БАКАЛАВРСЬКА РОБОТА

БР.КІ-10.00.00.000 ПЗ

Група КІ-21-1

Копко Ростислав

2025

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

Копко Ростислав Михайлович

УДК 004.02

БАКАЛАВРСЬКА РОБОТА

Розробка серверної частини компоненти "Професійні досягнення працівників" 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 Копко Р. М.
(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)

м. 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 СТУДЕНТОВІ

Копку Ростиславу Михайловичу

(прізвище, ім’я, по батькові)

1. Тема роботи Розробка серверної частини компоненти “Професійні досягнення працівників” 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 Облік професійних досягнень науково-педагогічних працівників, як прикладна задача у складі інформаційної системи університету, розгляд наявних прикладів вирішення схожих задач, постановка задачі. 2 Обґрунтування вибору програмного забезпечення, розробка структури таблиць бази даних, розробка діаграм взаємодії з користувачем. 3 Розробка функцій для обробки даних інформаційної системи, створення шляхів для адресації по сторінкам інформаційної системи

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 та вибір літератури для написання бакалаврської роботи	24.02.2025 – 07.03.2025	виконав
2	Опрацювання предметної області та аналіз аналогів інформаційних систем	08.03.2025 – 16.03.2025	виконав
3	Розробка структури бази даних, створення діаграм	17.03.2025 – 05.04.2025	виконав
4	Розробка коду компоненти для серверної частини інформаційної системи та перевірка працездатності	06.04.2025 – 30.04.2025	виконав
5	Оформлення пояснювальної записки	01.05.2025 – 30.05.2025	виконав
6	Підготовка до здачі бакалаврської роботи	31.05.2025 – 07.06.2025	виконав

Студент _____ Копко Р. М.
(підпис) (прізвище та ініціали)

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

АНОТАЦІЯ

Основним завданням цієї роботи є створення серверної частини компоненти “Професійні досягнення працівників” інформаційної системи ІФНТУНГ, що надаватиме доступ до даних про кваліфікації, досвід, вчені звання, наукові ступені та рівень володіння англійською мовою працівників університету. Для вирішення цієї задачі використано платформу XAMPP і мову програмування PHP, що забезпечує простоту розробки частини інформаційної системи. Задачі включають створення структури таблиць бази даних компоненти “Професійні досягнення працівників” для частини інформаційної системи, маршрутизацію по відповідних сторінках системи та реалізацію методів для роботи із таблицями бази даних. Результатом роботи є функціональна частина інформаційної системи, яка надає доступ до професійних досягнень працівників. Розроблена частина інформаційної системи відповідає вимогам сучасних систем і може бути розширена новими функціями для покращення користувацького досвіду.

Ключові слова: XAMPP, PHP, ІНФОРМАЦІЙНА СИСТЕМА, БАЗА ДАНИХ.

ANNOTATION

The main task of this work is to develop the server-side component of the "Professional Achievements of Employees" module of the IFNTUOG information system, which will provide access to data on the qualifications, experience, academic titles, scientific degrees, and English language proficiency of the university staff. To achieve this goal, the XAMPP platform and PHP programming language were used, ensuring the simplicity of developing this part of the information system. The tasks include creating the database table structure for the "Employees' Professional Achievements" component, routing to the relevant system pages, and implementing methods for working with the database tables. The result of the work is a functional part of the information system that provides access to employees' professional achievements. The developed part of the information system meets the requirements of modern systems and can be expanded with new functions to enhance the user experience.

Keywords: XAMPP, PHP, INFORMATION SYSTEM, DATABASE.

ЗМІСТ

ВСТУП

Актуальність обраної теми. Відстеження та управління інформацією про професійні досягнення є важливим інструментом звітності щодо професійної компетенції науково-педагогічних працівників. Впровадження автоматизованих систем для зберігання та аналізу даних про професійні досягнення дозволяє ефективніше використовувати ці дані для отримання повної інформації про особу її досвід та професійне зростання. Тому створення модуля обліку професійних досягнень працівників є важливою прикладною задачею у складі проєктування інформаційної системи університету.

Об’єкт та предмет дослідження. Об’єктом дослідження є процес обліку професійних досягнень працівників університету. Предметом дослідження є методи і засоби обліку професійних досягнень працівників закладів освіти.

Мета та завдання роботи. Метою даної роботи є створення бекенд-системи, яка дозволяє зберігати та управляти даними про професійні досягнення працівників, використовуючи платформу XAMPP, локальний сервер MySQL, який надає ця платформа та мову програмування PHP. Завдання включають створення таблиць бази даних, для зберігання інформації про професійні досягнення працівників університету, написання коду, для взаємодії із вмістом таблиць, перевірку працездатності створеного програмного коду, та інші аспекти розробки бекенд-систем.

Методи дослідження. Методи дослідження включають аналіз існуючих інформаційних систем такого типу. Це допомагає зрозуміти сучасні тенденції та вимоги до таких систем. Вивчення документації щодо роботи з SQL-запитами та інструментами для роботи з базами даних є важливим етапом, оскільки це дозволяє ознайомитися з можливостями технологій, які будуть використані для розробки серверної частини інформаційної системи ІФНТУНГ. Також важливими етапами є проектування схеми бази даних, створення HTTP-запитів для взаємодії із таблицями бази даних.

Практичне значення отриманих результатів. Практичне значення отриманих результатів полягає у розробці серверної частини модуля «професійні досягнення працівників» у складі єдиної інформаційної системи університету, що дозволяє спростити облік інформації про професійну діяльність працівників. Результати роботи підтверджені відповідним актом про впровадження.

1 ОГЛЯД АНАЛОГІВ ІНФОРМАЦІЙНИХ СИСТЕМ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Облік професійних досягнень науково-педагогічних працівників, як прикладна задача у складі інформаційної системи університету

В 2024 році в Івано-Франківському національному технічному університеті нафти і газу розпочалося впровадження єдиної інформаційної системи університету. Першим кроком до впровадження стала реалізація модулів заповнення індивідуальних планів роботи викладачів. Паралельно, здійснювалася розробка клієнтської та серверної частини інших компонент.

Інформаційна система університету реалізовується на стеку технологій PHP+MySQL (серверна частина), HTML+CSS+Javascript (клієнтська частина). Взаємодія клієнтської та серверної частин здійснюється з допомогою API з авторизацією через Bearer-токен. Архітектура серверної частини містить файли конфігурацій, сервіси (для взаємодії з базами даних), моделі (для формування структури даних), контролери (для опрацювання запитів), а також проміжні модулі для авторизації та маршрутизації.

Станом на зараз, згідно Стратегії цифровізації ІФНТУНГ на 2024-2028 роки, стоїть завдання мінімізувати дублювання інформації в звітних документах, агрегувавши її в одну інформаційну систему з відповідною звітністю. Тому ведеться робота над модулями інформаційної системи з цією метою.

Зокрема, важливим компонентом звітності є інформація про відповідність працівників ліцензійним вимогам щодо провадження освітньої діяльності, відповідність навчальній дисципліні, яку вони викладають. Серед вимог, які ставляться до відповідності освітніх компонент - наукові публікації, методичні розробки, відповідний досвід роботи, науковий ступінь, тощо.

Розробка і впровадження такої системи повинна ґрунтуватися на низці нормативних і правових документів, що регламентують організацію освітньої та наукової діяльності в Україні. До ключових належать:

– Закон України “Про освіту” - визначає основні принципи освітнього процесу, вимоги до викладацької діяльності [1];

– Закон України “Про вищу освіту” - містить норми щодо кваліфікацій НПП, атестації, публікаційної активності [2];

– Закон України “Про захист персональних даних” - регламентує збереження та обробку персональної інформації працівників [3];

– Закон України «Про наукову і науково-технічну діяльність» - містить положення про результативність наукової діяльності [4].

Також важливим є урахування внутрішніх нормативних актів навчального закладу - регламентів, положень, шаблонів звітності тощо.

Архітектура інформаційної системи передбачає облік наукових публікацій та методичних розробок в окремих модулях. Крім того, окремою компонентою є облік професійних досягнень працівників.

Компонента "Професійні досягнення працівників" є частиною серверної інформаційної системи університету, яка відслідковує та надає доступ до даних про професійні досягнення працівників університету. Ця компонента включає в себе інформацію про кваліфікацію, досвід, вчені звання, наукові ступені та рівень володіння англійською мовою. Завдання цієї компоненти - надавати користувачу можливість переглядати важливу інформацію про професійні досягнення працівників. Адміністратор матиме додаткову можливість додавати, редагувати та видаляти інформацію про ці досягнення.

Головна мета цієї частини інформаційної системи зберігати дані про підвищення кваліфікації, досвід працевлаштування, вчені звання, наукові ступені та рівень володіння англійською мовою працівників університету, надавати доступ до даних адміністрації університету, викладачам та іншим користувачам системи, забезпечити зручну навігацію та швидкий пошук необхідної інформації. Також, серверна частина даної компоненти має забезпечувати інформаційну підтримку із формування звітів, зокрема, передбачених ліцензійними умовами щодо провадження освітньої діяльності [5].

Таким чином, розроблена компонента "Професійні досягнення

працівників" серверної частини інформаційної системи університету сприятиме ефективному керуванню даними про працівників університету та забезпечить зручний доступ до цієї інформації для всіх користувачів цієї системи.

Крім відображення інформації у внутрішній системі університету та звітах, передбачається інтеграція розроблюваної компоненти з новою версією сайту університету, яка на даний момент перебуває в розробці, для відображення даних у профілі працівника, де буде міститись загальна інформація, навчальна робота, наукова діяльність, кваліфікаційний рівень, науковий ступінь, вчене звання та посада.

1.2 Розгляд наявних прикладів вирішення схожих задач

Розглянемо наявний аналог інформаційної системи в мережі Інтернет.

Єдина державна електронна база з питань освіти - автоматизована система збирання, сертифікації, оброблення, зберігання та захисту даних, у тому числі персональних, щодо надавачів та отримувачів освітніх послуг в Україні.

Дані з цієї бази використовуються під час виготовлення:

– документів про освіту державного зразка;

– документів про вчені звання та наукові ступені;

– ліцензій на надання освітніх послуг та сертифікатів про акредитацію;

– учнівських (студентських) квитків.

Держателем (власником) Єдиної бази є МОН, адміністратором (розпорядником) Єдиної бази є державне підприємство «Інфоресурс», що належить до сфери управління МОН (Міністерство освіти і науки України) [6].

Бачимо, що ЄДЕБО має велику базу даних із різноманітною інформацією, яка стосується освітніх процесів. Оскільки ми розроблятимемо компоненту "Професійні досягнення працівників" серверної частини інформаційної системи університету, то нас цікавитиме як у державній інформаційній системі зберігаються відомості про вчені звання та наукові ступені працівників.

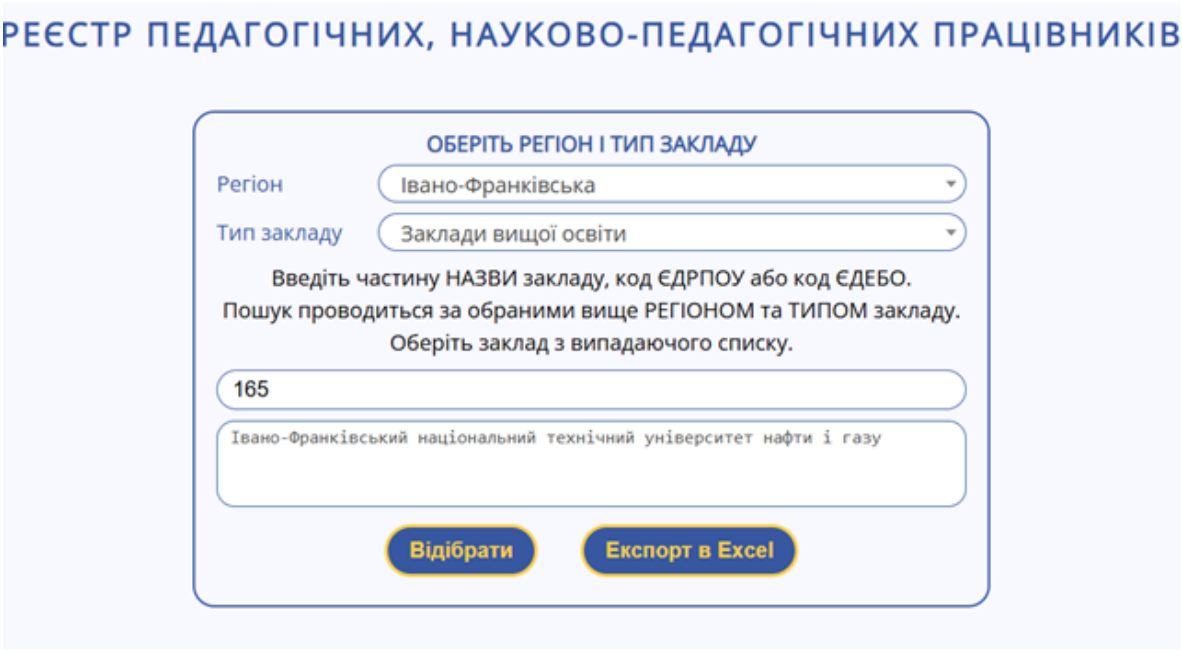

Рисунок 1.1 – Вибір навчального закладу в ЄДЕБО

Інформаційна система ЄДЕБО зображена на рисунку 1.1, пропонує користувачу на вибірку конкретні навчальні заклади. Щоб переглянути інформацію про працівників цих закладів. Це передбачає, що у базі даних ЄДЕБО міститься інформація про всі університети України.

Пошук

Ідентифікатор закладу освіти в ЄДЕБО	Назва закладу освіти, в якому працевлаштований працівник	Документ про науковий ступінь	Документ про вчене звання	Кваліфікаційна категорія	Педагогічне звання	Відомості про сертифікацію	Посада	Трудовий статус посади	Стаж наукової, педагогічної, науково-педагогічної роботи (повних років)	Дата початку стажу
165	Івано-Франківський національний технічний університет нафти і газу						викладач	Сумісництво	15	01.09.2010
165	Івано-Франківський національний технічний університет нафти і газу	Диплом кандидата наук, кандидат економічних наук	Атестат доцента, Доцент кафедри прикладної економіки				доцент	Основне місце роботи	23	01.09.2000
165	Івано-Франківський національний технічний університет нафти і газу	Диплом доктора наук, доктор психологічних наук; Диплом кандидата наук, кандидат психологічних наук	Атестат доцента, доцент кафедри вищої математики; Атестат професора, професор кафедри вищої математики				професор	Основне місце роботи	35	10.10.1989
165	Івано-Франківський національний технічний університет нафти і газу	Диплом кандидата наук, Кандидат економічних наук	Атестат доцента, доцент кафедри управління виробництвом				доцент	Основне місце роботи	27	01.08.1997
165	Івано-Франківський національний технічний університет нафти і газу	Диплом доктора наук, Доктор геологічних наук; Диплом кандидата наук, Кандидат геолого-мінералогічних наук	Атестат доцента, Доцент кафедри геофізичних досліджень свердловин; Атестат професора, Професор кафедри геофізичних досліджень свердловин; Атестат старшого наукового співробітника (старшого дослідника), Старший науковий співробітник з спеціальності геофізичні методи пошуків та розвідки родовищ корисних копалин				професор	Основне місце роботи	32	01.09.1991

Рисунок 1.2 – Вибір навчального закладу в ЄДЕБО

Розглянемо таблицю із інформацією яка вивелася про працівників ІФНТУНГ, на рисунку 1.2.

В цій таблиці знаходиться вся потрібна інформація із бази даних про професійні досягнення працівників та навіть стаж роботи. Тільки у такому поданні інформації є певні мінуси. Ми не бачимо конкретного працівника і його відповідних досягнень, а також багато пустих значень, що не несуть корисної інформації.

Наступним аналогом інформаційної системи є сайт Prometheus.

Prometheus (укр. Прометеус) — український громадський проєкт масових відкритих онлайн-курсів, запущений 2014 року. Головною метою проєкту є безкоштовне надання онлайн-доступу до курсів університетського рівня всім бажаючим, а також надання можливості публікувати та розповсюджувати такі курси провідним викладачам, університетам та компаніям [7].

