може застосовуватися для: реалізації та продажу навчальних курсів або інших інформаційних продуктів для зацікавлених користувачів; переведення навчального процесу закладів освіти в онлайн-формат; підготовки та підвищення кваліфікації працівників. [5]

Оскільки такі системи зазвичай є відкритими для розширення, то їхні власники прийняли рішення створити додаткові модулі, що забезпечать зв’язок системи з інтерфейсом певних платформ проведення відеоконференцій. Ці модулі можуть бути представлені або у вигляді частини самої системи, або плагіну, що потрібно окремо завантажити й установити. Так як у цій роботі розглядається інтеграція саме платформи

для проведення відеоконференцій Zoom, то ми розглядатимемо в подальшому лише ті LMS, що підтримують взаємодію з Zoom.

У першу чергу розглянемо одну з найпопулярніших систем управління навчанням Moodle [6]. Це система з відкритим вихідним кодом, яка забезпечує комунікацію між студентами й викладачами, створення курсів дистанційного навчання, а також проведення очного навчання. Moodle було вперше випущено у 2002 році й продовжує оновлюватися, тобто отримувати оновлення безпеки, нові функціональні розширення тощо. У якості мови програмування використовується PHP, що працює у зв'язці з реляційними базами даних (наприклад, Microsoft SQL Server, Postgre, MySQL). Moodle також має партнерські стосунки з українською компанією ТОВ "Техноматика" [7].

Серед основної функціональності платформи можна відзначити наступне:

- Робота над завданнями та здача на перевірку
- Форуми для обговорень
- Завантаження файлів
- Виставлення оцінок за здані завдання
- Листування за допомогою текстових повідомлень (чат)
- Календар
- Стрічка новин, анонси - можливо обрати рівень поширення: курс, група, весь сайт
- Організація контролю знань у вигляді тестових модулів
- Вікі – інформаційні сторінки, де зберігається більш детальна інформація на потрібну тематику

Moodle є модульною платформою з великою спільнотою розробників, що активно розміщують нові розширення й пропонують зміни (а також виправлення помилок) для вже існуючих. Модулі дозволяють, по-перше, додавати нові види діяльності, у тому числі ігри, по-друге, запровадити додаткові типи даних, які можуть бути розміщені

на сервері, а також змінювати теми оформлення сайту й методи аутентифікації та/або зарахування на курси. Студенти мають більш обмежений доступ у режимі учасників курсів з можливістю завантажувати виконані завдання, а викладачі можуть створювати курси й редагувати їх відповідно до навчального плану. [8]

Серед доступних модулів для завантаження існує такий, що додає до розгорнутої системи функціонал для взаємодії з Zoom. Він має назву Zoom meeting і підтримується розробниками на ім’я Jonathan Champ та Steve Bader. [9] За описом плагіну на сайті, він дозволяє створювати й редагувати зустрічі й вебінари, виконувати синхронізацію, а також створювати й відновлювати резервні копії. З повною документацією плагіну, а також з його вихідним кодом можна ознайомитися в GitHub репозиторії, до того ж, за наявності відповідних знань і навичок дозволено пропонувати зміни через створення гілок і від них - запитів на з’єднання (pull requests). [10]

Розберемо основні елементи інтерфейсу користувача плагіну, а також його функціональність. Для того, щоб використовувати плагін за призначенням, потрібно його завантажити з офіційного сайту Moodle, потім знайти на сервері спеціальну папку /mod/zoom і скопіювати туди завантажені файли, і тоді зі сторінки дистанційного курсу буде можливість працювати з конференціями або вебінарами. Окрім цього, необхідно переконатися, що для сервісу доступний корпоративний обліковий запис Zoom, що має права адміністратора. Для повноцінної роботи акаунт повинен мати план підписки для закладів освіти (в оригіналі Educational), так як базова безкоштовна версія має обмеження, наприклад, на можливість створювати конференції не від імені поточного користувача, тобто в ній зустріч може бути зареєстрована лише на той акаунт, з якого вона створюється. Повний список обмежень доступний для перегляду й ознайомлення на офіційному сайті Zoom Developers. [11] Крім самого облікового запису, потрібний також

створений заздалегідь Zoom застосунок (Zoom App), який надає клієнтський ідентифікатор, секретний ключ та ідентифікатор акаунту для роботи з плагіном. Ці дані потрібно внести в налаштуваннях плагіну до відповідних полів і зберегти, таким чином надаючи плагіну повноваження працювати з зустрічами від імені корпоративного акаунту.

Отже, у дистанційному курсі з'явиться секція "Зустрічі Zoom", що має такий вигляд, як на рис. 1.1.

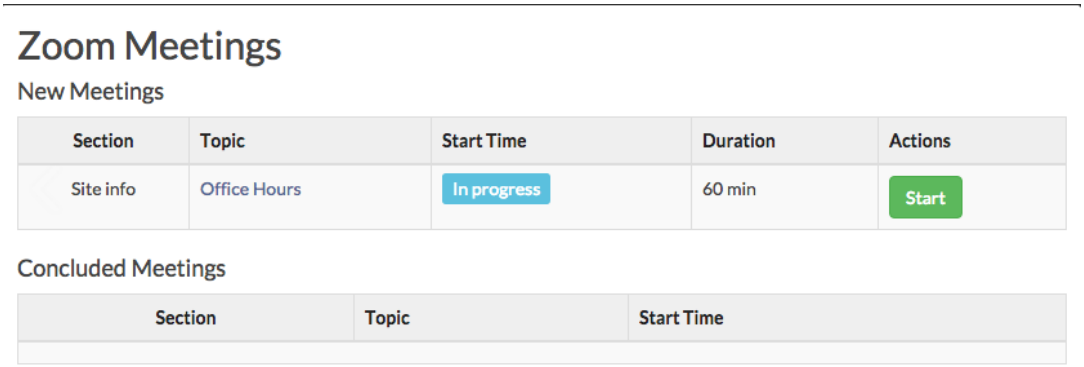

Рисунок 1.1. Секція "Зустрічі Zoom" у Moodle

На ілюстрації видно, що всі конференції поділяються на нові (New) та завершені (Concluded). Коли ми створюємо нову зустріч, вона відображається в таблиці нових, а після її проведення переміщується до завершених. Як видно з рисунку, кожна зустріч має свою тематику, тривалість, також відображається час початку або поточний стан (у прикладі конференцію вже розпочато за часом), а також є секція з діями, які можливо виконати щодо зустрічі. Для завершених подій відображається значно менше інформації задля спрощення вигляду таблиці й приховування не потрібної в цей момент часу інформації.

Для того, щоб створити нову зустріч у межах дистанційного курсу, необхідно натиснути відповідну кнопку, яка відображає форму додавання, зображену на рис. 1.2.

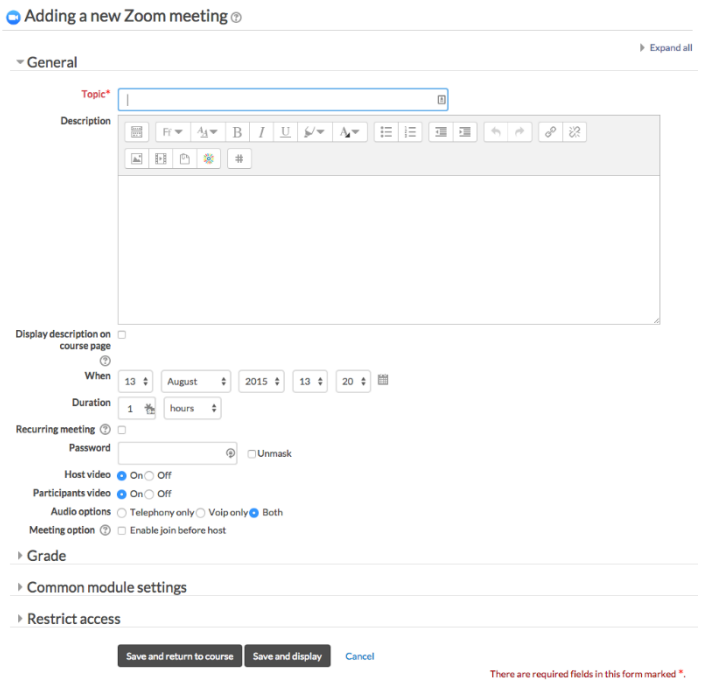

Рисунок 1.2. Форма створення нової конференції у Moodle

Для цього потрібно задати тематику зустрічі, час її початку з точністю до хвилин, тривалість заходу, тип (якщо поставити прапорець у рядку Recurring meeting - Повторювана зустріч - потрібно буде також указати, з якою періодичністю вона має відбуватися, наприклад, в певні дні тижня), пароль (наполегливо рекомендується вказувати пароль для кожної зустрічі задля запобігання неавторизованого доступу). За бажанням є можливість за замовчуванням дозволити або заборонити доступ до увімкнення веб камери організаторові та/або учасникам, а також дозволити або заборонити учасникам доєднуватися до того, як доєднається організатор. Досить важливими з точки зору безпеки є також налаштування заборони доступу, в яких можна задати, з яких доменів у адресах електронної пошти можна доєднуватися до конференції, або взагалі обмежити вхід лише для корпоративних облікових записів. Після створення плагін запропонує перейти на сторінку з деталями доданої зустрічі. На ній буде представлена таблиця, у якій міститиметься введена раніше інформація, а також згенероване посилання для долучення до конференції її учасників і кнопка початку (Start Meeting). За бажанням користувач може також повернутися до

списку всіх створених зустрічей. Після завершення конференції її організатор може повернутися до Moodle та відобразити статистику. В окремій таблиці буде відображено всіх учасників, хто був присутнім, їхні імена, час долучення та виходу, а також скільки часу кожен студент знаходився на дзвінку.

Якщо коротко підсумувати описаний функціонал плагіну, то цілком доречно зазначити, що він реалізовує зазначені розробниками можливості в повному обсязі, використовуючи відкриту платформу Moodle, яка знаходиться в користуванні значною кількістю відомих світових закладів вищої освіти, наприклад, Кембріджський університет, Каліфорнійський, Токійський, Гонконгський. З українських унівеситетів, зокрема Херсонщини, платформа інтегрована в середовище Херсонської державної морської академії (система ХДМА ОНЛАЙН [12]) й Херсонського державного університету (система KSU Online [13]).

Розглянемо ще одну аналогічну LMS під назвою EdisonOS. Це платформа, що також інтегрується в Zoom і надає викладачам інструментарій для роботи з курсами, їх створення, редагування й проведення. Інтерфейс системи вже за замовчуванням містить функціонал для роботи з Zoom, тому студенти й викладачі мають можливість працювати з курсами й проводити онлайн заняття без додаткових модулів. [14]

Інтеграція Zoom з LMS EdisonOS створює зручне середовище для проведення онлайн-занять та спрощує навчальний процес як для викладачів, так і для студентів. EdisonOS, як платформа для управління навчанням, дозволяє користувачам легко планувати і проводити відеоконференції безпосередньо з системи, забезпечуючи безперервний доступ до навчальних матеріалів та онлайн-зустрічей.

Після інтеграції Zoom з EdisonOS викладачі можуть створювати онлайн-сесії прямо з інтерфейсу LMS, автоматично додаючи ці зустрічі

до курсів, які вони ведуть. Це значно спрощує процес організації навчальних занять, оскільки всі дії виконуються в єдиній системі, без необхідності переходити на інші платформи. Студенти, в свою чергу, отримують доступ до запланованих зустрічей через свої акаунти EdisonOS, де вони можуть приєднатися до сесій в один клік. Приклад планування таких сесій зображений на рис. 1.3.

