ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені І.І.МЕЧНИКОВА
(повне найменування вищого навчального закладу)

Факультет математики, фізики та інформаційних технологій
(повне найменування інституту, назва факультету (відділення))

Кафедра математичного забезпечення комп'ютерних систем
(повна назва кафедри (предметної, циклової комісії))

Кваліфікаційна робота

на здобуття ступеня вищої освіти «бакалавр»
(освітньо-кваліфікаційний рівень)

Система підтримки науково-методичної діяльності ВНЗ.

The system of support for scientific and methodological activities of universities

Виконав: студент денної форми навчання
спеціальності 126 – Інформаційні системи та технології
(шифр і назва напряму підготовки, спеціальності)
Освітня програма «Інформаційні системи та технології»
(назва освітньої програми)
Сапожніков Віталій Сергійович
(прізвище, ім'я, по-батькові)

Керівник Ст. викл. Трубіна Н.Ф.
(науковий ступінь, вчене звання, прізвище та ініціали, **підпис**)

Рецензент Ст. викл. Лісіцина І.М.
(науковий ступінь, вчене звання, прізвище та ініціали)

Рекомендовано до захисту:
Протокол засідання кафедри
№___ від «___»_____ 2024 р.

Завідувач кафедри
_____ Євгеній МАЛАХОВ
(підпис) (ім'я, прізвище)

Захищено на засіданні ЕК № ___
протокол № __ від «___» ____ 2024 р.
Оцінка _____/_____/_____
(за національною шкалою, шкалою ECTS, бали)

Голова ЕК
_____ Володимир ВИЧУЖАНІН
(підпис) (ім'я, прізвище)

Одеса - 2024

АНОТАЦІЯ

У кваліфікаційній роботі розглянуто процеси, пов’язані з науково-методичною діяльністю закладів вищої освіти, розглянуто особливості предметної області університету в середовищі ОНУ імені І. І. Мечникова, визначено функціональність, котра потребує реалізації, та зібрано вимоги користувачів.

Проведено порівняльний аналіз наявних інформаційних систем, які забезпечують функціональність, необхідну для підтримки науково-методичної діяльності університету. У результаті прийнято рішення розробити власну систему організації процесів у ЗВО.

Спроєктовано схему бази даних. Обрано трирівневу архітектуру системи та паттерн проєктування MVC. Для розробки обрано стек технологій: СУБД Postgres, платформа ASP .NET Core — для серверної частини та HTML, CSS, JavaScript, бібліотека Bootstrap — для клієнтського інтерфейсу. Взаємодія користувачів із системою реалізована із дотриманням правил безпеки на всіх рівнях додатку.

Інформаційна система реалізована із функціональністю оцінки ефективності роботи викладачів, контролю відповідності публікацій викладача його дисципліні, планування видань кафедрою, контролю наповненості навчальної дисципліни матеріалами, перевірки виконання викладачем умов ліцензування, що стосуються наукових та методичних публікацій, а також підбором матеріалів, що належать певній дисципліні.

ABSTRACT

The qualification work considers the processes associated with the scientific and methodological activities of higher education institutions, examines the features of the university's subject area in the environment of the I. I. Mechnikov ONU, identifies the functionality that needs to be implemented, and collects user requirements.

A comparative analysis of existing information systems that provide the functionality necessary to support the university's scientific and methodological activities is carried out. As a result, a decision was made to develop an in-house system for organising processes in the university.

A database scheme was designed. A three-tier system architecture and MVC design pattern were chosen. A technology stack was selected for development: Postgres DBMS, ASP .NET Core platform for the server side and HTML, CSS, JavaScript, Bootstrap library for the client interface. User interaction with the system is implemented in compliance with security rules at all levels of the application.

The information system is implemented with the functionality of evaluating the effectiveness of lecturers, monitoring the relevance of the lecturer’s publications to his or her discipline, planning publications by the department, monitoring the completeness of the discipline with materials, checking whether the lecturer has fulfilled the licensing conditions related to scientific and methodological publications, and selecting materials related to a particular discipline.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

ЗВО – заклад вищої освіти;

ПЗ – програмне забезпечення;

MVC – Model View Controller (паттерн проєктування);

ORM – Object-Relational Mapping;

БД – база даних;

CRUD – Create, Read, Update, Delete;

LINQ – Language Integrated Query;

DTO – Data Transfer Object;

ВСТУП

Протягом останніх років тенденція цифровізації та автоматизації стала світовою. Порівнюючи автоматизовані та неавтоматизовані підприємства, можна побачити велику перевагу у продуктивності та ефективності перших завдяки зменшенню впливу людського фактору на виробництво [1].

До широкого впровадження інформаційних систем в навчальній галузі контроль та аудит часто здійснювався із використанням паперової документації. Здебільшого викладачі та працівники навчальних відділів самостійно заповнюють необхідні документи, що неодмінно призводить до виникнення помилок. У найгіршому випадку ці помилки впливають на інші документи, що створює ефект каскаду. Пошук та виправлення цих помилок потребує багато часу.

Використання інформаційної системи може сприяти прискоренню пошуку, узагальненню даних з різних джерел і зменшенню ймовірності виникнення помилок під час заповнення і подання інформації у різних формах.

Нині на ринку налічується певна кількість програмних продуктів, що дозволяють автоматизувати деякі процеси університету. До найбільш поширених можна віднести наступні: АСУ навчальним процесом «Директива», АСУ «Університет» (ТОВ «UNITEX+»), Адміністративно навчальна ІС Національного авіаційного університету України, Пакет комп’ютерних систем ПП «Політек-софт», Програмний комплекс «АЛЬМА-МАТЕР» та інші [2-7].

Натомість практика показує, що хоча ці системи і надають широкий спектр можливостей, проте мають ряд недоліків. Зокрема — відсутність персоналізації, надмірна універсальність та орієнтованість на досвідченого користувача [8].

Оскільки існуючі рішення не враховують особливостей та специфіки керування освітнім процесом обраного закладу вищої освіти, виникає необхідність у власній розробці. Таким чином, актуальність створення власної ІС підтримки науково-методичної діяльності ЗВО полягає в можливості запропонувати систему, котра відповідає правилам організації процесів в університеті та в повній мірі забезпечує виконання завдань обраного структурного підрозділу. Така система має бути достатньо простою та зрозумілою для користувача, з можливістю легкої навігації та зручного представлення інформації.

Метою цього проекту є скорочення кількості помилок та часу, що витрачається на аудит і контроль діяльності кафедри шляхом створення інформаційної системи підтримки науково-методичної діяльності у вищому навчальному закладі. Створювана система надає можливості:

– завідувачу кафедри проаналізувати ефективність роботи конкретного викладача, діяльність кафедри, планувати видання на календарний рік, перевіряти виконання викладачем певних пунктів ліцензійних вимог;

– гаранту освітньої програми перевірити відповідність публікацій викладача дисципліні, яку він викладає, переглянути наповненість навчальної дисципліни методичними матеріалами;

– навчальному відділу відслідковувати виконання планів видань усіх кафедр;

– викладачу переглянути заплановані публікації, додати нові наукові чи методичні публікації;

– студенту користуватись онлайн-бібліотекою наукових та методичних видань із можливістю підбору матеріалів по певній дисципліні.

Основні задачі, які необхідно вирішити для досягнення мети проекту:

1) аналіз предметної області та процесів, котрі стосуються науково-методичної діяльності кафедри;

2) визначення груп користувачів та формулювання їхніх вимог до створюваної інформаційної системи;

3) розробка архітектури та вибір шаблону проектування системи;

4) розробка бази даних – необхідно врахувати всі вимоги до даних, які зберігатимуться у системі, а також забезпечити можливість ефективного доступу до цих даних;

5) забезпечення цілісності та безпеки даних як на рівні БД, так і на рівні застосунку;

6) тестування ІС.

1 ТЕОРЕТИЧНІ ВІДОМОСТІ

1.1 Проблематика ведення документації в освітній галузі

В умовах посилення конкуренції між закладами вищої освіти в Україні, з’являється важлива потреба в радикальному оновленні їх управлінських та організаційних систем, що має на меті оптимізацію та збільшення продуктивності.

На сьогоднішній день, не зважаючи на значний інтелектуальний потенціал, інформатизація закладів вищої освіти залишається на низькому рівні, переважно обмежуючись застосуванням ІТ лише в бухгалтерському обліку та кадровому забезпеченні. Практика показує, що автоматизація основної діяльності залишається недостатньою [9].

Одеський національний університет імені І. І. Мечникова не є винятком, та на даний момент більшість операцій в університеті виконується вручну, починаючи від навчального відділу та закінчуючи діяльністю кафедри. Велика кількість документів обробляються співробітниками закладу освіти без допомоги автоматизованих систем. Такий підхід призводить до зростання витрат часу на ведення документації, появи помилок при обробці даних та їх перенесенні. Крім того, такий підхід робить неможливим швидкий доступ до актуальної інформації. Пошук та виправлення помилок у традиційній паперовій системі зазвичай віднімає багато часу, а також існує ризик виникнення нових помилок у процесі виправлення.

Ця ситуація вимагає негайного впровадження сучасних інформаційних систем, які б забезпечили автоматизацію та інтеграцію різноманітних процесів управління. Впровадження такої системи дозволить:

- зменшити час на обробку документів;
- знизити кількість помилок;
- забезпечити оперативний доступ до необхідної інформації;

– сприятиме підвищенню прозорості управлінських процесів;

– забезпечить механізми валідації даних, відповідно до нормативних правил та вимог предметної області;

– забезпечити безпеку даних та їх захист від несанкціонованого доступу.

Впровадження таких систем дозволить викладачам та адміністративному персоналу зосередитися на ключових аспектах своєї роботи, зменшивши час, витрачений на бюрократичні процедури. Це, в свою чергу, сприятиме підвищенню якості освіти та дослідницької діяльності, а також забезпечить більш ефективне використання ресурсів університету.

1.2 Опис предметної області

Щорічно заклади вищої освіти видають публікації, котрі умовно можна поділити на наукові та методичні.

Наукові публікації включають такі типи:

– наукова стаття;

– монографія;

– дисертація;

– патент;

– тези наукових конференцій.

Наукові статті класифікуються на журнали категорії A (Scopus, Web Of Science) та журнали категорії B (інші українські та іноземні журнали) [10]. Дисертації, у свою чергу, поділяються на магістерські, кандидатські та докторські. Кожна наукова публікація має такі властивості, як DOI (Digital Object Identifier), УДК (Універсальна десяткова класифікація) та сформоване посилання (URL).

Методичні публікації за класифікацією поділяються на такі види:

– підручники;

- навчальні посібники;
- курси лекцій;
- навчально-методичні посібники;
- практикуми;
- методичні посібники;
- методичні рекомендації;
- хрестоматії;
- словники;
- довідники.

Кожний календарний рік на засіданні кафедри затверджується план видань. Протягом року кафедра зобов'язана видати певну кількість методичних матеріалів, а наприкінці року складається звітність щодо запланованих та фактично виданих матеріалів. Завідувач кафедри несе відповідальність за виконання плану та звітність. Також дані стосовно планів видань та їх виконання збираються навчальним відділом для подальшого оформлення звітностей. Цей відділ відповідає за централізоване обліковування запланованих та фактично виданих матеріалів, що дозволяє забезпечити узгодженість і своєчасність звітності по всьому закладу вищої освіти.

Обидва типи публікацій – наукові та методичні – мають співвідноситись з дисциплінами, котрі викладаються на кафедрі. Також публікації можуть бути написані у співавторстві, що є звичайною практикою в науковій діяльності.

На цій основі у завідувача кафедри також виникають завдання, що стосуються звітності про роботу кафедри, зокрема звітність про методичну роботу, яка включає кореляцію між запланованими публікаціями та фактично виданими. Крім того, завідувач відслідковує наповненість певної дисципліни методичними матеріалами та звітує про роботу конкретного викладача протягом останнього року або іншого періоду.

Науково-педагогічні, педагогічні та наукові працівники, які забезпечують освітній процес, повинні мати не менше чотирьох досягнень у професійній діяльності за останні п’ять років, визначених у пункті 38 Ліцензійних умов впровадження освітньої діяльності [11].

Згідно з пунктом 38, досягнення у професійній діяльності зараховуються за останні п’ять років і включають:

1) наявність не менше п’яти публікацій у періодичних наукових виданнях, що включені до переліку фахових видань України або до наукометричних баз, зокрема Scopus та Web of Science Core Collection;

2) наявність одного патенту на винахід або п’яти деклараційних патентів на винахід чи корисну модель, включаючи секретні, або наявність не менше п’яти свідоцтв про реєстрацію авторського права на твір;

3) наявність виданого підручника чи навчального посібника (включаючи електронні) або монографії (загальним обсягом не менше 5 авторських аркушів), в тому числі виданих у співавторстві (обсягом не менше 1,5 авторського аркуша на кожного співавтора);

4) наявність виданих навчально-методичних посібників/посібників для самостійної роботи здобувачів вищої освіти та дистанційного навчання, електронних курсів на освітніх платформах ліцензіатів, конспектів лекцій/практикумів/методичних вказівок/рекомендацій/робочих програм, інших друкованих навчально-методичних праць загальною кількістю три найменування;

5) захист дисертації на здобуття наукового ступеня;

6) наявність апробаційних та/або науково-популярних, та/або консультаційних (дорадчих), та/або науково-експертних публікацій з наукової або професійної тематики загальною кількістю не менше п’яти публікацій.

Діяльність кафедри також включає в себе формування робочих програм, у яких фактично пов'язуються дисципліни із спеціальностями. В свою чергу, за кожною спеціальністю та рівнем підготовки (бакалавр, магістр,

аспірант) закріплюється єдиний Гарант освітньої програми. Гарант освітньої програми відслідковує наповненість дисципліни методичними матеріалами, а також перевіряє відповідність публікацій викладача дисципліні, яку той викладає.

1.3 Огляд аналогів

У сучасних університетах та інших закладах вищої освіти впроваджено різноманітні інформаційні системи для автоматизації адміністративних процесів. Ці системи надають засоби для збору, організації та обробки інформації, необхідної для ефективного функціонування університетської спільноти. У зв'язку з цим проведено огляд аналогів, що дозволяють порівняти створювану систему підтримки навчальної та методичної діяльності з іншими аналогічними рішеннями.

Наразі на ринку існує ряд інформаційних систем, призначених для управління навчальним процесом та підтримки академічних програм університетів. Серед них особливий інтерес представляють системи, які надають такі можливості:

– створення електронної бібліотеки з можливістю доступу до наукових публікацій, навчальних посібників, документів та іншої наукової літератури;

– розробка планів видань кафедрою, включаючи визначення переліку видань, обсягу, авторів та іншої необхідної інформації;

– виконання плану видань викладачами шляхом створення, редагування та публікації наукових праць, методичних матеріалів тощо;

– контролю наповненості дисциплін відповідними методичними матеріалами;

– ліцензування викладачів відповідно до норм законодавства, включаючи управління документами, що засвідчують кваліфікацію та право на викладання певних предметів.

Варто відзначити, що багато іноземних інформаційних систем надають доступ до документації та функціоналу лише для довірених користувачів, що негативно впливає на їх доступність для аналізу та порівняння. Крім того, відмінності у системах керування навчальним процесом у різних країнах можуть призвести до необхідності адаптації зазначених систем до місцевих умов.

Наведені нижче приклади найбільш розповсюджених інформаційних систем відображають спектр можливостей, які реалізовані в сучасних системах управління навчальним процесом та підтримки академічних програм.

1.3.1 Автоматизована система управління вищим навчальним закладом III - IV рівня акредитації – Uniteh+

ТОВ «Юнітех+» – розробник великих корпоративних інформаційних систем на українському ринку з 1992 року. Відповідно до поставленої задачі, їх продукт «Автоматизована система управління вищим навчальним закладом III - IV рівня акредитації» викликає зацікавленість.

Модуль «Навчальна частина – 01» АСУ «Університет» реалізує такі функції:

– облік та закріплення навчальних дисциплін за кафедрами;

– формування електронних навчальних план-графіків;

– формування бази даних про кількісний і якісний професорсько-викладацький склад кафедр;

– автоматизація аналізу та конвертації звітної інформації у файлові формати *.rtf, *.xls, *.pdf, *.htm по всім видам розрахунків з метою їх розповсюдження та подальшого використання за межами АСУ;

– автоматизація аналізу та формування звітної інформації для друку;

А також модуль «Кафедра – 01» АСУ «Університет» з функціями:

– облік та аналіз навчально-методичного забезпечення навчальних дисциплін та спеціальностей;

– формування тематичних планів з навчальних дисциплін кафедри;

– автоматизація аналізу навчально-методичної роботи кафедри

Отже, за переліком функцій ця система може задовольняти вимоги ОНУ імені І. І. Мечникова. Але потрібно зазначити, що з першого модуля, фактично, використовуватимуться лише 5 з шістнадцяти наявних, а з другого – 3 функції з десяти наявних. Хоча всі 26 функцій з двох модулів і представляють інтерес для ЗВО, проте у контексті даної роботи більша їх частина надлишкова.

Окрім цього, на сайті продукту вказано, що деякі клієнтські складові розроблено для платформи MS Windows XP що на сьогоднішній день застаріло та може викликати проблеми із зворотною сумісністю. Окрім цього, потрібно буде модифікувати систему, щоб та відповідала структурним стандартам та бізнес правилам ОНУ імені. І. І. Мечникова.

1.3.2 Адміністративно навчальна ІС інституту комп’ютерних технологій Національного авіаційного університету України

Адміністративно-навчальна інформаційна система Національного авіаційного університету України, розроблена для Інституту комп'ютерних технологій, структурована на два головні розділи: Адміністративний та Навчально-методичний. Ця система забезпечує можливість перегляду кадрової інформації щодо викладачів, студентів та аспірантів, формування особистих карток співробітників, аспірантів та докторантів, а також фільтрацію та пошук за обраними критеріями кадрового складу. Крім того, система забезпечує можливість формування та друку різноманітних звітів стосовно адміністративної інформації про інститут.

Також система включає в себе можливість побудови навчальних та робочих планів за спеціальностями та напрямками підготовки, а також генерацію окремим файлом отриманих навчальних і робочих планів. Забезпечується також збереження навчального та навчально-методичного матеріалу для студентів, необхідного для виконання атестаційних робіт та контролю знань та умінь. Крім того, система дозволяє переглядати наукові та навчально-методичні видання співробітників інституту, а також здійснювати пошук та фільтрацію серед збереженого навчального та наукового матеріалу.

Натомість існують деякі недоліки в адміністративно-навчальній інформаційній системі Інституту комп'ютерних технологій. По-перше, система не надає можливості створення електронної бібліотеки з доступом до наукових публікацій, навчальних посібників та інших наукових матеріалів, що обмежує доступ до необхідної літератури для студентів. Крім того, немає засобів для розробки планів видань кафедрами та контролю наповненості дисциплін методичними матеріалами, що необхідно для організації навчального процесу. Також система не надає можливості ліцензування викладачів відповідно до норм законодавства.

Отже, розглянута інформаційна система не відповідає висунутим вимогам з точки зору функціональності.

1.3.3 АСУ ВНЗ – Науково-Дослідний Інститут Прикладних Інформаційних технологій

Програмне рішення від Науково-Дослідного Інституту Прикладних Інформаційних технологій пропонує декілька модулів: АС «Приймальна комісія», АС «Деканат» та АС «Студмістечко». Із перелічених, найбільший інтерес викликає АС «Деканат» – програмно-технологічний комплекс управління навчальним процесом закладу освіти, призначений для організації

роботи методистів та зменшення кількості документації на паперових носіях. Із ключових функцій:

– розробка навчальних, робочих планів на навчальний рік, закріплення навчальних груп за планами;

– генерація та відновлення робочих навчальних планів за навчальними планами;

– індивідуальний робочий план викладача кафедри;

– створення розкладу, веб-розкладу;

– робота зі співробітниками, робота з студентами.

Виробник запевняє, що даний продукт відповідає вимогам ISO, рекомендований МОН України та користується попитом вже в більш ніж 70 ВНЗ. У цілому, дане рішення задовольняє необхідні вимоги, натомість існує певний ряд недоліків.

По-перше, вартість користування системою «Деканат» складає 36 тис. гривень, а разом із Веб-кабінетом Викладача та Студента загальна вартість становить 96 тис. гривень (станом на 08.02.2024 р.). По-друге, виробник вимагає щорічну оплату за підтримку функціонування системи. Також в даній системі відсутня підсистема автоматизації діяльності навчального відділу.

Отже, можна зробити висновок, що дана система, хоч і в певній мірі задовольняє висунуті вимоги, але не є достатньо гнучкою, а її купівля не є раціональною.

1.3.4 Програмний комплекс «АЛЬМА-МАТЕР» - ТОВ «Direct IT»

Програмний комплекс «АЛЬМА-МАТЕР» є ще одним рішенням у сфері автоматизації навчального процесу. Однією з головних переваг програмного комплексу є його модульність, що дозволяє придбати лише необхідні частини програмного забезпечення, а не весь комплекс, та адаптувати його під індивідуальні потреби закладу вищої освіти. Крім того, важливо відзначити,

що програмний комплекс розроблено з урахуванням вимог Болонського процесу, що є ключовим аспектом для сучасних університетів.

Програмний комплекс «Альма-Матер» складається з таких модулів: «Приймальна комісія», «Деканат/навчальна частина», «Навчально-методичний відділ» (з можливістю формування розкладу).

Розглянемо функціональність найбільш цікавого, відповідно до вимог поставленої задачі, модулю – «Навчально-методичний відділ»:

– облік викладацького складу;
– формування навчальних і робочих планів;
– автоматизація створення звітів для аналітичного відділу.

Однак, не зважаючи на переваги, варто зазначити основний недолік програмного комплексу «АЛЬМА-МАТЕР» - довгий час на впровадження системи та потребу у постійній матеріальній підтримці вже впровадженої системи. Це може стати вагомим фактором у виборі програмного забезпечення для університету, особливо з урахуванням обмеженого часу та ресурсів, що має бути враховано при прийнятті рішення щодо вибору та впровадження подібної інформаційної системи.

1.4 Результати огляду

Проведений аналіз існуючих інформаційних систем, призначених для управління навчальним процесом та підтримки академічних програм університетів, виявив ряд суттєвих недоліків. Системи, такі як Uniteh+, Адміністративно-навчальна ІС Інституту прикладних технологій, АСУ ВНЗ, та «АЛЬМА-МАТЕР», хоча й пропонують різноманітні функції, не здатні повноцінно задовольнити специфічні вимоги ОНУ імені І. І. Мечникова.

Недоліки розглянутих систем:

– обмежена функціональність — жодна з розглянутих систем не пропонує повного набору функцій, необхідних для комплексної підтримки

науково-методичної роботи, включаючи створення електронної бібліотеки, планування видань, контроль наповнення дисциплін методичними матеріалами, ліцензування викладачів тощо;

– висока вартість — деякі системи, хоча й мають певний набір необхідних функцій, потребують значних фінансових витрат на придбання та обслуговування;

– застарілість технологій — деякі системи розроблені на застарілих платформах, що може призвести до проблем із сумісністю та безпекою;

– необхідність у значних доробках — для адаптації існуючих систем до потреб ОНУ імені І. І. Мечникова необхідно внести значні доробки, що потребуватиме додаткових ресурсів та часу.

Враховуючи ці обмеження, стає очевидною необхідність розробки власної інформаційної системи підтримки науково-методичної діяльності, яка б відповідала всім вимогам та специфікаціям ОНУ імені І. І. Мечникова. Така система має бути гнучкою, масштабованою та інтегрованою з існуючою IT-інфраструктурою університету, а також забезпечувати високий рівень безпеки та зручності користування для всіх зацікавлених сторін.

1.5 Вимоги до системи

Розроблювана ІС підтримки науково-методичної діяльності університету покликана допомагати співробітникам ОНУ та, відповідно до висунутих вимог, передбачає такі ролі та задачі:

а) завідувач кафедри:

1) аналіз ефективності роботи конкретного викладача;

2) аналіз ефективності роботи кафедри;

3) створення плану видань на календарний рік та відслідкування прогресу його виконання;

4) перевірка виконання викладачем пунктів ліцензійних вимог що стосуються наукових та методичних публікацій;

б) гарант освітньої програми:

1) перевірка відповідності публікацій викладача дисципліні, яку він викладає;

2) перевірка наповненості навчальної дисципліни методичними матеріалами;

3) перевірка виконання викладачем пунктів ліцензійних вимог що стосуються наукових та методичних публікацій;

в) навчальний відділ:

1) відслідкування виконання планів видань усіх кафедр;

г) викладач:

1) перегляд запланованих публікацій;

2) виконання плану видань;

3) додавання нових наукових чи методичних публікацій;

д) студент:

1) функціональність онлайн-бібліотеки;

2) можливість підбору матеріалів по певній дисципліні;

3) пошукові запити за наявними матеріалами;

Повний перелік користувачів із їх задачами наведено у Додатку А.

Як було виявлено під час аналізу існуючих рішень на ринку, загальнодоступні інформаційні системи часто не враховують унікальні аспекти та потреби конкретного навчального закладу, пропонуючи занадто універсальний набір функцій, що не відповідає внутрішній структурі та бізнес-логіці установи. Водночас, вартість їх впровадження та обслуговування є значною. У контрасті до цього, індивідуально розроблені системи можуть точно відповідати вимогам навчального закладу, інтегруючись з його структурою та бізнес-процесами, а також забезпечити економію коштів на зовнішні послуги. Саме тому ініційовано проектування

та розробку інформаційної системи для підтримки науково-методичної діяльності в ОНУ імені І. І. Мечникова.

Для створення інформаційної системи, яка відповідатиме всім висунутим вимогам, необхідно розробити базу даних, яка зберігатиме всі необхідні сутності предметної області. Це включає розробку функцій та процедур для вирішення типових задач користувачів, а також програмні засоби для візуалізації цих даних. З метою забезпечення зручності та доступності, для ІС буде розроблено Web API та клієнтський Web-застосунок. Ці компоненти дозволять користувачам взаємодіяти з системою через інтернет, забезпечуючи ефективну роботу з даними у будь-який час і з будь-якого місця.