Студент чи учень може вибирати курс, який його цікавить та ознайомитися з інформацією про переваги цього курсу. Одним з елементів опису курсів є інформація про стаж роботи та навички викладачів.

Рисунок 1.3 – Інформація про викладача курсу на сайті Prometheus

На рисунку 1.3 показано, як користувач цього сайту може ознайомитися з інформацією про викладача курсу. Переглянути його досвід роботи, якими навичками володіє та, які технології використовує для роботи по цьому курсу.

Тут інформація структурно і гарно описана, проте варто більше описати саме викладацьку діяльність цієї особи, оскільки ми розглядаємо сайт із наданням освітніх послуг.

Останнім аналогом інформаційної системи, який розглянемо буде сайт EdEra.

EdEra — українська студія онлайн-освіти, яка понад 8 років створює онлайн-курси, навчальні платформи, інтерактивні ігри та підручники. У нас є власний R&D-юніт, продакшн повного циклу та натхненна команда у Києві. А в Берліні — технологічний стартап та унікальна LMS [8].

Лекторка — Анна Ляшенко, Senior Instructional Designer в Amazon та L&D-спеціалістка EdEra

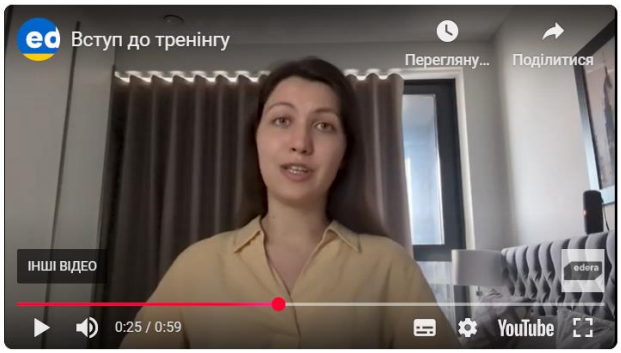

Рисунок 1.4 – Інформація про викладача курсу на сайті EdEra

На рисунку 1.4 показано, всю інформацію про викладача цього курсу. Коротке ознайомче відео та посаду викладача. Попередні аналоги інформаційних систем ЄДЕБО та Prometheus надавали більше інформації про стаж та досягнення викладачів. Можна сказати, що EdEra сконцентровує увагу користувачів на структурі та перевагах курсу, проте не ознайомлює із особою, яка буде проводити цей курс і її викладацьким досвідом.

Наша компонента "Професійні досягнення працівників", яку ми розроблятимемо матиме пряме відношення із працівниками університету. І розглядати конкретного викладача із його професійними досягненнями. Також ми передбачимо, що таблиці бази даних не дозволятимуть створення записів із пустими значеннями. Всі поля таблиць, які нестимуть важливу інформацію будуть обов’язковими для заповнення. В результаті чого ми матимемо повну та

детальну інформацію про кваліфікації, досвід, вчені звання, наукові ступені, рівень володіння англійською мовою працівників університету та інше.

Такі вимоги до розроблення компоненти "Професійні досягнення працівників" наддадуть переваги нашій інформаційній системі порівняно із аналогами, які ми розглянули.

Власником цієї інформаційної системи буде університет, адміністратори матимуть можливість додавати, редагувати чи видаляти інформацію із бази даних інформаційної системи. Користувачами будуть працівники університету, в планах вдосконалити інформаційну систему та додати також студентів до користувачів системи. Проте зараз сайт орієнтований на працівників університету.

1.3 Постановка задачі

Потрібно розробити серверну компоненту "Професійні досягнення працівників" для інформаційної системи університету. Система повинна забезпечувати зручне зберігання, редагування, видалення та перегляд професійних досягнень працівників. Завдання включають розробку структури таблиць бази даних, написання програми для взаємодії з таблицями, що виконуватиме методи CRUD, та створення відповідних маршрутів для переходу на сторінки інформаційної системи.

Вимоги до інформаційної системи:

Функціональні можливості:

– реалізація функцій CRUD (створення, читання, оновлення, видалення) для таблиць бази даних;

– запити до таблиць бази даних, на основі яких створюються функцій CRUD, мають бути оптимізованими для швидкої роботи інформаційної системи;

– за умови, коли інформаційній системі потрібно зробити унікальну вибірку інформації, то до функцій CRUD, які реалізують основну роботу із базою даних, створити додатковий метод для отримання потрібної інформації для

користувача.

База даних:

– структура таблиць бази даних повинна враховувати всі необхідні поля для зберігання інформації про професійні досягнення працівників;

– заповнення таблиць тестовими даними для перевірки функціональності системи.

Структура серверної частини інформаційної системи:

– шляхи (Routes): Відповідатимуть за переведення користувача на відповідні сторінки із потрібною інформацією. Помагають правильно побудувати посилання на сторінки сайту та задіють відповідні контролери, для маніпуляції із базою даних;

– моделі (Models): Відтворюватимуть структуру таблиці бази даних для програми. Тобто передають поля бази даних програмному коду, даючи доступ до подальших маніпуляції із даними, які містять поля цих таблиць;

– контролери (Controllers): Отримуватимуть події від зовнішнього світу (зазвичай через представлення), взаємодіятимуть з моделлю і відображатимуть відповідне представлення (інформацію) користувачеві;

– сервіси (Services): Файли в яких окремо винесені всі запити до таблиць бази даних, тобто функції CRUD. Далі вже готові запити викликаються контролером, відповідно до дій користувача.

Перевірка працездатності розробленої частини інформаційної системи:

– через локальний сервер перевірити вивід інформації у браузері, на сторінках новоствореної компоненти "Професійні досягнення працівників" інформаційної системи університету;

– на платформі Postman перевірити всі HTTP-запити, які виконуються завдяки функціям CRUD, для кожної таблиці бази даних, новоствореної компоненти.

2 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА ОСНОВА СТРУКТУРИ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Вимоги до програмного забезпечення

Для початку потрібно обрати мову програмування, на якій будемо писати програмний код для створення компоненти “Професійні досягнення працівників” серверної частини інформаційної системи університету. Оскільки проект вже є реалізований на PHP, то ми маємо притримуватися цієї мови програмування для створення наступних елементів проекту.

Фактори, які впливають на вибір мови програмування PHP:

– простота синтаксису, що робить цю мову програмування легкою для розуміння, особливо для розробників із невеликим досвідом;

– підтримується більшістю веб-серверів таких, як Apache, Nginx, IIS та операційних систем Windows, Linux, macOS. Також мова програмування добре інтегрується з багатьма базами даних, такими як MySQL, PostgreSQL чи MariaDB;

– оптимізована для роботи з веб-сервером і обробки HTTP-запитів, що робить його ефективним інструментом для створення динамічних веб-додатків;

– активний розвиток мови програмування, нові версії пропонують покращену продуктивність, безпеку та нові функції [9].

Наступним кроком потрібно вибрати редактор коду, в якому створюватиметься компонента інформаційної системи університету. Гарним варіантом буде Visual Studio Code.

Переваги редактора коду Visual Studio Code:

– vs code є абсолютно безкоштовним редактором коду, що робить його доступним для широкого кола людей;

– кросплатформність, працює на Windows, macOS і Linux, що дозволяє використовувати його незалежно від операційної системи;

– простота інтерфейсу, дозволяє швидко користуватися цим програмним забезпеченням;

– підтримка плагінів для мови програмування, її стилів оформлення, інтеграції з інструментами та багато іншого;

– git-інтеграція надає можливість роботи з системою контролю версій Git безпосередньо у середовищі редактора [10].

Останнім, що потрібно для успішного запуску проекту це середовище, яке надасть нам локальний сервер (MySQL) та можливість адмініструвати базу даних (phpMyAdmin), яка є частною розроблюваного проекту. Вибрано для виконання перерахованих задач збірку багатоплатформового веб-сервера XAMPP.

Переваги XAMPP:

– кросплатформність, працює на Windows, macOS і Linux, що робить його універсальним для різних операційних систем;

– простота встановлення та налаштування програмного забезпечення;

– безкоштовний та доступний для встановлення широкому колу людей;

– вбудовані компоненти, надає доступ до таких інструментів, як Apache, MySQL, PHP, Perl, phpMyAdmin та інші, необхідні для веб-розробки;

– дозволяє створювати та перевіряти працездатність веб-сайтів локально перед їх розгортанням в інтернеті;

– зручна панель керування для запуску та зупинки сервісів [11].

Переваги та доступність цього програмного забезпечення надає умови для створення компоненти “Професійні досягнення працівників” серверної частини інформаційної системи університету. Розробка цієї компоненти вимагатиме тільки часу на її створення та правильне налаштування програмного забезпечення. Завдяки зручному інтерфейсу середовища розробки та легкому синтаксису PHP, витрати часу на налаштування проекту та його написання будуть не великими.

2.2 Розробка структури бази даних

Для створення бази даних спершу необхідно визначити її структуру та вміст, після цього, можемо створити такі таблиці:

Таблиця 2.1 містить інформацію про підвищення кваліфікації працівників університету.

Таблиця 2.1 – ExcellenceCertificate

Ключове поле	Назва стовпця	Тип даних	Довжина	Дозволяє NULL	Вміст
+	id	int	11	-	ID запису
-	userid	int	11	-	ID користувача
-	name	text	65 535	-	Назва сертифікату
-	issuedate	datetime	-	-	Дата видачі
-	hours	float	-	-	Кількість годин
-	issuername	text	65 535	-	Назва установи, яка видала сертифікат
-	certificaturl	text	65 535	-	Посилання на сертифікат

Таблиця 2.2 містить інформацію про професійний досвід працівників університету.

Таблиця 2.2 – ProfessionalExperience

Ключове поле	Назва стовпця	Тип даних	Довжина	Дозволяє NULL	Вміст
+	id	int	11	-	ID запису
-	userid	int	11	-	ID користувача
-	title	text	65 535	-	Назва посади
-	workplace	text	65 535	-	Назва місця роботи
-	startdate	datetime	-	-	Дата початку роботи
-	enddate	datetime	-	+	Дата завершення роботи
-	proofurl	text	65 535	-	Посилання на підтвердження

Таблиця 2.3 містить інформацію про освіту, вчені звання та наукові ступені працівників університету.

Таблиця 2.3 – Education

Ключове поле	Назва стовпця	Тип даних	Довжина	Дозволяє NULL	Вміст
+	id	int	11	-	ID запису
-	userid	int	11	-	ID користувача
-	title	text	65 535	-	Назва навчального закладу
-	speciality	text	65 535	-	Спеціальність
-	diplomanumber	text	65 535	-	Номер диплома
-	issuedate	datetime	-	-	Дата видачі диплома
-	proofurl	text	65 535	-	Посилання на підтвердження

Таблиця 2.4 містить інформацію про сертифікати володіння англійською мовою працівників університету.

Таблиця 2.4 – LanguageLevel

Ключове поле	Назва стовпця	Тип даних	Довжина	Дозволяє NULL	Вміст
+	id	int	11	-	ID запису
-	userid	int	11	-	ID користувача
-	language	text	65 535	-	Мова
-	certificateissuer	text	65 535	-	Назва установи, яка видала сертифікат
-	level	text	65 535	-	Рівень володіння
-	issuedate	datetime	-	-	Дата видачі сертифікату
-	proofurl	text	65 535	-	Посилання на підтвердження

Дальше визначимо зв’язки між таблицями за допомогою ключових полів:

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 [12].

Отже, первинними ключами в базі даних є поля з назвою id, які знаходяться у відповідних таблицях 2.1, 2.2, 2.3 та 2.4. На цьому етапі розробки інформаційної системи первинні ключі не використовуються для зв'язування, а існують для ідентифікації записів. На рисунку 2.1, який подано нижче, можна помітити, що кожна таблиця має ще зовнішній ключі із назвою userid. Цим зовнішнім ключам відповідає первинний ключ id із таблиці User. Ця таблиця належить до іншої компоненти інформаційної системи, яка вже розроблена і представляє інформацію про користувачів (працівників університету). Завдяки побудові таких зв'язків ми об'єднуємо два фрагменти бази даних та ведемо детальний облік інформації про професійні досягнення кожного працівника університету.

Тепер розглянемо структуру фрагменту бази даних:

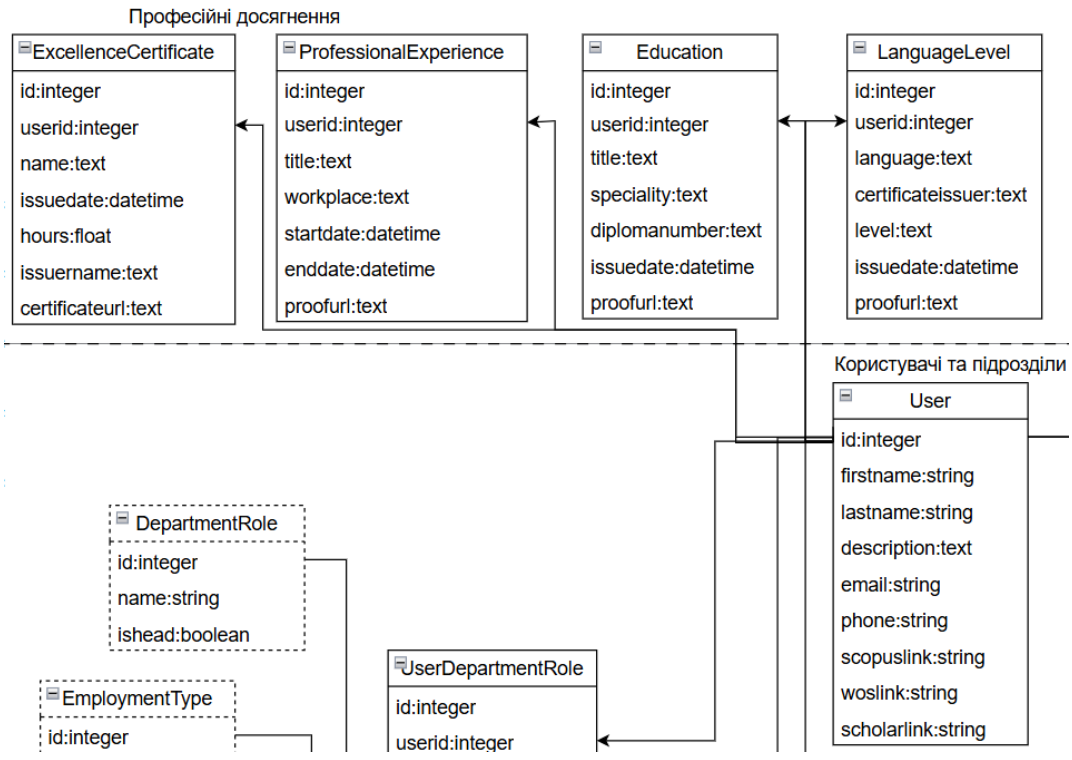

Рисунок 2.1 – Діаграма структури фрагменту бази даних

Зв’язки у цьому фрагменті бази даних будуть типу один до багатьох. Відносини такого роду виникають, коли запис однієї таблиці пов'язаний з декількома сутностями іншої [13].

Структура бази даних задовільняє вимоги інформаційної системи. Поля таблиць і їх тип даних відповідає інформації, яка міститиметься в цих таблицях.

2.3 Створення бази даних для компоненти інформаційної системи

Коли визначено структуру фрагменту бази даних, можна переходити до його створення. Завдяки можливостям платформи XAMPP ми можемо користуватися веб-додатком для адміністрування баз даних phpMyAdmin. Там ми будемо створювати фрагмент бази даних.