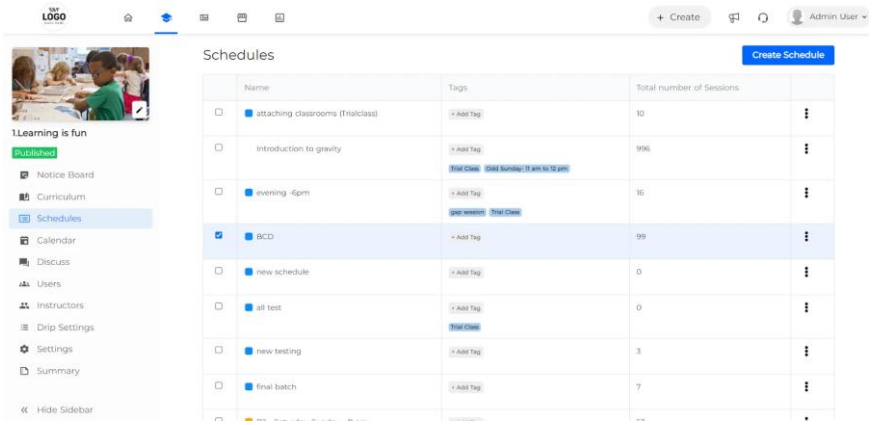

Рисунок 1.3. Вікно планування в EdisonOS

Також інтеграція дає можливість записувати сесії, а пізніше зберігати ці записи на платформі. Це дозволяє студентам, які не змогли бути присутніми на занятті, переглянути запис і не пропустити важливий матеріал. Крім того, всі дані про відвідуваність, тривалість сесій та інші важливі статистичні показники автоматично зберігаються в LMS, що полегшує відстеження прогресу і результатів студентів.

Варто також зазначити, що EdisonOS, на відміну від Moodle, є закритою платформою, тому його інтеграція з Zoom відбулася більш централізовано, тобто безпосередньо в програмному коді системи. З одного боку, це дозволяє користувачам одразу перейти до роботи з Zoom без додаткових конфігурацій і налаштувань; з іншого ж боку, це зменшує гнучкість функціоналу й вимагає від розробників і власників продукту активно спостерігати за змінами в Zoom API й оперативно додавати нові можливості або виправляти помилки, що виникають внаслідок критичних змін (так званих breaking changes) і впливають на якість роботи системи й відгуки користувачів. Тим не менше, завдяки

інтеграції Zoom в EdisonOS, учасники можуть також використовувати різноманітні функції Zoom, такі як чат, функція підняття руки, демонстрація екрана і робота в групах. Це створює інтерактивне навчальне середовище, де учасники можуть активно брати участь у процесі, задавати питання і отримувати відповіді в реальному часі.

1.2. SWOT аналіз систем дистанційного навчання з урахуванням інтеграції Zoom API

На основі наведених у попередньому підрозділі даних про системи Moodle та EdisonOS, визначимо їхні переваги й недоліки й розглянемо підходи, що були обрані розробниками цих LMS.

Розпочнемо з Moodle. У користувачів цієї системи є можливість проводити навчання в режимі реального часу через Zoom в рамках курсів Moodle. На курсах можливі наступні дії. Якщо учасниками освітнього процесу використовується лише Zoom, студентам і викладачам потрібно надати дозвіл на вхід і підтвердити присутність, коли учасники зайдуть. Крім того, є ймовірність того, що при великій кількості учасників деякі з них можуть залишити зустріч, і це не буде зафіксовано жодним чином. Однак у Moodle буде записана детальна інформація про участь, наприклад, час долучення здобувачів, як довго вони брали участь і чи залишилися до завершення заняття, а не лише сам факт їхньої присутності або відсутності. Для того, щоб налаштувати Zoom, зазвичай потрібно ввійти в Zoom, запланувати зустріч і розповсюдити інформацію про зустріч (URL-адреса або унікальний ідентифікатор та пароль) учасникам. Однак можуть виникнути такі проблеми, як неможливість отримати або перевірити електронну пошту, через що учасники не зможуть приєднатися до занять. Але якщо Zoom буде інтегрований до Moodle, організатор (викладач) може запланувати зустріч на Moodle. Це усуває необхідність надсилати учасникам

інформацію про кімнату для нарад, оскільки вона буде налаштована під час курсу. Учасники можуть просто увійти в систему Moodle зі своїми обліковими даними від університету, відкрити курс і натиснути на дію Zoom, для того щоб одразу приєднатися до зустрічі.

Після проведення сеансу онлайн навчання за допомогою Zoom може виникнути необхідність провести тест для оцінки знань. Досить популярною практикою є підготовка перевірочних завдань у текстовому форматі, наприклад, Word, і розміщення в дистанційному курсі. Але це вимагає часу й зусиль, по-перше, на розповсюдження завдань серед студентів, потім на саму роботу й збирання зданих робіт, а також на їхню перевірку викладачем.

У випадку з Moodle є можливість налаштувати тест як наступну дію після завершення онлайн заняття. Після завершення онлайн навчання за допомогою Zoom учасники можуть натиснути на тестову дію та відповісти на налаштовані запитання. Як тільки тест завершено, результати автоматично в той самий час оцінюються, і викладач може переглянути їх. Якщо результати тесту учасника опускаються нижче певного порогу, вони можуть легко переглянути записане відео онлайн заняття за допомогою функції запису Zoom і повторно скласти тест пізніше. [15]

З недоліків у Moodle Zoom плагіну виділяються наступні: сама платформа Moodle є досить об’ємною й займає багато місця на сервері, а також під час роботи вимагає як мінімум 8 ГБ оперативної пам’яті, окрім того, при великому навантаженні й одночасній кількості користувачів це може значно знизити продуктивність системи. Також користувачі відзначають, що інтерфейс користувача системи є досить застарілим і менш інтуїтивно зрозумілим у порівнянні з більш сучасними LMS. Окрім того, Moodle не має офіційної технічної підтримки, що вимагає від закладів освіти створення власних команд підтримки, що консультуватимуть учасників освітнього процесу.

Якщо звернутися до досвіду Херсонського державного університету, то на його платформі KSU Online, розробленій на базі рушію Moodle, також установлено описаний вище плагін Zoom meetings. Але варто зазначити, що ця система використовується не всіма викладачами й студентами навчального закладу, до того ж, не всі користувачі поінформовані відносно наявності подібного функціоналу системи, що вимагало б додаткових витрат часу на навчання спільноти університету користуванню цим модулем. До того ж, ураховуючи зазначені в цьому підрозділі недоліки LMS Moodle й обмеженість функціоналу самого плагіну, було вирішено лише взяти цю розробку в якості прикладу взаємодії з Zoom API для подальшого дослідження й проєктування відповідного сервісу.

Тепер перейдімо до системи Edison OS. Вона виділяється як провідна система управління навчанням для інтеграції з Zoom, пропонуючи потужний і зручний досвід як для викладачів, так і для учнів. Визначимо критерії, чому EdisonOS є оптимальним вибором для інтеграції Zoom.

По-перше, це бесперешкодний процес інтеграції EdisonOS з Zoom. Платформа розроблена для комфортної роботи з Zoom, гарантуючи, що всі можливості повністю функціональні й доступні в середовищі LMS. Повна інтеграція усуває технічні бар’єри, дозволяючи викладачам зосередитися на наданні якісного контенту, не приділяючи додаткового часу технічним складнощам.

По-друге, це централізоване навчання. EdisonOS дозволяє викладачам керувати всією навчальною діяльністю — планувати сеанси Zoom, проводити заняття в реальному часі, ділитися записами та розповсюджувати матеріали курсу — з єдиної централізованої платформи. Це гарантує, що учні отримають цілісний досвід, коли все, що їм потрібно, доступне в одному місці, покращуючи навчальний процес.

По-третє, це розширена аналітика та звітність. За допомогою EdisonOS викладачі можуть використовувати розширену аналітику й інструменти звітності для моніторингу та оцінки залучення й ефективності учнів. Інтеграція з Zoom надає вичерпні дані про відвідуваність, участь і взаємодію під час онлайн сесій, дозволяючи викладачам приймати рішення на основі наявних даних й адаптувати свої стратегії навчання для покращення результативності роботи студентів.

По-четверте, це масштабованість і гнучкість. EdisonOS створено для задоволення потреб як невеликих груп, так і великомасштабних онлайн курсів. Незалежно від того, чи викладач керує невеликим класом чи великою групою, інтеграція EdisonOS із Zoom гарантує, що платформа залишається швидкою та ефективною. Крім того, гнучкість, яку пропонує EdisonOS, дозволяє учням отримувати доступ до вмісту курсу та сеансів Zoom у зручний для них спосіб, задовольняючи різноманітні навчальні уподобання.

По-п'яте, це зручний інтерфейс. EdisonOS має інтуїтивно зрозумілий і простий у навігації інтерфейс, що робить його доступним для викладачів і учнів будь-якого рівня технічної підготовки. Інтеграція з Zoom розроблена таким чином, щоб бути простою та зрозумілою, з мінімальними потребами в налаштуванні. Такий зручний підхід значно скорочує траєкторію навчання та гарантує, що як викладачі, так і здобувачі освіти зможуть швидко освоїти платформу.

І остання перевага – це комплексна підтримка та ресурси. EdisonOS пропонує широку підтримку та ресурси задля допомоги викладачам у максимальній реалізації інтеграції Zoom. EdisonOS гарантує, що користувачі отримають допомогу, необхідну для створення успішного навчального онлайн середовища, від детальних посібників до швидкої підтримки клієнтів. [14]

Важливо також зазначити й описати недоліки LMS EdisonOS. Одним із головних недоліків є обмежена можливість налаштування інтерфейсу та функціональності, через що освітні установи чи індивідуальні викладачі мають обмежений контроль над виглядом і налаштуваннями системи відповідно до власних потреб. Це може бути незручним, якщо потрібно адаптувати систему під конкретний навчальний процес або корпоративний стиль.

Ще одним недоліком EdisonOS є залежність від стабільного інтернет-з’єднання для доступу до функцій платформи. Під час роботи у віддалених районах або при нестабільному підключенні до інтернету це може впливати на ефективність навчання, оскільки доступ до матеріалів та інтерактивних елементів стає складним. Крім того, деякі користувачі відзначають, що функції платформи можуть бути складними для освоєння, особливо для тих, хто не має досвіду роботи з LMS-системами, оскільки інтерфейс EdisonOS іноді не досить інтуїтивно зрозумілий.

Також варто зазначити, що EdisonOS може вимагати додаткових ресурсів для підтримки та обслуговування, особливо якщо необхідно інтегрувати платформу з іншими системами, такими як системи управління університетами чи сторонні додатки. Це може створювати додаткові витрати для навчальних закладів і бізнесу, особливо якщо потрібно залучати технічних спеціалістів для підтримки стабільної роботи платформи та вирішення можливих проблем.

Якщо підсумувати наведені аргументи за й проти використання різних LMS, то можна констатувати, що не існує системи, яка б повністю покривала потреби всіх закладів вищої освіти, оскільки ці сервіси, хоч і мають спільне призначення, проте суттєво відрізняються запропонованим функціоналом, технічними характеристиками й навіть видами ліцензії. Перед керівництвами університетів постає непроста задача вибору оптимальної платформи, яка могла б у більшості

задовольнити потреби викладачів та здобувачів, або ж розробки принципово нової системи, яка б базувалася на власній інфрастуктурі закладу й розвивалася б відповідно до затвердженого адміністрацією плану. Що стосується інтеграції можливостей Zoom, то у випадку з проєктуванням окремої платформи для університету окремою задачею також є розробка модуля-адаптера між Zoom API та університетською системою, яка проєктується.

РОЗДІЛ 2
Проєктування сервісу взаємодії з конференціями ConferenceAPI

2.1. Опис функціональних вимог до сервісу

У Херсонському державному університеті існує й розвивається система управління навчанням KSU24. Серед її функціоналу присутні такі сторінки, як розклад занять та електронний журнал. Інтерфейс сторінки з журналом зображений на рис. 2.1.