Оскільки дана ІС може бути частиною більш широкого комплексного продукту, важливо забезпечити можливість взаємодії між різними програмними компонентами комплексу. Це має бути враховано при її проектуванні та реалізації, щоб забезпечити інтеграцію з іншими системами та компонентами, які можуть бути додані в майбутньому.

2 ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

ІС підтримки науково-методичною діяльністю ЗВО розроблено у форматі веб-застосунку, що дозволяє користувачам зайти у систему та використовувати всю її функціональність за допомогою будь-якого пристрою. Це дозволяє уникнути потребу встановлення програмного забезпечення на пристрій користувача, а також усуває необхідність налаштування системи для доступу до необхідних ресурсів.

При проектуванні ІС, вирішено обрати трирівневу архітектуру (див. рис. 2.1)

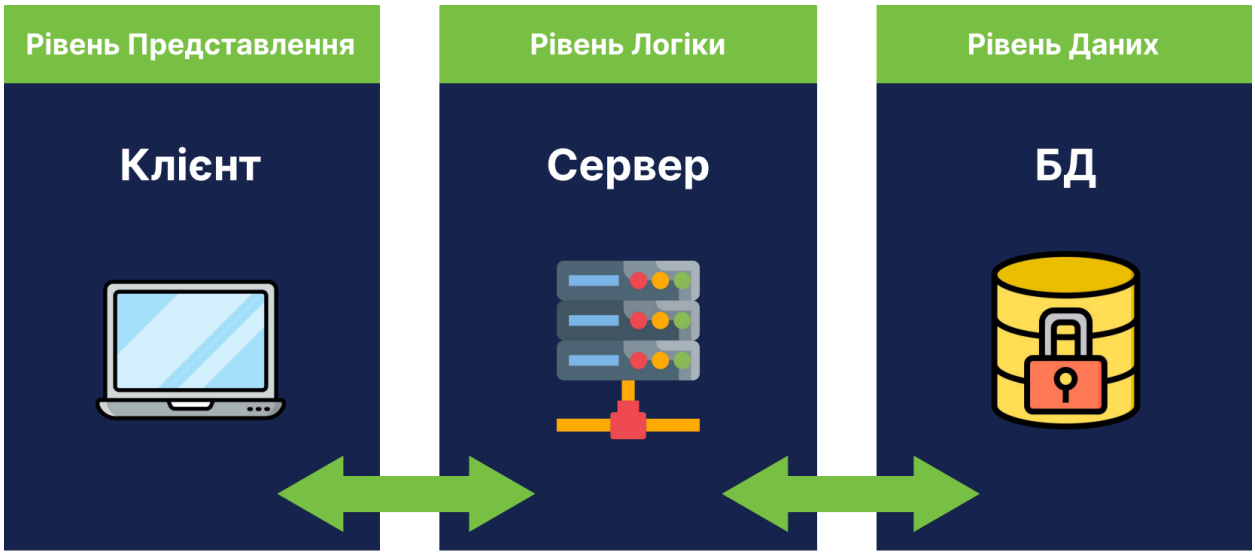


Рисунок 2.1 – Класична схема трирівневої архітектури

Запропонована архітектура визначається трьома рівнями: клієнтським, серверним та рівнем даних, що дозволяє розділити функціональність застосунку на компоненти та забезпечує їхню ефективну взаємодію.

Перший рівень, або клієнтський, насамперед включає інтерфейс користувача. Це область, де користувач безпосередньо взаємодіє з програмою, надаючи можливість введення та виведення даних. Клієнтський рівень може включати різні засоби, такі як веб-браузери та мобільні додатки,

які забезпечують зручний доступ користувачів до системи. Важливо відзначити, що клієнтський рівень не містить логіки, а лише отримує інформацію від сервера.

Другий рівень — серверний, відповідає за бізнес-логіку програмного забезпечення та обробку запитів користувачів. Серверний рівень складається з різноманітних сервісів, які забезпечують стабільну роботу системи та відповідають на запити користувачів.

Третій рівень — рівень даних, відповідає за зберігання та організацію даних. Цей рівень включає системи управління базами даних та набір інструментів для роботи з даними, що забезпечує їх ефективне та безпечне зберігання, а також доступ та обробку.

Однією з обґрунтованих причин застосування трирівневої архітектури є її здатність до розподілу функцій застосунку між різними рівнями. Порівняно дволанковою клієнт-серверною архітектурою, а також товстим клієнтом, трирівнева архітектура забезпечує більшу гнучкість, завдяки ізоляції рівнів один від одного. Це дозволяє зменшити взаємозалежність між компонентами та сприяє більш гнучкому розвитку та розширенню застосунку у майбутньому. Крім того, така архітектура забезпечує вищий рівень безпеки. Реалізація програм, які доступні через веб-браузер або тонкий клієнт, зазвичай базується на трирівневій архітектурі, що дозволяє ефективно розгортати програмний комплекс.

У якості шаблону проектування обрано шаблон MVC (Model-View-Controller), оскільки цей шаблон ефективно розподіляє відповідальності між компонентами застосунку. Він широко використовується для розробки веб-застосунків.

MVC - це архітектурний підхід до створення веб-застосунків, який розбиває логіку застосунку на три основні компоненти: модель, представлення і контролер. Кожен з цих компонентів виконує свою унікальну

функцію і взаємодіє з іншими компонентами для забезпечення коректної роботи застосунку (див. рис. 2.2).

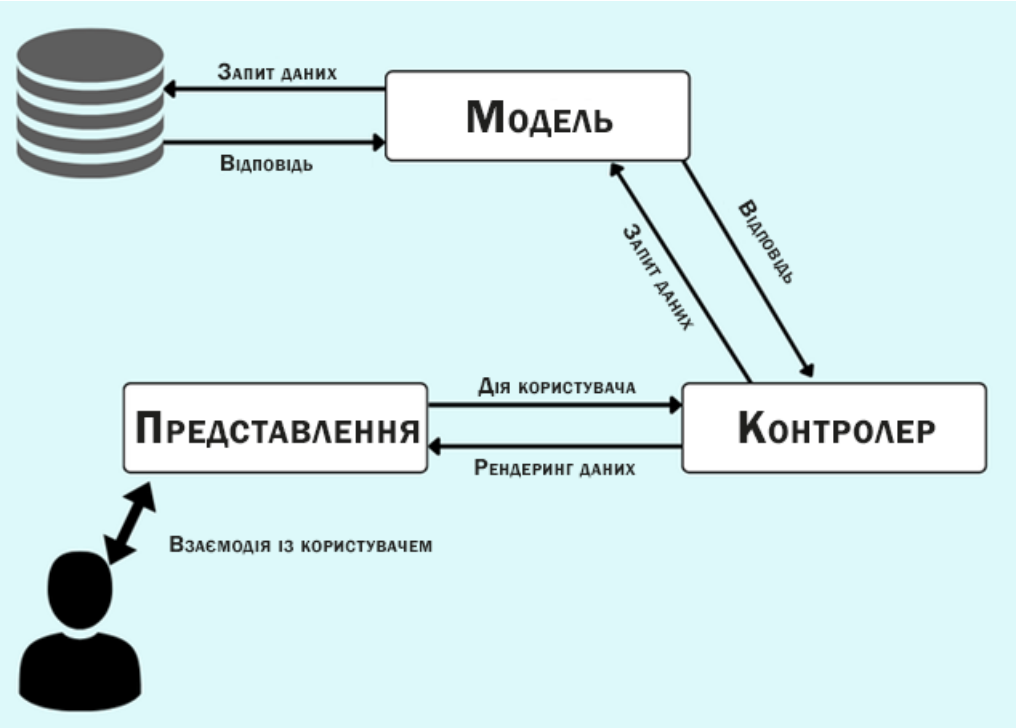


Рисунок 2.2 – Структура Model-View-Controller

Компонент «Модель» відповідає за обробку даних та реалізацію методів для їхньої обробки, включаючи взаємодію з базою даних та перевірку даних на валідність. Важливо відзначити, що модель є незалежною від інших компонентів системи: вона не взаємодіє з представленням (не визначає, як дані будуть візуалізовані) або контролером (не має прямого зв'язку з користувачем). Замість цього, модель просто надає доступ до даних та методів для їх управління, реагуючи на запити та змінюючи свій стан. Через свою незалежність від візуального представлення, модель може мати кілька різних способів відображення.

Компонент «Представлення» відповідає за візуалізацію даних для користувача та отримання від нього вхідних даних. Однак, важливо відзначити, що представлення не обробляє введені користувачем дані – це

завдання контролера. Зазвичай для створення представлення використовуються HTML, CSS та JavaScript, особливо при розробці веб-застосунків.

Компонент «Контролер» відповідає за забезпечення зв'язку між користувачем та системою. Він обробляє запити користувача та взаємодіє з моделлю для отримання необхідних даних. Також контролер взаємодіє з представленням для відображення даних користувачу. Функції контролера включають в себе валідацію введених даних, обробку помилок та вирішення різних проблем, пов'язаних з запитами користувача.

3 ІНФОРМАЦІЙНЕ МОДЕЛЮВАННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

У Додатку Б наведено сутності та їх атрибути, разом із переліком накладених обмежень, які є необхідними для вирішення поставлених завдань.

Зв'язки між сутностями в базі даних ІС підтримки науково-методичної діяльності ЗВО можуть бути класифіковані на такі типи:

1) один-до-одного;

2) один-до-багатьох;

3) багато-до-багатьох;

Розглянемо більш детально кожний зі зв’язків:

1) зв’язок один-до-одного існує, коли кожному екземпляру першого об’єкту стає у відповідність рівно один екземпляр другого об’єкту. Зазвичай, для представлення об’єктів із взаємозв’язком такого типу достатньо одного відношення. Натомість, якщо за умовами завдання або системи необхідно кожну сутність представити у вигляді окремого відношення, то для формалізації зв'язків необхідно атрибут первинного ключа одного з об’єктів додати в схему іншого в якості зовнішнього ключа. Такий зв’язок існує між сутностями:

а) person → lecturer;

б) person → headOfDepartment;

в) department → headOfDepartment;

г) publication → scientificPublication;

ґ) publication → methodologicalPublication;

д) scientificPublication → scientificArticle;

е) scientificPublication → scientificConferenceTheses;

є) scientificPublication → scientificDissertation;

ж) scientificPublication → scientificMonograph;

з) scientificPublication → scientificPatent;

2) зв’язок один-до-багатьох та один-до-багатьох умовний з боку однозв’язної сутності існує, коли (кожний) екземпляр першого об’єкта пов’язаний з багатьма екземплярами другого і кожний екземпляр другого об’єкта пов’язаний тільки з одним екземпляром першого. Для формалізації такого зв’язку первинний ключа сутності, що відповідає одиничному зв’язку, додається у якості зовнішнього ключа до сутності зі сторони множинного зв’язку. Такий зв’язок існує між сутностями:

а) plan (Y) → methodicalPublication;

б) educationYear → workPlan;

в) speciality → workPlan;

г) AspNetUsers → AspNetUserTokens;

г) AspNetUsers → AspNetUserLogins;

д) AspNetUsers → AspNetUserClaims;

3) зв’язок багато-до-багатьох існує, коли кожний екземпляр першого об’єкта пов’язаний з багатьма екземплярами другого, і кожний екземпляр другого об’єкта пов’язаний з багатьма екземплярами першого. Такий зв’язок, незалежно від умовності, вимагає трьох відношень для формалізації: по одному для кожної сутності та одне для зв'язку. Причому останнє повинно мати у схемі відношення первинні ключі двох інших. Такий зв’язок існує між сутностями:

а) lecturer → department, для формалізації зв'язку в якості асоціативної сутності використовується сутність departmentLecturer;

б) lecturer → discipline, для формалізації зв'язку в якості асоціативної сутності використовується сутність lecturerDiscipline;

в) discipline → publication, для формалізації зв'язку в якості асоціативної сутності використовується сутність disciplinePublication;

г) discipline → workPlan, для формалізації зв'язку в якості асоціативної сутності використовується сутність disciplineWorkPlan;

ґ) person → publication, для формалізації зв'язку в якості асоціативної сутності використовується сутність personPublication;

д) AspNetUsers → AspNetRoles, для формалізації зв'язку в якості асоціативної сутності використовується сутність AspNetUserRoles;

Зв'язок між сутностями department та headOfDepartment є прикладом зв'язку один-до-одного. Кожна кафедра має лише одного завідувача кафедри, водночас кожний завідувач кафедри працює лише на одній кафедрі.

Наступний приклад, розглянемо зв’язок між сутностями plan та methodicalPublication. Зрозуміло, що кожний план видань може включати в себе безліч методичних публікацій. Натомість, не кожна методична публікація включена до якогось плану. Отже, маємо зв’язок один-до-багатьох умовний зі сторони однозв’язної сутності.

Також розглянемо зв’язок між сутностями lecturer та discipline. В даному випадку кожний викладач може вести декілька дисциплін, при цьому кожна дисципліна може викладатись кількома викладачами. Маємо класичний зв’язок багато-до-багатьох.

Усі таблиці у базі даних ІС підтримки науково-методичної діяльності ЗВО перебувають у нормальній формі Бойса-Кодда. ER-діаграма, що відображає структуру моделі цієї інформаційної системи, наведена на рисунку 3.1. Для побудови цієї діаграми використаний програмний засіб DBeaver із застосуванням нотації "вороняча лапка".

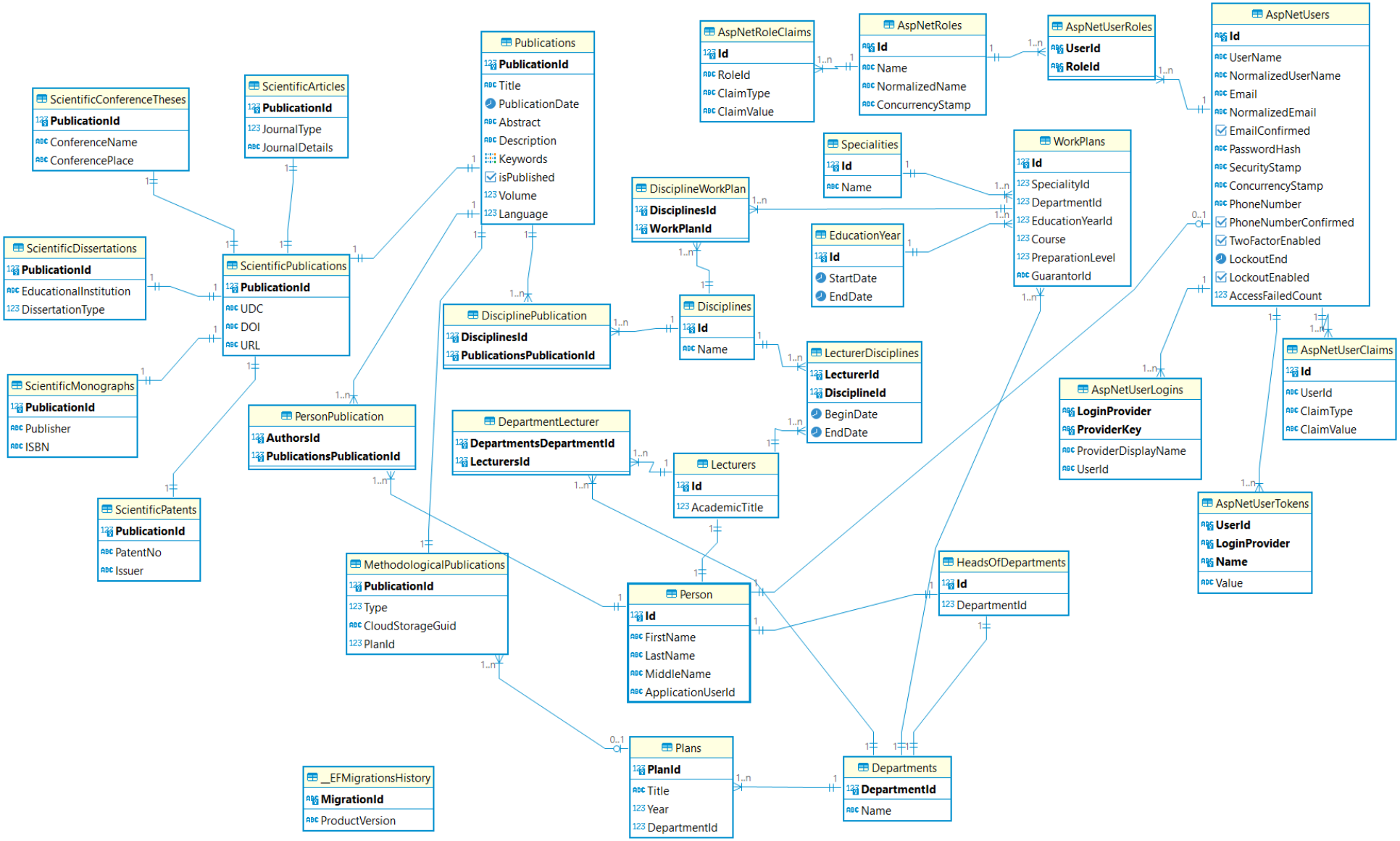


Рисунок 3.1 – ER-діаграма ІС підтримки науково-методичної діяльності ЗВО

4 ПРОГРАМНА МОДЕЛЬ ЗАСТОСУНКУ

4.1 Взаємодія програмних компонентів

Програмна архітектура застосунку ґрунтується на двох ключових компонентах: ASP.NET Core WebAPI та ASP.NET Core MVC. Ці компоненти тісно інтегровані для забезпечення функціональності та користувацького інтерфейсу.

ASP.NET Core WebAPI відповідає за реалізацію API, що забезпечує доступ до даних та виступає в ролі серверу застосунків. Його контролери обробляють HTTP-запити від клієнтських додатків, виконують логіку та CRUD-операції з даними, а потім формують відповіді у форматі JSON. Іншими словами, сервер застосунків виступає в якості посередника між зовнішніми запитами та внутрішньою бізнес-логікою.

ASP.NET Core MVC, з іншого боку, відповідає за створення візуального інтерфейсу користувача. Цей клієнтський застосунок використовує дані, отримані від ASP.NET Core WebAPI, для динамічного генерування сторінок, що відображаються користувачеві. MVC також обробляє дії користувачів, генерує відповідні HTTP-запити та взаємодіє з API для отримання необхідних даних (див. рис. 4.1).

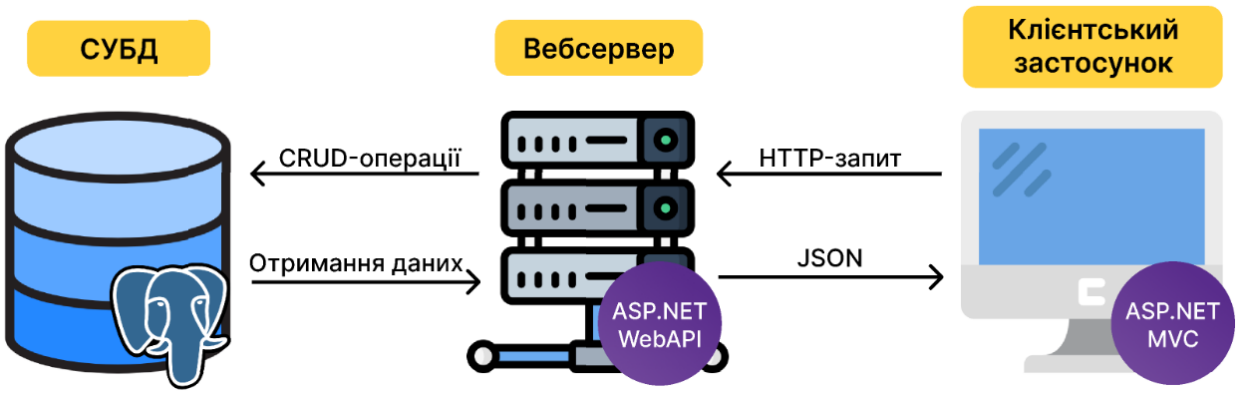


Рисунок 4.1 – Взаємодія компонентів

Взаємодія компонентів:

1) клієнтський застосунок ініціює HTTP-запит до ASP.NET Core WebAPI, щоб отримати доступ до даних або виконати певну дію;

2) ASP.NET Core Web API отримує запит, аналізує його та ідентифікує необхідні дії;

3) контролер взаємодіє з базою даних, сервісами або іншими ресурсами для отримання, оновлення або видалення даних;

4) за отриманими даними формується відповідь у форматі JSON;

5) ASP.NET Core Web API надсилає сформовану відповідь клієнтському додатку;

6) клієнтський застосунок отримує відповідь, оновлює інтерфейс користувача та виконує наступні дії згідно з отриманими даними.

Загалом, архітектура, що базується на ASP.NET Core Web API та ASP.NET Core MVC, реалізує чіткий поділ відповідальності, простоту тестування, масштабованість та гнучкість, роблячи її ефективним рішенням.

4.2 Взаємодія з БД

Для ефективної взаємодії з базою даних, контролери використовують екземпляри класу контексту, кожен з яких успадковує клас DbContext з простору імен Microsoft.EntityFrameworkCore. Цей клас є ключовим елементом в технології Entity Framework Core, яка надає можливість доступу до даних з використанням об'єктно-орієнтованого підходу.

Клас DbContext дозволяє взаємодіяти з об'єктами бази даних за допомогою LINQ (Language Integrated Query), надаючи можливість створювати LINQ to Entities запити, які конвертуються в SQL-запити та взаємодіють з базою даних. Крім того, він також підтримує використання Entity SQL, який надає більш гнучкий підхід до створення SQL-запитів.

Контролери використовують ці екземпляри контексту для отримання доступу до даних з бази даних і виконання різноманітних операцій, таких як читання, запис, оновлення та видалення даних. Крім того, вони відповідають за обробку запитів користувачів та взаємодію з іншими компонентами застосунку.

На рисунку 4.2 наведено загальний вигляд взаємодії контролера з класом, який успадковує DbContext. Цей процес включає в себе взаємодію з таблицями, представленнями, збереженими процедурами та функціями бази даних, що відображаються у вигляді набору класів-моделей. Класи-моделі представляють сутності бази даних і забезпечують зручний доступ та маніпулювання даними через контекст бази даних.

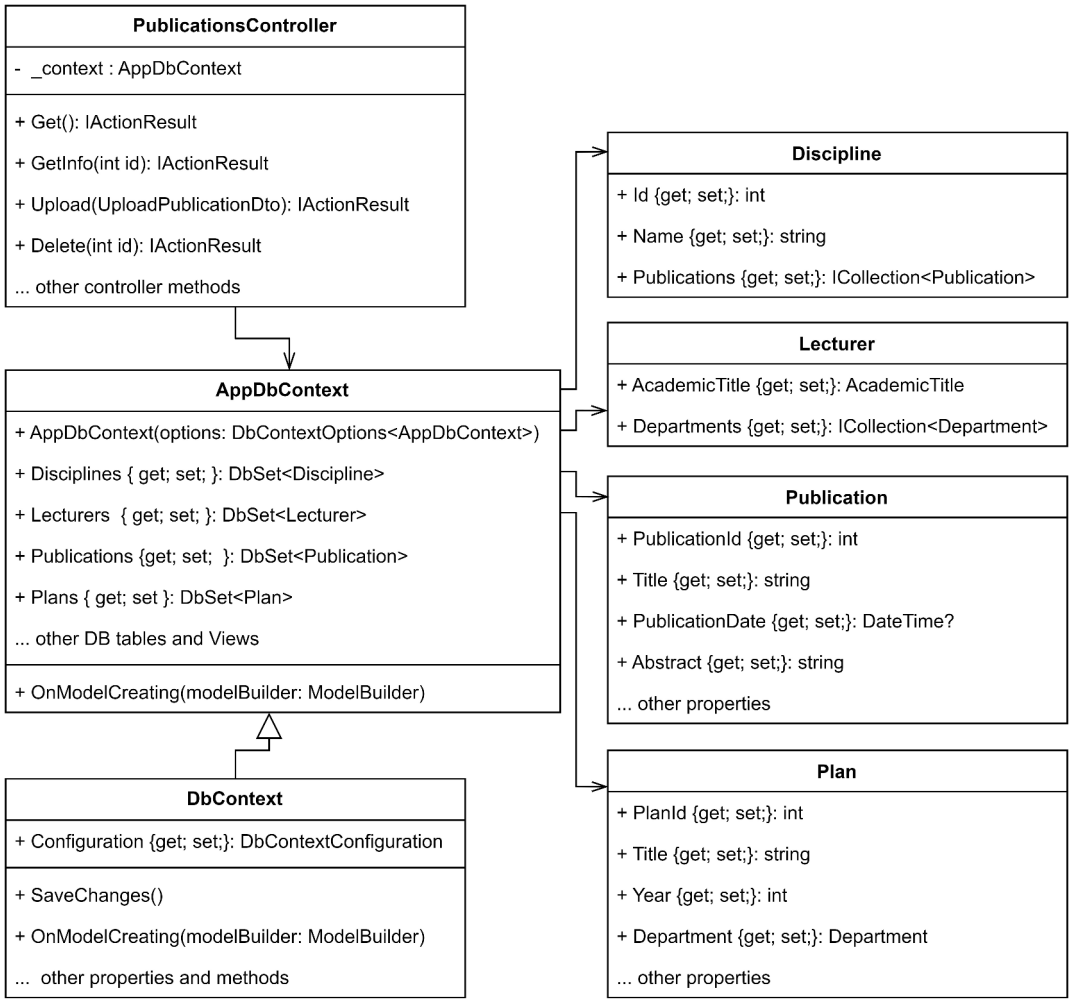


Рисунок 4.2 – UML-діаграма взаємодії контролеру з класом DbContext

Таким чином, використання класу DbContext в ASP.NET Core дозволяє створювати потужні та ефективні веб-застосунки з використанням бази даних для зберігання та обробки інформації.

