КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет комп'ютерних наук та кібернетики

Кафедра теорії та технології програмування

Кваліфікаційна робота

на здобуття ступеня магістра

за освітньо-науковою програмою «Інформатика»

спеціальності 122 «Комп'ютерні науки»

на тему:

СИСТЕМА ПІДТРИМКИ ПРОЦЕСУ РОЗРОБКИ РОБОЧИХ ПРОГРАМ НАВЧАЛЬНИХ ДИСЦИПЛІН ДЛЯ ЗАКЛАДІВ ВИЩОЇ ОСВІТИ

Виконав студент 2-го курсу магістратури
Дмитро БЕЗУХ _____
(підпис)

Науковий керівник:
доцент, кандидат фіз.-мат. наук
Людмила ОМЕЛЬЧУК _____
(підпис)

Засвідчую, що в цій роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Роботу розглянуто і допущено до захисту на засіданні кафедри теорії та технології програмування
«08» травня 2023 р.,
протокол № 16
Завідувач кафедри
Микола НІКІТЧЕНКО _____
(підпис)

Київ – 2023

РЕФЕРАТ

Обсяг роботи: основний текст 73 сторінки, 57 ілюстрацій, 26 таблиць, 24 джерела посилань.

АВТОМАТИЗАЦІЯ НАВЧАЛЬНОГО ПРОЦЕСУ, БАЗА ДАНИХ, ІНТЕРФЕЙС ПРОГРАМНОГО ПРОДУКТУ, ТЕХНІЧНЕ ЗАВДАННЯ ДО ПРОДУКТУ, ТЕХНОЛОГІЯ ПРОГРАМУВАННЯ, ХМАРНІ ТЕХНОЛОГІЇ.

Об’єктом даної роботи є процес розробки робочих програм навчальних дисциплін (РПНД) для закладів вищої освіти (ЗВО). Предметом роботи є система, для автоматизації процесу розробки РПНД, підтримки та надання актуальної інформації про освітні програми та їх складові, документообіг робочих програм.

Метою кваліфікаційної роботи є проєктування та розробка системи для автоматизації процесу створення РПНД та реалізації в ній основних засобів збереження, обробки та передачі інформації.

Методи розроблення: комп’ютерне моделювання, проєктування баз даних, розробка системи на основі моделі за допомогою ітеративної методології. Інструменти розроблення: інтегроване середовище розробки Visual Studio 2022, Visual Studio Code, SQL Server Management Studio.

Результати роботи: розглянуто процес розробки РПНД, визначено способи автоматизації даного процесу, проаналізовано наявні системи, які призначені для автоматизації управління навчальним закладом, розроблено інтерфейс користувача, спроєктовано та реалізовано БД, розроблено систему для підтримки процесу розробки РПНД для ЗВО та розгорнуто її в хмарному середовищі Microsoft Azure.

Розроблений програмний засіб може використовуватись для підтримки освітнього процесу у закладах вищої освіти.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

AOT – Ahead-of-Time, компіляція перед виконанням;

API – Application programming interface, прикладний програмний інтерфейс;

BLL – Business Logic Layer, логіка бізнес рівня;

CPU – Central processing unit, центральний процесор;

CSS – Cascading Style Sheets, каскадні таблиці стилів;

DAL – Data access layer, рівень доступу до даних;

ECTS – European Credit Transfer and Accumulation System, європейська кредитна трансферно-накопичувальна система;

EF – Entity Framework;

HTML – HyperText Markup Language, мова гіпертекстової розмітки;

HTTPS – HyperText Transfer Protocol Secure, протокол передачі даних;

IaaS – Infrastructure as a Service, інфраструктура як послуга;

IDE – Integrated development environment, інтегроване середовище розробки;

JIT – Just-in-Time, компіляція «на льоту»;

LINQ – Language Integrated Query, запити, інтегровані в мову;

MVC – Model–View–Controller, модель-представлення-контролер;

ORM – Object-relational mapping, об'єктно-реляційна проєкція;

PaaS – Platform as a service, платформа як послуга;

PL – Presentation Layer, презентаційний рівень;

POCO – Plain Old CLR Object;

REST – Representational State Transfer, підхід до архітектури мережевих протоколів;

SaaS – Software as a service, програмне забезпечення як послуга;

SQL – Structured Query Language, мова структурованих запитів;

W3C – World Wide Web Consortium, головна міжнародна організація, що розробляє й впроваджує технологічні стандарти;

WEB – Система доступу до пов'язаних між собою документів на різних комп'ютерах, підключених до Інтернету;

АСУ – Автоматизована система управління;

БД – База даних;

ЗВО – Заклад вищої освіти;

НП – Навчальний план;

ОП – Освітня програма;

РПНД – Робоча програма навчальної дисципліни;

СКБД – Система керування базами даних.

ВСТУП

Оцінка сучасного стану об’єкта розробки. Світовий досвід та наукові публікації свідчать, що інформатизація в європейських та інших закордонних закладах вищої освіти (ЗВО) є важливим напрямом розвитку [1-3]. Багато світових лідерів ІТ-індустрії працюють над впровадженням інформаційних технологій в освітній процес, однак, більшість систем управління ЗВО є комерційними продуктами з англомовним інтерфейсом, вимагають значних витрат на ліцензоване програмне забезпечення і не враховують специфіку вітчизняних університетів. Багато закладів освіти в різних країнах світу також шукають засоби автоматизації процесів освітньої діяльності, таких як системи управління освітнім процесом, електронним документообігом тощо [4].

ЗВО мають за мету забезпечити ефективну навчально-пізнавальну та наукову діяльність усіх учасників освітнього процесу. Однак, з розвитком технологій та загальної доступності до них, стає важливим застосування інновацій у сфері інформатики, що допоможе підвищити якість організації навчального процесу. Метою цих інновацій є зменшення кількості помилок, більш ефективне використання часу учасниками освітнього процесу, а також підвищення доступності інформації.

Збільшення продуктивності у розв’язанні організаційних питань в ЗВО допомагає зосередитися на самому процесі навчання та науковій діяльності, що в свою чергу підвищує рівень освіти в цілому. Це досягається шляхом кращого розподілу часу та зусиль, що дозволяє зосередитися на покращенні процесу навчання, замість витрачення зусиль на підтримання поточного рівня.

Актуальність роботи та підстави для її виконання. Одна із задач, яка виникає при організації освітнього процесу, полягає у формуванні робочих програм навчальних дисциплін (РПНД). Це завдання включає заповнення інформацією про дисципліну, таку як освітня програма, навчальні цілі, системи оцінювання, компетентності та програмні результати дисципліни

тощо. При цьому необхідно звертатись до інших нормативних документів, таких як: навчальний план, який містить детальну інформацію про кількість кредитів ECTS та кількість годин, виділених на різні складові обраної дисципліни та детальний опис освітньої програми, яка містить дисципліну.

З розвитком галузей знань, особливо в комп'ютерних науках, стає важливим постійно переглядати та оновлювати навчальний процес, щоб відповідати сучасним тенденціям та технологіям. Університети, які прагнуть залишатись в темпі розвитку, частіше оновлюють освітні програми та РПНД, що вимагає все більше часу та зусиль. Під час формування РПНД викладач змушений постійно перевіряти різні документи, шукати необхідну інформацію та переписувати її, що збільшує ризик помилок та затримує процес розробки РПНД. Тому використання програмних засобів для автоматизації цього процесу є важливим та актуальним.

Мета й завдання роботи. Метою кваліфікаційної роботи є проєктування та розробка системи для автоматизації процесу створення РПНД та реалізації в ній основних засобів збереження, обробки та передачі інформації. Для досягнення мети було поставлено та виконано наступні завдання:

- Проаналізувати процес створення РПНД та визначити шляхи його автоматизації.
- Проаналізувати існуючі системи управління ЗВО.
- Розробити вимоги до функціонала системи.
- Побудувати моделі баз даних, які повністю відповідатимуть всім вимогам, що висуваються до системи.
- Реалізувати трирівневу серверну архітектуру та WEB-частину застосунку для автоматизації процесу розробки РПНД.
- Розгорнути створену систему, використовуючи хмарні технології Microsoft Azure.
- Поглибити та закріпити навички роботи з технологією ASP.NET

Core 6.0, фреймворком Angular та взаємодією з базою даних.

Об'єкт, методи й засоби розроблення. Об’єктом даної роботи є процес розробки РПНД для ЗВО. Предметом роботи є система, для автоматизації процесу розробки РПНД, підтримки та надання актуальної інформації про освітні програми та їх складові, документообіг робочих програм.

Розробка системи складалася з чотирьох основних етапів:

1. Проєктування та розробка баз даних.
2. Реалізація трирівневої серверної архітектури (рівень доступу до даних (DAL), рівень бізнес логіки (BLL), презентаційний рівень (PL)).
3. Розробка WEB-частини.
4. Розгортання системи, використовуючи Microsoft Azure

Розробка вебзастосунку здійснювалась за допомогою ітеративної методології розробки, що передбачає поетапне створення та оцінювання початкової версії програми. Після оцінки здійснюється доопрацювання додатку з урахуванням отриманих зауважень та пропозицій. Цей механізм повторюється на кожному етапі розробки, доки не буде створено додаток, який повністю відповідає всім вимогам.

Для розробки програмного засобу були використані безкоштовні та доступні для вільного поширення інтегровані середовища розробки (IDE) Visual Studio 2022, Visual Studio Code та SQL Server Management Studio які підтримують мову програмування C#, фреймворк Angular та мову SQL.

Апробація роботи та публікації з теми роботи. За результатами роботи було опубліковано тези на конференції «Автоматизація та комп’ютерно-інтегровані технології у виробництві та освіті: стан, досягнення, перспективи розвитку» 13.04.2023 – 19.04.2023 [5].

РОЗДІЛ 1.
ОГЛЯД НАЯВНИХ НА РИНКУ СИСТЕМ

Ідея використання програмного забезпечення для покращення організації навчального процесу в ЗВО не є новою. Часто на вебсайтах факультетів можна знайти описи освітніх програм та навчальних планів, але ці матеріали можуть бути застарілими. Викладачі можуть обмінюватися актуальними матеріалами через електронну пошту, але це не завжди зручно та ефективно. Тому в багатьох університетах впроваджуються спеціальні системи автоматизації, які мають різний функціонал та допомагають викладачам та студентам отримувати актуальну інформацію та спрощувати процес організації навчальної діяльності.

1.1 АСУ «Університет»

На рис. 1 зображено вигляд головної сторінки додатку АСУ «Університет» [6]. Даний додаток представляє собою великий масив інформації, який розміщений в структурованому форматі на різні теми та рівні доступу. Він має низку розділів, які включають "Базовий", де надається інформація про ЗВО в цілому та його структурні підрозділи; "Оперативний", де розміщені оголошення, новини та посилання на вебгазети; "Співробітникам", де доступні документи, що регламентують навчальну та наукову діяльність; "Наука", де надається інформація про наукову діяльність; "Студентам", де представлена інформація про студентське життя; "Поступаючим", де розміщена інформація для абітурієнтів; "Контакти", де можна знайти інформацію про розташування вищого навчального закладу та адреси інформаційних ресурсів; і "Партнери", де надається інформація про партнерські стосунки. Кожен з цих тематичних розділів має деревоподібну

структуру з багатьма сторінками, доступними для загального користувача.

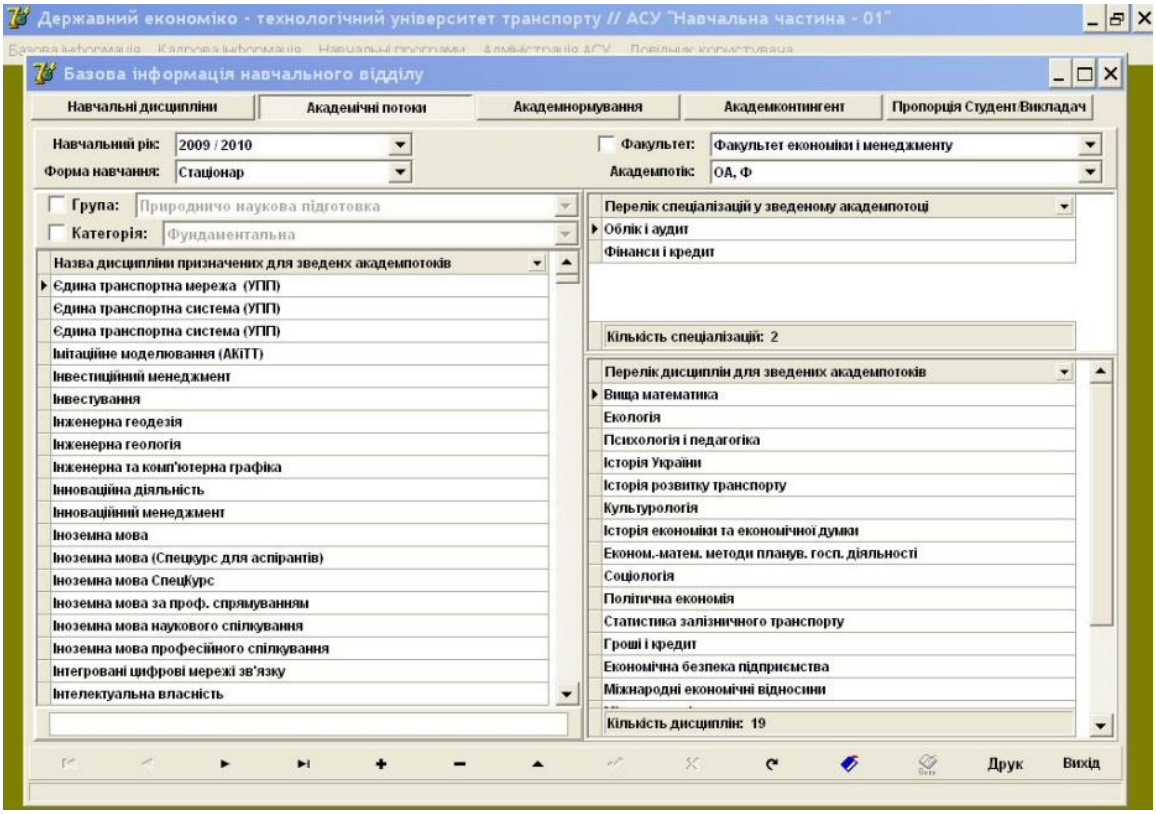


Рисунок 1 – Зовнішній вигляд додатку АСУ «Університет»

Також додаток надає можливість генерації різних офіційних довідкових та звітних документів про абітурієнтів в реальному часі. Ці документи можуть бути виведені на друк, опубліковані на вебсайті, конвертовані в інші формати для використання в інших програмах або архівовані.

1.2 АСУ «Вищий навчальний заклад»

На рис. 2 зображено схему роботи системи АСУ «Вищий навчальний заклад» [7]. Дана система є комплексним програмним забезпеченням, розробленим спеціально для управління навчальним закладом, і має велику кількість функцій і можливостей. Вона забезпечує зберігання всієї інформації в базі даних, яка належить навчальному закладу, і може використовуватись для

різних завдань та процесів. Складові цієї системи включають автоматизацію роботи приймальної комісії, деканату, дирекції студентського містечка, систему тестування, менеджер резервних копій, веброзклад, вебкабінет студента та систему контролю доступу. Завдяки цим компонентам, система дозволяє автоматизувати ряд процесів, таких як реєстрація абітурієнтів в приймальній комісії, зберігання особистих справ абітурієнтів, спрощення роботи методистів у деканаті, створення та редагування шаблонів звітів, створення резервних копій баз даних, вбудовування розкладу, реєстрація та авторизація студентів в особистому кабінеті, перегляд успішності та формування звітів про навчальний план та успішність, а також забезпечення системи контролю доступу.

Рисунок 2 – Схема роботи системи АСУ ВНЗ

1.3 Система «Triton»

Система «Triton» Київського національного університету імені Тараса Шевченка надає актуальну інформацію про навчальний процес, а саме оцінки авторизованого студента, графік контролю з дисциплін, інформацію про стипендію та навчальний план [8]. Також студенти мають можливість здійснювати вибір навчальних дисциплін (див. рис. 3).

