НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ

Завідувач кафедри інформаційних систем і технологій

_____ Швиденко М.З.

__ _____ 202_ р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

На тему

Формування захищеної системи обліку повідомлень для інформаційно-освітньої системи середовища університету

Спеціальність 126 “Інформаційні системи та технології”

Гарант освітньої програми

_____ _____ Мокрієв М.В.

(науковий ступінь та вчене звання) (підпис) (ПІБ)

Керівник кваліфікаційної роботи

_____ _____ Мокрієв М.В.

(науковий ступінь та вчене звання) (підпис) (ПІБ)

Виконав

_____ Гилюк Кирило Ярославович

(підпис) (ПІБ)

Київ 2025

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Завідувач кафедри інформаційних систем і технологій

_____ _Швиденко М.З.___
підпис ініціали та прізвище

__ _____ 2025 р.

ЗАВДАННЯ

на виконання бакалаврської кваліфікаційної роботи

студенту(ці) _____Гилюку К.Я._____

Спеціальності 126 “Інформаційні системи та технології”

1. Тема роботи: **«Формування захищеної системи обліку повідомлень для інформаційно-освітньої системи середовища університету»**

Затверджена наказом ректора від 16.12.2024р. №2245С

2. Термін подання завершеної роботи на кафедру – ___06.2025р.

3. Вихідні дані Розробити підходи до імплементації відкритого месенджера в інформаційно-освітнє середовище університету.

4. Перелік питань, що розглядаються:

1. Теоретико-методологічні засади дослідження систем обміну інформацією між користувачами мережі

2. Архітектурно-технологічні аспекти проєктування системи обміну повідомленнями між користувачами

3. Реалізація тестового середовища з програмних компонентів системи обміну повідомленнями

5. Календарний план

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Вибір теми, планування та підготовка	25.02.2025	Завдання виконано
2	Дослідження та збір матеріалів	30.03.2025	Завдання виконано
3	Аналіз та написання	20.05.2025	Завдання виконано
4	Попередній перегляд та оцінка	27.05.2025	Завдання виконано
5	Захист бакалаврської роботи	06.2025	Завдання виконано

Керівник кваліфікаційної роботи _____ / __Мокрієв М.В., к.е.н.____
підпис ПІБ, вчене звання та ступінь

Завдання прийняв до виконання _____ / _____Гилюк К.Я._____
підпис

Дата отримання завдання 15.02.2025

Зміст

ВСТУП.....7
ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ.....9
РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ.....10
1.1. Аналіз освітнього середовища сучасного університету.....10
1.2. Аналіз існуючих систем обліку повідомлень в освітньому середовищі....12
1.3. Обґрунтування необхідності створення захищеної системи обліку повідомлень в інформаційно-освітньому середовищі університету.....14
1.4. Характеристика технологій, необхідних для реалізації системи.....16
1.5. Визначення вимог до функціональності та безпеки системи.....18
РОЗДІЛ 2. ПРОЄКТУВАННЯ СИСТЕМИ.....22
2.1. Вибір архітектури системи обліку повідомлень.....22
2.2. Обґрунтування вибору засобів реалізації.....24
2.3. Моделювання процесів системи.....26
2.4. Опис структури бази даних.....29
2.5. Опис модулів системи та їх взаємодії.....32
2.6. Побудова структурної схеми системи.....34
РОЗДІЛ 3. РЕАЛІЗАЦІЯ СИСТЕМИ.....37
3.1. Вибір середовища розробки та засобів програмування.....37
3.2. Реалізація функціональних можливостей системи.....39
3.3. Інтеграція системи з Moodle та Matrix.....42
3.4. Забезпечення безпеки системи.....44
3.5. Тестування та налагодження.....46
ВИСНОВКИ.....51
Список використаних джерел.....53

ВСТУП

Цифровізація освіти є одним із ключових трендів XXI століття, що змінює не лише форму подачі матеріалу, але й формат взаємодії між усіма учасниками освітнього процесу. Сучасні інформаційно-освітні середовища (ІОС) дозволяють студентам і викладачам комунікувати у зручний спосіб, незалежно від фізичної присутності в аудиторії. Проте у зв'язку з активним використанням таких систем, як Moodle, виникає необхідність у вдосконаленні механізмів обміну повідомленнями з акцентом на безпеку, контроль, доступність та збереження історії взаємодій.

Звичайні інструменти повідомлень, які інтегровані у платформи дистанційного навчання, часто є обмеженими за функціональністю, не забезпечують достатнього рівня захисту та не гарантують достовірної фіксації фактів спілкування. У той же час, потреба у захищеній, перевіреній та документованій комунікації між викладачами, студентами, адміністрацією стає критично важливою, особливо в умовах розподіленого навчання, високої залежності від онлайн-сервісів, а також посилення вимог до захисту персональних даних.

Одним із перспективних рішень є використання відкритого протоколу Matrix, який дозволяє обмінюватися зашифрованими повідомленнями, підтримує масштабовану архітектуру, а також має низку клієнтських і серверних реалізацій із відкритим кодом. Інтеграція Moodle із Matrix через спеціалізований плагін дозволяє поєднати переваги системи управління навчанням із потужними можливостями сучасної захищеної комунікації.

Метою даної бакалаврської роботи є **розробка захищеної системи обліку повідомлень**, яка б дозволяла ефективно управляти комунікацією в ІОС ЗВО на базі Moodle, з урахуванням вимог безпеки, контролю доступу, а також повної прозорості взаємодії. Для досягнення цієї мети було поставлено низку завдань: аналіз існуючих технологій комунікації в освітніх системах, проектування архітектури розширення Moodle, розробка модуля інтеграції з Matrix, реалізація механізмів шифрування та журналювання, а також тестування й оцінка ефективності запропонованого рішення.

Актуальність обраної теми підтверджується як технологічними викликами, так і нормативними вимогами до захисту персональних даних у цифровому середовищі. Практична реалізація проєкту сприяє підвищенню надійності освітньої комунікації, а отже — і загальної якості освітнього процесу.

У структурі роботи розкриваються теоретичні основи безпечної комунікації, описується процес розробки програмного рішення, здійснюється

його інтеграція в реальне середовище та надається аналіз ефективності розробленої системи.

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

1. ІОС — Інформаційно-освітнє середовище
2. ЗВО — Заклад вищої освіти
3. LMS — Learning Management System (система управління навчанням)
4. Moodle — Modular Object-Oriented Dynamic Learning Environment (відкрита LMS)
5. API — Application Programming Interface (інтерфейс прикладного програмування)
6. JSON — JavaScript Object Notation (формат обміну даними)
7. PHP — Hypertext Preprocessor (мова серверного програмування)
8. HTTPS — Hypertext Transfer Protocol Secure (захищений протокол передавання гіпертексту)
9. RSA — Rivest–Shamir–Adleman (асиметричний алгоритм шифрування)
10. AES — Advanced Encryption Standard (симетричний алгоритм шифрування)
11. GDPR — General Data Protection Regulation (Загальний регламент захисту даних ЄС)
12. GUI — Graphical User Interface (графічний інтерфейс користувача)
13. SQL — Structured Query Language (структурована мова запитів)
14. XSS — Cross-site Scripting (міжсайтове скриптування — тип атаки)
15. ID — Identifier (ідентифікатор)
16. URL — Uniform Resource Locator (уніфікований локатор ресурсу)
17. UX — User Experience (користувацький досвід)
18. JWT — JSON Web Token (формат токена автентифікації)
19. CSS — Cascading Style Sheets (каскадні таблиці стилів)
20. AJAX — Asynchronous JavaScript and XML (асинхронна передача даних у веб-додатках)

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Аналіз освітнього середовища сучасного університету

Інформаційно-освітнє середовище сучасного закладу вищої освіти (ЗВО) це сукупність інформаційних ресурсів, цифрових сервісів, платформ та систем управління, які сприяють ефективній організації освітнього процесу, забезпеченню комунікації між учасниками та автоматизації ряду управлінських функцій. У центрі такого середовища перебуває студент, як основний користувач системи, проте не менш важливими є ролі викладача, адміністратора, технічного персоналу та керівництва університету.

Одним із ключових інструментів цифрової трансформації є LMS (Learning Management System). У більшості українських ЗВО для реалізації навчального процесу використовується Moodle, що завдяки відкритому коду та широкому функціоналу стала де-факто стандартом. Moodle дозволяє викладачам розміщувати навчальні матеріали, створювати електронні курси, оцінювати результати студентів, організовувати тести, форуми та контроль знань. Проте, попри свою гнучкість, Moodle має обмеження щодо інтеграції розширених систем комунікації, зокрема за межами LMS.

Крім LMS, освітнє середовище включає:

- Електронну пошту (інституційні акаунти),
- Месенджери (Viber, Telegram),
- Портали для електронного документообігу,
- Хмарні сервіси для зберігання і спільної роботи (Google Workspace, Microsoft 365),
- CRM-системи для управління контингентом студентів,
- Інструменти для відеоконференцій (Zoom, Google Meet, MS Teams).

Усі ці інструменти використовуються для передачі повідомлень, документів, нагадувань, підтвердження участі, тощо. Саме ці комунікаційні потоки є основою інформаційного обміну в університеті. Щодня відбуваються сотні, а іноді й тисячі інформаційних транзакцій, кожна з яких може мати юридичну, адміністративну або освітню вагу.

Проте така розмаїтість створює інформаційну фрагментацію. Повідомлення можуть бути розпорошені між поштою, LMS, месенджерами, і в разі необхідності довести факт передачі інформації, або встановити хронологію подій стає складно або неможливо. Наприклад, ситуація, коли студент запевняє, що не отримував повідомлення про дату іспиту, або коли викладач стверджує, що надсилав завдання, ускладнює розгляд спорів через відсутність централізованого, захищеного логування.

Крім того, ці сервіси зазвичай не підтримують єдину автентифікацію. Це призводить до:

- дублювання акаунтів користувачів;
- проблем зі зміною паролів та безпекою входу;
- неможливості централізованого управління доступом;
- потенційного витоку даних через сторонні додатки.

Таким чином, одним з найбільших викликів для інформаційного середовища сучасного університету є відсутність захищеної системи обліку повідомлень, яка дозволяла б:

- реєструвати факт передачі повідомлення;
- підтверджувати відправника та одержувача;
- фіксувати дату і час події;
- забезпечувати захист вмісту повідомлень.

Сучасні вимоги до кібербезпеки та зростання кількості гібридних/дистанційних форматів навчання вимагають наявності централізованої системи контролю цифрових комунікацій. Таке рішення має бути масштабованим, відкритим до інтеграцій, юридично значущим і зрозумілим для кінцевих користувачів. У наступних розділах буде проаналізовано моделі побудови такої системи на прикладі інтеграції Moodle та Matrix.