4.3 Ієрархія веб-сторінок

У клієнтському застосунку рівень представлень формує простір веб-сторінок. Після авторизації, в залежності від групової ролі користувача, йому доступний лише певний набір сторінок, необхідний для вирішення його задач. Наприклад, користувач «Викладач» має доступ до таких сторінок, як «Публікації» та «Плани видань». В свою чергу «Завідувач кафедри» отримує доступ до сторінок «Публікації», «Планування видань», «Дисципліни», «Робота кафедри», «Викладачі». Очевидно, що різні групи користувачів мають доступ до різних сторінок. Повна ієрархія веб-сторінок зображена на рис. 4.3.

Ієрархія веб-сторінок рівня представлень підкріплюється чіткою структурою навігації, що дозволяє користувачам легко знаходити потрібну інформацію та виконувати необхідні дії. Ця структура побудована за принципом розміщення сторінок в підпорядкованих меню або вкладених категоріях, що сприяє зручній навігації та використанню додатку. Такий підхід допомагає підтримувати логічну організацію і структуру додатку, що сприяє покращенню його ефективності та зручності використання для кожного типу користувача.

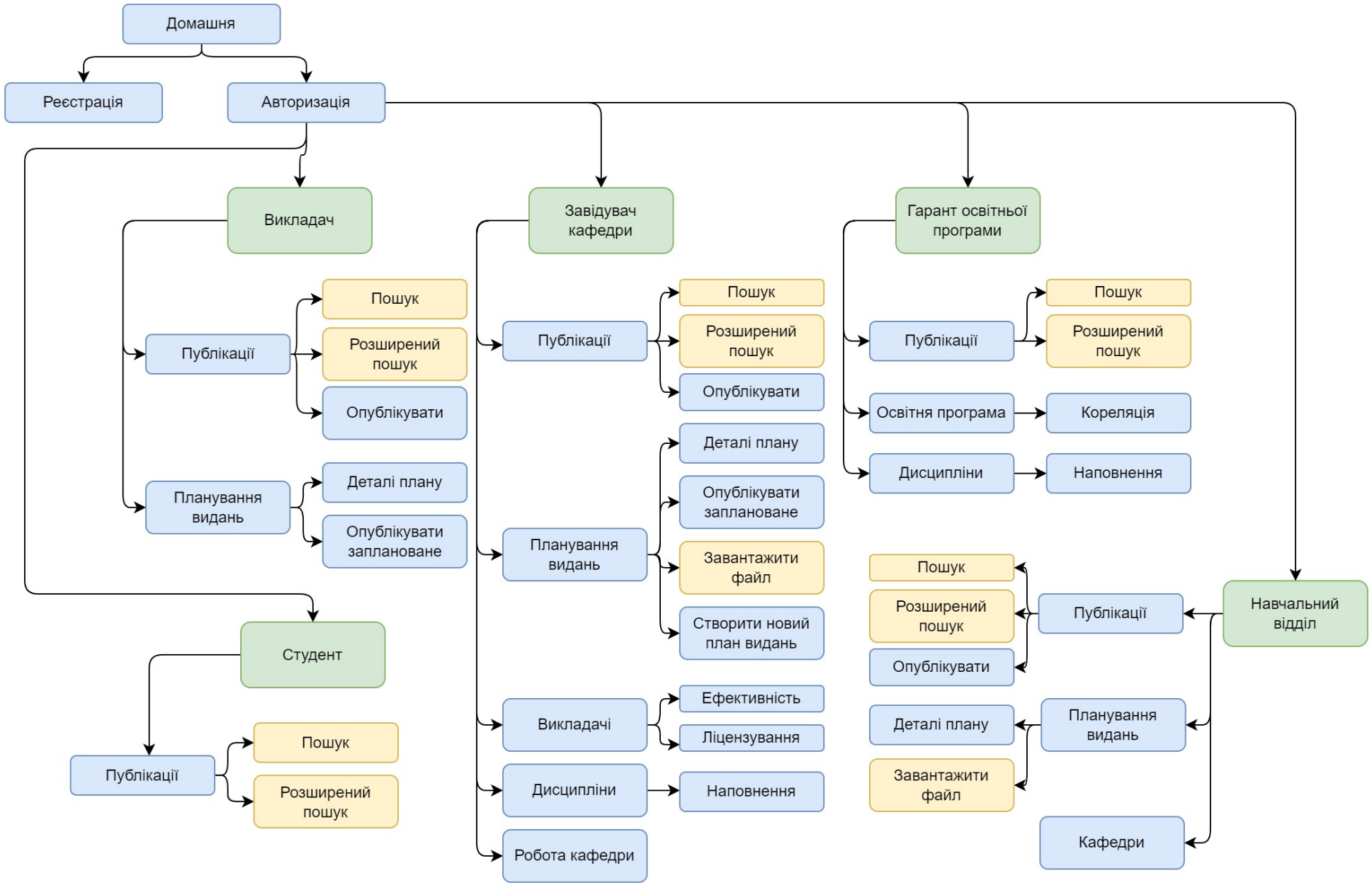


Рисунок 4.3 – Ієрархія веб-сторінок ІС

5 ВИБІР ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ СТВОРЕННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

ІС підтримки науково-методичної діяльності закладу вищої освіти була реалізована у формі веб-застосунку. Це розподілений застосунок, у якому браузер виступає в якості клієнта, а вебсервер - в якості сервера. У даному випадку браузер виконує функції тонкого клієнта, де логіка застосунку зосереджена на сервері, а браузер відповідає за відображення інформації, завантаженої з сервера, та передачу даних користувача. Вибір реалізації саме у веб-форматі обумовлено перевагами, що надають веб-застосунки, зокрема їхньою кросплатформністю, що дозволяє користувачам працювати незалежно від конкретної операційної системи.

Система розроблена із використанням такого стеку технологій:

1) СУБД PostgreSQL;

2) Back-end – ASP.NET Core WebApi;

3) Entity Framework Core та провайдер послуг Npgsql;

4) Front-end – ASP.NET Core MVC, Razor Views, що включає в себе HTML, CSS та JavaScript.

Додатково для створення інтерфейсу використовується JavaScript-бібліотека jQuery та фреймворк Bootstrap 5.

Систему управління базами даних PostgreSQL було обрано головним чином через наступні переваги:

1) вона є безкоштовною, що відрізняє її від інших продуктів, таких як Oracle, SQL Server, Firebird;

2) PostgreSQL має детальну документацію та активну спільноту користувачів; це сприяє швидкій та ефективній розробці, оскільки є можливість швидко знайти відповіді на свої питання та вирішити проблеми завдяки доступному джерелу знань;

3) відповідно до поставлених вимог безпеки, PostgreSQL забезпечує розподілення прав доступу, ефективне управління ролями та організацію рольової системи;

4) дозволяє створювати власні користувацькі функції та процедури, що відкриває широкі можливості для розширення функціональності системи під конкретні потреби проекту.

Серверну інфраструктуру забезпечує платформа ASP.NET Core, обрана через її просту структуру, яка сприяє швидкому створенню та модифікації компонентів застосунку, а також ефективній обробці HTTP-запитів користувачів та налаштуванню маршрутизації. Додатково, ASP.NET Core володіє вбудованими інструментами безпеки, такими як механізми автентифікації і авторизації, що забезпечують надійний захист даних та обмеження доступу до конфіденційної інформації. Крім того, ASP.NET Core відрізняється розширюваністю — вона дозволяє розширювати функціональність веб-застосунків за допомогою пакетів NuGet. Це, в свою чергу, означає підтримку Entity Framework Core, що дозволяє швидко та зручно створювати моделі даних та взаємодіяти з базою даних.

Entity Framework Core (EF Core) - це фреймворк об'єктно-реляційного відображення (ORM), розроблений компанією Microsoft для ефективної роботи з базами даних у проектах, що використовують .NET. EF Core спрощує взаємодію з реляційними базами даних, перетворюючи дані з них у формат, зручний для операцій з об'єктами .NET. Фреймворк автоматично створює SQL-запити на основі запитів LINQ, що значно полегшує процес доступу до даних та робить його зрозумілим та зручним для розробників.

Клієнтську частину реалізує технологія Razor Views, що є необхідною складовою ASP.NET Core і відповідає за представлення (View) у шаблоні MVC. Вибір Razor обумовлено його важливою роллю у відображенні сторінок в ASP.NET Core MVC. Razor використовується для створення динамічних шаблонів, які можуть відображати дані, передані з контролера, і

генерувати HTML на основі цих даних. Ця технологія має вбудовані функції, такі як відображення даних, ітерація по колекціях та умовні конструкції, що спрощує процес розробки інтерфейсу користувача та робить його більш зручним.

Додатково до цього, використання фреймворку Bootstrap 5 дозволяє створити чутливий (responsive) та адаптивний інтерфейс користувача, що оптимізує відображення на різних пристроях та розмірах екрану. Використання бібліотеки jQuery спрощує управління елементами DOM-дерева HTML-документу та мінімізує кількість написаного коду, що полегшує процес розробки та підтримки веб-застосунків.

6 ПРОГРАМНА РЕАЛІЗАЦІЯ

6.1 Створення бази даних

База даних інтегрує різноманітні компоненти, включаючи таблиці, послідовності, представлення, функції, збережені процедури та тригери. Розглянемо запити на створення кількох таблиць БД.

Для створення таблиці "Person" використовується SQL-запит CREATE TABLE IF NOT EXISTS (див. Лістинг 6.1). Цей запит перевіряє наявність таблиці "Person" перед її створенням, що дозволяє запускати запит безпечно, навіть якщо таблиця вже існує.

```
CREATE TABLE IF NOT EXISTS public."Person" (
    "Id" int4 GENERATED BY DEFAULT AS IDENTITY( INCREMENT BY 1
MINVALUE 1 MAXVALUE 2147483647 START 1 CACHE 1 NO CYCLE) NOT
NULL,
    "FirstName" text NOT NULL,
    "LastName" text NOT NULL,
    "MiddleName" text NOT NULL,
    "ApplicationUserId" text NULL,
    CONSTRAINT "PK_Person" PRIMARY KEY ("Id"),
    CONSTRAINT "FK_Person_AspNetUsers_ApplicationUserId" FOREIGN
KEY ("ApplicationUserId") REFERENCES public."AspNetUsers"("Id")
);
```

Лістинг 6.1 – Створення таблиці Person

Кожне поле таблиці має свій тип даних, який визначає формат інформації, що зберігається в цьому полі. Наприклад, поле "Id" є первинним ключем (PRIMARY KEY) і має тип "int4", який представляє цілі числа. Вказане поле також має атрибути GENERATED BY DEFAULT AS IDENTITY, які встановлюють автоматичне збільшення значень для ідентифікаторів за замовчуванням. Ці значення генеруються автоматично при вставці нових рядків у таблицю. Поле "FirstName", "LastName" та "MiddleName" мають тип

"text" і не допускають значень NULL, тобто вони обов'язкові для заповнення. Поле "ApplicationUserId" також має тип "text", але може приймати значення NULL.

Також визначено зовнішній ключ (FOREIGN KEY), який посилається на поле "Id" таблиці "AspNetUsers". Цей зв'язок не лише забезпечує цілісність даних між таблицями, але й визначає взаємозв'язок між користувачем системи та інформацією про нього, яка зберігається в таблиці "Person". Кожен користувач системи, що реєструється, отримує унікальний ідентифікатор, який автоматично надається полем "Id" таблиці "AspNetUsers". Цей ідентифікатор використовується як зовнішній ключ у таблиці "Person", щоб установити зв'язок між обліковим записом користувача і його персональними даними.

Для створення таблиці "PersonPublication" також використовується SQL-запит CREATE TABLE IF NOT EXISTS (див. Лістинг 6.2).

```
CREATE TABLE IF NOT EXISTS public."PersonPublication" (
    "AuthorsId" int4 NOT NULL,
    "PublicationsPublicationId" int4 NOT NULL,
    CONSTRAINT "PK_PersonPublication" PRIMARY KEY ("AuthorsId",
"PublicationsPublicationId"),
    CONSTRAINT "FK_PersonPublication_Person_AuthorsId" FOREIGN
KEY ("AuthorsId") REFERENCES public."Person"("Id") ON DELETE
CASCADE,
    CONSTRAINT
"FK_PersonPublication_Publications_PublicationsPublicationId"
FOREIGN KEY ("PublicationsPublicationId") REFERENCES
public."Publications"("PublicationId") ON DELETE CASCADE
);
```

Лістинг 6.2 – Створення асоціативної сутності PersonPublication

Поле "AuthorsId" та "PublicationsPublicationId" визначають зовнішні ключі, які посилаються на відповідні поля таблиць "Person" та "Publications". Ці ключі забезпечують зв'язок між записами цих таблиць та забезпечують

цілісність даних. Обмеження PRIMARY KEY визначає складний первинний ключ для таблиці "PersonPublication", який складається з двох полів. Це забезпечує унікальність комбінації значень обох полів в таблиці.

У даному запиті використовується ON DELETE CASCADE, що означає те, що при видаленні запису з батьківської таблиці, автоматично будуть видалені всі записи з дочірньої таблиці, які посилаються на цей запис.

У випадку таблиці "PersonPublication", якщо буде видалений запис з таблиці "Person" з ідентифікатором "Id", який використовується в полі "AuthorsId" таблиці "PersonPublication", всі записи в таблиці "PersonPublication", де це поле містить видалений ідентифікатор, також будуть автоматично видалені.

Те ж саме стосується і таблиці "Publications". Якщо буде видалений запис з таблиці "Publications", який має ідентифікатор "PublicationId", що використовується в полі "PublicationsPublicationId" таблиці "PersonPublication", всі відповідні записи з таблиці "PersonPublication" також будуть автоматично видалені.

Цей механізм дозволяє автоматично підтримувати цілісність даних у випадку видалення записів, що мають зв'язки з іншими таблицями, і впевнитися, що дані у всіх пов'язаних таблицях залишаються узгодженими.

Повний лістинг команд SQL-запитів, які використовуються для створення всіх таблиць БД наведені в Додатку В.

6.2 Запити до БД для вирішення задач користувачів

Для вирішення задач користувачів інформаційної системи підтримки науково-методичної діяльності ВНЗ до бази даних додано необхідні представлення, функції, збережені процедури та тригери.

1) Для вирішення задачі С1 створено представлення PublishedPublicationsWithAuthors (вихідний запит наведено у Додатку Д.1), для звернення до якого необхідний запит:

```
SELECT * FROM PublishedPublicationsWithAuthors;
```

2) для вирішення задач С2 та С3 створено функцію GetPublicationInfo (див. Додаток Е.1), для звернення до якої використовуються запити:

```
SELECT * FROM GetPublicationInfo(123);
```

або

```
SELECT CloudStorageGuid FROM GetPublicationInfo(123);
```

3) для вирішення задач С4-С5, ЗК7-ЗК8 та НВ5 використовується функція GetPublicationsByAuthorId (див. Додаток Е.2), для звернення до якої використовується запит:

```
WITH publications AS (
SELECT * FROM GetPublicationsByAuthorId(456)
)
SELECT * FROM publications
WHERE PublicationDate BETWEEN '2024-01-01' AND '2024-12-31';
```

Конструкція WITH в SQL використовується для створення тимчасових підзапитів, які можна використовувати в основному запиті. Це допомагає зробити код більш читабельним та організованим. У даному випадку WITH використовується для виклику функції GetPublicationsByAuthorId і подальшої фільтрації її результатів. Для фільтрації використовується SQL BETWEEN;

4) для задачі В1 та В6 створено збережену процедуру AddMethodologicalPublication, для звернення до якої виконується запит:

```
CALL AddMethodologicalPublication('Sample Title',
'2024-05-16 12:00:00', 'Sample Abstract', 'Sample Description',
ARRAY['Keyword1', 'Keyword2'], TRUE, 1, 0, 1, 'sample-guid',
1, ARRAY[1, 2, 3], p_publication_id);
```

Основою для цієї процедури слугує SQL-команда INSERT;

5) для задач В2-В3 та ЗК4 створено збережені процедури UpdateMethodologicalPublication та DeletePublication (див. Додаток Е.4, Додаток Е.5), основу котрих становлять SQL-команди UPDATE та DELETE відповідно;

6) для задач В4-В5, ЗК1-ЗК2, ЗК5, НВ2-НВ3 використовуються функції GetDepartmentPlans (див. Додаток Е.6) та GetPlanInfo (Додаток Е.7) відповідно;

7) для задачі ЗК3 створено функції AddPlan (Додаток Е.8) та AddPublicationsToPlan (Додаток Е.9), котрі викликаються послідовно одна за одною;

8) для задачі ЗК6 та Г1 створено функцію GetDisciplinePublications (Додаток Е.10), основою для котрих є команди JOIN, LEFT JOIN та WHERE. Викликається стандартним чином:

```
SELECT * FROM GetDisciplinePublications(1);
```

9) для задач НВ1 та НВ4 створено функцію GetAllDepartmentsPlans;

10) для вирішення задачі Г2 можна скористатись раніше створеною функцією GetDisciplinePublications, але з іншим запитом:

```
WITH publications AS (
   SELECT * FROM GetDisciplinePublications(1)
)
SELECT * FROM publications WHERE "AuthorId" = 7;
```

11) для вирішення задачі ЗК9 створено функцію GetPublicationsByDepartment, котра повертає інформацію про публікації, видані на певній кафедрі. Також, за необхідності, дані можна фільтрувати за допомогою запиту:

```
WITH publications AS (
   SELECT * FROM GetPublicationsByDepartment(1)
)
SELECT * FROM publications
WHERE PublicationDate BETWEEN '2024-01-01' AND '2024-12-31'
```

Для забезпечення цілісності інформації у базі даних створено тригерні функції та тригери. Наприклад, для таблиці Publications реалізовано тригер (див. Додаток Г.1-Г2), який забороняє встановлення значення isPublished в TRUE, якщо будь-яке з ключових полів публікації має значення NULL. Цей тригер працює наступним чином: перед оновленням запису викликається тригерна функція, яка перевіряє, чи встановлюється isPublished у TRUE.

Якщо так, функція перевіряє, чи поля таблиці мають значення NULL. У разі виявлення порушення цілісності даних, тригер блокує оновлення, генеруючи відповідне виключення. Це допомагає підтримувати повноту даних у системі.

Із повним переліком тригерних функцій та тригерів можна ознайомитись у Додатку Г.

6.3 Програмна реалізація інтерфейсу

Реалізація програмного інтерфейсу ASP.NET MVC застосунку включає кілька важливих етапів, кожен з яких має свої функціональність та цілі:

1) Моделі (Models): в цьому компоненті визначаються класи моделей, які представляють дані, що використовуються в застосунку. Моделі можуть містити властивості, методи та правила валідації даних для забезпечення консистентності та цілісності інформації;

2) Представлення (Views): створюються HTML-шаблони, які відображають дані користувачам. За допомогою мови розмітки, такої як Razor, дані з моделей вставляються в HTML-код, формуючи сторінки, форми та інші елементи інтерфейсу;

3) Контролери (Controllers): цей компонент відповідає за обробку запитів користувача та взаємодію з моделями та представленнями. Контролери приймають запити, виконують необхідну логіку, взаємодіють з моделями для отримання/збереження даних та передають результати до представлень для відображення;

4) маршрутизація: налаштування маршрутів, які визначають, які URL-шляхи відповідають яким контролерам та діям. Маршрутизація дозволяє визначити, як система має обробляти запити з різних URL-адрес;

5) аутентифікація та авторизація: додавання механізмів аутентифікації та авторизації користувачів. Налаштування системи входу/реєстрації, управління правами доступу;

6) обробка форм: реалізація механізмів для обробки форм, які вводять користувачі. Це включає валідацію введених даних, надсилання даних до серверу застосунків із подальшим збереженням інформації у базі даних, а також надсилання користувачу повідомлень про результати операцій.

7) логіка бізнес-процесів: реалізація логіки, яка визначає бізнес-процеси застосунку. Це включає обчислення, перевірки, маніпуляції з даними, взаємодія з іншими сервісами та зовнішніми API.

У даному веб застосунку типовою є задача виведення вмісту таблиці або представлення у вигляді форми або веб-сторінки. Розглянемо усю послідовність дій на прикладі, коли Студент хоче отримати інформацію про наявні публікації за пошуковими критеріями.

Для цього контролер PublicationsController має метод All (див. Лістинг 6.3). Відповідно до правил маршрутизації {controller}/{action}/{id?}, цей метод викликається, коли користувач виконує GET-запит за URL-адресою publications/all

```
[HttpGet("all")]
public async Task<IActionResult> All(string searchTerm, string authorName,
DateTime? StartDateFilter, DateTime? endDateFilter, int[]? categories = null)
{
    var result = await apiService.GetPublicationsAsync(
        searchTerm,authorName,startDateFilter,endDateFilter,categories);

    if (!result.Success)return BadRequest(result.ErrorMessage);

    var disciplines = (await apiService.GetDisciplinesListAsync());
    var categoriesList = new SelectList(disciplines,"Id","Name",categories);

    if(categories != null)
      foreach (var item in categoriesList)
        if (categories.Contains(int.Parse(item.Value))) item.Selected = true;

    ViewBag.SearchTerm = searchTerm;
    ViewBag.AuthorName = authorName;
    ViewBag.StartDateFilter = startDateFilter;
    ViewBag.EndDateFilter = endDateFilter;
    ViewBag.Categories = categoriesList;

    return View("Index", result.Data);
}
```

Лістинг 6.3 – Метод контролера, який повертає сторінку з публікаціями та пошуком за критеріями

Цей метод приймає параметри для фільтрації результатів, такі як searchTerm, authorName, StartDateFilter, endDateFilter, categories, та викликає сервіс apiService, щоб отримати список публікацій, які задовольняють вказані критерії. Якщо операція успішна, метод відображає сторінку Index з даними, отриманими від серверу застосунків. У разі невдачі метод повертає статус помилки та повідомлення про помилку. Крім того, метод отримує список категорій дисциплін і передає його у представлення для відображення у формі фільтрації.

Скорочений вміст представлення, у якому відбувається виведення інформації про наукові та методичні публікації для студентів, наведено у Лістингу 6.4. Синтаксис Razor дозволяє поєднувати HTML-код з C# кодом:

1) цикл foreach використовується для ітерації по колекції Model, що містить дані про публікації. Таким чином для кожного об'єкта publication у колекції виводяться різні властивості.

2) підставлення значень — для виведення властивостей об'єктів використовуються `@` символ та C# код. Наприклад, `@publication.Title` виводить заголовок публікації, а `@publication.PublicationDate.Year` — рік публікації;

3) вкладені цикли foreach використовуються для ітерації по колекціях властивостей публікацій, таких як список авторів (publication.Lecturers) та ключові слова (publication.Keywords);

4) умовні оператори if використовуються для умовного відображення роздільника коми між авторами публікації. Таким чином умова !author.Equals(publication.Lecturers.Last()) перевіряє, чи поточний автор є останнім у списку, щоб вирішити, чи додавати кому.

Цей шаблон дозволяє динамічно генерувати HTML-код на основі даних моделі та їх властивостей.

```
<table id="publicationTable" class="display">
 <tbody>
 @foreach (var publication in Model)
 {
  <tr>
   <td>
    <div class="py-1">
     <div>
       <a class="text-dark publication-title"
          asp-controller="Publications"
          asp-action="Index"
          asp-route-id="@publication.PublicationId">
           @publication.Title &mdash;
@publication.PublicationDate.Year.ToString()
       </a>
     </div>
     <div>
      @foreach (var author in publication.Lecturers)
      {
       <a class="author-list"
          asp-controller="Lecturers"
          asp-action="Index"
          asp-route-id="@author.Id">
           @author.LastName @author.FirstName @author.MiddleName
       </a>
       @if (!author.Equals(publication.Lecturers.Last()))
       {
        <span class="comma">, </span>
       }
      }
     </div>
     <div>
      <p class="publication-description">@publication.Description</p>
     </div>
     <div>
      @foreach (var keyWord in publication.Keywords)
      {
       <span class="badge bg-primary">
           @keyWord
       </span>
      }
     </div>
    </div>
   </td>
  </tr>
 }
 </tbody>
</table>
```

Лістинг 6.4 - Виведення вмісту представлення Index клієнту

Наступним розглянемо метод того ж контролеру UploadMonograph (див. Лістинг 6.5). Цей метод призначений для завантаження наукової монографії в систему. Позначка [HttpPost("upload-monograph")] вказує, що цей метод буде обробляти POST-запити за URL-адресою "upload-monograph".

```
[AuthorizeSession(Roles.Lecturer, Roles.HeadOfDepartment)]
[HttpPost("upload-monograph")]
public async Task<IActionResult> UploadMonograph(PostScientificMonographDto
publicationDto)
{
 if (!ModelState.IsValid)
  return View(publicationDto);

 await apiService.PostScientificMonographAsync(publicationDto);

 return RedirectToAction("All");
}
```

Лістинг 6.5 - Метод контролера для завантаження монографій

Як можна побачити, в методі є атрибут [AuthorizeSession(Roles.Lecturer, Roles.HeadOfDepartment)], що вказує на те, що для доступу до цього методу користувач повинен мати роль викладача або завідувача кафедри. Це забезпечує контроль доступу до функціоналу завантаження монографій у систему.

Першим кроком метод перевіряє валідність моделі, переданої з клієнта, за допомогою умови if (!ModelState.IsValid). Якщо дані моделі не є валідними, метод повертає представлення з тим самим об'єктом моделі publicationDto для відображення помилок введених даних.

Якщо модель валідна, метод викликає асинхронний метод apiService.PostScientificMonographAsync, передаючи йому об'єкт publicationDto, який містить дані про наукову монографію. Цей метод відповідає за відправку даних на сервер застосунку для подальшої обробки.

Після успішного завантаження монографії в систему, метод повертає перенаправлення (HTTP 302) на дію "All" контролера. Це означає, що після завантаження користувач буде перенаправлений на сторінку, яка відображає всі наявні монографії.

Аналогічним чином забезпечується створення усіх інших сторінок для користувачів.

6.4 Програмна реалізація серверу застосунків

У даному проекті для створення серверної частини застосунку використовується веб API на основі технології ASP.NET Core WebAPI. ASP.NET Core WebAPI - це фреймворк, який дозволяє створювати RESTful веб-сервіси для забезпечення взаємодії між клієнтами та сервером за допомогою HTTP-протоколу.