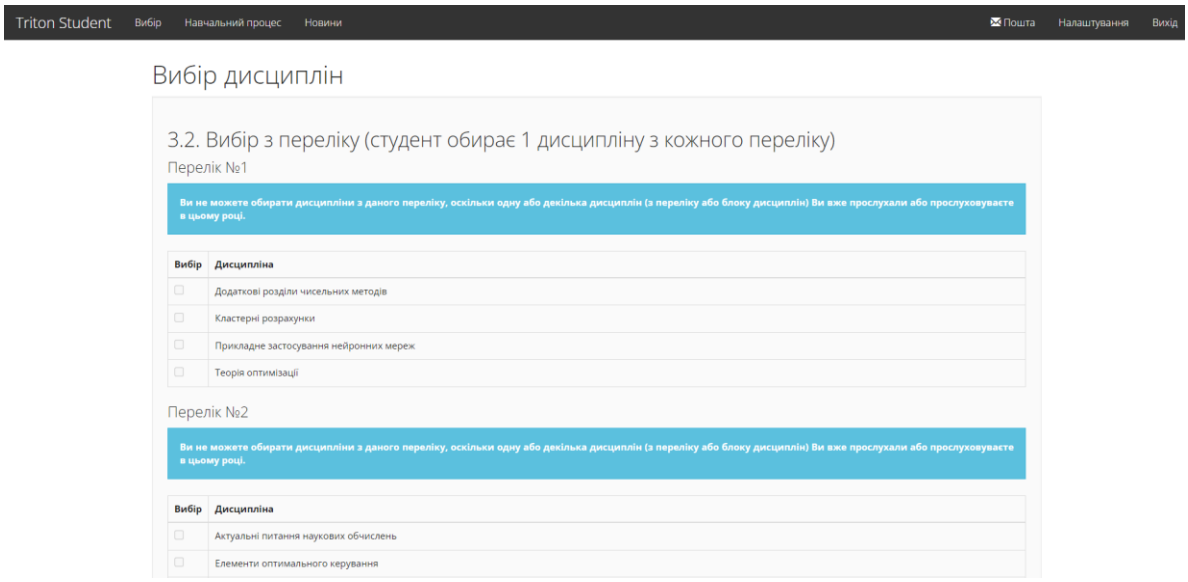


Рисунок 3 – Сторінка вибору дисципліни в системі «Triton»

Дана система дозволяє створювати навчальний план (НП), генерувати файли НП (див. рис. 4) та звіти, керувати інформацією про групи, здійснювати створення записів про студентів, їх переведення, керувати інформацією про аудиторний фонд ЗВО.

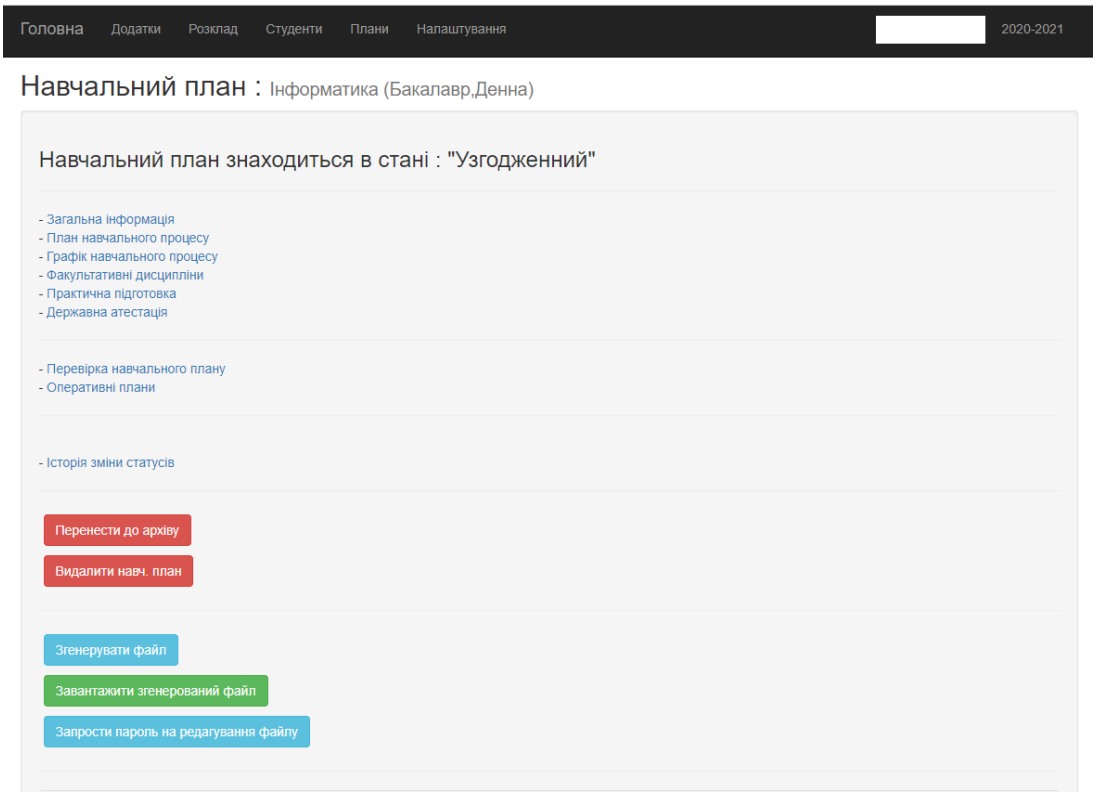


Рисунок 4 – Сторінка керування НП в системі «Triton»

Більшість з розглянутих систем, що використовуються в різних ЗВО, надають користувачам доступ до інформації про НП та ОП, розкладів, містять функціонал для керування інформацією про студентів, дисципліни та оцінки. Деякі з цих систем мають вбудований функціонал для автоматизованого формування навчальних планів. Однак, під час дослідження не було знайдено жодної системи, яка має можливість автоматичного заповнення позицій для РПНД. Це означає, що процес заповнення РПНД залишається ручним та вимагає відповідного втручання користувача, тому задача, яка розв’язується в даній кваліфікаційній роботі є актуальною.

РОЗДІЛ 2.
ОГЛЯД ВИКОРИСТАНИХ ТЕХНОЛОГІЙ

Для розробки та реалізації системи, для автоматизації процесу створення РПНД, використано наступні технології:

- технологія ASP.NET Core 6.0 [9-12];
- технологія Entity Framework Core [13, 14];
- фреймворк Angular [15-17];
- фреймворк Bootstrap [18, 19];
- трирівнева серверна архітектура [20];
- хмарні технології Microsoft Azure [21, 22].

2.1 Технологія ASP.NET Core

ASP.NET Core 6.0 [9-12] – це вебфреймворк з відкритим вихідним кодом, який був створений компанією Microsoft для розробки вебдодатків, сервісів та API. Це новітній розвиток популярного фреймворку ASP.NET, який забезпечує уніфіковану модель програмування для створення крос-платформних високопродуктивних вебдодатків. Основною мовою програмування, що використовується в ASP.NET Core 6.0, є C#, проте фреймворк також підтримує F# та Visual Basic. Фреймворк побудований на платформі .NET, яка надає розробникам великий набір бібліотек та інструментів для розробки вебдодатків. ASP.NET Core 6.0 є модульним, гнучким та масштабованим фреймворком, який містить безліч функцій, що роблять його ідеальним вибором для створення сучасних вебдодатків [12].

Кросплатформність. Кросплатформна розробка в ASP.NET Core 6.0 означає можливість створювати додатки, які без великих змін можуть працювати на різних операційних системах та архітектурах. Фреймворк надає

уніфіковану модель програмування для створення високопродуктивних вебдодатків, які можуть працювати на Windows, macOS та Linux з однією кодовою базою. Крос-платформна підтримка дозволяє використовувати один код для різних платформ, зменшуючи час та зусилля для розробки та підтримки додатків. ASP.NET Core 6.0 також підтримує контейнери Docker, що спрощує розгортання додатків в різних середовищах. В загальному, кросплатформна підтримка ASP.NET Core 6.0 робить його ідеальним вибором для створення вебдодатків та сервісів, які можуть працювати на будь-якій платформі без додаткових зусиль та модифікацій [11].

Висока продуктивність. ASP.NET Core 6.0 оптимізовано для високої продуктивності і може обробляти велику кількість паралельних запитів без зниження продуктивності програми. Він також включає такі функції, як кешування відповідей, які можуть значно підвищити продуктивність вебдодатків.

Модульна архітектура. ASP.NET Core 6.0 має модульну архітектуру, яка дозволяє легко додавати або видаляти компоненти за потреби. Це допомагає зберегти легкість програми, полегшує її обслуговування та масштабування [9].

Відкритий вихідний код. ASP.NET Core 6.0 має відкритий вихідний код, що означає, що розробники можуть отримати доступ до вихідного коду і зробити свій внесок у його розвиток.

Інтеграція з іншими технологіями. ASP.NET Core 6.0 інтегрується з іншими технологіями Microsoft, такими як Azure, що дозволяє легко створювати і розгортати додатки в хмарі. Він також підтримує широкий спектр фронтенд-технологій, таких як Angular, React і Vue.js, що полегшує створення сучасних вебдодатків.

Безпека. ASP.NET Core 6.0 включає вбудовані функції безпеки, такі як аутентифікація та авторизація, що полегшує захист вебдодатків. Він також

включає такі функції, як захист даних і підтримка HTTPS, які допомагають захистити додаток і його дані від атак [10].

2.2 Технологія Entity Framework Core

Технологія Entity Framework Core (EF Core) [13, 14] – це ORM (object-relational mapping) технологія, яка дозволяє .NET розробникам працювати з реляційними базами даних за допомогою коду на C#. Entity Framework Core розроблений, щоб бути легким, розширюваним і кросплатформним. Підтримує широкий спектр постачальників баз даних, включаючи Microsoft SQL Server, SQLite, MySQL, PostgreSQL та Oracle. За допомогою Entity Framework Core розробники можуть створювати моделі баз даних, використовуючи код C# або генеруючи їх з БД. Entity Framework Core може виконувати такі завдання, як створення, читання, оновлення та видалення записів у базі даних, а також складні операції з даними.

Однією з ключових переваг Entity Framework Core є те, що ця технологія допомагає писати чистіший і легший в обслуговуванні код. Абстрагуючись від деталей операцій з базами даних, розробники можуть зосередитися на написанні коду, який стосується бізнес-логіки та функціональності додатків. Це дозволяє писати коду, який менш схильний до помилок і який легше підтримувати з часом [13].

Ще однією перевагою Entity Framework Core є те, що він дозволяє працювати з базами даних, використовуючи об'єктно-орієнтовані методи програмування. Розробники можуть створювати класи, які відображаються на таблиці бази даних, і використовувати запити LINQ для отримання даних з бази даних. Це може полегшити роботу з даними у більш природній та інтуїтивно зрозумілий спосіб, замість того, щоб писати складні SQL запити [14].

Підтримка широкого спектру постачальників баз даних,

кросплатформна сумісність та підтримка сучасних практик розробки програмного забезпечення роблять дану технологію популярним вибором для створення надійних та масштабованих додатків, керованих базами даних.

Структура роботи технології EF Core зображена на рис. 5.

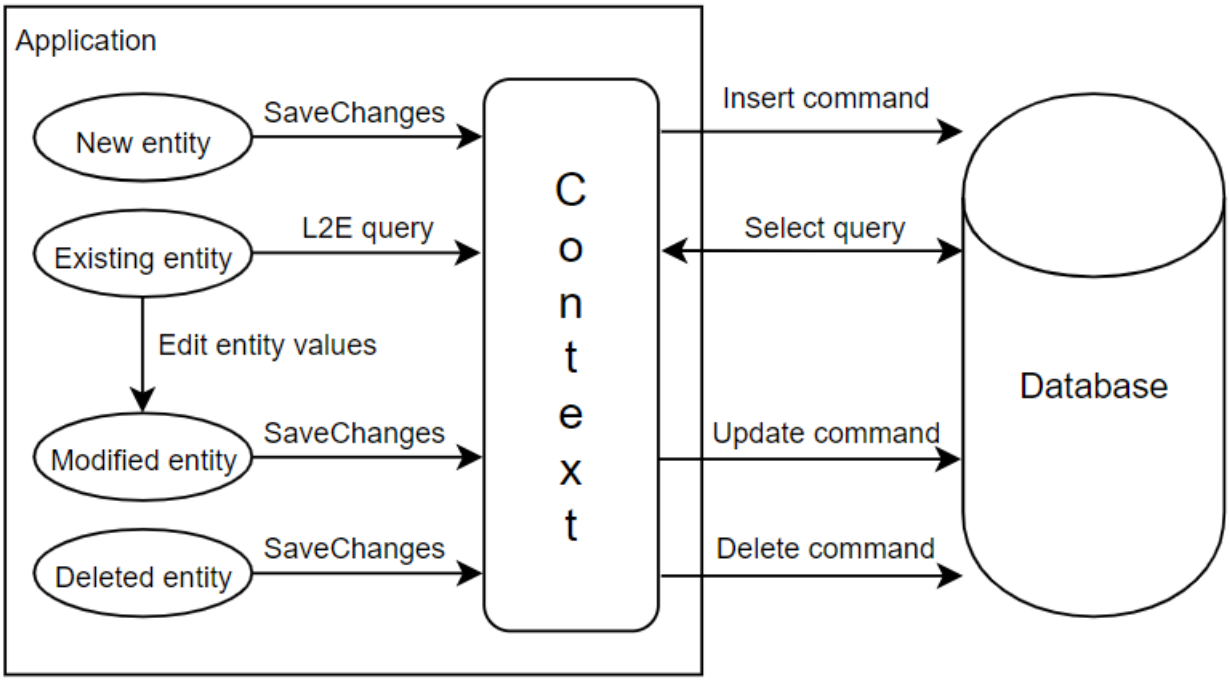


Рисунок 5 – Структура роботи технології EF Core

2.3 Фреймворк Angular

Angular [15-17] – це відомий фреймворк з відкритим вихідним кодом для створення вебдодатків, що розроблений та підтримується компанією Google. Це потужний і всеосяжний фреймворк для створення динамічних односторінкових вебдодатків.

Angular надає можливості для створення складних, керованих даними додатків, включаючи компоненти, шаблони, сервіси, маршрутизацію, форми та ін'єкцію залежностей. Він базується на архітектурі Model-View-Controller (MVC), яка розділяє додаток на три окремі шари: модель (дані), представлення (інтерфейс) і контролер (логіка).

Однією з ключових особливостей Angular є використання директив – спеціальних атрибутів HTML, які дозволяють розробникам прив'язувати поведінку до елементів інтерфейсу користувача. Angular також включає потужний набір вбудованих директив, таких як ngIf, ngFor та ngStyle [15].

Angular також приділяє велику увагу продуктивності та оптимізації, завдяки таким функціям, як компіляція Ahead-of-Time (AOT) та ліниве завантаження.

Ahead-of-Time (AOT) – це метод компіляції, що використовується в Angular, який перетворює TypeScript і HTML-код Angular-додатку в ефективний JavaScript-код під час процесу збірки, до того, як додаток буде розгорнуто в браузері. AOT має кілька переваг над компіляцією Just-in-Time (JIT), яка є методом компіляції за замовчуванням в Angular. Однією з головних переваг AOT є покращена продуктивність. Оскільки код компілюється заздалегідь, браузер може завантажувати і рендерити ваш додаток набагато швидше, ніж з JIT. AOT також зменшує розмір додатку за рахунок видалення непотрібного коду та оптимізації коду, що залишився. Ще одна перевага AOT полягає в тому, що він виявляє помилки шаблонів на ранній стадії, під час процесу збірки, а не під час виконання. Це може допомогти уникнути проблем, які можуть виникнути під час запуску програми у браузері [16, 17].

Ліниве завантаження дозволяє завантажувати модулі та компоненти лише тоді, коли вони потрібні, зменшуючи час початкового завантаження вашого додатку.

Окрім основного фреймворку, Angular також має багату екосистему інструментів та бібліотек, включаючи Angular Material, який надає набір готових компонентів інтерфейсу користувача, та RxJS, який надає потужний набір інструментів для роботи з асинхронними потоками даних [17].

В цілому, Angular – це потужний і всеосяжний фреймворк для створення складних вебдодатків, керованих даними. Він має сильний фокус на продуктивності та оптимізації, а також надає багатий набір інструментів та

бібліотек для роботи розробників. Схематичну архітектуру Angular додатка зображено на рис. 6.