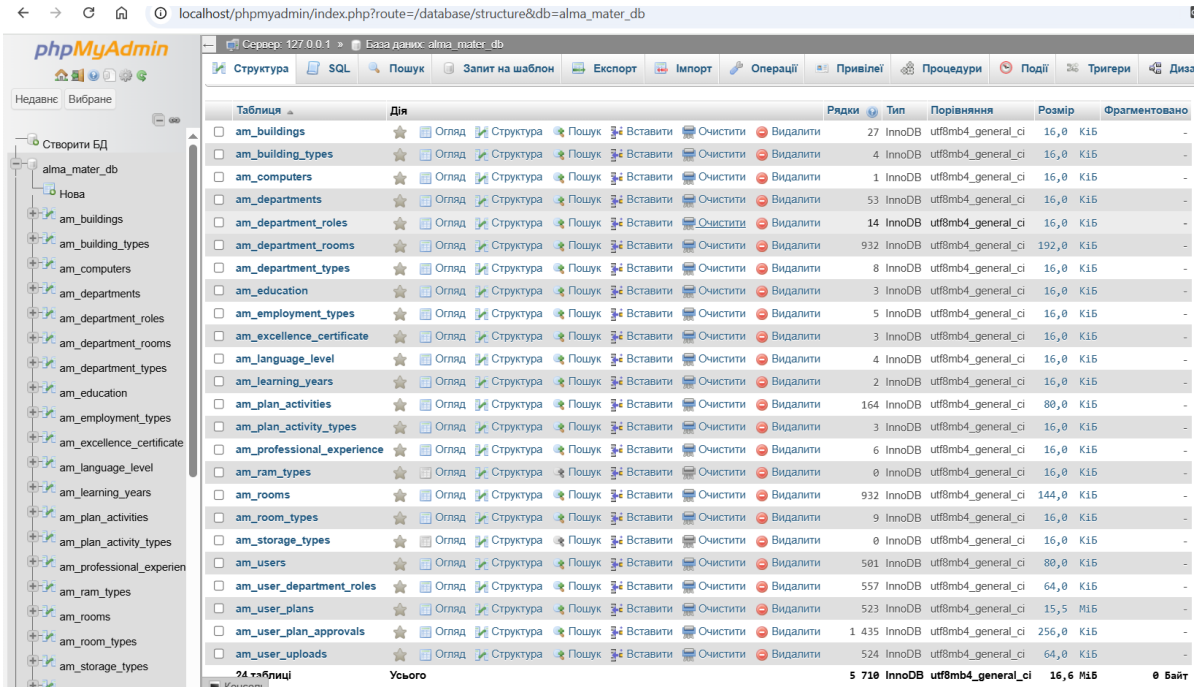

Рисунок 2.2 – Робоче середовище phpMyAdmin

На рисунку 2.2 можна побачити, що phpMyAdmin надає функціонал СУБД, щоб можна було виконувати різні маніпуляції із базами даних. Написання SQL-запитів, імпорт чи експорт бази даних та інші стандартні операції. Бачимо, що база даних інформаційної системи університету має назву alma_mater_db та

містить уже велику кількість таблиць, які відносяться до різних компонент інформаційної системи.

Скористалися дампом бази даних, щоби перенести базу на локальний сервер та працювати із нею [14].

Розроблювана компонента “Професійні досягнення працівників” вимагає створення таких таблиць, які вже присутні в базі даних: am_excellence_certificate, am_professional_experience, am_education та am_language_level. Назва таблиць формується за визначеним правилом. Початок має містити префікс “am”, що означає, що таблиці належать базі даних alma_mater_db. Дальше назва має збігатися із тією, яка визначена в діаграмі із структурую фрагмента бази даних. Всі літери в назві з малої літери, щоб розділити слова використовуємо нижнє підкреслення. Дотримання цих вимог забезпечить створення зрозумілих назв таблиць та полегшить написання запитів розробнику.

Саме створення таблиць бази даних і її полів виконувалося на вкладці меню SQL, в якій надається аркуш для виконання різних SQL-запитів, в нашому випадку CREATE TABLE.

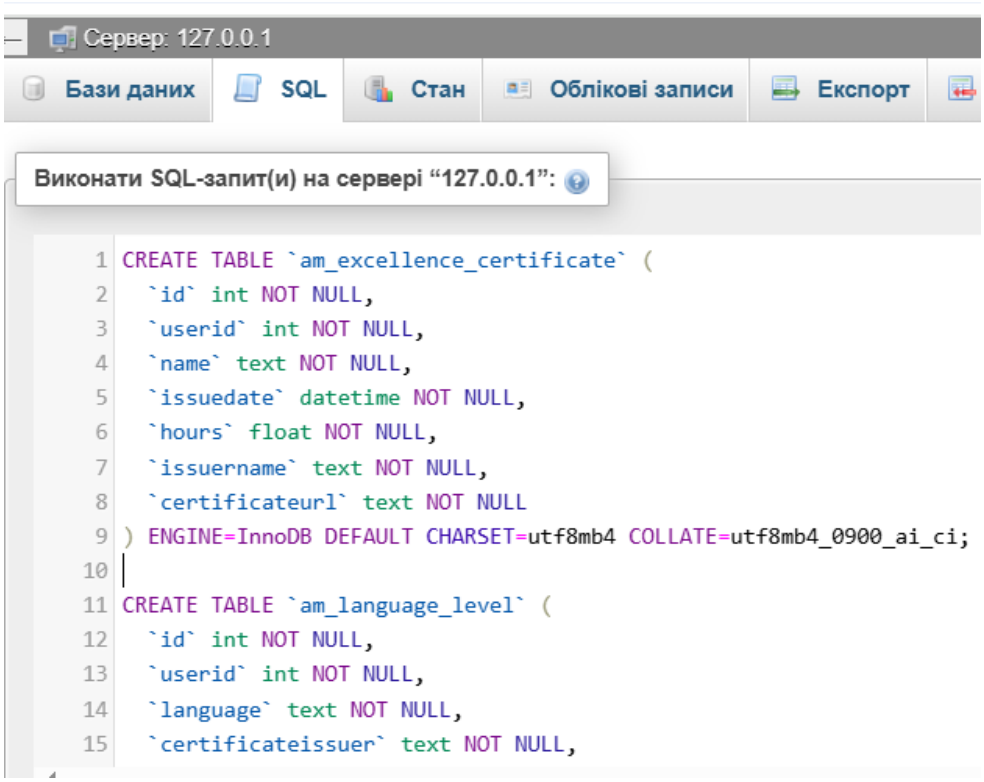

Рисунок 2.3 – Виконання запитів CREATE TABLE

На рисунку 2.3 бачимо вже готові запити на створення таблиць бази даних, важливим пунктом є вибір бази даних для якої будуть створюватися ці таблиці. Якщо перед створенням таблиць не вказати базу даних, то виникатиме помилка виконання цих запитів, оскільки phpMyAdmin не розуміє куди створювати ці таблиці.

Код CREATE TABLE:

Таблиця am_excellence_certificate - зберігатиме інформацію про підвищення кваліфікації працівників університету.

```
CREATE TABLE `am_excellence_certificate` (
  `id` int NOT NULL,
  `userid` int NOT NULL,
  `name` text NOT NULL,
  `issuedate` datetime NOT NULL,
  `hours` float NOT NULL,
  `issuername` text NOT NULL,
  `certificateurl` text NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

Таблиця am_professional_experience - зберігатиме інформацію про професійний досвід працівників університету.

```
CREATE TABLE `am_professional_experience` (
  `id` int NOT NULL,
  `userid` int NOT NULL,
  `title` text NOT NULL,
  `workplace` text NOT NULL,
  `startdate` datetime NOT NULL,
  `enddate` datetime NOT NULL,
  `proofurl` text NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

Таблиця am_education – зберігатиме інформацію про освіту, вчені звання та наукові ступені працівників університету.

```
CREATE TABLE `am_education` (
  `id` int NOT NULL,
  `userid` int NOT NULL,
  `title` text NOT NULL,
  `speciality` text NOT NULL,
  `diplomanumber` text NOT NULL,
  `issuedate` datetime NOT NULL,
  `proofurl` text NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

Таблиця am_language_level – зберігатиме інформацію про сертифікати володіння англійською мовою працівників університету.

```
CREATE TABLE `am_language_level` (
  `id` int NOT NULL,
  `userid` int NOT NULL,
  `language` text NOT NULL,
  `certificateissuer` text NOT NULL,
  `level` text NOT NULL,
  `issuedate` datetime NOT NULL,
  `proofurl` text NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

Зв’яжемо створені таблиці із таблицею користувачів на основі первинних і зовнішніх ключів.

```
ALTER TABLE `am_excellence_certificate`
ADD CONSTRAINT `fk_excellence_certificate_user`
FOREIGN KEY (`userid`) REFERENCES `am_users`(`id`);
ALTER TABLE `am_professional_experience`
ADD CONSTRAINT `fk_professional_experience_user`
FOREIGN KEY (`userid`) REFERENCES `am_users`(`id`);
ALTER TABLE `am_education`
ADD CONSTRAINT `fk_education_user`
FOREIGN KEY (`userid`) REFERENCES `am_users`(`id`);
ALTER TABLE `am_language_level`
```

```
    ADD CONSTRAINT `fk_language_level_user`
    FOREIGN KEY (`userid`) REFERENCES `am_users`(`id`);
```

Коли створено таблиці та встановлено зв’язки потрібно занести записи, щоб вони не були пустими. Оскільки невідома точно інформація про професійні досягнення працівників, а головним завданням є перевірити роботу методів CRUD, то додамо тестові дані. Додавання записів виконуємо через запити INSERT INTO.

Заносимо записи у таблицю am_excellence_certificate:

```
    INSERT INTO `am_excellence_certificate` (`id`, `userid`, `name`, `issuedate`, `hours`, `issuername`, `certificateurl`) VALUES
    (1, 101, 'Certificate 1', '2023-01-01', 5.5, 'Issuer A', 'http://example.com/certificate1'),
    (2, 102, 'Certificate 2', '2023-01-02', 6.0, 'Issuer B', 'http://example.com/certificate2'),
    (3, 103, 'Certificate 3', '2023-01-03', 4.5, 'Issuer C', 'http://example.com/certificate3');
```

Заносимо записи у таблицю am_professional_experience:

```
    INSERT INTO `am_professional_experience` (`id`, `userid`, `title`, `workplace`, `startdate`, `enddate`, `proofurl`) VALUES
    (1, 101, 'Job Title 1', 'Workplace A', '2021-01-01', '2022-01-01', 'http://example.com/proof1'),
    (2, 102, 'Job Title 2', 'Workplace B', '2020-01-01', '2021-01-01', 'http://example.com/proof2'),
    (3, 103, 'Job Title 3', 'Workplace C', '2019-01-01', '2020-01-01', 'http://example.com/proof3');
```

Заносимо записи у таблицю am_education:

```
    INSERT INTO `am_education` (`id`, `userid`, `title`, `speciality`, `diplomanumber`, `issuedate`, `proofurl`) VALUES
```

```
    (1, 101, 'Degree 1', 'Speciality A', 'Diploma 1', '2022-01-01',
'http://example.com/proof1'),
    (2, 102, 'Degree 2', 'Speciality B', 'Diploma 2', '2021-01-01',
'http://example.com/proof2'),
    (3, 103, 'Degree 3', 'Speciality C', 'Diploma 3', '2020-01-01',
'http://example.com/proof3');
```

Заносимо записи у таблицю am_language_level:

```
    INSERT INTO `am_language_level` (`id`, `userid`, `language`,
`certificateissuer`, `level`, `issuedate`, `proofurl`) VALUES
    (1, 101, 'English', 'Issuer A', 'C1', '2023-01-01',
'http://example.com/proof1'),
    (2, 102, 'French', 'Issuer B', 'B2', '2022-01-01',
'http://example.com/proof2'),
    (3, 103, 'German', 'Issuer C', 'B1', '2021-01-01',
'http://example.com/proof3');
```

На цьому етапі розробки проекту перевірено повторно вимоги технічного завдання і вирішено впровадити зміни до бази даних. Можна змінити структуру таблиці за допомогою команди ALTER TABLE [15].

У таблиці am_professional_experience поле enddate може бути NULL, оскільки це означатиме що працівник, ще є на цій посаді і не закінчив свій професійний досвід.

```
ALTER TABLE `am_professional_experience`
MODIFY COLUMN `enddate` datetime NULL;
```

У таблиці am_excellence_certificate поле hours, яке відповідає за час витрачений на підвищення кваліфікації працівника, не може бути NULL, якщо значення цьому полю не задати воно автоматично буде “0”.

```
ALTER TABLE `am_excellence_certificate`
MODIFY COLUMN `hours` float NOT NULL DEFAULT 0;
```

Зробимо ідентифікатори створених таблиць первинними ключами та надамо їм властивість AUTO_INCREMENT, це потім допоможе реалізовувати коректне додавання нових записів у таблиці, бо ідентифікатор автоматично збільшуватиметься.

```
ALTER TABLE `am_excellence_certificate`
  MODIFY `id` int NOT NULL AUTO_INCREMENT,
  ADD PRIMARY KEY (`id`);
ALTER TABLE `am_professional_experience`
  MODIFY `id` int NOT NULL AUTO_INCREMENT,
  ADD PRIMARY KEY (`id`);
ALTER TABLE `am_education`
  MODIFY `id` int NOT NULL AUTO_INCREMENT,
  ADD PRIMARY KEY (`id`);
ALTER TABLE `am_language_level`
  MODIFY `id` int NOT NULL AUTO_INCREMENT,
  ADD PRIMARY KEY (`id`);
```

На глобальному сервері відтворюватимуться аналогічні дії створення таблиць бази даних, встановлення зв’язків між таблицями на основі первинних і зовнішніх ключів та заповнення їх реальними записами. Для перевірки функціоналу розроблюваного програмного забезпечення нам достатньо локального сервера.

2.4 Розробка діаграм взаємодії з користувачем

Розглянемо процес взаємодії користувача з розроблюваною компонентою "Професійні досягнення працівників" частини серверної інформаційної системи університету. Для цього побудуємо діаграму, що має на меті описати кожну можливу дію користувача із компонентою та зв'язки між цими діями, щоб забезпечити повне розуміння роботи розроблюваної частини інформаційної системи. Вона має надавати всі можливості для повноцінної роботи із

інформацією про професійні досягнення працівників, такі як: авторизація, додавання, редагування, видалення та перегляд інформації.

Авторизація є першою умовою для доступу користувача до інформаційної системи. Після успішного проходження авторизації користувач матиме можливість виконувати маніпуляції із інформацією, яка зберігається в базі даних. Користувачу цієї інформаційної системи надаватиметься доступ до додавання, редагування, видалення та перегляду даних про професійні досягнення працівників університету.