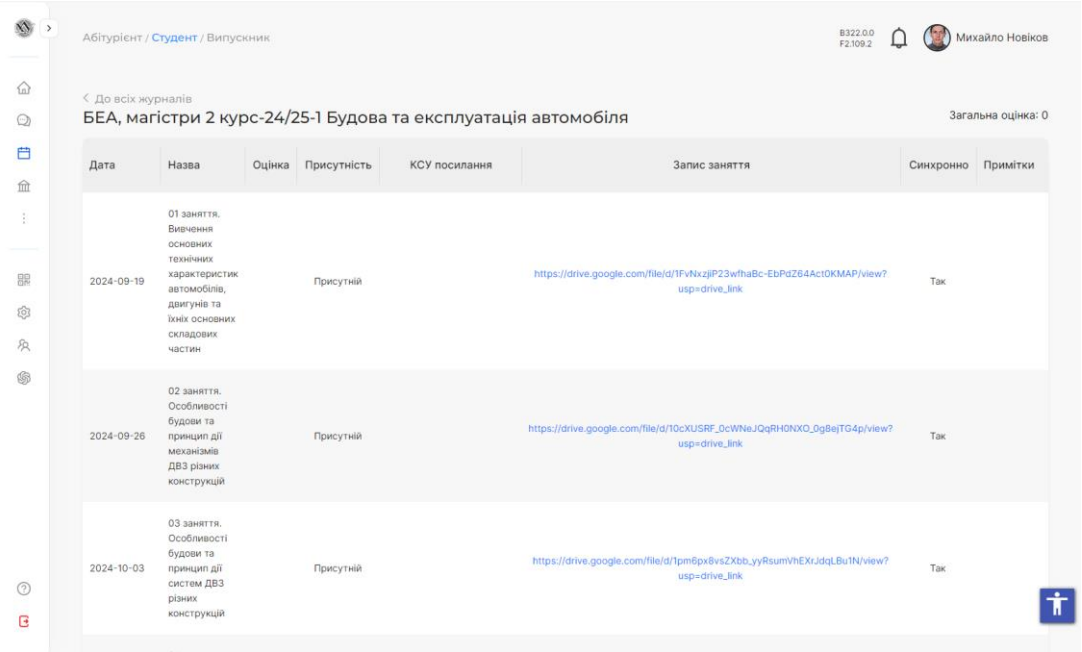

Рисунок 2.1. Інтерфейс вікна журналу (вигляд від ролі Студент)

Розробниками системи було задумано, що викладач може створювати журнали за власними дисциплінами для окремих навчальних груп. Кожна дисципліна має два журнали – лекційний та практичний. В обидвох видах журналів викладач має створити записи про заняття, проставляє відмітки про присутність або відсутність кожного студента, а також за необхідності вказує кількість балів, які заробив студент за заняття (або за певний вид діяльності на ньому).

Загальним призначенням цього модулю є фіксація успішності здобувачів освіти, а також контроль відвідування занять. [3]

Командою розробників було вирішено розширити можливості модулю таким чином, щоб викладачі й студенти могли долучатися до занять максимально швидко, використовуючи один-два кліки мишею. Відповідно до документації платформи Zoom, існує декілька способів долучення до зустрічей. Це може бути вхід за адресою електронної пошти, за запрошенням через календар, запрошення через обмін миттєвими повідомленнями, з веб браузера, з настільного або мобільного додатку Zoom тощо. Також існує варіант неавторизованого входу лише за ім’ям, але у випадку з закладами освіти це є неприпустимим. [16]

Після того, як було прийнято рішення про інтеграцію Zoom до KSU24, були також сформовані функціональні вимоги до сервісу в наступному вигляді:

- Створення викладачами Zoom конференцій до занять у журналі. Параметри зустрічі мають бути наступними: тематика (обов’язковий), тип (одноразова зустріч, запланована, повторювана без фіксованого часу, повторювана з фіксованим часом), дата й час початку зустрічі (опціонально, залежить від типу), тривалість зустрічі (у хвилинах), пароль (генерується автоматично, обов’язковий). Окрім базових, потрібно надати можливість задати додаткові налаштування конференції: зал очікування (увімкнути/вимкнути), можливість долучення учасників раніше за організатора, стан мікрофона під час долучення, чи потрібна автентифікація на зустрічі, які домени електронних адрес дозволені для долучення до зустрічі, а також список адрес користувачів, запрошених до участі.

- Редагування викладачами параметрів створеної Zoom конференції. Параметри в точності відповідають тим, що використовуються для створення конференцій.
- Видалення зустрічей за ідентифікатором зустрічі (на випадок, якщо в зустрічі немає потреби або вона створена помилково).
- Отримання інформації про конкретну зустріч за її ідентифікатором. Окрім уже відомих параметрів зі створення й редагування зустрічей, потрібні також наступні: електронна адреса організатора зустрічі, посилання для початку зустрічі (для організатора), посилання для долучення до зустрічі (для здобувачів освіти).
- Отримання за ідентифікатором завершеної конференції статистики по всіх її учасниках (як викладачу, так і студентах), а саме: електронна адреса учасника, час долучення до дзвінку, час залишення дзвінку, яка причина виходу, роль учасника.

Процеси взаємодії акторів (викладачів, студентів) та сервісів між собою має наступний вигляд. Викладач зі сторінки електронного журналу за допомогою кнопки Створити ініціює створення зустрічі за параметрами, що надає KSU24, та надсилає запит до системи. Система, в свою чергу, надсилатиме запит до нового сервісу, що сформує правильний авторизований запит до Zoom API. Після обробки даних Zoom API поверне результат, і якщо він успішний, то новий сервіс повідомить про це KSU24, надавши також дані про створену зустріч. Таким самим чином, тобто в такому самому порядку, відбуватимуться інші дії, пов'язані з Zoom. Детальніше з цими процесами можна ознайомитися за допомогою діаграми послідовності, побудованої відповідно до вимог.

Діаграма послідовності – це вид UML діаграм, що визначає взаємодії складових частин системи, дотримуючись упорядкування за

часом. Вона відповідно до своєї назви відображає об’єкти, які пов’язані між собою послідовностями. Вертикальні лінії означають паралельно запущені процеси або об’єкти, а горизонтальні стрілки – це обмін повідомленнями між частинами в тому порядку, якому вони були відправлені. Сутності, що мають бути відображені на такій діаграмі:

- Повідомлення
- Ті, хто викликає повідомлення – актори
- Позначення циклів
- Опціонально – значення, що повертаються

Читання такої діаграми має відбуватися від лівого верхнього кута вниз і праворуч. Мова UML суттєво вдосконалила можливості діаграм послідовності. Основу цих покращень становить концепція фрагментів взаємодії, які є невеликими частинами взаємодії в межах загальної структури. Об’єднання кількох фрагментів взаємодії дає змогу формувати різні комбіновані фрагменти, що використовуються для моделювання складних аспектів взаємодій, таких як паралелізм, умовні розгалуження та необов’язкові взаємодії. [17]

2.2. Технічні й системні вимоги до ConferenceAPI

Оскільки функціональні вимоги до сервісу сформовані, потрібно визначити також технічні й системні. Для того, щоб це зробити, потрібно звернутися до принципу роботи системи KSU24. Вона побудована в традиційній клієнт-серверній конфігурації REST з фронтенд та бекенд додатками в поєднанні з реляційною базою даних PostgreSQL та декількома іншими видами баз даних, зокрема NoSQL.

REST (Representational State Transfer), що перекладається як «передача репрезентативного стану», являє собою стиль проєктування для побудови розподілених систем, заснований на певних обмеженнях, які забезпечують їх ефективність та масштабованість. У REST-

архітектурі центральною концепцією є ресурс, який представляє конкретні об’єкти чи дані, до яких здійснюється доступ за допомогою уніфікованих ідентифікаторів.

Основні принципи REST включають:

- клієнт-серверну модель, де чітко розділені функції клієнта та сервера для покращення модульності системи;
- беззбережену взаємодію, що забезпечує незалежність запитів, дозволяючи кожному запиту містити всю необхідну інформацію для його виконання;
- стандартизований інтерфейс, який спрощує інтероперабельність та підтримку.

Цей набір принципів сприяє підвищенню продуктивності системи, спрощенню обслуговування і покращенню масштабованості, що робить REST популярним вибором для розробки вебсервісів та API. [18]

Бекенд частина, з якою буде побудовано основну взаємодію нового сервісу, заснована на базі фреймворку Django (мова програмування Python). Django — високорівневий Python-фреймворк з відкритим кодом, призначений для розробки вебсистем. Назву фреймворку було обрано на честь відомого джазового музиканта Джанго Рейнхардта, що відображало музичні вподобання одного із засновників проєкту. Сайти, розроблені на Django, будуються з одного або декількох модулів, які рекомендується розробляти як окремі компоненти. Такий підхід до модульності є однією з ключових архітектурних відмінностей Django порівняно з іншими фреймворками, як-от Ruby on Rails. Архітектурна модель Django має подібність до патерну «Модель-Вигляд-Контролер» (MVC). Однак функція, яка в класичній моделі MVC називається «контролер», у Django називається «вигляд» (view), тоді як роль «вигляду» MVC виконує компонент під назвою «шаблон»

(template). Відповідно, розробники Django визначають цей підхід як MTV — «Модель-Шаблон-Вигляд». [19]

Усі складові системи будуються та розміщуються на хмарних серверах за допомогою Docker контейнерів, що забезпечує модульність, тобто незалежність кожного компоненту на випадок виникнення критичних помилок. [20]

Для того, щоб мінімізувати втручання до вихідного коду API KSU24, найкращим рішенням буде створення окремого сервісу у вигляді API, що матиме прямий зв’язок із Zoom API та обмінюватиметься даними з API системи. До того ж, це забезпечить зручність розгортання, оскільки не вимагатиме перезавантаження основної системи. Воно може знадобитися лише під час упровадження змін до KSU24, що дозволять використовувати новий сервіс методом обміну даними через HTTP-запити.

Оскільки основна система використовує принцип контейнеризації через Docker, то є цілком доречним запровадити аналогічні можливості в новому сервісі. Це, по-перше, забезпечить зручне розгортання змін у сервісі, а по-друге, підтримає загальну архітектуру KSU24 як цілісної системи.

Усі компоненти KSU24, а саме фронтенд та бекенд додатки, використовують систему контролю версій Git та GitHub. Це надає можливість відстежувати історію змін, а також за допомогою поєднання з Docker оперативно підвантажувати новий програмний код і збирати його на сервері. Для сервісу взаємодії з Zoom також буде використано приватний GitHub репозиторій, створений співробітниками відділу цифрової інфраструктури Херсонського державного університету. [21] Його приватність обумовлюється тим, що додаток містить секретні ключі для доступу до Zoom API через акаунт університету, які, очевидно, не мають потрапити до сторонніх пристроїв та осіб.

Стосовно технологій, що мають бути використані для розробки сервісу, то було обрано мову програмування C# і фреймворк ASP.NET Core Web API версії 8 (останньої на момент розробки, версія 9 знаходиться в попередньому доступі). ASP.NET Core Web API — це структура для створення масштабованих і високопродуктивних Restful Web Services (API) за допомогою платформи ASP.NET Core. Це дозволяє розробникам створювати надійні та гнучкі API, які можуть використовувати різні клієнти, наприклад веб-додатки, мобільні додатки, настільні додатки та сторонні служби. [22]

Вибір фреймворку обумовлений наступними критеріями. По-перше, це платформа з відкритим вихідним кодом, що має велику спільноту й активно підтримується не тільки самими розробниками, а й небайдужі ентузіасти. По-друге, він є кросплатформовим, а отже, може бути розгорнутий на будь-яких серверах, що використовують різні операційні системи (з найпопулярніших – Windows та Linux), використовуючи, наприклад, контейнери Docker або подібні. По-третє, фреймворк надає широкий вибір можливостей для розробки, такі як асинхронне програмування, управління пам’яттю тощо. По-четверте, ASP.NET Core відзначається високою продуктивністю порівняно з попередніми розробками, що забезпечується максимальною оптимізацією компільованого коду та наявністю в білді лише найнеобхідніших бібліотек, що значно зменшує розмір збірки на виході. Окрім того, фреймворк підтримує популярну архітектуру побудови веб додатків MVC (Model-View-Controller – модель-відображення-контролер), що також спрощує розуміння принципів його роботи для розробників, що попередньо не ознайомлювалися з кодом. [23] Таким чином, ASP.NET Core визначено як оптимальну платформу для розробки сервісу.