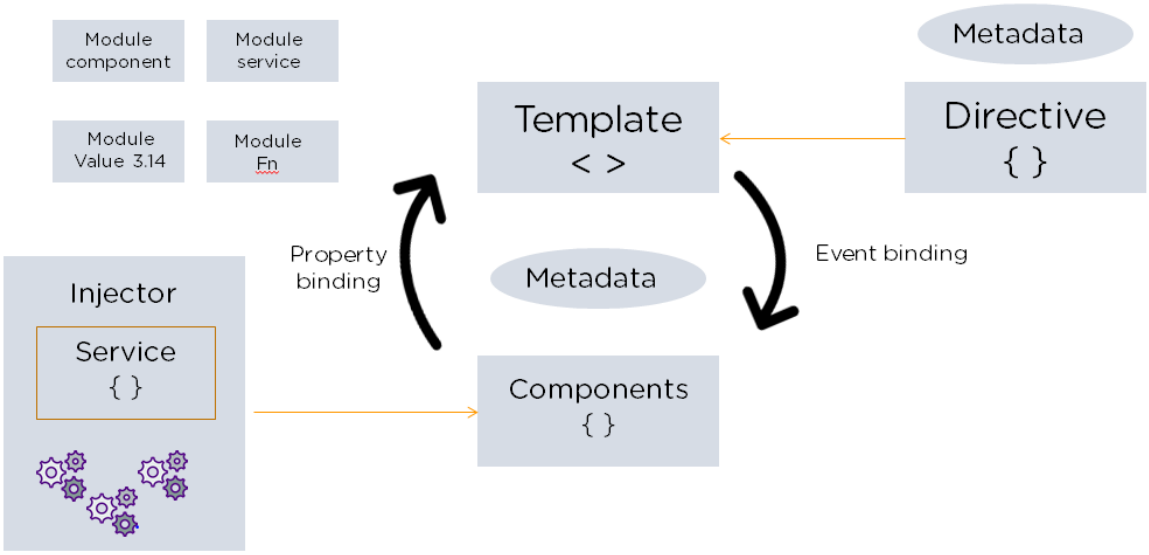


Рисунок 6 – Схематична архітектура Angular

2.4 Фреймворк Bootstrap

Bootstrap [18, 19] – це фронтенд-фреймворк з відкритим вихідним кодом для створення адаптивних вебдодатків. Він був розроблений компанією Twitter і зараз підтримується спільнотою розробників.

Основна мета Bootstrap – полегшити створення адаптивних, мобільних вебсторінок. Bootstrap надає набір готових компонентів HTML, CSS і JavaScript, які можна використовувати для швидкого створення загальних елементів інтерфейсу користувача, таких як навігаційні меню, форми, кнопки і модальні елементи. Ці компоненти розроблені для спільної роботи, що полегшує створення послідовного та цілісного користувацького інтерфейсу.

Bootstrap також включає адаптивну систему сітки, яка дозволяє створювати макети, що адаптуються до різних розмірів екранів і пристроїв. Це особливо корисно для створення додатків, які повинні добре працювати як на десктопних, так і на мобільних пристроях [19].

Однією з ключових переваг використання Bootstrap є те, що він може

заощадити багато часу і зусиль при створенні вебдодатків. Замість того, щоб створювати елементи користувацького інтерфейсу з нуля, даний фреймворк дозволяє використовувати готові компоненти та стилі. Це допоможе розробникам зосередитися на функціональності та змісті додатку, а не витрачати багато часу на користувацький інтерфейс.

Bootstrap також дуже дозволяє змінювати стилі та компоненти за замовчуванням, щоб відповідати необхідному зовнішньому вигляду додатку. Крім того, існує велика спільнота розробників, які роблять свій внесок у Bootstrap і створюють сторонні плагіни та розширення, ще більше розширюючи функціональність фреймворку [19].

Загалом, Bootstrap – це потужний і гнучкий фронтенд-фреймворк, який може заощадити розробникам час і зусилля при створенні адаптивних вебдодатків. Його готові компоненти та адаптивна система сітки в поєднанні з можливістю налаштування та підтримкою спільноти роблять його популярним вибором серед розробників.

2.5 Трирівнева серверна архітектура

Трирівнева архітектура [20] – це шаблон архітектури програмного забезпечення, який розділяє додаток на три окремі логічні рівні: рівень представлення, рівень бізнес-логіки та рівень доступу до даних (див. рис. 7).

Основна перевага трирівневої архітектури полягає в тому, що кожен рівень може розроблятися і підтримуватися окремою командою, оскільки він має свою власну інфраструктуру, що дозволяє оновлювати та масштабувати його без впливу на інші рівні.

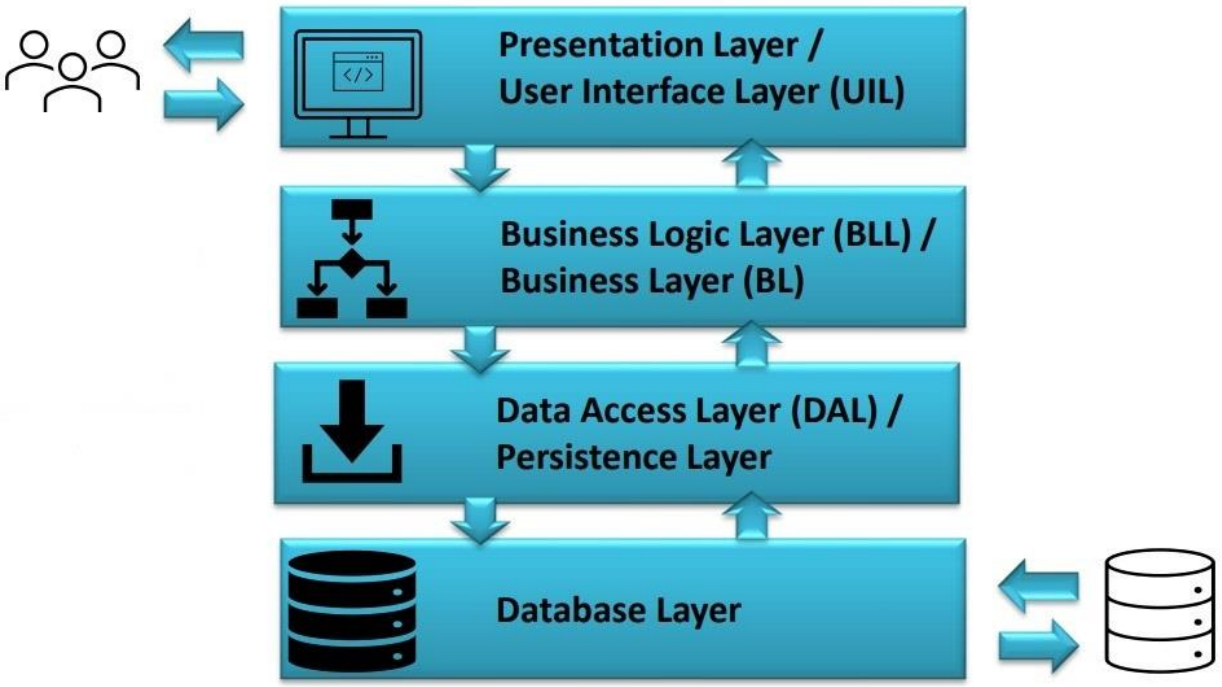


Рисунок 7 – Приклад трирівневої архітектури

Рівень доступу до даних. Це найнижчий рівень, який безпосередньо взаємодіє з системою зберігання даних, такою як база даних або файлова система. Він відповідає за отримання та зберігання даних, а також надає інтерфейс для рівня бізнес-логіки для доступу до даних та маніпулювання ними.

Рівень бізнес логіки. Це середній рівень, який містить бізнес-логіку програми. Він відповідає за обробку даних, введених користувачем, реалізацію бізнес-правил і логіки, а також управління потоком даних між рівнем представлення і рівнем доступу до даних.

Презентаційний рівень. Це найвищий рівень, який безпосередньо взаємодіє з кінцевим користувачем. Він відповідає за представлення інформації користувачеві та за збір вхідних даних. Рівень представлення часто включає компоненти інтерфейсу користувача, такі як форми, звіти та інформаційні панелі [20].

Трирівнева архітектура забезпечує чіткий розподіл завдань, що полегшує розробку, тестування, підтримку та масштабування додатків. Це

також сприяє повторному використанню коду, оскільки кожен рівень може бути розроблений незалежно і повторно використаний в інших додатках. Однак, реалізація тришарової архітектури може вимагати більше початкових зусиль, ніж простіші архітектури, а також більше ресурсів для управління та підтримки кожного рівня.

2.6 Хмарні технології Microsoft Azure

Microsoft Azure [21, 22] – це платформа хмарних обчислень і сервісів, що надається корпорацією Майкрософт (логотип компанії зображено на рис. 8). Вона пропонує широкий спектр хмарних технологій і послуг, включаючи обчислення, зберігання, бази даних, аналітику, штучний інтелект, машинне навчання, мережу, безпеку тощо. Azure використовується приватними особами, стартапами, підприємствами та урядами для створення, розгортання та управління додатками і сервісами в глобальному масштабі.

Azure надає широкий спектр послуг, включаючи інфраструктуру як послугу (IaaS), платформу як послугу (PaaS) та програмне забезпечення як послугу (SaaS). Віртуальні машини Azure пропонують гнучкі та масштабовані обчислювальні ресурси для запуску додатків і робочих навантажень на вимогу. Azure App Service пропонує повністю керовану платформу для створення, розгортання та масштабування веб і мобільних додатків. Azure Functions забезпечує безсерверні обчислення для програм і мікросервісів, керованих подіями. Azure Kubernetes Service – це повністю керована служба оркестрування контейнерів Kubernetes. Azure Cosmos DB - це глобально розподілена багатомодельна служба баз даних, призначена для масштабованих і високопродуктивних додатків [21].

Azure також надає широкий спектр інструментів і служб для розробників, включаючи Visual Studio, Visual Studio Code, Azure DevOps, Azure DevTest Labs і Azure Functions. Ці інструменти дозволяють розробникам

легко створювати, тестувати та розгортати програми та служби.

Azure доступний у понад 60 регіонах світу та забезпечує високу доступність і масштабованість. Також пропонує можливості гібридної хмари, що дозволяє компаніям використовувати Azure разом з існуючою локальною інфраструктурою. Azure надає функції безпеки та відповідності нормативним вимогам корпоративного рівня, включаючи управління ідентифікацією та доступом, шифрування, виявлення загроз тощо [22].

Загалом Microsoft Azure – це комплексна хмарна платформа з широким спектром послуг і можливостей, що дозволяє компаніям впроваджувати інновації та розвиватися в хмарі.

Рисунок 8 – Логотип Microsoft Azure

РОЗДІЛ 3.
ПРИЗНАЧЕННЯ ТА ЦІЛІ СТВОРЕННЯ СИСТЕМИ

3.1 Призначення системи

Призначенням системи є використання проєкту для автоматизації процесу створення та перегляду робочих програм, для детального ознайомлення користувачів з інформацією про навчальні дисципліни, освітні програми, компетентності та програмні результати. Діаграму прецедентів наведено в додатку А.

Кваліфікаційна робота передбачає:

- вивчення та дослідження різних методів, технологій, моделей та підходів, які використовуються для розробки вебзастосунів;
- дослідження та аналіз процесу створення робочих програм для подальшої автоматизації;
- аналіз наявних програмних засобів;
- проєктування та програмну реалізацію системи;
- розгортання системи за допомогою хмарних технологій Microsoft Azure.

Систему створено з метою:

- автоматизації процесу створення РПНД для закладів вищої освіти;
- документообіг підтверджених РПНД;
- ознайомлення користувачів із детальною інформацією про дисципліни, програмні результати, компетентності та освітні програми;
- забезпечення можливості зручного централізованого керування даними.

3.2 Вимоги до системи

Система повинна мати чотири типи користувачів: звичайні відвідувачі, викладачі, методисти та адміністратори.

Для того, щоб зробити вебзастосунок привабливим для користувачів, необхідно створити інтуїтивно зрозумілий і приємний дизайн. Оскільки більшість відвідувачів переглядає лише кілька сторінок і залишається на них незначну кількість часу, необхідно враховувати їхній поведінковий паттерн та забезпечити швидкий та легкий доступ до інформації, яка є найбільш цікавою для них. Крім того, дизайн має бути привабливим та зацікавлювати відвідувачів, що допоможе збільшити їхню увагу до сайту та збільшити ймовірність повернення на нього в майбутньому. Важливо також враховувати можливості контролю та персоналізації, які зможуть зацікавити відвідувачів і покращити їхній досвід використання сайту.

Викладачі, методисти та адміністратори проявляють більший інтерес до інформативності та можливостей системи, ніж до її зовнішнього вигляду. Сторінки на сайті вони переглядають більш детально і тримаються на ньому довший період часу, порівняно з відвідувачами.

Вебзастосунку передбачає виділення наступних функціональних підсистеми:

- підсистема звичайного відвідувача призначена для перегляду інформації про дисципліни, освітні програми, компетентності, програмні результати та робочі програми. Також має можливості для авторизації та завантаження файлів РПНД;
- підсистема викладача призначена для створення шаблону РПНД та завантаження готової робочої програми на перевірку для подальшого розміщення на сайті;
- підсистема методиста призначена для створення та редагування навчальних дисциплін, освітніх програма; створення, редагування та

додавання програмних результатів і компетентностей до наявних навчальних дисциплін та освітніх програм. Також має можливість перевіряти, підтверджувати, відхиляти (вказавши причину), створювати, завантажувати та видаляти завантажені РПНД викладачів;

- підсистема адміністратора призначена для керування користувачами та перегляду детальної інформації про них. Вона також забезпечує можливість керування правами доступу користувачів до різних розділів сайту, що забезпечує більш високий рівень безпеки та контролю.

3.3 Технічні вимоги до системи

У сучасному світі все більше людей користуються Інтернетом і для цього вони використовують різноманітні браузери. Однак, з великою кількістю різних браузерів виникає проблема несумісності вебсайтів. Кожен розробник браузера має свої підходи до відображення вебсторінок, що часто призводить до того, що вони виглядають по-різному в різних браузерах.

Ці проблеми стали причиною появи стандарту W3C, який містить тільки ті інструменти, які підтримують усі браузери. За статистикою, на сьогоднішній день (на момент написання кваліфікаційної роботи) більшість користувачів Інтернету використовують такі браузери [23]:

Google Chrome 79.7 %
Microsoft Edge 8.6 %
Mozilla Firefox 4.8 %
Apple Safari 3.9 %
Opera 2.2 %

Розробка системи має враховувати факт, що різні браузери можуть по-різному відображати вебсайти. Тому важливо забезпечити сумісність проєкту

з найбільш популярними браузерами - Google Chrome, Microsoft Edge та Mozilla Firefox. Оскільки ці браузери користуються широкою популярністю серед користувачів, то розробка проєкту з урахуванням їхніх особливостей дозволить досягти максимального охоплення аудиторії. Окрім того, необхідно забезпечити, щоб дизайн вебзастосунку був однаковим в усіх зазначених браузерах, а також щоб усі функціональні можливості працювали коректно без будь-яких помилок або затримок.

Застосунок повинен бути розроблений використовуючи технологію ASP.NET Core 6.0 та фреймворк Angular. Джерелом даних повинна бути СКБД, як сервер бази даних використовувати Microsoft Azure SQL Database.

Іноді користувачі можуть здійснювати помилкові дії, які призводять до некоректної роботи системи, наприклад, невірного введення даних в поля введення або пропуску обов'язкових полів. Тому важливо, щоб в системі була відповідна обробка помилок, яка дозволить користувачу отримати відповідні повідомлення про помилки та поради, як їх усунути. Це не тільки поліпшить користувацький досвід, але й допоможе зменшити кількість помилок, які можуть з'явитись. Крім того, дизайн повідомлень про помилки повинен відповідати загальному дизайну вебзастосунку, щоб не викликати плутанину та незручності для користувачів. Добре сформульовані та зрозумілі повідомлення про помилки є важливим елементом забезпечення якісного користувацького досвіду на сайті.