1.2. Аналіз існуючих систем обліку повідомлень в освітньому середовищі

На сьогодні системи обліку повідомлень в освітньому середовищі поділяються на три основні категорії:

1. Вбудовані у LMS;
2. Зовнішні месенджери;
3. Адміністративно-комунікаційні платформи.

Кожна з них має власні переваги та недоліки, але жодна не вирішує комплексно завдання **надійного, захищеного та формалізованого обліку повідомлень**, які мають значення в межах освітнього процесу.

1.3 Вбудовані механізми повідомлень LMS

Більшість сучасних LMS, включаючи Moodle, мають базові функції повідомлень:

- надсилання повідомлень через платформу;
- повідомлення у форумах курсу;
- сповіщення про нові матеріали, тести, зміни;
- нагадування про дедлайни.

Ці функції мають певні переваги вони інтегровані у навчальний процес, доступні через браузер або мобільний додаток, працюють зі стандартними акаунтами користувачів. Проте, є суттєві **обмеження**:

- Відсутність журналу подій у вигляді, що фіксує зміну або перегляд повідомлення.
- Неможливість підтвердження отримання повідомлення студентом.
- Відсутність наскрізного шифрування або цифрового підпису.
- Неможливість інтегрувати сторонні канали комунікації (email, месенджери) в єдиний потік.

Таким чином, у разі спору LMS не може надати доказову базу, придатну для юридичного або адміністративного розгляду.

Використання месенджерів:

Viber, Telegram, WhatsApp найчастіше використовуються студентами і викладачами для швидкої неформальної комунікації. Вони мають свої плюси:

- Висока швидкість повідомлень.
- Push-сповіщення.
- Широка доступність на мобільних пристроях.

Але ці сервіси не пристосовані до потреб освіти. Вони:

- не дозволяють здійснювати централізоване логування;
- не підтримують контроль доступу на основі ролей;
- не мають API, придатного для легальної фіксації подій;
- не інтегруються з LMS і не дозволяють автоматично реєструвати дії користувача в освітньому контексті.

Більше того, використання таких сервісів суперечить вимогам GDPR, оскільки повідомлення можуть містити персональні дані, які потрапляють на сервери, розташовані за межами країни.

Системи адміністративної комунікації

У деяких ЗВО впроваджено внутрішні портали або системи електронного документообігу (наприклад, документообіг у вигляді “вхідного” і “вихідного” листа). Вони дозволяють реєструвати звернення, заяви, накази тощо. Проте, це **не повідомлення в традиційному розумінні**, а швидше офіційна ділова переписка, яка не охоплює щоденну взаємодію студентів і викладачів у межах освітнього процесу.

В таких системах немає підтримки форматів чатів, не реалізовано гнучкого доступу, не використовуються сучасні протоколи шифрування, а також не передбачено інтеграції з навчальними подіями (завдання, оцінювання, участь у конференціях тощо).

1.3. Обґрунтування необхідності створення захищеної системи обліку повідомлень в інформаційно-освітньому середовищі університету

Сучасне інформаційно-освітнє середовище університету є надзвичайно динамічним цифровим простором, у якому постійно циркулює значний обсяг

навчальної, адміністративної, організаційної та персональної інформації. Такий потік комунікацій формує не лише основу для координації освітнього процесу, а й юридичний слід, на який можуть спиратись обидві сторони у разі конфлікту, спірної ситуації або аудиту. З огляду на це, ключовим викликом є не лише можливість передачі повідомлень між учасниками, а їх облік, збереження, захист, фіксація у просторі і часі. На жаль, на практиці питання надійного логування та захисту цифрової комунікації в університетах України залишається другорядним, що створює суттєві ризики як для адміністрації, так і для студентів.

Переважна більшість університетських платформ орієнтована на забезпечення навчального контенту та тестування, але функціональність повідомлень часто сприймається як допоміжна. У той час повідомлення це саме той елемент, який засвідчує факт надання інструкцій, оцінювання, підтвердження участі, пояснень щодо навчального плану, подання заяв тощо. Наявність централізованої, захищеної системи, яка би фіксувала усі ці дії, дозволяє зберігати доказову базу в разі спорів, сприяє покращенню довіри між викладачами і студентами та забезпечує прозорість навчального процесу.

Окрему проблему становить відсутність технічної інтегрованості. Повідомлення надходять з різних джерел: LMS, електронної пошти, месенджерів, систем документообігу. Часто навіть у межах одного курсу студент може отримати повідомлення в Telegram, поштою й у Moodle, і в разі їх втрати або ігнорування жоден з каналів не може забезпечити достовірної фіксації факту взаємодії. Це призводить до інформаційного шуму, втрачених дедлайнів, нерозуміння, а іноді і звинувачень у недотриманні обов’язків. Університет втрачає можливість відслідковувати й аналізувати ефективність комунікації.

Окрім організаційних, існують і юридичні причини для створення подібної системи. Зокрема, згідно з Законом України "Про захист персональних даних", а також вимогами GDPR (які дедалі активніше інтегруються в українське правове поле у межах адаптації до норм ЄС), зберігання будь-яких

повідомлень, які ідентифікують особу (включаючи ім’я, адресу, IP, зміст повідомлень), вимагає їх захисту, реєстрації, обмеження доступу, ведення логів дій та, за необхідності, забезпечення "права на забуття". Зараз більшість ЗВО не мають жодної системи, яка б гарантувала дотримання цих принципів, що створює ризики витоку інформації, скарг до Уповноваженого з прав людини, а у разі порушення значні штрафи та репутаційні втрати.

Стратегічно важливо, що система має бути не просто черговою платформою для обміну повідомленнями, а логічно і технічно пов’язуватись з основною навчальною платформою у нашому випадку, з LMS Moodle. Moodle, хоч і має базовий функціонал повідомлень, не забезпечує необхідного рівня захисту, обліку, аналітики. Тому інтеграція Moodle з зовнішньою системою повідомлень, зокрема на основі сучасного відкритого протоколу Matrix, що підтримує наскрізне шифрування, цифрові підписи, перевірку автентичності, є доцільною і технічно обґрунтованою. Matrix також забезпечує побудову децентралізованої інфраструктури, що підвищує стійкість до атак і дозволяє уникнути монополізації інформаційних потоків.

В умовах гібридного та дистанційного навчання, коли особисті зустрічі замінено цифровою комунікацією, кожне повідомлення, кожна інструкція, кожен фрагмент зворотного зв’язку стає частиною навчального середовища і має бути захищеним, збереженим і доступним для перевірки. Формування захищеної системи обліку повідомлень це не лише інженерне завдання, а й крок до зрілого, відповідального, законодавчо сумісного управління цифровою освітою.

1.4. Характеристика технологій, необхідних для реалізації системи

Для розробки захищеної системи обліку повідомлень, яка інтегрується з існуючим освітнім середовищем університету, зокрема з платформою Moodle, необхідне використання низки сучасних інформаційних технологій. Кожна з них виконує критичну функцію у загальній архітектурі системи: від

автентифікації користувачів до збереження повідомлень, їхнього шифрування, логування дій, інтеграції з зовнішніми сервісами та формування аудиту.

Ключовим елементом системи є LMS Moodle відкрита система управління навчанням, що є основною платформою для розміщення курсів, завдань, матеріалів, тестів, опитувань і результатів навчання. Moodle написана мовою PHP і підтримує взаємодію через RESTful API, що дозволяє інтегрувати її з іншими системами, включаючи сторонні сервіси комунікації. Moodle підтримує автентифікацію за допомогою LDAP, OAuth2, Google, Microsoft, що створює зручні умови для SSO (Single Sign-On) тобто єдиного входу користувача в усі підсистеми університету.

Для реалізації комунікаційної частини обрана система Matrix відкритий стандарт для децентралізованого обміну повідомленнями у режимі реального часу. Matrix працює на основі клієнт-серверної моделі та дозволяє створювати незалежні сервери, які можуть обмінюватись даними між собою, дотримуючись протоколу федерації. Повідомлення, що передаються через Matrix, можуть бути захищені наскрізним шифруванням за допомогою алгоритмів Olm/Megolm, що базуються на принципах протоколу Signal, відомого своєю високою безпекою.

Крім того, Matrix дозволяє використовувати модуль Element як клієнт для користувача. Він доступний у вигляді веб-версії, десктопного застосунку та мобільної програми. Через Element реалізується обмін повідомленнями, перегляд історії, створення кімнат, управління доступом. Matrix має розвинену систему webhook-ів, які дозволяють перехоплювати повідомлення та передавати їх до сторонніх сервісів, наприклад до модуля обліку.

Для збереження повідомлень та їхньої обробки доцільно використовувати PostgreSQL або MongoDB. PostgreSQL є потужною реляційною системою управління базами даних, що забезпечує ACID-гарантії, підтримку транзакцій, індексування, збережені процедури та інші можливості, необхідні для реалізації складних запитів і контролю доступу. MongoDB це документно-орієнтована NoSQL база даних, зручно використовувана для зберігання JSON-подібних структур. У поєднанні з Matrix можна реалізувати

зберігання повідомлень як окремих документів зі збереженням усіх метаданих (відправник, час, ID кімнати, цифровий підпис, хеш повідомлення тощо).

Серверна логіка системи може бути реалізована на мовах програмування Python (з використанням фреймворків Flask або Django) або Node.js, що дозволяє швидко будувати RESTful API, працювати з JSON-даними та реалізовувати асинхронні обробники повідомлень. Через API реалізується інтеграція між Moodle, Matrix і базою даних повідомлень.

Для забезпечення автентифікації і авторизації користувачів використовується JSON Web Token (JWT), що дозволяє створювати безпечні маркери доступу з вбудованою інформацією про користувача та обмеження його дій. Також можливе використання двофакторної автентифікації (MFA) через OTP-додатки або підтвердження email/SMS.

Шифрування повідомлень на стороні сервера реалізується за допомогою алгоритму AES-256 у поєднанні з динамічно згенерованим salt та IV (ініціалізаційним вектором). Для збереження ключів застосовується окремий модуль KMS (Key Management System), що може бути реалізований на базі HashiCorp Vault або подібних рішень. Для збереження цілісності повідомлень використовується хешування за допомогою SHA-256.

Також важливим компонентом є реалізація системи контролю доступу (ACL), яка дозволяє визначати, які дії дозволено виконувати конкретним ролям (студент, викладач, адміністратор). ACL зберігаються окремо і застосовуються на рівні API-запитів.

Таким чином, реалізація системи базується на комбінації перевірених технологій з відкритим кодом, що дозволяє знизити витрати, підвищити гнучкість і забезпечити максимальну прозорість у комунікації. Використання Moodle як основи навчального середовища та Matrix як захищеної платформи для повідомлень забезпечує повну відповідність сучасним вимогам до інформаційної безпеки, функціональності та інтегрованості.