Основною одиницею роботи з веб API є контролери. Контролер - це клас, який містить методи, котрі відповідають на різні HTTP-запити, такі як GET, POST, PUT, DELETE. Для створення контролера у ASP.NET Core використовується ключове слово Controller, а також відповідні атрибути, що вказують на маршрутизацію та інші налаштування.

Маршрутизація визначає, які URL-шляхи пов'язані з кожним методом контролера. У ASP.NET Core маршрутизація виконується за допомогою атрибутів маршрутизації, таких як [HttpGet], [HttpPost], [HttpPut], [HttpDelete] та інших. Маршрути можуть бути задані як шаблони URL-шляхів або використовувати атрибути маршрутизації для більш гнучкої настройки.

У методах контролерів можуть бути параметри, які використовуються для передачі даних у веб API. Ці параметри можуть бути отримані з URL-шляху, запиту, тіла запиту або заголовків. ASP.NET Core автоматично прив'язує параметри методів до даних запиту.

Методи контролерів повинні повертати об'єкт типу IActionResult, який представляє результат виконання методу. Це може бути відповідь з даними, статус-код, перенаправлення або інші операції. ASP.NET Core надає широкі можливості для формування різноманітних відповідей на запити.

Для виконання бізнес-логіки та взаємодії з базою даних у контролерах можуть використовуватися сервіси. Сервіси є класами, які надають функціональність, необхідну для обробки запитів в контролерах. Вони

можуть бути викликані з методів контролерів та виконувати різноманітні операції, такі як отримання, збереження, оновлення або видалення даних.

Розглянемо приклади використання таких сервісів для роботи з БД у деяких методах контролерів. Наприклад, метод контролеру, який взаємодіє з базою даних для отримання списку публікацій з урахуванням різних фільтрів, таких як пошуковий запит, ім'я автора, дати публікації та категорії, наведено у Додатку Ж.1. Його роботу можна розділити на декілька логічних етапів:

1) створення основи запиту — відбувається із використанням об’єкту типу IQueryable<Publication>, що дозволяє взаємодіяти з базою даних через LINQ. Цей тип використовується, оскільки він представляє запит, який можна виконати в БД, і отримати набір результатів. Використання типу IQueryable дозволяє будувати складні запити з використанням фільтрів, сортування та інших операцій, і тільки при необхідності (одноразово) виконувати їх у базі даних. У даному прикладі, метод Include() використовується для включення даних авторів публікацій у результати запиту, щоб уникнути проблем із «лінивим завантаженням». Потім за допомогою методу Where() встановлюється умова для вибору лише опублікованих публікацій;

2) додавання фільтрів на основі параметрів запиту — в залежності від переданих параметрів у запиті (наприклад, пошукового запиту, імені автора, дат початку та кінця, категорій), до початкового запиту додаються відповідні фільтри для обмеження результатів. Наприклад, LINQ-запит (див. Лістинг 6.6) буде трансформований в SQL-запит, який обмежує результати за допомогою умови, яка перевіряє дату публікації (див. Лістинг 6.7);

```
if (startDateFilter.HasValue)
{
 startDateFilter = DateTime.SpecifyKind(startDateFilter.Value,
DateTimeKind.Utc);
 query = query.Where(p => p.PublicationDate >=
startDateFilter.Value.Date);
}
```

Лістинг 6.6 – Фільтр публікацій за датою у формі LINQ

```
SELECT *
FROM Publications
WHERE PublicationDate >= @StartDateFilter
```

Лістинг 6.7 - Фільтр публікацій за датою у формі SQL

3) виконання сконструйованого запиту у БД та отримання колекції публікацій, які відповідають заданим критеріям;

4) отримані з БД дані про публікації проектуються в об'єкти DTO, які містять лише необхідну інформацію для відображення на сторінці веб-застосунку;

5) повернення результатів у вигляді об'єктів DTO, котрі упаковуються у відповідь HTTP (Ok), яку повертає сервер клієнту.

Взаємодія з БД також може відбуватись із використанням збережених процедур або функцій (див. Лістинг 6.8).

```
[Authorize(Roles=$"{Roles.HeadOfDepartment},{Roles.Lecturer}")]
[HttpDelete]
public async Task<IActionResult> Delete(int id)
{
 var result = await _appDbContext.ExecuteStoredProcedureAsync(
"DeletePublication", new SqlParameter("@PublicationId", id));

 if (result > 0) return Ok("Publication deleted successfully.");
 else return NotFound();
}
```

Лістинг 6.8 – Виконання збереженої процедури

У даному випадку відбувається видалення публікації за допомогою виклику збереженої процедури з контексту бази даних, програма передає параметри до цієї процедури, яка потім виконується на боці сервера БД. Після виконання процедури, база даних повертає результат, наприклад, кількість змінених рядків або значення виводу. Залежно від результату виклику процедури, програма приймає відповідне рішення, таке як повернення успішного повідомлення або відповідного статусу помилки.

Також розглянемо метод Login, котрий реалізує аутентифікацію та авторизацію користувача в системі (див. Додаток Ж.2). Ось короткий опис його функціональності:

1) перевірка валідності моделі — метод спочатку перевіряє, чи відповідає модель LoginUser заданим правилам валідації та, якщо модель не валідна, метод поверне відповідь з помилкою BadRequest;

2) аутентифікація користувача — після валідації моделі метод викликає сервіс аутентифікації (_authService) для спроби виконання входу користувача. Цей сервіс повертає пару значень: success, що показує, чи вдалося ввійти користувачу, і error, якщо є якась помилка. Якщо аутентифікація пройшла успішно, метод отримує роль користувача і генерує токен доступу для користувача за допомогою _authService.GenerateTokenString(user, role).

3) формування відповіді — у випадку успішної аутентифікації метод формує об'єкт відповіді з токеном доступу, інформацією про користувача та його роллю. Відповідь повертається у форматі JSON зі статусом HTTP Ok.

4) неуспішна аутентифікація — якщо аутентифікація не вдалася, метод повертає відповідь з помилкою BadRequest, яка містить текст помилки з сервісу аутентифікації.

Як можна побачити, застосунок реалізований за принципами сервісної архітектури, що сприяє зменшенню залежностей між його компонентами. Кожна функціональна частина, така як аутентифікація, або взаємодія з БД, винесена в окремий сервіс. Це дозволяє полегшити розробку, тестування та підтримку коду шляхом розділення відповідальностей.

Крім того, застосунок використовує принципи SOLID та дотримується чистої архітектури, що дозволяє йому бути легко масштабованим. Кожен компонент виконує свої власні функції і може бути замінений або модифікований без впливу на інші частини системи. В результаті отримано гнучкий, легко розширюваний та підтримуваний застосунок, який відповідає вимогам сучасної розробки ПЗ.

7 БЕЗПЕКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

На рівні бази даних безпека забезпечується шляхом створення ролей, кожна з яких відповідає конкретній групі користувачів у системі. У цьому контексті визначені ролі, такі як "Викладач", "Завідувач кафедри", "Гарант освітньої програми" та "Навчальний відділ". Для студентів, які не вимагають аутентифікації в системі, передбачено роль "unauthorized".

Кожна з вищезазначених ролей має свій набір прав і привілеїв, що визначає доступ до конкретних ресурсів та функціональності системи. Наприклад, роль "Гарант освітньої програми" може мати доступ до даних про закріплену за ним навчальну програму та публікації викладачів, але не мати права на редагування планів видань кафедри.

У той же час, студенти, які не мають аутентифікації в системі, під’єднуються до БД під роллю "unauthorized", що обмежує їх доступ до більшості конфіденційних даних та функціональності системи. Це обумовлено необхідністю забезпечення конфіденційності інформації та запобігання несанкціонованому доступу до даних.

У Додатку И наведений список прав та привілеїв для кожної ролі, що регламентують доступ до конкретних ресурсів бази даних.

Для забезпечення безпеки на рівні застосунку використовується технологія ASP.NET Identity. Ця технологія є вбудованою системою аутентифікації та авторизації для платформи ASP.NET. Вона надає розширені можливості для керування користувачами, ролями та правами в застосунку. Один з головних компонентів ASP.NET Identity - це клас UserManager, який відповідає за управління користувачами.

UserManager використовується для створення, редагування, видалення користувачів, а також для управління їхніми ролями та правами доступу. Він надає доступ до таких операцій, як аутентифікація, створення нових

облікових записів користувачів, зміна паролів, підтвердження адрес електронної пошти та інше.

Однією з ключових функцій UserManager є автоматичне хешування паролів користувачів. При створенні облікового запису користувача або зміні паролю, пароль автоматично хешується з використанням криптографічного алгоритму SHA-256, що забезпечує безпеку паролю під час зберігання в базі даних. Такий підхід унеможливлює прямий доступ до паролю користувача навіть адміністраторам бази даних, забезпечуючи високий рівень захисту конфіденційності даних.

В процесі роботи системи важливо забезпечити відповідність безпеки та контролю доступу до функцій та ресурсів для кожного користувача. Це досягається за допомогою послідовного проходження процесів аутентифікації та авторизації. Після успішної аутентифікації, система визначає активну роль користувача та надає йому відповідний рівень доступу до функціональності системи.

Авторизація визначає, які дії користувач може виконувати в системі після успішного входу. Це включає доступ до конкретних сторінок, функцій або ресурсів, що регулюються на основі ролей, до яких належить користувач.

Такий підхід дозволяє забезпечити контрольований доступ до функціональності системи та захистити конфіденційні дані від несанкціонованого доступу. Крім того, він забезпечує повну відповідність вимогам безпеки та дозволяє ефективно керувати правами доступу для кожного типу користувача у системі.

8 ОПИС РОБОТИ СИСТЕМИ

8.1 Авторизація

Після відкриття веб-сайту користувач одразу потрапляє на екран автентифікації, де йому необхідно ввести свої логін та пароль для входу до системи (див. рис. 8.1).

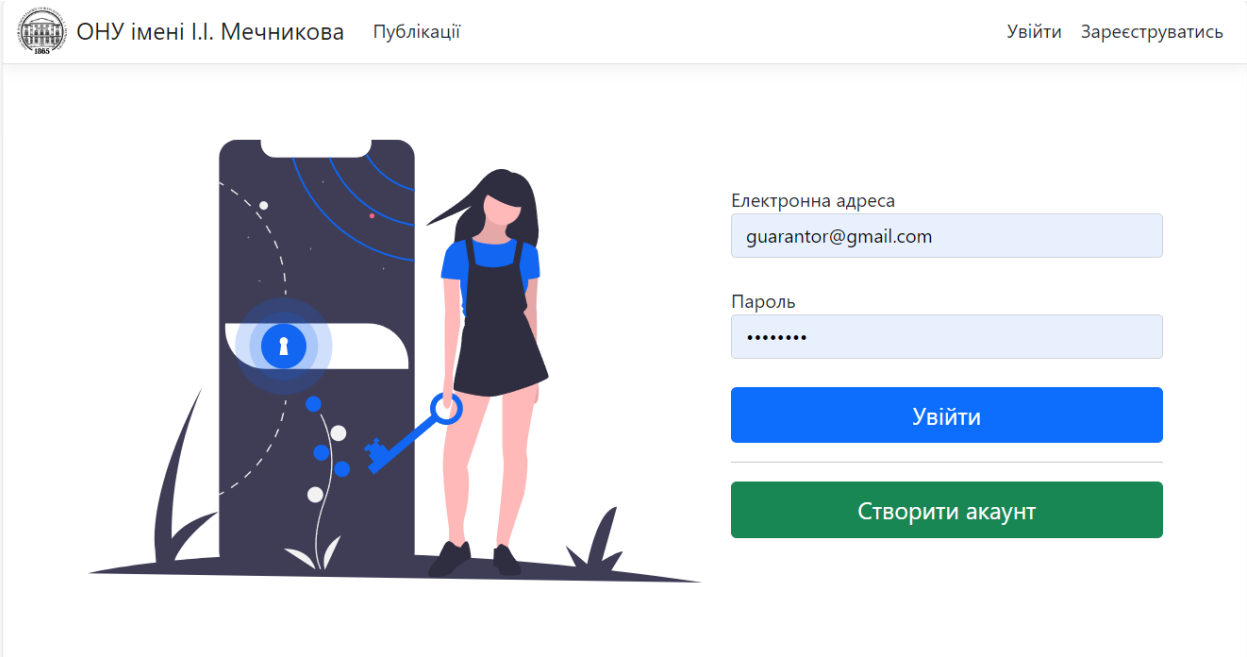


Рисунок 8.1 – Сторінка авторизації

Якщо користувач вводить неправильний пароль або логін, то він отримує відповідну помилку.

Далі покроково розглянемо інтерфейси кожної групи користувачів.

8.2 Інтерфейс Студента

Система не вимагає авторизації від користувача Студент, отже він одразу може перейти до підбору публікацій (див. рис. 8.2).

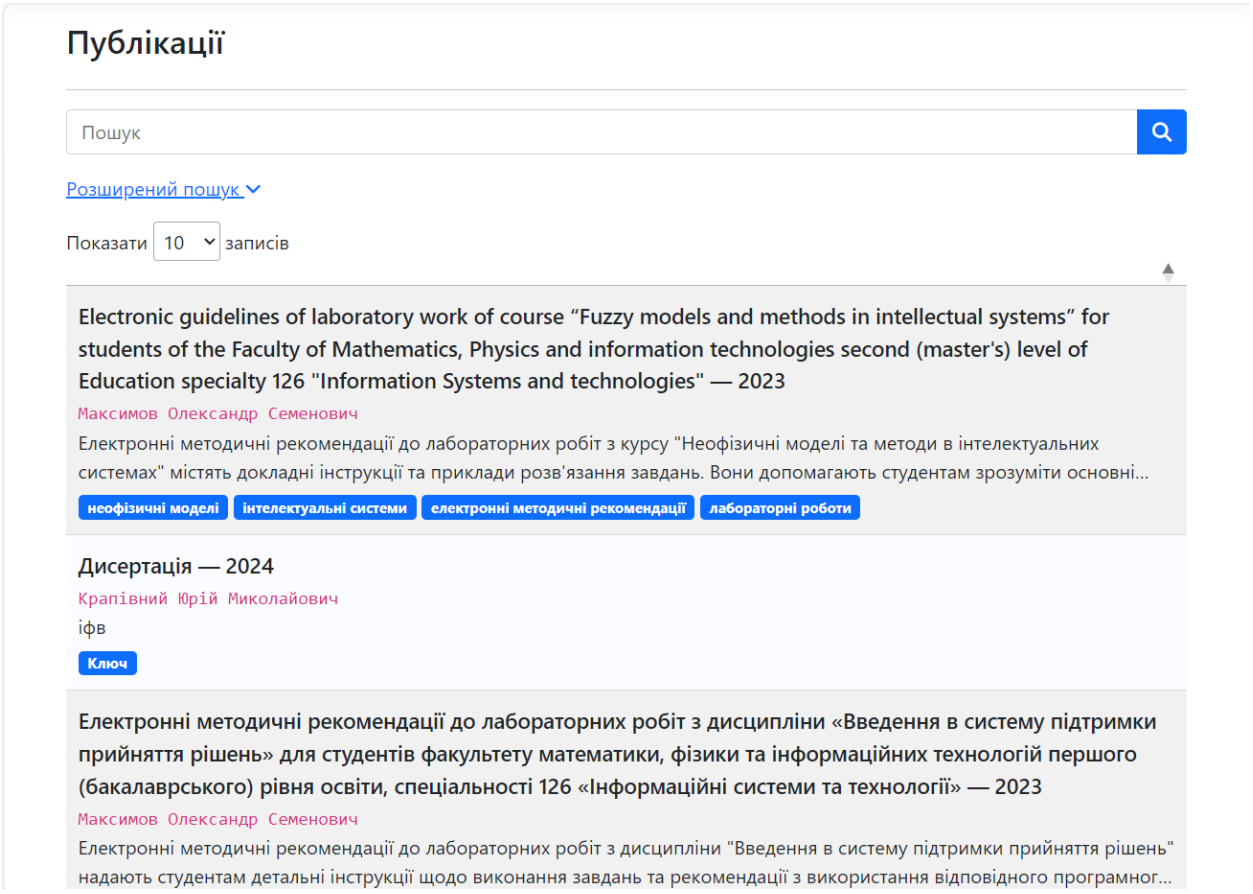


Рисунок 8.2 – Сторінка з публікаціями

Публікації презентовані користувачу у вигляді карток, кожна з яких містить: назву публікації, авторів, короткий опис та ключові слова.

Традиційно, угорі сторінки розташований рядок, у якому користувач може здійснити пошук за назвою, ключовими словами, описом тощо. Також присутній розширений пошук, котрий з’являється по натиску (див. рис. 8.3).

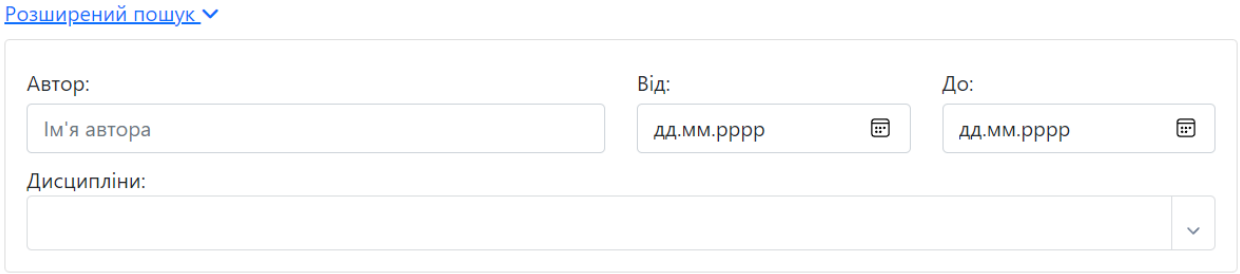


Рисунок 8.3 – Розширений пошук публікацій

У формі розширеного пошуку є така функціональність: пошук за ПІБ автора публікації, фільтрація за датою видання, пошук за дисципліною.

При натисканні на назву публікації відкривається сторінка із повною інформацією про публікацію (див. рис. 8.4).

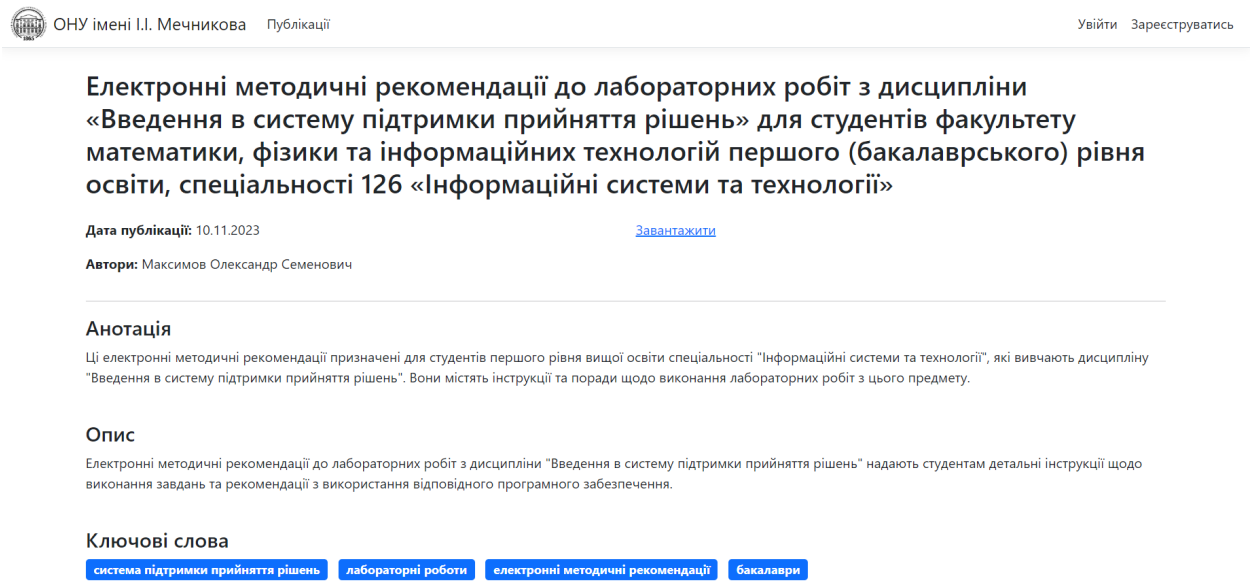


Рисунок 8.4 – Сторінка з деталями публікації

Також є можливість завантажити файл за посиланням, та він автоматично буде відкритий у сусідній вкладці.

8.3 Інтерфейс Викладача

Після авторизації, система одразу висвічує роль користувача, під якою він авторизувався, поряд із його електронною адресою (див. рис. 8.5).

На сторінці із публікаціями викладач може розмістити власну публікацію певної категорії. Для цього використовуються традиційні засоби екранних форм: текстові та числові поля, випадаючі списки (у тому числі з множинним вибором), календарі тощо (див. рис. 8.6).

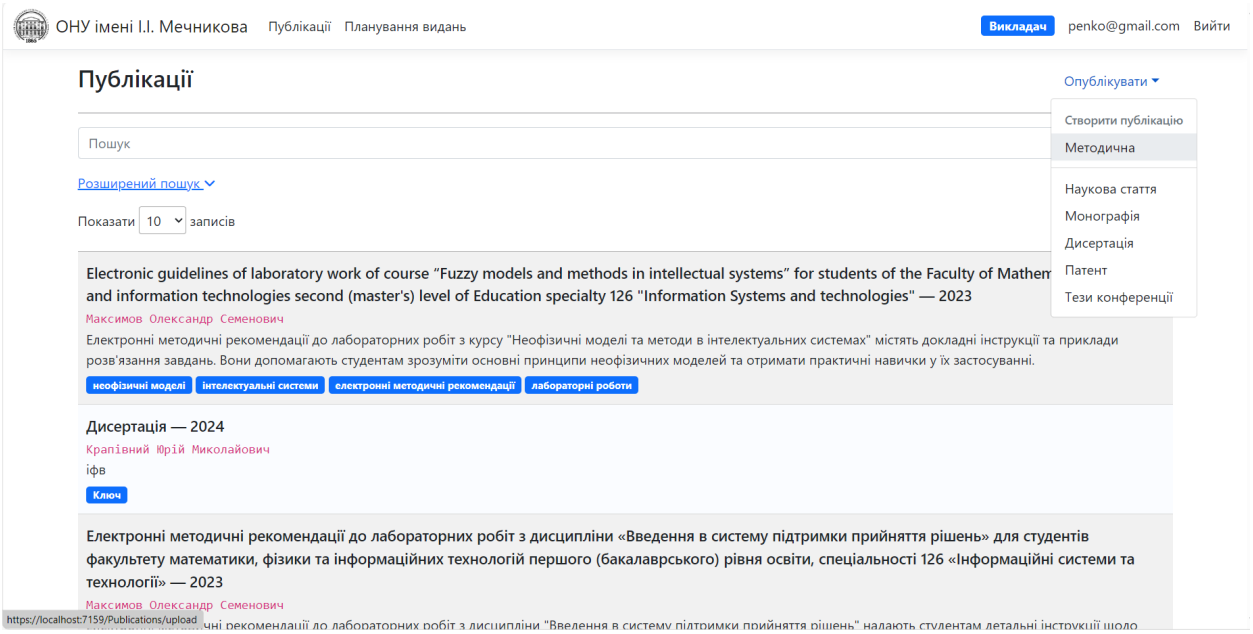


Рисунок 8.5 – Сторінка публікацій для викладача

Вочевидь, для різних типів публікацій передбачено певний набір реквізитів, виходячи із специфіки публікації.

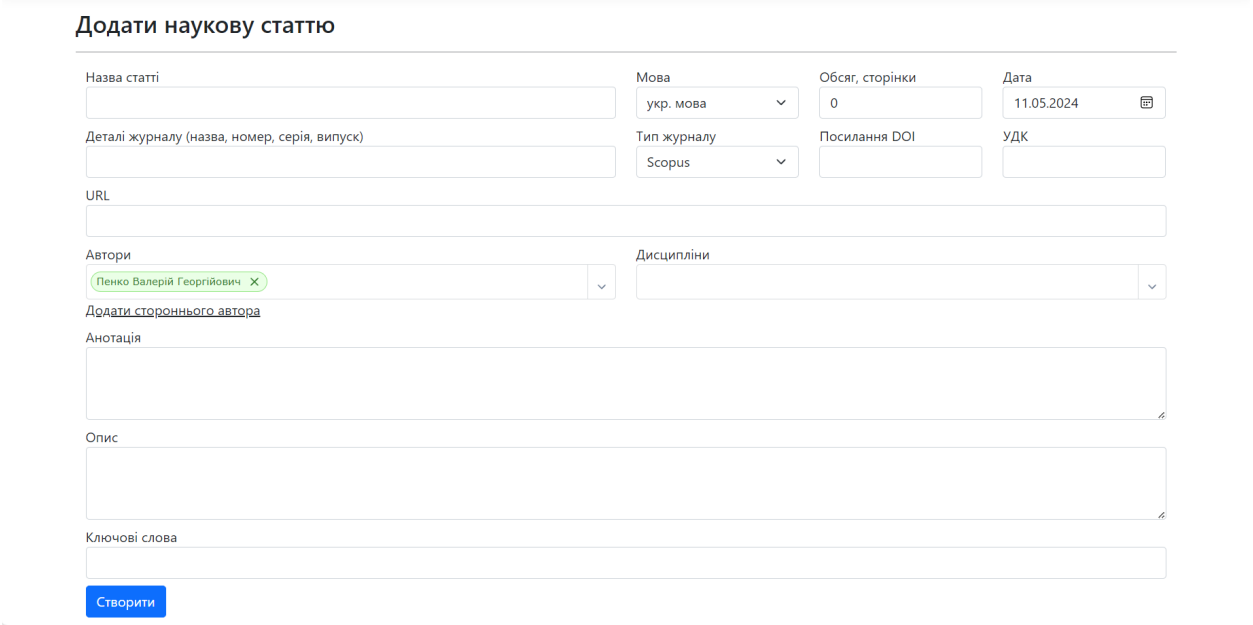


Рисунок 8.6 – Сторінка додавання наукових статей

Також викладачу доступний пункт меню «Планування видань». У ньому викладач може завантажити заплановану публікацію та відслідковувати свій прогрес у виконанні плану (див. рис. 8.7).