Після розробки, система повинна бути розгорнути, використовуючи хмарні технології Microsoft Azure, що дозволить забезпечити високу доступність та масштабованість. Розгортання системи відповідно до рекомендацій та кращих практик гарантує її надійність та стабільність. Крім того, хмарні технології Microsoft Azure надають можливість автоматичного масштабування ресурсів в залежності від потреб користувачів, що забезпечує високу продуктивність системи під час періодів збільшеного навантаження.

РОЗДІЛ 4.
РЕАЛІЗАЦІЯ СИСТЕМИ

4.1 Реалізація баз даних

Для розробки баз даних було використано інструмент Microsoft SQL Management Studio. Для коректного функціонування системи необхідно дві БД, одна з яких відображає всі залежності і включає всю необхідну інформацію з ОП та НП, яка має бути внесена до робочої програми, інша містить дані про користувачів та робочі програми. БД системи підтримки процесу розробки РПНД в ЗВО складається з 13 таблиць, структура яких зображена на діаграмі бази даних (див. додаток Б.1). Опис кожної з таблиць наведено у табл. 1.

Таблиця 1 – Короткий опис таблиць БД для ОП та НП

Номер	Таблиця	Опис
1	AreaOfExpertise	Таблиця для збереження галузей знань.
2	Competences	Таблиця для збереження інформації про компетентності.
3	EducationalPrograms	Таблиця для збереження інформації про освітні програми.
4	EducationalProgramsTypes	Таблиця для збереження рівнів підготовки за освітньою програмою.
5	Faculties	Таблиця для збереження інформації про факультети.
6	Universities	Таблиця для збереження

		інформації про університети.
7	ProgramResults	Таблиця для збереження інформації про програмні результати.
8	SelectiveBlocks	Таблиця для збереження інформації про вид дисципліни.
9	Specializations	Таблиця для збереження інформації про спеціальності.
10	SubjectCompetences	Таблиця для збереження відповідності між дисципліною та компетентністю.
11	SubjectProgramResults	Таблиця для збереження відповідності між дисципліною та програмним результатом.
12	Subjects	Таблиця для збереження інформації про дисципліни.
13	FinalControlTypes	Таблиця для збереження інформації про тип заключного контролю.

Таблиця dbo.AreaOfExpertise (див. табл. 2) містить номер та назву галузі знань, що в поєднані утворює унікальну галузь знань. Інформація про кожну компетентність зберігається в таблиці dbo.Competences (див. табл. 3) та містить назву, опис та ідентифікатор освітньої програми, до якої відноситься компетентність.

Таблиця 2 – Структура таблиці dbo.AreaOfExpertise

Атрибут	Тип	Опис
Id	uniqueidentifier	Ідентифікатор запису в таблиці.

Number	int	Номер галузі знань.
Name	nvarchar(100)	Назва галузі знань.

Таблиця 3 – Структура таблиці dbo.Competences

Атрибут	Тип	Опис
Id	uniqueidentifier	Ідентифікатор запису в таблиці.
Name	nvarchar(100)	Назва компетентності.
Description	nvarchar(MAX)	Опис компетентності.
EducationalProgramId	uniqueidentifier	Ідентифікатор освітньої програми.

Таблиця dbo.EducationalPrograms (див. табл. 4) зберігає необхідну інформацію, яка стосується освітніх програм, а саме назву, ідентифікатор факультету, ідентифікатор спеціальності та ідентифікатор рівня підготовки за освітньою програмою. Знайшовши записи в БД про факультет та спеціальність, можна отримати унікальні ідентифікатори галузі знань та університету до якого відноситься освітня програма, використовуючи які можна отримати необхідну інформацію.

Таблиця 4 – Структура таблиці dbo.EducationalPrograms

Атрибут	Тип	Опис
Id	uniqueidentifier	Ідентифікатор запису в таблиці.
Name	nvarchar(100)	Назва освітньої програми.
FacultyId	uniqueidentifier	Ідентифікатор факультету.
SpecializationId	uniqueidentifier	Ідентифікатор

		спеціальності.
EducationalProgramsTypeId	uniqueidentifier	Ідентифікатор рівня підготовки освітньої програми.

Необхідну інформацію про рівні підготовки за освітніми програмами, факультети, університети та спеціальності можна отримати із таблиць dbo.EducationalProgramsTypes, dbo.Faculties, dbo.Universities та dbo.Specializations відповідно (див. табл. 5 – 8).

Таблиця 5 – Структура таблиці dbo.EducationalProgramsTypes

Атрибут	Тип	Опис
Id	uniqueidentifier	Ідентифікатор запису в таблиці.
Name	nvarchar(100)	Назва рівня підготовки за освітньою програмою.

Таблиця 6 – Структура таблиці dbo.Faculties

Атрибут	Тип	Опис
Id	uniqueidentifier	Ідентифікатор запису в таблиці.
Name	nvarchar(100)	Назва факультету.
UniversityId	uniqueidentifier	Ідентифікатор університету.

Таблиця 7 – Структура таблиці dbo.Universities

Атрибут	Тип	Опис
Id	uniqueidentifier	Ідентифікатор запису в таблиці.
Name	nvarchar(100)	Назва університету

Таблиця 8 – Структура таблиці dbo.Specializations

Атрибут	Тип	Опис
Id	uniqueidentifier	Ідентифікатор запису в таблиці.
Number	int	Номер спеціальності.
Name	nvarchar(100)	Назва спеціальності
AreaOfExpertiseId	uniqueidentifier	Ідентифікатор галузі знань.

Інформація про програмні результати міститься в таблиці dbo.ProgramResults (див. табл. 9) та містить назву, опис та ідентифікатор освітньої програми, до якої відноситься програмний результат.

Таблиця 9 – Структура таблиці dbo.ProgramResults

Атрибут	Тип	Опис
Id	uniqueidentifier	Ідентифікатор запису в таблиці.
Name	nvarchar(100)	Назва програмного результату.
Description	nvarchar(MAX)	Опис програмного результату.
EducationalProgramId	uniqueidentifier	Ідентифікатор освітньої програми.

Таблиця dbo.Subjects (див. табл. 10) містить детальну інформацію про дисципліни, а саме назву, кількість кредитів, семестр, кількість годин, відведених різного типу занять, унікальний ідентифікатор освітньої програми. Для кожної дисципліни обрано тип заключного контролю та вид дисципліни, дана інформація зберігається в таблицях dbo.FinalControlTypes та dbo.SelectiveBlocks відповідно (див. табл. 11 – 12). Також таблиця dbo.Subjects містить унікальний ідентифікатор робочої програми (за наявності).

Таблиця 10 – Структура таблиці dbo.Subjects

Атрибут	Тип	Опис
Id	uniqueidentifier	Ідентифікатор запису в таблиці.
Name	nvarchar(100)	Назва дисципліни.
Credits	int	Кількість кредитів ECTS.
Semester	int	Семестр навчання.
LecturesHours	int	Кількість годин відведених для лекційних занять.
SeminarsHours	int	Кількість годин відведених для семінарських занять.
PracticalClassesHours	int	Кількість годин відведених для практичних занять.
LaboratoryClassesHours	int	Кількість годин відведених для лабораторних занять.
TrainingsHours	int	Кількість годин відведених для тренінгів.
ConsultationsHours	int	Кількість годин відведених для консультацій.
SelfWorkHours	int	Кількість годин відведених для самостійної роботи.
SelectiveBlockId	uniqueidentifier	Ідентифікатор виду дисципліни.
FinalControlTypeId	uniqueidentifier	Ідентифікатор типу заключного контролю.
EducationalProgramId	uniqueidentifier	Ідентифікатор освітньої програми.
WorkingProgramId	uniqueidentifier	Ідентифікатор робочої програми (за наявності).

Таблиця 11 – Структура таблиці dbo.FinalControlTypes

Атрибут	Тип	Опис
Id	uniqueidentifier	Ідентифікатор запису в таблиці.
Name	nvarchar(100)	Назва типу заключного контролю.

Таблиця 12 – Структура таблиці dbo.SelectiveBlocks

Атрибут	Тип	Опис
Id	uniqueidentifier	Ідентифікатор запису в таблиці.
Name	nvarchar(100)	Назва виду дисципліни

Для кожної дисципліни може бути обрано необхідні компетентності та програмні результати. Необхідні зв’язки зберігаються в таблицях dbo.SubjectCompetences та dbo.SubjectProgramResults (див. табл. 13 – 14).

Таблиця 13 – Структура таблиці dbo.SubjectCompetences

Атрибут	Тип	Опис
Id	uniqueidentifier	Ідентифікатор запису в таблиці.
SubjectId	uniqueidentifier	Ідентифікатор дисципліни.
CompetenceId	uniqueidentifier	Ідентифікатор компетентності.

Таблиця 14 – Структура таблиці dbo.SubjectProgramResults

Атрибут	Тип	Опис
Id	uniqueidentifier	Ідентифікатор запису в таблиці.
SubjectId	uniqueidentifier	Ідентифікатор дисципліни.
ProgramResultId	uniqueidentifier	Ідентифікатор компетентності.

Інформація про РПНД та користувачів системи зберігається в окремій БД, яка складається з 5 таблиць, структура яких зображена на діаграмі бази

даних (див. додаток Б.2). Опис кожної з таблиць наведено у табл. 15.

Таблиця 15 – Короткий опис таблиць БД для РПНД та користувачів

Номер	Таблиця	Опис
1	WorkingPrograms	Таблиця для збереження інформації про робочі програми.
2	AspNetUsers	Таблиця для збереження інформації про користувачів системи.
3	AspNetRoles	Таблиця ролей системи.
4	AspNetUserRoles	Таблиця для збереження відповідності між користувачем та його роллю.

Таблиця dbo.WorkingPrograms (див. табл. 16) містить необхідну інформацію про робочу програму дисципліни, а саме назву, назву файлу (ідентифікатор) робочої програми, який зберігається в Azure Blob Storage, її доступність та ідентифікатор користувача який створив та підтвердив робочу програму.

Таблиця 16 – Структура таблиці dbo.WorkingPrograms

Атрибут	Тип	Опис
Id	uniqueidentifier	Ідентифікатор запису в таблиці.
Name	nvarchar(300)	Назва робочої програми.
FileName	nvarchar(100)	Назва файлу робочої програми, розміщеного в Azure Blob Storage.
IsAvailable	bit	Прапорець, який показує чи є робоча програма доступною.
SubjectId	uniqueidentifier	Ідентифікатор дисципліни.
CreatedById	uniqueidentifier	Ідентифікатор користувача, який

		створив робочу програму.
ApprovedById	uniqueidentifier	Ідентифікатор користувача, який підтвердив робочу програму.

Таблиця AspNetUsers (див. табл. 17) містить необхідну інформацію про користувачів системи. Наявні ролі зберігаються в таблиці dbo.AspNetRoles (див. табл. 18), а зв'язок між користувачем та роллю в таблиці dbo.AspNetUserRoles (див. табл. 19).

Таблиця 17 – Структура таблиці dbo.AspNetUsers

Атрибут	Тип	Опис
Id	uniqueidentifier	Ідентифікатор запису в таблиці.
FirstName	nvarchar(256)	Ім'я користувача.
LastName	nvarchar(256)	Прізвище користувача.
IsFirstPasswordChanged	bit	Прапорець, який показує чи було змінено пароль.
Email	nvarchar(256)	Електронна пошта.
NormalizedEmail	nvarchar(256)	Нормалізоване значення електронної пошти.
PasswordHash	nvarchar(MAX)	Геш паролю.

Таблиця 18 – Структура таблиці dbo.AspNetRoles

Атрибут	Тип	Опис
Id	uniqueidentifier	Ідентифікатор запису в таблиці.
Name	nvarchar(256)	Назва ролі користувача в системі.
NormalizedName	nvarchar(256)	Нормалізоване значення назви ролі.

Таблиця 19 – Структура таблиці dbo.AspNetUserRoles

Атрибут	Тип	Опис
UserId	uniqueidentifier	Ідентифікатор користувача.
RoleId	uniqueidentifier	Ідентифікатор ролі.

4.2 Реалізація серверної архітектури

В якості шаблону архітектури програмного забезпечення було обрано трирівневу серверну архітектуру. Реалізація такої архітектури в системі забезпечує зменшення зв'язності між різними компонентами та підвищення їх незалежності. Крім того, це дозволяє забезпечити більш гнучку та масштабовану систему, яка може працювати з різними джерелами даних та іншими системами.

На рівні DAL (Data Access Layer) здійснюється доступ до бази даних, де зберігається інформація про об'єкти, які використовуються в системі. Для зручності роботи з даними, на рівні DAL створюються класи-моделі (див. додаток В.1), які відображають базу даних і представляють її в коді програми. Ці моделі можуть містити як окремі таблиці, так і зв'язки між таблицями, включаючи зв'язки багато-до-багатьох. Моделі на рівні DAL використовуються для взаємодії з базою даних на рівні коду. Вони надають зручний і простий спосіб доступу до даних і дозволяють проводити операції з даними, такі як вставка, оновлення, видалення та пошук без необхідності вручну писати складні SQL запити. Крім того, використання моделей на рівні DAL дозволяє зменшити залежність від бази даних і спростити розробку додатку. Це означає, що якщо у майбутньому потрібно буде змінити базу даних або її структуру, то це не вплине на код програми, якщо моделі на рівні DAL будуть належним чином оновлені.

На рівні BLL (Business Logic Layer) знаходяться всі класи, що стосуються бізнес-логіки, такі як класи для обробки даних, перетворення

даних, валідації даних та інші (див. додаток В.2). Ці класи служать для забезпечення відповідності бізнес-правил та дозволяють підтримувати консистентність даних у системі. Одне з основних завдань BLL - це виконання всіх операцій з даними, необхідних для відповіді на запити користувачів або інших систем. Цей рівень забезпечує повну інкапсуляцію бази даних і забезпечує безпеку взаємодії між DAL та вищими рівнями. Всі дані, які були отримані на рівні DAL, оброблюються на рівні BLL, щоб перетворити їх у відповідний формат та забезпечити потрібну логіку.

На рівні PL (Presentation Layer) класи моделі використовуються для відображення даних в інтерфейсі користувача (див. додаток В.3). Ці класи можуть містити як дані, так і логіку пов'язану з їх відображенням. Вони зазвичай представляють собою прості POCO (Plain Old CLR Object) класи з властивостями, які мають відповідати полям бази даних. Ці класи можуть бути використані для передачі даних між інтерфейсом користувача та BLL, а також для збереження даних у базі даних через DAL.

У табл. 20 – 26 наведено основні методи управління даними застосунку та описано функції, які вони виконують.

Таблиця 20 – Кінцеві точки (endpoints) класу AccountController

Номер	**Метод**	**Опис**
1	LoginAsync	Авторизація до системи.
2	LogoutAsync	Вихід із системи.
3	GetByIdAsync	Отримання детальної інформації про користувача, використовуючи його ідентифікатор.
4	GetByEmailAsync	Отримання детальної інформації про користувача, використовуючи його електронну пошту.
5	GetAuthorizedAsync	Отримання детальної інформації

		про авторизованого користувача.
6	GetAllAsync	Отримання інформації про всіх користувачів системи.
7	CreateAsync	Створення нового користувача.
8	ChangePasswordAsync	Зміна паролю.

Таблиця 21 – Кінцеві точки (endpoints) класу SubjectController

Номер	**Метод**	**Опис**
1	GetAllAsync	Отримання інформації про всі дисципліни.
2	GetAsync	Отримання детальної інформації про дисципліну за ідентифікатором.
3	CreateAsync	Створення нової дисципліни.
4	UpdateAsync	Оновлення інформації про дисципліну.

Таблиця 22 – Кінцеві точки (endpoints) класу EducationalProgramController

Номер	**Метод**	**Опис**
1	GetAllAsync	Отримання інформації про всі освітні програми.
2	GetAsync	Отримання детальної інформації про освітню програму за ідентифікатором.
3	CreateAsync	Створення нової освітньої програми.
4	UpdateAsync	Оновлення інформації про освітню програму.

Таблиця 23 – Кінцеві точки (endpoints) класу CompetenceController

Номер	Метод	Опис
1	GetAllAsync	Отримання інформації про всі компетентності.
2	CreateAsync	Створення нової компетентності.
3	DeleteAsync	Видалення компетентності.

Таблиця 24 – Кінцеві точки (endpoints) класу ProgramResultController

Номер	Метод	Опис
1	GetAllAsync	Отримання інформації про всі програмні результати.
2	CreateAsync	Створення нового програмного результату.
3	DeleteAsync	Видалення програмного результату.