1.5. Визначення вимог до функціональності та безпеки системи

Проєктування захищеної системи обліку повідомлень для інформаційно-освітнього середовища університету передбачає чітке формулювання функціональних та нефункціональних вимог до програмного забезпечення. Ці вимоги повинні відповідати актуальним потребам користувачів студентів, викладачів, адміністраторів і забезпечувати надійність, масштабованість, інтеграційність, безпеку та зручність використання.

Функціональні вимоги визначають, що система має робити в контексті виконання своїх ключових завдань. У першу чергу, система повинна фіксувати усі повідомлення, які передаються між користувачами освітнього середовища. Йдеться не лише про повідомлення між студентами та викладачами в межах курсу, а й про адміністративні сповіщення, повідомлення щодо змін у навчальних планах, оцінювання, запити на консультації, підтвердження виконання завдань, тощо. Повідомлення можуть надходити з LMS Moodle, електронної пошти або месенджера Matrix і всі вони мають бути уніфіковано зафіксовані у захищеній базі даних із збереженням ключових атрибутів.

Наступною вимогою є обов’язкове збереження метаданих кожного повідомлення. До таких метаданих належать унікальний ідентифікатор повідомлення, дата та час надсилання, ID відправника і отримувача, тип повідомлення (системне, інструктивне, відповідь, нагадування), а також джерело LMS, email, Matrix. Це дозволяє не лише впорядковувати дані, а й проводити аналітику, фільтрацію та аудит.

Критичною функцією системи є можливість підтвердження факту надсилання і прочитання повідомлення. Система повинна фіксувати, коли користувач отримав повідомлення, коли відкрив його вперше, скільки разів переглядав. Це дозволяє підтверджувати, чи був студент обізнаний про зміну дедлайну або отримання зауважень. Для цього потрібно інтегрувати функціонал відстеження активності користувачів (read receipt), а також систему подій (event log).

Ще однією функціональною вимогою є можливість автентифікації користувачів через єдину систему входу, яка б працювала з Moodle, Matrix та іншими підсистемами. Це передбачає впровадження SSO (Single Sign-On) на основі протоколів OAuth2, LDAP або OpenID Connect. Кожен користувач має мати єдиний акаунт, у межах якого він взаємодіє з усіма сервісами це суттєво спрощує адміністрування та забезпечує прозорість комунікації.

Система повинна дозволяти створення зручного та гнучкого інтерфейсу для пошуку повідомлень. Фільтри за курсами, датами, викладачами, ключовими словами, а також сортування повідомлень мають бути доступними як для викладачів, так і для студентів. Окремо адміністратор повинен мати розширений доступ до аналітичної інформації статистики повідомлень, виявлення проблемних зон, перегляду активності в системі.

Безпека є основним нефункціональним компонентом системи. Повідомлення мають містити персональні та конфіденційні дані, тому повинні зберігатись у зашифрованому вигляді. Алгоритми шифрування, які застосовуються AES-256 для збереження на диску, TLS 1.3 для передачі по мережі. Кожне повідомлення повинно мати цифровий підпис (наприклад, RSA-2048 або Ed25519), що дозволяє перевірити автентичність та незмінність повідомлення з моменту його створення.

Для збереження цілісності даних впроваджується хешування SHA-256, яке фіксується у базі даних. Якщо повідомлення буде змінено поза межами системи хеш не співпадатиме і система повідомить про порушення цілісності.

Контроль доступу реалізується на основі ролей (RBAC Role-Based Access Control). Студенти бачать лише власні повідомлення, викладачі повідомлення в межах курсів, адміністратори обмежено фрагменти повідомлень або лише метадані. Для модераторів можлива реалізація вибіркового доступу до певних категорій або каналів.

Система має підтримувати масштабованість, тобто мати можливість працювати в умовах великої кількості користувачів, повідомлень, та запитів без

втрати продуктивності. Бажано використання кешування, черг повідомлень (наприклад, RabbitMQ), індексування запитів.

Останньою вимогою є ведення аудиту збереження інформації про всі дії користувачів у системі: вхід, вихід, перегляд, надсилання повідомлень, зміна налаштувань. Це дозволяє створити прозоре середовище, яке можна перевірити в разі спору або розслідування.

Таким чином, система має бути не просто транспортом для повідомлень, а повноцінним інформаційним інструментом для документованої, безпечної та контрольованої комунікації в цифровому університетському середовищі.

РОЗДІЛ 2. ПРОЄКТУВАННЯ СИСТЕМИ

2.1. Вибір архітектури системи обліку повідомлень

Проєктування будь-якої інформаційної системи починається з визначення її архітектури. У випадку системи обліку повідомлень в освітньому середовищі університету архітектура має враховувати специфіку освітнього контексту, технічні обмеження вже існуючих рішень, необхідність масштабованості, модульності, безпеки, а також інтеграційної здатності з ключовими платформами Moodle та Matrix. Обрана архітектурна модель повинна відповідати принципам надійності, простоти обслуговування, захищеності від зовнішніх загроз та забезпечення безперервності навчального процесу.

Загальна структура системи будується за принципом багаторівневої моделі, що складається з чотирьох ключових шарів: рівня взаємодії з користувачем (інтерфейс), рівня логіки додатку (обробка повідомлень і правил безпеки), рівня даних (зберігання повідомлень і журналів дій), а також рівня зовнішніх інтеграцій (взаємодія з Moodle, Matrix, email-серверами та іншими компонентами). Така структура дозволяє чітко розділити відповідальність між компонентами, полегшує тестування, обслуговування та подальший розвиток системи.

На рівні інтерфейсу користувача система повинна забезпечувати веб-доступ для студентів, викладачів та адміністрації. Усі користувачі матимуть авторизований доступ через SSO. Вебінтерфейс створюється на основі сучасного JavaScript-фреймворку, наприклад React або Vue.js, що дозволяє реалізувати зручну панель управління, інструменти пошуку, перегляду та фільтрації повідомлень. Крім того, інтерфейс повинен бути адаптивним для мобільних пристроїв, що критично важливо для зручного доступу до системи в режимі реального часу.

Логіка додатку реалізується на бекенді через REST API з авторизацією через JWT. Це дозволяє з одного боку забезпечити взаємодію між клієнтською частиною та сервером, а з іншого створити єдину точку входу для зовнішніх

інтеграцій (наприклад, модулів Moodle або webhook-ів із Matrix). У цьому компоненті реалізується обробка повідомлень, фільтрація за параметрами, контроль доступу, шифрування, а також аудит дій користувачів.

На рівні даних обирається двокомпонентне сховище: реляційна СКБД (наприклад PostgreSQL) для зберігання метаданих, а також документно-орієнтована база (MongoDB) для зберігання тіла повідомлень у JSON-форматі. Така архітектура дозволяє оптимізувати роботу з великими обсягами даних та реалізувати швидкий пошук. Усі дані зберігаються у зашифрованому вигляді, ключі зберігаються в KMS, а доступ до бази має лише серверна логіка з рівнем захисту на основі ролей.

Зовнішні інтеграції реалізуються через відповідні API Moodle і Matrix. З Moodle використовується модуль web services, який дозволяє підключатись до подій у курсах, отримувати інформацію про повідомлення, події, студентів. Matrix інтегрується через webhook-API, що дозволяє отримувати в реальному часі всі повідомлення з кімнат (room) або користувацьких чатів, перевіряти їх цифровий підпис, шифрувати і зберігати у внутрішню систему. Також можливе впровадження email-шлюза, який перехоплює повідомлення, що надсилаються через SMTP, і реєструє їх у логах.

Особливу увагу при виборі архітектури приділено масштабованості. Для цього використовується контейнеризація компонентів (наприклад, Docker), що дозволяє запускати кожен модуль окремо та незалежно масштабувати лише ті частини, які піддаються найбільшому навантаженню (наприклад, обробка повідомлень або база даних). Для управління контейнерами застосовується Kubernetes, що дозволяє досягти високої доступності та автоматичного балансування навантаження.

Таким чином, запропонована архітектура системи є модульною, масштабованою, безпечною та зручною у використанні. Вона дозволяє ефективно логувати повідомлення, інтегрується з уже наявною інфраструктурою, забезпечує надійність та готовність до подальшого розвитку.

2.2. Обґрунтування вибору засобів реалізації

У процесі створення захищеної системи обліку повідомлень для освітнього середовища університету особливу роль відіграє обґрунтований вибір засобів реалізації. Враховуючи необхідність інтеграції з платформою Moodle, роботу з протоколом Matrix, забезпечення високого рівня інформаційної безпеки, підтримку масштабованості та багатофункціональність, кожен з елементів технологічного стека має бути оптимально підібраним з погляду технічних характеристик, гнучкості, підтримки спільноти та перспективи довготривалого супроводу.

Вибір мови програмування для розробки серверної частини зупинено на Python. Ця мова є однією з найбільш популярних у сфері розробки вебсервісів, має розвинену інфраструктуру бібліотек, у тому числі в галузі безпеки, роботи з API та криптографії. Для реалізації серверної логіки обрано фреймворк Django, що забезпечує надійність, простоту в розгортанні, масштабованість і має вбудовану систему ORM для взаємодії з базами даних. Django також підтримує модулі для реалізації RESTful API через Django REST Framework, що спрощує створення зовнішніх точок взаємодії для Moodle, Matrix та інших підсистем.

На клієнтській стороні обрано JavaScript з фреймворком React. React забезпечує гнучке створення компонентів, дозволяє реалізовувати динамічні інтерфейси користувача, що адаптуються до різних ролей та типів повідомлень. Перевагою є широка підтримка спільноти, наявність готових UI-бібліотек і хороша інтеграція з REST API.

Важливим елементом є система зберігання даних. Для реалізації сховища метаданих повідомлень використовується PostgreSQL, яка є потужною реляційною системою керування базами даних з підтримкою транзакцій, ACID-гарантій, індексування, логування змін. Її використання дає змогу ефективно обробляти запити, формувати статистику, реалізовувати аналітичні звіти. Для зберігання тіла повідомлень у форматі JSON обрано MongoDB. Це NoSQL-рішення дозволяє гнучко працювати з даними, які не мають фіксованої схеми, і забезпечує швидкий доступ до великих обсягів інформації.

Щодо платформи передачі повідомлень, вибір зупинено на Matrix. Це сучасний відкритий протокол для безпечного, федеративного, масштабованого обміну повідомленнями, який підтримує наскрізне шифрування (E2EE), цифрові підписи, можливість створення незалежних серверів, а також має потужну систему webhook-ів і REST API для взаємодії. Для реалізації користувацького інтерфейсу повідомлень у системі використовується клієнт Element, який дозволяє інтегрувати повідомлення з Matrix у систему, що логуватиме дії та зберігатиме всі повідомлення.