Для того, щоб розробити сервіс із використанням обраної технології, потрібні відповідні інструменти розробки. Компанія

Microsoft надає власний засіб під назвою Visual Studio. [24] Це середовище розробки, що є основним для .NET і рекомендованим для всіх. Вибір цього IDE обумовлено, по-перше, тим, що воно є розширюваним на основі робочого навантаження розробника, тобто він може встановити лише ті розширення, які потрібні в конкретний момент часу. Воно також має цілком функціональні засоби для написання коду, об'єднані в одне загальне середовище. Також Visual Studio підтримує значну кількість інших технологій та мов програмування (наприклад, JavaScript), які можна використовувати для доповнення проєктів додатковими модулями. Крім того, середовище розробки надає можливості для кросплатформової розробки, інтеграції систем контролю версій, а віднедавна ще також інструменти з застосуванням штучного інтелекту для зручного доповнення програмного коду. Visual Studio поставляється у трьох випусках – Community (безкоштовний, для некомерційного використання, для опенсорс або індивідуальних розробників), Professional (для поодиноких розробників та малих команд) і Enterprise (для бізнесу різноманітного масштабу). Для розробки сервісу інтеграції Zoom можливо використати будь-який з випусків, але для зменшення витрат [25]

У якості альтернативи стандартним засобам Microsoft є незалежний продукт виробництва компанії JetBrains під назвою Rider. [] Це також середовище розробки .NET додатків, яке має весь необхідний функціонал для роботи. З переваг цього засобу можна виділити, по-перше, його кросплатформовість, особливо виділяється його адаптація до платформ, не пов'язаних із Windows, наприклад, MacOS. Також до додатку інтегровано зручні інструменти з інтелектуального аналізу коду, що спрощує розробникові написання коду. Важливо, що Rider має в наявності всі потрібні модулі для .NET розробки, такі як .NET CLI, MSBuild, виконувач тестів і відлагоджувач. Важливою перевагою Rider є його продуктивність і стабільність, менше використання ресурсів

комп’ютера під час роботи незалежно від розмірів проєкту, з яким іде робота. Також віднедавна продукт має безкоштовну ліцензію для некомерційного використання, що розширює коло його потенційних користувачів. [27]

Вибір інструменту для розробки залежить від багатьох факторів, у тому числі від особистих поглядів розробника. У конкретному випадку з сервісом інтеграції Zoom цей вибір не має особливого значення. Оскільки розробка відбувається в середовищі операційної системи Windows, то цілком доречно використовувати Visual Studio.

Варто також зазначити, що розміщення готового сервісу відбуватиметься за допомогою засобів Docker у хмарному середовищі, аналогічно до інших складових KSU24. Отже, для розробки початкової конфігурації контейнера необхідно мати локальне середовище для його тестового розгортання й запуску. Для цього використовується Windows-версія Docker Desktop. Docker Desktop — це програма, яка оперативно встановлюється для середовища Mac, Linux або Windows, що дозволяє створювати, ділитися та запускати контейнерні програми та мікросервіси.

Він забезпечує простий GUI (графічний інтерфейс користувача), який дозволяє керувати контейнерами, програмами та зображеннями безпосередньо з робочої машини. Docker Desktop скорочує час, витрачений на складні налаштування, тож ви можете зосередитися на написанні коду. Він піклується про відображення портів, проблеми з файловою системою та інші параметри за замовчуванням і регулярно оновлюється виправленнями помилок і оновленнями безпеки. [28] Він є безкоштовним для некомерційного застосування, отже, цілком відповідає поставленим умовам. Перед установкою важливо перевірити сумісність середовища Docker з комп’ютером розробника. Системні вимоги до Docker наступні: WSL версії 1.1.3.0 або новіше; Windows 11 64-bit: Home або Pro версії 22H2 або вище, або Enterprise або Education

версії 22H2 або вище; Windows 10 64-bit: Мінімальна вимога – це Home або Pro 22H2 (build 19045) або вище, або Enterprise або Education 22H2 (build 19045) або вище. Необхідно також активувати WSL 2 функцію у Windows. Для успішного запуску WSL 2 у Windows 10 або Windows 11 необхідні наступні апаратні вимоги: 64-розрядний процесор із трансляцією адрес другого рівня (SLAT), 4 ГБ системної оперативної пам’яті. Потрібно також увімкнути апаратну віртуалізацію в BIOS. [29]

РОЗДІЛ 3
Реалізація сервісу ConferenceAPI у вигляді прикладного програмного інтерфейсу

3.1. Початкові налаштування для корпоративного акаунту Zoom

Перед тим, як починати розробку сервісу, необхідно підготувати університетський корпоративний акаунт Zoom для роботи. Для цього, отримавши електронну адресу та пароль від облікового запису, потрібно авторизуватися в Zoom у браузері на головній сторінці. [30] Після цього необхідно перейти на сторінку Zoom App Marketplace, а саме в розділ створених додатків [31]. Він має такий вигляд, як зображено на рис. 3.1.

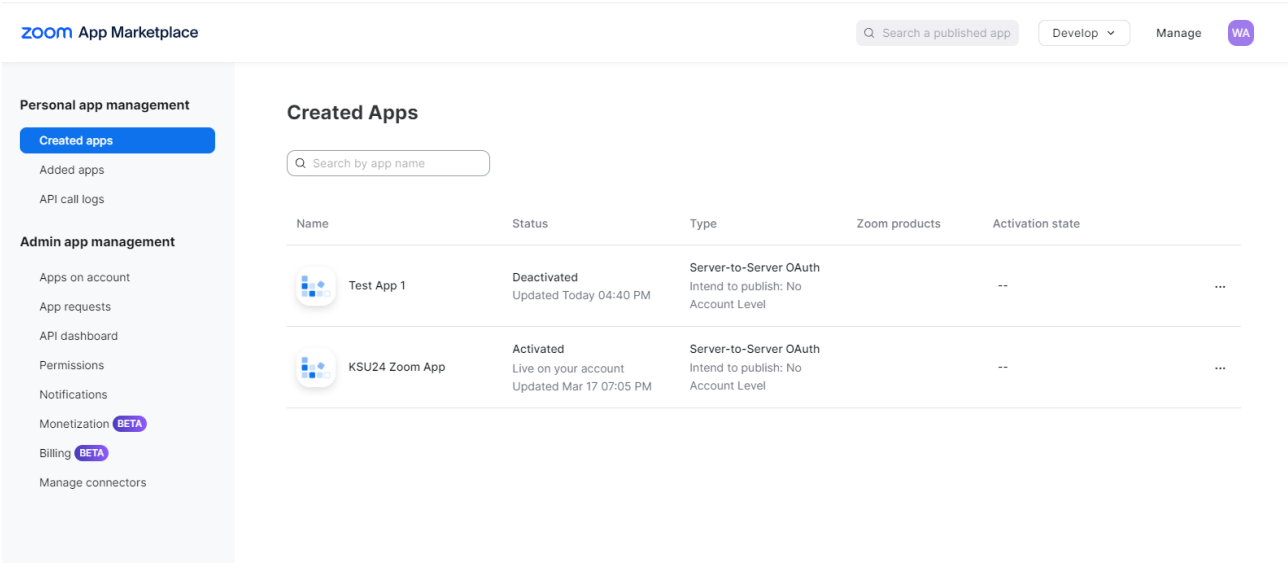

Рисунок 3.1. Вікно Zoom додатків поточного акаунту

Для роботи з Zoom API необхідно створювати Zoom App у Zoom App Marketplace, оскільки це дозволяє встановити офіційну інтеграцію, яка забезпечує доступ до всіх функцій і ресурсів платформи через безпечні інтерфейси. Zoom App діє в якості мосту між зовнішньою системою та Zoom, надаючи додатковий рівень захисту даних через Oauth авторизацію та контроль прав доступу.

OAuth, або Open Authorization, є протоколом авторизації, який дозволяє одному вебдодатку отримувати доступ до ресурсів іншого

додатка від імені користувача без необхідності надання його облікових даних. Іншими словами, OAuth забезпечує можливість входу на сайт або в додаток через обліковий запис іншої платформи, без передачі логіна та пароля. Основна мета протоколу – підвищити безпеку та зручність для користувачів і додатків. Замість того щоб ділитися повними обліковими даними з кожним сервісом, користувачі можуть надати доступ лише до вибраних частин інформації, обмежуючи його до мінімально необхідного для функціонування додатків. Цей підхід суттєво полегшив роботу мільйонам користувачів і розробників, створюючи безпечний і зручний спосіб обміну даними у цифровому середовищі. [32]

Отже, Marketplace гарантує, що всі створені додатки проходять базову перевірку безпеки, а також інтегруються через уніфіковані стандарти. Це критично важливо, адже Zoom надає доступ до конфіденційної інформації, такої як відеозаписи, календарі зустрічей, управління учасниками тощо. Без цього механізму користувачі або адміністратори могли б зіткнутися з неавторизованим доступом, що може призвести до витоку даних. Крім того, процес створення Zoom App у Marketplace дозволяє точніше налаштувати права доступу до ресурсів Zoom API, що забезпечує мінімальний доступ та підвищену безпеку.

Отже, для створення такого додатку потрібно натиснути Develop – Build App. У вікні, що з'явилося, треба обрати Server to Server OAuth App, оскільки сервіс не матиме власного інтерфейсу користувача й не буде доступним напряму кінцевому користувачеві. Після обрання потрібного перемикача і введення унікального імені додатку відкриється форма, в якій буде необхідно задати налаштування майбутнього додатку. На першій сторінці буде необхідно скопіювати всі три параметри, що пропонуються – Account ID, Client ID та Client Secret. Ці дані є конфіденційними, тому вкрай важливо не допустити можливості, щоб сторонні особи могли отримати таку інформацію. Приклад такої сторінки зображено на рис. 3.2.

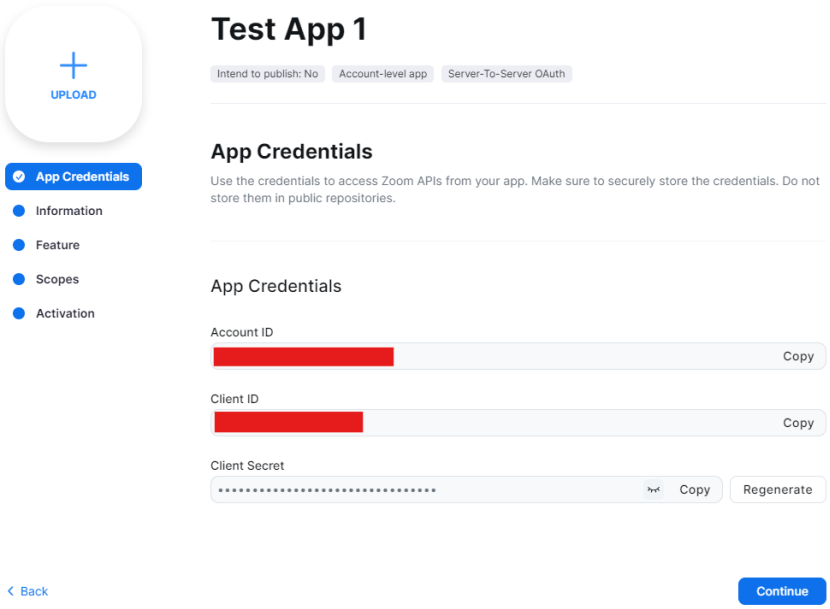

Рисунок 3.2. Сторінка Credentials (чутливі дані приховано з міркувань безпеки)

На вкладці Information необхідно заповнити всі поля, оскільки згодом це буде потрібно для активації додатку, а за потреби – для його публікації в маркетплейсі. Вміст полів не має значення, проте для інформативності варто вказати актуальну інформацію про себе як про розробника та про організацію (якщо вона є).