Таблиця 25 – Кінцеві точки (endpoints) класу WorkingProgramController

Номер	Метод	Опис
1	GetAllAsync	Отримання інформації про всі РПНД.
2	GetAsync	Отримання детальної інформації про РПНД за ідентифікатором.
3	CreateAsync	Створення нової РПНД.
4	GenerateTemplateAsync	Генерація системою шаблону РПНД із автоматично заповненими полями.
5	GetWorkingProgramFileAsync	Завантаження файлу РПНД.

6	ApproveWorkingProgramAsync	Підтвердження РПНД.
7	DeleteAsync	Видалення РПНД.
8	RejectAsync	Відхилення РПНД.

Таблиця 26 – Кінцеві точки (endpoints) класу LookupController

Номер	**Метод**	**Опис**
1	GetAreaOfExpertiseAsync	Отримання переліку галузей знань.
2	GetSpecializationsAsync	Отримання переліку спеціальностей для обраної за ідентифікатором галузі знань.
3	GetUniversitiesAsync	Отримання переліку університетів.
4	GetFacultiesAsync	Отримання переліку факультетів для обраного за ідентифікатором університету.
5	GetEducationalProgramsTypesAsync	Отримання переліку рівнів підготовки освітніх програм.
6	GetEducationalProgramsAsync	Отримання переліку освітніх програм.
7	GetCompetencesAsync	Отримання переліку компетентностей для обраної за ідентифікатором освітньої програми.

8	GetProgramResultsAsync	Отримання переліку програмних результатів для обраної за ідентифікатором освітньої програми.
9	GetFinalControlTypesAsync	Отримання переліку типів заключного контролю.
10	GetSelectiveBlocksAsync	Отримання переліку видів дисципліни.
11	GetSubjectsAsync	Отримання переліку дисциплін.
12	GetSubjectsByEpIdAsync	Отримання переліку дисциплін за ідентифікатором освітньої програми.

Swagger [24] – це набір інструментів, який дозволяє створювати, описувати та відображати API. Під час розробки застосунку з використанням REST API для полегшення подальшого розширення та збільшення читабельності API було згенеровано документацію за допомогою Swagger (див. рис. 9). Документація Swagger включає опис кожного методу API, доступного в системі, а також інформацію про параметри, типи повернення та коди помилок, що повертаються. Це дозволяє розробникам та користувачам легко зрозуміти, як взаємодіяти з системою та виконувати необхідні операції. Загальна документація Swagger сприяє ефективнішому спілкуванню між розробниками та покращує загальну розуміння системи.

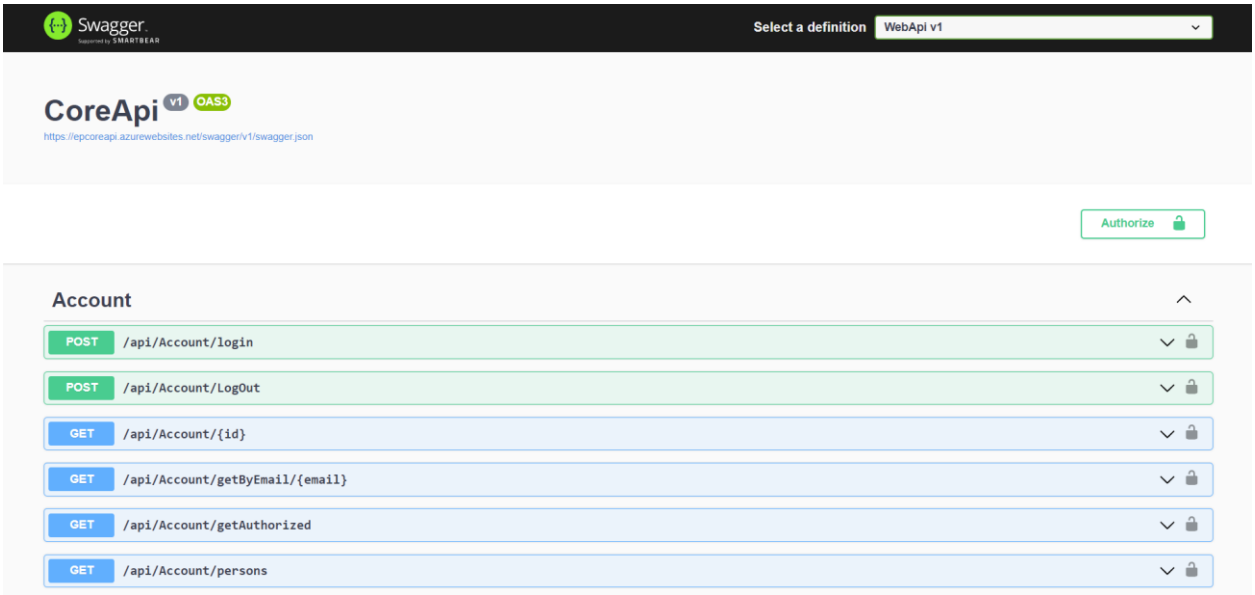


Рисунок 9 – Документація Swagger

4.3 Генерація файлу РПНД

Система має вбудований механізм, який надає викладачам можливість генерації файлу-шаблону робочої програми для, обраної за ідентифікатором, навчальної дисципліни. Генерація файлу-шаблону здійснюється автоматично за певним алгоритмом на основі даних, взятих із БД.

Програмний код методу наведено в додатку Г. На початку необхідно отримати всі дані із БД, які пов’язані із обраною навчальною дисципліною. Для цього викликається метод, що повертає модель з необхідними даними, а саме назву дисципліни, кількість кредитів ECTS, номер семестру, кількість годин відведених на різні типи занять (лекції, семінари, практичні заняття, лабораторні заняття, тренінги, консультації та самостійна робота), вид дисципліни, тип заключного контролю, інформація про освітню програму (назва, рівень підготовки, спеціальність, галузь знань, назва факультету та університету), перелік програмних результатів та компетентностей.

На наступному кроці необхідно отримати файл-шаблон РПНД, який знаходиться в сховищі Azure (Azure Blob Storage). Для цього використовується клас FileProvider, який забезпечує комунікацію із функцією Azure

(GetBlobFile), яка звертається до сховища та повертає необхідний файл-шаблон (програмний код наведено в додатку Д.1). Нижче наведено програмний код методу GetFileAsync класу FileProvider.

```
public async Task<Result<BlobFileGetModel>> GetFileAsync(string
fullFileName)
        {
            using var client = new HttpClient();
            var url =
_settings.AzureFunctionGetFileUrl.Replace("{fullFileName}",
fullFileName);

            var result = await client.GetAsync(url);
            if (result.StatusCode != HttpStatusCode.OK)
            {
                return
Result.NotFound<BlobFileGetModel>(BlErrors.FileNotFound);
            }

            var responseContent = await result.Content.ReadAsStringAsync();
            var model =
JsonConvert.DeserializeObject<BlobFileGetModel>(responseContent);

            return Result.Success(model);
        }
```

Також клас FileProvider забезпечує комунікацію із функцією Azure (PostBlobFile), яка завантажує у сховище готовий файл РПНД (програмний код наведено в додатку Д.2). Нижче наведено програмний код методу PostFileAsync класу FileProvider.

```
public async Task<Result<Guid>> PostFileAsync(IFormFile file)
        {
            var blobFileId = Guid.NewGuid();
            using var stream = new MemoryStream();
            await file.CopyToAsync(stream);

            var request = new
            {
                FileName = blobFileId.ToString(),
```

```
            Contents = stream.ToArray(),
        };
        var stringContent = new
StringContent(JsonConvert.SerializeObject(request), Encoding.UTF8,
"application/json");

        using var client = new HttpClient();
        var result = await
client.PostAsync(_settings.AzureFunctionPostFileUrl, stringContent);
        if (result.StatusCode != HttpStatusCode.OK)
        {
            return Result.Failure<Guid>(BlErrors.FileUploadFailed);
        }

        return Result.Success(blobFileId);
    }
```

Файл-шаблон РПНД містить теги у вигляді <НАЗВАТЕГУ>, які необхідно замінити інформацією із БД для обраної дисципліни. Для цього метод ConvertToDictionary формує словник, ключем якого є тег, а значенням є відповідна інформація із БД. Далі, використовуючи дану структуру, виконується заміна тегів у файлі. Приклад згенерованого системою шаблону наведено в додатку Е.

Цей механізм значно полегшує роботу викладачам та іншим співробітникам, які займаються розробкою РПНД, і дозволяє ефективно використовувати час та зусилля при створенні таких файлів.

4.4 Особливості розгортання системи

Система була розгорнута в Microsoft Azure – хмарному сервісі, що надає можливості розміщення та керування різноманітними додатками в хмарному середовищі [17].

На рис. 10 зображено діаграму взаємодії компонентів системи, яка відображає взаємодію між ними та сервісами системи, їхні процеси та залежності. Це графічне зображення надає візуалізацію архітектури та

допомагає розуміти взаємодію між компонентами. Перелік ресурсів системи, розгорнутої в хмарному середовищі Microsoft Azure, зображено на рис. 11.

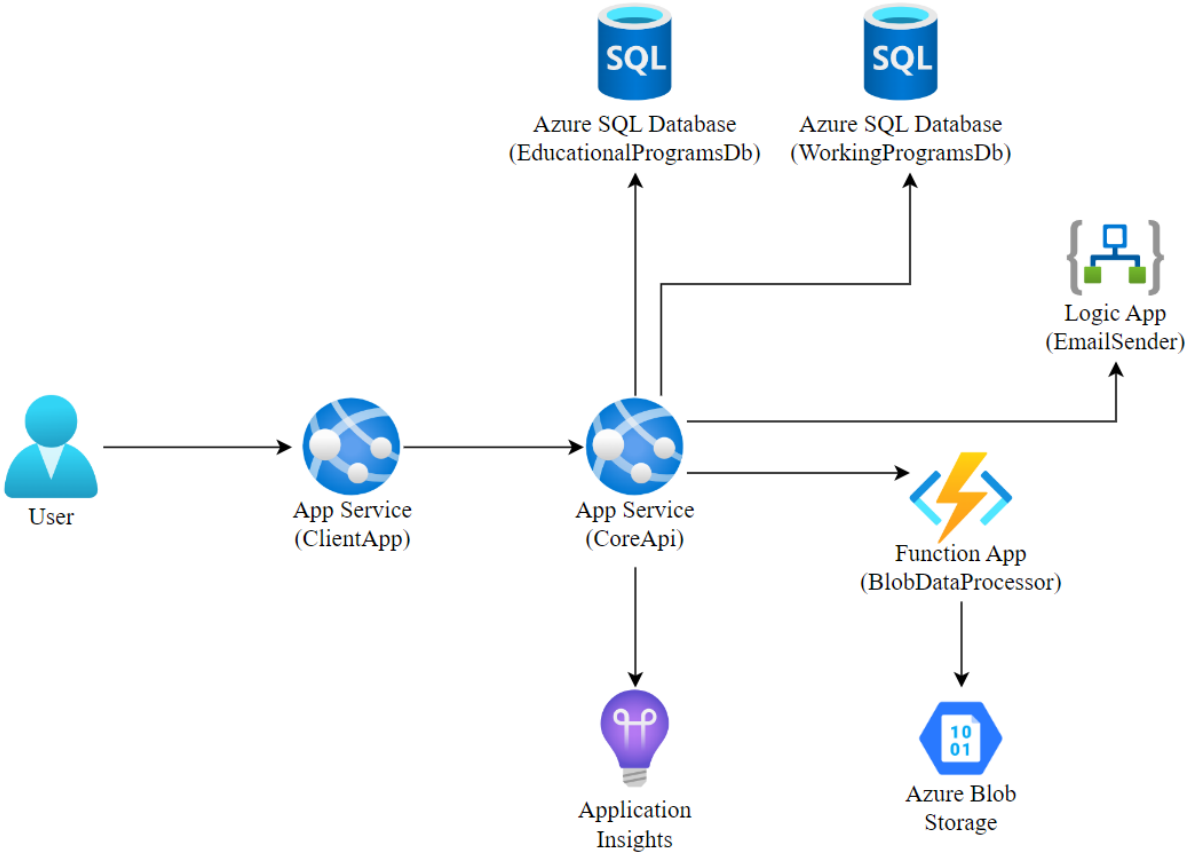


Рисунок 10 – Взаємодія компонентів системи, розгорнутої в Microsoft Azure

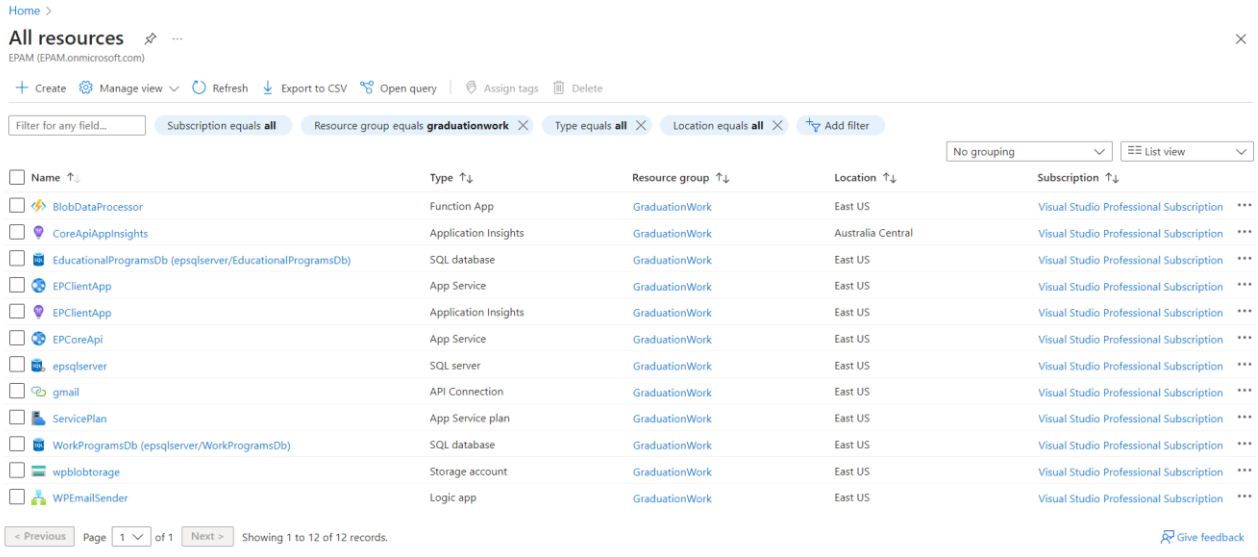


Рисунок 11 – Перелік ресурсів розгорнутої системи

БД системи знаходяться на сервері (Azure SQL Server), доступ до якого надається лише після успішної автентифікації. На рис. 12 зображено процес автентифікації для доступу до БД EducationalProgramsDb. Приклад виконання запиту для отримання інформації про компетентності зображено на рис. 13.

