БАКАЛАВРСЬКА РОБОТА

БР.КІ-22.00.00.000 ПЗ

Група КІ-21-1

Щеснюк Петро

2025

Івано-Франківський Національний Технічний Університет Нафти і Газу
Факультет Інформаційних Технологій
Кафедра комп'ютерних систем і мереж

Щеснюк Петро Васильович

УДК 004.2

БАКАЛАВРСЬКА РОБОТА

Розробка серверної частини компоненти "Методичне забезпечення дисциплін" інформаційної системи ІФНТУНГ

Комп'ютерна інженерія
(назва освітньої програми)

123 - Комп'ютерна інженерія
(шифр і назва спеціальності)

Робота містить результати власних досліджень, використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело:

Здобувач освітнього ступеня Щеснюк П. В.
(підпис, прізвище та ініціали здобувача)

Науковий керівник Мойсеєнко О. В., к. т. н., доцент
(підпис, прізвище, ім'я, по батькові, науковий ступінь, вчене звання керівника)

Допущено до захисту

Завідувач кафедри КСМ

д.т.н., професор Мельничук С. І.
(посада) (підпис) (дата) (прізвище та ініціали)

Івано-Франківськ – 2025

Івано-Франківський Національний Технічний Університет Нафти і Газу

Факультет Інформаційних Технологій

Кафедра Комп'ютерних систем і мереж

Освітній рівень Бакалавр

Спеціальність 123 Комп’ютерна інженерія

ЗАТВЕРДЖУЮ

Завідувач кафедри КСМ

Мельничук С. І.

"05" травня 2025 року

З А В Д А Н Н Я

НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

Щеснюку Петру Васильовичу

(прізвище, ім’я, по батькові)

1. Тема роботи Розробка серверної частини компоненти “Методичне забезпечення дисциплін” інформаційної системи ІФНТУНГ

керівник роботи Мойсеєнко Олена Володимирівна, к. т. н., доцент

затверджені наказом закладу вищої освіти від " 05 " травня 2025 року № 275/7

2. Термін подання студентом роботи 12 червня 2025 року

3. Вихідні дані до роботи Методичні вказівки, технічна література

4. Зміст пояснювальної записки: 1 Аналіз предметної галузі та постановка задачі. 2 Розробка структури бази даних і серверної частини компоненти «методичне забезпечення дисциплін». 3 Розробка програмного забезпечення.

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

6. Консультанти розділів роботи

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

7. Дата видачі завдання – 29 січня 2025 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Термін виконання етапів роботи	Примітка
1	Підготовка до виконання бакалаврської роботи, вибір літератури	24.02.2025 – 07.03.2025	виконав
2	Пошук наявних прикладів вирішення схожих задач, опрацювання предметної області	08.03.2025 – 16.03.2025	виконав
3	Розробка структури бази даних, структури серверної частини та діаграм	17.03.2025 – 05.04.2025	виконав
4	Розробка коду для компоненти серверної частини та перевірка його роботи	06.04.2025 – 30.04.2025	виконав
5	Підготовка остаточного варіанту та оформлення пояснювальної записки	01.05.2025 – 30.05.2025	виконав
6	Підготовка до здачі та здача бакалаврської роботи	31.05.2025 – 10.06.2025	виконав

Студент _____ Щеснюк П. В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Мойсеєнко О. В.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Бакалаврська робота присвячена розробці компоненти “Методичне забезпечення дисциплін” серверної частини інформаційної системи університету. Актуальність теми обумовлена необхідністю вдосконалення доступу до навчальних матеріалів та покращення організації освітнього процесу.

Основним завданням є створення функціоналу для управління методичними матеріалами, організація їх зберігання та забезпечення зручного доступу до них. Для вирішення поставленої задачі використовується чистий PHP для реалізації серверної частини, який продукує дані у форматі JSON та підтримує CRUD-операції за допомогою методів POST, GET, PUT і DELETE.

У процесі роботи розроблено структуру бази даних, діаграму послідовностей, use-case діаграму та структуру серверної частини компоненти, реалізовано API для управління методичними матеріалами, а також впроваджено авторизацію за допомогою токенів для забезпечення безпеки доступу. Проведено тестування компонентів системи для перевірки їх коректності та надійності.

Результатом роботи є функціональний серверний модуль, який забезпечує надійне зберігання методичних матеріалів і зручний доступ до них для студентів та викладачів. Висновки підтверджують ефективність реалізованої системи та її відповідність вимогам сучасних інформаційних систем в освіті.

Ключові слова: ІНФОРМАЦІЙНА СИСТЕМА, МЕТОДИЧНЕ ЗАБЕЗПЕЧЕННЯ, СЕРВЕРНА ЧАСТИНА, PHP, JSON, CRUD, АВТОРИЗАЦІЯ, ТОКЕН, БАЗА ДАНИХ.

ANNOTATION

The bachelor's thesis is dedicated to the development of the “Methodological Support for Disciplines” component of the server-side of the university's information system. The relevance of the topic is due to the need to improve access to educational materials and enhance the organization of the educational process.

The main objective is to create functionality for managing methodological materials, organizing their storage, and ensuring convenient access to them. To achieve this, pure PHP is used for the server-side implementation, which produces data in JSON format and supports CRUD operations using POST, GET, PUT, and DELETE methods.

During the course of the work, the database schema, sequence diagram, use-case diagram, and the server-side architecture of the component were developed. An API for managing methodical materials was implemented, and token-based authorization was introduced to ensure secure access. The components of the system were tested to verify their correctness and reliability.

The result of the work is a functional server module that ensures reliable storage of methodological materials and convenient access to them for students and teachers. The conclusions confirm the effectiveness of the implemented system and its compliance with the requirements of modern information systems in education.

Keywords: INFORMATION SYSTEM, METHODOLOGICAL SUPPORT, SERVER-SIDE, PHP, JSON, CRUD, AUTHORIZATION, TOKEN, DATABASE.

ЗМІСТ

ВСТУП

Актуальність обраної теми. Сучасний освітній процес вимагає ефективної організації доступу до методичних матеріалів, які є основою якісного навчання. Університетські інформаційні системи повинні забезпечувати зручний доступ до навчальних ресурсів для студентів та викладачів, а також ефективне управління методичним забезпеченням дисциплін. Проте багато існуючих рішень мають обмежену функціональність або не забезпечують належного рівня безпеки та зручності використання. Це створює потребу у розробці сучасних серверних компонентів, які дозволяють управляти методичними матеріалами та організовувати їх зберігання.

Об’єкт та предмет дослідження. Об'єктом дослідження є процес обліку методичного забезпечення дисциплін. Предметом дослідження є методи та засоби реалізації серверної частини компоненти “Методичне забезпечення дисциплін” на основі чистого PHP з використанням JSON-формату даних та CRUD-операцій, а також забезпечення безпеки доступу за допомогою токенів.

Мета та завдання роботи. Метою роботи є розробка серверної частини компоненти “Методичне забезпечення дисциплін” інформаційної системи університету, яка забезпечить ефективне управління методичними матеріалами та їх зберігання. Для досягнення мети необхідно виконати такі завдання:

– розробити структуру бази даних для зберігання методичних матеріалів;

– реалізувати API для управління методичними матеріалами за допомогою CRUD-операцій (POST, GET, PUT, DELETE);

– забезпечити передачу даних у форматі JSON для інтеграції з клієнтськими додатками;

– впровадити авторизацію за допомогою токенів для забезпечення безпеки доступу;

– провести тестування компонентів системи для перевірки їх коректності та надійності.

Методи дослідження. У роботі застосовуються такі методи дослідження:

– аналіз літературних джерел – для вивчення сучасних підходів до розробки серверних частин інформаційних систем та авторизації за допомогою токенів;

– проєктування бази даних – для розробки структури зберігання методичних матеріалів;

– методи об’єктно-орієнтованого програмування – для реалізації логіки серверної частини на чистому PHP;

– перевірка працездатності програмного забезпечення – для оцінки функціональності системи та її компонентів.

Практичне значення отриманих результатів. Результати роботи використані як компонент єдиної інформаційної системи університету, що дозволить підвищити ефективність управління навчальними матеріалами та покращити доступ до них для студентів та викладачів. Практичне значення підтверджене відповідним актом про впровадження.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Поняття навчально-методичного комплексу дисципліни

Навчально-методичний комплекс дисципліни (НМКД) – це систематизований набір нормативних, організаційних та навчально-методичних матеріалів, які забезпечують ефективне викладання, вивчення та контроль знань студентів у межах конкретної навчальної дисципліни. НМКД слугує фундаментом для реалізації робочої програми навчального курсу, сприяє підвищенню якості навчального процесу та полегшує доступ студентів і викладачів до необхідних освітніх ресурсів [1].

Основні функції НМКД:

– організаційна: забезпечення чіткого планування навчального процесу;

– навчальна: підтримка викладання теоретичного матеріалу та практичних занять;

– методична: надання рекомендацій щодо виконання завдань і підготовки до контрольних заходів;

– контрольна: створення умов для об’єктивної перевірки знань студентів.

Структура НМКД:

– робоча програма навчальної дисципліни – офіційний документ, що визначає цілі, завдання, зміст, види та методи навчальної діяльності;

– конспекти лекцій – систематизовані матеріали для викладання теоретичних аспектів дисципліни;

– методичні вказівки для проведення лабораторних, практичних та семінарських занять – детальні інструкції для організації активної навчальної діяльності;

– тематика курсових робіт і методичні рекомендації – орієнтири для самостійної роботи студентів над проектами, що розвивають дослідницькі навички;

– матеріали для самостійної роботи – завдання, графіки та рекомендації для індивідуального опрацювання навчального матеріалу;

– індивідуальні завдання – спеціально розроблені завдання для перевірки глибини знань та рівня їх засвоєння;

– засоби діагностики знань – тести, екзаменаційні білети та інші інструменти для контролю успішності.

Вимоги до НМКД:

– матеріали повинні бути розроблені державною мовою, відповідно до науково-методичних стандартів;

– НМКД проходить обов’язкову експертизу методичними комісіями навчального закладу;

– наявність якісно підготовленого НМКД є критерієм для акредитації навчальних програм.

1.2 Огляд нормативної та технічної документації

Для забезпечення процесу розробки компоненти "Методичне забезпечення дисциплін" серверної частини інформаційної системи університету необхідно врахувати нормативну базу та технічні вимоги, які регламентують створення, збереження та управління навчальними матеріалами у закладах вищої освіти.

Нормативно-правова база**:**

– Закон України "Про вищу освіту" – визначає основні засади організації освітнього процесу, акцентує увагу на важливості електронного документообігу та збереження методичних матеріалів [2];

– Закон України "Про захист персональних даних" – встановлює вимоги до обробки персональних даних, що важливо для роботи із студентськими базами та викладацькою інформацією [3];

– державний стандарт України ДСТУ ISO/IEC 27001 – надає стандарти для забезпечення інформаційної безпеки в освітніх інформаційних системах [4];

– рекомендації МОН України щодо електронного документообігу – визначають особливості роботи з електронними освітніми ресурсами [5].

Крім того, система має виконувати функцію формування звітів з метою обліку та верифікації показників, пов’язаних з навчально-методичною літературою, а саме:

– звіту для рейтингування науково-педагогічних працівників ІФНТУНГ згідно Положення ІФНТУНГ про рейтинг;

– звіту по навчально-методичному забезпеченню дисциплін, що викладаються;

– показниках науково-педагогічних працівників щодо ліцензійних умов провадження освітньої діяльності.

Технічна документація:

– вимоги до програмного забезпечення інформаційних систем навчальних закладів – описують функціональні та нефункціональні характеристики ІТ-систем;

– методичні рекомендації з розробки веб-орієнтованих інформаційних систем – регламентують підходи до створення сучасних веб-додатків;

– стандарти обміну інформацією між освітніми платформами – забезпечують інтеграцію різних систем та платформ.

1.3 Аналіз існуючих рішень

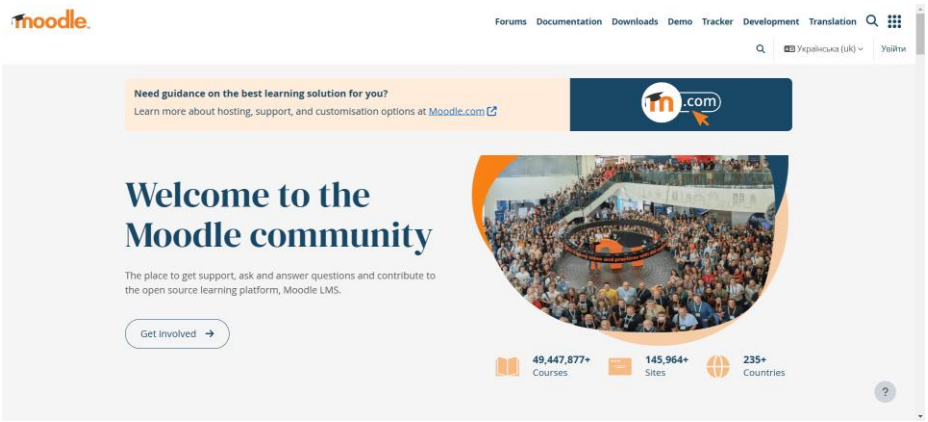

Рисунок 1.1 – Moodle

На ринку програмного забезпечення існують різні платформи, які забезпечують управління навчальними матеріалами та методичним забезпеченням дисциплін.

Moodle (рис. 1.1) – відкрита система управління навчальним процесом (Learning Management System, LMS), яка підтримує зберігання методичних матеріалів, інтеграцію з іншими сервісами та забезпечує доступ до ресурсів для студентів і викладачів [6].

Переваги: гнучкість налаштувань, велика спільнота користувачів, можливість адаптації до потреб закладу.

Недоліки: потребує високої кваліфікації для адміністрування, значне споживання серверних ресурсів.

Anthology (раніше Blackboard) (рис. 1.2) – потужна комерційна платформа для організації навчального процесу [7].

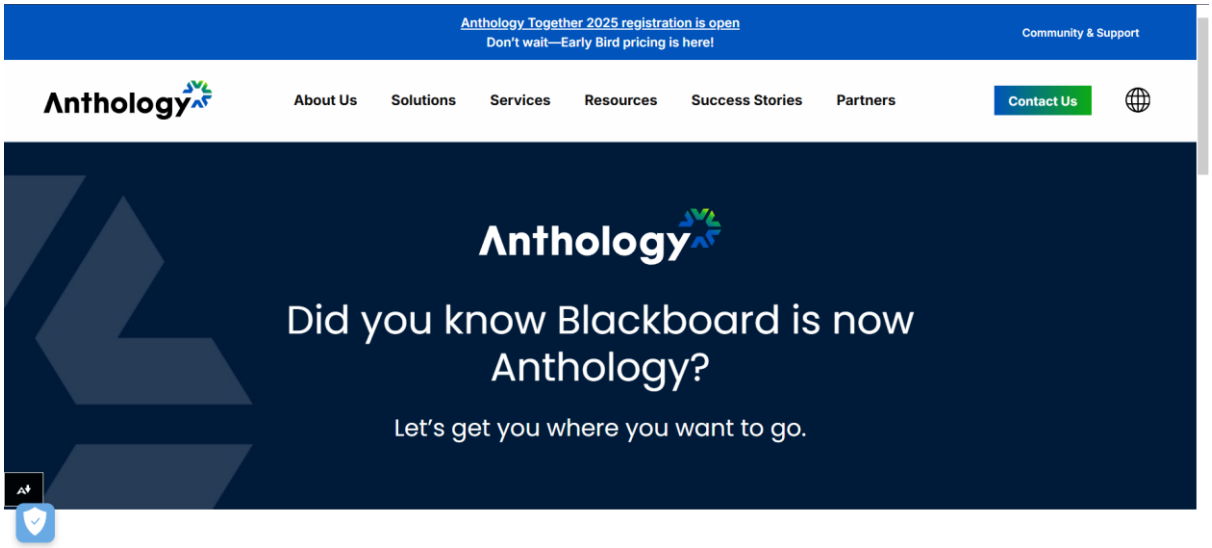

Рисунок 1.2 – Anthology

Переваги: висока якість технічної підтримки, інтеграція з багатьма освітніми сервісами.

Недоліки: висока вартість ліцензій, складність адаптації до локальних освітніх вимог.

Google Classroom (рис. 1.3) – хмарна платформа для організації дистанційного навчання [8].

Переваги: проста інтеграція з екосистемою Google, зручний інтерфейс.

Недоліки: обмеження в налаштуваннях функціоналу, залежність від стабільності зовнішніх серверів.

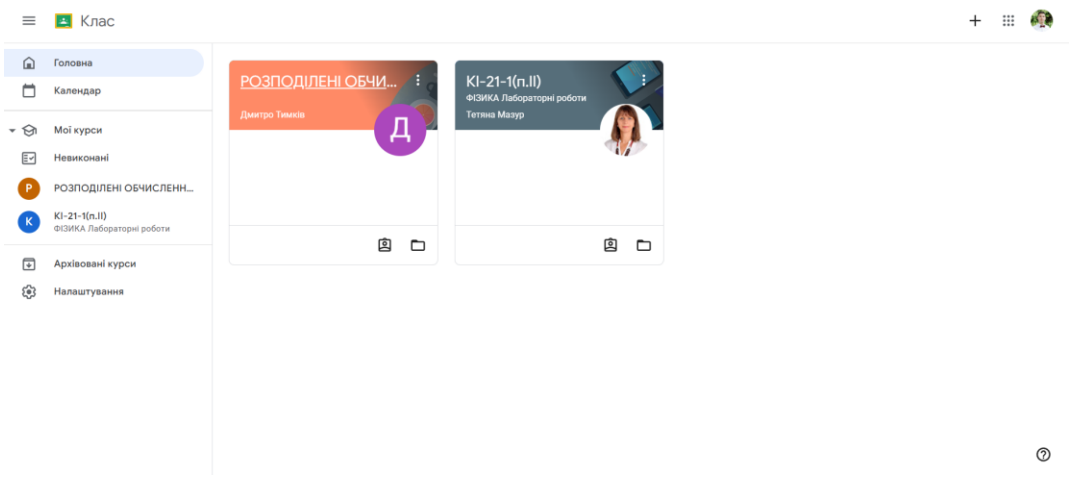

Рисунок 1.3 – Google Classroom

Для кращого розуміння можливостей та обмежень існуючих інформаційних систем, які використовуються для управління навчальними матеріалами, було проведено порівняння Moodle, Anthology та Google Classroom за основними характеристиками:

Таблиця 1.1 – Порівняльна таблиця характеристик платформ

Характеристика	**Назва інформаційної системи**		
	Moodle	**Anthology (Blackboard)**	**Google Classroom**
Функціонал	Широкий набір модулів для управління курсами, підтримка SCORM, налаштування ролей користувачів	Підтримка змішаного навчання, інтеграція з CRM-системами, аналітика навчального процесу	Основні функції для організації завдань, комунікація в групах, інтеграція з Google Workspace
Вартість	Безкоштовна, з відкритим кодом, але потребує витрат на адміністрування	Висока вартість ліцензій, залежність від кількості користувачів	Безкоштовна, але є обмеження на обсяг хмарного сховища

Кінець таблиці 1.1

Безпека	Високий рівень безпеки за рахунок оновлень та підтримки SSL, налаштування прав доступу	Розширені налаштування безпеки, відповідність стандартам FERPA	Залежність від політики безпеки Google, обмежені можливості контролю даних
Масштабованість	Вимагає потужних серверів, підтримка хмарних рішень	Легка масштабованість завдяки хмарній архітектурі	Масштабованість обмежена можливостями облікового запису Google
Кастомізація	Висока гнучкість налаштувань, можливість розробки власних плагінів	Обмежена можливість налаштування інтерфейсу, потребує експертних знань	Мінімальні можливості налаштування дизайну та функціоналу

Проблеми та недоліки існуючих рішень

Хоча аналізовані платформи мають широкий функціонал, вони мають певні обмеження, які ускладнюють їх впровадження у специфічних умовах університету.