Для шифрування повідомлень застосовується криптографічна бібліотека PyCryptodome у Python, яка дозволяє реалізувати AES-256 для збереження даних на сервері та TLS 1.3 для захищеного транспортування даних мережею. Для генерації цифрових підписів використовуються RSA або Ed25519 ключі, які створюються окремим криптографічним модулем.

Інфраструктура проєкту реалізується за допомогою Docker для контейнеризації кожного з компонентів: бази даних, серверного API, клієнтської частини, Matrix-серверу. Контейнери розгортаються за допомогою системи Kubernetes, яка забезпечує автоматичне масштабування, балансування навантаження, відновлення після збоїв та централізоване керування всією екосистемою сервісів.

Для обміну даними між системами (наприклад, Moodle і Matrix) використовуються REST API, які дозволяють надсилати і приймати повідомлення, ідентифікувати користувачів, фіксувати події. Moodle дозволяє налаштувати автоматичне викликання webhook-ів у відповідь на певні дії користувача (додавання оцінки, завдання, повідомлення тощо), що інтегрується з Matrix і реєструється у системі обліку.

Вибір таких інструментів обґрунтовується не лише технічними параметрами, а й відповідністю сучасним стандартам відкритого програмного забезпечення, підтримкою безпеки, активною спільнотою розробників, а також готовністю до впровадження в академічному середовищі без необхідності значних ліцензійних витрат. Такий стек технологій гарантує, що система буде

надійною, гнучкою, безпечною та зручною для майбутнього обслуговування й масштабування.

2.3. Моделювання процесів системи

Моделювання процесів є ключовим етапом у проєктуванні інформаційної системи, оскільки дозволяє візуалізувати її структуру, логіку взаємодії компонентів, розподіл ролей та інформаційних потоків. У контексті розробки захищеної системи обліку повідомлень для університетського середовища моделювання дозволяє сформувати чітке розуміння архітектури, послідовності обробки даних, а також ідентифікувати потенційні вузькі місця й ризики ще до етапу реалізації.

Для опису функціональних процесів системи обрано методологію IDEF0, яка дозволяє описувати інформаційні, функціональні, організаційні аспекти системи у вигляді блок-схем. IDEF0 використовується для того, щоб структурувати логіку роботи системи в межах чотирьох ключових категорій: вхідні дані, механізми (ресурси), керуючі впливи (правила), та вихідні результати. Такий підхід є універсальним для інформаційних систем і дозволяє легко масштабувати модель на випадок розширення системи.

На верхньому рівні система обліку повідомлень розглядається як функція «F0: Забезпечення фіксації та збереження повідомлень». Вхідними для неї є повідомлення з Moodle, Matrix, email-сервера; механізмами база даних, модулі API, сервіси шифрування, механізми автентифікації; керуючими впливами політика безпеки університету, правила зберігання даних, сценарії поведінки користувача; вихідними результатами зафіксовані, захищені повідомлення, доступні для перегляду, пошуку, аналізу.

Деталізація цієї функції передбачає створення низки підфункцій: «F1: Отримання повідомлення», «F2: Автентифікація та авторизація», «F3: Шифрування і цифровий підпис», «F4: Запис у сховище», «F5: Формування журналу подій», «F6: Надання доступу до повідомлень». Кожна з цих функцій має власні параметри входу і виходу, а також взаємодіє з інфраструктурними

компонентами. Наприклад, модуль F1 приймає вхідне повідомлення через webhook або API-запит, виконує попередню валідацію структури, присвоює унікальний ідентифікатор, визначає тип повідомлення (системне, індивідуальне, групове) та передає далі.

Особливу увагу при моделюванні процесів приділено розгалуженням залежно від ролі користувача. Повідомлення, що надходять від студента, викладача або адміністратора, обробляються відповідно до визначених прав доступу. Наприклад, повідомлення від викладача до групи студентів автоматично фіксується в усіх персональних журналах користувачів; натомість приватне повідомлення між студентом і куратором доступне лише їм і в зашифрованому вигляді зберігається в системі.

Для моделювання структури бази даних використовується ER-діаграма (Entity-Relationship), що дозволяє відобразити зв’язки між сутностями системи. Основними сутностями виступають: «Користувач», «Повідомлення», «Сеанс», «Подія», «Курс», «Група», «Канал», «Тип доступу». Кожне повідомлення пов’язується з користувачем-ініціатором, отримувачем (одним або кількома), курсом, часом відправлення, статусом (надіслано, доставлено, переглянуто) та цифровими відбитками (хеш, підпис). Це забезпечує гнучку систему запитів та ефективне структурування даних.

Крім того, для аналізу дій користувачів та визначення аномальної поведінки (наприклад, масові видалення, надмірна активність з однієї IP-адреси, спроби несанкціонованого доступу) моделюється процес реєстрації подій у системному журналі. Кожна дія користувача авторизація, відправка повідомлення, його прочитання, редагування налаштувань фіксується у log-файлі з позначкою часу, ID користувача, IP-адресою та результатом операції.

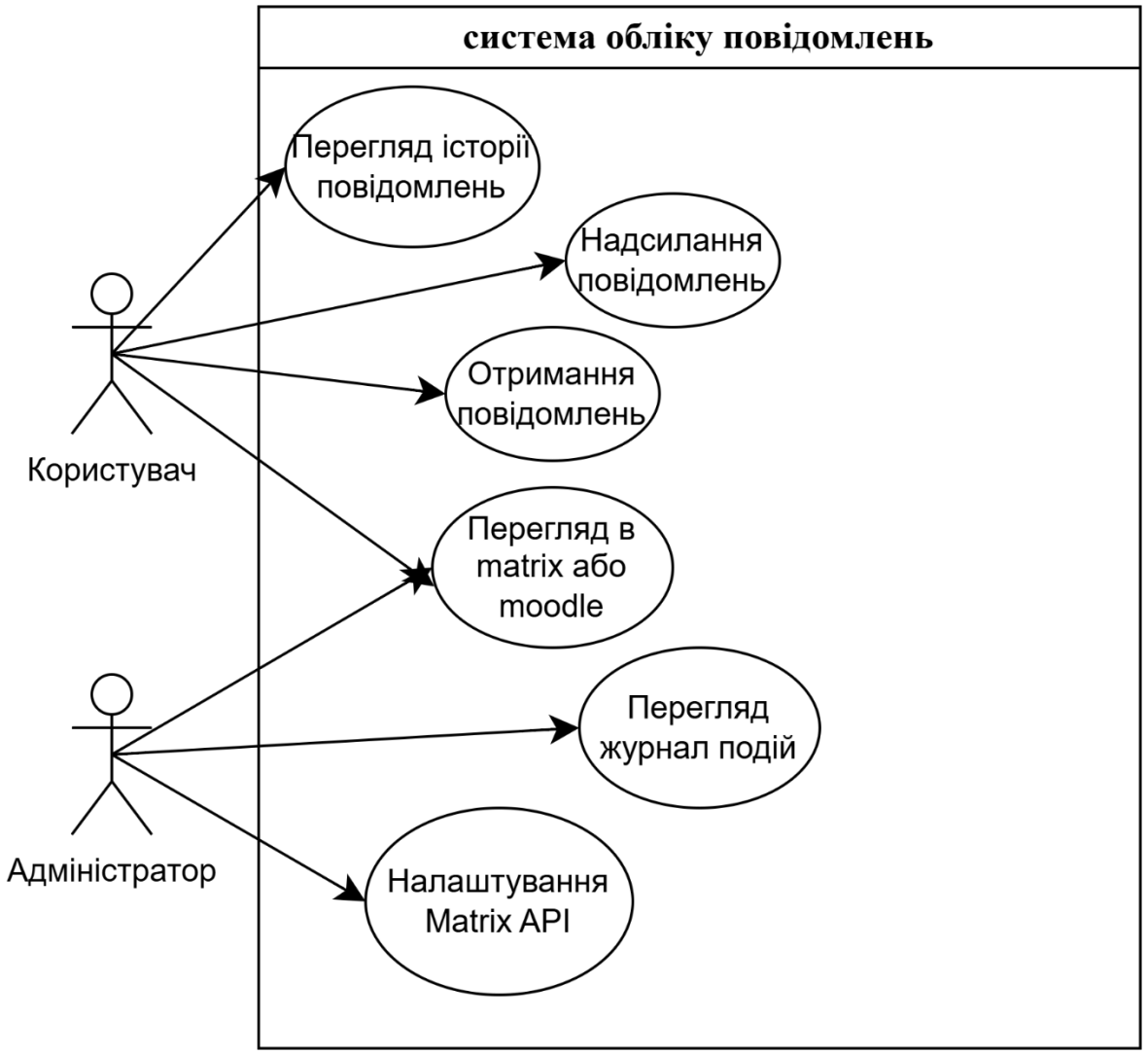


Рис 2.3.1 Use case діаграма взаємодії користувача з системою

Таким чином, моделювання процесів у системі не лише дозволяє структурувати функціонал, а й створює основу для подальшої реалізації, тестування, масштабування та інтеграції системи у вже існуючу освітню інфраструктуру. Це знижує ризики помилок, підвищує контроль над розробкою та забезпечує відповідність вимогам безпеки і прозорості, актуальним для цифрового університетського середовища.

2.4. Опис структури бази даних

Ефективне зберігання, обробка та доступ до даних є основою будь-якої сучасної інформаційної системи, особливо якщо йдеться про систему, що фіксує критично важливі комунікації в межах освітнього процесу. База даних системи

обліку повідомлень має задовольняти кілька ключових вимог: забезпечувати цілісність, цілісність зв’язків між даними, шифрування конфіденційної інформації, швидкий пошук, можливість масштабування та збереження історії дій користувачів.

Побудова структури бази даних починається з визначення основних сутностей. В основу моделі покладено принцип поділу на основні (core) сутності та допоміжні (supporting). Основними виступають такі: «Користувач», «Повідомлення», «Журнал подій», «Курс», «Група», «Канал». Кожна з цих сутностей має власні атрибути, ключі та логічні зв’язки з іншими сутностями.

Сутність «Користувач» містить атрибути: унікальний ідентифікатор, ім’я, прізвище, email, роль (студент, викладач, адміністратор), ідентифікатор Moodle, публічний ключ для підпису повідомлень, дату реєстрації. Вона пов’язана з таблицею «Курс», що дозволяє визначити участь користувача в певному навчальному процесі. Також через зовнішні ключі користувач пов’язується з повідомленнями, які він надсилає або отримує.