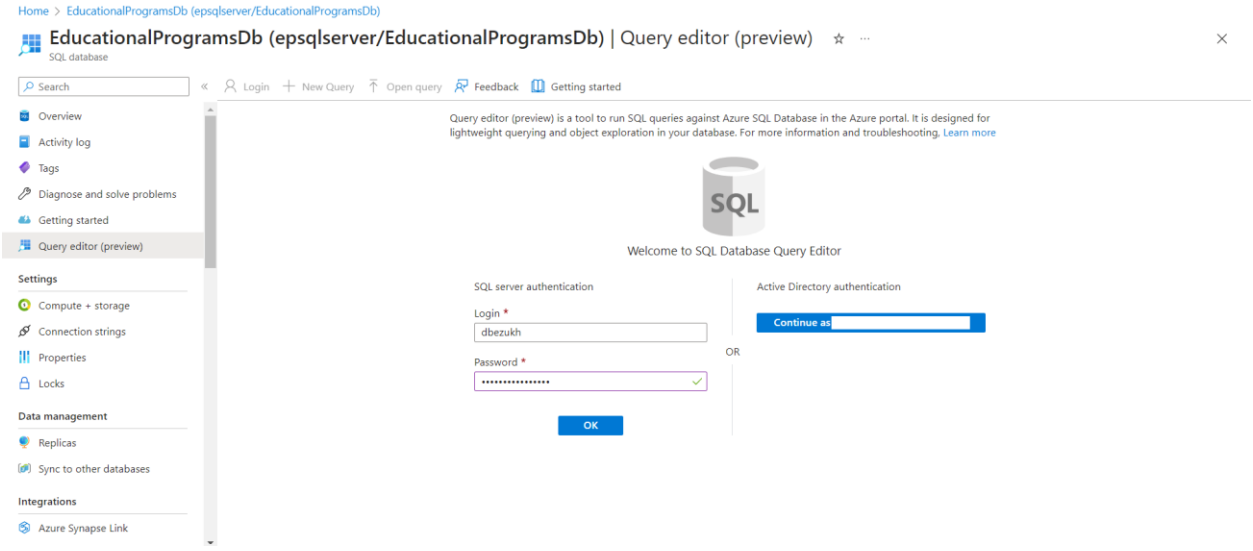


Рисунок 12 – Автентифікація до SQL Database

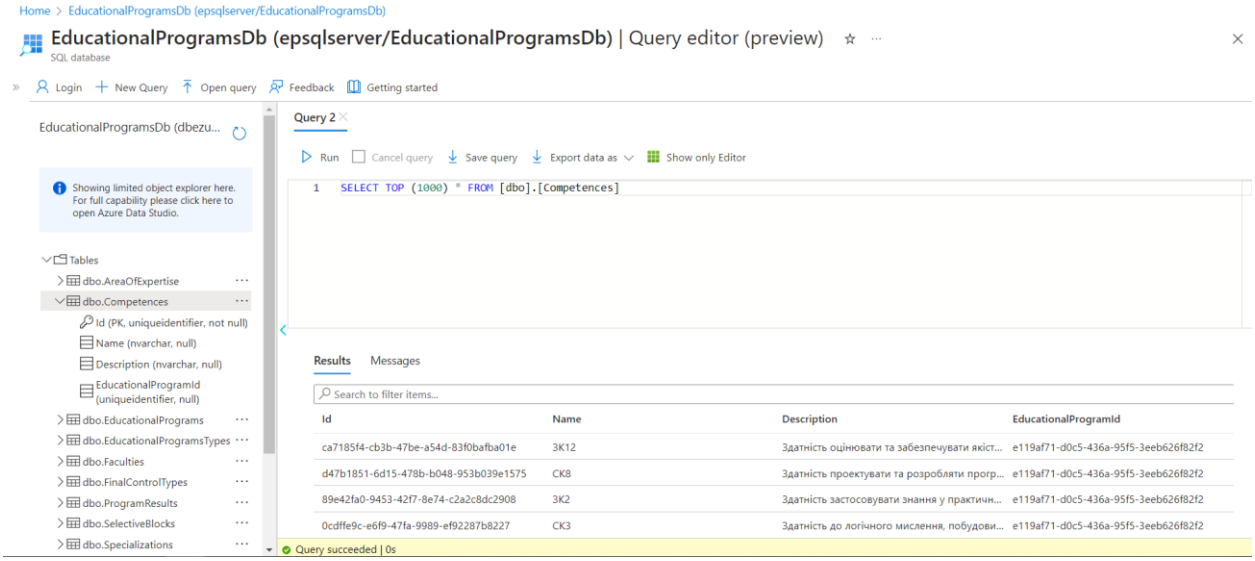


Рисунок 13 – Приклад запиту до БД

Для сповіщення користувачів електронною поштою використовується компонент WPEmailSender. На рис. 14 наведено структуру тіла запиту, який

повинен містити електронну адресу користувача, якому необхідно надіслати лист, тему листа та текст повідомлення. У відповідь на коректний запит, компонент надсилає електронний лист, схема якого наведена на рис. 15.

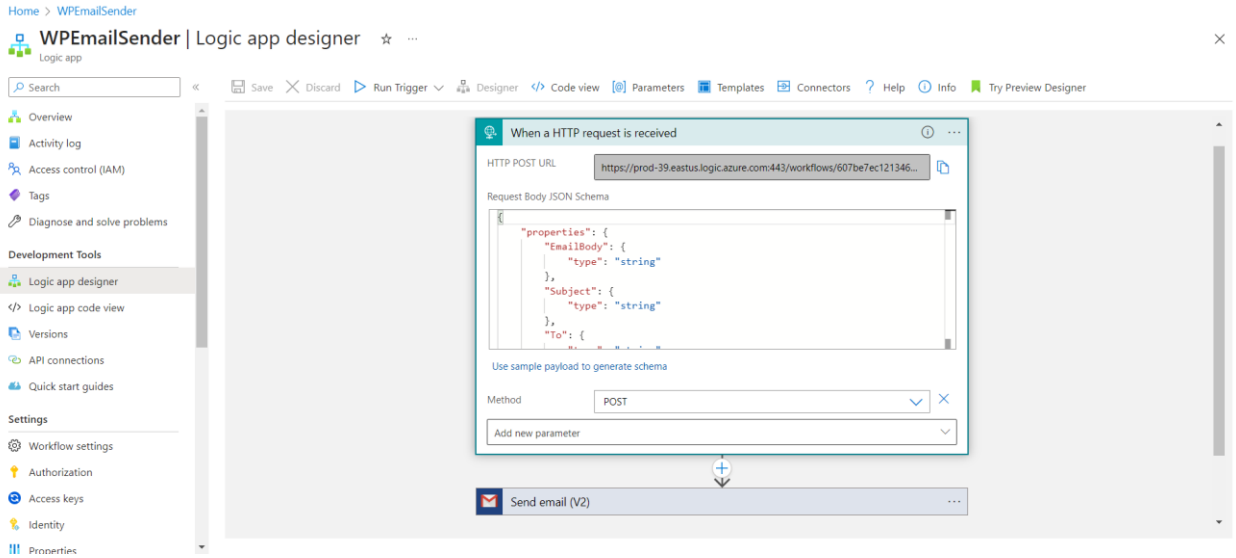


Рисунок 14 – Компонент WPEmailSender (вимоги до вхідного запиту)

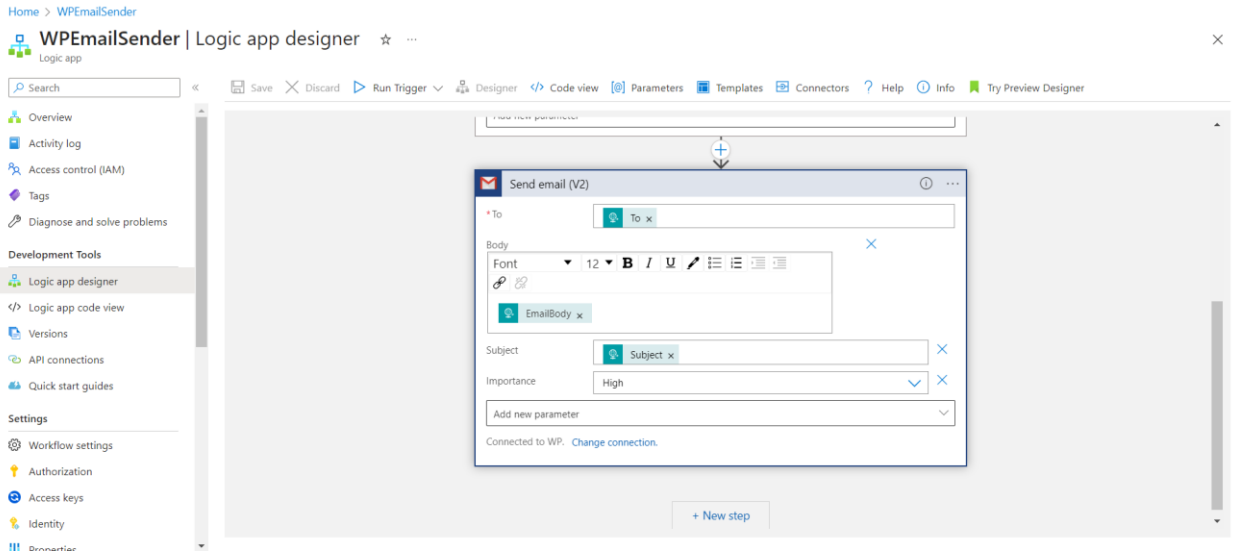


Рисунок 15 – Компонент WPEmailSender (обробка вхідного запиту)

Також даний компонент містить історію всіх запитів та детальну інформацію про них (див. рис. 16).

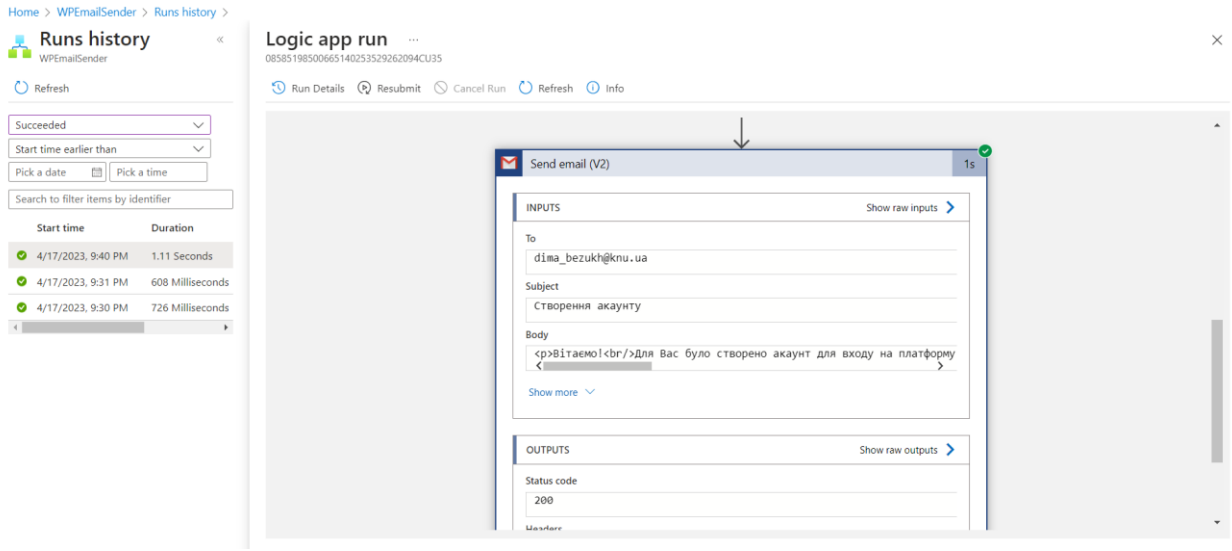


Рисунок 16 – Компонент WPEmailSender (приклад успішного виконання запиту)

Для взаємодії сервера та сховища, в якому зберігаються шаблон та файли РПНД, використовується функція Azure BlobDataProcessor. На рис. 17 зображено загальну інформацію про даний компонент.

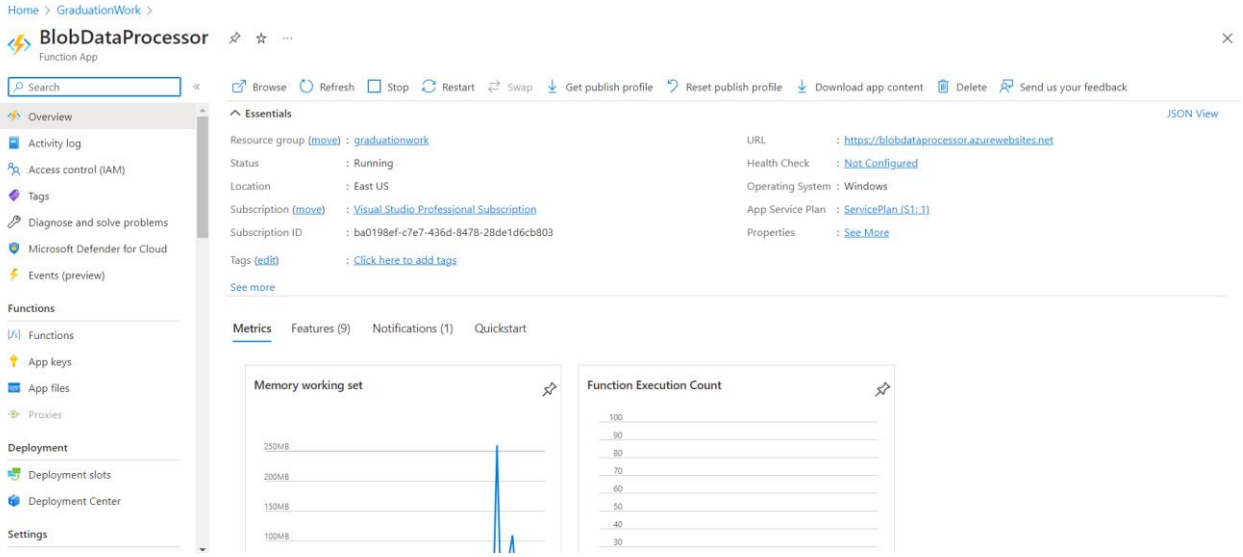


Рисунок 17 – Загальна інформація про компонент BlobDataProcessor

Для взаємодії функцій Azure та сховища із файлами, необхідно вказати підключення та назву контейнера. Ця інформація задається через конфігурацію (див. рис 18).

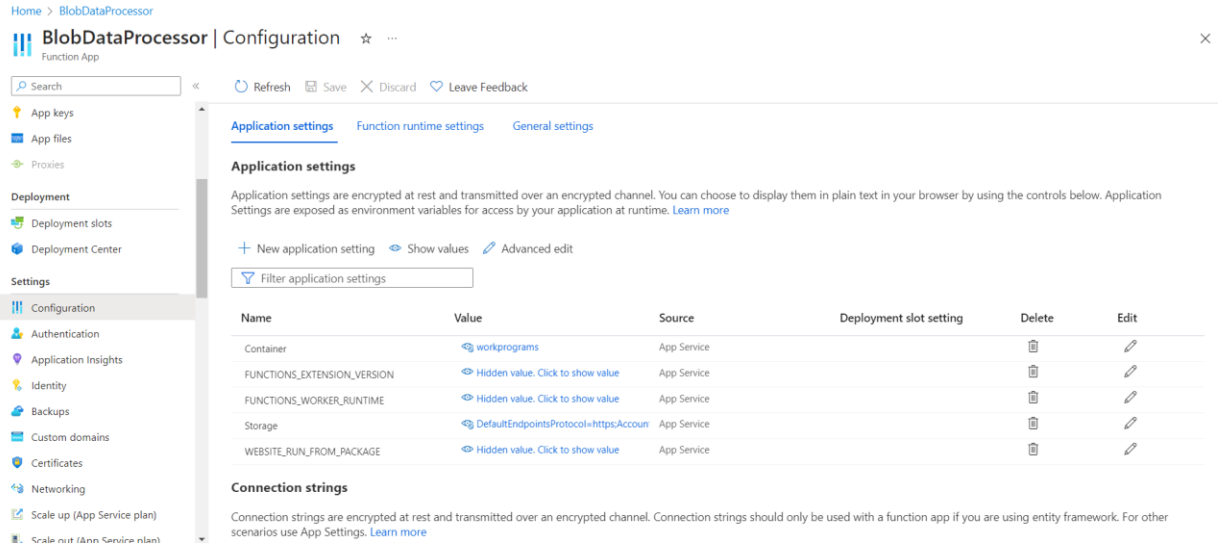


Рисунок 18 – Конфігурація компонента BlobDataProcessor

На рис. 19 зображено приклад запиту до функції Azure GetBlobFile, яка, в якості вхідного параметра, приймає повну назву файлу, який необхідно отримати зі сховища.