Moodle**:**

– потребує високої кваліфікації адміністраторів та програмістів для налаштування і підтримки системи;

– значне споживання серверних ресурсів, що потребує додаткових інвестицій у технічне забезпечення;

– обмеження у налаштуванні інтеграції з внутрішніми інформаційними системами університету.

Anthology (Blackboard):

– висока вартість ліцензій робить цю платформу недоцільною для впровадження в бюджетних освітніх установах;

– складність адаптації до локальних освітніх стандартів та вимог;

– закритий код ускладнює розширення функціоналу відповідно до специфічних потреб закладу.

Google Classroom:

– обмежений функціонал для управління навчальними матеріалами та методичними ресурсами;

– залежність від стабільності зовнішніх серверів та політики конфіденційності Google;

– відсутність можливостей для гнучкого налаштування прав доступу до навчальних матеріалів.

Ці обмеження унеможливлюють адаптацію цих платформ до специфічних вимог університету, зокрема до:

– централізованого управління методичними матеріалами з урахуванням ролей користувачів;

– інтеграції з внутрішньою інформаційною системою університету;

– підтримки локальних освітніх стандартів та зручного адміністрування.

Аналіз обліку методичних матеріалів на прикладі університетських сайтів

Для виокремлення типових підходів до «обліку методичного забезпечення» було досліджено розділи «Навчально-методичне забезпечення» на сайтах провідних ЗВО України.

КПІ ім. Ігоря Сікорського:

– каталог друкованих видань та методичних рекомендацій, розбитий за кафедрами та спеціальностями [9];

– для кожного документа наведено бібліографічний опис, анотацію та посилання на електронний ресурс.

Економічний факультет КНУ ім. Т. Шевченка:

– перелік робочих програм дисциплін, у яких окремо перелічено підручники, посібники та методичні рекомендації з деталізацією авторів, року видання та форми доступу.

Аналіз існуючих методів та засобів

У процесі розробки подібних компонент використовуються різноманітні алгоритмічні, програмні та апаратні засоби:

Алгоритмічні методи:

– пошук і фільтрація навчальних матеріалів;

– обробка великих обсягів даних для аналітики навчального процесу.

Програмні засоби:

– використання баз даних (наприклад, MySQL, PostgreSQL) для централізованого зберігання даних;

– розробка серверної логіки з використанням мов програмування (наприклад, PHP, Python, JavaScript).

Апаратні засоби:

– використання локальних серверів для перевірки та впровадження програмного забезпечення;

– хмарні сервіси для масштабованості та забезпечення безперервного доступу до системи.

Інституційні репозитарії як окрема модель зберігання освітніх ресурсів

У сучасному освітньому середовищі інституційні репозитарії набувають все більшої важливості як платформи для збереження, організації та поширення цифрових ресурсів, створених співтовариством закладу освіти. Інституційний репозитарій є спеціалізованою цифровою системою, що забезпечує збирання, збереження та надання доступу до наукових та методичних матеріалів, таких як наукові статті, дисертації, підручники, посібники, презентації, а також відкриті освітні ресурси [10] [11].

Основними перевагами інституційних репозитаріїв є:

– відкритий доступ до знань: репозитарії сприяють ширшому розповсюдженню академічного контенту, забезпечуючи доступ до матеріалів внутрішнім та зовнішнім користувачам;

– підвищення видимості досліджень: завдяки централізованому зберіганню даних академічна спільнота може легко здійснювати пошук і аналіз наукових робіт, що підвищує якість академічного обміну;

– довгострокове збереження інформації: інституційні репозитарії відіграють важливу роль у збереженні цифрових ресурсів, що є критично важливим для архівування результатів досліджень та історичних даних закладу.

Такі репозитарії часто використовують стандартизовані протоколи, наприклад, OAI-PMH, що дозволяє забезпечити сумісність з іншими інформаційними системами та сприяти інтеграції даних у єдині цифрові екосистеми. На практиці це означає, що репозитарій може бути інтегрований з платформами дистанційного навчання або з іншими системами управління науковими даними, що дає можливість автоматично оновлювати та синхронізувати інформацію.

Проте, незважаючи на свій потенціал, інституційні репозитарії мають і певні недоліки:

– витрати на впровадження і супровід: забезпечення функціонування репозитарію вимагає значних фінансових і людських ресурсів, що може стати перешкодою для бюджетних освітніх установ;

– інтеграційні проблеми: університетські інформаційні системи часто розробляються без урахування специфічних вимог репозитаріїв, що може ускладнювати їх сумісність і переналаштування існуючих систем;

– висока кваліфікація персоналу: ефективне адміністрування репозитарію вимагає спеціалізованих знань у сфері інформаційних технологій та бібліотечної справи, що не завжди доступно у всіх навчальних закладах.

Таким чином, аналізуючи існуючі підходи до управління навчальними та методичними матеріалами, можна стверджувати, що інституційні репозитарії представляють собою важливий компонент сучасних інформаційних систем. Вони не лише забезпечують централізоване зберігання та ефективне адміністрування цифрових ресурсів, але й сприяють розвитку відкритих освітніх практик та інтеграції з іншими платформами. При цьому необхідно враховувати як їхні суттєві переваги, так і виклики, пов’язані з впровадженням та експлуатацією таких систем у специфічних умовах кожного закладу.

1.4 Постановка задачі

Вимоги до інформаційної системи

1. Функціональні можливості:

– реалізація операцій створення, читання, оновлення та видалення (CRUD) для роботи з інформацією про методичні матеріали дисциплін;

– оптимізація запитів до бази даних для забезпечення високої швидкодії системи;

– наявність додаткових методів для специфічних вибірок даних: наприклад, пошук матеріалів за дисципліною або за типом забезпечення.

2. База даних:

– структура таблиць має враховувати всі необхідні атрибути для зберігання детальної інформації про методичні матеріали (ідентифікатор, назва, тип, посилання, опис, рік видання, список авторів тощо);

– таблиці повинні бути наповнені тестовими даними, що дає змогу перевірити коректність роботи системи.

3. Структура серверної частини:

– шляхи (routes): забезпечують правильну маршрутизацію HTTP-запитів до відповідних контролерів і гарантовано коректне відображення інформації;

– моделі (models): відображають схему бази даних у вигляді програмних об’єктів для зручної маніпуляції даними;

– контролери (controllers): отримують запити від користувача, взаємодіють із моделями та повертають оброблену інформацію;

– сервіси (services): містять усі необхідні запити до бази даних, реалізують операції CRUD та формують додаткові дані за потребою.

4. Перевірка працездатності системи:

– локальна перевірка: Оцінка відображення інформації через браузер із використанням локального сервера;

– перевірка HTTP-запитів: Використання платформи Postman для

підтвердження коректної роботи всіх операцій CRUD.

План реалізації вимог

На основі вище зазначених вимог планується розробити серверну компоненту «Методичне забезпечення дисциплін» для інформаційної системи університету, що сприятиме централізованому управлінню навчальними ресурсами. Для реалізації завдань необхідно:

– створити структуру таблиць бази даних: розробити схему, яка охоплює всі необхідні атрибути для зберігання детальної інформації про методичні матеріали (ідентифікатор, назва, тип, посилання, опис, рік видання, список авторів тощо);

– розробити програмний код для взаємодії з базою даних: написати код, що забезпечує реалізацію операцій CRUD для обліку методичних матеріалів;

– розробити додаткові функції: імплементувати специфічні методи для унікальних вибірок даних, таких як пошук матеріалів за дисципліною або за типом забезпечення відповідно до вимог інформаційної системи;

– створити серверну архітектуру: сформувати наступні основні компоненти серверної частини:

– шляхи (routes): організувати маршрутизацію HTTP-запитів до відповідних контролерів;

– моделі (models): відтворити структурну модель бази даних у вигляді об’єктів для зручної обробки даних;

– контролери (controllers): обробити запити, взаємодіяти із моделями та повернути оброблену інформацію користувачу;

– сервіси (services): інкапсулювати всі запити до бази даних, реалізувати функції CRUD та сформувати додаткові дані залежно від дій користувача.

– провести фінальну перевірку працездатності:

– виконати локальну перевірку відображення даних через браузер;

– перевірити коректність функціонування CRUD-операцій із використанням платформи Postman.

Таким чином, реалізація поставлених завдань дозволить створити інтегровану та функціональну серверну частину інформаційної системи університету, що забезпечить централізований доступ та ефективне управління методичним забезпеченням дисциплін.

2 РОЗРОБКА СТРУКТУРИ БАЗИ ДАНИХ І СЕРВЕРНОЇ ЧАСТИНИ КОМПОНЕНТИ «МЕТОДИЧНЕ ЗАБЕЗПЕЧЕННЯ ДИСЦИПЛІН»

2.1 Вимоги до програмного забезпечення

Для розробки серверної частини компоненти «Методичне забезпечення дисциплін» інформаційної системи університету було прийнято рішення використовувати технології, що вже застосовуються у існуючих інформаційних системах закладу. Оскільки основна частина проекту університетського сайту реалізована на PHP, розробка нової компоненти побудована на цій мові програмування дозволяє забезпечити єдину архітектуру, спрощує інтеграцію з внутрішніми системами та сприяє повторному використанню коду.

Для реалізації серверної логіки компоненти «Методичне забезпечення дисциплін» було обрано мову програмування PHP. Її переваги полягають у простоті синтаксису, широкій підтримці популярних веб-серверів (Apache, Nginx) та баз даних (MySQL, PostgreSQL), високій продуктивності обробки HTTP-запитів та активній спільноті розробників, що гарантує постійні оновлення та безпеку коду [12].

При розробці програмного забезпечення обрано редактор Visual Studio Code. Цей інструмент є безкоштовним, кросплатформним та має зручний інтерфейс із широкою підтримкою плагінів для PHP, що сприяє підвищенню продуктивності розробки та інтеграції з системами контролю версій, зокрема Git [13].

Для локального тестування та адміністрування баз даних використано збірку XAMPP [14]. Її основні переваги – кросплатформеність, простота встановлення та налаштування, доступність і безкоштовність, а також зручна панель керування для запуску Apache, MySQL, PHP та phpMyAdmin. Це дає

змогу швидко налаштувати середовище розробки, що є особливо важливим для бюджетних освітніх проектів.

Структуру серверної частини розробленої компоненти можна розбити на кілька ключових елементів: база даних, моделі, контролери, сервіси та маршрути. База даних включає таблиці для зберігання інформації про методичні матеріали дисциплін із визначеними атрибутами (ідентифікатори, назви, типи, посилання, описи, роки видання та списки авторів). Моделі відображають ці таблиці у вигляді об’єктів, забезпечуючи зручну маніпуляцію даними. Контролери отримують HTTP-запити від користувачів, направляють їх до сервісного шару, який забезпечує виконання операцій CRUD із оптимізованими SQL-запитами. Правильна маршрутизація запитів гарантує ефективну інтеграцію між користувацьким інтерфейсом та серверною логікою.

Таким чином, обрані програмні засоби, середовище розробки та локальний сервер створюють оптимальні умови для реалізації серверної частини компоненти «Методичне забезпечення дисциплін». Ця система забезпечує зберігання, редагування, видалення та перегляд методичних матеріалів, що сприяє інтеграції з інформаційною системою університету та підвищенню ефективності освітніх процесів.

2.2 Структура бази даних

Для створення бази даних серверної компоненти «Методичне забезпечення дисциплін» необхідно спочатку визначити її логічну структуру та вміст, виходячи із функціональних вимог системи. Згідно з технічним завданням, для зберігання, обліку та взаємодії з інформацією про методичні матеріали дисциплін розроблено наступні таблиці:

Таблиця 2.1 містить інформацію про типи методичних забезпечень, що використовуються в університеті, наприклад: конспект лекцій, електронний курс, посібник тощо.

Таблиця 2.1 – LearningResourceType

Ключове поле	Назва стовпця	Тип даних	Довжина	Дозволяє NULL	Вміст
+	id	int	11		Унікальний ідентифікатор типу
-	name	text	65 535		Назва типу методичного забезпечення

Таблиця 2.2 зберігає інформацію про самі методичні матеріали. Вона містить посилання на ресурс, його опис, рік створення та інші параметри.

Таблиця 2.2 – LearningResource

Ключове поле	Назва стовпця	Тип даних	Довжина	Дозволяє NULL	Вміст
+	id	int	11		Унікальний ідентифікатор ресурсу
-	resourcetype_id	int	11		Зовнішній ключ, що посилається на LearningResourceType.id
-	resourcelink	text	65 535		Посилання на методичний матеріал
-	resourcedescription	text	65 535	+	Опис змісту матеріалу
-	resourceyear	int	11		Рік створення або оновлення ресурсу

Таблиця 2.3 забезпечує зв’язок між методичними ресурсами та їх авторами. Вона потрібна для збереження інформації про співучасть різних авторів у створенні методичних матеріалів.

Таблиця 2.3 – LearningResourceAuthor

Ключове поле	Назва стовпця	Тип даних	Довжина	Дозволяє NULL	Вміст
+	id	int	11		Унікальний ідентифікатор запису
-	resource_id	int	11		Зовнішній ключ, що посилається на LearningResource.id
-	author_id	int	11		Ідентифікатор автора (посилання на запис у таблиці User)

Таблиця 2.4 встановлює зв’язок між методичними матеріалами та конкретними дисциплінами. Таким чином, вона дозволяє прив’язувати ресурси до відповідних курсів.

Таблиця 2.4 – CourseResource

Ключове поле	Назва стовпця	Тип даних	Довжина	Дозволяє NULL	Вміст
+	id	int	11		Унікальний ідентифікатор запису
-	resource_id	int	11		Зовнішній ключ, що посилається на LearningResource.id
-	course_id	int	11		Зовнішній ключ, що посилається на запис про дисципліну (Course.id)
-	courseresourcetype_id	int	11		Зовнішній ключ, що посилається на CourseResourceType.id

Таблиця 2.5 описує типи забезпечення для дисциплін, наприклад: основна література, додаткова література. Вона дозволяє класифікувати методичні матеріали за їх призначенням.

Таблиця 2.5 – CourseResourceType

Ключове поле	Назва стовпця	Тип даних	Довжина	Дозволяє NULL	Вміст
+	id	int	11		Унікальний ідентифікатор типу
-	name	text	65 535		Назва типу забезпечення дисципліни

2.3 Взаємозв’язки між таблицями

Визначимо зв’язки між таблицями та створимо структурну схему бази даних (рис. 2.1).

Зв’язки між таблицями встановлюються за допомогою ключових полів, які містять спеціальні значення для ідентифікації записів в різних таблицях та дозволяють об’єднати їх для формування єдиної інформаційної моделі.

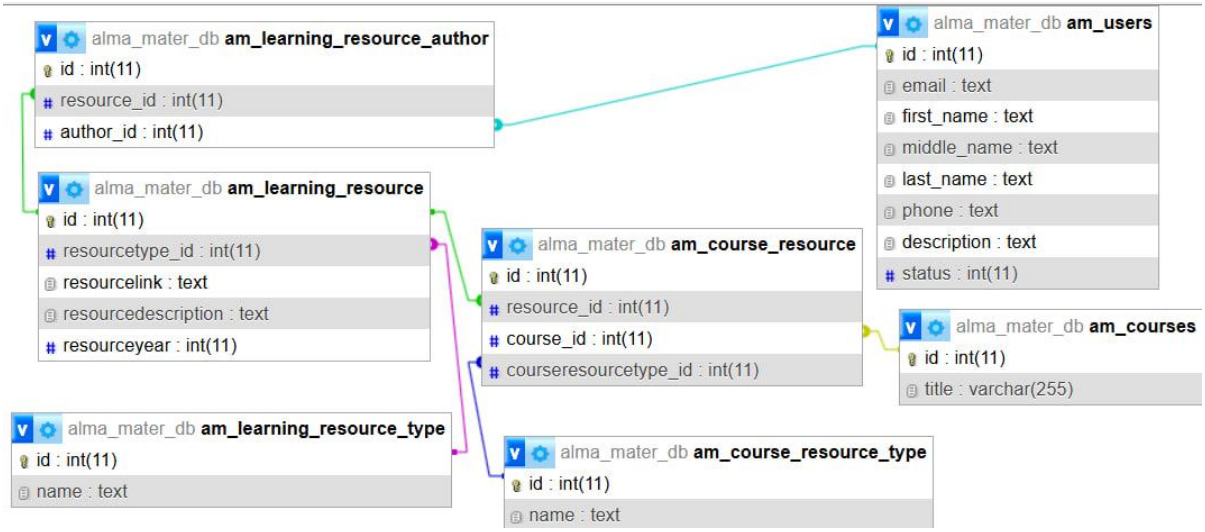

Рисунок 2.1 – Структура бази даних

Первинний ключ – це одне або кілька полів (стовпців), комбінація значень яких однозначно визначає кожний запис у таблиці.

Зовнішній (вторинний) ключ – це одне або кілька полів (стовпців) у таблиці, що містять посилання на поле або поля первинного ключа в іншій таблиці. Зовнішній ключ визначає спосіб об'єднання таблиць [15].

Крім того, структура передбачає встановлення зв'язків із таблицями з інших компонент, зокрема:

– таблиця Course з компоненти навчальні дисципліни;

– таблиця User з компоненти користувачі та підрозділи.

Між таблицями LearningResourceType та LearningResource встановлюється зв'язок «один до багатьох»: кожен запис у таблиці LearningResourceType (за унікальним полем id) може відповідати багатьом записам у таблиці LearningResource, де поле resourcetype_id містить значення, що відповідає id з таблиці LearningResourceType. Це дозволяє класифікувати методичні матеріали за типами (наприклад, конспект, електронний курс, посібник). При цьому, для одного конкретного типу методичного забезпечення може існувати велика кількість записів (методичних матеріалів), але кожен окремий запис методичного забезпечення може мати лише один конкретний тип.