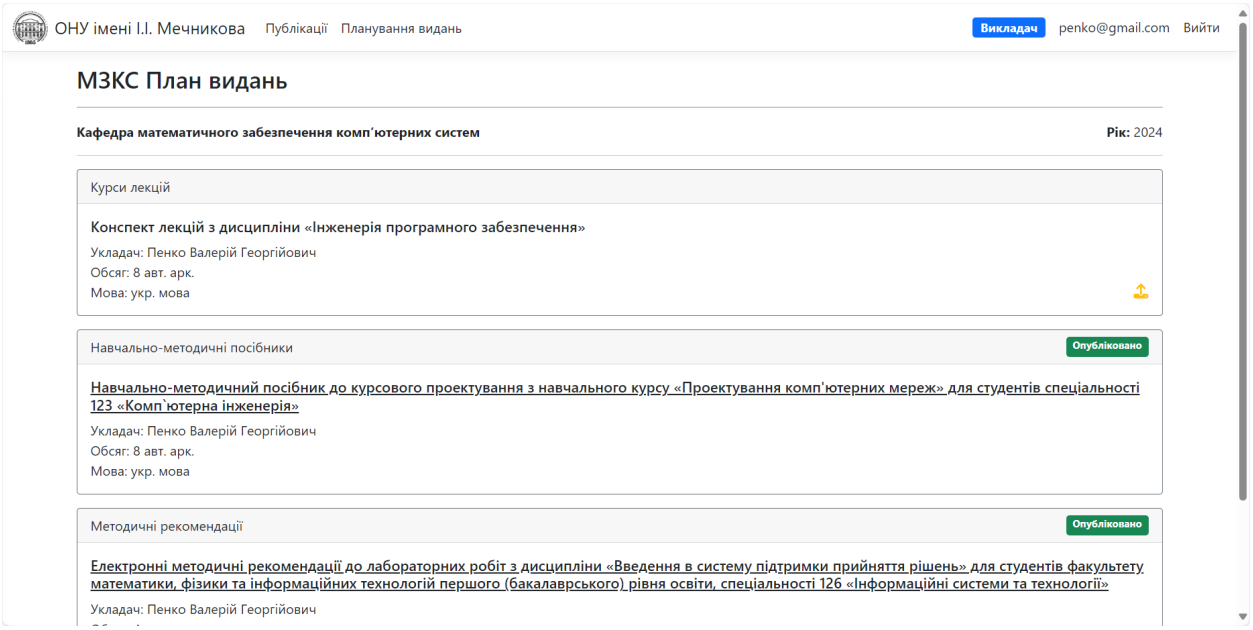


Рисунок 8.7 – План видань кафедри для викладача

Насамкінець, при натисканні на власну електронну адресу у верхньому правому кутку екрану, відкривається Особистий кабінет викладача (див. рис. 8.8). У своєму особистому кабінеті викладач може редагувати або видаляти публікації, фільтрувати бібліографію за певним періодом та переглядати статистику власних публікацій.

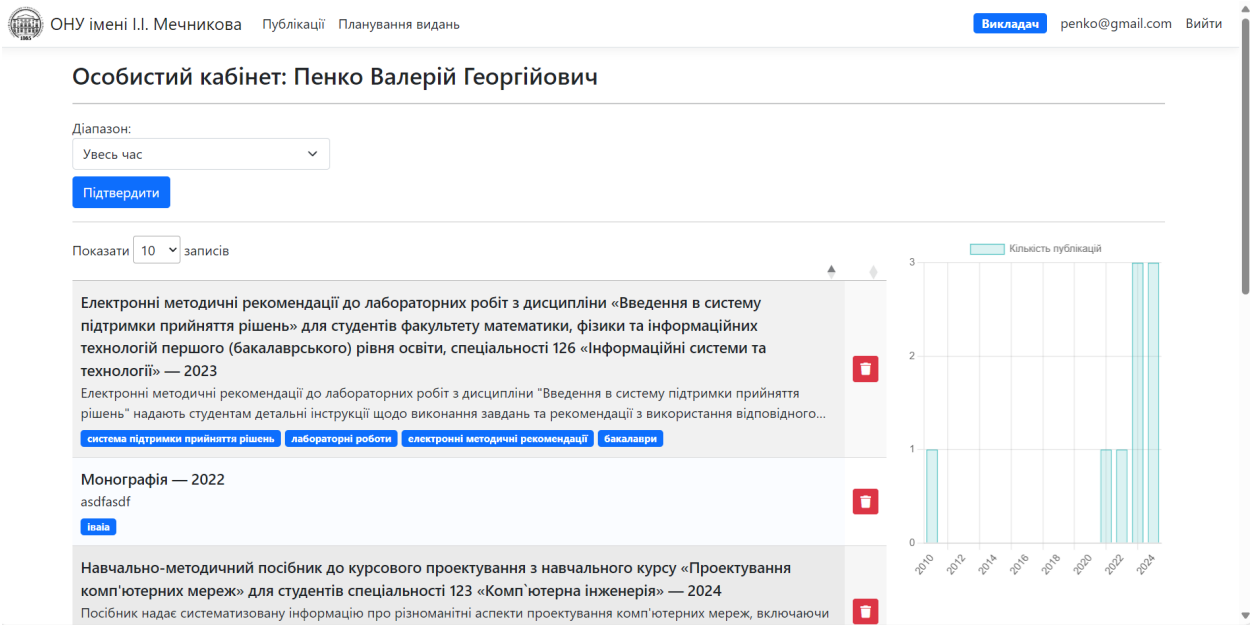


Рисунок 8.8 – Особистий кабінет викладачами

8.4 Інтерфейс Завідувача кафедри

Завідувач кафедри, окрім меню «Публікації», також отримує такі: «Планування видань», «Викладачі», «Дисципліни» та «Робота кафедри» (див. рис. 8.9).

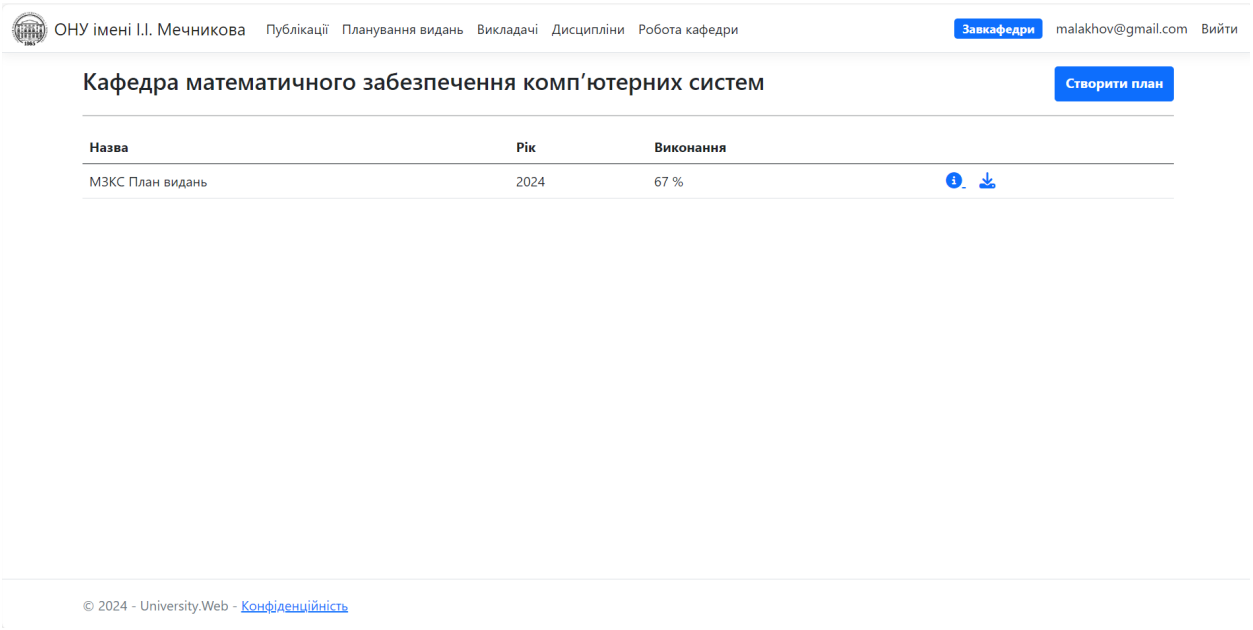


Рисунок 8.9 – Планування видань для Завідувача кафедри

Почнемо із планування видань на календарний рік. На даній сторінці Завідувачу кафедри відображаються усі наявні плани видань із короткою інформацією: назва, календарний рік, прогрес виконання. Можна переглянути більш детальну інформацію, натиснувши на відповідну кнопку поряд із планом (див. рис. 8.10).

Доступна така інформація: перелік публікацій, котрі мали бути опубліковані, статус (опубліковано чи ні), укладачів, обсяг, мову та тип методичного видання. Деталі публікацій можна змінювати до того, як вони були опубліковані викладачем.

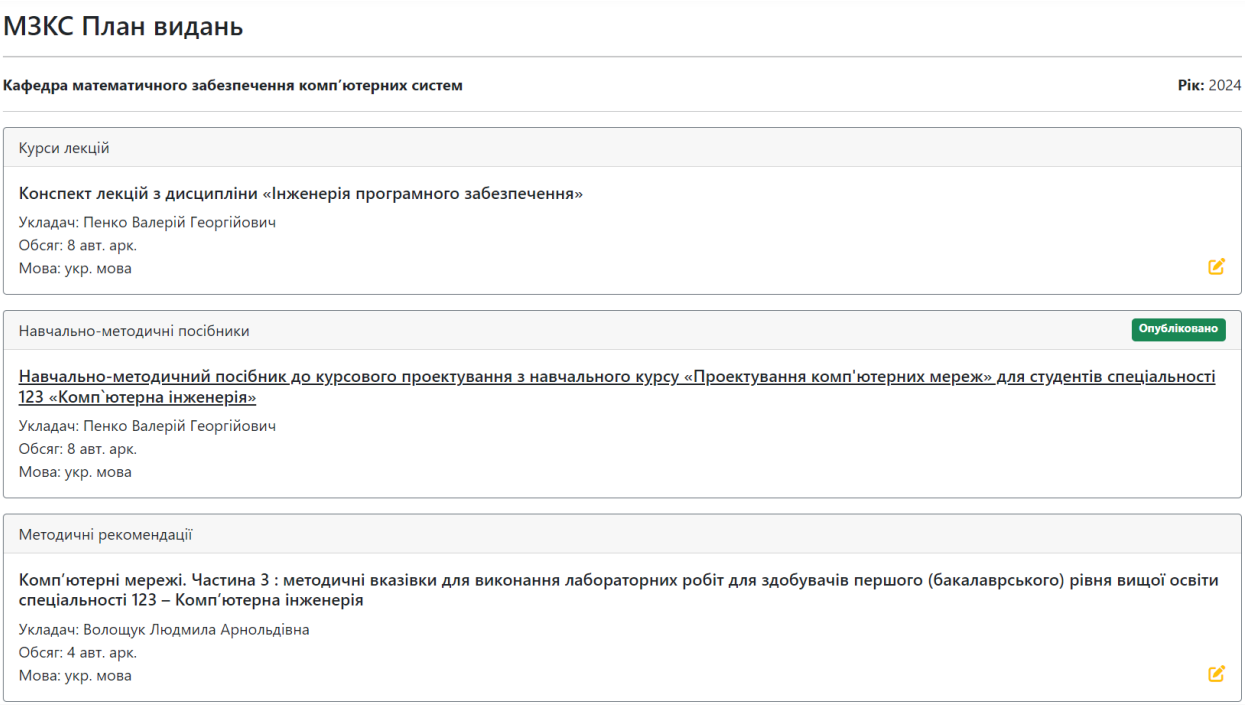
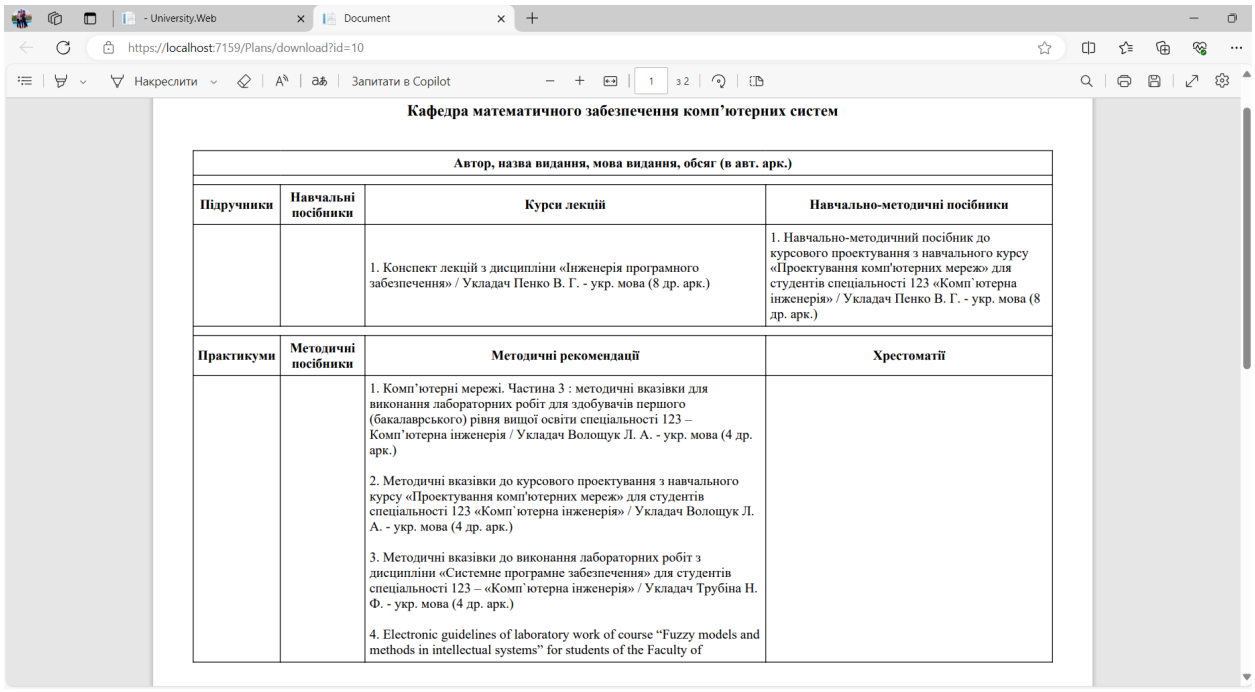


Рисунок 8.10 – Деталі плану видань для Завідувача кафедри

Створений план видань можна також завантажити, та він відкриється у сусідньому вікні у форматі PDF-документу за зразком, затвердженим кафедрою (див. рис. 8.11).



Кафедра математичного забезпечення комп'ютерних систем

Автор, назва видання, мова видання, обсяг (в авт. арк.)			
Підручники	**Навчальні посібники**	**Курси лекцій**	**Навчально-методичні посібники**
		1. Конспект лекцій з дисципліни «Інженерія програмного забезпечення» / Укладач Пенко В. Г. - укр. мова (8 др. арк.)	1. Навчально-методичний посібник до курсового проектування з навчального курсу «Проектування комп'ютерних мереж» для студентів спеціальності 123 «Комп`ютерна інженерія» / Укладач Пенко В. Г. - укр. мова (8 др. арк.)
Практикуми	**Методичні посібники**	**Методичні рекомендації**	**Хрестоматії**
		1. Комп'ютерні мережі. Частина 3 : методичні вказівки для виконання лабораторних робіт для здобувачів першого (бакалаврського) рівня вищої освіти спеціальності 123 – Комп'ютерна інженерія / Укладач Волощук Л. А. - укр. мова (4 др. арк.) 2. Методичні вказівки до курсового проектування з навчального курсу «Проектування комп'ютерних мереж» для студентів спеціальності 123 «Комп`ютерна інженерія» / Укладач Волощук Л. А. - укр. мова (4 др. арк.)  3. Методичні вказівки до виконання лабораторних робіт з дисципліни «Системне програмне забезпечення» для студентів спеціальності 123 – «Комп`ютерна інженерія» / Укладач Трубіна Н. Ф. - укр. мова (4 др. арк.) 4. Electronic guidelines of laboratory work of course "Fuzzy models and methods in intellectual systems" for students of the Faculty of	

Рисунок 8.11 – Документ плану видань

Створити план видань кафедри можна за формою, наведеною на рисунку 8.12.

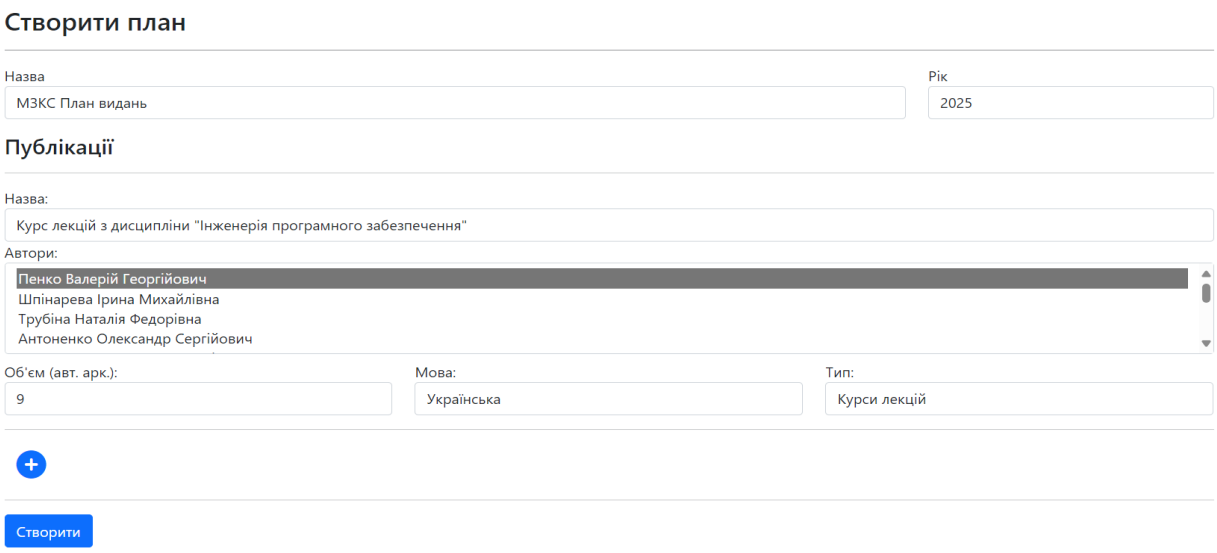


Рисунок 8.12 – Форма створення плану видань кафедри

Перейдемо до наступного пункту меню — «Викладачі» (див. рис. 8.13). На даній сторінці відображено перелік викладачів, котрі закріплені за кафедрою, разом з їх вченим званням. Для кожного викладача доступні такі дії: перевірка ефективності роботи та контроль умов ліцензування викладача.

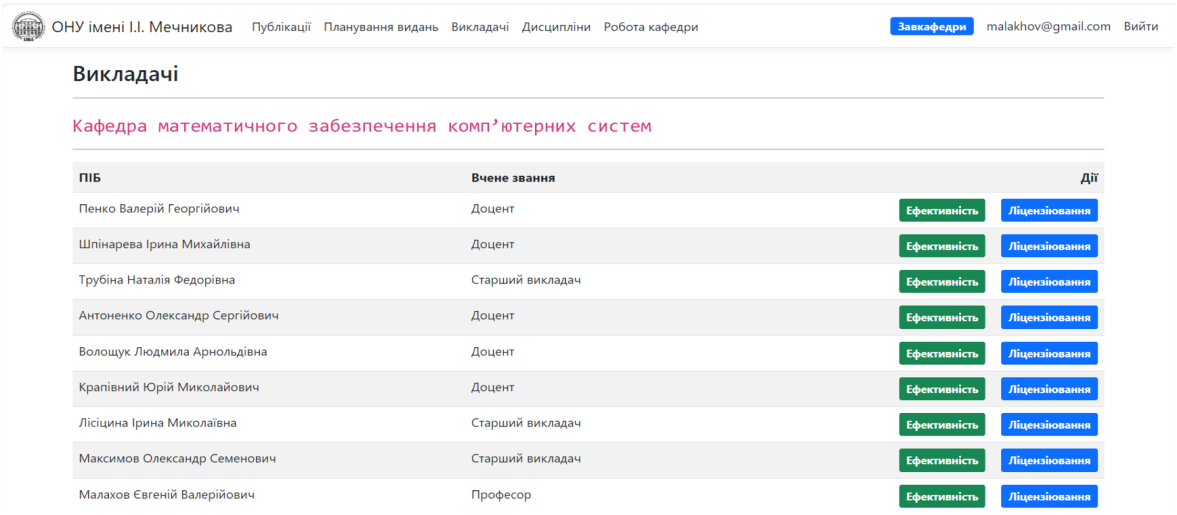


Рисунок 8.13 – Сторінка «Викладачі»

На сторінці «Ефективність роботи викладача» є можливість фільтрації публікацій за періодом, доступні такі як: 1 рік, 5 років, увесь час та власний діапазон (див. рис. 8.14).

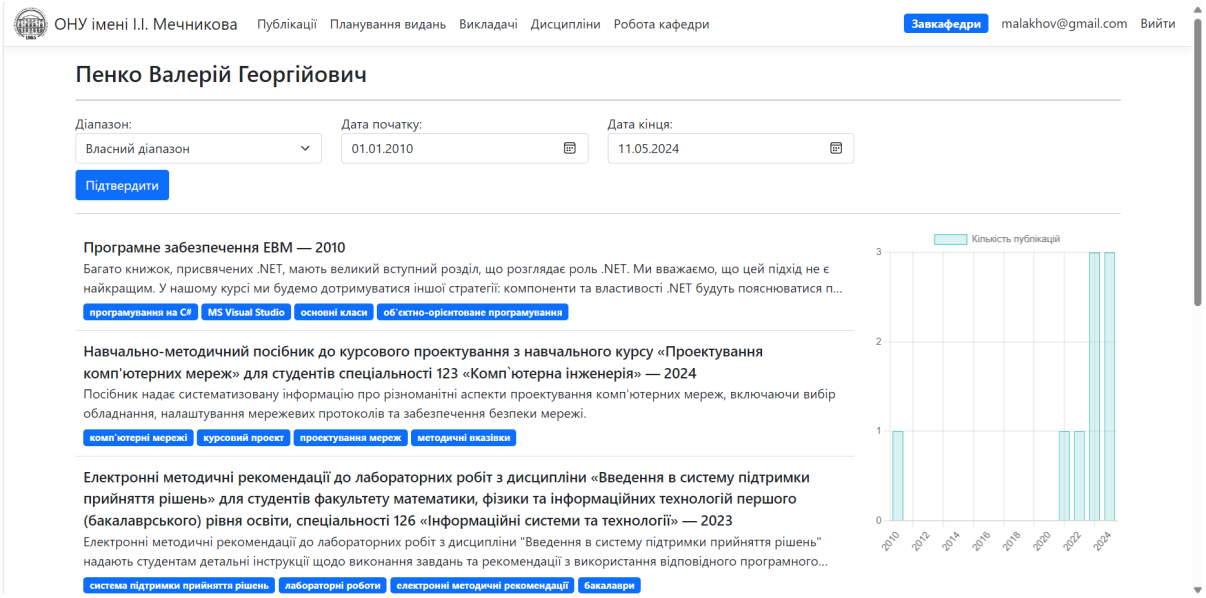
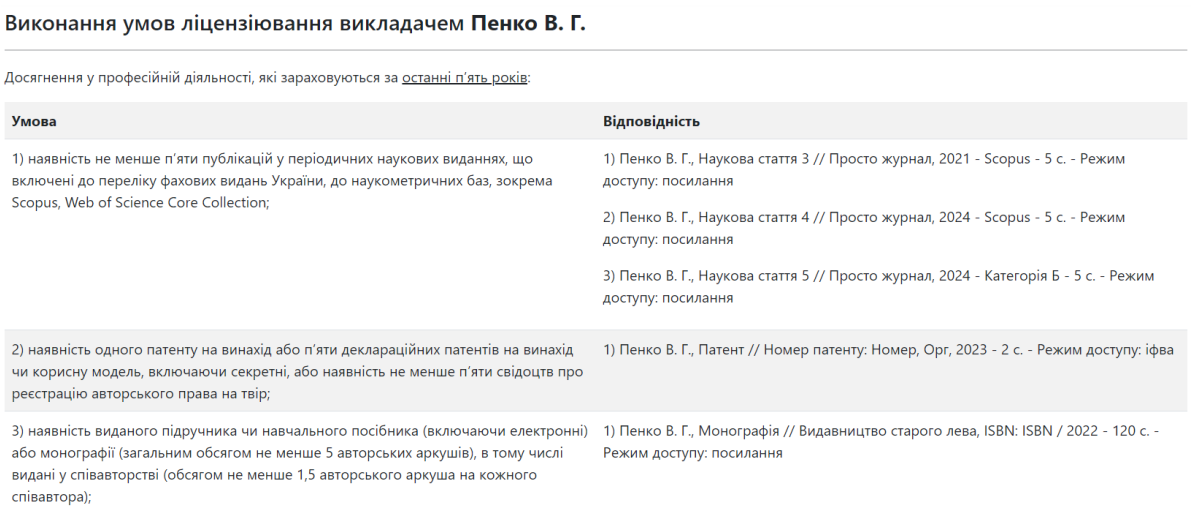


Рисунок 8.14 – Ефективність роботи викладача

На сторінці «Ліцензування» передбачено відображення деяких пунктів із умов ліцензування, що стосуються наукових та методичних видань та публікації, котрі задовольняють критеріям (див. рис. 8.15).



Виконання умов ліцензіювання викладачем Пенко В. Г.