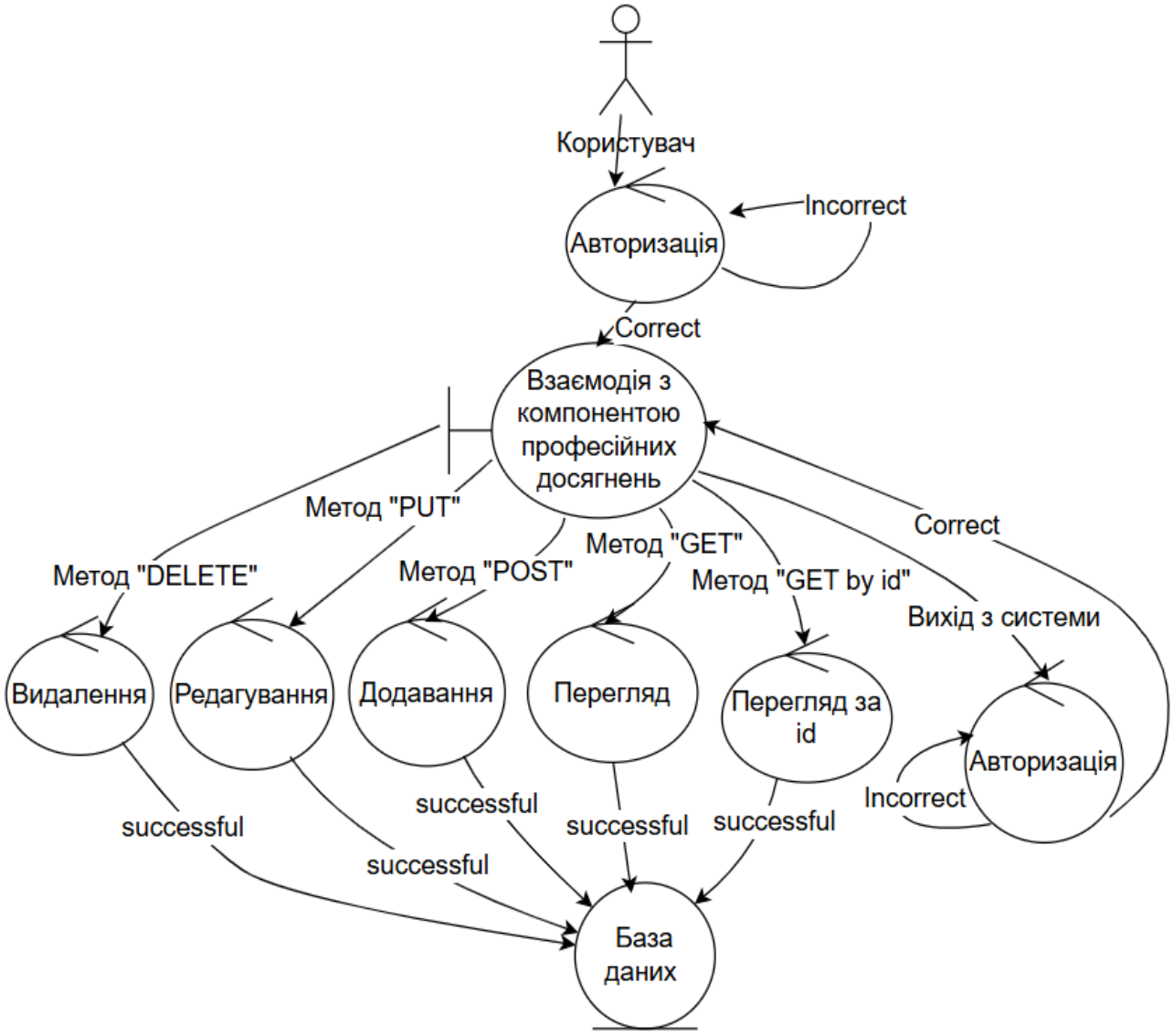

Рисунок 2.4 – Діаграма функціонування серверної частини інформаційної системи

На рисунку 2.4 можна побачити що користувач не матиме доступу до інформаційної системи поки не авторизується. Дальше з огляду на роботу бекенд частини цієї системи користувач, задіюючи HTTP-методи виконує запити до бази даних і маніпулює інформацією, яка в ній міститься.

Також додамо діаграму взаємодії із сервером, щоб показати як відбувається взаємодія сервера із базою даних:

На діаграмі відбуваються такі дії: користувач заходить у веб-додаток, щоб переглянути інформацію, в нашому випадку про професійні досягнення викладачів, після цього MySQL сервер надсилає запит до бази даних, яка у відповідь користувачу повертає дані для перегляду. Також користувач має можливість додавати, редагувати, видаляти та вибирати за ідентифікатором інформацію яку потрібно, в залежності від маніпуляцій із даними, які він бажає здійснити, MySQL сервер надсилає запит до бази даних і база даних, в свою чергу, повертає серверу дані і після чого користувач може переглянути результат маніпуляцій з даними.

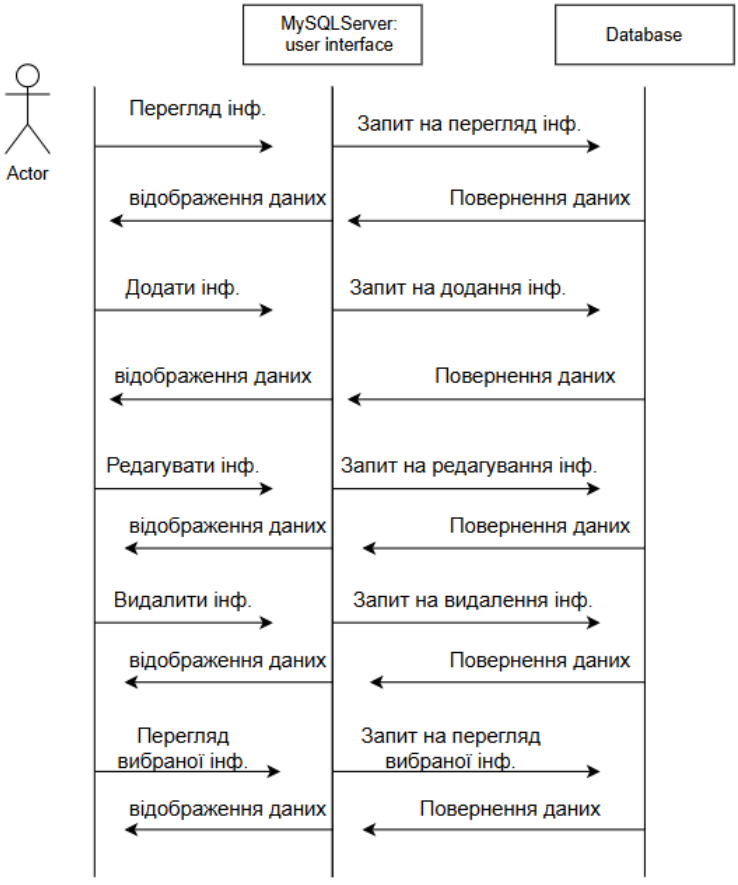

Рисунок 2.5 - Діаграма взаємодії із сервером

Іншими словами, діаграма послідовностей, яка зображена на рисунку 2.5 відображає часові особливості передачі і прийому повідомлень об'єктами.

Після створення бази даних та визначення дій, які користувач може виконувати, починаємо розробку програмного забезпечення. Для цього нам потрібно визначити структуру інформаційної системи. Наша система складатиметься з двох основних компонент із своїм функціоналом фронтенду та бекенду, оскільки поставленою задачею є створення серверної частини інформаційної системи, зосередимося на описі бекенду.

Структурна схема серверної частини інформаційної системи матиме наступний вигляд:

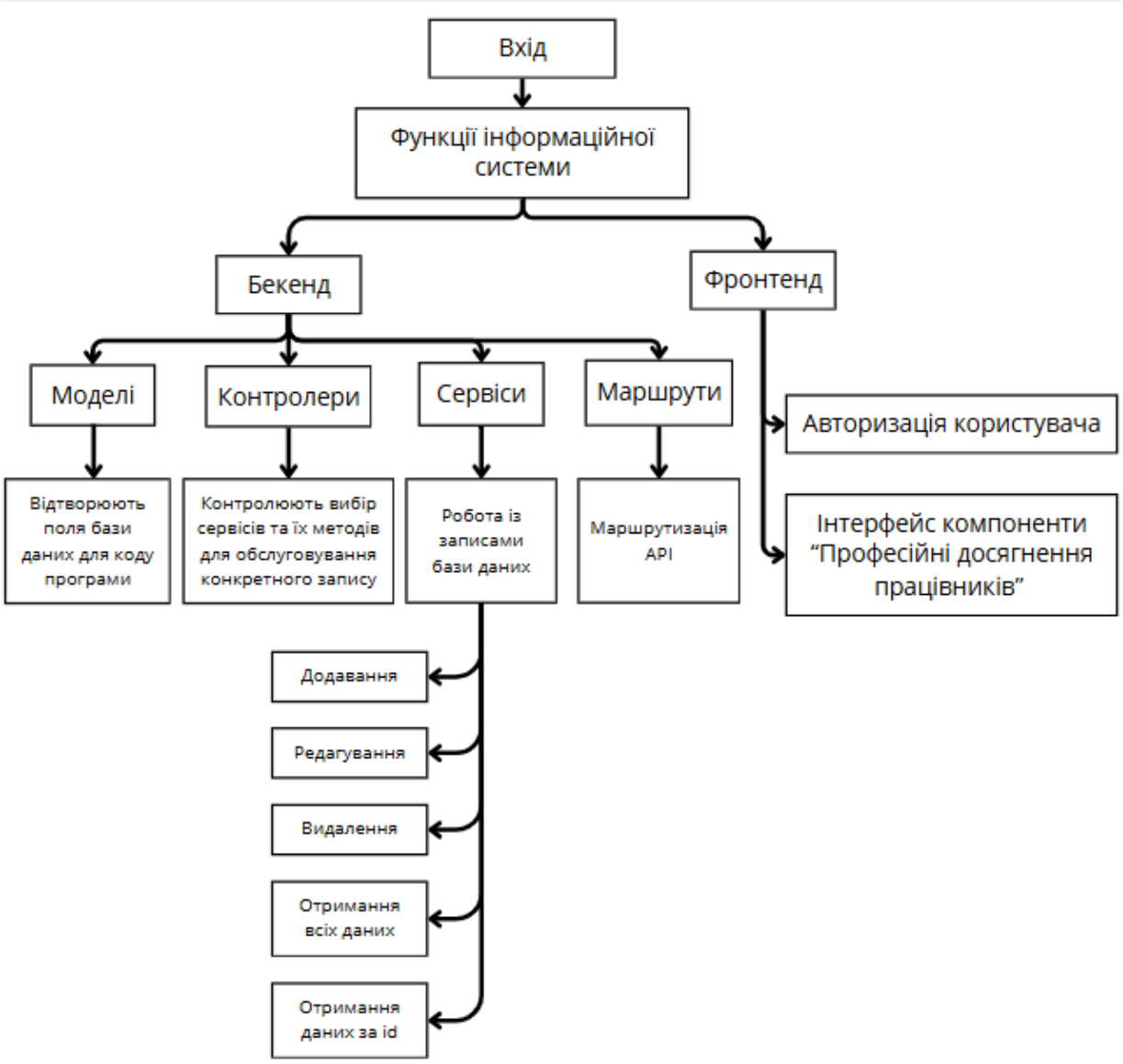

Рисунок 2.6 – Структурна схема серверної частини інформаційної системи

На рисунку 2.6 показано структуру проекту та дії, які виконують основні компоненти. Щодо фронтенду, визначено тільки базові потреби для інформаційної системи. Користувач через браузер виконуватиме авторизацію та матиме інтерфейс через, який взаємодіятиме із інформаційною системою.

Для бекенд частини описано детально структуру інформаційної системи. Моделі, Контролери, Сервіси та Маршрути є файлами, або папками із цими файлами, які мають свою зрозумілу назву та свій програмний код. Кожен із цих файлів відповідає за конкретні дії із базою даних та сервером. В результаті поєднання всього функціоналу, отримаємо працездатну компоненту про "Професійні досягнення працівників" для інформаційної системи університету. Користувач матиме можливість переглядати, редагувати, додавати та видаляти інформацію із бази даних для розроблюваної компоненти.

3 РОЗШИРЕННЯ ФУНКЦІОНАЛУ ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1 Розробка програмного коду

Для підключення до бази даних будемо використовувати файли DBProvider.php та config.php:

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

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

Код, який показано вище із файлу DBProvider.php відповідає за саме з’єднання із базою даних до проекту. Для створення підключення до бази даних потрібні такі параметри адреса сервера бази даних (наприклад, "localhost"), ім'я користувача(логін), пароль користувача та назва бази даних [16].

Самі дані для підключення та додаткові ресурси для проведення автентифікації користувача вказані у файлі 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";
```

Звернемо увагу на основні дані для створення підключення до бази даних. Оскільки робота проводиться на локальному сервері, тому назва відповідно localhost. При реєстрації для доступу до локального сервера, використано назву root, пароль не вказано, щоб не втратити дані і легко їх запам'ятовувати. Останнім параметром є назва бази даних, якщо один з параметрів буде мати не коректний символ, то підключення не відбудеться.

Дальше розглянемо створення програмного коду для одної із таблиць бази даних, яка розроблялася для компоненти "Професійні досягнення працівників", нехай це буде таблиця Education. Для неї маємо створити згідно структури проекту код моделі, контролера, сервісу та маршруту.

Код файлу Education.php:

```
<?php
    require_once('BaseModel.php');
    class Education extends BaseModel {
        private $userid;
        private $title;
        private $speciality;
        private $diplomanumber;
        private $issuedate;
        private $proofurl;

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

```
            $this->title = $paramArray['title'];
            $this->speciality = $paramArray['speciality'];
            $this->diplomanumber = $paramArray['diplomanumber'];
            $this->issuedate = $paramArray['issuedate'];
            $this->proofurl = $paramArray['proofurl'];
        }

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

Реалізовано модель для таблиці бази даних, яка містить інформацію про освіту. Конструктор ініціалізує дані, використовуючи асоціативний масив $paramArray. Кожне приватне поле (наприклад, $userid, $title) заповнюється відповідним значенням із масиву. Метод getAsObject() використовує функцію get_object_vars, щоб повернути властивості вказаного об'єкта у вигляді асоціативного масиву [17].

Тобто асоціативний масив у цьому коді моделі представляє дані, отримані з бази даних, і використовується для заповнення властивостей об'єкта класу Education. Ключі масиву (id, userid, title, і т.д.) відповідають назвам колонок у таблиці бази даних, а значення ключів - даним цих колонок.

Код для роботи із записами бази даних EducationService.php:

```
<?php
    require_once('./app/services/BaseService.php');
    require_once('./app/models/Education.php');
    class EducationService extends BaseService {
        static $tableName='am_education';
```

Виконує додавання об'єкта освіти Education до локального масиву даних ($dataArray) у вигляді екземпляра моделі.

Якщо параметр id не заданий, створюється новий унікальний id,

збільшуючи лічильник $this->index. Створюється новий об'єкт класу Education, заповнений даними з $paramArray. Додає цей об'єкт у масив dataArray. Та повертає id нового запису.

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

Виконує вставлення нового запису у таблицю бази даних із даними про освіту, завдяки виконанню відповідного SQL-запиту INSERT INTO, bind_param() прив’язує запиту відповідні дані для занесення у базу даних.

```
public function insertToDatabase($conn, $paramArray) {
    $request = $conn->prepare("INSERT INTO ".$this::$tableName." VALUES (DEFAULT, ?,?,?,?,?,?)");
    $request->bind_param("ssssss", $userid, $title, $speciality, $diplomanumber, $issuedate, $proofurl);

    $userid = $paramArray['userid'];
    $title = $paramArray['title'];
    $speciality = $paramArray['speciality'];
    $diplomanumber = $paramArray['diplomanumber'];
    $issuedate = $paramArray['issuedate'];
    $proofurl = $paramArray['proofurl'];
    $request->execute();
    $request->close();
}
```

Виконує видалення запису із бази даних за заданим id, за допомогою SQL-запиту DELETE, де значення id передається через bind_param().

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

Виконує оновлення запису у базі даних за заданим id, використовуючи SQL-запит UPDATE для оновлення всіх полів таблиці, значення для запиту передаються через bind_param().

```
    public function updateDatabaseById($conn, $id, $paramArray) {
                $request = $conn->prepare("UPDATE ".$this::$tableName."
SET `userid`=?, `title`=?, `speciality`=?, `diplomanumber`=?,
`issuedate`=?, `proofurl`=? WHERE `id`=?");
                $request->bind_param("sssssss", $userid, $title,
$speciality, $diplomanumber, $issuedate, $proofurl, $id);
                $userid = $paramArray['userid'];
                $title = $paramArray['title'];
                $speciality = $paramArray['speciality'];
                $diplomanumber = $paramArray['diplomanumber'];
                $issuedate = $paramArray['issuedate'];
                $proofurl = $paramArray['proofurl'];
                $request->execute();
                $request->close();
            }
```

Виконує отримання запису із бази даних за заданим id і додає його до масиву даних, використовуючи SQL-запит SELECT, де id має конкретне значення. Якщо знайдено дані, кожний запис додається в масив dataArray через функцію add(). Якщо результат порожній, нічого не робиться.

```
    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 "";
            }
        }
```

Виконує отримання всіх записів з таблиці та впорядковує їх за id, використовуючи SQL-запит SELECT для отримання всіх записів та ORDER BY для впорядкування. Якщо знайдено результати, кожний запис додається в масив dataArray через функцію add(). Якщо немає результатів, нічого не робиться.