Між таблицями LearningResource та LearningResourceAuthor утворюється зв’язок «один до багатьох»: кожен методичний ресурс може бути пов’язаний із кількома авторами. При цьому поле resource_id у таблиці LearningResourceAuthor посилається на id таблиці LearningResource, а поле author_id являє собою ідентифікатор автора та є зовнішнім ключем, що посилається на запис у таблиці User, де міститься інформація про користувача (автора).

Окрім цього, таблиця LearningResourceAuthor виконана як зв'язна для таблиць LearningResource і User, оскільки кожен автор може бути автором кількох методичних забезпечень, а кожне методичне забезпечення може бути написаним кількома авторами. Це дозволяє гнучко відображати взаємозв’язки між методичними матеріалами та їхніми авторами в системі.

Між таблицями LearningResource та CourseResource встановлюється зв’язок «один до багатьох»: один методичний ресурс може бути прив’язаний до декількох дисциплін. Для цього поле resource_id таблиці CourseResource містить значення id з таблиці LearningResource, що дозволяє визначати, які ресурси використовуються для конкретних курсів.

Між таблицями CourseResource та CourseResourceType встановлюється зв’язок «один до багатьох»: кожен тип забезпечення для дисциплін (який зберігається в таблиці CourseResourceType) може бути пов’язаний з багатьма записами у таблиці CourseResource. При цьому поле courseresourcetype_id таблиці CourseResource посилається на id таблиці CourseResourceType, що дає можливість класифікувати ресурси за їх функціональним призначенням (наприклад, основна література, додаткова література).

Між таблицями CourseResource та Course (дисциплінами) встановлюється зв’язок «один до багатьох»: поле course_id у таблиці CourseResource є зовнішнім ключем, що посилається на запис про дисципліну (Course.id). Це дозволяє ідентифікувати, до якої дисципліни належить певний методичний ресурс.

Крім цього, таблиця CourseResource виконує роль зв'язної між таблицями LearningResource і Course: кожне методичне забезпечення може бути прив’язане

до декількох дисциплін, а кожна дисципліна може мати кілька методичних ресурсів. Така організація даних забезпечує гнучку класифікацію та багаторазове використання одного ресурсу для різних курсів.

Ці взаємозв'язки забезпечують оптимальне зберігання та організацію даних, сприяють цілісності інформації та ефективній обробці методичних матеріалів дисциплін. Реалізація зв'язків типу «один до багатьох» гарантує, що зміни в одній таблиці автоматично відображатимуться у всій системі, забезпечуючи інтегрованість даних.

2.4 Створення таблиць бази даних

Для створення таблиць у розроблюваній компоненті я використовую СУБД MySQL. MySQL – це вільна система керування реляційними базами даних, яка була розроблена для підвищення швидкодії обробки великих обсягів даних та є альтернативою комерційним СКБД [16]. Для роботи з базами даних у середовищі локального сервера використовую phpMyAdmin, що входить до складу XAMPP. phpMyAdmin – це зручний веб-інтерфейс, який дозволяє створювати, моделювати та адмініструвати бази даних MySQL без необхідності використання окремих клієнтів, таких як MySQL Workbench.

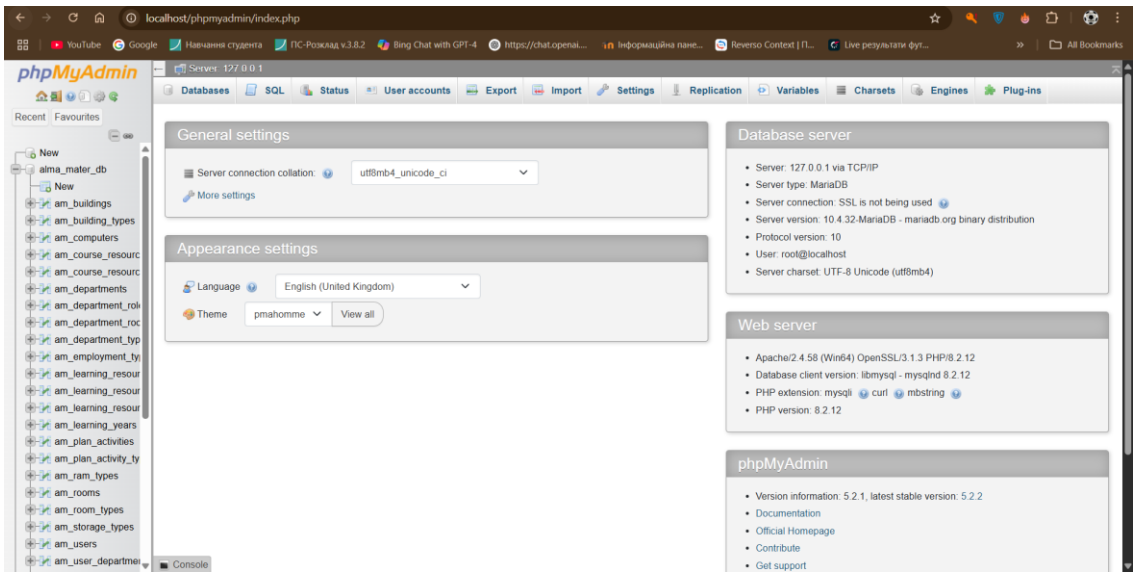

Рисунок 2.2 – Робоче середовище phpMyAdmin

На рисунку 2.2 представлено інтерфейс phpMyAdmin, що забезпечує доступ до широкого функціоналу для роботи з базами даних: написання SQL-запитів, імпорт/експорт даних та виконання інших стандартних операцій адміністрування. База даних університетської інформаційної системи, яку ми використовуємо, має назву alma_mater_db і вже містить безліч таблиць, що відповідають різним підсистемам. Завдяки використанню dump-файлу, база була успішно імпортована на локальний сервер, що значно спрощує подальшу роботу.

Оскільки база alma_mater_db вже існує, всі нові таблиці створюються безпосередньо в рамках цієї бази. Для компоненти «Методичне забезпечення дисциплін» використовується сталий шаблон іменування: імена таблиць починаються з префікса am_, що допомагає однозначно віднести їх до даної системи, при цьому всі символи представлені малими літерами, а окремі слова відокремлюються символом підкреслення. Такий підхід сприяє уніфікації назв і спрощує написання SQL-запитів для розробників.

Використання вкладки SQL у phpMyAdmin дозволяє запускати потрібні запити безпосередньо на сервері. Сам процес створення таблиць здійснюється через виконання SQL-команди CREATE TABLE.

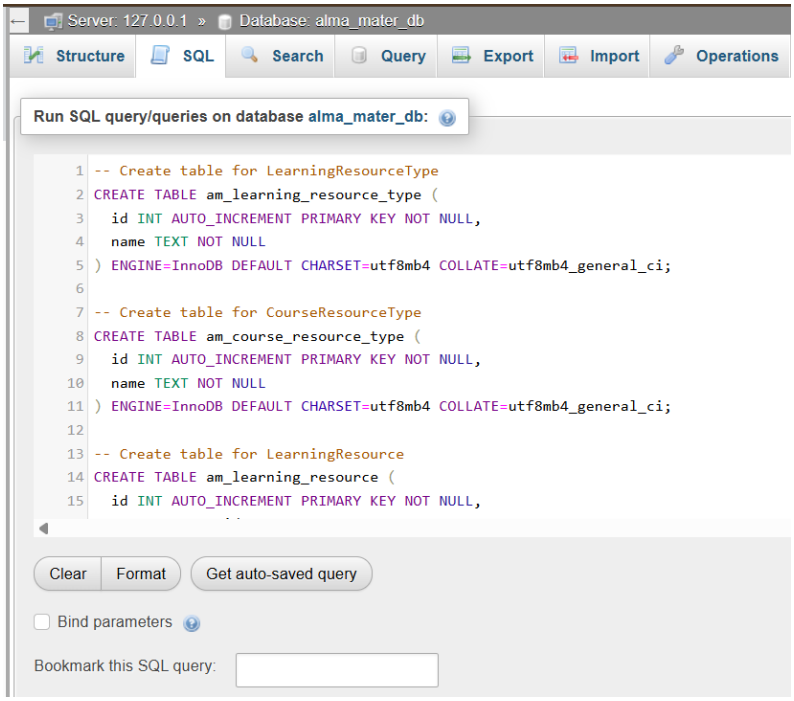

Рисунок 2.3 – Виконання запитів CREATE TABLE

На рисунку 2.3 відображено приклади SQL-запитів для створення таблиць. Важливо зауважити, що перед виконанням команд необхідно попередньо вибрати конкретну базу даних (у нашому випадку – alma_mater_db), інакше phpMyAdmin не знатиме, куди відносити створювані таблиці, що може призвести до помилок.

2.4 Створення та заповнення таблиць бази даних

Для організації модулю «Методичне забезпечення дисциплін» в межах існуючої бази alma_mater_db створюємо низку нових таблиць і заповнюємо їх тестовими даними через phpMyAdmin.

Визначення структури таблиць:

1. am_learning_resource_type – перелік типів навчальних матеріалів (конспект, електронний курс, посібник тощо):

```
CREATE TABLE am_learning_resource_type (
  id INT AUTO_INCREMENT PRIMARY KEY NOT NULL,
  name TEXT NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

2. am_course_resource_type – категорії забезпечення для дисциплін (основна література, додаткова література, електронні ресурси):

```
CREATE TABLE am_course_resource_type (
  id INT AUTO_INCREMENT PRIMARY KEY NOT NULL,
  name TEXT NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

3. am_learning_resource – власне методичні матеріали:

```
CREATE TABLE am_learning_resource (
  id INT AUTO_INCREMENT PRIMARY KEY NOT NULL,
```

```
  resourcetype_id INT NOT NULL,
  resourcelink TEXT NOT NULL,
  resourcedescription TEXT,
  resourceyear INT NOT NULL,
  FOREIGN KEY (resourcetype_id) REFERENCES
am_learning_resource_type(id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

4. am_learning_resource_author – зв'язок між ресурсами та їх авторами (замість author_id може використовуватися посилання на таблицю користувачів am_users):

```
CREATE TABLE am_learning_resource_author (
  id INT AUTO_INCREMENT PRIMARY KEY NOT NULL,
  resource_id INT NOT NULL,
  author_id INT NOT NULL,
  FOREIGN KEY (resource_id) REFERENCES am_learning_resource(id),
  FOREIGN KEY (author_id) REFERENCES am_users(id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

5. am_course_resource – прив'язка матеріалів до конкретних дисциплін:

```
CREATE TABLE am_course_resource (
  id INT AUTO_INCREMENT PRIMARY KEY NOT NULL,
  resource_id INT NOT NULL,
  course_id INT NOT NULL,
  courseresourcetype_id INT NOT NULL,
  FOREIGN KEY (resource_id) REFERENCES am_learning_resource(id),
  FOREIGN KEY (course_id) REFERENCES am_course(id),
  FOREIGN KEY (courseresourcetype_id) REFERENCES
am_course_resource_type(id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

Наповнення тестовими даними

Оскільки реальні дані ще не доступні, для перевірки CRUD-операцій

додаємо тестові записи:

1. Типи ресурсів:

```
INSERT INTO am_learning_resource_type (name) VALUES
  ('Конспект'),
  ('Електронний курс'),
  ('Посібник'),
  ('Конспект лекцій'),
  ('Методичні вказівки з лабораторної роботи'),
  ('Методичні вказівки з самостійної роботи'),
  ('Методичні вказівки з практичної роботи'),
  ('Електронний навчальний курс'),
  ('Підручник');
```

2. Категорії забезпечення дисциплін:

```
INSERT INTO am_course_resource_type (name) VALUES
  ('Основна література'),
  ('Додаткова література'),
  ('Електронні ресурси');
```

3. Методичні матеріали:

```
INSERT INTO am_learning_resource (resourcetype_id, resourcelink, resourcedescription, resourceyear) VALUES
    (1, 'https://drive.google.com/view/123123', 'Слабінога М.О. Фронтенд-розробка та веб-дизайн: методичні вказівки з виконання лабораторних робіт -123 с. - 2023.', 2023),
    (3, 'https://example.com/resource/3', 'Методичний матеріал для дисципліни "Математика"', '2023');
```

4. Автори ресурсів:

```
INSERT INTO am_learning_resource_author (resource_id, author_id) VALUES
```

```
    (1, 1),
    (1, 2),
    (1, 3),
    (4, 1);
```

5. Прив’язка до дисциплін:

```
    INSERT INTO am_course_resource (resource_id, course_id,
courseresourcetype_id) VALUES
    (1, 1, 1),
    (1, 2, 2);
```

Таким чином, розширення існуючої бази даних університету шляхом створення нових таблиць для компоненти «Методичне забезпечення дисциплін», їх наповнення тестовими даними та встановлення належних взаємозв’язків, забезпечує необхідну структуру для зберігання, організації та подальшої обробки інформації про методичні матеріали дисциплін. При цьому інформація розподіляється по різних таблицях, між якими встановлено чіткі зв’язки, що дозволяє привести структуру бази даних до третьої нормальної форми, забезпечуючи оптимальну організацію даних. Це дозволяє безперешкодно переходити до реалізації операцій CRUD та інтегрувати серверну частину з іншими компонентами інформаційної системи університету.

2.5 Use-case діаграма

Для демонстрації процесу взаємодії користувача з серверною частиною компоненти «Методичне забезпечення дисциплін» інформаційної системи університету було створено use-case діаграму (рис. 2.4). Ця діаграма ілюструє всі можливі сценарії взаємодії з системою та відображає послідовність операцій, які здійснюються для керування даними про методичне забезпечення.

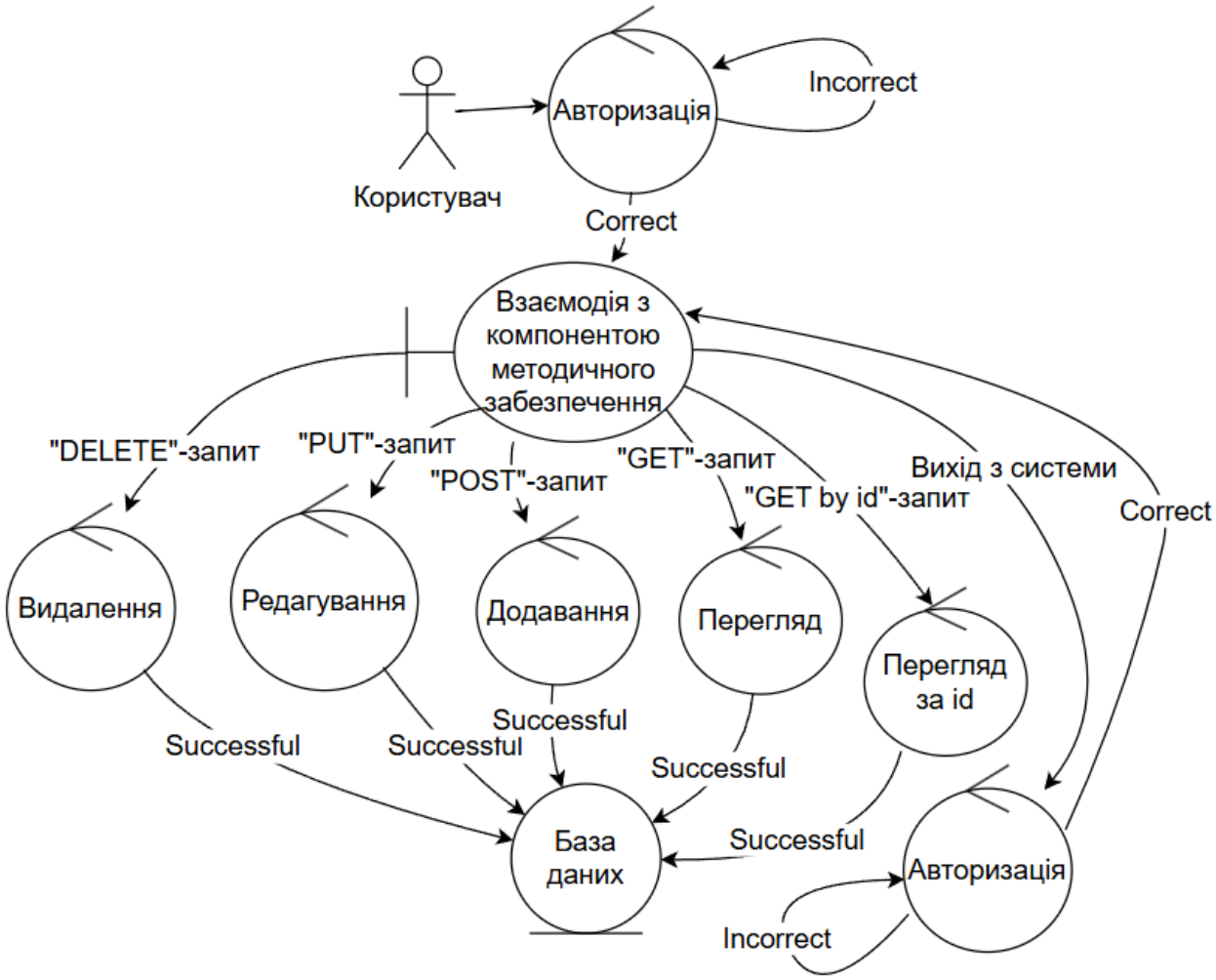

Рисунок 2.4 – Use-case діаграма

Проаналізуємо взаємодію адміністратора з модулем «Методичне забезпечення дисциплін» у серверній частині університетської інформаційної системи. На діаграмі варіантів використання (рис. 2.4) відображено всі можливі дії користувача та зв’язки між ними. Цей графічний опис демонструє, як адміністратор отримує доступ до таких операцій над даними про методичні матеріали дисциплін:

1. Авторизація;

Користувач спочатку проходить перевірку облікових даних. Без успішного входу до системи жодна інша функція недоступна.

2. Додавання записів;

Через HTTP-запит (POST) адміністратор може створювати нові записи з інформацією про методичні матеріали.

3. Редагування наявних даних;

Використовуючи PUT-запит, користувач оновлює вміст існуючих записів.

4. Видалення непотрібної інформації;

За допомогою DELETE-запиту адміністратор усуває записи, які більше не потрібні.

5. Перегляд даних.

GET-запит дозволяє отримувати як повний перелік матеріалів, так і деталі окремих записів.

Таким чином, завдяки чіткому визначенню ролей та операцій у use-case діаграмі забезпечується злагоджена робота серверної частини системи й гнучке управління даними методичного забезпечення дисциплін.

2.6 Діаграма послідовностей

Побудуємо діаграму послідовностей (рис. 2.5), яка ілюструє обмін повідомленнями між користувачем, серверною логікою та базою даних у модулі «Методичне забезпечення дисциплін».

На діаграмі (рис. 2.5) видно такі кроки:

1. Ініціювання запиту;

Адміністратор надсилає HTTP-запит (GET) до веб-інтерфейсу з метою отримати перелік наявних методичних матеріалів або деталізовану інформацію за конкретним ID.

2. Передача на сервер;

Веб-сервер отримує запит, формує SQL-команду і відправляє її MySQL-серверу.

3. Обробка на рівні БД;

MySQL-сервер виконує запит до сховища даних і повертає набір записів продавцю сервера.

4. Відповідь користувачу.

Сервер обробляє результати запиту й надсилає клієнту готовий JSON- або HTML-відповідь для відображення в браузері.

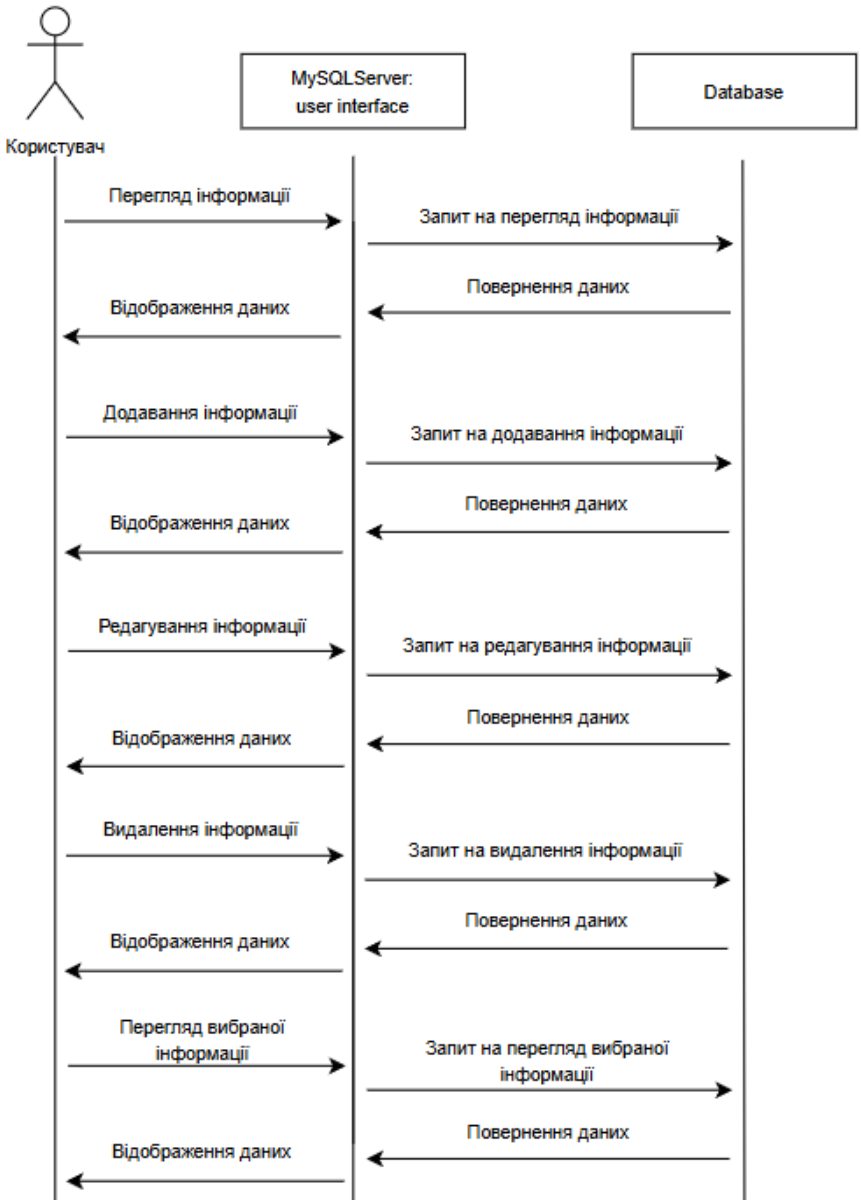

Рисунок 2.5 – Діаграма послідовностей взаємодії користувача з SQL-сервером та базою даних

Окрім операції перегляду, на діаграмі наведені потоки для CRUD-функціоналу:

– створення (POST): клієнт відправляє дані нового матеріалу → сервер генерує INSERT-запит → БД підтверджує успішне додавання → відповідь клієнту;

– оновлення (PUT): клієнт передає змінені поля й ID → сервер формує UPDATE-запит → БД виконує оновлення → результат надсилається назад;

– видалення (DELETE): клієнт зазначає ID запису → сервер адресує DELETE-запит → БД видаляє строку → відповідь із статусом операції.

Таким чином, діаграма послідовностей відображає часову послідовність звернень і відповідей між усіма трьома учасниками – користувачем, серверною частиною та СУБД, що дозволяє чітко простежити логіку обміну даними та гарантувати коректність обробки кожного запиту.

2.7 Розробка структури серверної частини компоненти

Після проєктування структури бази даних і визначення всіх необхідних операцій для користувача переходимо до етапу безпосередньої розробки ПЗ. Насамперед окреслимо загальну архітектуру системи – вона складається з двох головних модулів: клієнтської частини (фронтенду) та серверної (бекенду). У межах цієї роботи зосередимося на організації бекенду.

На структурній схемі серверної частини (рис. 2.6) наведено основні компоненти проєкту та їх ролі:

– моделі (models): описують структуру даних і відображаються на таблиці БД;

– контролери (controllers): обробляють вхідні HTTP-запити, визначають логіку відповіді;

– сервіси (services): містять бізнес-правила та алгоритми взаємодії з даними;

– маршрути (routes): задають кінцеві точки API й скеровують запити до відповідних контролерів.

З погляду фронтенду досить визначити базові вимоги: авторизація виконується через веб-інтерфейс, а всі подальші дії користувач здійснює зручною панеллю управління.

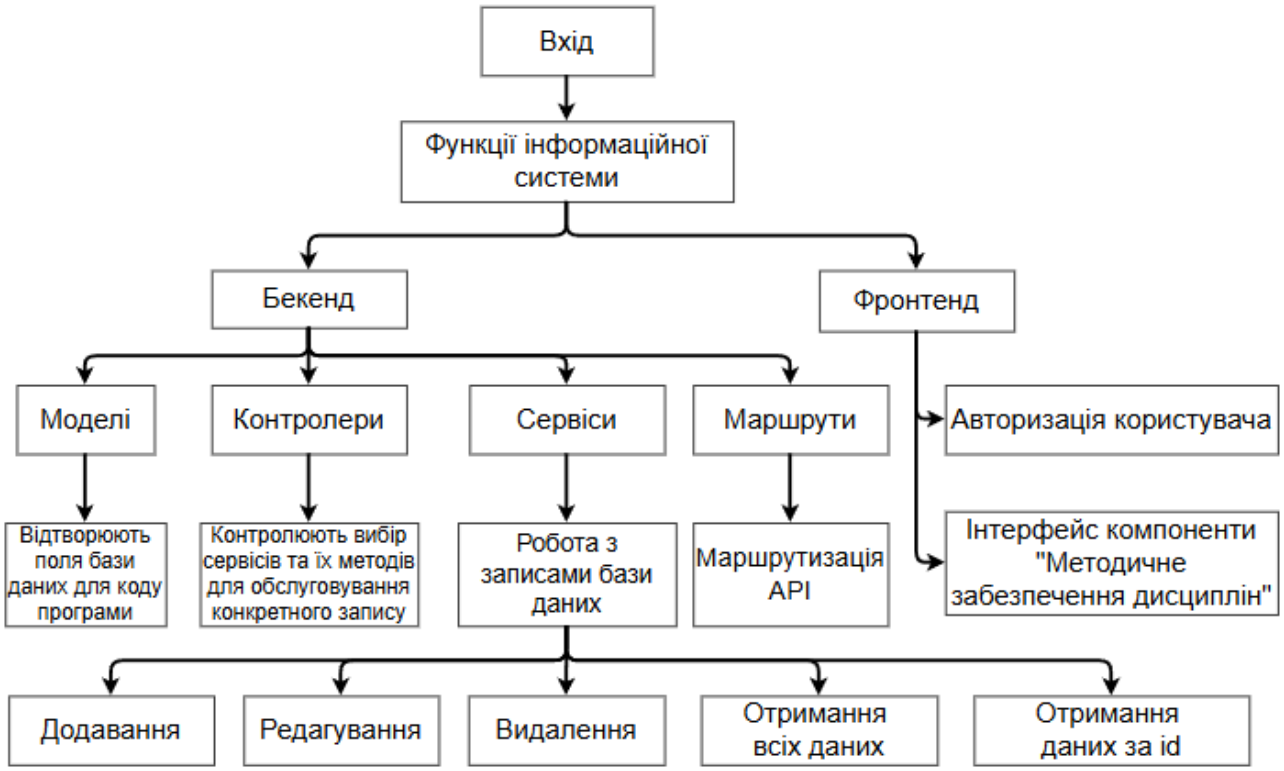

Рисунок 2.6 – Структурна схема серверної частини компоненти

Поєднання вищезгаданих блоків гарантує, що серверна частина коректно обробляє запити CRUD-операцій (створення, читання, оновлення, видалення) для модуля «Методичне забезпечення дисциплін». В результаті отримаємо повністю функціональний компонент, що дозволяє адміністратору переглядати, додавати, змінювати та видаляти інформацію про методичні матеріали дисциплін.

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Розробка програмного коду

Щоб забезпечити взаємодію серверної частини з СУБД, у проєкті використовуються два файли: DBProvider.php та config.php.

У файлі DBProvider.php реалізовано встановлення з'єднання з MySQL за допомогою об'єкта класу mysqli. Наведений нижче код ініціює підключення до бази даних, використовуючи параметри, що задаються у файлі налаштувань:

```
<?php
$AMDBConnection = new mysqli($AMServerName, $AMUserName, $AMPassword, $AMDatabase);

// Check connection
if ($AMDBConnection->connect_error) {
  die("Connection failed: " . $AMDBConnection->connect_error);
}
?>
```

Цей код відповідає за встановлення зв'язку із сервером бази даних, використовуючи такі параметри, як адреса сервера (наприклад, "localhost"), логін користувача, відповідний пароль та назву бази даних.

У файлі config.php міститься повний набір конфігураційних налаштувань, необхідних для роботи системи. Серед них – основні дані для підключення до бази даних, а також додаткові параметри автентифікації.

```
<?php
//Server URL
$AMServerURL = 'http://localhost';
//Base URL of the Project
$AMProjectBaseUrl = '/alma-mater-backend';
//AM Database Login and Password
```

```
$AMServerName = "localhost";
$AMUserName = "root";
$AMPassword = "";
$AMDatabase = "alma_mater_db";
//API Bearer Token
$AMBearerToken = '…';
$AMAppToken = '…';
//Google OAuth credentials
$AMGoogleClientID = '…';
$AMGoogleSecret = '…';
$AMGoogleAuthEndpoint = 'https://accounts.google.com/o/oauth2/auth';
$AMGoogleTokenEndpoint = 'https://oauth2.googleapis.com/token';
$AMAuthCallbackURL = "/auth/callback";
```

Звернемо увагу, що оскільки система розгорнута на локальному сервері, використовується адреса localhost. Для доступу до бази даних використовується користувач root без пароля, а назва бази даних – alma_mater_db. Правильність цих налаштувань є критичною, оскільки навіть незначна помилка може спричинити невдале встановлення з’єднання.

Таким чином, коректно налаштоване підключення до бази даних є фундаментальним етапом для забезпечення стабільної роботи серверної частини проекту, адже воно гарантує безперебійну обробку запитів і забезпечення автентифікації користувачів.

Далі розглянемо створення програмного коду для однієї з таблиць бази даних, яка розробляється для компоненти «Методичне забезпечення дисциплін». Наведено реалізацію файлу LearningResource.php, що моделює таблицю am_learning_resource.

Вміст файлу LearningResource.php:

```
<?php
require_once('BaseModel.php');
class LearningResource extends BaseModel {
    private $resourcetype_id;
    private $resourcelink;
```

```
        private $resourcedescription;
        private $resourceyear;

        public function __construct($paramArray) {
            $this->id = $paramArray['id'];
            $this->resourcetype_id = $paramArray['resourcetype_id'];
            $this->resourcelink = $paramArray['resourcelink'];
            $this->resourcedescription                                    =
$paramArray['resourcedescription'];
            $this->resourceyear = $paramArray['resourceyear'];
        }

        public function getAsObject() {
            return get_object_vars($this);
        }
    }
    ?>
```

Цей код визначає клас LearningResource, який наслідує базовий клас BaseModel та відповідає за моделювання таблиці am_learning_resource. Основні властивості класу включають:

– $resourcetype_id – тип методичного ресурсу;

– $resourcelink – посилання на ресурс;

– $resourcedescription – опис ресурсу;

– $resourceyear – рік створення ресурсу.

Конструктор класу приймає асоціативний масив $paramArray і ініціалізує властивості об'єкта значеннями з цього масиву, де ключі відповідають назвам колонок у таблиці бази даних. Метод getAsObject() повертає всі властивості об’єкта у вигляді асоціативного масиву для зручності подальшої обробки даних.

Таким чином, клас LearningResource дозволяє створювати об'єкти методичних ресурсів та ефективно маніпулювати їхніми атрибутами в рамках реалізації компоненту «Методичне забезпечення дисциплін». Цей підхід сприяє чіткому моделюванню даних та забезпечує зручну інтеграцію з операціями

CRUD у серверній частині інформаційної системи університету.

Далі наведено реалізацію класу LearningResourceService, який забезпечує взаємодію з таблицею am_learning_resource (що містить дані про методичні матеріали дисциплін) за допомогою повного циклу CRUD-операцій. Цей сервіс є невід'ємною частиною серверної логіки компоненти «Методичне забезпечення дисциплін» та використовується контролерами для обробки HTTP-запитів.

Вміст файлу LearningResourceService.php:

```
<?php
require_once('./app/services/BaseService.php');
require_once('./app/models/LearningResource.php');

class LearningResourceService extends BaseService {
    static $tableName = 'am_learning_resource';

    public function add($paramArray) {
        if (!isset($paramArray['id'])) {
            $paramArray['id'] = ++$this->index;
        }
        $learningResource = new LearningResource($paramArray);
        array_push($this->dataArray, $learningResource);
        return $paramArray['id'];
    }
```

Метод add($paramArray): Створює новий об'єкт класу LearningResource на основі даних, отриманих у вигляді асоціативного масиву, і додає його до локального масиву даних. Якщо ідентифікатор відсутній, він генерується автоматично шляхом збільшення лічильника.

```
    public function insertToDatabase($conn, $paramArray) {
        $request = $conn->prepare("INSERT INTO " . self::$tableName
. " VALUES (DEFAULT, ?, ?, ?, ?)");
        $request->bind_param("isss",              $resourcetype_id,
$resourcelink, $resourcedescription, $resourceyear);
        $resourcetype_id    = $paramArray['resourcetype_id'];
```

```
            $resourcelink        = $paramArray['resourcelink'];
            $resourcedescription = $paramArray['resourcedescription'];
            $resourceyear        = $paramArray['resourceyear'];
            $request->execute();
            $request->close();
        }
```

Метод insertToDatabase($conn, $paramArray): Виконує вставлення нового запису у таблицю за допомогою підготовленого SQL-запиту `INSERT INTO`, де дані з масиву прив'язуються до запиту через метод bind_param().

```
        public function deleteFromDatabase($conn, $id) {
            $request = $conn->prepare("DELETE FROM " . self::$tableName
. " WHERE id=?");
            $request->bind_param("i", $id);
            $request->execute();
            $request->close();
        }
```

Метод deleteFromDatabase($conn, $id): Видаляє запис із таблиці за заданим ідентифікатором, використовуючи SQL-запит `DELETE`, де значення id передається через bind_param().

```
        public function updateDatabaseById($conn, $id, $paramArray) {
            $request = $conn->prepare("UPDATE " . self::$tableName . "
SET   `resourcetype_id`=?,   `resourcelink`=?,   `resourcedescription`=?,
`resourceyear`=? WHERE `id`=?");
            $request->bind_param("isssi",             $resourcetype_id,
$resourcelink, $resourcedescription, $resourceyear, $id);
            $resourcetype_id    = $paramArray['resourcetype_id'];
            $resourcelink        = $paramArray['resourcelink'];
            $resourcedescription = $paramArray['resourcedescription'];
            $resourceyear        = $paramArray['resourceyear'];
            $request->execute();
            $request->close();
        }
```

Метод updateDatabaseById($conn, $id, $paramArray): Оновлює запис у таблиці через SQL-запит `UPDATE`, в якому всі поля оновлюються відповідно до даних, що містяться у масиві $paramArray.

```
        public function getFromDatabaseById($conn, $id) {
            $request = "SELECT * FROM " . self::$tableName . " WHERE id="
. $id;
            $result = $conn->query($request);
            if ($result->num_rows > 0) {
                while ($row = $result->fetch_assoc()) {
                    $this->add($row);
                }
            } else {
                echo "";
            }
        }
```

Метод getFromDatabaseById($conn, $id): Використовуючи SQL-запит `SELECT`, отримує запис із таблиці за заданим id і додає його до локального масиву даних, що дозволяє пізніше обробляти ці дані.

```
        public function getAllFromDataBase($conn) {
            $request = "SELECT * FROM " . self::$tableName . " ORDER BY
id ASC";
            $result = $conn->query($request);
            if ($result->num_rows > 0) {
                while ($row = $result->fetch_assoc()) {
                    $this->add($row);
                }
            } else {
                echo "";
            }
        }
    }
    ?>
```

Метод getAllFromDataBase($conn): Отримує всі записи з таблиці, відсортовані за зростанням id, і додає їх до локального масиву для подальшого використання.

Цей клас є важливою складовою бекенд-логіки компоненту «Методичне забезпечення дисциплін», оскільки дозволяє ефективно управляти даними таблиці am_learning_resource через реалізацію повного циклу CRUD-операцій, що сприяє інтеграції серверної частини з іншими компонентами інформаційної системи університету.

Далі наведено контролер, який використовує сервіс LearningResourceService для виконання операцій додавання, оновлення, видалення та отримання даних за допомогою повного циклу CRUD. Також наведено код базового контролера BaseController.php з методом processRequest(), який аналізує HTTP-запит і викликає відповідні методи сервісу.

Файл LearningResourceController.php

У цьому файлі відбувається початкова ініціалізація контролера. Спочатку підключаються необхідні файли: сервіс для роботи з методичними ресурсами та базовий контролер. Далі створюється об'єкт сервісу, після чого він передається до конструктора базового контролера разом із з’єднанням до бази даних. На завершення викликається метод processRequest(), який відповідальний за обробку HTTP-запитів.