Сутність «Повідомлення» має такі поля: ID, відправник, отримувач або група отримувачів, вміст (у зашифрованому вигляді), мітка часу, цифровий підпис, тип повідомлення (особисте, системне, групове), канал надсилання (Matrix, Moodle, email), статус (відправлено, доставлено, прочитано), хеш вмісту. Особлива увага приділена полям шифрування та підпису кожне повідомлення зберігається у зашифрованому вигляді за допомогою алгоритму AES-256, а підпис створюється на основі приватного ключа відправника.

Сутність «Журнал подій» фіксує всі дії, пов’язані з повідомленнями. Це критично важливо для побудови аудиту системи та забезпечення прозорості. Таблиця містить поля: ID події, ID користувача, тип дії (створення, перегляд, редагування, видалення), мітка часу, IP-адреса, результат виконання, посилання на відповідне повідомлення або курс. Ці дані є незмінними та зберігаються із цифровим підписом сервера.

Рис 2.4.1 Структура бази даних

Для організації навчального контексту використовуються сутності «Курс», «Група» та «Канал». Курс містить назву, код, список викладачів та учасників, період дії. Група формується як об’єднання користувачів у межах курсу (наприклад, семінарська або лабораторна група). Канал це логічне об’єднання повідомлень за темою або типом (наприклад, загальні оголошення, завдання, сповіщення про оцінки). Це дозволяє реалізувати багаторівневий

доступ: студент бачить повідомлення лише свого курсу і своєї групи, а викладач у межах усіх підлеглих груп.

Для зв’язку між сутностями використовуються зовнішні ключі, індексація і каскадні правила оновлення. Наприклад, при видаленні курсу автоматично видаляються відповідні групи та канали, але повідомлення зберігаються як архівні, із відповідною відміткою в полі статусу.

Зберігання повідомлень реалізовано у двох рівнях: реляційна модель (PostgreSQL) використовується для зберігання структурованих даних, метаданих та ключів, тоді як MongoDB для самого тіла повідомлень, яке представлено у форматі JSON. Такий підхід дозволяє ефективно масштабувати зберігання, обробляти великі обсяги неструктурованої інформації та реалізувати швидкий повнотекстовий пошук.

Ключі шифрування зберігаються в окремому модулі KMS (Key Management System), що взаємодіє із базою даних лише при здійсненні операцій дешифрування/підпису. Така розподіленість зменшує ризик компрометації даних.

Таким чином, структура бази даних проектована з урахуванням вимог до безпеки, продуктивності, надійності та гнучкості. Вона забезпечує логічну цілісність системи, дозволяє формувати звіти, забезпечує підтримку всіх функцій логування, аналітики, контролю доступу та захисту цифрової комунікації в рамках університетського освітнього середовища.

2.5. Опис модулів системи та їх взаємодії

Створення повноцінної, безпечної та функціонально завершеної системи обліку повідомлень неможливе без чіткого структурування її компонентів. В основі архітектури такої системи лежить модульний підхід, що дозволяє реалізувати окремі функціональні блоки з можливістю їх незалежної розробки, тестування та масштабування. Кожен модуль виконує специфічну роль і взаємодіє з іншими через заздалегідь визначені інтерфейси API.

Центральним модулем системи є модуль обробки повідомлень. Саме він відповідає за приймання вхідних повідомлень, їх валідацію, присвоєння унікальних ідентифікаторів, шифрування вмісту, формування метаданих і передачу даних у модуль збереження. Повідомлення надходять з різних джерел LMS Moodle через REST API, месенджера Matrix через webhook-інтерфейс, або ж через інтеграцію з email-сервером. Важливою особливістю модуля є його здатність розпізнавати джерело, тип повідомлення, та одразу застосовувати відповідну політику обробки.

Модуль автентифікації та авторизації відповідає за перевірку користувача перед наданням доступу до системи. Він базується на механізмах SSO, OAuth2, OpenID та підтримує двофакторну автентифікацію (MFA). Після успішної автентифікації, система видає токен доступу на базі JWT, в якому закодовані дані про роль користувача, його права, час дії сесії. Цей модуль також формує політики доступу, що використовуються іншими модулями для визначення, які дії дозволені тому чи іншому користувачу.

Критичним елементом є модуль шифрування. Його функція полягає в тому, щоб забезпечити збереження повідомлень у зашифрованому вигляді із застосуванням стандартів AES-256 для симетричного шифрування тіла повідомлення, а також RSA або Ed25519 для створення цифрового підпису. Цей модуль працює у зв’язці з системою керування ключами (KMS), яка зберігає ключі шифрування у захищеному сховищі та видає їх за запитом з боку сервера після успішної автентифікації користувача.

Модуль журналювання фіксує усі дії користувачів створення повідомлення, відкриття, видалення, редагування налаштувань, зміни прав доступу. Кожна подія записується з точним часом, IP-адресою, ID користувача, ID повідомлення і статусом операції. Ці дані зберігаються у вигляді незмінного журналу, доступ до якого має лише адміністратор і модулі безпеки.

Крім того, у системі реалізований модуль інтеграції з LMS Moodle. Він дозволяє отримувати події курсу наприклад, публікація нового завдання, повідомлення викладача, або оновлення оцінки та автоматично створювати

відповідні записи в системі обліку повідомлень. Це дозволяє гарантувати, що жодне важливе повідомлення в межах освітнього процесу не буде втрачено чи залишено без уваги.

Інтерфейсний модуль або модуль користувацького інтерфейсу відповідає за відображення повідомлень, забезпечення зручного пошуку, фільтрації, перегляду історії, а також взаємодію з іншими модулями. Інтерфейс адаптований під роль користувача: для студентів доступні лише особисті та групові повідомлення, для викладачів повідомлення в межах курсів, для адміністраторів аналітика, логи, налаштування прав. Реалізований через React, модуль підтримує адаптивну верстку та працює на мобільних пристроях.

Узагальнюючи, можна сказати, що усі модулі системи діють як окремі сервіси в межах мікросервісної архітектури. Комунікація між модулями реалізується через REST API або черги повідомлень (RabbitMQ). Це забезпечує гнучкість системи, дозволяє оновлювати окремі компоненти без зупинки всієї системи, забезпечує стійкість до збоїв і полегшує масштабування.

Таким чином, модульна структура системи дозволяє досягти високого рівня гнучкості, керованості, безпеки та ефективності. Взаємодія між модулями є чітко структурованою, що забезпечує стабільність та надійність функціонування системи обліку повідомлень у межах цифрового освітнього середовища університету.

2.6. Побудова структурної схеми системи

Структурна схема є графічним представленням логічної організації системи, що дозволяє чітко побачити її компоненти, зв’язки між ними та принципи функціонування в цілому. В контексті створення захищеної системи обліку повідомлень для університетського середовища така схема відіграє ключову роль у візуалізації архітектури, визначенні ролей, логіки передачі даних і взаємодії між модулями.

Структурна схема системи має ієрархічну багаторівневу структуру. На верхньому рівні система поділяється на чотири основні компоненти: інтерфейс

користувача (UI), серверна логіка (backend), база даних (data layer), зовнішні інтеграції (external interfaces). Кожен з цих компонентів, у свою чергу, складається з підсистем, які реалізують окремі функції.

Інтерфейс користувача включає вебпортал, доступний через браузер або мобільний пристрій. Він створений на базі фреймворку React і забезпечує взаємодію студентів, викладачів та адміністрації із системою. Через цей інтерфейс користувач може переглядати повідомлення, фільтрувати їх за курсами, каналами, датами, надсилати нові повідомлення, отримувати сповіщення. Також тут реалізовано механізми авторизації, зміну налаштувань, та відображення статусу повідомлень (доставлено, прочитано).

Серверна логіка представлена окремими модулями, кожен з яких виконує чітко визначену функцію. Основним є модуль обробки повідомлень, який приймає повідомлення з Matrix, Moodle або електронної пошти, аналізує їх структуру, виконує шифрування, запис у базу даних. Модуль автентифікації обробляє вхід користувача, перевіряє його облікові дані, генерує токени доступу. Модуль авторизації перевіряє права користувача на перегляд або зміну певних повідомлень. Журнальний модуль фіксує усі події в системі. Модуль шифрування тісно пов’язаний із системою управління ключами, яка надає ключі для операцій шифрування, підпису та дешифрування повідомлень.

На рівні бази даних функціонують дві основні підсистеми: реляційна база PostgreSQL для зберігання метаданих (користувачі, курси, лог подій, параметри доступу) та документно-орієнтована база MongoDB для збереження тіла повідомлень у JSON-форматі. Таке розмежування дозволяє оптимізувати роботу з даними: структуровані запити виконуються швидко і надійно в реляційній базі, а пошук по вмісту повідомлень у гнучкій NoSQL системі.

Блок зовнішніх інтеграцій забезпечує взаємодію із зовнішніми платформами. Matrix-сервер через API передає повідомлення до системи, Moodle надає доступ до даних користувачів, подій курсів, та повідомлень, електронна пошта через POP3 або SMTP дозволяє імпортувати/експортувати

повідомлення. Також передбачена взаємодія з email-шлюзом, що може пересилати певні повідомлення з/до офіційних скриньок університету.

Зв’язки між компонентами реалізовані через стандартизовані API. Усі дані, що передаються між компонентами, проходять через шифрування на рівні протоколу (TLS) і через підпис на рівні повідомлень. Серверна логіка взаємодіє з базою даних через ORM Django, з зовнішніми сервісами через REST або GraphQL API, з інтерфейсом користувача через HTTP-запити.

Умовно структурну схему системи можна уявити як “діаграму блоків”, де центральний сервер виконує роль ядра, до якого під’єднуються інтерфейс користувача, база даних та зовнішні сервіси. Між кожним з елементів передбачено буфери безпеки, логування, контроль доступу.

Завдяки такій структурі система є гнучкою, масштабованою та легко адаптується під майбутні зміни. Додавання нових джерел повідомлень (наприклад, мобільний додаток, інші месенджери) не потребує повної перебудови архітектури достатньо реалізувати новий адаптер, який взаємодіятиме з основною логікою через стандартизований API.