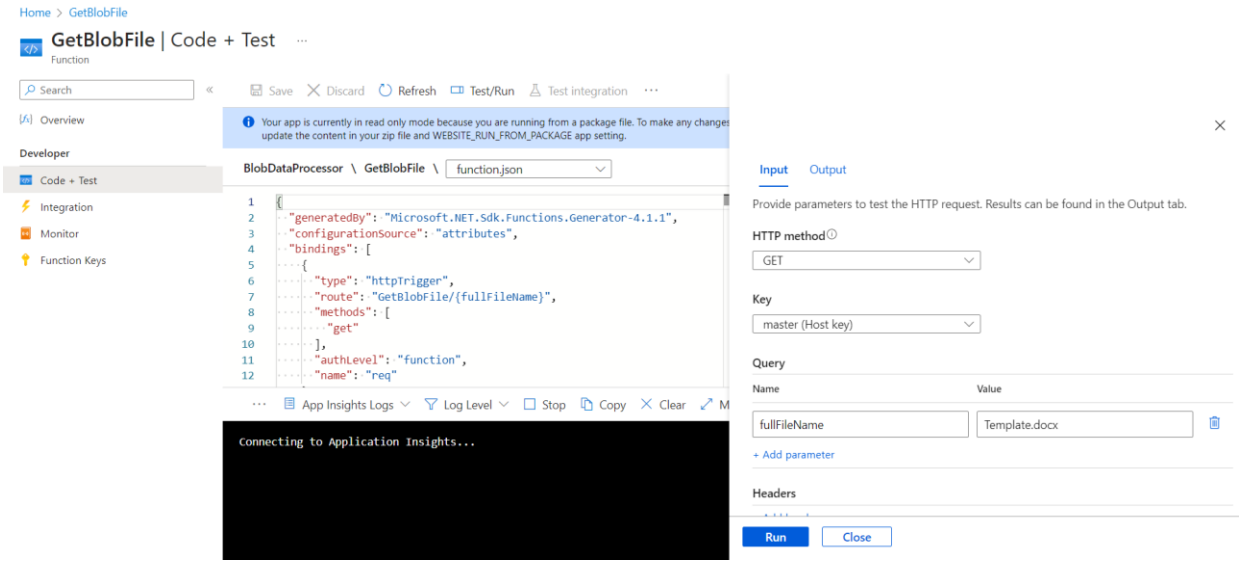


Рисунок 19 – Приклад запиту до функції Azure компонента BlobDataProcessor

В якості результату, функція Azure GetBlobFile повертає бінарне представлення потрібного файлу (див. рис. 20) або повідомлення про помилку, якщо файл не знайдено.

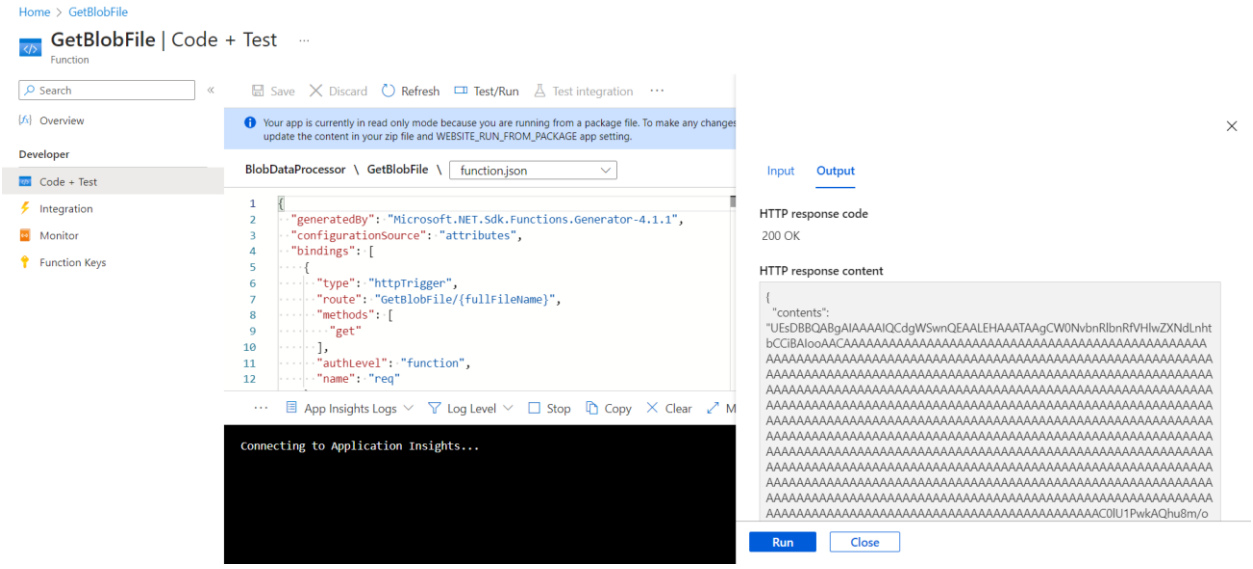


Рисунок 20 – Результат виконання запиту до функції Azure компонента BlobDataProcessor

Компонент BlobDataProcessor дозволяє переглядати різні метрики. Наприклад, середній час обробки запиту та кількість запитів у певний проміжок часу (див. рис. 21).

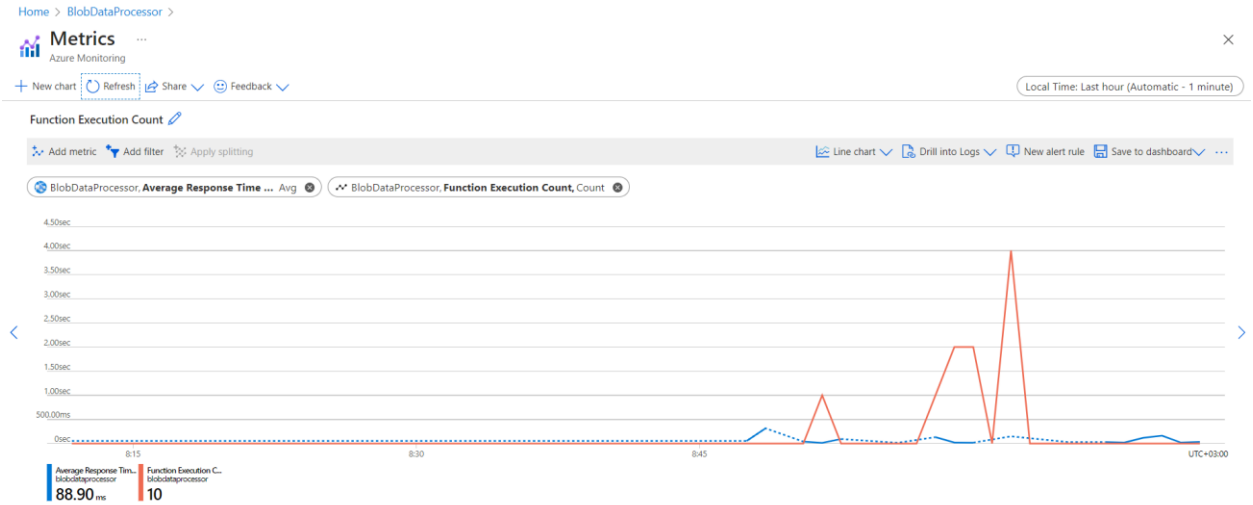


Рисунок 21 – Метрики компонента BlobDataProcessor

Для зберігання файлів використовується Azure Blob Storage. В ньому міститься файл-шаблон РПНД із тегами та завантажені у систему файли РПНД (див. рис. 22).

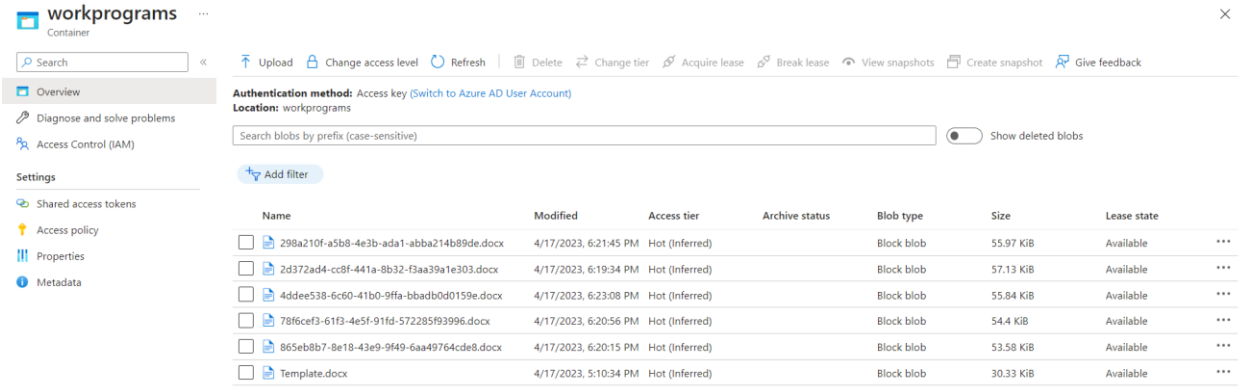


Рисунок 22 – Вміст контейнера workprograms сховища wpblobstorage

Ключовою частиною системи є сервер, який розгорнутий у вебдодатку EPCoreApi. Через нього відбувається взаємодія з іншими компонентами, такими як EducationalProgramsDb, WorkingProgramsDb, BlobDataProcessor та WPEmailSender. Для цього в конфігурацію потрібно додати підключення до БД та необхідні Urls (див. рис. 23).

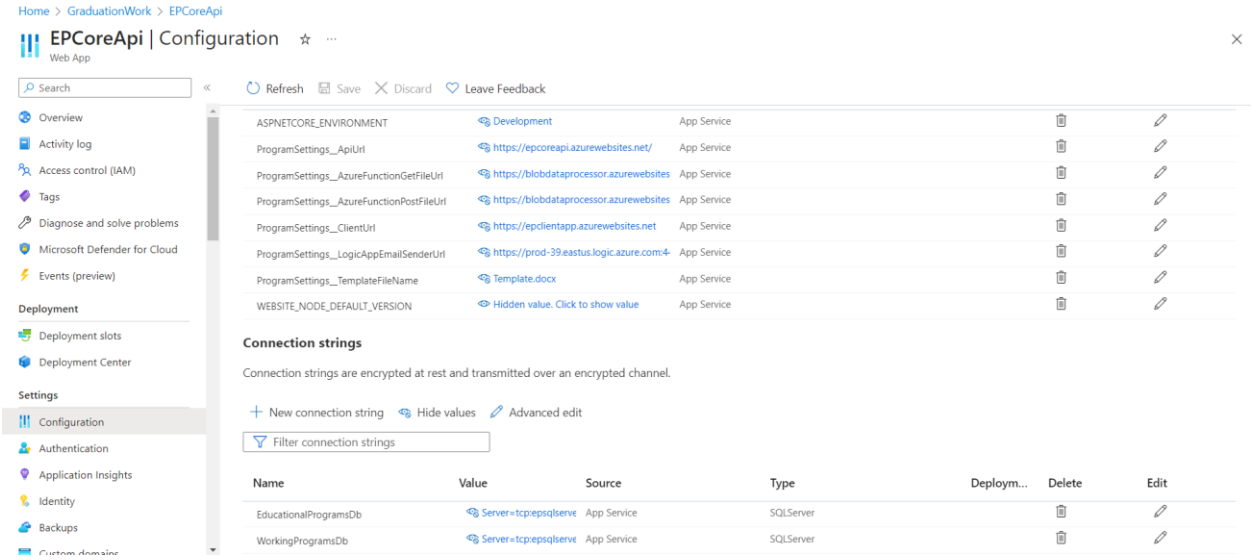


Рисунок 23 – Конфігурація компонента EPCoreApi

Для аналізу роботи сервера в систему було інтегровано інструмент моніторингу та аналізу продуктивності Azure App Insights. Даний компонент забезпечує повний контроль над роботою сервера, дозволяючи відстежувати різноманітні метрики, такі як кількість запитів, час відповіді, витрати ресурсів, помилки та багато іншого (див. рис 24). Це дозволяє оперативно виявляти та вирішувати проблеми продуктивності, вдосконалювати роботу сервера та забезпечувати надійну та ефективну роботу системи. На рис. 25 зображено детальну інформацію про запит для створення РПНД.

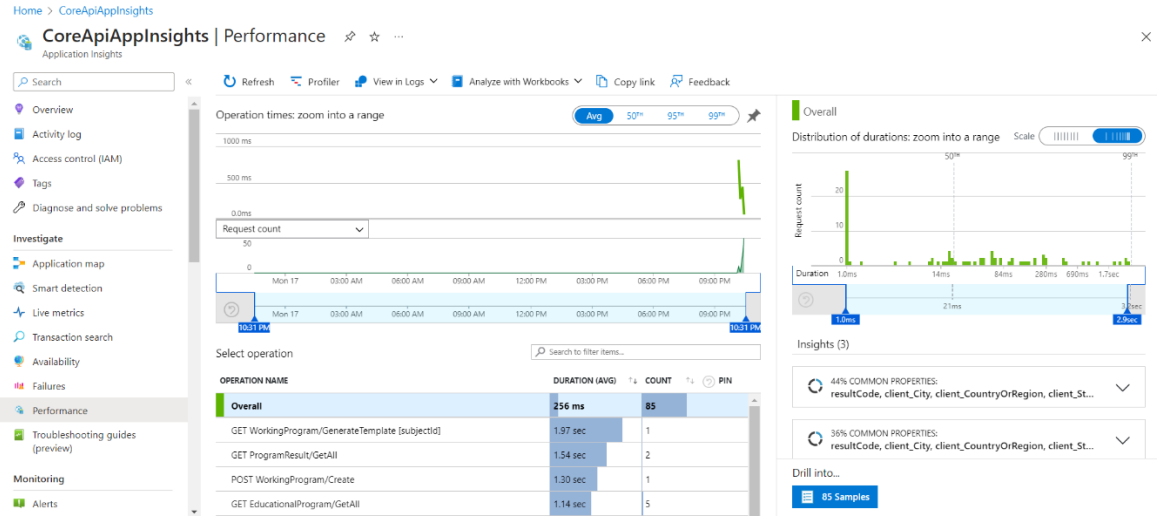


Рисунок 24 – Компонент CoreApiAppInsights (продуктивність запитів)

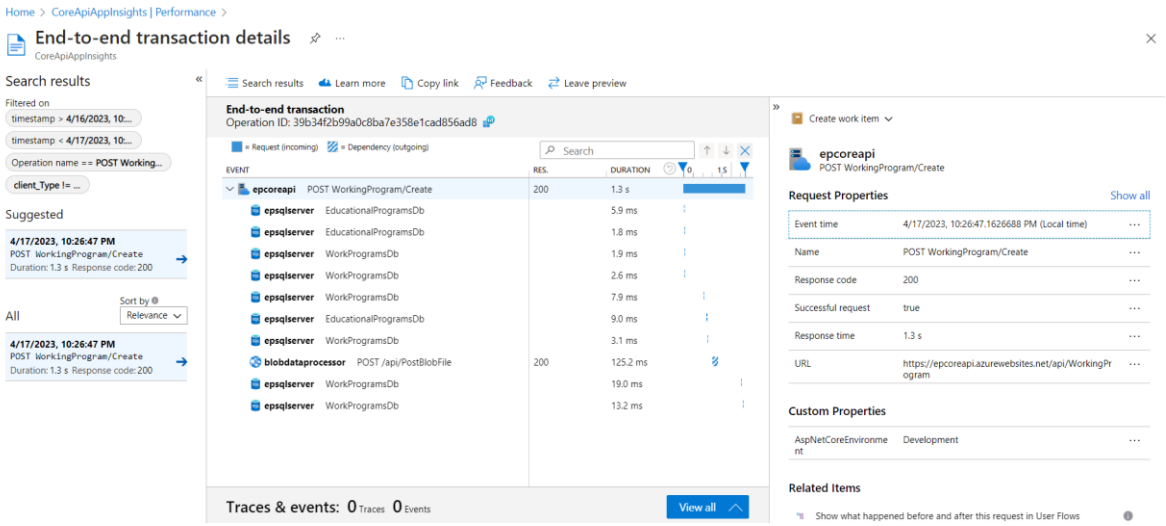


Рисунок 25 – Компонент CoreApiAppInsights (детальна інформація про запит)

У системі налаштовано горизонтальне масштабування App Service Plan за такими правилами: ресурс автоматично масштабується на 1 екземпляр, якщо відсоток використання процесорного часу (CPU) перевищує 75%. Також, якщо відсоток використання CPU менший за 25%, ресурс автоматично зменшується на 1 екземпляр, як мінімум до 1 екземпляра (див. рис. 26). Це дозволяє системі адаптуватись до змін навантаження і забезпечувати оптимальні ресурси для виконання в залежності від потреб.