```
<?php
require_once('./app/services/LearningResourceService.php');
require_once('./app/controllers/BaseController.php');

$learningResourceService = new LearningResourceService();
$controller = new BaseController($AMDBConnection, $learningResourceService);
$controller->processRequest();
?>
```

Файл BaseController.php

Цей файл містить клас BaseController, який інкапсулює логіку обробки HTTP-запитів. Метод processRequest() аналізує тип запиту (GET, POST, PUT, DELETE) та викликає відповідні методи сервісу для отримання, додавання, оновлення чи видалення даних.

```
<?php
class BaseController {
    private $list;
    public $conn;

    public function __construct($conn, $list) {
        $this->conn = $conn;
        $this->list = $list;
    }

    public function processRequest() {
        if ($_SERVER['REQUEST_METHOD'] == 'GET') {
            header("HTTP/1.1 200 OK");
            if (isset($_REQUEST['id'])) {
                $this->list->getFromDatabaseById($this->conn,
$_REQUEST['id']);
                echo $this->list->getAsJSON();
            } else {
                $this->list->getAllFromDataBase($this->conn);
                echo $this->list->getAsJSON();
            }
        } else if ($_SERVER['REQUEST_METHOD'] == 'POST') {
            $data                                                =
get_object_vars(json_decode(file_get_contents('php://input')));
            $this->list->insertToDatabase($this->conn, $data);
            $this->list->getFromDatabaseById($this->conn,    $this-
>conn->insert_id);
            header("HTTP/1.1 200 OK");
            echo $this->list->getAsJSON();
        } else if ($_SERVER['REQUEST_METHOD'] == 'PUT') {
           $data                                                 =
get_object_vars(json_decode(file_get_contents('php://input')));
```

```
                $this->list->updateDatabaseById($this->conn,
$data['id'], $data);
                $this->list->getFromDatabaseById($this->conn,
$data['id']);
                header("HTTP/1.1 200 OK");
                echo $this->list->getAsJSON();
            } else if ($_SERVER['REQUEST_METHOD'] == 'DELETE') {
                $data                                                =
get_object_vars(json_decode(file_get_contents('php://input')));
                $this->list->deleteFromDatabase($this->conn,
$data['id']);
                header("HTTP/1.1 200 OK");
                echo '{"status":"OK"}';
            }
        }
    }
    ?>
```

GET-запит:

– якщо параметр id передається, контролер отримує конкретний запис з бази, викликаючи метод getFromDatabaseById();

– якщо id не заданий, виконується завантаження всіх записів із викликом методу getAllFromDataBase(). Після цього дані перетворюються у формат JSON для відправки клієнту.

POST-запит: контролер зчитує дані зі стандартного потоку php://input, що дозволяє отримати інформацію у форматі JSON. Ці дані використовуються для створення нового запису за допомогою методу insertToDatabase(). Після успішного додавання, новостворений запис отримується та повертається у форматі JSON.

PUT-запит: для оновлення даних також зчитуються дані із потоку, після чого викликається метод updateDatabaseById(), який вносить зміни у базу даних. Контролер повертає оновлений запис для підтвердження успішності операції.

DELETE-запит: отримуючи id запису для видалення, контролер викликає метод deleteFromDatabase(), після чого повертає статус операції у вигляді JSON.

Таким чином, реалізація класів LearningResourceService та BaseController забезпечує повний цикл CRUD-операцій для роботи з таблицею методичних ресурсів. Контролер відповідає за обробку HTTP-запитів і виклик відповідних методів сервісу, що сприяє інтеграції серверної логіки компоненти «Методичне забезпечення дисциплін» з іншими частинами інформаційної системи університету.

Далі в файлі AppRouter.php визначається маршрут для сторінки, яка відображає інформацію про навчальні ресурси (методичні матеріали). Маршрутизація здійснюється за допомогою конструкції `switch-case`, що дозволяє задіювати конкретний контролер залежно від URL-запиту.

Частина вмісту файлу AppRouter.php:

```
switch ($request) {
    case $AMProjectBaseUrl . '/learning-resources':

require_once("./app/controllers/LearningResourceController.php");
        break;
}
```

Опис роботи маршруту під час виконання HTTP-запитів

Цей фрагмент коду виконує наступні дії:

1. Перевірка запиту:

Конструкція `switch` аналізує значення змінної `$request`. Якщо URL-запит збігається з рядком `"$AMProjectBaseUrl/learning-resources"`, тобто містить потрібний маршрут для сторінки з навчальними ресурсами, виконується наступний крок;

2. Підключення контролера:

За співпадінням маршруту виконується команда

`require_once("./app/controllers/LearningResourceController.php");`

Це дозволяє підключити файл контролера, який містить всю необхідну логіку для обробки HTTP-запитів, пов'язаних із методичними ресурсами (таблиця am_learning_resource);

3. Завершення блоку маршруту:

Команда `break;` завершує виконання цього блоку `case`, що запобігає подальшій обробці інших випадків.

Таким чином, механізм маршрутизації забезпечує, що коли користувач переходить на URL-адресу, що відповідає шляху `"$AMProjectBaseUrl/learning-resources"`, система автоматично задіє відповідний контролер. Контролер обробляє запити (додавання, оновлення, видалення, отримання даних) через виклик методів сервісного шару, що взаємодіє з базою даних.

Аналогічні дії виконуємо для інших таблиць модуля методичного забезпечення.

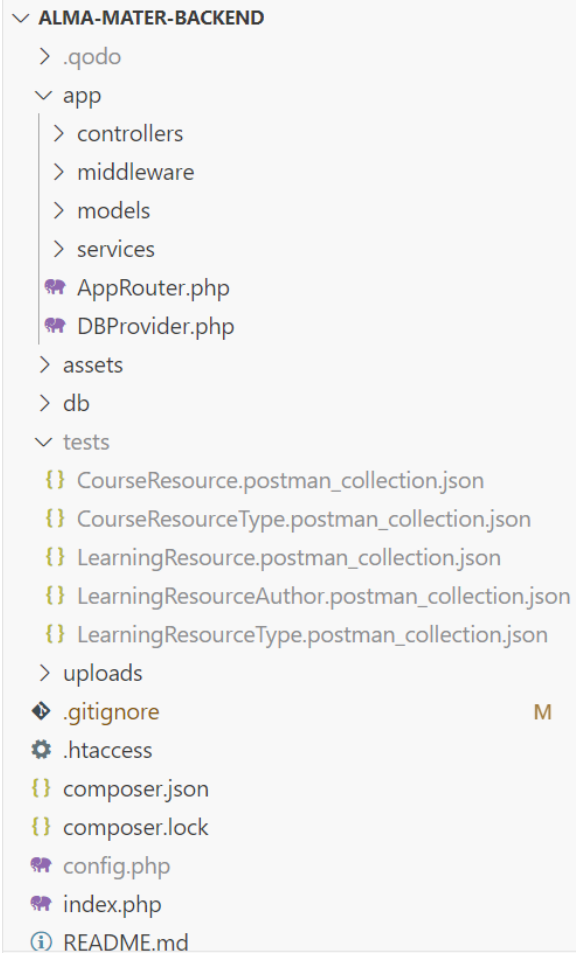

Рисунок 3.1 – Структура проекту

На рисунку 3.1 представлено організацію файлової структури проекту, розробленого для серверної частини інформаційної системи університету, що реалізує компоненту «Методичне забезпечення дисциплін». Структура включає ієрархічно впорядковані каталоги та файли, зокрема: контролери, моделі, сервіси, маршрути, конфігураційні файли та інші ресурси, необхідні для роботи з базою даних та виконання CRUD-операцій. Така організація спрощує навігацію по проекту, сприяє ефективному управлінню кодом і полегшує подальше розширення функціональності системи.

3.2 Реалізація специфічних запитів

Згідно з постановою Про затвердження Ліцензійних умов провадження освітньої діяльності №1187 [17], до функціоналу розроблюваної компоненти «Методичне забезпечення дисциплін» було додано специфічні запити для вибірки інформації за додатковими критеріями. Метою цього доповнення є забезпечення можливості відбору даних за наступними параметрами:

– вибірка всього методичного забезпечення для заданої дисципліни за останні 5 років;

– відбір записів за ідентифікатором автора;

– вибірка записів за ідентифікатором автора, які були створені протягом останніх 5 років;

– отримання всіх посібників і підручників для конкретного автора за останні 5 років.

Реалізація додаткових запитів у коді

LearningResourceController.php

Контролер розподіляє запити залежно від параметра action у HTTP-запиті, що дозволяє викликати відповідні методи сервісу для реалізації потрібної логіки.

```
if (!isset($_REQUEST['action'])) {
    $controller = new BaseController($AMDBConnection,
```

```
$learningResourceService);
        $controller->processRequest();
    } elseif ($_REQUEST['action'] == 'getByCourseIdLast5Years') {
        $learningResourceService-
>getByCourseIdLast5Years($AMDBConnection, $_REQUEST['course_id']);
        echo $learningResourceService->getAsJSON();
    } elseif ($_REQUEST['action'] == 'getByAuthorId') {
        $learningResourceService->getByAuthorId($AMDBConnection,
$_REQUEST['author_id']);
        echo $learningResourceService->getAsJSON();
    } elseif ($_REQUEST['action'] == 'getByAuthorIdLast5Years') {
        $learningResourceService-
>getByAuthorIdLast5Years($AMDBConnection, $_REQUEST['author_id']);
        echo $learningResourceService->getAsJSON();
    }          elseif          ($_REQUEST['action']          ==
'getManualsTextbooksByAuthorLast5Years') {
        $learningResourceService-
>getManualsTextbooksByAuthorLast5Years($AMDBConnection,
$_REQUEST['author_id']);
        echo $learningResourceService->getAsJSON();
    }
```

– якщо action дорівнює getByCourseIdLast5Years, контролер викликає метод, який повертає всі методичні ресурси для заданої дисципліни за останні 5 років;

– якщо action дорівнює getByAuthorId, викликається метод для отримання всіх ресурсів, пов’язаних із заданим автором;

– аналогічно, при значенні getByAuthorIdLast5Years або getManualsTextbooksByAuthorLast5Years – викликається відповідний метод сервісу, після чого виконана вибірка конвертується у формат JSON для повернення клієнту.

LearningResourceService.php

Кожен із методів сервісу відповідає за побудову спеціального SQL-запиту

із відповідними умовами:

1. Отримання методичного забезпечення за дисципліною за останні 5 років;

Метод getByCourseIdLast5Years визначає часовий поріг (поточний рік мінус 5) і виконує запит із приєднанням до таблиці, яка зберігає зв'язки ресурсів із дисциплінами. Використання ключового слова `DISTINCT` гарантує повернення лише унікальних записів:

```
    public function getByCourseIdLast5Years($conn, $courseId)
      {
          $yearLimit = date("Y") - 5;
          $query = "SELECT DISTINCT lr.*
                    FROM " . self::$tableName . " lr
                    JOIN am_course_resource cr ON lr.id = cr.resource_id
                    WHERE cr.course_id = " . $courseId . " AND
lr.resourceyear >= " . $yearLimit;

          $result = $conn->query($query);
          if ($result && $result->num_rows > 0) {
              while ($row = $result->fetch_assoc()) {
                  $this->add($row);
              }
          } else {
              echo "";
          }
      }
```

2. Отримання методичного забезпечення за ідентифікатором автора;

Метод getByAuthorId реалізує вибірку ресурсів, пов'язаних з певним автором, через об'єднання таблиці ресурсів із таблицею зв'язків, де зберігається значення `author_id`:

```
    public function getByAuthorId($conn, $authorId)
    {
        $query = "SELECT lr.*
```

```
                    FROM " . self::$tableName . " lr
                    JOIN am_learning_resource_author lra ON lr.id =
lra.resource_id
                    WHERE lra.author_id = " . $authorId;

            $result = $conn->query($query);
            if ($result && $result->num_rows > 0) {
                while ($row = $result->fetch_assoc()) {
                    $this->add($row);
                }
            } else {
                echo "";
            }
        }
```

3. Отримання методичного забезпечення за ідентифікатором автора за останні 5 років;

Метод getByAuthorIdLast5Years доповнює попередню логіку умовою часової вибірки, що обмежує результати записами з останніх 5 років:

```
        public function getByAuthorIdLast5Years($conn, $authorId)
        {
            $yearLimit = date("Y") - 5;
            $query = "SELECT lr.*
                    FROM " . self::$tableName . " lr
                    JOIN am_learning_resource_author lra ON lr.id =
lra.resource_id
                    WHERE lra.author_id = " . $authorId . " AND
lr.resourceyear >= " . $yearLimit;

            $result = $conn->query($query);
            if ($result && $result->num_rows > 0) {
                while ($row = $result->fetch_assoc()) {
                    $this->add($row);
                }
            } else {
                echo "";
```

```
        }
    }
```

4. Отримання посібників і підручників для автора за останні 5 років.

Метод getManualsTextbooksByAuthorLast5Years орієнтований на спеціальну категоризацію матеріалів. Запит включає додаткове з'єднання із таблицею типів ресурсів, що дозволяє відфільтрувати записи за назвами типів, такими як «Підручник» та «Посібник», а також застосувати часовий фільтр:

```
    public function getManualsTextbooksByAuthorLast5Years($conn, $authorId)
    {
        $yearLimit = date("Y") - 5;
        $query = "SELECT lr.*
                  FROM " . self::$tableName . " lr
                  JOIN am_learning_resource_author lra ON lr.id = lra.resource_id
                  JOIN am_learning_resource_type lrt ON lr.resourcetype_id = lrt.id
                  WHERE lra.author_id = " . $authorId . "
                    AND lr.resourceyear >= " . $yearLimit . "
                    AND lrt.name IN ('Підручник', 'Посібник')";

        $result = $conn->query($query);
        if ($result && $result->num_rows > 0) {
            while ($row = $result->fetch_assoc()) {
                $this->add($row);
            }
        } else {
            echo "";
        }
    }
```

Ці методи забезпечують розширену функціональність компоненти, дозволяючи отримати інформацію про методичне забезпечення за різними

критеріями (за дисципліною, за ідентифікатором автора, із часовими обмеженнями, а також спеціальний відбір для посібників і підручників). Завдяки такому підходу система набуває гнучкості, що є важливим для забезпечення повноцінного управління даними в рамках розроблюваної інформаційної системи університету.

3.3 Робота серверної частини

Після завершення розробки програмного коду компоненти «Методичне забезпечення дисциплін» переходимо до перевірки її роботи шляхом виконання HTTP-запитів.

Протокол передачі гіпертексту (HTTP) забезпечує зв'язок між клієнтом і сервером за принципом запит-відповідь: веб-браузер може виступати клієнтом, який надсилає запит до сервера, а сервер повертає відповідь із зазначенням стану запиту та, за необхідності, запитуваним контентом [18].

З огляду на це, для тестування реалізованих методів GET, POST, PUT і DELETE, описаних у базовому контролері, спочатку перевіряється їхня робота через веб-браузер. Наприклад, при переході за відповідною URL-адресою активується метод GET, який відображає дані з таблиці (рис. 3.2).

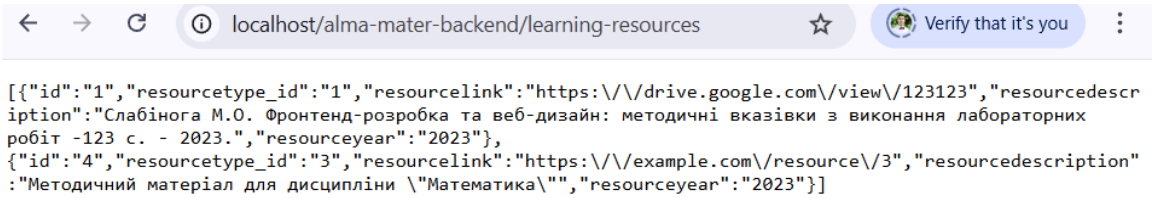

Рисунок 3.2 – Результат виконання методу GET у браузері

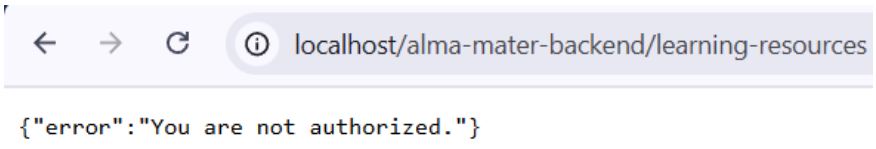

Рисунок 3.3 – Результат виконання методу GET у браузері при неуспішній автентифікації

На рисунку 3.3 видно, що при переході на сторінку компоненти «Методичне забезпечення дисциплін» за допомогою браузера виконується HTTP-запит методу GET, але автентифікація за допомогою Bearer Token не проходить успішно. Оскільки система працює на локальному сервері, перевірка Bearer Token завжди повертає помилку авторизації через відсутність можливості передати дані користувача у запиті. Для тестування повного функціоналу не рекомендується використовувати браузер – замість цього застосуємо спеціалізовані інструменти, такі як Postman.

Postman – це платформа для розробки, тестування та співпраці над API, що дозволяє виконувати HTTP-запити із реальним процесом авторизації через Bearer Token, який передаються через заголовок «Authorization: Bearer <token>» [19], що забезпечує коректну автентифікацію та перевірку всіх CRUD-методів. Таким чином, використання Postman є більш комплексним підходом для тестування бекенд-логіки серверної частини компоненти «Методичне забезпечення дисциплін».

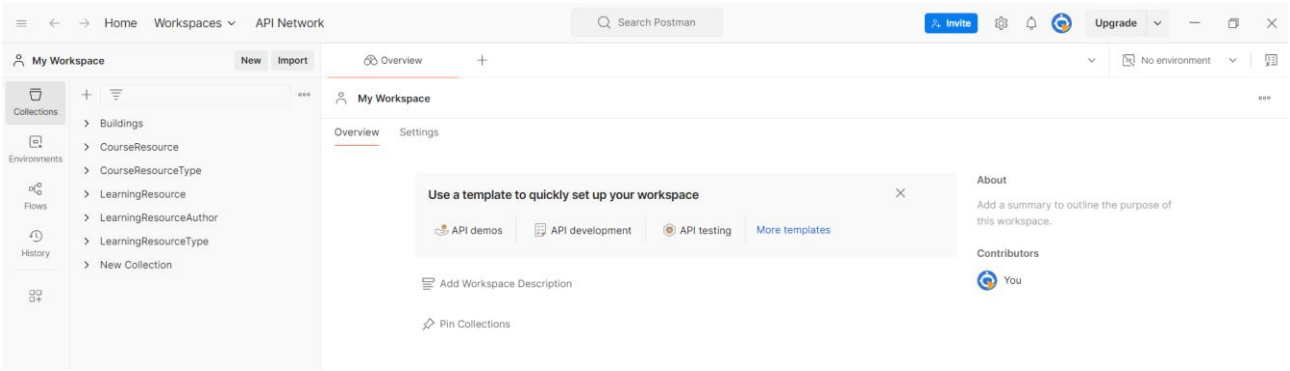

Рисунок 3.4 – Робоче середовище Postman

На робочому середовищі Postman (рис. 3.4) в лівому меню розташовані колекції запитів із різними HTTP-методами: GET (для отримання всіх записів або конкретного запису за id), POST, PUT та DELETE. Використання Bearer Token відповідно до протоколу OAuth 2.0 дозволяє безпечно здійснювати автентифікацію користувача та отримувати доступ до захищених ресурсів [20].

Для зручності конфігурування запитів у Postman створено глобальні

змінні. Так, змінна apiUrl (рис. 3.5) дозволяє вказати базову адресу API лише одного разу – при її зміні всі запити автоматично отримають оновлений URL. Окрім цього, введено змінну bearerToken, яка відповідає за передачу токена доступу в заголовку Authorization: Bearer <token>. Цей токен дозволяє серверу ідентифікувати користувача та надати йому необхідні права; у разі його відсутності або некоректності сервер поверне помилку 401 Unauthorized.

Bearer-токен вважається секретним, тому в налаштуваннях Postman він позначається як secret – його значення приховано, щоб уникнути випадкового розголошення та несанкціонованого доступу до API.

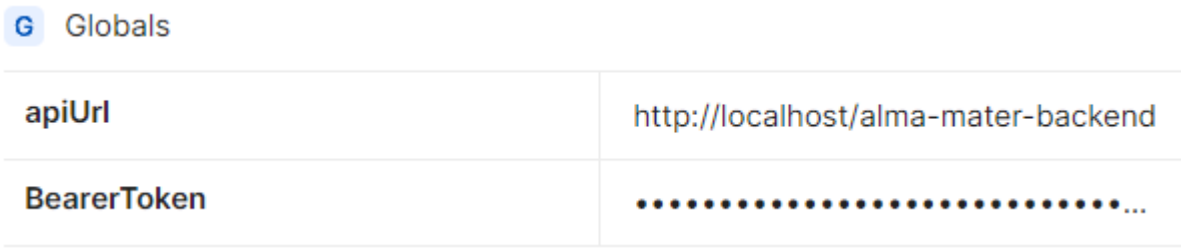

Рисунок 3.5 – Глобальні змінні в Postman

Початок JSON-коду:

```
{    "info": {        "_postman_id": "unique-id-3",
        "name": "LearningResource",
        "description": "API for managing Learning Resources.",
schema":
"https://schema.getpostman.com/json/collection/v2.1.0/collection.json"
        },    "item": [        {            "name": "Get all learning
resources",
                "request": {                "method": "GET",
                    "header": [],                "url": {
                        "raw": "{{apiUrl}}/learning-resources",
                        "host": ["{{apiUrl}}"],
                                                "path":    ["learning-
resources"]                }            }        },
```

Цей JSON-код є структурою колекції запитів для Postman, призначених для роботи з API управління навчальними ресурсами. Він описує різні запити для CRUD-операцій. Детальний опис кожного блоку:

1. info:
   – містить метадані колекції:
     – _postman_id: унікальний ідентифікатор колекції;
     – name: назва колекції – "LearningResource";
     – description: опис API – "API for managing Learning Resources";
     – schema: посилання на схему формату Postman колекції.
2. item:
   – містить масив запитів, де кожен об'єкт описує один із HTTP-запитів.

Get all learning resources:

– метод: GET;
– опис: отримання всіх навчальних ресурсів;
– URL: {{apiUrl}}/learning-resources.

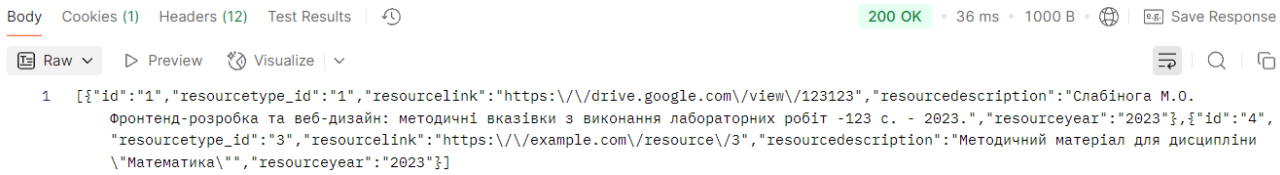

Рисунок 3.6 – Результат виконання GET-запиту для всіх записів

На рисунку 3.6 бачимо, як, скориставшись відповідним маршрутом та натиснувши кнопку «Send» (синього кольору), сервер повертає результат виконання GET-запиту повертає масив із записом про навчальний ресурс. У відповіді містяться такі дані:

– id: 1 – унікальний ідентифікатор ресурсу;
– resourcetype_id: 1 – ідентифікатор типу ресурсу;
– resourcelink: посилання на ресурс;
– resourcedescription: опис ресурсу;

– resourceyear: 2023 – рік публікації ресурсу.

Цей результат показує, що запит успішно виконався і сервер повернув інформацію з бази даних.

Далі виконуємо GET-запит для отримання конкретного запису за його ID, тобто інформації про певне методичне забезпечення.

Продовження JSON-коду:

```
{            "name": "Get learning resource by id",
            "request": {                "method": "GET",
                "header": [],                "url": {
                    "raw": "{{apiUrl}}/learning-resources?id=1",
                    "host": ["{{apiUrl}}"],
                    "path": ["learning-resources"],
                        "query": [{ "key": "id", "value": "1"
}]              }            }        },
```

Get learning resource by id:

– метод: GET;

– опис: отримання конкретного навчального ресурсу за id;

– URL: {{apiUrl}}/learning-resources?id=1;

– параметр запиту: id=1.

Цей запит (рис. 3.7) дозволяє отримати дані лише про той ресурс, який має ідентифікатор id=1.

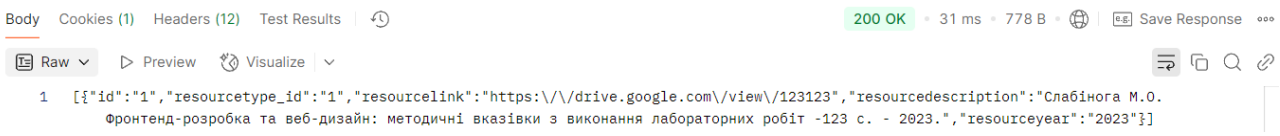

Рисунок 3.7 – Результат виконання GET-запиту за конкретним id

За рисунком 3.7 видно, що при передачі в запиті параметра id=1 сервер повертає дані конкретного навчального ресурсу з ідентифікатором 1 – запит виконано успішно.

Далі виконуємо POST-запит для додавання нового запису в базу даних на сервері, тобто інформації про певне методичне забезпечення.

Продовження JSON-коду:

```
{            "name": "Add learning resource",
            "request": {                "method": "POST",
                "header": [],                "body": {
                        "mode": "raw",                        "raw":
"{\n    \"resourcetype_id\":\"1\",\n    \"resourcelink\":\"https://drive
.google.com/view/123123\",\n    \"resourcedescription\":\"Слабінога М.О.
Фронтенд-розробка та веб-дизайн: методичні вказівки з виконання
лабораторних робіт -123 с. -
2023.\",\n    \"resourceyear\":\"2023\",\n    \"authors\":[123,423,2]\n}
",
                        "options": { "raw": { "language": "json" }
}            },
                "url": {                "raw": "{{apiUrl}}/learning-
resources",
                        "host": ["{{apiUrl}}"], "path": ["learning-
resources"] }        }        },
```

Add learning resource:

– метод: POST;

– опис: додавання нового навчального ресурсу;

– URL: {{apiUrl}}/learning-resources;

– тіло запиту (JSON):

```
{
    "resourcetype_id": "1",
    "resourcelink": "https://drive.google.com/view/123123",
    "resourcedescription": "Слабінога М.О. Фронтенд-розробка
та веб-дизайн: методичні вказівки з виконання лабораторних робіт -
123 с. - 2023.",
    "resourceyear": "2023",
```

```
        "authors": [123, 423, 2]
    }
```

У нашому випадку було додано запис (рис. 3.8), ідентичний першому, але з ID, більшим на 1. Це відбулося завдяки налаштуванню автоінкремента для первинного ключа в таблиці бази даних, що автоматично збільшує значення ID для кожного нового запису.

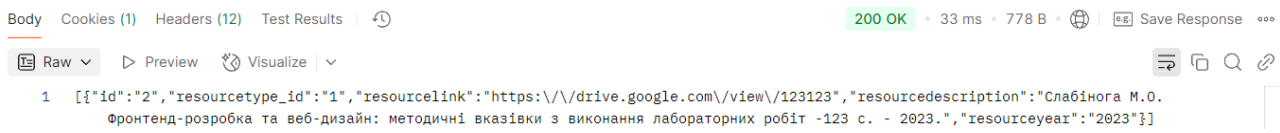

Рисунок 3.8 – Результат виконання POST-запиту

Далі виконуємо PUT-запит для редагування конкретного запису в базі даних на сервері.

Продовження JSON-коду:

```
{            "name": "Edit learning resource by ID",
            "request": {                "method": "PUT",
                "header": [],                "body": {
                        "mode": "raw",                        "raw":
"{\n    \"id\":1,\n    \"resourcetype_id\":\"1\",\n    \"resourcelink\":
\"https://drive.google.com/view/123123\",\n    \"resourcedescription\":\
"Updated
description\",\n    \"resourceyear\":\"2023\",\n    \"authors\":[123,423
,2]\n}",
                        "options": { "raw": { "language": "json" }
}            },
                "url": {                "raw": "{{apiUrl}}/learning-
resources",
"host": ["{{apiUrl}}"],                        "path": ["learning-
resources"]    }        }        },
```

Edit learning resource by ID:

– метод: PUT;

– опис: редагування наявного навчального ресурсу;

– URL: {{apiUrl}}/learning-resources;

– тіло запиту (JSON):

```
{
    "id": 1,
    "resourcetype_id": "1",
    "resourcelink": "https://drive.google.com/view/123123",
    "resourcedescription": "Updated description",
    "resourceyear": "2023",
    "authors": [123, 423, 2]
}
```

Після виконання запиту (рис. 3.9) було відредаговано запис із ID = 1. Зокрема, змінено значення поля resourcedescription на "Updated description".

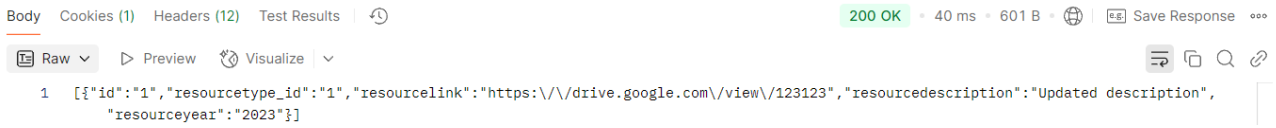

Рисунок 3.9 – Результат виконання PUT-запиту

І нарешті виконуємо DELETE-запит для видалення конкретного запису в таблиці.

Продовження JSON-коду:

```
{        "name": "Delete learning resource by ID",
        "request": {            "method": "DELETE",
            "header": [],            "body": {
            "mode": "raw",            "raw": "{\"id\": 3}"          },
            "url": {            "raw": "{{apiUrl}}/learning-resources",
              "host": ["{{apiUrl}}"],            "path": ["learning-
resources"]            }        }        }    ]}
```

Цей JSON-код представляє запит для видалення навчального ресурсу за його ідентифікатором (ID). Він описує HTTP-запит методом DELETE на певну URL-адресу {{apiUrl}}/learning-resources. У тілі запиту вказується ID ресурсу, який необхідно видалити, у форматі JSON: {"id": 3}.

Цей запит видаляє ресурс із бази даних за допомогою зазначеної URL-адреси та методу DELETE.

Після виконання цього запиту запис із ID = 3 було успішно видалено.

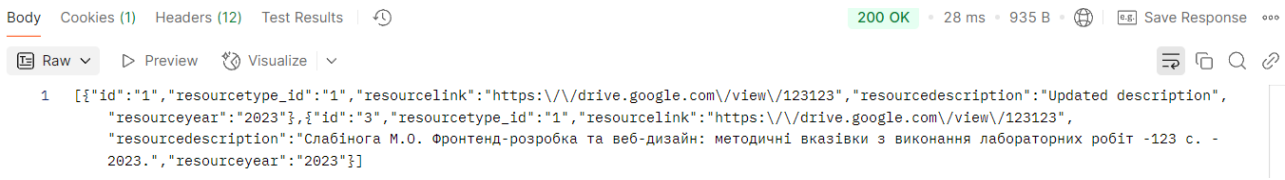

Рисунок 3.10 – Список методичного забезпечення до видалення

На рисунку 3.10 демонструється повний список записів методичного забезпечення до виконання DELETE-запиту. Серед записів присутній ресурс із ID = 3, який має бути видалений.

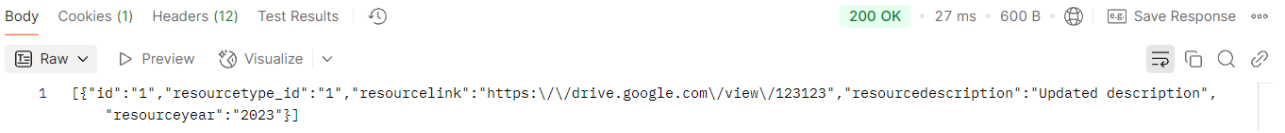

Рисунок 3.11 – Список методичного забезпечення після видалення

Рисунок 3.11 показує оновлений список записів після виконання DELETE-запиту. Помітно, що запис із ID = 3 відсутній, що підтверджує успішне видалення ресурсу з бази даних.

Рисунок 3.12 – Результат виконання DELETE-запиту

На рисунку 3.12 відображається відповідь сервера на DELETE-запит, що містить інформацію про успішне видалення (статус відповіді 200 та повідомлення про успіх). Це свідчить про коректну роботу методу DELETE у системі.

Перевірка специфічних запитів через Postman

Для забезпечення повноцінності функціоналу розробленої компоненти було додано такі спеціальні запити:

1. Отримання методичного забезпечення за дисципліною за останні 5 років;

Запит:

– метод: GET;

– URL: `{{apiUrl}}/learning-resources?action=getByCourseIdLast5Years&course_id=1`;

– параметри запиту:

– `action`: getByCourseIdLast5Years;

– `course_id`: 1.

Цей запит забезпечує вибірку всіх методичних ресурсів, що використовуються у заданій дисципліні, та були створені протягом останніх 5 років. Завдяки умові по року запиту повертаються лише свіжі записи, що значно полегшує аналіз актуальної інформації з курсу.

Продовження JSON-коду:

```
{    "name": "Get learning resources by course id last 5 years",
    "request": {
        "method": "GET",
        "header": [],
        "url": {
            "raw": "{{apiUrl}}/learning-resources?action=getByCourseIdLast5Years&course_id=1",
            "host": ["{{apiUrl}}"],
            "path": ["learning-resources"],
            "query": [
```

```
                    {        "key":        "action",        "value":
"getByCourseIdLast5Years" },
                    { "key": "course_id", "value": "1" }
                ]       }    }}
```

На рисунку 3.13 видно результат виконання першого запиту, який був спрямований на отримання всіх навчальних ресурсів. Відповідь повертається у форматі JSON і містить наступні поля:

– id: "1" – унікальний ідентифікатор ресурсу;

– resourcetype_id: "1" – позначає тип ресурсу (зв’язок з відповідною таблицею типів);

– resourcelink: "https://drive.google.com/view/123123" – посилання на ресурс;

– resourcedescription: містить опис ресурсу, наприклад, "Слабінога М.О. Фронтенд-розробка та веб-дизайн: методичні вказівки з виконання лабораторних робіт -123 с. - 2023.";

– resourceyear: "2023" – рік публікації або створення ресурсу.

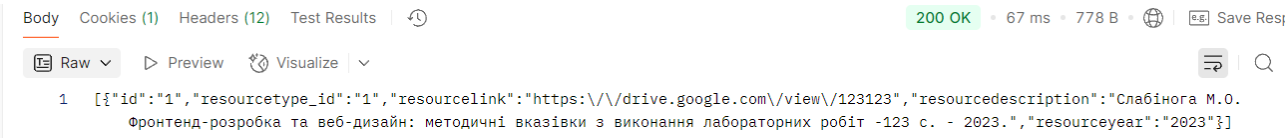

Рисунок 3.13 – Результат виконання запиту для отримання методичного забезпечення за дисципліною за останні 5 років

Відповідь показує, що запит виконано успішно (статус 200 OK) і сервер повернув дані за запитом всіх навчальних ресурсів, що підтверджує коректну роботу API.

2. Отримання методичного забезпечення за ідентифікатором автора;

Запит:

– метод: GET;

– URL: `{{apiUrl}}/learning-resources?action=getByAuthorId&author_id=1`;

– параметри запиту:

– `action`: getByAuthorId;

– `author_id`: 1.

Продовження JSON-коду:

```
{    "name": "Get learning resources by author id",
    "request": {
        "method": "GET",
        "header": [],
        "url": {
            "raw":                                "{{apiUrl}}/learning-
resources?action=getByAuthorId&author_id=1",
            "host": ["{{apiUrl}}"],
            "path": ["learning-resources"],
            "query": [
                { "key": "action", "value": "getByAuthorId" },
                { "key": "author_id", "value": "1" }
            ]        }    }}
```

Цей запит повертає всі записи з методичними ресурсами, які пов’язано з певним автором, ідентифікатор якого відповідає зазначеному значенню. Запит не використовує умов часових обмежень, тому повернеться повний перечень записів для встановленого автора.

Відповідь містить JSON-масив з даними про кожний методичний ресурс цього автора (рис. 3.14).

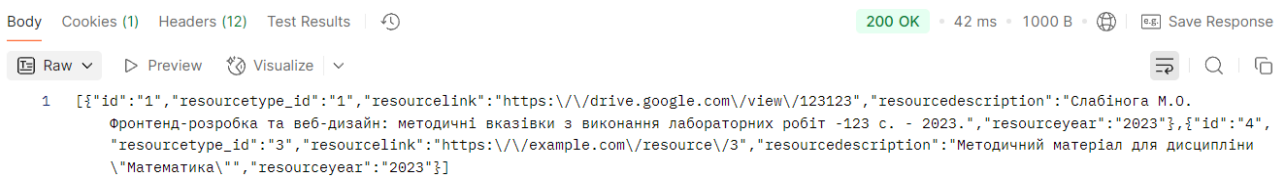

Рисунок 3.14 – Результат виконання запиту для вибірки ресурсів за id автора

3. Отримання методичного забезпечення за ідентифікатором автора за останні 5 років;

Запит:

– метод: GET;

– URL: `{{apiUrl}}/learning-resources?action=getByAuthorIdLast5Years&author_id=1`;

– параметри запиту:

  – `action`: getByAuthorIdLast5Years;

  – `author_id`: 1.

Продовження JSON-коду:

```
{           "name": "Get learning resources by author id last 5 years",
          "request": {
              "method": "GET",
              "header": [],
              "url": {
                  "raw":                              "{{apiUrl}}/learning-resources?action=getByAuthorIdLast5Years&author_id=1",
                  "host": ["{{apiUrl}}"],
                  "path": ["learning-resources"],
                  "query": [
                      {          "key":          "action",          "value": "getByAuthorIdLast5Years" },
                      { "key": "author_id", "value": "1" }
                  ]         }         }}
```

Подібно до попереднього запиту, цей виконує вибірку ресурсів за конкретним автором, але додає додаткову умовну перевірку: повертаються лише ті записи, які були створені або оновлені протягом останніх 5 років. Це дозволяє отримати актуальну інформацію про діяльність автора.

Результатом є JSON-масив із записами, що відповідають встановленому часовому інтервалу (рис. 3.15).

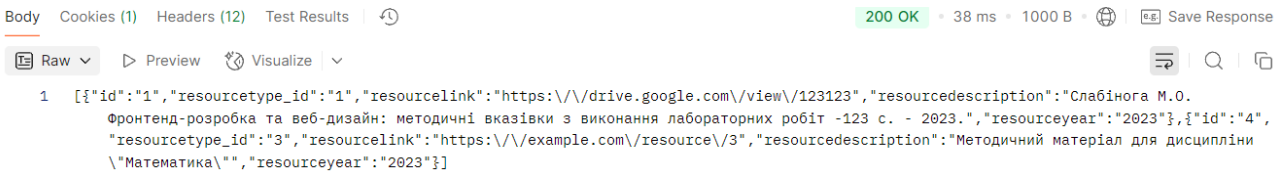

Рисунок 3.15 – Результат виконання запиту для отримання ресурсів за id автора за останні 5 років

4. Отримання посібників і підручників для автора за останні 5 років.

Запит:

– метод: GET;

– URL: `{{apiUrl}}/learning-resources?action=getManualsTextbooksByAuthorLast5Years&author_id=1`;

– параметри запиту:

– `action`: getManualsTextbooksByAuthorLast5Years;

– `author_id`: 1.

Кінець JSON-коду:

```
{    "name": "Get manuals & textbooks by author last 5 years",
    "request": {
        "method": "GET",
        "header": [],
        "url": {
            "raw": "{{apiUrl}}/learning-resources?action=getManualsTextbooksByAuthorLast5Years&author_id=1",
            "host": ["{{apiUrl}}"],
            "path": ["learning-resources"],
            "query": [
                { "key": "action", "value": "getManualsTextbooksByAuthorLast5Years" },
                { "key": "author_id", "value": "1" }
            ]        }    }}
```

Цей запит виконує спеціалізовану фільтрацію ресурсів певного автора, при цьому повертаються лише ті записи, які відносяться до категорій «Підручник» і

«Посібник» і відповідають умовам вибірки за останні 5 років. Завдяки цьому забезпечується цільова вибірка, корисна для аналізу навчальних матеріалів, що відповідають специфічним завданням.

Рисунок 3.16 – Результат виконання запиту для отримання посібників і підручників за id автора за останні 5 років

Сервер повертає JSON-масив, де кожен об’єкт представляє собою дані про методичний ресурс типу «Підручник» або «Посібник», створений протягом останніх 5 років (рис. 3.16).

Таким чином, виконання HTTP-запитів за допомогою Postman демонструє правильну роботу реалізованого CRUD-функціоналу для компоненти «Методичне забезпечення дисциплін». Ця перевірка забезпечує інтеграцію серверної частини з інформаційною системою університету, а використання Postman за допомогою глобальних змінних (apiUrl та BearerToken) дозволяє ефективно тестувати всю логіку та забезпечити коректну автентифікацію через Bearer Token згідно з протоколом OAuth 2.0.

ВИСНОВКИ

У результаті виконання поставлених завдань було реалізовано серверну компоненту «Методичне забезпечення дисциплін» для інформаційної системи університету. Ця компонента стала вагомим кроком у напрямку централізованого зберігання, представлення та керування навчальними ресурсами, а також відзначилась високим ступенем інтеграції з існуючими частинами системи.

Перш за все, було спроєктовано структуру бази даних, яка охоплює всі необхідні атрибути методичних матеріалів – включно з назвою, типом, описом, роком видання, переліком авторів та посиланням на ресурс. Базу було нормалізовано до третьої нормальної форми, що гарантує оптимальне зберігання та узгодженість даних.

Особливу увагу приділено налагодженню логічних зв’язків між таблицями всередині компоненти, зокрема:

– LearningResource має зв’язки з LearningResourceType, CourseResource, LearningResourceAuthor;

– реалізовано таблицю-зв’язку між ресурсом і автором, що дозволяє обробляти багатокористувацькі матеріали;

– структура підтримує класифікацію за типами забезпечення та прив’язку до конкретних дисциплін.

Крім того, було встановлено міжкомпонентні зв’язки з іншими модулями системи, що вже були розроблені раніше:

– методичні ресурси прив’язуються до дисциплін через таблицю Course (з компоненти «Навчальні дисципліни»);

– автори ресурсів ідентифікуються через User (компонента «Користувачі та підрозділи»).

Це дозволило досягти високого рівня узгодженості та масштабованості всієї системи.

На базі розробленої структури було створено програмну реалізацію всіх необхідних операцій. Зокрема:

– реалізовано набір CRUD-функцій у відповідному сервісному шарі (LearningResourceService), що забезпечує додавання, оновлення, видалення та перегляд записів;

– створено додаткові методи вибірки, які дозволяють швидко знаходити матеріали за дисципліною чи типом;

– вся логіка організована у вигляді серверної архітектури:

- routes – організовують маршрутизацію HTTP-запитів;
- controllers – обробляють запити та викликають бізнес-логіку;
- services – інкапсулюють взаємодію з базою даних;
- models – відображають структуру таблиць як об’єкти PHP-класів.

Після реалізації функціоналу було проведено перевірку працездатності розробленої системи:

– у браузері – для перевірки коректності структури відповідей і базової взаємодії;

– у Postman – для глибокої перевірки всіх CRUD-операцій, параметрів запитів, а також відповіді сервера, включно з автентифікацією.

Результатом стало створення ефективної, масштабованої та інтегрованої компоненти, яка повністю відповідає вимогам, зазначеним у початковому плані. Усі етапи – від проектування схеми до налаштування зв’язків і перевірки – були реалізовані послідовно, якісно та з урахуванням кращих практик розробки серверної логіки.

Таким чином, було досягнуто головну мету роботи: створено серверну частину інформаційної системи університету для централізованого керування методичним забезпеченням дисциплін, що суттєво покращує організацію освітнього процесу. Розроблена компонента вже зараз готова до впровадження та може слугувати основою для подальшого розширення – наприклад, через реалізацію розширеного пошуку або додавання функцій завантаження файлів.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Навчально-методичний комплекс дисципліни. URL: https://uk.wikipedia.org/wiki/%D0%9D%D0%B0%D0%B2%D1%87%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE-%D0%BC%D0%B5%D1%82%D0%BE%D0%B4%D0%B8%D1%87%D0%BD%D0%B8%D0%B9_%D0%BA%D0%BE%D0%BC%D0%BF%D0%BB%D0%B5%D0%BA%D1%81_%D0%B4%D0%B8%D1%81%D1%86%D0%B8%D0%BF%D0%BB%D1%96%D0%BD%D0%B8 (дата звернення: 2 березня 2025р.).

2. Закон України “Про вищу освіту”. URL: https://ips.ligazakon.net/document/T141556?an=1 (дата звернення: 3 березня 2025р.).

3. Закон України “Про захист персональних даних”. URL: https://zakon.rada.gov.ua/laws/show/2297-17#Text (дата звернення: 3 березня 2025р.).

4. ДСТУ ISO/IEC 27001:2015 – Системи управління інформаційною безпекою. URL: https://online.budstandart.com/ua/catalog/doc-page.html?id_doc=66910 (дата звернення: 3 березня 2025р.).

5. Про обробку інформації в програмно-апаратному комплексі «Система електронного документообігу Міністерства освіти і науки України». URL: https://mon.gov.ua/npa/pro-obrobku-informaciyi-v-programno-aparatnomu-kompleksi-sistema-elektronnogo-dokumentoobigu-ministerstva-osviti-i-nauki-ukrayini (дата звернення: 3 березня 2025р.).

6. Moodle. URL: https://uk.wikipedia.org/wiki/Moodle (дата звернення: 5 березня 2025р.).

7. Anthology Inc. URL: https://en.wikipedia.org/wiki/Anthology_Inc. (дата звернення: 5 березня 2025р.).

8. Google Classroom. URL: https://uk.wikipedia.org/wiki/Google_Classroom (дата звернення: 5 березня 2025р.).

9. Навчально-методичне забезпечення. URL: https://iat.kpi.ua/teaching-and-methodological-support-ua/ (дата звернення: 6 березня 2025р.).

10. The Role of Institutional Repositories in Higher Education: Purpose and Level of Openness. URL: https://link.springer.com/chapter/10.1007/978-3-658-38703-7_4 (дата звернення: 7 березня 2025р.).

11. Brief Information about Institutional Repository. URL: https://www.lisedunetwork.com/brief-information-institutional-repository/ (дата звернення: 7 березня 2025р.).

12. PHP. URL: https://en.wikipedia.org/wiki/PHP (дата звернення: 9 березня 2025р.).

13. Visual Studio Code. URL: https://en.wikipedia.org/wiki/Visual_Studio_Code (дата звернення: 9 березня 2025р.).

14. XAMPP. URL: https://uk.wikipedia.org/wiki/XAMPP (дата звернення: 9 березня 2025р.).

15. МійКлас. Ключі. Зв’язки між записами і таблицями URL: https://www.miyklas.com.ua/p/informatica/10-klas/sistemi-keruvannia-bazami-danikh-326161/kliuchi-zv-iazki-tablitci-326454/re-3a0a4ef3-824d-4761-9bb3-db724525d1e3 (дата звернення: 11 березня 2025р.).

16. MySQL. URL: https://uk.wikipedia.org/wiki/MySQL (дата звернення: 19 березня 2025р.).

17. Про затвердження Ліцензійних умов провадження освітньої діяльності URL: https://zakon.rada.gov.ua/laws/show/1187-2015-%D0%BF#Text (дата звернення: 13 квітня 2025р.).

18. HTTP методи запиту. URL: https://w3schoolsua.github.io/tags/ref_httpmethods.html#gsc.tab=0 (дата звернення: 14 квітня 2025р.).

19. Postman (software). URL: https://en.wikipedia.org/wiki/Postman_(software) (дата звернення: 15 квітня 2025р.).

20. Протокол передачі. Протокол Bearer. URL:

https://www.vpnunlimited.com/ua/help/cybersecurity/bearer-protocol?srsltid=AfmBOooGyXCBidy9OFa3gCStsGvsD6ppiCA7xOysmmwwm4BAouQL6gMs (дата звернення: 16 квітня 2025р.).

ДОДАТКИ

Код компоненти “Методичне забезпечення дисциплін” серверної частини інформаційної системи університету

BaseModel.php

```
<?php
    abstract class BaseModel {
        protected $id;
        public abstract function __construct($paramArray);
        public function getId(){
            return $this->id;
        }

        public function getAsJSON(){
           echo json_encode($this->getAsObject(),JSON_UNESCAPED_UNICODE);
        }
      }
?>
```

CourseResource.php

```
<?php
require_once('BaseModel.php');
class CourseResource extends BaseModel
{

    private $resource_id;
    private $course_id;
    private $courseresourcetype_id;

    public function __construct($paramArray)
    {
        $this->id = $paramArray['id'];
        $this->resource_id = $paramArray['resource_id'];
        $this->course_id = $paramArray['course_id'];
                                    $this->courseresourcetype_id        =
$paramArray['courseresourcetype_id'];
    }

    public function getAsObject()
    {
        return get_object_vars($this);
    }
}
```

CourseResourceType.php

```
<?php
require_once('BaseModel.php');
class CourseResourceType extends BaseModel
{

    private $name;

    public function __construct($paramArray)
```

```
    {
        $this->id = $paramArray['id'];
        $this->name = $paramArray['name'];
    }
    public function getAsObject()
    {
        return get_object_vars($this);
    }
}
```

LearningResource.php

```
<?php
require_once('BaseModel.php');
class LearningResource extends BaseModel
{

    private $resourcetype_id;
    private $resourcelink;
    private $resourcedescription;
    private $resourceyear;

    public function __construct($paramArray)
    {
        $this->id = $paramArray['id'];
        $this->resourcetype_id = $paramArray['resourcetype_id'];
        $this->resourcelink = $paramArray['resourcelink'];
        $this->resourcedescription = $paramArray['resourcedescription'];
        $this->resourceyear = $paramArray['resourceyear'];
    }

    public function getAsObject()
    {
        return get_object_vars($this);
    }
}
```

LearningResourceAuthor.php

```
<?php
require_once('BaseModel.php');
class LearningResourceAuthor extends BaseModel
{

    private $resource_id;
    private $author_id;

    public function __construct($paramArray)
    {
        $this->id = $paramArray['id'];
        $this->resource_id = $paramArray['resource_id'];
        $this->author_id = $paramArray['author_id'];
    }

    public function getAsObject()
    {
        return get_object_vars($this);
    }
}
```

LearningResourceType.php

```
<?php
require_once('BaseModel.php');
class LearningResourceType extends BaseModel
{
    private $name;

    public function __construct($paramArray)
    {
        $this->id = $paramArray['id'];
        $this->name = $paramArray['name'];
    }

    public function getAsObject()
    {
        return get_object_vars($this);
    }
}
```

BaseService.php

```
<?php
abstract class BaseService
{
    protected $index;
    protected $dataArray;
    public function __construct()
    {
        $this->dataArray = [];
        $this->index = 0;
    }
    public function getAsJSON()
    {
        $jsonArray = [];
        for ($i = 0; $i < count($this->dataArray); $i++) {
            array_push($jsonArray, $this->dataArray[$i]->getAsObject());
        }
        return json_encode($jsonArray, JSON_UNESCAPED_UNICODE);
    }
    public function getAsArray()
    {
        $jsonArray = [];
        for ($i = 0; $i < count($this->dataArray); $i++) {
            array_push($jsonArray, $this->dataArray[$i]->getAsObject());
        }
        return $jsonArray;
    }
    public abstract function insertToDatabase($conn, $paramArray);
    public abstract function deleteFromDatabase($conn, $id);
    public abstract function updateDatabaseById($conn, $id, $paramArray);
    public abstract function getFromDatabaseById($conn, $id);
    public abstract function getAllFromDataBase($conn);
}
```

CourseResourceService.php

```
<?php
require_once('./app/services/BaseService.php');
require_once('./app/models/CourseResource.php');
```

```
class CourseResourceService extends BaseService
{
    static $tableName = 'am_course_resource';

    public function add($paramArray)
{
        if (!isset($paramArray['id'])) {
            $paramArray['id'] = ++$this->index;
        }
        $courseResource = new CourseResource($paramArray);
        array_push($this->dataArray, $courseResource);
        return $paramArray['id'];
    }

    public function insertToDatabase($conn, $paramArray)
    {
         $request = $conn->prepare("INSERT INTO " . $this::$tableName . "
VALUES (DEFAULT, ?, ?, ?)");
                $request->bind_param("iii",   $resource_id,   $course_id,
$courseresourcetype_id);
        $resource_id = $paramArray['resource_id'];
        $course_id = $paramArray['course_id'];
        $courseresourcetype_id = $paramArray['courseresourcetype_id'];
        $request->execute();
        $request->close();
    }

    public function deleteFromDatabase($conn, $id)
    {
         $request = $conn->prepare("DELETE FROM " . $this::$tableName . "
WHERE id=?");
        $request->bind_param("i", $id);
        $request->execute();
        $request->close();
    }

    public function updateDatabaseById($conn, $id, $paramArray)
    {
         $request = $conn->prepare("UPDATE " . $this::$tableName . " SET
`resource_id`=?, `course_id`=?, `courseresourcetype_id`=? WHERE `id`=?");
                $request->bind_param("iiii",   $resource_id,   $course_id,
$courseresourcetype_id, $id);
        $resource_id = $paramArray['resource_id'];
        $course_id = $paramArray['course_id'];
        $courseresourcetype_id = $paramArray['courseresourcetype_id'];
        $request->execute();
        $request->close();
    }

    public function getFromDatabaseById($conn, $id)
    {
        $request = "SELECT * FROM " . $this::$tableName . " WHERE id=" .
$id;
        $result = $conn->query($request);
        if ($result->num_rows > 0) {
            while ($row = $result->fetch_assoc()) {
```

```
                $this->add($row);
            }
        } else {
            echo "";
        }
}

    public function getAllFromDataBase($conn)
    {
        $request = "SELECT * FROM " . $this::$tableName . " ORDER BY id
ASC";
        $result = $conn->query($request);
        if ($result->num_rows > 0) {
            while ($row = $result->fetch_assoc()) {
                $this->add($row);
            }
        } else {
            echo "";
        }
    }
}
```

CourseResourceTypeService.php

```
<?php
require_once('./app/services/BaseService.php');
require_once('./app/models/CourseResourceType.php');

class CourseResourceTypeService extends BaseService
{
    static $tableName = 'am_course_resource_type';

    public function add($paramArray)
    {
        if (!isset($paramArray['id'])) {
            $paramArray['id'] = ++$this->index;
        }
        $courseResourceType = new CourseResourceType($paramArray);
        array_push($this->dataArray, $courseResourceType);
        return $paramArray['id'];
    }

    public function insertToDatabase($conn, $paramArray)
    {
        $request = $conn->prepare("INSERT INTO " . $this::$tableName . "
VALUES (DEFAULT, ?)");
        $request->bind_param("s", $name);
        $name = $paramArray['name'];
        $request->execute();
        $request->close();
    }

    public function deleteFromDatabase($conn, $id)
    {
        $request = $conn->prepare("DELETE FROM " . $this::$tableName . "
WHERE id=?");
        $request->bind_param("i", $id);
```