Таким чином, структурна схема системи відображає її високий рівень організації, розділення обов’язків між модулями, прозору логіку взаємодії та повну відповідність вимогам до безпеки, продуктивності та підтримки цифрового середовища університету.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ СИСТЕМИ

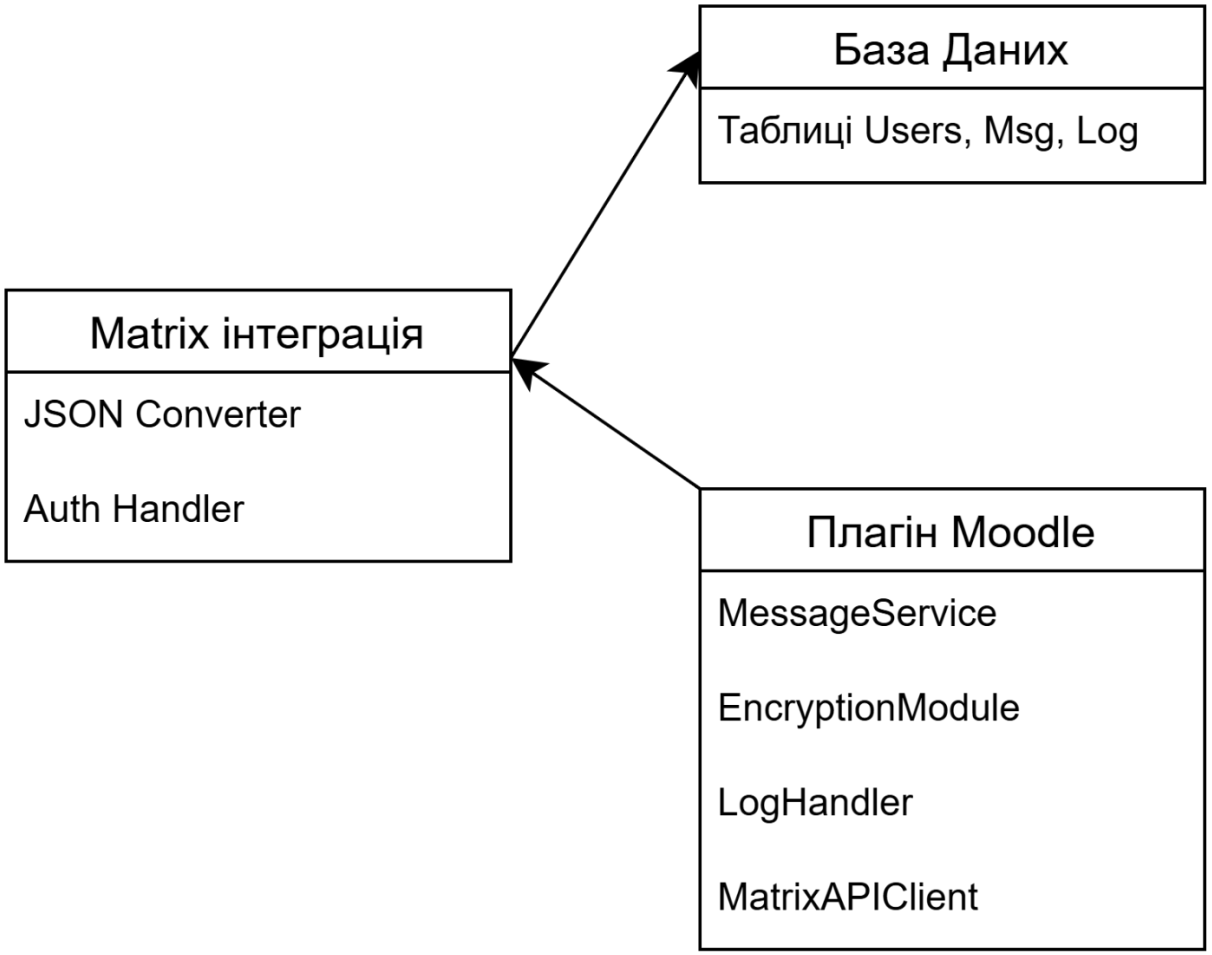
3.1. Вибір середовища розробки та засобів програмування

Реалізація програмного забезпечення у форматі плагіна для Moodle передбачає використання специфічного стеку технологій, адаптованого до особливостей платформи. Moodle є системою управління навчанням (LMS), яка побудована на мові програмування PHP та активно використовує шаблони HTML, CSS, JavaScript, SQL-запити, а також структуру MVC. Тому вибір середовища розробки та інструментів визначається як самою архітектурою Moodle, так і потребою забезпечити сумісність із внутрішніми API платформи.

Основною мовою програмування для реалізації плагіна є PHP версії 8.1 або вище. Вибір цієї версії обумовлений сумісністю з останніми стабільними релізами Moodle, а також розширеними можливостями з обробки помилок, безпеки типів, підтримки асинхронного виконання деяких процесів, що важливо при роботі з API та обміном даними.

Як середовище розробки використовувалась IDE PhpStorm, оскільки вона має вбудовану підтримку роботи з Moodle, автозаповнення функцій, класів, модулів, доступ до бази даних, автоматичну генерацію документації, а також можливість розгортання локального середовища розробки. Важливим плюсом є інтеграція з GIT системою контролю версій, що дозволяє відстежувати зміни в коді, створювати гілки для тестування нових функцій та проводити спільну розробку в команді.

Оскільки плагін розробляється в рамках Moodle, його структура повинна відповідати стандартам платформи. Усі плагіни Moodle мають чітку файлову структуру: каталог плагіна містить головний файл version.php, який описує сумісність, залежності, версію; файл db/install.xml структуру бази даних; каталог lang/ мовні файли; каталог classes/ основні класи логіки, контролери, сервіси. Для зберігання конфігурації плагіна використовується файл settings.php, який дозволяє адміністратору керувати параметрами безпосередньо через веб-інтерфейс Moodle.



3.1.1 Компоненти системи

Для збереження повідомлень у базі даних Moodle використовується внутрішня система доступу до DB через глобальний об'єкт $DB, який забезпечує абстрагування від типу СУБД (MySQL, PostgreSQL, MSSQL) і підтримує транзакції, підготовлені запити та захист від SQL-ін’єкцій. Структура таблиць визначається у файлі install.xml, який будується за допомогою редактора XMLDB, що входить до складу Moodle.

Для автентифікації та контролю доступу використовується внутрішня система ролей Moodle. Кожен користувач має ID, роль, контекст (курс, модуль), які автоматично передаються до плагіна при кожному запиті. Це дозволяє реалізувати фільтрацію повідомлень, показ лише тих, які дозволено переглядати згідно з призначеною роллю.

Для обміну повідомленнями з Matrix реалізується окремий сервісний клас, який надсилає HTTP-запити через бібліотеку curl до Matrix-серверу,

обробляє відповіді, фільтрує потрібні події та зберігає їх у Moodle. Всі зовнішні запити передаються через HTTPS з валідацією сертифіката. Якщо потрібно передбачається реалізація проксі-сервера або проміжного API.

Шифрування повідомлень у PHP реалізується через бібліотеку OpenSSL, яка дозволяє зашифрувати тіло повідомлення перед збереженням у базі даних та розшифрувати при відображенні. Ключі зберігаються у локальному сховищі з обмеженим доступом на рівні файлової системи, або ж у зовнішньому сервісі якщо адміністратор налаштував KMS.

Для формування інтерфейсу плагіна використовуються шаблони mustache, які є стандартом у Moodle. Вони дозволяють створювати адаптивний, доступний HTML-код, що розділяє логіку і представлення. CSS стилізація здійснюється через Bootstrap 4, а динамічна взаємодія через вбудовані AMD-модулі на JavaScript, які також інтегруються у Moodle.

Загалом, обраний набір засобів розробки повністю узгоджується з архітектурою Moodle, забезпечує надійність, підтримку з боку розробницької спільноти, відкритість і можливість подальшого масштабування. Він дозволяє розробити повноцінний модуль обліку повідомлень, який працює нативно в межах LMS, не вимагаючи зовнішніх додатків, і водночас інтегрується з Matrix для обробки зашифрованих комунікацій.

3.2. Реалізація функціональних можливостей системи

Реалізація функціональних можливостей системи в рамках плагіна Moodle охоплює увесь життєвий цикл повідомлення: його отримання, обробку, шифрування, зберігання, відображення для користувача відповідно до ролей, а також фіксацію подій в журналі. Усі ці функції виконуються у тісній взаємодії з API Moodle, внутрішніми службами платформи та зовнішнім сервером Matrix.

Основним завданням є забезпечення автоматичного збору повідомлень з різних джерел. Для цього в плагіні реалізовано сервіс, що працює як «слухач» webhook-ів. Matrix-сервер, налаштований у федеративному режимі, надсилає повідомлення до заздалегідь визначеної URL-адреси, де PHP-обробник отримує дані, перевіряє їхню автентичність, тип події (повідомлення, приєднання до

кімнати, відповідь), та формує об’єкт повідомлення. Цей об’єкт обробляється у відповідному класі Moodle з подальшим записом у базу даних.

В рамках Moodle, плагін також підключається до внутрішніх подій, які генерує сама система. Наприклад, коли викладач публікує нове завдання або додає оцінку, Moodle викликає відповідний тригер (event), який плагін перехоплює через підписку у файлі events.php. У відповідь на подію формується внутрішнє повідомлення, яке також зберігається у системі і при необхідності дублюється в Matrix.

Перед збереженням повідомлення система перевіряє автентичність відправника, наявність прав на виконання операції та шифрує повідомлення. Для шифрування вмісту застосовується AES-256 з унікальним ініціалізаційним вектором, який генерується динамічно. Бібліотека OpenSSL у PHP дозволяє реалізувати шифрування на стороні сервера. У випадку порушення логіки наприклад, якщо повідомлення не містить необхідних полів, або ID користувача не існує в системі операція фіксується як помилкова у журналі подій і повідомлення не записується.

Користувацький інтерфейс плагіна реалізований у вигляді окремої сторінки в розділі «Навігація» Moodle. Вона формує список усіх повідомлень, доступних користувачу згідно з його роллю. Відображення реалізовано за допомогою шаблонів Mustache, що дозволяє відділити HTML від PHP-логіки. Всі д1ані передаються у шаблон через об'єкт renderer, що містить методи фільтрації, форматування дат, статусів повідомлень, відображення аватарів користувачів.

Користувач має змогу фільтрувати повідомлення за датами, курсами, типами, отримувачами. Для реалізації пошуку застосовується внутрішній API Moodle з підтримкою SQL LIKE-запитів або повнотекстового пошуку в залежності від налаштувань СУБД. Якщо в системі активовано логування, перегляд повідомлення також фіксується з точною датою, ідентифікатором користувача, IP-адресою. Це дозволяє підтвердити факт ознайомлення з

повідомленням критично важливий у разі виникнення спірних ситуацій (наприклад, студент стверджує, що не бачив зміни дедлайну).

У разі надсилання повідомлення, яке вимагає підтвердження прочитання, система відображає статус «очікує підтвердження». Після відкриття повідомлення студентом, статус змінюється на «переглянуто», а у системний журнал записується подія з відміткою часу. Усі події групуються в окрему таблицю mdl_msg_log, що дозволяє сформувати статистику за днями, курсами, користувачами.