Досягнення у професійній діяльності, які зараховуються за останні п'ять років:

Умова	Відповідність
1) наявність не менше п'яти публікацій у періодичних наукових виданнях, що включені до переліку фахових видань України, до наукометричних баз, зокрема Scopus, Web of Science Core Collection;	1) Пенко В. Г., Наукова стаття 3 // Просто журнал, 2021 - Scopus - 5 с. - Режим доступу: посилання 2) Пенко В. Г., Наукова стаття 4 // Просто журнал, 2024 - Scopus - 5 с. - Режим доступу: посилання 3) Пенко В. Г., Наукова стаття 5 // Просто журнал, 2024 - Категорія Б - 5 с. - Режим доступу: посилання
2) наявність одного патенту на винахід або п'яти деклараційних патентів на винахід чи корисну модель, включаючи секретні, або наявність не менше п'яти свідоцтв про реєстрацію авторського права на твір;	1) Пенко В. Г., Патент // Номер патенту: Номер, Орг, 2023 - 2 с. - Режим доступу: ifва
3) наявність виданого підручника чи навчального посібника (включаючи електронні) або монографії (загальним обсягом не менше 5 авторських аркушів), в тому числі видані у співавторстві (обсягом не менше 1,5 авторського аркуша на кожного співавтора);	1) Пенко В. Г., Монографія // Видавництво старого лева, ISBN: ISBN / 2022 - 120 с. - Режим доступу: посилання

Рисунок 8.15 – Виконання умов ліцензування викладачем

У меню «Дисципліни» доступний перелік дисциплін, котрі викладаються на кафедрі (див. рис. 8.16).

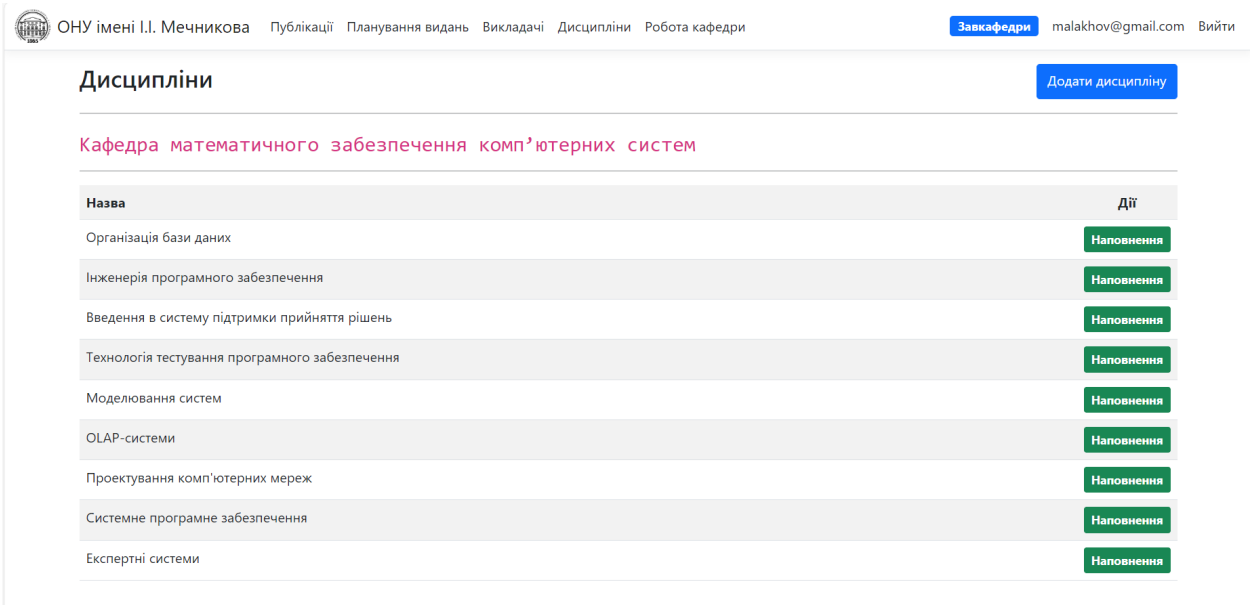


Рисунок 8.16 – Сторінка «Дисципліни»

Для кожної дисципліни доступна можливість переглянути її наповненість як методичними, так і науковими матеріалами (див. рис. 8.17).

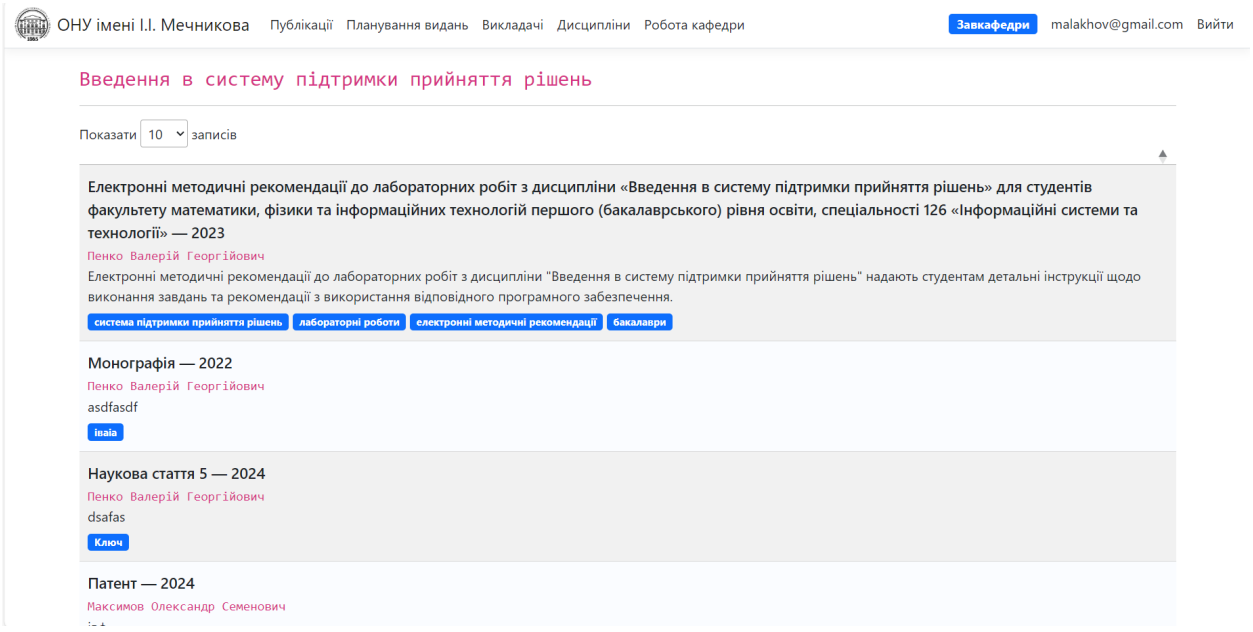


Рисунок 8.17 - Сторінка «Наповненість дисципліни»

Врешті-решт, пункт меню «Робота кафедри» дозволяє відстежувати публікації, видані на кафедрі. Функціонал включає в себе фільтрацію за часовим діапазоном, типом публікації та підтипом. Наприклад (див. рис. 8.18), за 2023 рік було видано 15 методичних публікацій, усього — 30 публікацій. Також можливо здійснювати сортування за назвою, датою видання, автором та здійснювати пошук за ключовими словами.

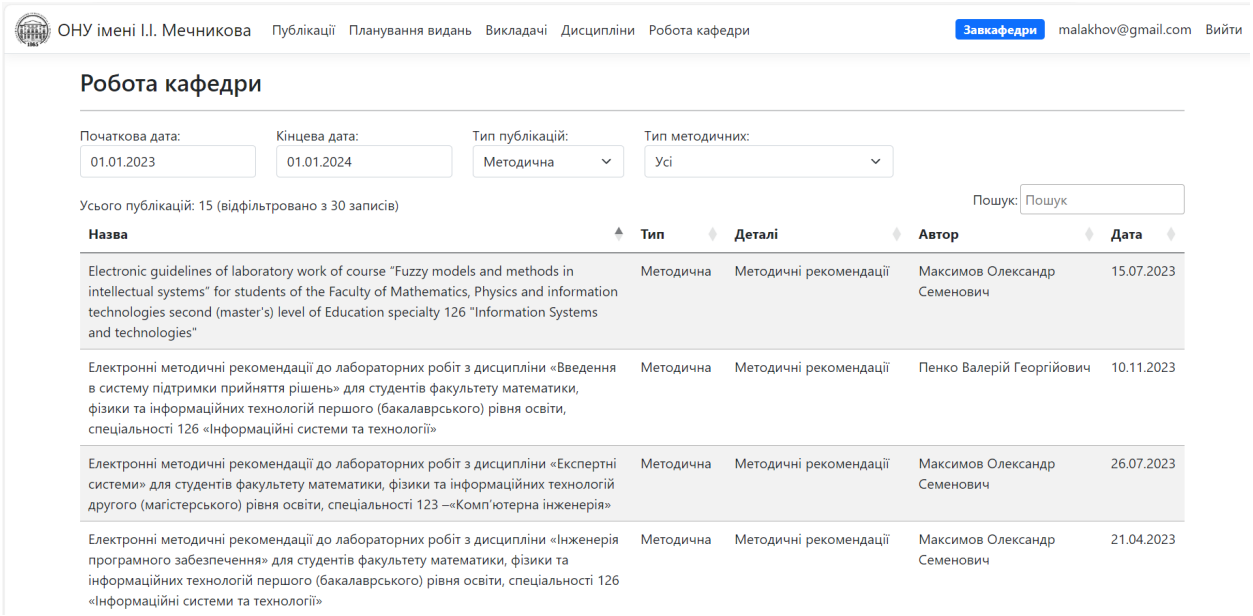


Рисунок 8.18 – Сторінка «Робота кафедри»

8.5 Інтерфейс Гаранта освітньої програми

Гаранту доступний пункт меню «Освітня програма», де кожна програма закріплена за кожним гарантом (див. рис. 8.19). Таким чином, на сторінці перераховано дисципліни, котрі викладаються на навчальній програмі та викладачі, котрі викладають ці дисципліни.

Ця сторінка передбачає можливість перевірки відповідності публікацій викладача дисципліні, котру він викладає.

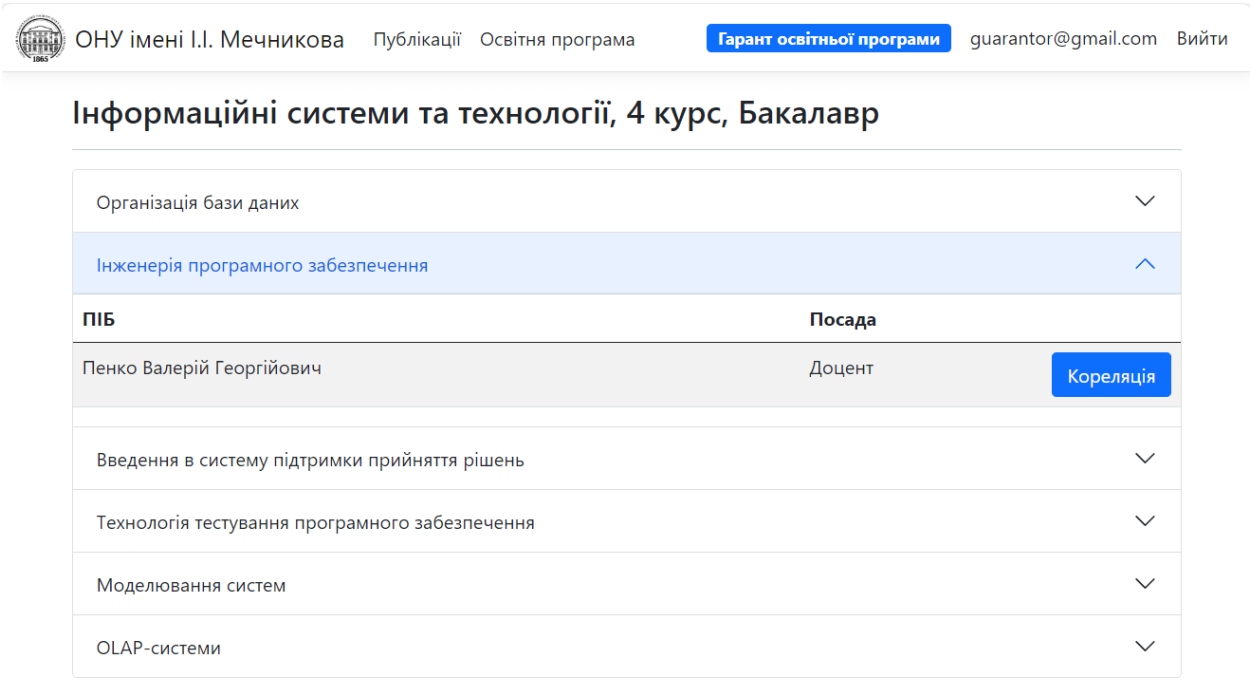


Рисунок 8.19 – Сторінка «Освітня програма»

Таким чином (див. рис. 8.20), гаранту освітньої програми відображаються публікації, котрі стосуються дисципліни викладача. Традиційно, наявні можливості фільтрації за календарним періодом, типом публікацій, сортування за властивостями та здійснення пошукового запиту.

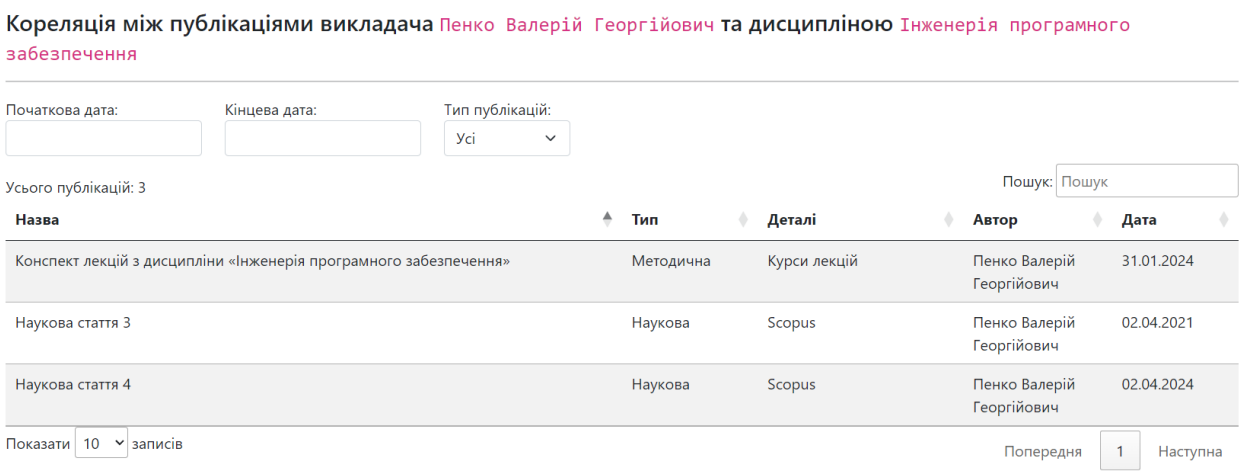


Рисунок 8.20 – Сторінка «Відповідність публікацій викладача дисципліні»

8.6 Інтерфейс Методичного відділу

Методичному відділу доступні пункти меню «Кафедри» та «Планування видань».

Сторінка «Кафедри» передбачає перелік усіх кафедри, котрі є під контролем методичного відділ відповідно із переліком викладачів, закріпленими за цими кафедрами (див. рис. 8.21).

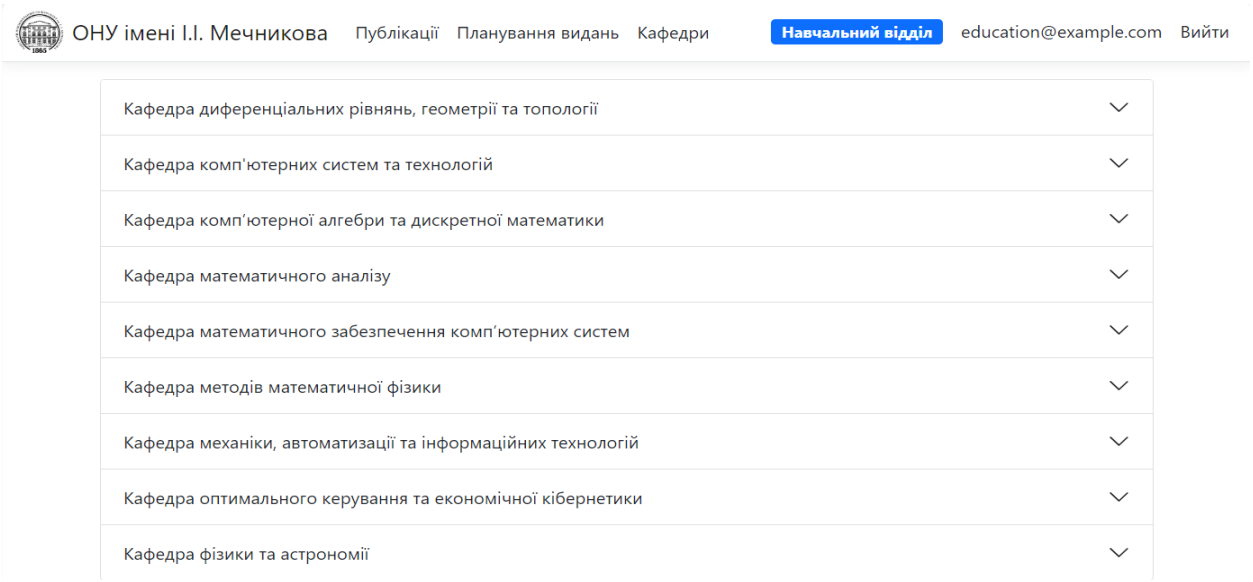


Рисунок 8.21 – Сторінка «Кафедри»

Також Методичний відділ має можливість перевіряти та відслідковувати прогрес виконання планів видань на календарний рік кожної кафедри (див. рис. 8.22).

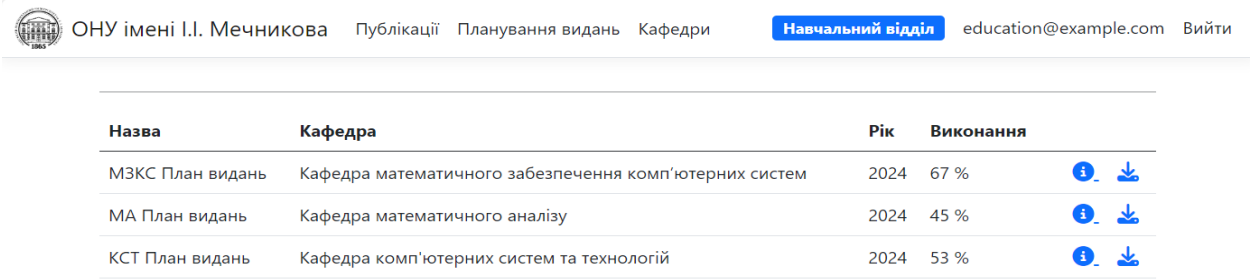


Рисунок 8.22 – Сторінка «Планування видань» для Методичного відділу

ВИСНОВКИ

У результаті аналізу предметної області сформульована мета створення та визначено основні особливості функціонування інформаційної системи підтримки науково-методичної діяльності ЗВО, визначено коло зацікавлених осіб та перелік їх вимог та задач, котрі вони мають вирішувати за допомогою створюваної ІС.

Інформаційну систему реалізовано у форматі веб-застосунку. Система має функціонал для контролю наповненості навчальних дисциплін методичними матеріалами, оцінки ефективності роботи викладачів, планування видань кафедрами, контролю відповідності публікацій викладача його дисципліні, а також перевірки виконання викладачем деяких умов ліцензування.

Завдяки використанню трирівневої архітектури та паттерну проєктування MVC досягнуто незалежність компонентів та висока масштабованість системи. Також важливо зазначити, що дана система розроблена на базі REST API, що надає можливість її інтеграції з іншими подібними інформаційними системами та забезпечує стандартизований обмін даними між ними.

Для гарантування безпеки та контролю доступу до даних використовується система ролей та привілеїв. Механізм аутентифікації та авторизації забезпечує необхідний рівень захисту від несанкціонованого доступу до інформації.

В порівнянні з аналогами, створена система не тільки виявилася більш доступною для ОНУ імені І. І. Мечникова, але й надає саме той функціонал, який був запитаний співробітниками університету. Створення веб-застосунку дозволило зменшити витрати часу на аудит та контроль діяльності кафедри, а також зменшити кількість помилок. Це дозволяє співробітникам кафедри

краще використовувати свій час і поліпшити навчальний процес в університеті.

Створена ІС є добре структурованою, а отже в ній можливе нарощування функціоналу користувачів вже наявних категорій, а також додавання нових користувачів. Корисним буде втілення функціональності для динамічної генерації документів та допомоги у створенні публікацій (наприклад, шаблони робочих програм).

Роботу було апробовано на XX Всеукраїнській конференції студентів і молодих науковців «Інформатика, інформаційні системи та технології».

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Automation of Production Activities of an Industrial Enterprise based on the ERP System / N. V. Syreyshchikova et al. Procedia Manufacturing. 2020. Vol. 46. P. 525–532. URL: https://doi.org/10.1016/j.promfg.2020.03.075.

2. АСУ навчальним процесом «Директива» – розробник ТОВ «Комп'ютерні інформаційні технології» [Електронний ресурс] — Режим доступу: http://www.kitsoft.com.ua

3. «Автоматизована система управління вищим навчальним закладом III – IV рівня акредитації» модуль «Навчальна частина – 01» АСУ «Університет». Компанія Юнітех+ [Електронний ресурс] – Режим доступу: https://www.unitex.com.ua/products/commercial-software/automated-system-for-higher-education-institution/

4. Адміністративно навчальна інформаційна система інституту комп'ютерних технологій Національного авіаційного університету України [Електронний ресурс] – Режим доступу: http://surl.li/spjjs

5. Пакет комп'ютерних систем ПП «Політек-софт» [Електронний ресурс] – Режим доступу: http://www.politek-soft.kiev.ua

6. Програма автоматизації ВНЗ АЛЬМА-МАТЕР [Електронний ресурс] – Режим доступу: http://www.alma-mater.org.ua

7. Науково-Дослідний Інститут Прикладних Інформаційних технологій «АСУ ВНЗ» [Електронний ресурс] – Режим доступу: https://vuz.osvita.net/

8. Гриценко В. Г. Аналіз сучасного стану використання інформаційно-комунікаційних технологій в управлінні вищим навчальним закладом / В. Г. Гриценко // Збірник наукових праць Кам'янець-Подільського національного університету імені Івана Огієнка. Серія : Педагогічна. - 2014. - Вип. 20. - С. 256-259. [Електронний ресурс]. – Режим доступу: http://ped-series.kpnu.edu.ua/article/view/36898/33116

9. Sidhu A. Role of Digitalization in Higher Education: Looking Through the Lens of Opportunities. Digitalization of Higher Education. New York, 2023. P. 1–20. URL: https://doi.org/10.1201/9781003412151-1

10. Вячеслав Труба. Положення про організацію освітнього процесу в Одеському національному університеті імені І. І. Мечникова — Одеса, 2022.