```
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 "";}
        }
    }
?>
```

Ці всі методи знаходяться у файлі сервісу, створеннні для маніпуляцій із базою даних та використовуються контролером для виконання HTTP-методів.

Наступним йде код самого контролера, розглянемо як він працює.

Код контролера EducationController.php:

```
<?php
require_once ("./app/services/EducationService.php");
require_once ("./app/controllers/BaseController.php");
$educationService = new EducationService();
```

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

Цей код не містить важливого функціонування оскільки створює лише екземпляри EducationService() та BaseController() відповідно від сервісів він бере запити для обробки даних, від базового контролера обробку цих запитів в залежності від HTTP-методу.

Код контролера BaseController.php:

```
<?php
class BaseController{
    private $list;
    public $conn;
    public function __construct($conn,$list){
        $this->list=$list;
        $this->conn=$conn;  }
```

Основний метод processRequest(), який виконує конкретну дію залежно від типу HTTP-запиту.

```
  public function processRequest(){
```

Отримання даних з бази даних. Якщо параметр id заданий у запиті ($_REQUEST['id']), отримує конкретний запис із бази даних за цим ідентифікатором, то виконується метод getFromDatabaseById. Якщо параметр id не заданий, отримує всі записи з бази за методом getAllFromDataBase.

```
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();}
```

Додає новий запис до бази даних, використовуючи метод insertToDatabase сервісу для вставки запису до бази даних.

```
    } 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();
```

Оновлює існуючий запис у базі даних, використовуючи метод updateDatabaseById для оновлення даних запису, вибраного за id.

```
    } 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();
```

Видаляє запис із бази даних, за його ідентифікатором, використовуючи метод deleteFromDatabase.

```
    } 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"}';
            }
        }
    }
```

В підсумку контролер для таблиці Education створює екземпляри потрібних класів та імплементує їхні методи у своїх цілях. В цьому випадку використовує методи сервісів для роботи із записами бази даних, та метод базового контролера, який від дій користувача задіює конкретний метод сервісу.

Останнім фрагментом коду буде частина маршруту, яка буде додаватися до основної частини шляху та формуватиме посилання на конкретну сторінку інформаційної системи.

Фрагмент коду маршруту AppRouter.php:

```
    case $AMProjectBaseUrl.'/educations' :
                            require_once
("./app/controllers/EducationController.php");
                            break;
```

Оскільки в нас є багато різних компонент інформаційної системи вирішено для маршрутизації використати конструкцію switch, яка допоможе задавати конкретний ресурс [18].

Якщо значення змінної збігається з case $AMProjectBaseUrl.'/educations', то виконується код у цьому блоці.

Підключення файлу EducationController.php, відбувається через require_once, тобто задіює методи для роботи із базою даних конкретно із таблицею Education. Завершення виконання блоку case відбувається через команду break.

Загалом функціонал бекенду такий, користувач переходить по різних

сторінках інформаційної системи, які мають свій URL (маршрут). В залежності від маршруту задіюється відповідний контролер, який приймає від користувача HTTP-запити, задіює відповідні методи сервісів та виконує роботу із записами бази даних. Модель створена для відтворення полів таблиць бази даних у коді програми.

За таким прикладом створюємо решту програмного коду для інших таблиць бази даних.

3.2 Перевірка працездатності інформаційної системи

Коли створено програмний код можна перейти до перевірки працездатності інформаційної системи, для цього нам потрібно виконати всі HTTP-запити.

Hypertext Transfer Protocol (HTTP) (Протокол передачі гіпертексту) - призначений для забезпечення зв'язку між клієнтами та серверами.

HTTP працює як протокол запиту-відповіді між клієнтом та сервером.

Веб-браузер може бути клієнтом, а додаток на комп'ютері, на якому розміщений веб-сайт, може бути сервером.

Приклад: клієнт (браузер) відправляє HTTP-запит на сервер; потім сервер повертає відповідь клієнту. Відповідь містить інформацію про стан запиту і може також містити запитуваний контент [19].

Тепер зрозуміло, як виконуються HTTP-запити за допомогою методів GET, POST, PUT та DELETE, які розглядали раніше у файлі базового контролера.

Перевіримо роботу розробленої компоненти у браузері звернемося до даних у таблиці Education, оскільки її взяли для огляду.

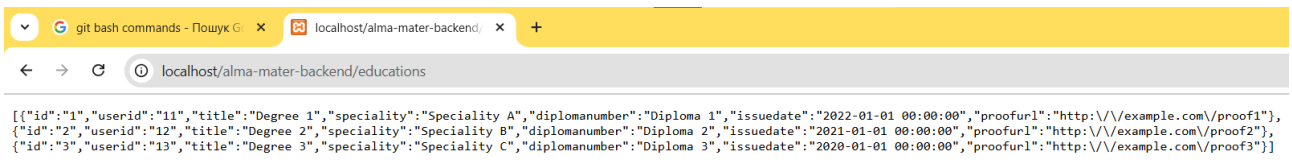

Рисунок 3.1 – Результат виконання методу GET

На рисунку 3.1 показано, що перейшовши за відповідним посиланням у браузері спрацьовує метод GET та виводить інформацію, яка є в таблиці із даними про освіту. Варто зазначити, оскільки ми працюємо на локальному сервері, то автентифікація завжди буде помилковою. Тому нам треба відключити перевірку в коді на Bearer Token, просто за коментувавши її. Проте цей підхід не є коректним, також не маючи фронтенд частини інформаційної системи, щоб перевірити весь функціонал потрібно скористатися програмним забезпеченням Postman.

Postman — це індійська глобальна компанія програмного забезпечення, яка пропонує розробникам платформу API для проектування, створення , тестування та співпраці над API [20].

Це програмне забезпечення зможе замість браузера, використовуючи бекенд частину інформаційної системи та реальну автентифікацію через Bearer Token виконати потрібні HTTP-запити.

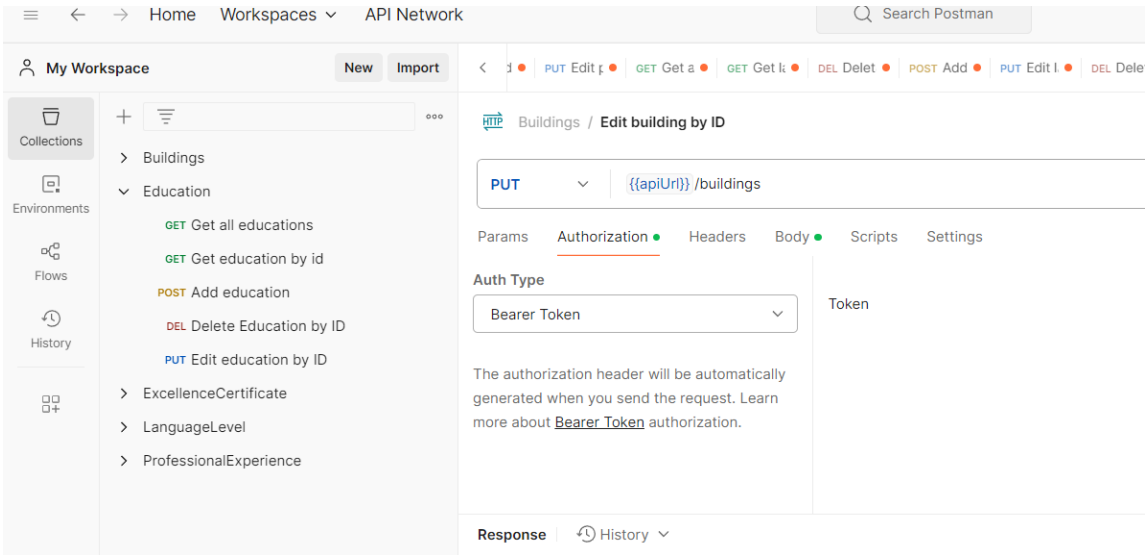

Рисунок 3.2 – Робоче середовище Postman

На рисунку 3.2 бачимо, що у нас є в лівому меню колекції, кожна з яких має свої методи, які виконує контролер, а саме GET (усі записи), GET (за id), POST, DELETE та PUT. У пункті авторизації маємо вказати тип Bearer Token та надати значення цього токена, щоб успішно пройшла авторизація.

Протокол Bearer є важливою частиною авторизаційної структури OAuth 2.0. Він дозволяє безпечно отримувати доступ до веб-ресурсів, не розкриваючи облікові дані користувача. Bearer Token носій виступає як доказ авторизації, дозволяючи його власнику отримати доступ до захищених ресурсів на основі дозволів, пов'язаних із токеном.

Як працює протокол Bearer?

Коли користувач надає додатку авторизацію для доступу до своєї інформації, додаток запитує токен доступу у сервера авторизації. Цей токен доступу потім включається в HTTP-запити до сервера ресурсів. Цей токен слугує доказом згоди користувача і дозволяє додатку отримати доступ до запитаних ресурсів. Bearer Token носій включається в заголовок «Authorization»HTTP-запиту, зазвичай у форматі «Authorization: Bearer <token>».

Протокол Bearer спрощує процес доступу до захищених ресурсів, використовуючи механізм автентифікації на основі токенів. Токен-бере носій, виданий сервером авторизації, дозволяє додатку ідентифікувати та авторизувати користувача без необхідності розкривати облікові дані користувача. Це забезпечує безпечний і зручний спосіб отримання доступу до веб-ресурсів [21].

Тепер зрозуміло, які переваги з огляду на безпеку надає авторизація користувача через Bearer Token. Коли все налаштовано перевіримо роботу цих методів, відправивши відповідні їм HTTP-запити.

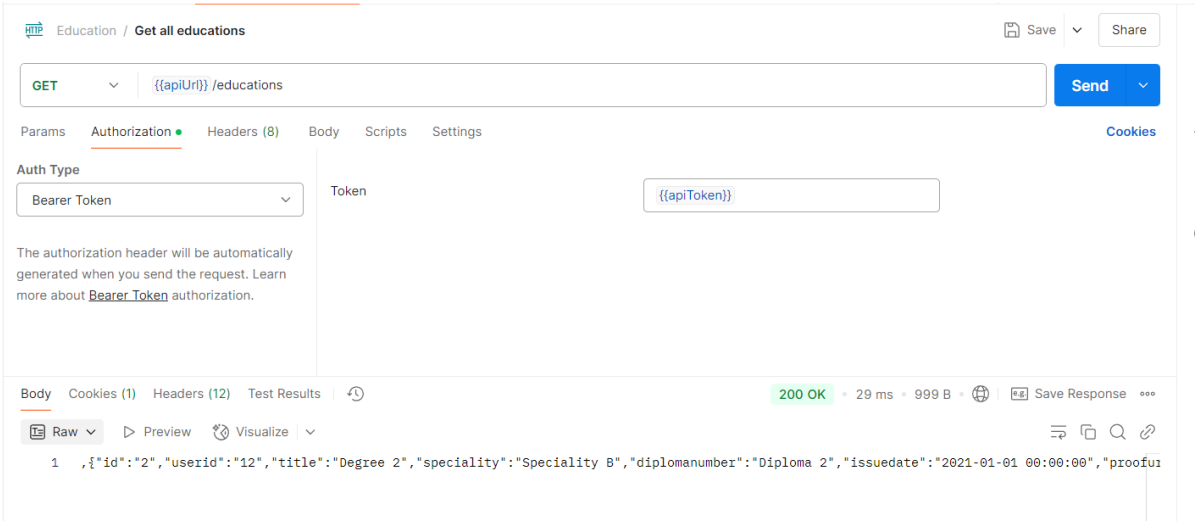

Рисунок 3.3 – Виконання методу GET для всіх записів

На рисунку 3.3 можна побачити, що, використовуючи конкретний маршрут та метод для HTTP-запиту, натискаємо кнопку синього кольору “Send” та отримуємо відповідь з успішним виконанням. Статус відповіді 200, що означає про успіх виконання запиту і нижче у вигляді відповіді бачимо дані, які користувач отримує в результаті успішного завершення методу GET.

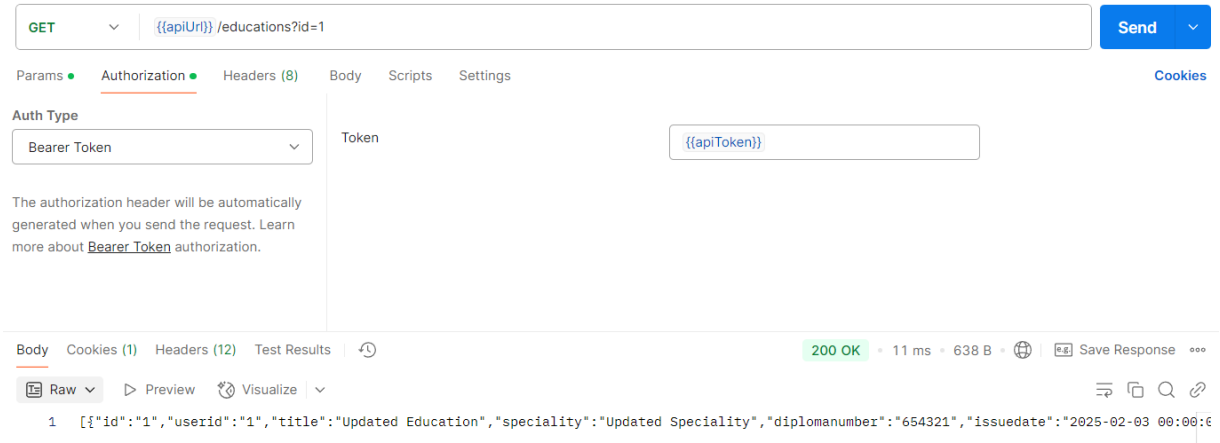

Рисунок 3.4 – Виконання методу GET за id

На рисунку 3.4 бачимо виконання також методу GET, але тут отримання даних відбувається за вказаним id. За виглядом посилання зрозуміло, що хочемо отримати запис із ідентифікатором рівним одиниці. Запит успішно виконано, оскільки статус відповіді 200, і отримано інформацію про запис з id один.

Перейдемо до виконання методу POST, в який передамо нові дані, для їх занесення у таблицю бази даних.

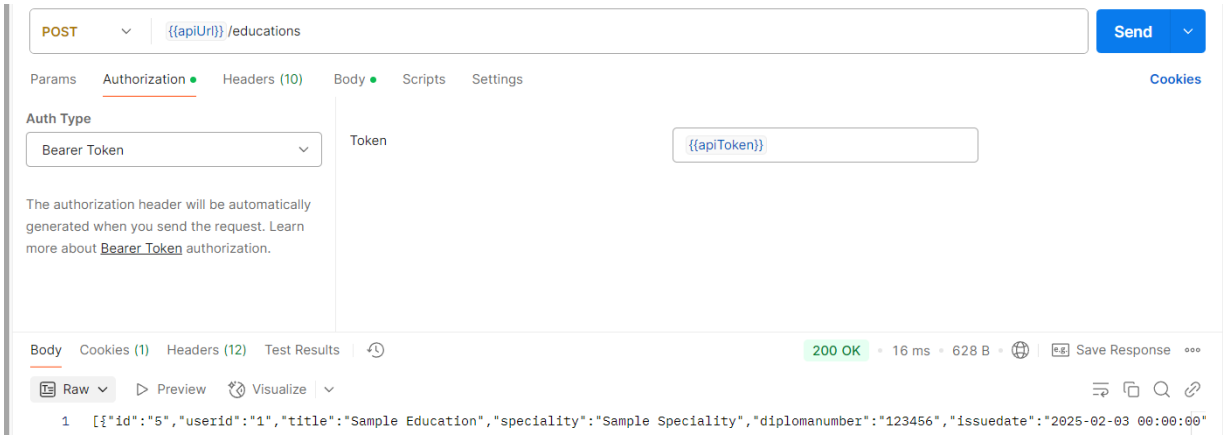

Рисунок 3.5 – Виконання методу POST

На рисунку 3.5 бачимо виконання методу POST. Він також успішно виконується, в результаті отримуємо даний запис із id 5, ідентифікатор має номер п’ять, оскільки вже виконувалося раніше додавання запису і через те, що первинний ключ, є автоінкрементом ми отримуємо запис із ідентифікатором 5.

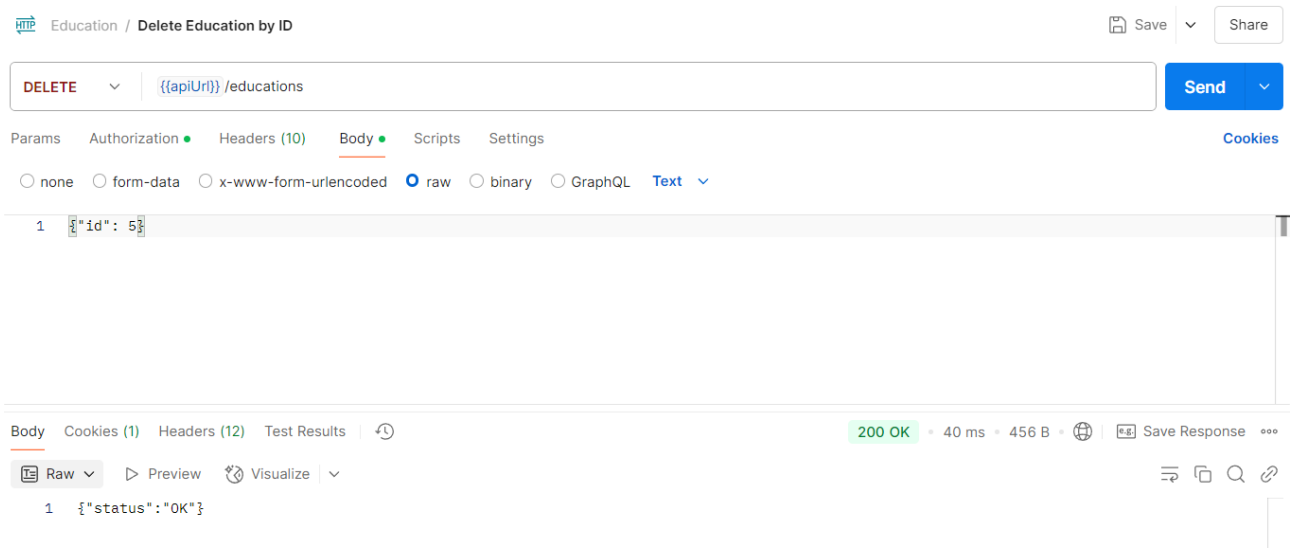

Рисунок 3.6 – Виконання методу DELETE

На рисунку 3.6 бачимо виконання методу DELETE, у пункті “Body” вибираємо опцію “raw”, яка у зрозумілому вигляді дозволяє вказати, який запис хочемо видалити за його ідентифікатором. Вибираємо id 5, щоб видалити запис, який щойно додали. Запит виконався успішно і бачимо відповідь, яку він повернув.

Рисунок 3.7 – Перевірка виконання методів POST та DELETE

На рисунку 3.7 можна помітити, що останнім записом у таблиці бази даних Education є запис з ідентифікатором чотири. З цього робимо висновок, що метод POST додав запис із id 5, потім метод DELETE видалив його.

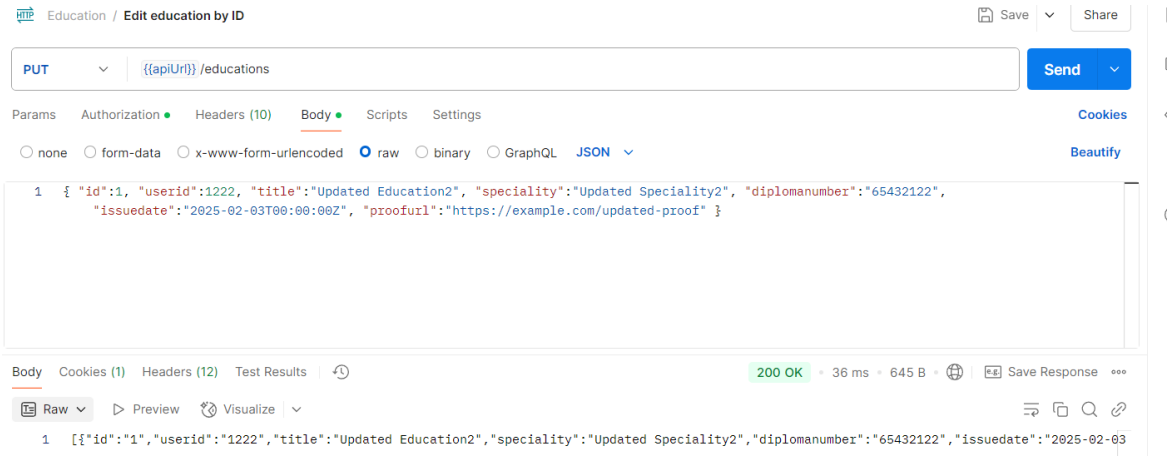

Рисунок 3.8 – Виконання методу PUT

На рисунку 3.8 бачимо виконання методу PUT, у пункті “Body” так само вибираємо опцію “raw”, яка у зрозумілому вигляді дозволяє витягнути запис із таблиці бази даних і відредагувати його значення. Запит виконався успішно і бачимо відповідь, яку він повернув. Отримали запис із таким самим id, але вже із оновленою інформацією.

В результаті перевірки працездатності розробленої компоненти про “Професійні досягнення працівників” для серверної частини інформаційної системи університету, переконалися, що бекенд програми працює коректно. Методи, які відповідають за виконання HTTP-запитів з боку користувача та маніпуляцій із базою даних видають очікувані результати.

3.3 Додаткові специфікації

Згідно постанови Про затвердження Ліцензійних умов провадження освітньої діяльності №1187, вирішено доповнити функціонал вибору інформації для розроблюваної компоненти про професійні досягнення працівників університету.

Для всіх таблиць, які є в цій компоненті, додатково зробити вибірку інформації за конкретним userid. Для цього потрібно вдосконалити контролер та дописати метод сервісу для виконання такої дії.

Розглянемо таку модифікацію на прикладі роботи із таблицею am_excellence_certificate.

Код контролера ExcellenceCertificateController.php:

```
<?php
require_once("./app/services/ExcellenceCertificateService.php");
require_once("./app/controllers/BaseController.php");
$excellenceCertificateService = new ExcellenceCertificateService();
if (!isset($_REQUEST['action'])) {
    $controller = new BaseController($AMDBConnection,
$excellenceCertificateService);
    $controller->processRequest();
} else if ($_REQUEST['action'] == 'getUserByID') {
    $excellenceCertificateService-
>getFromDatabaseByUserID($AMDBConnection, $_REQUEST['userid']);
    echo $excellenceCertificateService->getAsJSON();
} else if ($_REQUEST['action'] == 'getUserByIDLast5Years') {
    $excellenceCertificateService-
>getFromDatabaseByUserIDLast5Years($AMDBConnection, $_REQUEST['userid']);
    echo $excellenceCertificateService->getAsJSON();
} else if ($_REQUEST['action'] == 'getTotalHoursLast5Years') {
    $excellenceCertificateService-
>getTotalHoursLast5YearsByUserID($AMDBConnection, $_REQUEST['userid']);
    echo json_encode($excellenceCertificateService->summaryData,
JSON_UNESCAPED_UNICODE);
}
?>
```

За реалізацію вище вказаної вибірки відповідає запит із action == getUserByID, він задіює відповідний методу сервісу, який в цьому випадку вибиратиме записи для конкретного працівника про його підвищення кваліфікації.

Код сервісу ExcellenceCertificateServices.php:

```
    public function getFromDatabaseByUserID($conn, $userid) {
                $request = $conn->prepare("SELECT `" . $this::$tableName
. "`.* FROM `" . $this::$tableName . "` WHERE userid = ?");
                $request->bind_param("s", $userid);
                $request->execute();
                $result = $request->get_result();
                if ($result->num_rows > 0) {
                    while ($row = $result->fetch_assoc()) {
                        $this->add($row);  }
                } else {
                    echo "";
                }
            }
```

Цей метод базується на виконанні запиту до бази даних, де додатковим параметром вибірки інформації є userid. За допомогою оператора WHERE, який використовується для фільтрації рядків у результаті запиту, можна реалізувати таку вибірку інформації [22].

Для решти таблиць цієї компоненти логіка коду аналогічна. Наступною модифікацією коду є вибірка інформації із всіх таблиць створеної компоненти за userid та за останні 5 років. У цьому випадку додатковою є умова часу. Для виконання цього запиту у контролері є спеціальна умова action == getUserByIDLast5Years. Вона задіює відповідний метод сервісу.

Код сервісу ExcellenceCertificateServices.php:

```
    public function getFromDatabaseByUserIDLast5Years($conn, $userid) {
                $query = "SELECT `" . $this::$tableName . "`.* FROM `" .
$this::$tableName . "`
                            WHERE   userid   =   ?   AND   issuedate   >=
DATE_SUB(NOW(), INTERVAL 5 YEAR)";

                $request = $conn->prepare($query);
                $request->bind_param("s", $userid);
```

```
            $request->execute();
            $result = $request->get_result();
            if ($result->num_rows > 0) {
                while ($row = $result->fetch_assoc()) {
                    $this->add($row);
                }
            } else {
                echo "";
            }
        }
```

Метод подібний до попереднього тільки в запиті SELECT міститься додаткова умова issuedate >= DATE_SUB(NOW(), INTERVAL 5 YEAR), де issuedate є полем в таблиці бази даних із датою в цьому випадку підвищення кваліфікації. DATE_SUB допомагає знайти інтервал часу в 5 років від сьогоднішньої дати. Тобто реалізує вибір записів за останні 5 років [23].

Для таблиці із даними про підвищення кваліфікації також потрібно вибрати інформацію про суму годин, які пішли на це підвищення і також за останні 5 років. Контролер виконує цей запит коли задіюється action == getTotalHoursLast5Years. Ось метод сервісу.

Код сервісу ExcellenceCertificateServices.php:

```
    public $summaryData = [];

        public function getTotalHoursLast5YearsByUserID($conn, $userid) {
            $query = "SELECT SUM(hours) AS total_hours FROM `" . $this::$tableName . "`
                    WHERE userid = ? AND issuedate >= DATE_SUB(NOW(), INTERVAL 5 YEAR)";

            $request = $conn->prepare($query);
            $request->bind_param("s", $userid);
            $request->execute();
            $result = $request->get_result();
```

```
                if ($result->num_rows > 0) {
                    $row = $result->fetch_assoc();
                    $this->summaryData = ["total_hours" => 
$row["total_hours"] ?? 0];
                } else {
                    $this->summaryData = ["total_hours" => 0];
                }
            }
```

Цей метод відрізняється від попередніх оскільки має іншу логіку пошуку інформації. У цьому випадку в запиті SELECT для підрахунку суми годин використано агрегатну функцію SUM(). Її особливість полягає в тому, що вона повертає одне значення після виконання запиту, в нашому випадку суму годин, якщо відповідних даних не знайдеться для підрахунку суми годин, функція поверне NULL.

Наші методи сервісів орієнтовані на роботу із масивами даних, а в цьому випадку ми отримуємо одне значення, тому було вирішено завдяки змінній summaryData обійти проблему із обробкою масивів даних, а також додали логіку при якій результат NULL буде вважатися 0, щоб не пошкодити обчислення суми годин.

Останнім додатковим функціоналом є підрахунок стажу користувача для таблиці із професійним досвідом.

Ось код контролера ProfessionalExperienceController.php:

```
<?php
require_once("./app/services/ProfessionalExperienceService.php");
require_once("./app/controllers/BaseController.php");
$professionalExperienceService = new 
ProfessionalExperienceService();
……
} else if ($_REQUEST['action'] == 'getTotalExperienceYears') {
```

```
        $professionalExperienceService-
>getTotalExperienceYearsByUserID($AMDBConnection, $_REQUEST['userid']);
        echo    json_encode($professionalExperienceService->summaryData,
JSON_UNESCAPED_UNICODE);
    }?>
```

За виконання методу сервісів для підрахунку стажу потрібно задіяти запит контролера із action == getTotalExperienceYears.

Код сервісу ProfessionalExperienceService.php:

```
    public $summaryData = [];