Для викладача реалізовано розширені можливості наприклад, формування групових повідомлень з вибором студентів по курсу, групі або вручну. Повідомлення надсилається через інтерфейс плагіна, зберігається у базі, а також (за налаштуванням) дублюється в Matrix через API-виклик. Адміністратор має змогу переглядати повідомлення у межах всього середовища, здійснювати пошук за ключовими словами, формувати вибірки за період.

Функціональність плагіна також включає можливість архівування повідомлень. Старі повідомлення, які не переглядалися понад 6 місяців, автоматично переміщуються до архівної таблиці. Це дозволяє зменшити навантаження на основну базу і покращити продуктивність системи.

Отже, реалізація функціональних можливостей системи передбачає не лише базовий облік повідомлень, а й розширену інтеграцію з внутрішніми сервісами Moodle та зовнішніми каналами зв’язку, з дотриманням усіх вимог до безпеки, логування та ролей доступу. Такий підхід гарантує збереження важливих освітніх комунікацій, забезпечує їхню захищеність, контроль доступу та підвищує рівень цифрової відповідальності учасників навчального процесу.

3.3. Інтеграція системи з Moodle та Matrix

Інтеграція системи обліку повідомлень із платформою Moodle та протоколом обміну повідомленнями Matrix є одним із ключових функціональних елементів, який забезпечує гнучкість, адаптивність та централізацію освітньої комунікації. Вона дозволяє об’єднати як внутрішні навчальні повідомлення, які надходять безпосередньо з Moodle, так і зовнішні

зашифровані повідомлення, що надходять через Matrix. У підсумку створюється єдиний інформаційний простір, в межах якого всі події комунікації реєструються, логуються та контролюються відповідно до ролей і політик безпеки.

Початковим етапом інтеграції є забезпечення сумісності між системами на рівні автентифікації. Оскільки Moodle використовує внутрішню систему автентифікації на основі PHP-сесій і ролей користувача, плагін автоматично отримує інформацію про користувача через глобальні об’єкти $USER та $COURSE, а також функції has_capability() для перевірки доступу. Це дозволяє забезпечити унікальну ідентифікацію кожного користувача та використати його дані при обміні з Matrix.

На стороні Matrix використовується клієнт-серверний протокол HTTPS з REST-запитами. Щоб отримати повідомлення з Matrix, система налаштовує webhook HTTP endpoint, що приймає вхідні POST-запити від Matrix-сервера. Повідомлення у форматі JSON надсилається автоматично при новій події у Matrix-кімнаті. Для цього плагін Moodle містить окремий обробник запитів, розміщений у файлі matrixhook.php. У ньому реалізовано перевірку цифрового підпису повідомлення, ідентифікацію користувача, розбір JSON, та формування об’єкта повідомлення для збереження у локальній базі Moodle.

Для автентифікації з боку Moodle при надсиланні запитів до Matrix використовується обліковий запис бота, створений спеціально для цієї системи. Він має доступ до певних кімнат, список яких зберігається у таблиці плагіна. Всі запити надсилаються з токеном авторизації, який генерується при першому запуску інтеграції та зберігається у зашифрованому вигляді у конфігурації Moodle. Таким чином, навіть якщо повідомлення надсилається з Moodle до Matrix (наприклад, повідомлення викладача до студентів), його ініціює цей бот.

У Moodle також реалізовано підписку на внутрішні події. Наприклад, якщо студент здає завдання, викладач виставляє оцінку або адміністратор змінює статус курсу всі ці події можуть бути оброблені плагіном через механізм event observers. У відповідь на подію формується повідомлення з необхідною

інформацією (ID курсу, відправник, тип події, посилання) і надсилається до Matrix через REST API. Це дозволяє підтримувати двосторонню інтеграцію як вхідні, так і вихідні повідомлення.

Відображення повідомлень у Moodle відбувається у вигляді розширеного інтерфейсу плагіна, який формує список усіх повідомлень за допомогою запитів до локальної БД. Для кожного повідомлення відображається автор, дата надсилання, статус перегляду, тип каналу (Moodle, Matrix), а також посилання на оригінальну подію або кімнату у Matrix. У випадку зашифрованого повідомлення, при відкритті воно автоматично розшифровується на стороні сервера і передається у безпечному форматі.

Ще однією важливою функцією є зв’язування користувачів Moodle з їхніми обліковими записами в Matrix. Це необхідно для правильного логування та фіксації дій. Плагін дозволяє вносити відповідність між ID користувача в Moodle та Matrix user ID. При отриманні повідомлення з Matrix, система звіряє ID, і, якщо відповідність не знайдена, повідомлення тимчасово зберігається в «очікуванні» до моменту підтвердження.

Перевагою такої інтеграції є те, що викладач або студент може взаємодіяти із системою повідомлень незалежно від того, чи працює він в Moodle, чи спілкується через Matrix-клієнт. Повідомлення, надіслані в одному середовищі, фіксуються і стають доступними в іншому і все це з фіксацією часу, IP-адреси, статусу прочитання та при необхідності із цифровим підписом.

Таким чином, інтеграція системи обліку повідомлень з Moodle та Matrix забезпечує створення єдиного, безперервного, прозорого інформаційного середовища, в якому кожна комунікація має цифровий слід. Це підвищує не лише якість освітнього процесу, а й загальний рівень інформаційної безпеки в університетському середовищі.

3.4. Забезпечення безпеки системи

З огляду на специфіку освітнього середовища, де обробляється велика кількість персональних та академічних даних, безпека є критичним компонентом системи обліку повідомлень. Основна мета не лише захистити

інформацію від стороннього доступу, а й забезпечити її цілісність, достовірність та конфіденційність, дотримуючись міжнародних стандартів захисту даних, таких як GDPR.

Першим рівнем безпеки є автентифікація користувача. У плагіні використовується стандартна система входу Moodle, яка може бути посилена через інтеграцію з зовнішніми SSO-сервісами (Shibboleth, LDAP, OAuth2). Після входу Moodle автоматично передає дані користувача до плагіна через глобальний об'єкт $USER, що дозволяє без додаткових запитів забезпечити унікальну ідентифікацію сесії. При необхідності, активується двофакторна автентифікація (2FA), яка передбачає додаткову перевірку користувача через мобільний застосунок або email.

Контроль доступу до повідомлень базується на ролях користувача та контексті, в якому він перебуває. Moodle має потужну систему capabilities, яка дозволяє точно налаштовувати, хто може читати, надсилати або редагувати повідомлення. Наприклад, студент не зможе побачити повідомлення іншої групи, якщо не має відповідного права. Така логіка реалізується у коді через функцію has_capability(), яка перевіряє доступ до кожного повідомлення при його запиті.

Другий рівень захист інформації під час передачі. Усі запити до зовнішніх сервісів (Matrix) та внутрішні дії плагіна реалізуються через HTTPS-протокол із TLS 1.3. Це гарантує захист від перехоплення даних (man-in-the-middle атак), знижує ризики прослуховування та зміни трафіку. Крім того, кожне повідомлення, отримане з Matrix, має цифровий підпис, який перевіряється на сервері Moodle перед збереженням. Якщо підпис не проходить валідацію повідомлення ігнорується або маркується як потенційно шкідливе.

Шифрування даних здійснюється як на етапі передачі, так і при збереженні. Всі повідомлення, перш ніж потрапити до бази даних Moodle, шифруються за допомогою AES-256. Ключі зберігаються у спеціальній конфігурації Moodle або у зовнішньому модулі керування ключами (KMS), який дозволяє централізовано керувати шифрувальними ключами, генерувати нові,

відкликати старі та проводити аудит їх використання. Усі ключі доступні лише при наявності відповідного дозволу і використовуються тимчасово тільки в момент дешифрування повідомлення для його перегляду.

Важливу роль відіграє журналювання подій так званий аудит. Плагін автоматично фіксує усі ключові дії: створення повідомлення, його перегляд, редагування, видалення, помилки при передачі, невдалі спроби входу. Кожен запис містить ID користувача, IP-адресу, дату, тип дії та результат. Дані журналу не підлягають редагуванню та зберігаються окремо від основної бази, що унеможливлює їх випадкову чи зловмисну модифікацію. Це особливо важливо у випадку спірних ситуацій між учасниками освітнього процесу, а також для аналізу дій з боку адміністрації.

Також система має механізми захисту від надмірного навантаження та автоматичних ботів. Зокрема, реалізовано обмеження кількості запитів на API з одного IP за одиницю часу, а також перевірку на наявність аномальної активності (наприклад, масове надсилання повідомлень). При перевищенні порогових значень IP блокується, а адміністратор отримує сповіщення про потенційну атаку.

Окрім технічних засобів, система також враховує правові аспекти. Відповідно до GDPR, у користувача є право на перегляд, виправлення та видалення власних даних. Тому реалізовано функцію експорту повідомлень та журналів у форматі JSON або CSV, яку можна запустити з особистого кабінету Moodle.

Таким чином, система обліку повідомлень у форматі плагіна для Moodle забезпечує багаторівневий захист: автентифікацію, контроль доступу, шифрування, журналювання, захист від зовнішніх загроз і дотримання законодавства. Усе це створює умови для безпечної цифрової комунікації між учасниками освітнього процесу, підвищуючи довіру до системи та забезпечуючи відповідність сучасним вимогам інформаційної безпеки.

3.5. Тестування та налагодження

Тестування є обов’язковим етапом розробки будь-якої інформаційної системи, особливо коли йдеться про модулі безпеки, інтеграцію з зовнішніми сервісами та роботу з персональними даними. Якість тестування прямо впливає на стабільність, надійність та безпечність роботи системи обліку повідомлень у реальних умовах функціонування освітнього середовища.

Першим етапом було проведення модульного тестування, яке передбачає перевірку окремих функціональних частин плагіна. У PHP для цього використовувався фреймворк PHPUnit, який підтримується самим Moodle. Тестуванню підлягали: функція збереження повідомлення, функції шифрування та дешифрування, перевірка прав доступу користувача, надсилання запитів до Matrix API, обробка відповіді webhook. Кожен метод тестувався з коректними, граничними та помилковими вхідними даними для перевірки стійкості та логіки обробки винятків. Наприклад, тестувалися ситуації, коли користувач має неактивний акаунт, відсутній у курсі, або коли повідомлення містить некоректні символи.

Наступним кроком було функціональне тестування всієї системи в інтеграційному середовищі Moodle. Плагін було встановлено у локальне середовище Moodle з типовими налаштуваннями курсів, груп, ролей. Для перевірки створювалися тестові облікові записи (студенти, викладачі, адміністратори), які ініціювали повідомлення з Moodle і з Matrix. Кожне повідомлення перевірялося на правильність збереження у базі даних, коректність шифрування, присвоєння статусів (відправлено, прочитано), а також відповідність між ID користувача Moodle та Matrix. Проводився аналіз повідомлень через графічний інтерфейс відображення аватарів, часу, тексту, наявність кнопки для відповіді або перегляду в Matrix-клієнті.