11. Про затвердження Ліцензійних умов провадження освітньої діяльності: Постанова Кабінету Міністрів України від 30.12.2015р. № 1187. [Електронний ресурс]. – Режим доступу: https://zakon.rada.gov.ua/laws/show/1187-2015-п#Text

12. Малахов Є.В., Основи проектування БД: Конспект лекцій. [Електронний ресурс] – Режим доступу: http://pub.onu.edu.ua/images/imem/kaf_mat_zabezpech/navch-metod-material/Orhanizatsiia_baz_danykh.zip

ДОДАТОК А

Задачі користувачів ІС

Таблиця А.1 – Список задач користувачів ІС підтримки науково-методичної діяльності ЗВО

Номер задачі	**Задача**	**Вхідні дані для вирішення задачі**	**Вихідні дані / Результат виконання**
Студент			
С1	Переглянути доступні публікації за критеріями	Критерії: пошуковий запит, ПІБ автора, початкова дата, кінцева дата, назва дисципліни	Набір публікацій: назва, рік видання, автори, короткий опис, ключові слова
С2	Переглянути деталі публікації	Назва	Дата видання, автори, анотація, опис, ключові слова
С3	Завантажити публікацію	Назва	Файл у форматі pdf
С4	Переглянути бібліографію автора	ПІБ автора	Набір публікацій: назва, рік видання, автори, короткий опис, ключові слова
С5	Перегляд статистики публікацій автора за різні роки	ПІБ автора	Діаграма динаміки публікацій за роками

Продовження таблиці А.1

Номер задачі	**Задача**	**Вхідні дані для вирішення задачі**	**Вихідні дані / Результат виконання**
Викладач			
В1	Створити нову публікацію	Назва, дата, анотація, опис, ключові слова, автори, мова, тип (підручник, курс лекцій, практикум тощо), файл формату pdf	Нова публікація
В2	Редагувати свою публікацію	Назва, анотація, опис, ключові слова, автори, мова, тип (підручник, курс лекцій, практикум тощо)	Відредагована публікація
В3	Видалити свою публікацію	Назва публікації	Публікація видалена
В4	Переглянути плани видань кафедри	-	Назва плану, рік
В5	Переглянути заплановані видання	Назва плану	Заплановані публікації власного авторства: назва, тип, укладачі, обсяг, мова

Продовження таблиці А.1

Номер задачі	**Задача**	**Вхідні дані для вирішення задачі**	**Вихідні дані / Результат виконання**
В6	Виконати план	Назва плану, назва запланованої публікації, опис, анотація, ключові слова, файл формату pdf	Нова публікація відповідно до плану
Завідувач кафедри			
ЗК1	Переглянути плани видань власної кафедри	-	Назва плану, рік, прогрес (відсоток виконання)
ЗК2	Переглянути деталі виконання плану	Назва плану	Заплановані публікації на кафедрі: назва, тип, укладачі, обсяг, мова, виконання (так / ні)
ЗК3	Створити новий план видань	Назва плану, рік, публікації: назва видання, укладачі, об’єм (в авт. арк.), мова видання, тип видання	Новий план видань
ЗК4	Змінювати деталі видань у плані видань	Назва плану, назва видання, деталі видання: укладачі, об’єм (в авт. арк.), мова видання, тип	Відредагований план

Продовження таблиці А.1

Номер задачі	**Задача**	**Вхідні дані для вирішення задачі**	**Вихідні дані / Результат виконання**
ЗК5	Завантажити план видань	Назва плану	Документ формату pdf згідно стандарту
ЗК6	Перевірити наповненість дисципліни методичними матеріалами	Назва дисципліни	Перелік методичних матеріалів, відповідних цій дисципліні
ЗК7	Звітність стосовно ефективності роботи викладача	ПІБ викладача, критерії: діапазон років (1 рік, 5 років, увесь час, власний діапазон)	Кількість та перелік публікацій
ЗК8	Перевірка ліцензійних вимог викладача, котрі стосуються наукових та методичних публікацій	ПІБ викладача	Перелік бібліографічних посилань на публікації, що регламентовані вимогами
ЗК9	Звітність стосовно роботи кафедри	Критерії: початкова та кінцева дати, тип публікацій (наукова / методична, вид)	Перелік публікацій: назва, автор, тип, додаткові відомості, дата завантаження

Продовження таблиці А.1

Номер задачі	**Задача**	**Вхідні дані для вирішення задачі**	**Вихідні дані / Результат виконання**
Навчальний відділ			
НВ1	Переглянути плани видань всіх кафедр	-	Назва плану, назва кафедри, рік, прогрес (відсоток виконання)
НВ2	Переглянути деталі виконання плану	Назва плану	Заплановані публікації на кафедрі: назва, тип, укладачі, обсяг, мова, виконання (так / ні)
НВ3	Завантажити план видань	Назва плану	Документ формату pdf згідно стандарту
НВ4	Переглядати статистику виконання планів видань всіх кафедр	Рік	Звіт або графік статистики виконання планів видань
НВ5	Перевірка ліцензійних вимог викладачів	ПІБ викладача, кафедра	Перелік бібліографічних посилань на публікації
Гарант освітньої програми			
Г1	Перевірити наповненість дисципліни методичними матеріалами	Назва дисципліни	Перелік методичних матеріалів, що відповідають цій дисципліні

Продовження таблиці А.1

Номер задачі	**Задача**	**Вхідні дані для вирішення задачі**	**Вихідні дані / Результат виконання**
Г2	Демонстрація відповідності викладача дисципліні, що він викладає, за його публікаціями	ПІБ викладача, назва дисципліни	Перелік методичних та наукових публікацій викладача, пов’язаних із його дисципліною

ДОДАТОК Б

Опис сутностей предметної області

Таблиця Б.1 – Опис сутностей предметної області ІС підтримки науково-методичної діяльності ЗВО

Ім’я атрибуту	Призначення атрибуту	Обмеження
Person		
Id	ідентифікатор особи	первинний ключ
FirstName	ім’я особи	не порожнє, менше 50 символів
LastName	прізвище особи	не порожнє, менше 50 символів
MiddleName	по-батькові особи	не порожнє, менше 50 символів
ApplicationUserId	ідентифікатор користувача	може бути порожнє, зовнішній ключ для зв’язку із сутністю ApplicationUser (Id)
Publications		
PublicationId	ідентифікатор публікації	первинний ключ
Title	заголовок публікації	не порожнє, менше 256 символів
PublicationDate	дата публікації	може бути порожнє
Abstract	анотація	може бути порожнє, менше 4096 символів
Description	опис	може бути порожнє, менше 4096 символів
Keywords	ключові слова	може бути порожнє
isPublished	чи опубліковано	не порожнє
Volume	обсяг	може бути порожнє, менше 4096
Language	мова	може бути порожнє

Продовження Таблиці Б.1

Ім’я атрибуту	Призначення атрибуту	Обмеження
PersonPublication		
AuthorsId	ідентифікатор автора	складений первинний ключ, зовнішній ключ для зв’язку із сутністю Person (Id)
PublicationsPublicationId	ідентифікатор публікації	складений первинний ключ, зовнішній ключ для зв’язку із сутністю Publications (PublicationId)
Lecturers		
Id	ідентифікатор викладача	первинний ключ, зовнішній ключ для зв’язку із сутністю Person (Id)
AcademicTitle	посада	не порожнє
Departments		
DepartmentId	ідентифікатор кафедри	первинний ключ
Name	назва	не порожнє, менше 256 символів
DepartmentLecturer		
DepartmentsDepartmentId	ідентифікатор кафедри	складений первинний ключ, зовнішній ключ для зв’язку із сутністю Departments (DepartmentId)
LecturersId	ідентифікатор викладача	складений первинний ключ, зовнішній ключ для зв’язку із сутністю Lecturers (Id)

Продовження Таблиці Б.1

Ім’я атрибуту	Призначення атрибуту	Обмеження
Disciplines		
Id	ідентифікатор дисципліни	первинний ключ
Name	назва	не порожнє, менше 256 символів
EducationYear		
Id	ідентифікатор	первинний ключ
StartDate	початок навчального року	не порожнє
EndDate	кінець навчального року	не порожнє
DisciplinePublication		
DisciplinesId	ідентифікатор дисципліни	складений первинний ключ, зовнішній ключ для зв’язку із сутністю Disciplines (Id)
PublicationsPublicationId	ідентифікатор публікації	складений первинний ключ, зовнішній ключ для зв’язку із сутністю Publications (PublicationId)
DisciplineWorkPlan		
DisciplinesId	ідентифікатор дисципліни	складений первинний ключ, зовнішній ключ для зв’язку із сутністю Disciplines (Id)
WorkPlanId	ідентифікатор робочого плану	складений первинний ключ, зовнішній ключ для зв’язку із сутністю WorkPlan (Id)

Продовження Таблиці Б.1

Ім’я атрибуту	Призначення атрибуту	Обмеження
HeadsOfDepartments		
Id	ідентифікатор завідувача кафедри	первинний ключ, зовнішній ключ для зв’язку із сутністю Person (Id)
DepartmentId	ідентифікатор кафедри	зовнішній ключ для зв’язку із сутністю Department (Id)
LecturerDisciplines		
LecturerId	ідентифікатор викладача	складений первинний ключ, зовнішній ключ для зв’язку із сутністю Lecturer (Id)
DisciplineId	ідентифікатор дисципліни	складений первинний ключ, зовнішній ключ для зв’язку із сутністю Discipline (Id)
BeginDate	початок викладання	не порожнє, значення більше за 01.01.1970
EndDate	кінець викладання	може бути порожнім, значення більше за дату початку викладання
MethodologicalPublications		
PublicationId	Ідентифікатор публікації	первинний ключ, зовнішній ключ для зв’язку із сутністю Publication (Id)
Type	тип методичної публікації	не порожнє
CloudStorageGuid	посилання на публікацію	може бути порожнє

Продовження Таблиці Б.1

Ім’я атрибуту	Призначення атрибуту	Обмеження
PlanId	ідентифікатор плану видань	зовнішній ключ для зв’язку із сутністю Plans (Id)
Plans		
PlanId	ідентифікатор плану видань	первинний ключ
Title	назва плану	не порожнє, менше 256 символів
Year	календарний рік	не порожнє, значення більше за 01.01.1970
DepartmentId	ідентифікатор кафедри	зовнішній ключ для зв’язку із сутністю Department (Id)
ScientificArticles		
PublicationId	ідентифікатор публікації	первинний ключ, зовнішній ключ для зв’язку із сутністю Publication (Id)
JournalType	тип наукового журналу	не порожнє
JournalDetails	деталі публікації в журналі	не порожнє, менше 512 символів
ScientificConferenceTheses		
PublicationId	ідентифікатор публікації	первинний ключ, зовнішній ключ для зв’язку із сутністю Publication (Id)
ConferenceName	назва наукової конференції	не порожнє, менше 256 символів
ConferencePlace	місце проведення наукової конференції	не порожнє, менше 512 символів

Продовження Таблиці Б.1

Ім’я атрибуту	Призначення атрибуту	Обмеження
ScientificDissertations		
PublicationId	ідентифікатор публікації	первинний ключ, зовнішній ключ для зв’язку із сутністю Publication (Id)
EducationalInstitution	назва закладу, у якому захищено	не порожнє, менше 512 символів
DissertationType	тип дисертації	не порожнє
ScientificMonographs		
PublicationId	ідентифікатор публікації	первинний ключ, зовнішній ключ для зв’язку із сутністю Publication (Id)
Publisher	назва видання	не порожнє
ISBN	код ISBN	не порожнє, менше 17 символів
ScientificPatents		
PublicationId	ідентифікатор публікації	первинний ключ, зовнішній ключ для зв’язку із сутністю Publication (Id)
PatentNo	номер патенту	не порожнє, менше 12 символів
Issuer	видавець патенту	не порожнє. менше 512 символів
ScientificPublications		
PublicationId	ідентифікатор публікації	первинний ключ, зовнішній ключ для зв’язку із сутністю Publication (Id)

Продовження Таблиці Б.1

Ім’я атрибуту	Призначення атрибуту	Обмеження
UDC	УДК публікації	не порожнє, значення менше 12 символів
DOI	DOI публікації	не порожнє, значення менше 128 символів
URL	посилання на публікацію	не порожнє, значення менше 256 символів
Specialities		
Id	ідентифікатор спеціальності	первинний ключ
Name	назва спеціальності	не порожнє, значення менше 256 символів
WorkPlans		
Id	ідентифікатор робочого плану	первинний ключ
SpecialityId	ідентифікатор спеціальності	зовнішній ключ для зв’язку із сутністю Specialities (Id)
DepartmentId	ідентифікатор кафедри	зовнішній ключ для зв’язку із сутністю Departments (DepartmentId)
EducationYearId	ідентифікатор навчального року	Зовнішній ключ для зв’язку із сутністю EducationYear (Id)
Course	курс	не порожнє, значення в проміжку [1, 9]
PreparationLevel	рівень підготовки	не порожнє
GuarantorId	ідентифікатор гаранта освітньої програми	зовнішній ключ для зв’язку із сутністю AspNetUsers (Id)

ДОДАТОК В

Запити на створення таблиць бази даних

```
CREATE TABLE public."Person" (
     "Id" int4 GENERATED BY DEFAULT AS IDENTITY( INCREMENT BY 1
MINVALUE 1 MAXVALUE 2147483647 START 1 CACHE 1 NO CYCLE) NOT
NULL,
     "FirstName" VARCHAR(50) NOT NULL,
     "LastName" VARCHAR(50) NOT NULL,
     "MiddleName" VARCHAR(50) NOT NULL,
     "ApplicationUserId" text NULL,
     CONSTRAINT "PK_Person" PRIMARY KEY ("Id"),
     CONSTRAINT "FK_Person_AspNetUsers_ApplicationUserId" FOREIGN
KEY ("ApplicationUserId") REFERENCES public."AspNetUsers"("Id")
);
```

Лістинг В.1 – Створення людини

```
CREATE TABLE public."Publications" (
     "PublicationId" int4 GENERATED BY DEFAULT AS
IDENTITY( INCREMENT BY 1 MINVALUE 1 MAXVALUE 2147483647 START 1
CACHE 1 NO CYCLE) NOT NULL,
     "Title" VARCHAR(256) NOT NULL,
     "PublicationDate" timestamptz NULL,
     "Abstract" text NULL,
     "Description" text NULL,
     "Keywords" _text NULL,
     "isPublished" bool NOT NULL,
     "Volume" int4 NULL CHECK ("Volume" < 4096 AND "Volume" > 0),
     "Language" int4 NULL,
     CONSTRAINT "PK_Publications" PRIMARY KEY ("PublicationId"),
     CHECK (CHAR_LENGTH("Abstract") <= 4096),
     CHECK (CHAR_LENGTH("Description") <= 4096)
);
```

Лістинг В.2 – Створення публікацій

```
CREATE TABLE public."PersonPublication" (
     "AuthorsId" int4 NOT NULL,
     "PublicationsPublicationId" int4 NOT NULL,
     CONSTRAINT "PK_PersonPublication" PRIMARY KEY ("AuthorsId",
"PublicationsPublicationId"),
     CONSTRAINT "FK_PersonPublication_Person_AuthorsId" FOREIGN
KEY ("AuthorsId") REFERENCES public."Person"("Id") ON DELETE
CASCADE,
     CONSTRAINT
"FK_PersonPublication_Publications_PublicationsPublicationId"
FOREIGN KEY ("PublicationsPublicationId") REFERENCES
public."Publications"("PublicationId") ON DELETE CASCADE
);
CREATE INDEX "IX_PersonPublication_PublicationsPublicationId" ON
public."PersonPublication" USING btree
("PublicationsPublicationId");
```

Лістинг В.3 – Створення зв’язку багато до багатьох

між людьми та публікаціями

```
CREATE TABLE public."Lecturers" (
     "Id" int4 NOT NULL,
     "AcademicTitle" int4 NOT NULL,
     CONSTRAINT "PK_Lecturers" PRIMARY KEY ("Id"),
     CONSTRAINT "FK_Lecturers_Person_Id" FOREIGN KEY ("Id")
REFERENCES public."Person"("Id") ON DELETE CASCADE
);
```

Лістинг В.4 – Створення викладачів

```
CREATE TABLE public."Departments" (
     "DepartmentId" int4 GENERATED BY DEFAULT AS
IDENTITY( INCREMENT BY 1 MINVALUE 1 MAXVALUE 2147483647 START 1
CACHE 1 NO CYCLE) NOT NULL,
     "Name" VARCHAR(256) NOT NULL,
     CONSTRAINT "PK_Departments" PRIMARY KEY ("DepartmentId")
);
```

Лістинг В.5 – Створення кафедр

```
CREATE TABLE public."DepartmentLecturer" (
     "DepartmentsDepartmentId" int4 NOT NULL,
     "LecturersId" int4 NOT NULL,
     CONSTRAINT "PK_DepartmentLecturer" PRIMARY KEY
("DepartmentsDepartmentId", "LecturersId"),
     CONSTRAINT
"FK_DepartmentLecturer_Departments_DepartmentsDepartmentId"
FOREIGN KEY ("DepartmentsDepartmentId") REFERENCES
public."Departments"("DepartmentId") ON DELETE CASCADE,
     CONSTRAINT "FK_DepartmentLecturer_Lecturers_LecturersId"
FOREIGN KEY ("LecturersId") REFERENCES public."Lecturers"("Id")
ON DELETE CASCADE
);
CREATE INDEX "IX_DepartmentLecturer_LecturersId" ON
public."DepartmentLecturer" USING btree ("LecturersId");
```

Лістинг В.6 – Створення зв’язку багато-до-багатьох
між викладачами та кафедрами

```
CREATE TABLE public."Disciplines" (
     "Id" int4 GENERATED BY DEFAULT AS IDENTITY( INCREMENT BY 1
MINVALUE 1 MAXVALUE 2147483647 START 1 CACHE 1 NO CYCLE) NOT
NULL,
     "Name" VARCHAR(256) NOT NULL,
     CONSTRAINT "PK_Disciplines" PRIMARY KEY ("Id")
);
```

Лістинг В.7 – Створення дисциплін

```
CREATE TABLE public."EducationYear" (
     "Id" int4 GENERATED BY DEFAULT AS IDENTITY( INCREMENT BY 1
MINVALUE 1 MAXVALUE 2147483647 START 1 CACHE 1 NO CYCLE) NOT
NULL,
     "StartDate" date NOT NULL,
     "EndDate" date NOT NULL,
     CONSTRAINT "PK_EducationYear" PRIMARY KEY ("Id")
);
```

Лістинг В.8 – Створення навчального року

```
CREATE TABLE public."DisciplinePublication" (
     "DisciplinesId" int4 NOT NULL,
     "PublicationsPublicationId" int4 NOT NULL,
     CONSTRAINT "PK_DisciplinePublication" PRIMARY KEY
("DisciplinesId", "PublicationsPublicationId"),
     CONSTRAINT
"FK_DisciplinePublication_Disciplines_DisciplinesId" FOREIGN KEY
("DisciplinesId") REFERENCES public."Disciplines"("Id") ON
DELETE CASCADE,
     CONSTRAINT
"FK_DisciplinePublication_Publications_PublicationsPublicationId
" FOREIGN KEY ("PublicationsPublicationId") REFERENCES
public."Publications"("PublicationId") ON DELETE CASCADE
);
CREATE INDEX
"IX_DisciplinePublication_PublicationsPublicationId" ON
public."DisciplinePublication" USING btree
("PublicationsPublicationId");
```

Лістинг В.9 – Створення зв’язку багато-до-багатьох
між дисциплінами та публікаціями

```
CREATE TABLE public."DisciplineWorkPlan" (
     "DisciplinesId" int4 NOT NULL,
     "WorkPlanId" int4 NOT NULL,
     CONSTRAINT "PK_DisciplineWorkPlan" PRIMARY KEY
("DisciplinesId", "WorkPlanId"),
     CONSTRAINT "FK_DisciplineWorkPlan_Disciplines_DisciplinesId"
FOREIGN KEY ("DisciplinesId") REFERENCES
public."Disciplines"("Id") ON DELETE CASCADE,
     CONSTRAINT "FK_DisciplineWorkPlan_WorkPlans_WorkPlanId"
FOREIGN KEY ("WorkPlanId") REFERENCES public."WorkPlans"("Id")
ON DELETE CASCADE
);
CREATE INDEX "IX_DisciplineWorkPlan_WorkPlanId" ON
public."DisciplineWorkPlan" USING btree ("WorkPlanId");
```

Лістинг В.10 – Створення зв’язку багато-до-багатьох
між дисциплінами та робочими планами

```
CREATE TABLE public."HeadsOfDepartments" (
     "Id" int4 NOT NULL,
     "DepartmentId" int4 NOT NULL,
     CONSTRAINT "PK_HeadsOfDepartments" PRIMARY KEY ("Id"),
     CONSTRAINT "FK_HeadsOfDepartments_Departments_DepartmentId"
FOREIGN KEY ("DepartmentId") REFERENCES
public."Departments"("DepartmentId") ON DELETE CASCADE,
     CONSTRAINT "FK_HeadsOfDepartments_Person_Id" FOREIGN KEY
("Id") REFERENCES public."Person"("Id") ON DELETE CASCADE
);
CREATE UNIQUE INDEX "IX_HeadsOfDepartments_DepartmentId" ON
public."HeadsOfDepartments" USING btree ("DepartmentId");
```

Лістинг В.11 – Створення завідувачів кафедри

```
CREATE TABLE public."LecturerDisciplines" (
     "LecturerId" int4 NOT NULL,
     "DisciplineId" int4 NOT NULL,
     "BeginDate" date NOT NULL,
     "EndDate" date NULL,
     CONSTRAINT "PK_LecturerDisciplines" PRIMARY KEY
("LecturerId", "DisciplineId"),
     CONSTRAINT "FK_LecturerDisciplines_Disciplines_DisciplineId"
FOREIGN KEY ("DisciplineId") REFERENCES
public."Disciplines"("Id") ON DELETE CASCADE,
     CONSTRAINT "FK_LecturerDisciplines_Lecturers_LecturerId"
FOREIGN KEY ("LecturerId") REFERENCES public."Lecturers"("Id")
ON DELETE CASCADE
);
CREATE INDEX "IX_LecturerDisciplines_DisciplineId" ON
public."LecturerDisciplines" USING btree ("DisciplineId");
```

Лістинг В.12 – Створення зв’язку багато-до-багатьох
між дисциплінами та викладачами

```
CREATE TABLE public."MethodologicalPublications" (
     "PublicationId" int4 NOT NULL,
     "Type" int4 NULL,
     "CloudStorageGuid" text NULL,
     "PlanId" int4 NULL,
     CONSTRAINT "PK_MethodologicalPublications" PRIMARY KEY
("PublicationId"),
     CONSTRAINT "FK_MethodologicalPublications_Plans_PlanId"
FOREIGN KEY ("PlanId") REFERENCES public."Plans"("PlanId") ON
DELETE SET NULL,
     CONSTRAINT
"FK_MethodologicalPublications_Publications_PublicationId"
FOREIGN KEY ("PublicationId") REFERENCES
public."Publications"("PublicationId") ON DELETE CASCADE
);
CREATE INDEX "IX_MethodologicalPublications_PlanId" ON
public."MethodologicalPublications" USING btree ("PlanId");
```

Лістинг В.13 – Створення методичних публікацій

```
CREATE TABLE public."Plans" (
     "PlanId" int4 GENERATED BY DEFAULT AS IDENTITY( INCREMENT BY
1 MINVALUE 1 MAXVALUE 2147483647 START 1 CACHE 1 NO CYCLE) NOT
NULL,
     "Title" text NOT NULL,
     "Year" int4 NOT NULL,
     "DepartmentId" int4 NOT NULL,
     CONSTRAINT "PK_Plans" PRIMARY KEY ("PlanId"),
     CONSTRAINT "FK_Plans_Departments_DepartmentId" FOREIGN KEY
("DepartmentId") REFERENCES public."Departments"("DepartmentId")
ON DELETE CASCADE
);
CREATE INDEX "IX_Plans_DepartmentId" ON public."Plans" USING
btree ("DepartmentId");
```

Лістинг В.14 – Створення планів видань

```
CREATE TABLE public."ScientificArticles" (
     "PublicationId" int4 NOT NULL,
     "JournalType" int4 NOT NULL,
     "JournalDetails" text NOT NULL,
     CONSTRAINT "PK_ScientificArticles" PRIMARY KEY
("PublicationId"),
     CONSTRAINT
"FK_ScientificArticles_ScientificPublications_PublicationId"
FOREIGN KEY ("PublicationId") REFERENCES
public."ScientificPublications"("PublicationId") ON DELETE
CASCADE);
```

Лістинг В.15 – Створення наукових статей

```
CREATE TABLE public."ScientificConferenceTheses" (
     "PublicationId" int4 NOT NULL,
     "ConferenceName" text NOT NULL,
     "ConferencePlace" text NOT NULL,
     CONSTRAINT "PK_ScientificConferenceTheses" PRIMARY KEY
("PublicationId"),
     CONSTRAINT
"FK_ScientificConferenceTheses_ScientificPublications_Publicati~
" FOREIGN KEY ("PublicationId") REFERENCES
public."ScientificPublications"("PublicationId") ON DELETE
CASCADE);
```

Лістинг В.16 – Створення тез наукових конференцій

```
CREATE TABLE public."ScientificDissertations" (
     "PublicationId" int4 NOT NULL,
     "EducationalInstitution" text NOT NULL,
     "DissertationType" int4 NOT NULL,
     CONSTRAINT "PK_ScientificDissertations" PRIMARY KEY
("PublicationId"),
     CONSTRAINT
"FK_ScientificDissertations_ScientificPublications_PublicationId
" FOREIGN KEY ("PublicationId") REFERENCES
public."ScientificPublications"("PublicationId") ON DELETE
CASCADE);
```

Лістинг В.17 – Створення наукових дисертацій

```
CREATE TABLE public."ScientificMonographs" (
     "PublicationId" int4 NOT NULL,
     "Publisher" text NOT NULL,
     "ISBN" text NOT NULL,
     CONSTRAINT "PK_ScientificMonographs" PRIMARY KEY
("PublicationId"),
     CONSTRAINT
"FK_ScientificMonographs_ScientificPublications_PublicationId"
FOREIGN KEY ("PublicationId") REFERENCES
public."ScientificPublications"("PublicationId") ON DELETE
CASCADE);
```

Лістинг В.18 – Створення наукових монографій

```
CREATE TABLE public."ScientificPatents" (
     "PublicationId" int4 NOT NULL,
     "PatentNo" text NOT NULL,
     "Issuer" text NOT NULL,
     CONSTRAINT "PK_ScientificPatents" PRIMARY KEY
("PublicationId"),
     CONSTRAINT
"FK_ScientificPatents_ScientificPublications_PublicationId"
FOREIGN KEY ("PublicationId") REFERENCES
public."ScientificPublications"("PublicationId") ON DELETE
CASCADE);
```

Лістинг В.19 – Створення наукових патентів

```
CREATE TABLE public."ScientificPublications" (
     "PublicationId" int4 NOT NULL,
     "UDC" text NOT NULL, "DOI" text NULL, "URL" text NULL,
     CONSTRAINT "PK_ScientificPublications" PRIMARY KEY
("PublicationId"),
     CONSTRAINT
"FK_ScientificPublications_Publications_PublicationId" FOREIGN
KEY ("PublicationId") REFERENCES
public."Publications"("PublicationId") ON DELETE CASCADE);
```

Лістинг В.20 – Створення наукових публікацій

```
CREATE TABLE public."Specialities" (
     "Id" int4 GENERATED BY DEFAULT AS IDENTITY( INCREMENT BY 1
MINVALUE 1 MAXVALUE 2147483647 START 1 CACHE 1 NO CYCLE) NOT
NULL,
     "Name" text NOT NULL,
     CONSTRAINT "PK_Specialities" PRIMARY KEY ("Id")
);
```

Лістинг В.21 – Створення спеціальностей

```
CREATE TABLE public."WorkPlans" (
     "Id" int4 GENERATED BY DEFAULT AS IDENTITY( INCREMENT BY 1
MINVALUE 1 MAXVALUE 2147483647 START 1 CACHE 1 NO CYCLE) NOT
NULL,
     "SpecialityId" int4 NOT NULL,
     "DepartmentId" int4 NOT NULL,
     "EducationYearId" int4 NOT NULL,
     "Course" int4 NOT NULL,
     "PreparationLevel" int4 NOT NULL,
     "GuarantorId" text DEFAULT ''::text NOT NULL,
     CONSTRAINT "PK_WorkPlans" PRIMARY KEY ("Id"),
     CONSTRAINT "FK_WorkPlans_AspNetUsers_GuarantorId" FOREIGN
KEY ("GuarantorId") REFERENCES public."AspNetUsers"("Id") ON
DELETE CASCADE,
     CONSTRAINT "FK_WorkPlans_Departments_DepartmentId" FOREIGN
KEY ("DepartmentId") REFERENCES
public."Departments"("DepartmentId") ON DELETE CASCADE,
     CONSTRAINT "FK_WorkPlans_EducationYear_EducationYearId"
FOREIGN KEY ("EducationYearId") REFERENCES
public."EducationYear"("Id") ON DELETE CASCADE,
     CONSTRAINT "FK_WorkPlans_Specialities_SpecialityId" FOREIGN
KEY ("SpecialityId") REFERENCES public."Specialities"("Id") ON
DELETE CASCADE
);
```