```
        $request->execute();
        $request->close();
    }

    public function updateDatabaseById($conn, $id, $paramArray)
    {
   $request = $conn->prepare("UPDATE " . $this::$tableName . " SET `name`=?
WHERE `id`=?");
        $request->bind_param("si", $name, $id);
        $name = $paramArray['name'];
        $request->execute();
        $request->close();
    }

    public function getFromDatabaseById($conn, $id)
    {
        $request = "SELECT * FROM " . $this::$tableName . " WHERE id=" .
$id;
        $result = $conn->query($request);
        if ($result->num_rows > 0) {
            while ($row = $result->fetch_assoc()) {
                $this->add($row);
            }
        } else {
            echo "";
        }
    }

    public function getAllFromDataBase($conn)
    {
         $request = "SELECT * FROM " . $this::$tableName . " ORDER BY id
ASC";
        $result = $conn->query($request);
        if ($result->num_rows > 0) {
            while ($row = $result->fetch_assoc()) {
                $this->add($row);
            }
        } else {
            echo "";
        }
    }
}
```

LearningResourceService.php

```
<?php
require_once('./app/services/BaseService.php');
require_once('./app/models/LearningResource.php');

class LearningResourceService extends BaseService
{
    static $tableName = 'am_learning_resource';

    public function add($paramArray)
    {
        if (!isset($paramArray['id'])) {
            $paramArray['id'] = ++$this->index;
```

```
        }
        $learningResource = new LearningResource($paramArray);
        array_push($this->dataArray, $learningResource);
        return $paramArray['id'];
    }

    public function insertToDatabase($conn, $paramArray)
{
        $request = $conn->prepare("INSERT INTO " . $this::$tableName . "
VALUES (DEFAULT, ?, ?, ?, ?)");
          $request->bind_param("isss", $resourcetype_id, $resourcelink,
$resourcedescription, $resourceyear);
        $resourcetype_id = $paramArray['resourcetype_id'];
        $resourcelink = $paramArray['resourcelink'];
        $resourcedescription = $paramArray['resourcedescription'];
        $resourceyear = $paramArray['resourceyear'];
        $request->execute();
        $request->close();
    }

    public function deleteFromDatabase($conn, $id)
    {
        $request = $conn->prepare("DELETE FROM " . $this::$tableName . "
WHERE id=?");
        $request->bind_param("i", $id);
        $request->execute();
        $request->close();
    }

    public function updateDatabaseById($conn, $id, $paramArray)
    {
         $request = $conn->prepare("UPDATE " . $this::$tableName . " SET
`resourcetype_id`=?,      `resourcelink`=?,      `resourcedescription`=?,
`resourceyear`=? WHERE `id`=?");
          $request->bind_param("isssi", $resourcetype_id, $resourcelink,
$resourcedescription, $resourceyear, $id);
        $resourcetype_id = $paramArray['resourcetype_id'];
        $resourcelink = $paramArray['resourcelink'];
        $resourcedescription = $paramArray['resourcedescription'];
        $resourceyear = $paramArray['resourceyear'];
        $request->execute();
        $request->close();
    }

    public function getFromDatabaseById($conn, $id)
    {
        $request = "SELECT * FROM " . $this::$tableName . " WHERE id=" .
$id;
        $result = $conn->query($request);
        if ($result->num_rows > 0) {
            while ($row = $result->fetch_assoc()) {
                $this->add($row);
            }
        } else {
            echo "";
        }
```

```
    }

    public function getAllFromDataBase($conn)
    {
         $request = "SELECT * FROM " . $this::$tableName . " ORDER BY id
ASC";
        $result = $conn->query($request);
        if ($result->num_rows > 0) {
            while ($row = $result->fetch_assoc()) {
                $this->add($row);
            }
        } else {
            echo "";
        }
    }

    public function getByCourseIdLast5Years($conn, $courseId)
    {
        $yearLimit = date("Y") - 5;
        $query = "SELECT DISTINCT lr.*
              FROM " . $this::$tableName . " lr
              JOIN am_course_resource cr ON lr.id = cr.resource_id
               WHERE cr.course_id = " . $courseId . " AND lr.resourceyear
>= " . $yearLimit;

        $result = $conn->query($query);
        if ($result && $result->num_rows > 0) {
            while ($row = $result->fetch_assoc()) {
                $this->add($row);
            }
        } else {
            echo "";
        }
    }

    public function getByAuthorId($conn, $authorId)
    {
        $query = "SELECT lr.*
              FROM " . $this::$tableName . " lr
                     JOIN  am_learning_resource_author  lra  ON  lr.id  =
lra.resource_id
              WHERE lra.author_id = " . $authorId;

        $result = $conn->query($query);
        if ($result && $result->num_rows > 0) {
            while ($row = $result->fetch_assoc()) {
                $this->add($row);
            }
        } else {
            echo "";
        }
    }

    public function getByAuthorIdLast5Years($conn, $authorId)
    {
        $yearLimit = date("Y") - 5;
```

```
        $query = "SELECT lr.*
              FROM " . $this::$tableName . " lr
                    JOIN  am_learning_resource_author  lra  ON  lr.id  =
lra.resource_id
              WHERE lra.author_id = " . $authorId . " AND lr.resourceyear
>= " . $yearLimit;

        $result = $conn->query($query);
        if ($result && $result->num_rows > 0) {
            while ($row = $result->fetch_assoc()) {
                $this->add($row);
            }
        } else {
            echo "";
        }
    }
         public   function   getManualsTextbooksByAuthorLast5Years($conn,
$authorId)
    {
        $yearLimit = date("Y") - 5;
        $query = "SELECT lr.*
              FROM " . $this::$tableName . " lr
                    JOIN  am_learning_resource_author  lra  ON  lr.id  =
lra.resource_id
               JOIN am_learning_resource_type lrt ON lr.resourcetype_id =
lrt.id
              WHERE lra.author_id = " . $authorId . "
                AND lr.resourceyear >= " . $yearLimit . "
                AND lrt.name IN ('Підручник', 'Посібник')";

        $result = $conn->query($query);
        if ($result && $result->num_rows > 0) {
            while ($row = $result->fetch_assoc()) {
                $this->add($row);
            }
        } else {
            echo "";
        }
    }
}
```

LearningResourceAuthorService.php

```
<?php
require_once('./app/services/BaseService.php');
require_once('./app/models/LearningResourceAuthor.php');

class LearningResourceAuthorService extends BaseService
{
    static $tableName = 'am_learning_resource_author';

    public function add($paramArray)
    {
        if (!isset($paramArray['id'])) {
            $paramArray['id'] = ++$this->index;
        }
       $learningResourceAuthor = new LearningResourceAuthor($paramArray);
```

```
        array_push($this->dataArray, $learningResourceAuthor);
        return $paramArray['id'];
    }

    public function insertToDatabase($conn, $paramArray)
    {
        $request = $conn->prepare("INSERT INTO " . $this::$tableName . "
VALUES (DEFAULT, ?, ?)");
        $request->bind_param("ii", $resource_id, $author_id);
        $resource_id = $paramArray['resource_id'];
        $author_id = $paramArray['author_id'];
        $request->execute();
        $request->close();
    }

    public function deleteFromDatabase($conn, $id)
    {
        $request = $conn->prepare("DELETE FROM " . $this::$tableName . "
WHERE id=?");
        $request->bind_param("i", $id);
        $request->execute();
        $request->close();
    }

    public function updateDatabaseById($conn, $id, $paramArray)
    {
         $request = $conn->prepare("UPDATE " . $this::$tableName . " SET
`resource_id`=?, `author_id`=? WHERE `id`=?");
        $request->bind_param("iii", $resource_id, $author_id, $id);
        $resource_id = $paramArray['resource_id'];
        $author_id = $paramArray['author_id'];
        $request->execute();
        $request->close();
    }

    public function getFromDatabaseById($conn, $id)
    {
        $request = "SELECT * FROM " . $this::$tableName . " WHERE id=" .
$id;
        $result = $conn->query($request);
        if ($result->num_rows > 0) {
            while ($row = $result->fetch_assoc()) {
                $this->add($row);
            }
        } else {
            echo "";
        }
    }

    public function getAllFromDataBase($conn)
    {
         $request = "SELECT * FROM " . $this::$tableName . " ORDER BY id
ASC";
        $result = $conn->query($request);
        if ($result->num_rows > 0) {
            while ($row = $result->fetch_assoc()) {
```

```
                $this->add($row);
            }
        } else {
            echo "";
        }
    }
}
```

LearningResourceTypeService.php

```
<?php
require_once('./app/services/BaseService.php');
require_once('./app/models/LearningResourceType.php');

class LearningResourceTypeService extends BaseService
{
    static $tableName = 'am_learning_resource_type';

    public function add($paramArray)
    {
        if (!isset($paramArray['id'])) {
            $paramArray['id'] = ++$this->index;
        }
        $learningResourceType = new LearningResourceType($paramArray);
        array_push($this->dataArray, $learningResourceType);
        return $paramArray['id'];
    }

    public function insertToDatabase($conn, $paramArray)
    {
        $request = $conn->prepare("INSERT INTO " . $this::$tableName . "
VALUES (DEFAULT, ?)");
        $request->bind_param("s", $name);
        $name = $paramArray['name'];
        $request->execute();
        $request->close();
    }

    public function deleteFromDatabase($conn, $id)
    {
        $request = $conn->prepare("DELETE FROM " . $this::$tableName . "
WHERE id=?");
        $request->bind_param("i", $id);
        $request->execute();
        $request->close();
    }

    public function updateDatabaseById($conn, $id, $paramArray)
    {
        $request = $conn->prepare("UPDATE " . $this::$tableName . " SET
`name`=? WHERE `id`=?");
        $request->bind_param("si", $name, $id);
        $name = $paramArray['name'];
        $request->execute();
        $request->close();
    }
```

```
    public function getFromDatabaseById($conn, $id)
    {
        $request = "SELECT * FROM " . $this::$tableName . " WHERE id=" .
$id;
        $result = $conn->query($request);
        if ($result->num_rows > 0) {
            while ($row = $result->fetch_assoc()) {
                $this->add($row);
            }
     } else {
            echo "";
        }
    }

    public function getAllFromDataBase($conn)
    {
        $request = "SELECT * FROM " . $this::$tableName . " ORDER BY id
ASC";
        $result = $conn->query($request);
        if ($result->num_rows > 0) {
            while ($row = $result->fetch_assoc()) {
                $this->add($row);
            }
        } else {
            echo "";
        }
    }
}
```

BaseController.php

```
<?php
class BaseController
{
    private $list;
    public $conn;
    public function __construct($conn, $list)
    {
        $this->list = $list;
        $this->conn = $conn;
    }
    public function processRequest()
    {
        if ($_SERVER['REQUEST_METHOD'] == 'GET') {
            header("HTTP/1.1 200 OK");
            if (isset($_REQUEST['id'])) {
                        $this->list->getFromDatabaseById($this->conn,
$_REQUEST['id']);
                echo $this->list->getAsJSON();
            } else {
                $this->list->getAllFromDataBase($this->conn);
                echo $this->list->getAsJSON();
            }
        } else if ($_SERVER['REQUEST_METHOD'] == 'POST') {
                                                    $data        =
get_object_vars(json_decode(file_get_contents('php://input')));
            $this->list->insertToDatabase($this->conn, $data);
```

```
            $this->list->getFromDatabaseById($this->conn, $this->conn-
>insert_id);
            header("HTTP/1.1 200 OK");
            echo $this->list->getAsJSON();
        } else if ($_SERVER['REQUEST_METHOD'] == 'PUT') {
                                                        $data        =
get_object_vars(json_decode(file_get_contents('php://input')));
             $this->list->updateDatabaseById($this->conn, $data['id'],
$data);
            $this->list->getFromDatabaseById($this->conn, $data['id']);
            header("HTTP/1.1 200 OK");
            echo $this->list->getAsJSON();
        } else if ($_SERVER['REQUEST_METHOD'] == 'DELETE') {
                                                        $data        =
get_object_vars(json_decode(file_get_contents('php://input')));
            $this->list->deleteFromDatabase($this->conn, $data['id']);
            header("HTTP/1.1 200 OK");
            echo '{"status":"OK"}';
        }
    }
}
```

CourseResourceController.php

```
<?php
require_once('./app/services/CourseResourceService.php');
require_once('./app/controllers/BaseController.php');

$courseResourceService = new CourseResourceService();

$controller        =        new        BaseController($AMDBConnection,
$courseResourceService);
$controller->processRequest();
```

CourseResourceTypeController.php

```
<?php
require_once('./app/services/CourseResourceTypeService.php');
require_once('./app/controllers/BaseController.php');

$courseResourceTypeService = new CourseResourceTypeService();

$controller        =        new        BaseController($AMDBConnection,
$courseResourceTypeService);
$controller->processRequest();
```

LearningResourceController.php

```
<?php
require_once("./app/services/LearningResourceService.php");
require_once("./app/controllers/BaseController.php");

$learningResourceService = new LearningResourceService();

if (!isset($_REQUEST['action'])) {
    $controller = new BaseController($AMDBConnection,
$learningResourceService);
    $controller->processRequest();
} elseif ($_REQUEST['action'] == 'getByCourseIdLast5Years') {
    $learningResourceService->getByCourseIdLast5Years($AMDBConnection,
$_REQUEST['course_id']);
```

```
    echo $learningResourceService->getAsJSON();
} elseif ($_REQUEST['action'] == 'getByAuthorId') {
    $learningResourceService->getByAuthorId($AMDBConnection,
$_REQUEST['author_id']);
    echo $learningResourceService->getAsJSON();
} elseif ($_REQUEST['action'] == 'getByAuthorIdLast5Years') {
    $learningResourceService->getByAuthorIdLast5Years($AMDBConnection,
$_REQUEST['author_id']);
    echo $learningResourceService->getAsJSON();
} elseif ($_REQUEST['action'] ==
'getManualsTextbooksByAuthorLast5Years') {
    $learningResourceService-
>getManualsTextbooksByAuthorLast5Years($AMDBConnection,
$_REQUEST['author_id']);
    echo $learningResourceService->getAsJSON();
}
```

LearningResourceAuthorController.php

```
<?php
require_once('./app/services/LearningResourceAuthorService.php');
require_once('./app/controllers/BaseController.php');

$learningResourceAuthorService = new LearningResourceAuthorService();

$controller = new BaseController($AMDBConnection,
$learningResourceAuthorService);
$controller->processRequest();
```

LearningResourceTypeController.php

```
<?php
require_once('./app/services/LearningResourceTypeService.php');
require_once('./app/controllers/BaseController.php');

$learningResourceTypeService = new LearningResourceTypeService();

$controller = new BaseController($AMDBConnection,
$learningResourceTypeService);
$controller->processRequest();
```

AppRouter.php

```
<?php
require_once('./app/middleware/auth.php');
header('Content-type: text/plain; charset=utf-8');
$request = strtok($_SERVER["REQUEST_URI"], '?');
if (strpos($request, "/auth")) {
    if ($request == $AMProjectBaseUrl . '/auth/auth-url') {
        echo generateAuthURL($AMGoogleAuthEndpoint, $AMGoogleClientID,
$AMServerURL, $AMProjectBaseUrl, $AMAuthCallbackURL);
    } else if ($request == $AMProjectBaseUrl . '/auth/profile') {
                    echo    json_encode(getProfileInfo($AMBearerToken),
JSON_UNESCAPED_UNICODE);
    } else if ($request == $AMProjectBaseUrl . '/auth/callback') {
        if (isset($_GET['code'])) {
                                                $result =
getAccessTokenandAuthorizeUser($AMGoogleTokenEndpoint, $AMGoogleClientID,
$AMGoogleSecret, $AMServerURL, $AMProjectBaseUrl, $AMAuthCallbackURL,
$_GET['code'], $AMBearerToken);
            header('Location: ' . $AMServerURL);
```

```
            echo json_encode($result, JSON_UNESCAPED_UNICODE);
        }
    } else if ($request == $AMProjectBaseUrl . '/auth/logout') {
        $result = logOut();
        echo json_encode($result, JSON_UNESCAPED_UNICODE);
    }
} else {
    if ($_SERVER['REQUEST_METHOD'] == "OPTIONS") {
        header('Access-Control-Allow-Origin: *');
           header("Access-Control-Allow-Headers: X-API-KEY, Origin, X-
Requested-With,
            Content-Type, Accept, Access-Control-Request-Method,Access-
Control-Request-
            Headers, Authorization");
        header("HTTP/1.1 200 OK");
        die();
    }
    if (strpos($request, "file-upload") && $_SERVER['REQUEST_METHOD'] ==
"POST") {
        if ($_POST['token'] != $AMBearerToken) {
            http_response_code(401);
            require_once ("./assets/401.php");
            echo $_POST['token'];
        } else {
            require_once("./app/controllers/UserUploadController.php");
        }
    } else {
        if ((getBearerToken() != $AMBearerToken) && (getBearerToken() !=
$AMAppToken)) {
        http_response_code(401);
        require_once("./assets/401.php");
        } else {
        switch ($request) {
            case $AMProjectBaseUrl . '/learning-resource-type':
                 require_once("./app/controllers/LearningResourceTypeCont
roller.php");
                break;
            case $AMProjectBaseUrl . '/course-resource-type':
                 require_once("./app/controllers/CourseResourceTypeContro
ller.php");
                break;
            case $AMProjectBaseUrl . '/learning-resources':
                 require_once("./app/controllers/LearningResourceControll
er.php");
                break;
            case $AMProjectBaseUrl . '/learning-resource-author':
                 require_once("./app/controllers/LearningResourceAuthorCo
ntroller.php");
                break;
            case $AMProjectBaseUrl . '/course-resource':
                 require_once("./app/controllers/CourseResourceController
.php");
                break;
            default:
                http_response_code(404);
                require_once("./assets/404.php");
```

```
                break;
            }
        }
    }
}
```

DBProvider.php

```
<?php
$AMDBConnection = new mysqli($AMServerName, $AMUserName, $AMPassword,
$AMDatabase);

// Check connection
if ($AMDBConnection->connect_error) {
  die("Connection failed: " . $AMDBConnection->connect_error);
}
?>
```

config.php

```
<?php
//Server URL
$AMServerURL = 'http://localhost';
//Base URL of the Project
$AMProjectBaseUrl = '/alma-mater-backend';
//AM Database Login and Password
$AMServerName = "localhost";
$AMUserName = "root";
$AMPassword = "";
$AMDatabase = "alma_mater_db";
//API Bearer Token
$AMBearerToken = '…';
$AMAppToken = '…';
//Google OAuth credentials
$AMGoogleClientID = '…';
$AMGoogleSecret = '…';
$AMGoogleAuthEndpoint = 'https://accounts.google.com/o/oauth2/auth';
$AMGoogleTokenEndpoint = 'https://oauth2.googleapis.com/token';
$AMAuthCallbackURL = "/auth/callback";
```

БІБЛІОГРАФІЧНА ДОВІДКА

Тема бакалаврської роботи: ***Розробка серверної частини компоненти “Методичне забезпечення дисциплін” інформаційної системи ІФНТУНГ***

Обсяг пояснювальної записки 70 аркушів:

6 таблиць;

25 рисунків;

1 додаток.

Дата завершення роботи: *12 червня 2025р*.

Підпис студента- _____*Щеснюк П.В.*