На вкладці Feature зберігаються секретний та верифікаційний токени. Їх можна зберегти на випадок непередбачуваних обставин, де вони могли б знадобитися, але необхідно пам’ятати, що секретний токен має обмежений термін придатності, тому після спливання терміну він має бути поновлений (тобто перестворений).

Наступна вкладка, Scopes, є одною з найважливіших. На ній для додатку задаються так звані scopes (найближчий за змістом переклад – межі). Це особливі параметри, що дозволяють або забороняють додатку (а отже, і Zoom API) доступи до певних дій або даних. Наприклад, це можуть бути перегляд інформації про користувачів, зустрічі, вебінари, акаунти. Окрім того, до деяких параметрів включений як доступ на читання/перегляд, так і на зміну параметру (параметрів). Для того, щоб

додаток функціонував належним чином відповідно до вимог, необхідно додати наступні параметри:

- View account info /account:read:admin
- View and manage account info /account:write:admin
- View all users' contacts /contact:read:admin
- View all users' usage statistics of CRC /dashboard_crc:read:admin
- View overview of usage statistics for Meetings and Zoom Rooms /dashboard_home:read:admin
- View all users' usage statistics of team chat messages and message types /dashboard_im:read:admin
- View all users' meetings information on Dashboard /dashboard_meetings:read:admin
- View all users' webinar information on Dashboard /dashboard_webinars:read:admin
- View all users' Zoom Room usage statistics and information /dashboard_zr:read:admin
- Manage device /device:read:admin
- View and manage device /device:write:admin
- View and manage sub account's H.323 devices /h323:master
- View all users' H.323 devices /h323:read:admin
- View and manage all users' H.323 devices /h323:write:admin
- View information barriers /information_barriers:read:admin
- View and manage information barriers /information_barriers:write:admin
- View all user meetings /meeting:read:admin
- View and manage all user meetings /meeting:write:admin
- View all user recordings /recording:read:admin
- View and manage all user recordings /recording:write:admin

- View report data /report:read:admin
- View your team chat history report /report_chat:read:admin
- View all user roles /role:read:admin
- View and manage all user roles /role:write:admin
- View all users' tracking fields /tracking_fields:read:adminDelete
- View and manage all users' tracking fields /tracking_fields:write:admin
- View all user information /user:read:admin
- View users information and manage users /user:write:admin
- View all user Webinars /webinar:read:admin
- View and manage all user Webinars /webinar:write:admin
- Get account’s whiteboard(s) /whiteboard:read:admin
- View event's Access Link /zoom_events_access_links:read:admin
- View attendee actions performed by host for an event/session /zoom_events_attendee_actions:read:admin
- View and manage attendee actions performed by host for an event/session /zoom_events_attendee_actions:write:admin
- View all events information /zoom_events_basic:read:admin
- View and manage events information /zoom_events_basic:write:admin
- View Zoom events coeditor information /zoom_events_coeditors:read:admin
- View exhibitor information for an event /zoom_events_exhibitors:read:admin
- View Zoom event recordings information /zoom_events_recordings:read:admin
- View event registrants information /zoom_events_registrants:read:admin
- View event session information /zoom_events_sessions:read:admin
- View and manage event session information /zoom_events_sessions:write:admin

Після того, як задані всі потрібні scopes, можна переходити на останню вкладку Activation. Якщо додаток не активовано, він не дозволить жодному застосунку, що посилається на нього, надсилати запити до Zoom API. Якщо всі дані було вказано вірно, то після натискання Activate на сторінці відображатиметься повідомлення “Your app is activated on the account”, а кнопка активації заміниться на Deactivate. Якщо якась частина обов’язкових даних не заповнена, то замість кнопки активації буде напис, що міститиме всі помилки, що необхідно виправити. Для прикладу, напис Select at least one scopes означає, що в додатку має бути обраний як мінімум один параметр.

3.2. Розробка API та його конфігурація

Перш за все, необхідно клонувати створений заздалегідь порожній репозиторій на свій пристрій. Знайшовши зручну папку для розміщення проєкту, необхідно в ній відкрити або консольне вікно Windows (CMD), або Windows PowerShell, а потім перейти до GitHub і, скопіювавши адресу репозиторію, виконати операцію клонування. Це виконується за допомогою наступної команди: git clone https://github.com/ksu-24/ksu24.zoom.git . Приклад успішного виконання можна бачити на рис. 3.3.

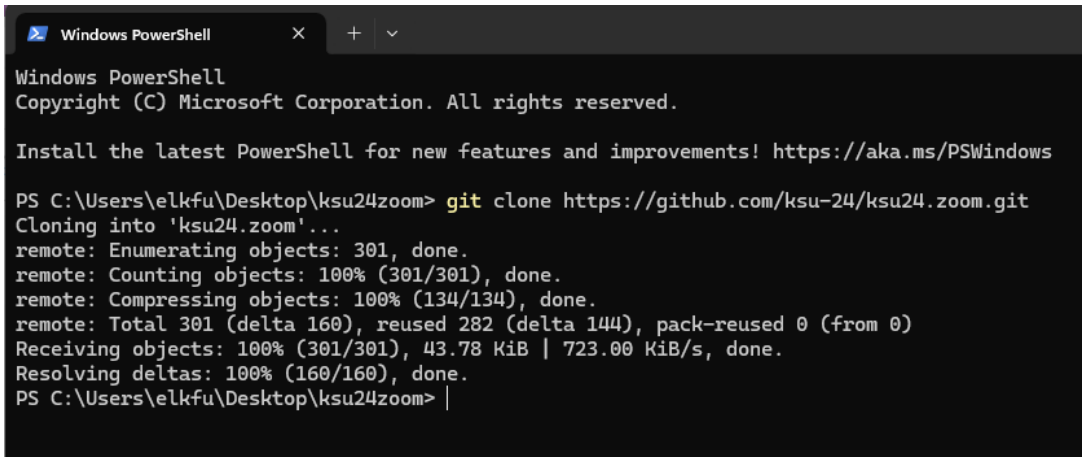

Рисунок 3.3. Приклад клонування створеного репозиторію

Після цього у клонованому репозиторії потрібно створити нове рішення (Visual Studio Solution, SLN). У головному вікні Visual Studio потрібно виконати наступні дії: File – New – Project, у відкритому вікні знайти шаблон ASP.NET Core Web API, а в наступному ввести наступні дані: Project name (ім’я проєкту) – ConferenceAPI.Web, Location (локація) – оберіть клоновану папку з репозиторієм, у списку Solution (рішення) залишити Create new solution (створити нове рішення – оскільки нам необхідно створити саме новий проєкт), обов’язково зняти прапорець Place solution and project in the same directory (розмістити рішення й проєкт в одній папці – зазвичай, вважається неприпустимою практикою в .NET розробці) і в полі Solution name (ім’я рішення) проконтролювати наявність правильного імені рішення – ConferenceAPI. У наступному вікні потрібно зазначити налаштування проєкту Web API. Вміст наступного вікна зображений на рис. 3.4.

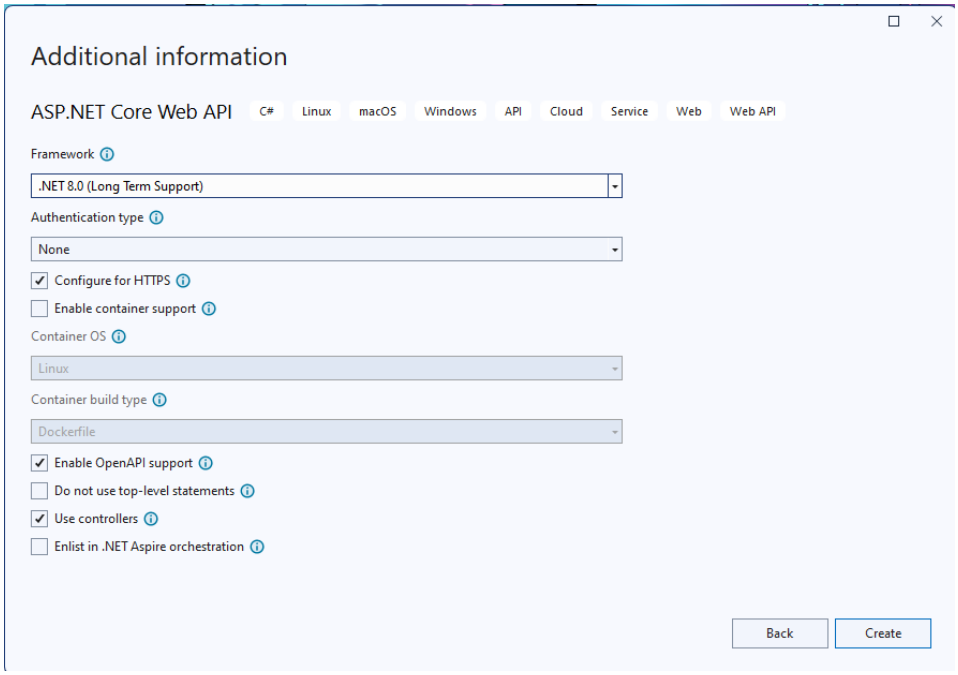

Рисунок 3.4. Вікно налаштувань проєкту ASP.NET Core Web API

Розберемо його основні параметри. У верхньому списку має бути зазначена найсвіжіша стабільна версія .NET станом на час написання роботи - .NET 8. Оскільки лише KSU24 взаємодіятиме з цим сервісом, тобто він використовуватиметься лише як внутрішній, то задання

способу автентифікації не є необхідним, що й зазначено в наступному списку. Рекомендується також задіяти налаштування HTTPS задля безпеки. Так як сервіс розгортатиметься в Docker, то наступний прапорець обов'язковий для обрання. У списку операційних систем має стояти Linux, а тип збирання контейнера – Dockerfile.

Важливо також позначити прапорець, що увімкне підтримку OpenAPI. OpenAPI – це опис прикладного програмного інтерфейсу для архітектури REST API, що дозволяє описати кінцеві точки (ендпоінти) інтерфейсу та операції (GET, POST тощо), вхідні й вихідні дані для кожної операції, методи автентифікації (якщо вона є) і додаткові дані на кшталт ліцензійних даних, умов використання, контактів, версіонування тощо. У загальному, OpenAPI допомагає візуалізувати документацію до програмного інтерфейсу, таким чином спрощуючи розуміння структури додатку та його можливостей. [33]

Після цього потрібно переконатися, що прапорець Do not use top-level statements (не використовувати високорівневі вирази) не обраний, так як його обрання призведе до створення проєкту в старому шаблоні ASP.NET Core Web API – з відокремленими класами Program та Startup. Новий шаблон ASP.NET Core Web API, що використовує високорівневі вирази, надає розробникам більш легкий і зрозумілий спосіб створення API. Завдяки спрощеному синтаксису, цей підхід зменшує кількість коду, необхідного для виконання базових завдань, що в свою чергу знижує ймовірність помилок і полегшує підтримку коду. Високорівневі вирази дозволяють реалізовувати функції через зрозумілі й компактні методи, не потребуючи традиційних класів контролерів, що часто є зайвими при розробці невеликих сервісів або мікросервісів. Цей підхід також поліпшує продуктивність, оскільки дозволяє зосередитися на ключовій бізнес-логіці без витрат на зайву структуру, яка може ускладнювати систему. У результаті, розробники можуть швидше отримати функціональний прототип або готовий сервіс, що особливо

цінно для стартапів і команд, які працюють за гнучкими методологіями розробки. Крім того, високорівневі вирази дозволяють інтегрувати сучасні принципи DI (Dependency Injection – ін'єкція залежностей), авторизації та логування без надмірного коду, роблячи додаток більш гнучким і легким у тестуванні. Завдяки цьому API стає більш стійким і легко масштабованим, а також краще підходить для побудови хмарних сервісів. Таким чином, найкращим варіантом є використання високорівневих виразів.