            public    function    getTotalExperienceYearsByUserID($conn,
$userid) {
                $query = "SELECT
                            SUM(TIMESTAMPDIFF(MONTH,          startdate,
IFNULL(enddate, NOW()))/12 AS total_years
                            FROM `" . $this::$tableName . "`
                            WHERE userid = ?";
                            $request = $conn->prepare($query);
                $request->bind_param("s", $userid);
                $request->execute();
                $result = $request->get_result();
                if ($result->num_rows > 0) {
                    $row = $result->fetch_assoc();
                    $this->summaryData      =      ["total_years"      =>
round($row["total_years"] ?? 0, 2)];
                } else {
                    $this->summaryData = ["total_years" => 0];
                }
            }
```

Бачимо, що логіка цього методу подібна до методу із підрахунком суми годин, оскільки ці методи використовують агрегатну функцію SUM(). Відрізняється тільки вирахування стажу у запиті SELECT. Функція TIMESTAMPDIFF вираховує різницю між двома датами початком та кінцем

роботи. Ця різниця подається у місяцях, тому щоб перевести її у роки отримане значення ділимо на 12. Також враховуємо, що поле enddate може бути NULL, бо працівник університету, досі працює тоді ми вибираємо теперішню дату і реалізуємо цю логіку через функцію IFNULL. Весь код можна переглянути в додатку А.

Продемонструємо роботу нового функціоналу.

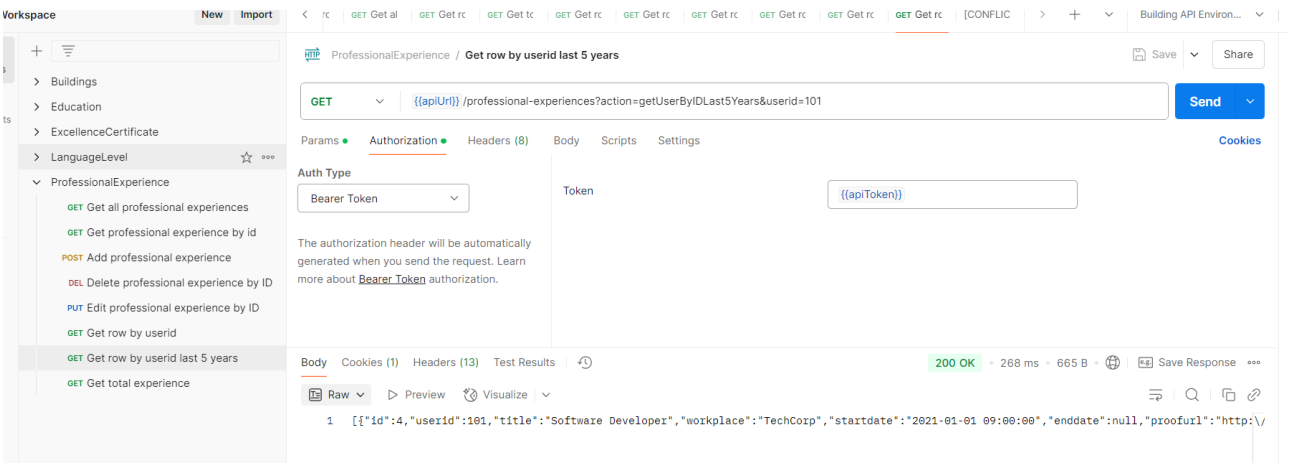

Рисунок 3.9 – Виконання методу GET до таблиці am_professional_experience

На рисунку 3.9 бачимо виконання методу GET за userid та часом в останні 5 років. Спеціально в запиті вказуємо конкретний action та передаємо userid. Контролер задіює відповідний метод сервісу та передає дані (userid) метод виконується і повертає запис із таблиці, в якому міститься вся інформація про професійний досвід користувача з id =101. Решта методів відпрацювали також успішно.

ВИСНОВКИ

В ході виконання цієї роботи було розроблено компоненту про “Професійні досягнення працівників” для серверної частини інформаційної системи університету, яка відслідковує та надає доступ до даних про професійні досягнення працівників університету. Вона включає в себе інформацію про кваліфікації, досвід, вчені звання, наукові ступені та рівень володіння англійською мовою.

Проведено аналіз сайтів аналогів та інформацію, яку вони надають своїм користувачам, визначено переваги та недоліки цих сайтів, щоб покращити власну інформаційну систему. Визначено ключові атрибути та способи представлення даних, які були використані при проектуванні бази даних. Визначення цих атрибутів та обмежень дозволяє користувачу отримувати детальну інформацію про професійні досягнення працівників університету.

Було розроблена структура бази даних, яка містить інформацію про професійні досягнення працівників. Ця структура бази даних була розроблена з урахуванням вимог до інформаційних систем та вимог до зберігання даних.

Інформаційна система, яка була розроблена в рамках цієї роботи, дозволяє виконувати типові операції з базою даних, такі як отримання, додавання, видалення та редагування даних. Результати перевірки працездатності інформаційної системи показали, що вона працює коректно та забезпечує виконання CRUD-методів.

Розроблена інформаційна система може бути корисною для освітніх закладів, щоб вести статистику про працівників цих закладів і їх професійні досягнення.

Таким чином, завдяки виконаній роботі досягнута мета бакалаврської роботи – розробка серверної частини інформаційної системи, яка зберігатиме дані про професійні досягнення працівників університету. Надаватиме доступ до даних адміністрації університету, викладачам та іншим користувачам системи.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Про освіту - Всі документи - Верховна Рада України. URL: https://zakon.rada.gov.ua/laws/show/2145-19#Text (дата звернення: 02.06.2025).

2. Про вищу освіту» від 01.07.2014 № 1556-VII - Всі документи. URL: https://zakon.rada.gov.ua/laws/show/1556-18#Text (дата звернення: 02.06.2025).

3. Про захист персональних даних. URL: https://zakon.rada.gov.ua/laws/show/2297-17#Text (дата звернення: 02.06.2025).

4. Про наукову і науково-технічну діяльність - Всі документи. URL: https://zakon.rada.gov.ua/laws/show/848-19#Text (дата звернення: 02.06.2025).

5. Про затвердження Ліцензійних умов провадження освітньої діяльності. URL: https://zakon.rada.gov.ua/laws/show/1187-2015-%D0%BF#Text (дата звернення: 30.05.2025).

6. Єдина державна електронна база з питань освіти. URL: https://uk.wikipedia.org/wiki/Єдина_державна_електронна_база_з_питань_освіти (дата звернення: 18.02.2025).

7. Prometheus. URL: https://uk.wikipedia.org/wiki/Prometheus (дата звернення: 9.03.2025).

8. EdEra — студія онлайн-освіти. URL: https://ed-era.com/about-us/ (дата звернення: 9.03.2025).

9. Переваги та недоліки використання мови PHP для Backend розробки. URL: https://smart-solutions.com.ua/archives/637 (дата звернення: 26.05.2025).

10. Що таке Visual Studio Code 2025. URL: https://top-keis.com.ua/shho-take-visual-studio-code/ (дата звернення: 26.05.2025).

11. XAMPP сервер: встановлення і налаштування в ОС Linux. URL: https://kr-labs.com.ua/blog/xampp-setup-and-install-in-linux/ (дата звернення: 26.05.2025).

12. Ключі. Зв'язки між записами і таблицями – Мій Клас. 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: 16.03.2025).

13. Лекція: Залежності між таблицями у базі даних. URL: https://javarush.com/ua/quests/lectures/ua.questhibernate.level17.lecture03 (дата звернення: 26.05.2025).

14. Дамп бази даних MySQL: що це таке і як зробити. URL: https://hostiq.ua/wiki/ukr/migrate-mysql-database/ (дата звернення: 26.05.2025).

15. ALTER TABLE в SQL - aCode. URL: https://acode.com.ua/alter-table-sql/ (дата звернення: 26.05.2025).

16. Підключення до баз даних MySQL із використанням PHP. URL: https://www.godaddy.com/uk-ua/help/pidklyuchennya-do-baz-danih-mysql-iz-vikoristannyam-php-216 (дата звернення: 26.05.2025).

17. get_object_vars(). URL: https://php.org.ua/manual/uk/function.get-object-vars.md (дата звернення: 26.05.2025).

18. Довідник по основам PHP — switch | База знань IT - IT Wiki. URL: https://itwiki.dev/php/php-reference/language-control-structures/switch (дата звернення: 26.05.2025).

19. HTTP Методи запиту - W3Schools українською - GitHub Pages. URL: https://w3schoolsua.github.io/tags/ref_httpmethods.html#gsc.tab=0 (дата звернення: 26.03.2025).

20. Postman (software). URL: https://en.wikipedia.org/wiki/Postman_(software) (дата звернення: 26.03.2025).

21. Що таке Bearer Protocol - Терміни та визначення у сфері. URL: https://www.vpnunlimited.com/ua/help/cybersecurity/bearer-protocol?srsltid=AfmBOooGyXCBidy9OFa3gCStsGvsD6ppiCA7xOysmmwwm4BAouQL6gMs (дата звернення: 27.03.2025).

22. Що таке WHERE в SQL? Онлайн довідник на itProger. URL: https://itproger.com/ua/spravka/sql/where (дата звернення: 26.05.2025).

23. Функції роботи з датою, практичні приклади. URL: https://javarush.com/ua/quests/lectures/ua.questhibernate.level04.lecture03 (дата звернення: 26.05.2025).

ДОДАТКИ

ДОДАТОК А

Код компоненти “Професійні досягнення працівників” серверної частини інформаційної системи університету

Education.php

```
<?php
    require_once('BaseModel.php');
    class Education extends BaseModel {
        private $userid;
        private $title;
        private $speciality;
        private $diplomanumber;
        private $issuedate;
        private $proofurl;
        public function __construct($paramArray) {
            $this->id = $paramArray['id'];
            $this->userid = $paramArray['userid'];
            $this->title = $paramArray['title'];
            $this->speciality = $paramArray['speciality'];
            $this->diplomanumber = $paramArray['diplomanumber'];
            $this->issuedate = $paramArray['issuedate'];
            $this->proofurl = $paramArray['proofurl'];
        }
        public function getAsObject() {
            return get_object_vars($this);
        }
}
?>
```

ExcellenceCertificate.php

```
<?php
    require_once('BaseModel.php');
    class ExcellenceCertificate extends BaseModel {
        private $userid;
        private $name;
        private $issuedate;
        private $hours;
        private $issuername;
        private $certificateurl;
```

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


            $this->userid = $paramArray['userid'];
            $this->name = $paramArray['name'];
            $this->issuedate = $paramArray['issuedate'];
            $this->hours = $paramArray['hours'];
            $this->issuername = $paramArray['issuername'];
            $this->certificateurl = $paramArray['certificateurl'];
        }
        public function getAsObject() {
            return get_object_vars($this);
        }}
?>
```

LanguageLevel.php

```
<?php
    require_once('BaseModel.php');
    class LanguageLevel extends BaseModel {
        private $userid;
        private $language;
        private $certificateissuer;
        private $level;
        private $issuedate;
        private $proofurl;
        public function __construct($paramArray) {
            $this->id = $paramArray['id'];
            $this->userid = $paramArray['userid'];
            $this->language = $paramArray['language'];
            $this->certificateissuer = $paramArray['certificateissuer'];
            $this->level = $paramArray['level'];
            $this->issuedate = $paramArray['issuedate'];
            $this->proofurl = $paramArray['proofurl'];
        }
        public function getAsObject() {
            return get_object_vars($this);
        }
    }
?>
```

ProfessionalExperience.php

```
<?php
    require_once('BaseModel.php');
    class ProfessionalExperience extends BaseModel {
        private $userid;
        private $title;
        private $workplace;
        private $startdate;
        private $enddate;
        private $proofurl;
        public function __construct($paramArray) {
            $this->id = $paramArray['id'];
            $this->userid = $paramArray['userid'];
```

```
            $this->title = $paramArray['title'];
            $this->workplace = $paramArray['workplace'];
            $this->startdate = $paramArray['startdate'];
            $this->enddate = $paramArray['enddate'];
            $this->proofurl = $paramArray['proofurl'];
}
        public function getAsObject() {
            return get_object_vars($this);
        }
    }
?>
```

EducationService.php

```
<?php
    require_once('./app/services/BaseService.php');
    require_once('./app/models/Education.php');
    class EducationService extends BaseService {
        static $tableName='am_education';

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

        public function insertToDatabase($conn, $paramArray) {
            $request = $conn->prepare("INSERT INTO ".$this::$tableName."
VALUES (DEFAULT, ?,?,?,?,?,?)");
            $request->bind_param("ssssss", $userid, $title, $speciality,
$diplomanumber, $issuedate, $proofurl);
            $userid = $paramArray['userid'];
            $title = $paramArray['title'];
            $speciality = $paramArray['speciality'];
            $diplomanumber = $paramArray['diplomanumber'];
            $issuedate = $paramArray['issuedate'];
            $proofurl = $paramArray['proofurl'];
            $request->execute();
            $request->close();
        }

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

        public function updateDatabaseById($conn, $id, $paramArray) {
            $request = $conn->prepare("UPDATE ".$this::$tableName." SET
```

```
`userid`=?, `title`=?, `speciality`=?, `diplomanumber`=?, `issuedate`=?,
`proofurl`=? WHERE `id`=?");
            $request->bind_param("sssssss", $userid, $title, $speciality,
$diplomanumber, $issuedate, $proofurl, $id);
            $userid = $paramArray['userid'];
            $title = $paramArray['title'];
            $speciality = $paramArray['speciality'];
            $diplomanumber = $paramArray['diplomanumber'];
            $issuedate = $paramArray['issuedate'];
            $proofurl = $paramArray['proofurl'];
            $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 getFromDatabaseByUserID($conn, $userid) {
            $request = $conn->prepare("SELECT `" . $this::$tableName .
"`.* FROM `" . $this::$tableName . "` WHERE userid = ?");
            $request->bind_param("s", $userid);
            $request->execute();
            $result = $request->get_result();

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

        public function getFromDatabaseByUserIDLast5Years($conn, $userid)
{
            $query = "SELECT `" . $this::$tableName . "`.* FROM `" .
$this::$tableName . "`
                        WHERE userid = ? AND issuedate >= DATE_SUB(NOW(),
INTERVAL 5 YEAR)";

            $request = $conn->prepare($query);
            $request->bind_param("s", $userid);
            $request->execute();
            $result = $request->get_result();
```

```
        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 "";
            }
        }
    }
?>
```

ExcellenceCertificateService.php

```
<?php
    require_once('./app/services/BaseService.php');
    require_once('./app/models/ExcellenceCertificate.php');
    class ExcellenceCertificateService extends BaseService {
        static $tableName='am_excellence_certificate';

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

        public function insertToDatabase($conn, $paramArray) {
            $request = $conn->prepare("INSERT INTO ".$this::$tableName."
VALUES (DEFAULT, ?,?,?,?,?,?)");
            $request->bind_param("ssssss",  $userid,  $name,  $issuedate,
$hours, $issuername, $certificateurl);
            $userid = $paramArray['userid'];
            $name = $paramArray['name'];
            $issuedate = $paramArray['issuedate'];
            $hours = $paramArray['hours'];
            $issuername = $paramArray['issuername'];
            $certificateurl = $paramArray['certificateurl'];
            $request->execute();
```

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

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

 public function updateDatabaseById($conn, $id, $paramArray) {
            $request = $conn->prepare("UPDATE ".$this::$tableName." SET
`userid`=?,    `name`=?,    `issuedate`=?,    `hours`=?,    `issuername`=?,
`certificateurl`=? WHERE `id`=?");
            $request->bind_param("sssssss", $userid, $name, $issuedate,
$hours, $issuername, $certificateurl, $id);
            $userid = $paramArray['userid'];
            $name = $paramArray['name'];
            $issuedate = $paramArray['issuedate'];
            $hours = $paramArray['hours'];
            $issuername = $paramArray['issuername'];
            $certificateurl = $paramArray['certificateurl'];
            $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 getFromDatabaseByUserID($conn, $userid) {
            $request = $conn->prepare("SELECT `" . $this::$tableName .
"`.* FROM `" . $this::$tableName . "` WHERE userid = ?");
            $request->bind_param("s", $userid);
            $request->execute();
            $result = $request->get_result();

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

```
    public function getFromDatabaseByUserIDLast5Years($conn, $userid) {
            $query = "SELECT `" . $this::$tableName . "`.* FROM `" .
$this::$tableName . "`
                        WHERE userid = ? AND issuedate >= DATE_SUB(NOW(),
INTERVAL 5 YEAR)";

            $request = $conn->prepare($query);
            $request->bind_param("s", $userid);
            $request->execute();
     $result = $request->get_result();

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

        public $summaryData = [];

        public function getTotalHoursLast5YearsByUserID($conn, $userid) {
            $query = "SELECT SUM(hours) AS total_hours FROM `" .
$this::$tableName . "`
                    WHERE userid = ? AND issuedate >= DATE_SUB(NOW(),
INTERVAL 5 YEAR)";

            $request = $conn->prepare($query);
            $request->bind_param("s", $userid);
            $request->execute();
            $result = $request->get_result();

            if ($result->num_rows > 0) {
                $row = $result->fetch_assoc();
                $this->summaryData = ["total_hours" =>
$row["total_hours"] ?? 0];
            } else {
                $this->summaryData = ["total_hours" => 0];
            }
        }

        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 "";
            }
        }}
?>
```

LanguageLevelService.php

```
<?php
    require_once('./app/services/BaseService.php');
    require_once('./app/models/LanguageLevel.php');
    class LanguageLevelService extends BaseService {
        static $tableName='am_language_level';

        public function add($paramArray) {

          if(!isset($paramArray['id'])) {
                $paramArray['id']=++$this->index;
            }
            $languageLevel = new LanguageLevel($paramArray);
            array_push($this->dataArray, $languageLevel);
            return $paramArray['id'];
        }

        public function insertToDatabase($conn, $paramArray) {
            $request = $conn->prepare("INSERT INTO ".$this::$tableName."
VALUES (DEFAULT, ?,?,?,?,?,?)");
            $request->bind_param("ssssss",      $userid,      $language,
$certificateissuer, $level, $issuedate, $proofurl);
            $userid = $paramArray['userid'];
            $language = $paramArray['language'];
            $certificateissuer = $paramArray['certificateissuer'];
            $level = $paramArray['level'];
            $issuedate = $paramArray['issuedate'];
            $proofurl = $paramArray['proofurl'];
            $request->execute();
            $request->close();
        }

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

        public function updateDatabaseById($conn, $id, $paramArray) {
            $request = $conn->prepare("UPDATE ".$this::$tableName." SET
`userid`=?,      `language`=?,      `certificateissuer`=?,      `level`=?,
`issuedate`=?, `proofurl`=? WHERE `id`=?");
            $request->bind_param("sssssss",      $userid,      $language,
$certificateissuer, $level, $issuedate, $proofurl, $id);
            $userid = $paramArray['userid'];
            $language = $paramArray['language'];
            $certificateissuer = $paramArray['certificateissuer'];
            $level = $paramArray['level'];
            $issuedate = $paramArray['issuedate'];
            $proofurl = $paramArray['proofurl'];
            $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 getFromDatabaseByUserID($conn, $userid) {
            $request = $conn->prepare("SELECT `" . $this::$tableName .
"`.* FROM `" . $this::$tableName . "` WHERE userid = ?");
            $request->bind_param("s", $userid);
            $request->execute();
            $result = $request->get_result();

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

        public function getFromDatabaseByUserIDLast5Years($conn, $userid)
{
            $query = "SELECT `" . $this::$tableName . "`.* FROM `" .
$this::$tableName . "`
                        WHERE userid = ? AND issuedate >= DATE_SUB(NOW(),
INTERVAL 5 YEAR)";

            $request = $conn->prepare($query);
            $request->bind_param("s", $userid);
            $request->execute();
            $result = $request->get_result();

            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 "";
            }
        }
    }
?>
```