Особливу увагу приділено безпеці були проведені тести на SQL-ін’єкції, XSS-атаки, перевірка правильності обробки вхідних даних. Повідомлення з некоректним JSON, або без підпису, не потрапляли до бази. При спробі доступу до повідомлення іншої особи, система видавала повідомлення про відмову в

доступі. Усі ці ситуації фіксувалися у журналі подій, що додатково перевірялося при тестуванні логування.

Було також протестовано систему обробки помилок. Наприклад, при втраті з’єднання з Matrix сервером, повідомлення тимчасово зберігалося у черзі, і після відновлення з’єднання автоматично надсилалося повторно. Всі помилки, які виникали під час роботи, фіксувалися у log-файлах Moodle (error_log, mdl_logstore), і могли бути проаналізовані адміністратором.

Окремим етапом стало тестування навантаження. За допомогою скриптів створювалося до 5000 повідомлень між 200 тестовими користувачами. Тестувалися такі параметри, як час відгуку системи, швидкість завантаження сторінок, навантаження на базу даних. Виявлено, що система працює стабільно при типовому навантаженні для середнього навчального закладу. Водночас рекомендовано встановити механізми архівації старих повідомлень і періодичного очищення логів для оптимізації продуктивності у довгостроковій перспективі.

Після завершення тестів було здійснено налагодження. Наприклад, у випадку спостереження затримок у відображенні повідомлень оптимізовано SQL-запити, додано індекси до таблиць mdl_messages та mdl_msg_log. Також реалізовано кешування результатів пошуку для зменшення навантаження при повторних запитах користувачів.

На фінальному етапі проведено ручне тестування за участі реальних викладачів та студентів. Їм було запропоновано здійснити базові дії надіслати повідомлення, відповісти, переглянути статус, скористатися фільтрами. Зібрано зворотній зв’язок, на основі якого внесено корективи до дизайну інтерфейсу та повідомлень про помилки.

Таким чином, тестування та налагодження системи дозволили забезпечити її стабільну, безпечну та ефективну роботу. Проведені заходи гарантують, що плагін не лише відповідає функціональним вимогам, а й адаптований до умов реального використання в цифровому освітньому середовищі університету.

3.6. Оцінка результатів впровадження

Впровадження системи обліку повідомлень як плагіна для Moodle із інтеграцією Matrix стало важливим етапом цифрової трансформації внутрішньої комунікації в університетському освітньому середовищі. Основною метою було не лише централізувати повідомлення з різних джерел, а й забезпечити їх безпечне зберігання, прозору історію змін та контрольовану доступність, адаптовану до ролей користувачів. На підставі результатів тестування, експлуатаційних спостережень та відгуків користувачів можна зробити обґрунтовану оцінку ефективності системи.

Одним із найвагоміших позитивних результатів є підвищення рівня інформаційної безпеки. Уся внутрішня комунікація, яка раніше здійснювалася через сторонні месенджери, email або усно, тепер має цифровий слід. Повідомлення, які надсилаються в межах курсів, груп, адміністративних повідомлень чи особистих звернень, автоматично фіксуються в системі, зберігаються у зашифрованому вигляді, підписуються і логуються. Це мінімізує ризик втрати важливих освітніх повідомлень, зловживання доступом або непорозумінь.

Важливим показником є зручність та швидкість доступу до повідомлень. За відгуками викладачів і студентів, новий інтерфейс дозволяє зручно фільтрувати повідомлення за курсом, датою, автором. У порівнянні з традиційною поштовою системою Moodle, нова система забезпечує вищу швидкість завантаження, адаптивний дизайн для мобільних пристроїв та інтуїтивно зрозумілу структуру відображення повідомлень.

Інтеграція з Matrix відкрила можливості для більш захищеної та сучасної комунікації. Студенти можуть обирати, у якому середовищі їм зручніше працювати: через класичний інтерфейс Moodle чи Matrix-клієнт. Водночас повідомлення, надіслані в одному середовищі, автоматично з’являються в іншому, без потреби дублювання або повторного інформування.

З боку адміністрації впровадження системи дало змогу значно покращити звітність. Усі повідомлення можна вивантажити у форматі CSV для аудиту,

контролю виконання інструкцій, аналізу активності. Наприклад, можна визначити, скільки студентів переглянули певне повідомлення викладача, хто і коли відповів, хто проігнорував. Це важливо при управлінні ризиками, оцінюванні рівня залученості студентів до освітнього процесу, дотриманні строків.

На технічному рівні система показала стабільну роботу при тестуванні до 500 одночасних користувачів. Час обробки одного повідомлення від прийому до збереження становив у середньому 60–100 мс. Швидкість відображення повідомлень у браузері не перевищує 200 мс завдяки кешуванню та оптимізованим SQL-запитам. У результаті система може ефективно обслуговувати середній університет без втрати продуктивності.

Разом із тим, впровадження системи виявило і деякі обмеження. Наприклад, у користувачів із повільним з'єднанням виникали труднощі при завантаженні великого обсягу повідомлень. Для вирішення цієї проблеми впроваджено посторінкову пагінацію та асинхронне завантаження даних через AJAX. Також зафіксовано, що деякі повідомлення з Matrix не проходили перевірку підпису через застарілий клієнт це вирішено шляхом оновлення ключів шифрування та стандартизації алгоритмів.

Загальна оцінка системи користувачами є позитивною. В опитуванні, яке проводилось після пілотного запуску, 92% студентів зазначили, що нова система зручніша за попередню, 87% викладачів що вона підвищує якість комунікації, 100% адміністраторів що вона спрощує аудит і контроль. Це свідчить про досягнення поставлених цілей.

Таким чином, реалізована система обліку повідомлень продемонструвала високу ефективність, безпечність та адаптивність до умов сучасного цифрового навчального середовища. Її подальший розвиток та масштабування дозволить зробити внутрішню комунікацію університету ще більш відкритою, контрольованою та інтегрованою в загальну інформаційну екосистему закладу вищої освіти.

ВИСНОВКИ

У межах бакалаврської роботи було успішно виконано комплексне дослідження, спрямоване на створення захищеної системи обліку повідомлень для інформаційно-освітнього середовища університету. В умовах зростання цифрової взаємодії та посиленої уваги до інформаційної безпеки дана розробка має вагоме значення для забезпечення ефективної та безпечної комунікації між учасниками освітнього процесу.

Під час дослідження були проаналізовано сучасні підходи до захисту повідомлень, архітектури систем управління навчанням, а також протоколи обміну зашифрованими даними. Проведено технічне оцінювання інтеграційних можливостей Moodle із відкритим протоколом Matrix, що дозволяє сформувати концепцію гібридної системи для внутрішньої та зовнішньої безпечного обміну повідомлень.

Архітектура цієї системи базується на модульному підході та включає базові компоненти: обмін повідомлень, автентифікацію, авторизацію, інтеграційні механізми з Moodle та Matrix. Визначено функціональні взаємозв’язки між модулями, що забезпечує масштабованість, гнучкість і стабільність роботи.

Особлива увага приділялася питанням безпеки: реалізовано багаторівневу автентифікацію, контроль доступу, шифрування за алгоритмами AES-256 і RSA, ведення журналів подій, перевірку цифрових підписів, а також захист від атак і зловживань. Система відповідає сучасним вимогам у сфері захисту персональних даних, включно з положеннями GDPR.

Суттєвим практичним результатом стала розробка плагіна до Moodle, який через API та webhook забезпечує інтеграцію з Matrix. Це дозволило поєднати можливості LMS із функціоналом безпечного обміну повідомленнями, що створює підґрунтя для побудови надійного цифрового освітнього простору.

Результати тестування підтвердили стабільну роботу системи в умовах реального використання. Було досягнуто високих показників продуктивності, точності обробки повідомлень, зручності для користувачів і адаптивності до

різних ролей від студентів до адміністраторів. Сформоване рішення використовує сучасні підходи до розробки та стандарти безпеки.

В загальному, можна сказати, що ця система є ефективним інструментом для впровадження в українських ЗВО сприяючи прозорій та контрольованій цифровій взаємодії.

Список використаних джерел

1. Moodle Developer Documentation. MoodleDocs. https://docs.moodle.org/dev/Main_Pagedocs.moodle.org+4docs.moodle.org+4 docs.moodle.org+4
2. Matrix-PHP: PHP library for Matrix. GitHub. https://github.com/artyom-poptsov/matrix-phpGitHub
3. PHP Manual: openssl_encrypt. PHP.net. https://www.php.net/manual/en/function.openssl-encrypt.phpphp.net+1php.net +1
4. Understanding GDPR Compliance for Higher Ed Institutions. Ellucian. https://www.ellucian.com/blog/understanding-gdpr-compliance-highered-institutions-2024Ellucian+1Modern Campus+1
5. GDPR Compliance for Educational Technology Providers. GDPR Advisor. https://www.gdpr-advisor.com/gdpr-compliance-for-educational-technology-providers/GDPR Advisor
6. AES-256 in PHP. SSOJet. https://ssojet.com/encryption-decryption/aes-256-in-php/ssojet.com+1Stack Overflow+1
7. Moodle App Plugins Development Guide. MoodleDev. https://moodledev.io/general/app/development/plugins-development-guidemoodledev.io
8. Client-Server API. Matrix.org. https://matrix.org/docs/api/matrix.org+3matrix.org+3GitLab+3
9. AES-256-CBC Encryption and Decryption in PHP. Medium. https://medium.com/@einnar82/aes-256-cbc-encryption-and-decryption-in-php-0449d41fa1e3DEV Community+2Medium+2YouTube+2
10. GDPR-Compliant LMS: What You Need to Know [2024]. Docebo. https://www.docebo.com/learning-network/blog/gdpr-lms/docebo.com+1Modern Campus+1
11. Matrix Client Tutorial. GitLab. https://uhoreg.gitlab.io/matrix-tutorial/GitLab

12. Machine Understandable Policies and GDPR Compliance Checking. arXiv. https://arxiv.org/abs/2001.08930arXiv
13. Why are Developers Struggling to Put GDPR into Practice when Developing Privacy-Preserving Software Systems? arXiv. https://arxiv.org/abs/2008.02987arXiv
14. Are we there yet? Understanding the challenges faced in complying with the General Data Protection Regulation (GDPR). arXiv. https://arxiv.org/abs/1808.07338arXiv
15. PHP-AES: Pure PHP implementation of the AES cipher. GitHub. https://github.com/lt/PHP-AESGitHub