Наступний прапорець, Use controllers (використовувати контролери), також має бути обраним. У нових версіях .NET було додано новий спосіб визначення ендпоінтів, окрім класичних контролерів – це Minimal API. Мінімальний API — це новий підхід до створення API без багатьох складних структур MVC. Мінімальний API містить лише основні компоненти, необхідні для створення HTTP API, якими є лише csproj і файл Program.cs. Вони не замінюють фреймворк MVC, а є лише альтернативним способом побудови REST API, який зробив його надзвичайно простим із мінімальними залежностями. [] Але зважаючи на те, що Minimal API станом на зараз не набув достатнього поширення в .NET суспільстві, для розробки сервісу буде більш оптимальним використання контролерів у класах.

Останній прапорець, що дозволяє огорнути рішення в структуру .NET Aspire, має бути вимкненим. Ця технологія забезпечує замкнену структуру рішення, що має спільну вхідну точку й може об'єднувати всі проєкти в єдиний цілісний додаток. .NET Aspire розроблено для легкого створення видимих і надійних хмарних додатків, одночасно покращуючи продуктивність для розробників, які створюють ці додатки швидкими темпами. .NET Aspire робить це в п'яти основних областях: інформаційна панель .NET Aspire, оркестрування додатків, виявлення служб, компонентів та їхнє розгортання. Кожну з цих частин можна адаптувати окремо до існуючої програми або прийняти разом, якщо

створюється новий проект. [35] Оскільки сервіс включатиме в себе лише один веб проєкт, то жодної необхідності у використанні .NET Aspire немає. Після цього, коли всі параметри задано, потрібно створити рішення, натиснувши кнопку Create.

Отже, рішення створено. У стандартному шаблоні ASP.NET Core Web API присутній демонстративний код, що імітує показ прогнозу погоди, що допомагає ознайомитися з принципом роботи API. Для подальшої розробки потрібно видалити цей код, залишивши лише базові елементи програмного інтерфейсу.

Для початку потрібно завантажити й установити необхідні бібліотеки для функціоналу сервісу. Для цього виконаємо команду Tools – NuGet Package Manager – Manage NuGet Packages For Solution і на першій вкладці Browse знайдемо й встановимо потрібні бібліотеки (або, як заведено казати в .NET, пакети). NuGet є офіційним менеджером пакетів для .NET і є невід'ємною частиною екосистеми .NET. Це дозволяє розробникам ділитися багаторазовим кодом, керувати залежностями та без зусиль інтегрувати бібліотеки сторонніх розробників у власні програми. Незалежно від того, чи відбувається розробка над простою консольною програмою чи великомасштабною веб-програмою ASP.NET Core, NuGet відіграє ключову роль у сучасних робочих процесах розробки. [36]

Ось повний список потрібних пакетів (деякі з них будуть встановленими за замовчуванням через налаштування базового шаблону ASP.NET Core Web API):

- AutoMapper – це об'єктно-об'єктний маппер. Відображення об'єкт-об'єкт працює шляхом перетворення вхідного об'єкта одного типу на вихідний об'єкт іншого типу. Що робить AutoMapper таким, що привертає увагу, так це те, що він надає певні конвенції, щоб позбавити від зайвої роботи з'ясування того, як відобразити тип A у

тип B. Поки тип B відповідає встановленим угодам AutoMapper, для відображення двох типів майже не потрібна ручна конфігурація. [37]

- FluentValidation – це безкоштовна бібліотека .NET, яка допомагає швидко й легко впроваджувати правила перевірки. За допомогою неї розробник створює окремий загальнодоступний клас для своїх правил, заохочуючи зручність обслуговування та запобігаючи проблемам із розповсюдженням коду. [38]
- FluentValidation.DependencyInjectionExtensions
- Microsoft.AspNetCore.WebUtilities
- Microsoft.Extensions.Caching.Abstractions
- Microsoft.Extensions.Http
- Microsoft.VisualStudio.Azure.Containers.Tools.Targets
- Serilog.AspNetCore – це діагностична бібліотека логування для програм .NET. Його легко налаштувати, він має прозорий і зрозумілий API і працює на всіх останніх платформах .NET. Незважаючи на те, що це може бути корисним також у найпростіших програмах, підтримка Serilog для структурованого логування набуває більшого значення при інструментуванні складних, розподілених і асинхронних програм і систем. [39]
- Serilog.Enrichers.Environment
- Serilog.Sinks.Console
- Swashbuckle.AspNetCore

Після того, як усі потрібні пакети встановлені, перейдемо до головної точки входу – файл Program.cs, що під час збирання проєкту трансформується в клас. Розпочнімо з того, що задамо конфігурацію для сервісу (див. рис. 3.5).

```
IConfiguration configuration = new ConfigurationBuilder()
    .AddJsonFile("appsettings.json", optional: false, reloadOnChange: true)
    .AddJsonFile($"appsettings.{Environment.GetEnvironmentVariable("ASPNETCORE_ENVIRONMENT")}.json", optional: false, reloadOnChange: true)
    .Build();
```

Рис. 3.5. Конфігурація додатку

Конфігурація залежатиме від середовища, в якому розгорнуто додаток. У головному файлі appsettings.json розташовуються адреси Zoom API та ключі для зв’язку з акаунтом Zoom університету. У файлах appsettings.Development.json і appsettings.Production.json знаходяться налаштування логів: для Dev задано рівень логування Debug, для Prod – Information. Файл appsettings.json в ASP.NET Core — це файл конфігурації у форматі JSON, який використовується для зберігання даних конфігурації налаштувань програми. Він забезпечує структурований спосіб визначення налаштувань конфігурації, таких як рядки з’єднання з базою даних, параметри логування, ключі API та інші глобальні параметри рівня програми, необхідні програмі, яку потрібно змінити, не перекомпілюючи її щоразу. [40]

Наступним кроком буде конфігурація логування. За допомогою бібліотеки Serilog задамо наступні налаштування: зчитувати налаштування з конфігураційних файлів; записувати логи до консолі; додавати в логи контекст, тобто назву сервісу, з якого він записується; додавати назву середовища в кожен лог.

Після цього потрібно підключити логгер до програми, після чого додати сервіси та HTTP клієнти. Задля двох останніх дій створюється папка Extensions, у ній статичний клас ServiceCollectionExtensions, а в ньому відповідні методи ConfigureServices та AddHttpClients. У першому налаштовуються базові параметри програми, наприклад, підключаються контролери, а також кешування. Також додаються обробник помилок, маппер, валідатори й сервіси взаємодії з Zoom. У другому методі налаштовуються два HTTP клієнти, один з яких використовуватиметься для отримання токену, а другий буде взаємодіяти з Zoom API і працювати з конференціями. Варто відзначити, що для Zoom Auth клієнту задається незвичайний спосіб авторизації через Base64 рядок, в якому шифруються секретні ключі. З інших базових налаштувань також

варто відзначити визначення політик CORS, які дозволяють тим чи іншим клієнтам вільно звертатися до сервісу.

Для подальшої роботи до рішення потрібно додати ще один проєкт – бібліотеку класів під назвою ConferenceAPI.Core. Це робиться для поділу логіки між окремими прошарками додатку. Створена бібліотека класів матиме функцію сервісного прошарку.

В основному проєкті в папці Controllers потрібно створити контролер для роботи з конференціями – ZoomApiController. У ньому, відповідно до вимог, будуть наступні ендпоінти:

- Створити зустріч – POST, повертатиме 201 Created або 400 BadRequest
- Редагувати зустріч – PUT, 204 NoContent або 400 BadRequest
- Отримати зустріч – GET, 200 OK або 400 BadRequest
- Видалити зустріч – DELETE, 204 NoContent або 400 BadRequest
- Отримати статистику за ідентифікатором зустрічі – GET, 200 OK або 400 BadRequest

Кожен ендпоінт звертається до сервісу IZoomApiService до відповідного методу. Інтерфейс сервісу, а також сам ZoomApiService розміщені в бібліотеці класів ConferenceAPI.Core. Кожен з методів має приблизно однакову схему взаємодії з Zoom: спочатку відбувається звернення по токен до окремого сервісу IZoomAuthService, у випадку невдачі повертається неуспішний результат, що повідомляє про помилку авторизації. Після отримання токену відбувається запит на Zoom API з його використанням, зовнішній ресурс повертає результат, а на його основі сервіс формує власний результат і відправляє до контролера, який, у свою чергу, робить відповідь HTTP кодом. Задля полегшення розуміння процесів під час виконання запитів усі етапи максимально логуються за допомогою Serilog і його методів LogInformation, LogWarning або LogError.

Зазначимо також декілька окремих аспектів певних методів. Під час створення нової зустрічі пароль для неї генерується випадковим чином у методі GenerateRandomPassword. Для методів CreateZoomMeetingAsync та EditZoomMeetingInfoAsync також відбувається на рівні контролера мапінг (копіювання параметрів) за допомогою AutoMapper.

Перейдемо до сервісу IZoomAuthService. Це вкрай важливий сервіс, що містить логіку отримання авторизаційного токену для Zoom API. Він взаємодіє з окремим зовнішнім Zoom сервісом, що за наданими параметрами генерує токен. Для цього раніше був налаштований HTTP клієнт, а також додані 2 параметри grant_type та account_id. Важливим аспектом є те, що отриманий токен одразу відправляється до кешу в пам’яті, і якщо запит буде надіслано вдруге, то токен вже буде взято з кешу, що значно прискорить роботу сервісу. Час, протягом якого токен є валідним, визначений розробниками Zoom і складає 1 годину.

Тепер, коли визначено основну логіку додатку, необхідно додати конфігурацію Docker. Вона додається до API проєкту у вигляді пари файлів docker-compose.yml (див. рис. 3.6) та docker-compose.override.yml (див. рис. 3.7), а також Dockerfile і декількох додаткових допоміжних файлів.

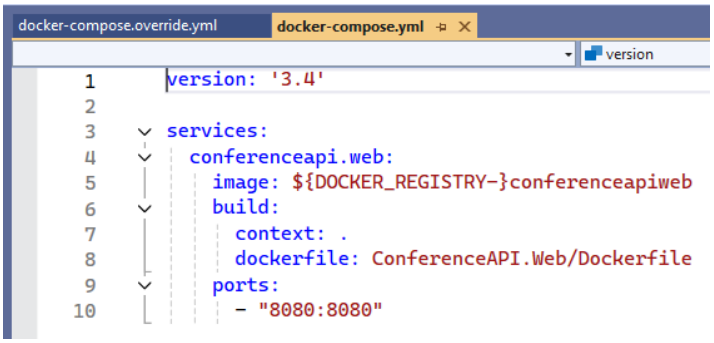

Рисунок 3.6. docker-compose файл

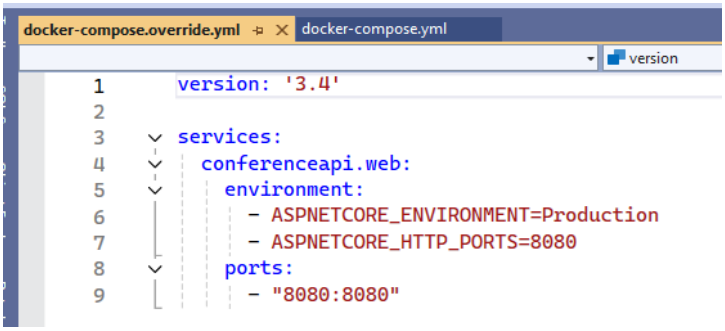

Рисунок 3.7. docker-compose override файл

Вміст Dockerfile генерується Visual Studio автоматично і майже не потребує допрацювання, а файли docker-compose створюються порожніми, тому потребують ручного заповнення, як на рисунках.