ProfessionalExperienceService.php

```
<?php
    require_once('./app/services/BaseService.php');
    require_once('./app/models/ProfessionalExperience.php');
    class ProfessionalExperienceService extends BaseService {
        static $tableName='am_professional_experience';

        public function add($paramArray) {
            if(!isset($paramArray['id'])) {
                $paramArray['id']=++$this->index;
            }
            $professionalExperience = new
ProfessionalExperience($paramArray);
            array_push($this->dataArray, $professionalExperience);
            return $paramArray['id'];
        }
        public function insertToDatabase($conn, $paramArray) {
            $request = $conn->prepare("INSERT INTO ".$this::$tableName."
VALUES (DEFAULT, ?,?,?,?,?,?)");
            $request->bind_param("ssssss", $userid, $title, $workplace,
$startdate, $enddate, $proofurl);
            $userid = $paramArray['userid'];
            $title = $paramArray['title'];
            $workplace = $paramArray['workplace'];
            $startdate = $paramArray['startdate'];
            $enddate = $paramArray['enddate'];
            $proofurl = $paramArray['proofurl'];
            $request->execute();
            $request->close();
        }

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

        public function updateDatabaseById($conn, $id, $paramArray) {
```

```
$request = $conn->prepare("UPDATE ".$this::$tableName." SET `userid`=?,
`title`=?, `workplace`=?, `startdate`=?, `enddate`=?, `proofurl`=? WHERE
`id`=?");
            $request->bind_param("sssssss", $userid, $title, $workplace,
$startdate, $enddate, $proofurl, $id);
            $userid = $paramArray['userid'];
            $title = $paramArray['title'];
            $workplace = $paramArray['workplace'];
            $startdate = $paramArray['startdate'];
            $enddate = $paramArray['enddate'];
            $proofurl = $paramArray['proofurl'];
            $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 getFromDatabaseByUserID($conn, $userid) {
            $request = $conn->prepare("SELECT `" . $this::$tableName .
"`.* FROM `" . $this::$tableName . "` WHERE userid = ?");
            $request->bind_param("s", $userid);
            $request->execute();
            $result = $request->get_result();

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

        public function getFromDatabaseByUserIDLast5Years($conn, $userid)
{
            $query = "SELECT `" . $this::$tableName . "`.* FROM `" .
$this::$tableName . "`
                        WHERE userid = ? AND startdate >= DATE_SUB(NOW(),
INTERVAL 5 YEAR)";

            $request = $conn->prepare($query);
            $request->bind_param("s", $userid);
            $request->execute();
            $result = $request->get_result();
```

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

          public $summaryData = [];

          public function getTotalExperienceYearsByUserID($conn, $userid) {
              $query = "SELECT
                            SUM(TIMESTAMPDIFF(MONTH,                startdate,
IFNULL(enddate, NOW()))/12 AS total_years

              FROM `" . $this::$tableName . "`
                            WHERE userid = ?";

              $request = $conn->prepare($query);
              $request->bind_param("s", $userid);
              $request->execute();
              $result = $request->get_result();

              if ($result->num_rows > 0) {
                  $row = $result->fetch_assoc();
                  $this->summaryData        =        ["total_years"        =>
round($row["total_years"] ?? 0, 2)];
              } else {
                  $this->summaryData = ["total_years" => 0];
              }
          }


          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"}';  }   }}
```

EducationController.php

```
<?php
require_once ("./app/services/EducationService.php");
require_once ("./app/controllers/BaseController.php");
$educationService = new EducationService();
if (!isset($_REQUEST['action'])) {
    $controller = new BaseController($AMDBConnection, $educationService);
    $controller->processRequest();
} else if ($_REQUEST['action'] == 'getUserByID') {
    $educationService->getFromDatabaseByUserID($AMDBConnection,
$_REQUEST['userid']);
    echo $educationService->getAsJSON();
} else if ($_REQUEST['action'] == 'getUserByIDLast5Years') {
    $educationService-
>getFromDatabaseByUserIDLast5Years($AMDBConnection, $_REQUEST['userid']);
    echo $educationService->getAsJSON();
}
?>
```

ExcellenceCertificateController.php

```
<?php
require_once("./app/services/ExcellenceCertificateService.php");
require_once("./app/controllers/BaseController.php");
$excellenceCertificateService = new ExcellenceCertificateService();
if (!isset($_REQUEST['action'])) {
    $controller = new BaseController($AMDBConnection,
$excellenceCertificateService);
    $controller->processRequest();
} else if ($_REQUEST['action'] == 'getUserByID') {
    $excellenceCertificateService-
>getFromDatabaseByUserID($AMDBConnection, $_REQUEST['userid']);
    echo $excellenceCertificateService->getAsJSON();
} else if ($_REQUEST['action'] == 'getUserByIDLast5Years') {
    $excellenceCertificateService-
>getFromDatabaseByUserIDLast5Years($AMDBConnection, $_REQUEST['userid']);
    echo $excellenceCertificateService->getAsJSON();
} else if ($_REQUEST['action'] == 'getTotalHoursLast5Years') {
    $excellenceCertificateService-
>getTotalHoursLast5YearsByUserID($AMDBConnection, $_REQUEST['userid']);

  echo json_encode($excellenceCertificateService->summaryData,
JSON_UNESCAPED_UNICODE);
}
?>
```

LanguageLevelController.php

```
<?php
require_once ("./app/services/LanguageLevelService.php");
require_once ("./app/controllers/BaseController.php");
$languageLevelService = new LanguageLevelService();
if (!isset($_REQUEST['action'])) {
    $controller = new BaseController($AMDBConnection,
$languageLevelService);
    $controller->processRequest();
} else if ($_REQUEST['action'] == 'getUserByID') {
    $languageLevelService->getFromDatabaseByUserID($AMDBConnection,
$_REQUEST['userid']);
    echo $languageLevelService->getAsJSON();
} else if ($_REQUEST['action'] == 'getUserByIDLast5Years') {
    $languageLevelService-
>getFromDatabaseByUserIDLast5Years($AMDBConnection, $_REQUEST['userid']);
    echo $languageLevelService->getAsJSON();
}
?>
```

ProfessionalExperienceController.php

```
<?php
require_once("./app/services/ProfessionalExperienceService.php");
require_once("./app/controllers/BaseController.php");
$professionalExperienceService = new ProfessionalExperienceService();
if (!isset($_REQUEST['action'])) {
    $controller = new BaseController($AMDBConnection,
$professionalExperienceService);
```

```
$controller->processRequest();
} else if ($_REQUEST['action'] == 'getUserByID') {
    $professionalExperienceService-
>getFromDatabaseByUserID($AMDBConnection, $_REQUEST['userid']);
    echo $professionalExperienceService->getAsJSON();
} else if ($_REQUEST['action'] == 'getUserByIDLast5Years') {
    $professionalExperienceService-
>getFromDatabaseByUserIDLast5Years($AMDBConnection, $_REQUEST['userid']);
    echo $professionalExperienceService->getAsJSON();
} else if ($_REQUEST['action'] == 'getTotalExperienceYears') {
    $professionalExperienceService-
>getTotalExperienceYearsByUserID($AMDBConnection, $_REQUEST['userid']);
    echo         json_encode($professionalExperienceService->summaryData,
JSON_UNESCAPED_UNICODE);
}
?>
```

config.php

```
<?php
    $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";
```

DBProvider.php

```
<?php
$AMDBConnection = new mysqli($AMServerName, $AMUserName, $AMPassword,
$AMDatabase);
if ($AMDBConnection->connect_error) {
  die("Connection failed: " . $AMDBConnection->connect_error);}?>
```

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,$AM
```

```
ProjectBaseUrl,$AMAuthCallbackURL);
        }else if ($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,$AMGoogleC
lientID,$AMGoogleSecret,$AMServerURL,$AMProjectBaseUrl,$AMAuthCallbackUR
L,$_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.'/room-types' :
                        require_once
("./app/controllers/RoomTypeController.php");
                        break;
                    case $AMProjectBaseUrl.'/building-types' :
                        require_once
("./app/controllers/BuildingTypeController.php");
                        break;
                    case $AMProjectBaseUrl.'/pc-storage-types' :
                        require_once
("./app/controllers/PCStorageTypeController.php");
                        break;
                    case $AMProjectBaseUrl.'/pc-ram-types' :
                        require_once
("./app/controllers/PCRAMTypeController.php");
```

```
                        break;
                    case $AMProjectBaseUrl.'/buildings' :
                    require_once
("./app/controllers/BuildingController.php");
                        break;
                    case $AMProjectBaseUrl.'/rooms' :
                        require_once
("./app/controllers/RoomController.php");
                        break;
                    case $AMProjectBaseUrl.'/computers' :
                        require_once
("./app/controllers/ComputerController.php");
                        break;
                    case $AMProjectBaseUrl.'/users' :
                        require_once
("./app/controllers/UserController.php");
                        break;
                    case $AMProjectBaseUrl.'/department-roles' :
                        require_once
("./app/controllers/DeparmentRoleController.php");
                        break;
            case $AMProjectBaseUrl.'/departments' :
                        require_once
("./app/controllers/DepartmentController.php");
                        break;
                    case $AMProjectBaseUrl.'/department-types' :
  require_once ("./app/controllers/DepartmentTypeController.php");
                        break;
                    case $AMProjectBaseUrl.'/employment-types' :
                        require_once
("./app/controllers/EmploymentTypeController.php");
                        break;
                    case $AMProjectBaseUrl.'/learning-years' :
                        require_once
("./app/controllers/LearningYearController.php");
                        break;
                    case $AMProjectBaseUrl.'/plan-activities' :
                        require_once
("./app/controllers/PlanActivityController.php");
                        break;
                    case $AMProjectBaseUrl.'/plan-activity-types' :
     require_once ("./app/controllers/PlanActivityTypeController.php");
                        break;
                    case $AMProjectBaseUrl.'/user-department-roles' :
                        require_once
("./app/controllers/UserDepartmentRoleController.php");
                        break;
                    case $AMProjectBaseUrl.'/user-plan-approvals' :
                        require_once
("./app/controllers/UserPlanApprovalController.php");
                        break;
                    case $AMProjectBaseUrl.'/user-plans' :
                        require_once
("./app/controllers/UserPlanController.php");
                        break;
```

```
                    case $AMProjectBaseUrl.'/user-uploads' :
                        require_once
("./app/controllers/UserUploadController.php");
                        break;
// Routes create Rostyslav Kopko
                    case $AMProjectBaseUrl.'/excellence-certificates' :
                        require_once
("./app/controllers/ExcellenceCertificateController.php");
                        break;
                    case $AMProjectBaseUrl.'/professional-experiences' :
                        require_once
("./app/controllers/ProfessionalExperienceController.php");
                        break;
                    case $AMProjectBaseUrl.'/educations' :
                        require_once
("./app/controllers/EducationController.php");
                        break;
                    case $AMProjectBaseUrl.'/language-levels' :
                        require_once
("./app/controllers/LanguageLevelController.php");
                        break;
                    default:
                        http_response_code(404);
                        require_once ("./assets/404.php");   break;}}}}
?>
```

Create.sql

```
CREATE TABLE `am_excellence_certificate` (
  `id` int NOT NULL,
  `userid` int NOT NULL,
  `name` text NOT NULL,
  `issuedate` datetime NOT NULL,
  `hours` float NOT NULL,
  `issuername` text NOT NULL,
  `certificateurl` text NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

CREATE TABLE `am_professional_experience` (
  `id` int NOT NULL,
  `userid` int NOT NULL,
  `title` text NOT NULL,
  `workplace` text NOT NULL,
  `startdate` datetime NOT NULL,
  `enddate` datetime NOT NULL,
  `proofurl` text NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
CREATE TABLE `am_education` (
  `id` int NOT NULL,
  `userid` int NOT NULL,
  `title` text NOT NULL,
  `speciality` text NOT NULL,
  `diplomanumber` text NOT NULL,
  `issuedate` datetime NOT NULL,
```

```
`proofurl` text NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
CREATE TABLE `am_language_level` (
  `id` int NOT NULL,
  `userid` int NOT NULL,
  `language` text NOT NULL,
  `certificateissuer` text NOT NULL,
  `level` text NOT NULL,
  `issuedate` datetime NOT NULL,
  `proofurl` text NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

ALTER TABLE `am_excellence_certificate`
ADD CONSTRAINT `fk_excellence_certificate_user`
FOREIGN KEY (`userid`) REFERENCES `am_users`(`id`);
ALTER TABLE `am_professional_experience`
ADD CONSTRAINT `fk_professional_experience_user`
FOREIGN KEY (`userid`) REFERENCES `am_users`(`id`);
ALTER TABLE `am_education`
ADD CONSTRAINT `fk_education_user`
FOREIGN KEY (`userid`) REFERENCES `am_users`(`id`);
ALTER TABLE `am_language_level`
ADD CONSTRAINT `fk_language_level_user`
FOREIGN KEY (`userid`) REFERENCES `am_users`(`id`);

INSERT INTO `am_language_level` (`id`, `userid`, `language`,

`certificateissuer`, `level`, `issuedate`, `proofurl`) VALUES
(1,    101,    'English',    'Issuer    A',    'C1',    '2023-01-01',
'http://example.com/proof1'),
(2,    102,    'French',    'Issuer    B',    'B2',    '2022-01-01',
'http://example.com/proof2'),
(3,    103,    'German',    'Issuer    C',    'B1',    '2021-01-01',
'http://example.com/proof3');

INSERT INTO `am_education` (`id`, `userid`, `title`, `speciality`,
`diplomanumber`, `issuedate`, `proofurl`) VALUES
(1, 101, 'Degree 1', 'Speciality A', 'Diploma 1', '2022-01-01',
'http://example.com/proof1'),
(2, 102, 'Degree 2', 'Speciality B', 'Diploma 2', '2021-01-01',
'http://example.com/proof2'),
(3, 103, 'Degree 3', 'Speciality C', 'Diploma 3', '2020-01-01',
'http://example.com/proof3');
INSERT INTO `am_professional_experience` (`id`, `userid`, `title`,
`workplace`, `startdate`, `enddate`, `proofurl`) VALUES
(1, 101, 'Job Title 1', 'Workplace A', '2021-01-01', '2022-01-01',
'http://example.com/proof1'),
(2, 102, 'Job Title 2', 'Workplace B', '2020-01-01', '2021-01-01',
'http://example.com/proof2'),
(3, 103, 'Job Title 3', 'Workplace C', '2019-01-01', '2020-01-01',
'http://example.com/proof3');
INSERT INTO `am_excellence_certificate` (`id`, `userid`, `name`,
`issuedate`, `hours`, `issuername`, `certificateurl`) VALUES
(1,   101,   'Certificate   1',   '2023-01-01',   5.5,   'Issuer   A',
'http://example.com/certificate1'),
```

```
(2,   102,   'Certificate   2',   '2023-01-02',   6.0,   'Issuer   B',
'http://example.com/certificate2'),
(3,   103,   'Certificate   3',   '2023-01-03',   4.5,   'Issuer   C',
'http://example.com/certificate3');
ALTER TABLE `am_professional_experience`
MODIFY COLUMN `enddate` datetime NULL;

ALTER TABLE `am_excellence_certificate`
MODIFY COLUMN `hours` float NOT NULL DEFAULT 0;

ALTER TABLE `am_excellence_certificate`
  MODIFY `id` int NOT NULL AUTO_INCREMENT,
  ADD PRIMARY KEY (`id`);

ALTER TABLE `am_professional_experience`
  MODIFY `id` int NOT NULL AUTO_INCREMENT,
  ADD PRIMARY KEY (`id`);

ALTER TABLE `am_education`
  MODIFY `id` int NOT NULL AUTO_INCREMENT,
  ADD PRIMARY KEY (`id`);

ALTER TABLE `am_language_level`
  MODIFY `id` int NOT NULL AUTO_INCREMENT,
  ADD PRIMARY KEY (`id`);
```

БІБЛІОГРАФІЧНА ДОВІДКА

Тема дипломної роботи: **Розробка серверної частини компоненти “Професійні досягнення працівників” інформаційної системи ІФНТУНГ**

Обсяг пояснювальної записки 53 аркуші.

- 4 таблиці;
- 19 рисунків;
- 1 додаток.

Дата завершення роботи: 11 червня 2025 р.

Підпис студента дипломника _____ Копко Р.М.