Лістинг В.22– Створення робочих планів

ДОДАТОК Г

Запити на створення додаткових функцій та тригерів для збереження цілісності ІС та розв'язання задач користувачів

```
CREATE OR REPLACE FUNCTION check_publication_fields()
RETURNS TRIGGER AS $$
BEGIN
    -- Перевірка, чи встановлюється значення isPublished у TRUE
і чи є NULL у важливих полях
    IF NEW."isPublished" = TRUE THEN
        IF NEW."PublicationDate" IS NULL OR
           NEW."Abstract" IS NULL OR
           NEW."Description" IS NULL OR
           NEW."Keywords" IS NULL OR
           NEW."Volume" IS NULL OR
           NEW."Language" IS NULL THEN
            RAISE EXCEPTION 'Cannot set isPublished to TRUE when
important fields are NULL';
        END IF;
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

Лістинг Г.1 – Створення тригерної функції для перевірки заповнення усіх полів в опублікованому видання

```
CREATE TRIGGER prevent_null_fields_on_publish
BEFORE UPDATE ON "Publications"
FOR EACH ROW
EXECUTE FUNCTION check_publication_fields();
```

Лістинг Г.2 – Створення тригеру на основі функції check_publication_fields

ДОДАТОК Д

Запити на створення представлень для вирішення задач користувачів

```
CREATE VIEW PublishedPublicationsWithAuthors AS
SELECT
    p."PublicationId",
    p."Title",
    p."PublicationDate",
    p."Description",
    p."Keywords",
    pe."Id" AS "AuthorId",
    pe."ApplicationUserId",
    pe."FirstName",
    pe."LastName",
    pe."MiddleName",
    pe."DepartmentId",
    pe."AcademicTitle"
FROM
    "Publications" AS p
LEFT JOIN "PersonPublication" AS pp ON p."PublicationId" =
pp."PublicationsPublicationId"
LEFT JOIN "Person" AS pe ON pp."AuthorsId" = pe."Id"
WHERE
    p."isPublished"
ORDER BY
    p."PublicationDate" DESC;
```

Лістинг Д.1 – Представлення для короткого перегляду публікацій

ДОДАТОК Е

Запити на створення функцій та збережених процедур для вирішення задач користувачів

```
CREATE OR REPLACE FUNCTION GetPublicationInfo(publication_id
INT)
RETURNS TABLE (
    PublicationId INT, Title TEXT,
    PublicationDate TIMESTAMP, Description TEXT,
    Keywords TEXT, AuthorId INT,
    ApplicationUserId INT, FirstName TEXT,
    LastName TEXT, MiddleName TEXT,
    DepartmentId INT, AcademicTitle TEXT,
    CloudStorageGuid TEXT, Language INT, Volume INT
) AS $$
BEGIN
    RETURN QUERY
    SELECT
        p."PublicationId", p."Title",
        p."PublicationDate", p."Description",
        p."Keywords", pe."Id" AS "AuthorId",
        pe."ApplicationUserId", pe."FirstName",
        pe."LastName", pe."MiddleName",
        pe."DepartmentId", pe."AcademicTitle",
        mp."CloudStorageGuid", mp."Language",
        mp."Volume"
    FROM
        "Publications" AS p
    LEFT JOIN "PersonPublication" AS pp ON p."PublicationId" =
pp."PublicationsPublicationId"
    LEFT JOIN "Person" AS pe ON pp."AuthorsId" = pe."Id"
    LEFT JOIN "MethodologicalPublications" AS mp ON
p."PublicationId" = mp."PublicationId"
    WHERE
        p."PublicationId" = publication_id AND
        p."isPublished";
END;
$$ LANGUAGE plpgsql;
```

Лістинг Е.1 – Функція для отримання деталей публікації

```
CREATE OR REPLACE FUNCTION GetPublicationsByAuthorId(author_id
INT)
RETURNS TABLE (
    PublicationId INT, Title TEXT,
    PublicationDate TIMESTAMP, Description TEXT,
    Keywords TEXT, AuthorId INT,
    ApplicationUserId INT, FirstName TEXT,
    LastName TEXT, MiddleName TEXT,
    DepartmentId INT, AcademicTitle TEXT,
    CloudStorageGuid TEXT, Language  INT, Volume  INT
) AS $$
BEGIN
    RETURN QUERY
    SELECT
        p."PublicationId", p."Title",
        p."PublicationDate", p."Description",
        p."Keywords", pe."Id" AS "AuthorId",
        pe."ApplicationUserId", pe."FirstName",
        pe."LastName", pe."MiddleName",
        pe."DepartmentId", pe."AcademicTitle",
      mp."CloudStorageGuid", mp."Language",
      mp."Volume"
    FROM
        "Publications" AS p
    LEFT JOIN "PersonPublication" AS pp ON p."PublicationId" =
pp."PublicationsPublicationId"
    LEFT JOIN "Person" AS pe ON pp."AuthorsId" = pe."Id"
    LEFT JOIN "MethodologicalPublications" AS mp ON p."Id" =
mp."Id"
    WHERE
        pe."Id" = author_id AND
      p."isPublished";
END;
$$ LANGUAGE plpgsql;
```

Лістинг Е.2 – Функція для перегляду публікацій автора

```
CREATE OR REPLACE PROCEDURE AddMethodologicalPublication(
    IN p_title TEXT,
    IN p_publication_date TIMESTAMPTZ,
    IN p_abstract TEXT,
    IN p_description TEXT,
    IN p_keywords _TEXT,
    IN p_is_published BOOL,
    IN p_volume INT,
    IN p_language INT,
    IN p_type INT,
    IN p_cloud_storage_guid TEXT,
    IN p_plan_id INT,
    IN p_disciplines_ids INT[],
    OUT p_publication_id INT
)
LANGUAGE plpgsql
AS $$
DECLARE
    discipline_id INT;
BEGIN
    INSERT INTO "Publications" ("Title", "PublicationDate",
"Abstract", "Description", "Keywords", "isPublished", "Volume",
"Language")
    VALUES (p_title, p_publication_date, p_abstract,
p_description, p_keywords, p_is_published, p_volume, p_language)
    RETURNING "PublicationId" INTO p_publication_id;

    INSERT INTO "MethodologicalPublications" ("PublicationId",
"Type", "CloudStorageGuid", "PlanId")
    VALUES (p_publication_id, p_type, p_cloud_storage_guid,
p_plan_id);

    FOREACH discipline_id IN ARRAY p_disciplines_ids
    LOOP
        INSERT INTO "DisciplinePublication" ("DisciplinesId",
"PublicationsPublicationId")
        VALUES (discipline_id, p_publication_id);
    END LOOP;
END;
$$;
```

Лістинг Е.3 – Збережена процедура для додавання методичної публікації в систему

```
CREATE OR REPLACE PROCEDURE UpdateMethodologicalPublication(
    p_publication_id INT, p_title TEXT,
    p_publication_date TIMESTAMPTZ, p_abstract TEXT,
    p_description TEXT, p_keywords TEXT[],
    p_is_published BOOLEAN, p_volume INT, p_language INT,
    p_type INT, p_cloud_storage_guid TEXT, p_plan_id INT,
    p_disciplines_ids INT[]
)
LANGUAGE plpgsql
AS $$
BEGIN
    -- Оновлення інформації у таблиці Publications
    UPDATE "Publications"
    SET
        "Title" = COALESCE(p_title, "Title"),
        "PublicationDate" = COALESCE(p_publication_date,
"PublicationDate"),
        "Abstract" = COALESCE(p_abstract, "Abstract"),
        "Description" = COALESCE(p_description, "Description"),
        "Keywords" = COALESCE(p_keywords, "Keywords"),
        "isPublished" = COALESCE(p_is_published, "isPublished"),
        "Volume" = COALESCE(p_volume, "Volume"),
        "Language" = COALESCE(p_language, "Language")
    WHERE
        "PublicationId" = p_publication_id;
    -- Оновлення інформації у таблиці MethodologicalPublications
    UPDATE "MethodologicalPublications"
    SET
        "Type" = COALESCE(p_type, "Type"),
        "CloudStorageGuid" = COALESCE(p_cloud_storage_guid,
"CloudStorageGuid"),
        "PlanId" = COALESCE(p_plan_id, "PlanId")
    WHERE
        "PublicationId" = p_publication_id;
    -- Видалення існуючих записів у таблиці
DisciplinePublication
    DELETE FROM "DisciplinePublication"
    WHERE "PublicationsPublicationId" = p_publication_id;
    -- Додавання нових записів у таблиці DisciplinePublication
    IF p_disciplines_ids IS NOT NULL THEN
        FOREACH d_id IN ARRAY p_disciplines_ids
        LOOP
            INSERT INTO "DisciplinePublication"
("DisciplinesId", "PublicationsPublicationId")
            VALUES (d_id, p_publication_id);
        END LOOP;
    END IF;
END;
$$;
```

Лістинг Е.4 – Збережена процедура для редагування публікації

```
CREATE OR REPLACE PROCEDURE DeletePublication(
    p_publication_id INT
)
LANGUAGE plpgsql
AS $$
BEGIN
    -- Видалення записів у таблиці DisciplinePublication
    DELETE FROM "DisciplinePublication"
    WHERE "PublicationsPublicationId" = p_publication_id;

    -- Видалення запису у таблиці Publications
    DELETE FROM "Publications"
    WHERE "PublicationId" = p_publication_id;
END;
$$;
```

Лістинг Е.5 – Збережена процедура для видалення публікації

```
CREATE OR REPLACE FUNCTION GetDepartmentPlans(department_id INT)
RETURNS TABLE (PlanTitle TEXT, Year INT,CompletionRate NUMERIC)
AS $$
DECLARE
    published_count INT;
    total_count INT;
BEGIN
    -- Кількість опублікованих публікацій
    SELECT COUNT(*)
    INTO published_count
    FROM "Publications" AS p
    WHERE p."PlanId" IN (
        SELECT "PlanId" FROM "Plans" WHERE "DepartmentId" =
department_id
    ) AND p."isPublished";

    -- Загальна кількість публікацій
    SELECT COUNT(*)
    INTO total_count
    FROM "Publications" AS p
    WHERE p."PlanId" IN (
        SELECT "PlanId" FROM "Plans" WHERE "DepartmentId" =
department_id
    );

    -- Розрахунок відсотку виконання плану
    IF total_count > 0 THEN
        CompletionRate := (published_count::NUMERIC /
total_count::NUMERIC) * 100;
    ELSE
        CompletionRate := 0;
    END IF;

    -- Повернення інформації про план
    RETURN QUERY
    SELECT
        p."Title" AS "PlanTitle",
        p."Year",
        CompletionRate
    FROM "Plans" AS p WHERE p."DepartmentId" = department_id;
END;
$$ LANGUAGE plpgsql;
```

Лістинг Е.6 – Функція для перегляду прогресу виконання планів видань

```
CREATE OR REPLACE FUNCTION GetPlanInfo(plan_id INT)
RETURNS TABLE (
    PlanId INT, DepartmentId INT, PlanTitle TEXT, Year INT,
DepartmentName TEXT,
    PublicationId INT, PublicationTitle TEXT, PublicationDate
TIMESTAMPTZ,
    Abstract TEXT, Description TEXT, Keywords TEXT[], Language
INT,
    Volume INT, isPublished BOOLEAN, CloudStorageGuid TEXT, Type
INT,
    AuthorId INT, ApplicationUserId INT, AuthorFirstName TEXT,
    AuthorLastName TEXT, AuthorMiddleName TEXT,
AuthorDepartmentId INT,
    AuthorAcademicTitle TEXT
) AS $$
BEGIN
    RETURN QUERY
    SELECT
        p."PlanId", p."DepartmentId", p."Title" AS "PlanTitle",
p."Year", d."Name" AS "DepartmentName",
        pub."PublicationId", pub."Title" AS "PublicationTitle",
pub."PublicationDate",
        pub."Abstract", pub."Description", pub."Keywords",
pub."Language",
        pub."Volume", pub."isPublished", mp."CloudStorageGuid",
mp."Type",
        pers."AuthorsId" AS "AuthorId", per."ApplicationUserId",
per."FirstName" AS "AuthorFirstName",
        per."LastName" AS "AuthorLastName", per."MiddleName" AS
"AuthorMiddleName",
        per."DepartmentId" AS "AuthorDepartmentId",
per."AcademicTitle" AS "AuthorAcademicTitle"
    FROM
        "Plans" AS p
    INNER JOIN "Departments" AS d ON p."DepartmentId" =
d."DepartmentId"
    LEFT JOIN "MethodologicalPublications" AS mp ON p."PlanId" =
mp."PlanId"
    LEFT JOIN "Publications" AS pub ON mp."PublicationId" =
pub."PublicationId"
    LEFT JOIN "PersonPublication" AS pers ON pub."PublicationId"
= pers."PublicationsPublicationId"
    LEFT JOIN "Person" AS per ON pers."AuthorsId" = per."Id"
    WHERE p."PlanId" = plan_id
    ORDER BY
        p."PlanId", d."DepartmentId", pub."PublicationId",
pers."AuthorsId";
END;
$$ LANGUAGE plpgsql;
```

Лістинг Е.7 – Функція для отримання деталей плану кафедри

```
CREATE OR REPLACE PROCEDURE AddPlan(
    p_title TEXT, p_year INT, p_department_id INT
)
LANGUAGE plpgsql
AS $$
BEGIN
    INSERT INTO "Plans" ("Title", "Year", "DepartmentId")
    VALUES (p_title, p_year, p_department_id);
END;
$$;
```

Лістинг Е.8 – Процедура для створення плану видань на календарний

```
CREATE OR REPLACE PROCEDURE AddPublicationsToPlan(
    p_plan_id INT, p_publication_ids INT[]
)
LANGUAGE plpgsql
AS $$
DECLARE
    publication_id INT;
BEGIN
    -- Перебір кожної публікації та додавання до плану
    FOREACH publication_id IN ARRAY p_publication_ids
    LOOP
        INSERT INTO "MethodologicalPublications"
("PublicationId", "PlanId")
        VALUES (publication_id, p_plan_id);
    END LOOP;
END;
$$;
```

Лістинг Е.9 – Процедура для додавання публікацій до плану видань

```
CREATE OR REPLACE FUNCTION
GetDisciplinePublications(discipline_id INT)
RETURNS TABLE (
  DisciplineId INT, DisciplineName TEXT, DisciplinesId INT,
  PublicationsPublicationId INT, PublicationId INT,
  Abstract TEXT, Description TEXT, Keywords TEXT[],
  Language INT, PublicationDate TIMESTAMPTZ,
  Title TEXT, Volume INT, isPublished BOOLEAN,
  CloudStorageGuid TEXT, PlanId INT, Type INT,
  DOI TEXT, UDC TEXT, URL TEXT, JournalDetails TEXT,
  JournalType TEXT, ConferenceName TEXT,
  ConferencePlace TEXT, DissertationType TEXT,
  EducationalInstitution TEXT, ISBN TEXT, Publisher TEXT,
  Issuer TEXT, PatentNo TEXT, AuthorId INT,
```

```
  ApplicationUserId INT, FirstName TEXT, LastName TEXT,
  MiddleName TEXT, DepartmentId INT, AcademicTitle TEXT
) AS $$
BEGIN
  RETURN QUERY
  SELECT
    d."Id" AS "DisciplineId", d."Name" AS "DisciplineName",
    dp."DisciplinesId", dp."PublicationsPublicationId",
    p."PublicationId", p."Abstract", p."Description",
    p."Keywords", p."Language", p."PublicationDate",
    p."Title", p."Volume", p."isPublished",
    mp."CloudStorageGuid", mp."PlanId", mp."Type",
    sp."DOI", sp."UDC", sp."URL", sa."JournalDetails",
    sa."JournalType", sc."ConferenceName",
    sc."ConferencePlace", sd."DissertationType",
    sd."EducationalInstitution", sm."ISBN",
    sm."Publisher", spat."Issuer", spat."PatentNo",
    pe."Id" AS "AuthorId", pe."ApplicationUserId",
    pe."FirstName", pe."LastName", pe."MiddleName",
    pe."DepartmentId", pe."AcademicTitle"
  FROM "Disciplines" AS d
  JOIN "DisciplinePublication" AS dp ON d."Id" =
dp."DisciplinesId"
  JOIN "Publications" AS p ON dp."PublicationsPublicationId" =
p."PublicationId"
  LEFT JOIN "MethodologicalPublications" AS mp ON
p."PublicationId" = mp."PublicationId"
  LEFT JOIN "ScientificPublications" AS sp ON p."PublicationId"
= sp."PublicationId"
  LEFT JOIN "ScientificArticles" AS sa ON p."PublicationId" =
sa."PublicationId"
  LEFT JOIN "ScientificConferenceTheses" AS sc ON
p."PublicationId" = sc."PublicationId"
  LEFT JOIN "ScientificDissertations" AS sd ON p."PublicationId"
= sd."PublicationId"
  LEFT JOIN "ScientificMonographs" AS sm ON p."PublicationId" =
sm."PublicationId"
  LEFT JOIN "ScientificPatents" AS spat ON p."PublicationId" =
spat."PublicationId"
  LEFT JOIN "PersonPublication" AS pp ON p."PublicationId" =
pp."PublicationsPublicationId"
  LEFT JOIN "Person" AS pe ON pp."AuthorsId" = pe."Id"
  WHERE p."isPublished" = TRUE AND
    d."Id" = discipline_id
END;
$$ LANGUAGE plpgsql;
```

Лістинг Е.10 – Функція для отримання публікацій за дисципліною

```
CREATE OR REPLACE FUNCTION GetAllDepartmentsPlans()
RETURNS TABLE (
    DepartmentId INT, PlanTitle TEXT,
    Year INT, CompletionRate NUMERIC
) AS $$
DECLARE
    department RECORD;
BEGIN
    FOR department IN
        SELECT "DepartmentId" FROM "Departments"
    LOOP
        RETURN QUERY
        SELECT
            department."DepartmentId",
            dp."PlanTitle",
            dp."Year",
            dp."CompletionRate"
        FROM GetDepartmentPlans(department."DepartmentId") AS
dp;
    END LOOP;
END;
$$ LANGUAGE plpgsql;
```

Лістинг Е.11 – Функція для отримання інформації по виконанню календарних планів видань усіма кафедрами

```
CREATE OR REPLACE FUNCTION
GetPublicationsByDepartment(department_id INT)
RETURNS TABLE (
    PublicationId INT, Title TEXT, PublicationDate TIMESTAMPTZ,
    Abstract TEXT, Description TEXT, Keywords TEXT[],
    isPublished BOOLEAN, Volume INT, Language INT,
    LecturerId INT, LecturerFirstName TEXT,
    LecturerLastName TEXT, LecturerMiddleName TEXT,
    AcademicTitle INT
) AS $$
BEGIN
    RETURN QUERY
    SELECT
        pub."PublicationId", pub."Title",
        pub."PublicationDate", pub."Abstract",
        pub."Description", pub."Keywords",
        pub."isPublished", pub."Volume",
        pub."Language", lec."Id" AS "LecturerId",
        per."FirstName" AS "LecturerFirstName",
        per."LastName" AS "LecturerLastName",
        per."MiddleName" AS "LecturerMiddleName",
        lec."AcademicTitle"
    FROM "Publications" AS pub
    INNER JOIN
        "PersonPublication" AS pp ON pub."PublicationId" =
pp."PublicationsPublicationId"
    INNER JOIN
        "Person" AS per ON pp."AuthorsId" = per."Id"
    INNER JOIN
        "Lecturers" AS lec ON per."Id" = lec."Id"
    INNER JOIN
        "DepartmentLecturer" AS dl ON lec."Id" =
dl."LecturersId"
    WHERE
        dl."DepartmentsDepartmentId" = department_id;
END;
$$ LANGUAGE plpgsql;
```

Лістинг Е.12 – Функція для отримання публікацій, виданих на кафедрі

ДОДАТОК Ж

Вихідний код основних класів програмної реалізації

```
[HttpGet]
public async Task<IActionResult> Get(
 string? searchTerm, string? authorName, DateTime?
startDateFilter,
 DateTime? endDateFilter, string? categories)
{
 IQueryable<Publication> query =
_appDbContext.Publications.Include(p => p.Authors)
  .Where(p => p.isPublished == true);

 if (!string.IsNullOrEmpty(categories))
 {
  int[] categoryIds =
categories?.Split(',').Select(int.Parse).ToArray();
  query = query.Where(p => p.Disciplines.Any(d =>
categoryIds.Contains(d.Id)));
 }
 if (!string.IsNullOrEmpty(searchTerm))
 {
  query = query.Where(p =>
   EF.Functions.Like(p.Title.ToLower(), $"%
{searchTerm.ToLower()}%") ||
   EF.Functions.Like(p.Description.ToLower(), $"%
{searchTerm.ToLower()}%") ||
   EF.Functions.Like(p.Abstract.ToLower(), $"%
{searchTerm.ToLower()}%") ||
   p.Keywords.Any(k => EF.Functions.Like(k.ToLower(), $"%
{searchTerm.ToLower()}%")));
 }

 if (!string.IsNullOrEmpty(authorName))
 {
  var lastName = authorName.Split(' ')[0];
  var firstName = authorName.Split(' ')[1];
  var middleName = authorName.Split(' ')[2];
  middleName = middleName.Remove(middleName.Length - 1);

  query = query.Where(p => p.Authors.Any(a =>
   EF.Functions.Like(a.FirstName, $"%{firstName}%") &&
   EF.Functions.Like(a.LastName, $"%{lastName}%") &&
```

```
   EF.Functions.Like(a.MiddleName, $"%{middleName}%")));
 }

 if (startDateFilter.HasValue)
 {
  startDateFilter = DateTime.SpecifyKind(startDateFilter.Value,
DateTimeKind.Utc);
  query = query.Where(p => p.PublicationDate >=
startDateFilter.Value.Date);
 }

 if (endDateFilter.HasValue)
 {
  endDateFilter = DateTime.SpecifyKind(endDateFilter.Value,
DateTimeKind.Utc);
  query = query.Where(p => p.PublicationDate <=
endDateFilter.Value.Date);
 }

 var publicationShortDtos = await query.Take(100)
  .Select(item => new PublicationShortDto
  {
   PublicationId = item.PublicationId,
   Title = item.Title,
   PublicationDate = item.PublicationDate ??
DateTime.Now.ToUniversalTime(),
   Description = item.Description,
   Keywords = item.Keywords,
   Lecturers = item.Authors
  }).ToListAsync();

 return Ok(publicationShortDtos);
}
```

Лістинг Ж.1 – Код контролеру, що здійснює пошук публікацій в БД за критеріями

```
public async Task<IActionResult> Login(LoginUser user)
{
 if (!ModelState.IsValid) return BadRequest(ModelState);

 var (success, error) = await _authService.Login(user);

 if (success)
 {
  var role = await _authService.GetRole(user.Email);
  var token = _authService.GenerateTokenString(user, roles);
  ApplicationUser userInfo = await
_authService.GetUserInfo(user);

  var response = new
  {
   Token = token,
   UserInfo = userInfo,
   UserRole = roles.First()
  };

  JsonSerializerSettings settings = new JsonSerializerSettings()
  {
   ReferenceLoopHandling = ReferenceLoopHandling.Ignore
  };

  return Ok(JsonConvert.SerializeObject(response,
Formatting.Indented, settings));
 }

 return BadRequest($"Login failed. {error}");
}
```

Лістинг Ж.2 – Метод авторизації та аутентифікації

ДОДАТОК И

Розподіл привілеїв на об’єкти БД

Таблиця И.1 – Привілеї ролей на об’єкти БД

Назва	Користувачі бази даних					
	Студент	Викладач	Завідувач кафедри	Гарант освітньої програми	Навчальний відділ	Адміністратор
Таблиці						
DepartmentLecturer	-	R	CRUD	R	R	CRUD
Departments	-	R	R	R	R	CRUD
DisciplinePublication	R	CRUD	CRUD	R	R	CRUD
DisciplineWorkPlan	-	R	CRUD	R	R	CRUD
Disciplines	R	R	R	R	R	CRUD
EducationYear	-	R	R	R	R	CRUD
HeadsOfDepartments	-	R	R	R	R	CRUD
LecturerDisciplines	-	R	CRUD	R	R	CRUD
Lecturers	-	R	R	R	R	CRUD
MethodologicalPublications	R	CRUD	CRUD	R	R	CRUD
Person	R	R	R	R	R	CRUD
PersonPublication	R	CRUD	CRUD	R	R	CRUD
Plans	-	R	CRUD	-	R	CRUD
Publications	R	CRUD	CRUD	R	R	CRUD
ScientificArticles	R	CRUD	CRUD	R	R	CRUD
ScientificConferenceTheses	R	CRUD	CRUD	R	R	CRUD
ScientificDissertations	R	CRUD	CRUD	R	R	CRUD
ScientificMonographs	R	CRUD	CRUD	R	R	CRUD
ScientificPatents	R	CRUD	CRUD	R	R	CRUD

Продовження Таблиці И.1

Назва	Користувачі бази даних					
	Студент	Викладач	Завідувач кафедри	Гарант освітньої програми	Навчальний відділ	Адміністратор
ScientificPublications	R	CRUD	CRUD	R	R	CRUD
Specialities	-	R	CRUD	R	R	CRUD
WorkPlans	-	R	CRUD	R	R	CRUD
Представлення						
PublishedPublicationsWith Authors	R	R	R	R	R	R
Функції						
GetPublicationInfo	X	X	X	X	X	X
GetPublicationsByAuthorId	X	X	X	X	X	X
GetDepartmentPlans	-	X	X	-	X	X
GetPlanInfo	-	X	X	-	X	X
GetDisciplinePublications	-	-	X	X	-	X
GetAllDepartmentsPlans	-	-	-	-	X	X
GetPublicationsBy Department	-	-	X	-	-	X
Збережені процедури						
AddMethodological Publication	-	X	X	-	-	X
UpdateMethodological Publication	-	X	X	-	-	X
DeletePublication	-	X	X	-	-	X
AddPlan	-	-	X	-	-	X

Умовні позначення: C – Create, R – Read, U – Update, D – Delete, X – Execute.