Після завершення розробки, а також поверхневого код рефакторингу, необхідно зібрати рішення відповідною командою середовища розробки. І лише у випадку, коли збірка була успішною, переходити до тестування.

3.3. Тестування

Для тестування сервісу потрібно використати стандартний засіб під назвою Postman. Postman – це багатофункціональний інструмент для тестування, створення та автоматизації API-запитів, який дозволяє спростити процес розробки та інтеграції. З його допомогою розробники можуть створювати HTTP-запити різних типів, таких як GET, POST, PUT, DELETE, і надсилати їх на сервер для перевірки взаємодії клієнт-сервер, аналізу відповідей, часу виконання та інших параметрів. У ньому можна працювати з запитами у зручному графічному інтерфейсі, де користувачам доступні детальні налаштування запитів, тіла запитів, параметрів, заголовків та інше. Відповідь сервера з’являється миттєво і показує код відповіді, статус та саме тіло, що допомагає розробникам ідентифікувати можливі помилки.

Особливістю Postman є можливість зберігати запити у вигляді колекцій, що робить його корисним для роботи з багатоступеневими

сценаріями API-тестування, а також для документації та подальшого використання запитів. Інтеграція з тестами дозволяє налаштувати перевірки результатів та поведінки серверів автоматично, використовуючи фреймворк тестування на основі JavaScript. Це робить Postman ефективним інструментом не тільки для тестування, а й для навчання, відлагодження та побудови різних сценаріїв використання API.

Перейдемо до тестування сервісу. Для початку спробуємо додати нову зустріч з наступними параметрами: {

"topic": "Test Meeting",
"start_time": "2024-10-29T09:00:00",
"duration": 60,
"type": 2,
"settings": {
"waiting_room": **true**,
"join_before_host": **false**,
"mute_upon_entry": **true**,
"meeting_invitees": []
},
"email": "mykhailo.novikov@university.ks.ua"
}

Після виконання запиту отримуємо наступний результат (див. рис. 3.8).

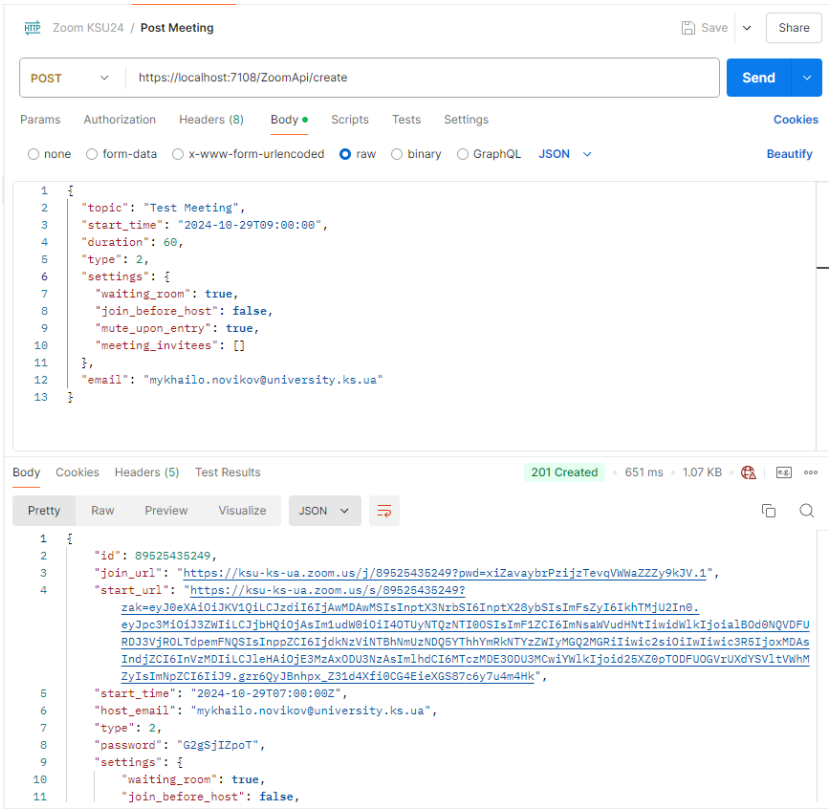

Рисунок 3.8. Створення зустрічі

Отримано успішний результат зі створеною зустріччю, наявні посилання для старту та долучення, пароль, ідентифікатор, дата початку та інші корисні дані. Якщо валідація з одного або декількох полів не пройшла успішно, то повертатиметься результат 400 Bad Request з описом помилок, які потрібно виправити для успішного створення.

Перевіримо наступний ендпоінт, пов’язаний з отриманням інформації про зустріч через ID. Його результат можна побачити на рис. 3.9.

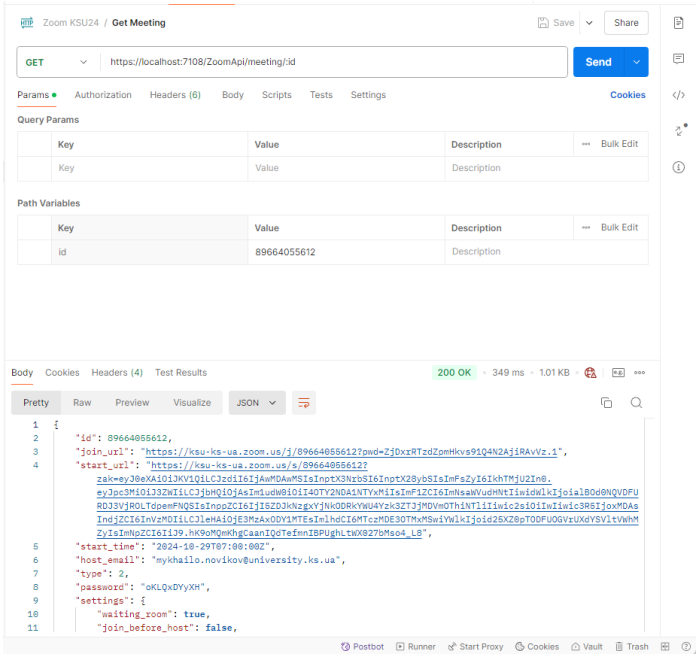

Рисунок 3.9. Отримання інформації про зустріч по ID

За отриманим результатом конференція отримується належним чином, усі необхідні дані відображаються на екрані, а саме два посилання для старту й доєднання, а також дата початку зустрічі й її тип. Якщо зустріч, ідентифікатор якої введено в запит, не існує, то помилка виглядатиме наступним чином: {"code":3001,"message":"Meeting does not exist: 8966405561.","errors":null} , а статус код, що повертається API, відповідає 400 Bad Request. Як видно з тіла об’єкту, кожна помилка від Zoom API має свій унікальний код. Ці коди детально описані в документації Zoom, посилання на яку присутні в Readme репозиторію.

Наступним до перевірки буде метод PUT, що змінює дані конференції. До змін дані мають наступний вигляд:

{"id":89664055612,"join_url":"https://ksu-ks-ua.zoom.us/j/89664055612?pwd=ZjDxrRTzdZpmHkvs91Q4N2AjiRAvVz.1","start_url":"https://ksu-ks-ua.zoom.us/s/89664055612?zak=eyJ0eXAiOiJKV1QiLCJzdiI6IjAwMDAwMSIsInptX3NrbSI6InptX28ybSIsImFsZyI6IkhTMjU2In0.eyJpc3MiOiJ3ZWIiLCJjbHQiOjAsIm1udW0iOiI4OTY2NDA1NTYxMiIsImF1ZCI6ImNsaWVudHNtIiwidWlkIjoialBOd0NQVDFURDJ3VjROLTdpemFNQSIsInppZCI6IjIISDJkNzgxYjNkODRkYWU4Yzk3ZTJjMDVmOThiNTliIiwic2siOiIwIiwic3R5IjoxMDAsImdjZCI6InVzMDIiLCJleHAiOjE3MzAxODY1MTEsImlhdCI6MTczMDE3OTMxMSwiYWlkIjoid2SXZ0pTODFUOGVrUXdYSVltVWhMZyIsImNpZCI6II9.hK9oMQmKhgCaanIQdTefmnI8PUghLtWX027bMso4_L8","start_time":"2024-10-29T07:00:00Z","host_email":"mykhailo.novikov@university.ks.ua","type":2,"password":"oKLQxDYyXH","settings":{"waiting_room":true,"join_before_host":false,

mI1MWM2NTU0MjhjZjQ1MGFhZWIxYTgwNTVlZDc4ZjNjIiwic2siOiIwIiwic3R5IjoxMDAsIndjZCI6InVzMDIiLCJleHAiOjE3MzAxODczMTAsImlhdCI6MTczMDE4MDExMCwiYWlkIjoid25XZ0pTODFUOGVrUXdYSVltVWhMZyIsImNpZCI6IiJ9.Jq1vNSZRj4lfgRu3zVL7UgoiSTxXVTPUbCsXdpGP-58","start_time":"2024-10-29T07:00:00Z","host_email":"mykhailo.novikov@university.ks.ua","type":2,"password":"oKLQxDYyXH","settings":{"waiting_room":true,"join_before_host":false,"mute_upon_entry":true,"meeting_authentication":false,"authentication_domains":null,"meeting_invitees":[]}}.

Спробуємо відредагувати наступним чином: змінимо рік на 2025, а meeting_authentication – на true. Результат запиту на зміну буде 204 No Content, а об’єкт конференції:

"start_time":"2025-10-29T07:00:00Z","host_email":"mykhailo.novikov@university.ks.ua","type":2,"password":"oKLQxDYyXH","settings":{"waiting_room":true,"join_before_host":false,"mute_upon_entry":true**,"meeting_authentication":true**,"authentication_domains":"","meeting_invitees":[].

Як видно, зміна відбулася успішно. Якщо ідентифікатор конференції невірний, то отримаємо помилку:

{"code":3001,"message":"Meeting does not exist: 89664055612.","errors":null}

Наступний метод, який ми перевіряємо – це видалення зустрічі. Його результат зображено на рис. 3.10.

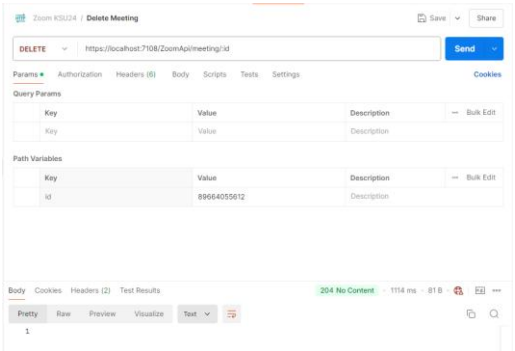

Рисунок 3.10. Видалення зустрічі

Результат – 204 No Content, отже, видалення було успішним. Якщо надати в запит невірний ID, то помилка виглядатиме наступним чином:

{"code":3001,"message":"Meeting does not exist: 89664055612.","errors":null}

Останній запит, який потрібно перевірити – це запит на статистику. Головна умова – зустріч має бути завершеною. В інакшому випадку отримаємо помилку наступного виду:

{"code":3001,"message":"Meeting does not exist: 82081383418.","errors":null}

Коли зустріч завершено, то результат матиме вигляд, як на рис. 3.11.


```
{
    "participants": [
        {
            "email": "mykhailo.novikov@university.ks.ua",
            "join_time": "2024-10-29T05:52:54Z",
            "leave_time": "2024-10-29T05:54:41Z",
            "leave_reason": "Михайло Новіков left the meeting.<br>Reason: Host ended the meeting.",
            "role": "host"
        },
        {
            "email": null,
            "join_time": "2024-10-29T05:53:19Z",
            "leave_time": "2024-10-29T05:53:26Z",
            "leave_reason": "Михайло Миколайович Новіков left the meeting.<br>Reason: Leave waiting room.",
            "role": "attendee"
        },
        {
            "email": null,
            "join_time": "2024-10-29T05:53:26Z",
            "leave_time": "2024-10-29T05:54:34Z",
            "leave_reason": "Михайло Миколайович Новіков left the meeting.<br>Reason: left the meeting.",
            "role": "attendee"
        }
    ],
    "total_records": 3
}
```


Рисунок 3.11. Статистика по учасниках конференції

З особливостей цього ендпоінту відзначимо, що для кожного учасника відображається вся активність, що відбувалася протягом зустрічі, особливо долучення та вихід з неї – час та причина. В кожного також відображено роль і електронну адресу. У деяких рядках може бути null, оскільки ці адреси не додані до університетського облікового запису Zoom, від імені якого й виступає сервіс.

Таким чином, тестування сервісу ConferenceAPI завершено, основні можливості додатку перевірені, сервіс працює належним чином. У подальшому додаток буде допрацьовуватися відповідно до нових вимог,

а також виконуватиметься підтримка у випадках виникнення критичних помилок або важливих змін, що можуть призвести до неочікуваних результатів.

ВИСНОВКИ

Розроблений сервіс ConferenceAPI буде інтегровано в систему керування процесами університету KSU24 і стане в нагоді спільноті Херсонського державного університету, таким чином, викладачі й здобувачі вищої освіти матимуть можливість покращити й оптимізувати процес дистанційного навчання. Це забезпечуватиметься додатковим функціоналом, що дозволить викладачам створювати й розпочинати заняття зі сторінки академічного журналу, а після завершення конференції – отримати детальну статистику з даними про те, хто був присутнім, протягом якого часу здобувачі знаходилися на парі, а також чому той чи інший здобувач залишив конференцію.

У результаті дослідження були проаналізовані аналогічні рішення в популярних системах управління навчанням, таких як Moodle та EdisonOS, порівняні особливості й можливості систем, а також визначено наявність та зручність інтеграції фуункціоналу роботи з конференціями Zoom у розглянутих LMS. Проведений аналіз показав, що універсальних та ідеальних систем дистанційного навчання не існує, тому розробка такої системи (з інтеграцією Zoom) є цілком доречною альтернативою відомим рішенням, тому що це дозволяє уникнути проблем, з якими стикалися користувачі інших платформ.

Наступним кроком було сформовано реєстр вимог (функціональних, технічних, системних) і побудовано UML діаграму послідовностей, на якій відображено всі процеси в додатку між собою та з акторами.

Для забезпечення взаємодії з Zoom API було сконфігуровано університетський обліковий запис Zoom, а на ньому було створено Zoom App, параметри якого були внесені в ConferenceAPI в розділ конфігурації.

Сам сервіс було розроблено й представлено у вигляді ASP.NET Core Web API з можливістю розгортання на серверах будь-якої потужності й з будь-якою архітектурою, а також за необхідності – у хмарному середовищі. Під час розробки й відлагодження було досліджено можливості таких технологій, як ASP.NET Core, REST API, Git, а також системи контейнеризації Docker і Docker Compose. Відповідно до поставлених вимог, сервіс налагоджений для створення, перегляду, редагування, видалення конференцій, а також для перегляду статистики по завершених зустрічах та їхніх учасниках.

Подальший розвиток сервісу включає в себе підтримку сервісу відповідно до можливих змін у Zoom API, розширення ендпоінтів для отримання й передавання більшої кількості параметрів зустрічей, упровадження централізованого логування тощо. Станом на зараз сервіс розгорнутий у хмарному середовищі Херсонського державного університету й успішно функціонує за призначенням.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Головна | KSU24. URL: https://ksu24.kspu.edu (дата звернення: 28.10.2024)

2. Академічні журнали | KSU24. URL: https://ksu24.kspu.edu/gradebooks (дата звернення: 28.10.2024)

3. Попов, С. А. Проєктування та розроблення сервісної архітектури управління бізнес-процесами університету. Сервіс «Деканат» = Design and development of service architecture of business process management of the university. Service "Dean's Office": кваліфікаційна робота на здобуття ступеня вищої освіти «магістр» / С. А. Попов ; наук. керівник доц. В. С. Круглик, к. фіз.-матем. наук, доц. В. А. Єрмолаєв ; Міністерство освіти і науки України ; Херсонський держ. ун-т, ф-т комп'ютерних наук, фізики та математики, Кафедра інформатики, програмної інженерії та економічної кібернетики. – Херсон : ХДУ, 2020. – 42 с.

4. Гунько, С. Е. Проєктування та розроблення сервісної архітектури управління бізнес-процесами університету. Електронний журнал = Design and development of service architecture of business process management of the university E-register: кваліфікаційна робота на здобуття ступеня вищої освіти «магістр» / С. Е. Гунько ; наук. керівник д.п.н., проф. О. В. Співаковський ; Міністерство освіти і науки України ; Херсонський держ. ун-т, Ф-т комп'ютерних наук, фізики та математики, Кафедра інформатики, програмної інженерії та економічної кібернетики. – Херсон : ХДУ, 2020. – 42 с.

5. Що таке LMS та як підібрати собі LMS-систему - Блог про email та інтернет-маркетінг. URL: https://sendpulse.ua/blog/learning-management-system (дата звернення: 28.10.2024)

6. На головну | Moodle.org. URL: https://moodle.org/?lang=uk (дата звернення: 28.10.2024)

7. Elearning Company Offering Custom Solutions to Empower Learners | Technomatix. URL: https://www.tmx-learning.com (дата звернення: 28.10.2024)

8. Moodle - Вікіпедія. URL: https://uk.wikipedia.org/wiki/Moodle (дата звернення: 28.10.2024)

9. Moodle Plugins directory: Zoom meeting | Moodle.org. URL: https://moodle.org/plugins/mod_zoom (дата звернення: 28.10.2024)

10. ncstate-delta/moodle-mod_zoom: Moodle plugin for Zoom meeting. URL: https://github.com/ncstate-delta/moodle-mod_zoom (дата звернення: 28.10.2024)

11. Rate limits - Zoom Developers. URL: https://developers.zoom.us/docs/api/rate-limits/ (дата звернення: 28.10.2024)

12. На головну | ХДМА ОНЛАЙН. URL: https://mdl.ksma.ks.ua (дата звернення: 28.10.2024)

13. KSU-online. URL: https://ksuonline.kspu.edu/ (дата звернення: 28.10.2024)

14. Zoom and LMS Integration: Benefits, How to do? URL: https://www.edisonos.com/blog/integrating-zoom-with-learning-management-systems-a-comprehensive-guide (дата звернення: 28.10.2024)

15. How to integrate Moodle and Zoom for online training? | e-Learning Blog | e-Learning by Human Science Co., Ltd. URL: https://hs-learning.jp/en/blog/240408/ (дата звернення: 28.10.2024)

16. Joining a Zoom meeting. URL: https://support.zoom.com/hc/en/article?id=zm_kb&sysparm_article=KB0060732 (дата звернення: 28.10.2024)

17. Діаграма послідовності – Вікіпедія. URL: https://uk.wikipedia.org/wiki/Діаграма_послідовності (дата звернення: 28.10.2024)

18. Що таке RESTful API? | DevZone. URL: https://devzone.org.ua/post/shcho-take-restful-api (дата звернення: 28.10.2024)

19. Django – Вікіпедія. URL: https://uk.wikipedia.org/wiki/Django (дата звернення: 28.10.2024)

20. Сенчишен, Д. О. Проєктування та розроблення сервісної архітектури управління бізнес-процесами університету. Сервіс «Відділ кадрів» = Design and development of service architecture of business process management of the university. Service "Human Resources Department": кваліфікаційна робота на здобуття ступеня вищої освіти «магістр» / Д. О. Сенчишен ; наук. керівник кандидат фізико-математичних наук, доктор педагогічних наук, професор О. В.Співаковський ; Міністерство освіти і науки України ; Херсонський держ. ун-т, ф-т комп'ютерних наук, фізики та математики, Кафедра інформатики, програмної інженерії та економічної кібернетики. – Херсон : ХДУ, 2020. – 50 с.

21. ksu-24/ksu24.zoom. URL: https://github.com/ksu-24/ksu24.zoom (дата звернення: 28.10.2024)

22. ASP.NET Core Web API Tutorials - Dot Net Tutorials. URL: https://dotnettutorials.net/course/asp-net-core-web-api-tutorials/ (дата звернення: 28.10.2024)

23. Why ASP.NET Core is the Best Framework for Web App Development? URL: https://radixweb.com/blog/8-reasons-asp-dot-net-core-is-best-framework (дата звернення: 28.10.2024)

24. Visual Studio: IDE and Code Editor for Software Developers and Teams. URL: https://visualstudio.microsoft.com (дата звернення: 28.10.2024)

25. What is the Visual Studio IDE? | Microsoft Learn. URL: https://learn.microsoft.com/en-us/visualstudio/get-started/visual-studio-ide?view=vs-2022 (дата звернення: 28.10.2024)

26. Rider: The Cross-Platform .NET IDE from JetBrains. URL: https://www.jetbrains.com/rider/ (дата звернення: 28.10.2024)

27. 5 Reasons Why I Choose JetBrains Rider Over Visual Studio | LinkedIn. URL: https://www.linkedin.com/pulse/5-reasons-why-i-choose-jetbrains-rider-over-visual-studio-műller/ (дата звернення: 28.10.2024)

28. Docker Desktop | Docker Docs. URL: https://docs.docker.com/desktop/ (дата звернення: 28.10.2024)

29. Windows | Docker Docs. URL: https://docs.docker.com/desktop/install/windows-install/ (дата звернення: 28.10.2024)

30. One platform to connect | Zoom. URL: https://us02web.zoom.us (дата звернення: 28.10.2024)

31. App Marketplace. URL: https://marketplace.zoom.us/user/build (дата звернення: 28.10.2024)

32. Oauth що це: визначення технології та основні принципи роботи. URL: https://foxminded.ua/oauth-shcho-tse/ (дата звернення: 28.10.2024)

33. What Is OpenAPI? | Swagger Docs. URL: https://swagger.io/docs/specification/v3_0/about/ (дата звернення: 28.10.2024)

34. Introduction to minimal API in .Net 6 | by Vijaynath Viswanathan | Medium. URL: https://medium.com/@vijaynathv/introduction-to-minimal-api-in-net-6-3b46cbc312a1 (дата звернення: 28.10.2024)

35. What is .NET Aspire? | Microsoft Learn. URL: https://learn.microsoft.com/en-us/shows/dotnet-aspire-2024/what-is-dotnet-aspire (дата звернення: 28.10.2024)

36. A Comprehensive Guide to .NET Core NuGet Package Manager | by Ravi Patel | Oct, 2024 | Medium. URL: https://medium.com/@ravipatel.it/a-comprehensive-guide-to-net-core-nuget-package-manager-9f845876a9c7 (дата звернення: 28.10.2024)

37. Getting Started Guide — AutoMapper documentation. URL: https://docs.automapper.org/en/stable/Getting-started.html (дата звернення: 28.10.2024)

38. Solving .NET Validation Challenges with Fluent Validation. URL: https://www.site24x7.com/learn/fluent-validation-in-aspnet.html (дата звернення: 28.10.2024)

39. serilog/serilog: Simple .NET logging with fully-structured events. URL: https://github.com/serilog/serilog (дата звернення: 28.10.2024)

40. ASP.NET Core appsettings.json - Dot Net Tutorials. URL: https://dotnettutorials.net/lesson/asp-net-core-appsettings-json-file/ (дата звернення: 28.10.2024)

ДОДАТКИ

Додаток А

Діаграма послідовностей