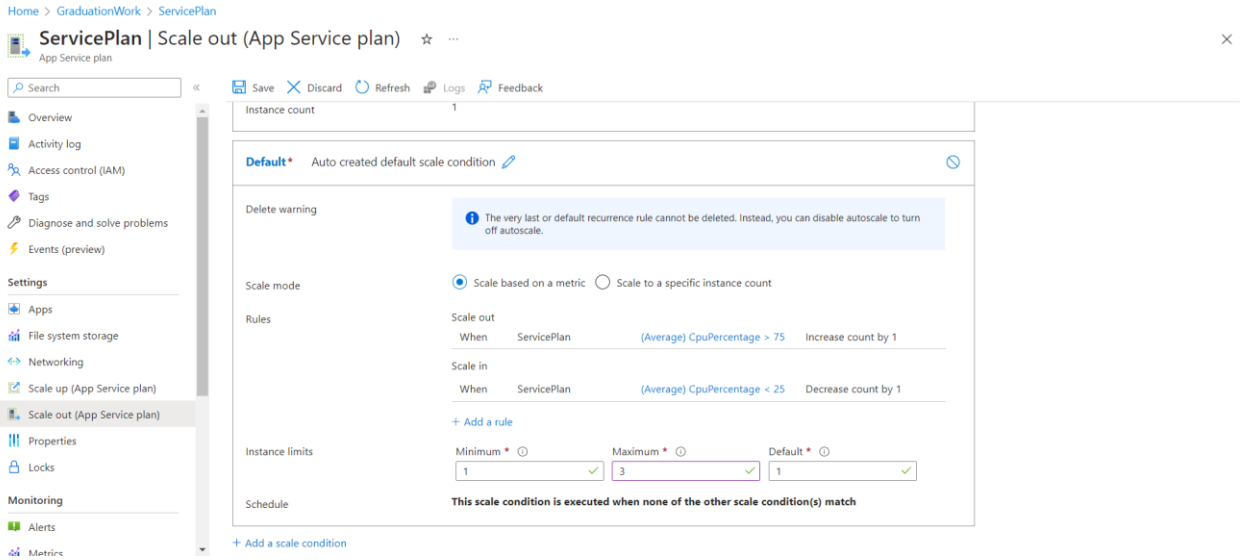


Рисунок 26 – Налаштування горизонтального масштабування App Service Plan

РОЗДІЛ 5.
ІНСТРУКЦІЯ КОРИСТУВАЧА

5.1 Інструкція неавторизованого користувача

Зайшовши вперше на сайт, перед користувачем відображається сторінка із переліком дисциплін (див. рис 27) та короткою інформацією про них. Наявна можливість виконувати пошук за такими параметрами: назва дисципліни, освітня програма, вид дисципліни (див. рис. 28). Система включає в себе пагінацію, що робить процес перегляду інформації більш зручним та простим. Завдяки даній функції, перелік дисциплін можна переглядати сторінками, при цьому на кожній сторінці буде відображатись по 7 дисциплін, що сприяє покращенню організації і структуруванню представленої інформації. Для зручності навігації, в лівій частині екрану знаходиться меню, що дозволяє швидко та зручно знаходити потрібну інформацію.

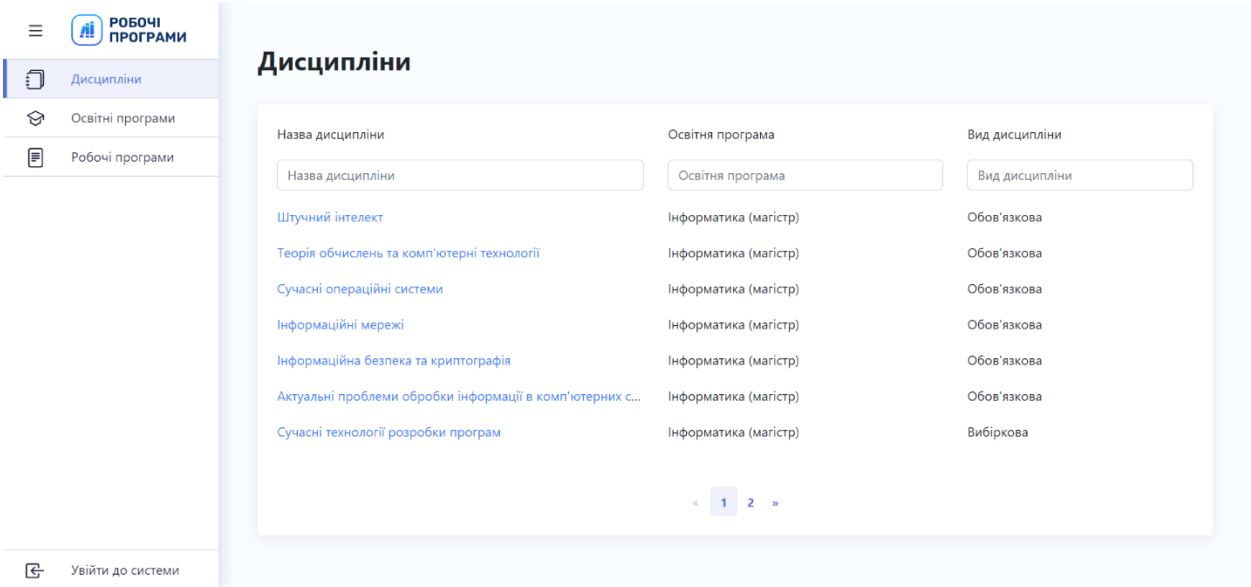


Рисунок 27 – Перелік дисциплін

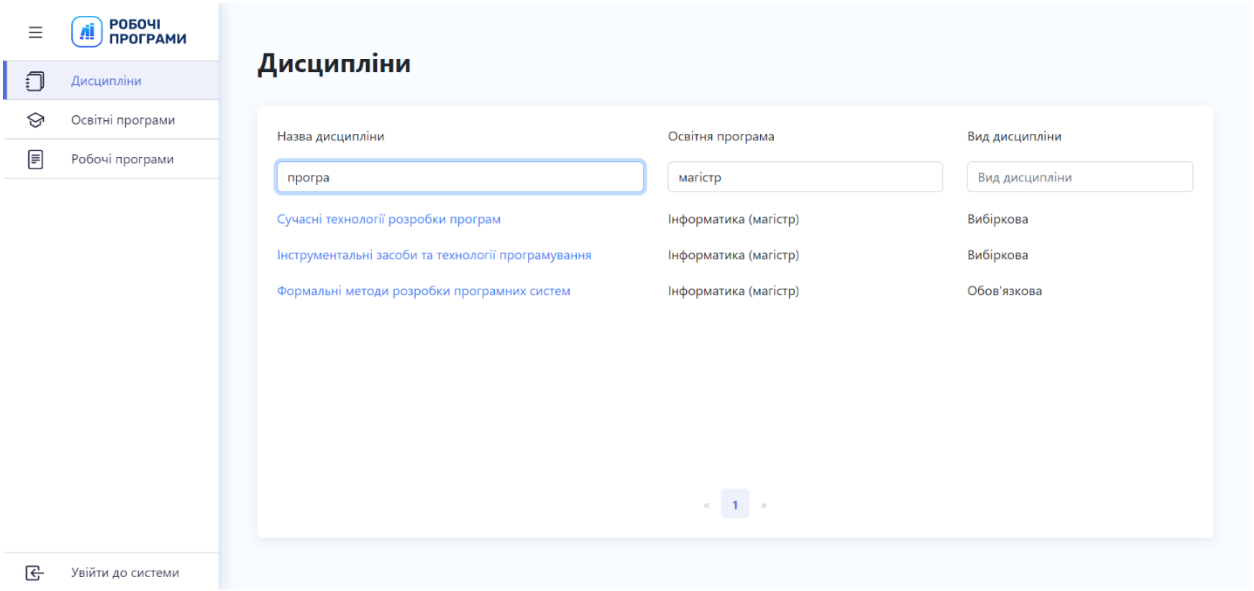


Рисунок 28 – Пошук дисциплін за параметрами

Перейшовши на сторінку дисципліни, користувач може переглянути коротку інформацію про освітню програму, форму заключного контролю, кількість кредитів ECTS, семестр, вид дисципліни, кількість годин відведених на різні типи роботи, інформацію про компетентності та програмні результати (див. рис. 29 – 30).

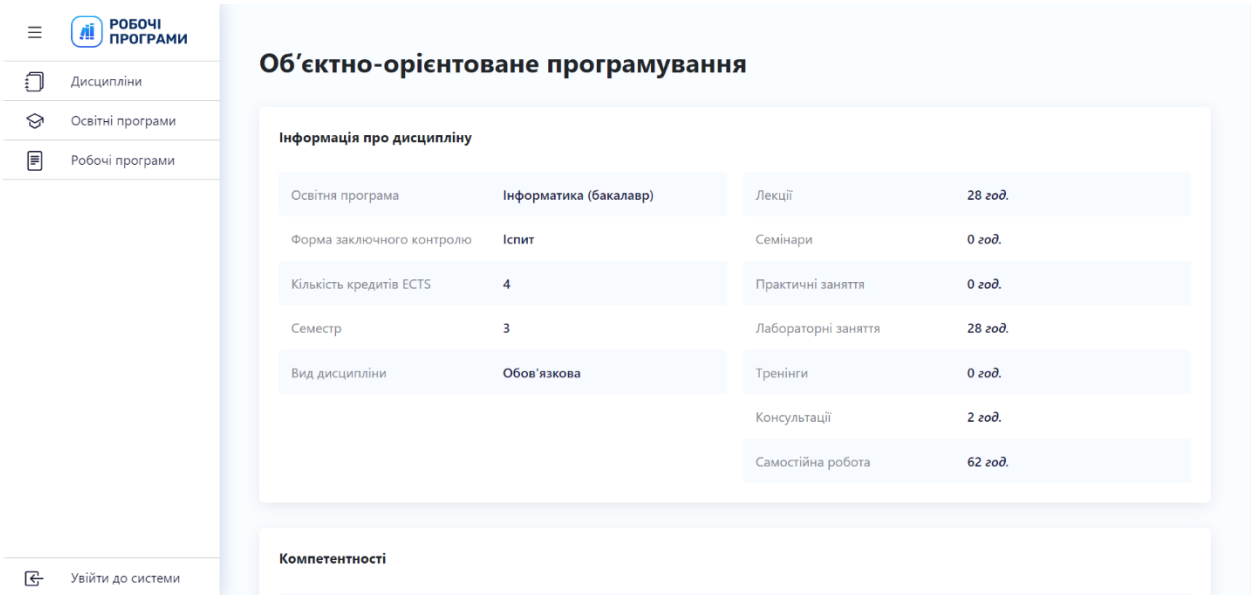


Рисунок 29 – Сторінка детальної інформації про дисципліну (основна інформація)

Рисунок 30 – Інформація про компетентності та програмні результати обраної дисципліни

Відкривши вкладинку «Освітні програми», перед користувачем відображається коротка інформація про ОП, є можливість виконувати пошук за назвою, університетом, спеціальністю та освітнім рівнем ОП (див. рис. 31).

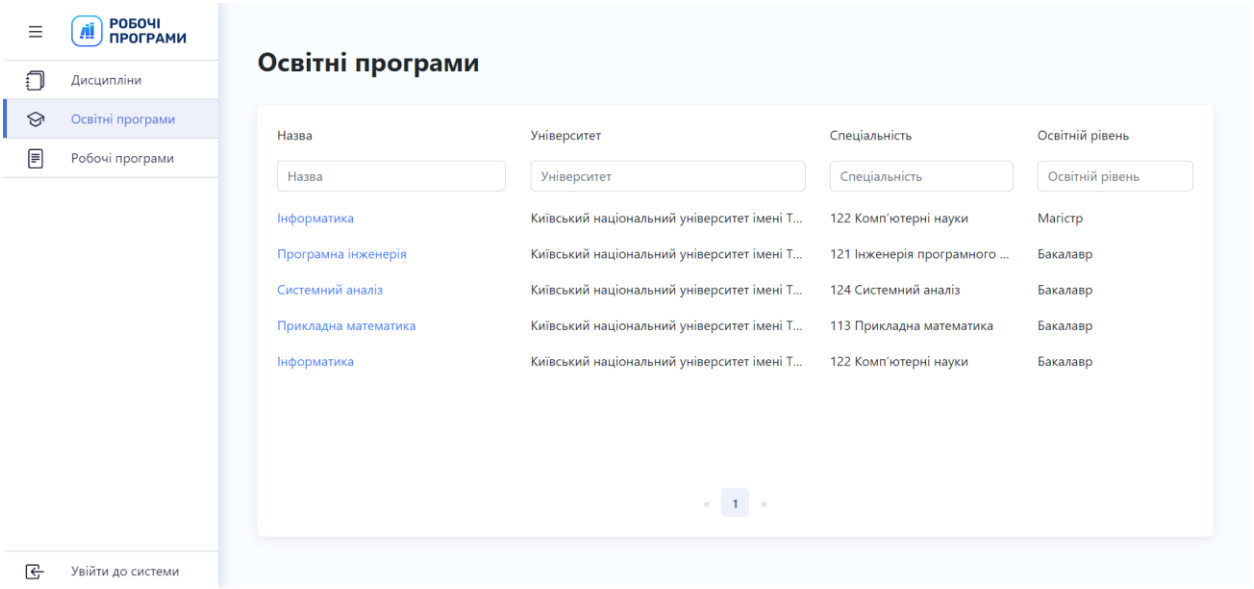


Рисунок 31 – Перелік ОП

Перейшовши на сторінку ОП, користувач може переглянути коротку інформацію про університет, факультет, галузь знань, спеціальність, освітній рівень, перелік компетентностей та програмних результатів (див. рис 32). Також на даній сторінці наявний перелік дисциплін та посилання на сторінку детальної інформації.

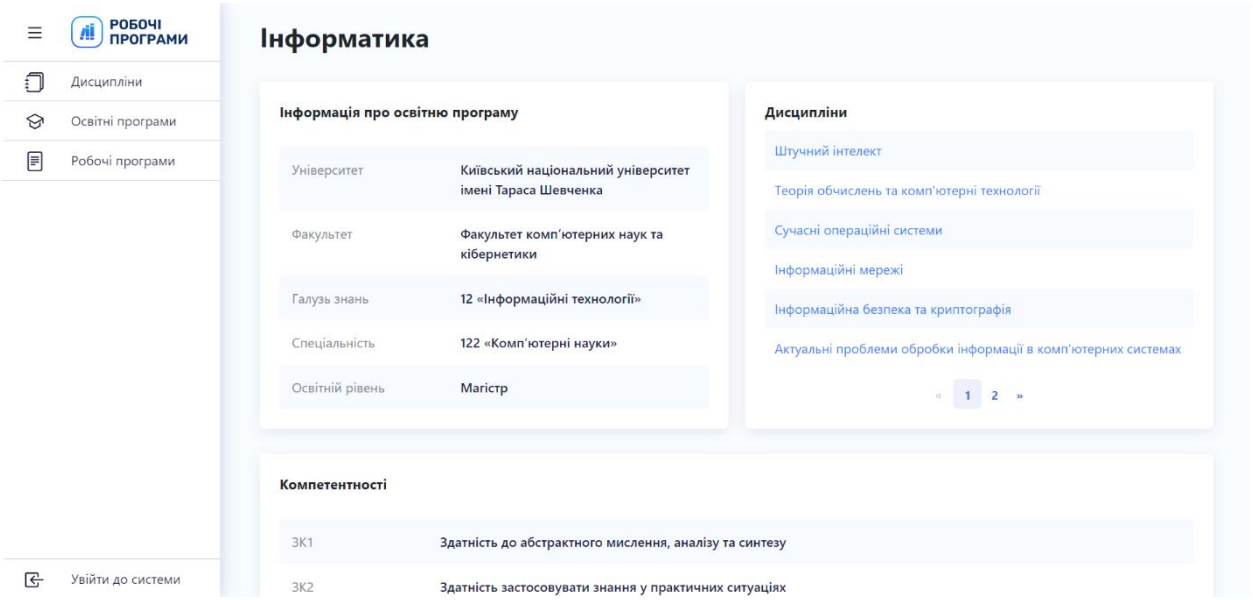


Рисунок 32 – Детальна інформація про обрану ОП

Відкривши вкладинку «Робочі програми», перед користувачем відображається коротка інформація про РПНД, є можливість виконувати пошук за назвою РПНД та назвою дисципліни, для якої вона створена (див. рис. 33).

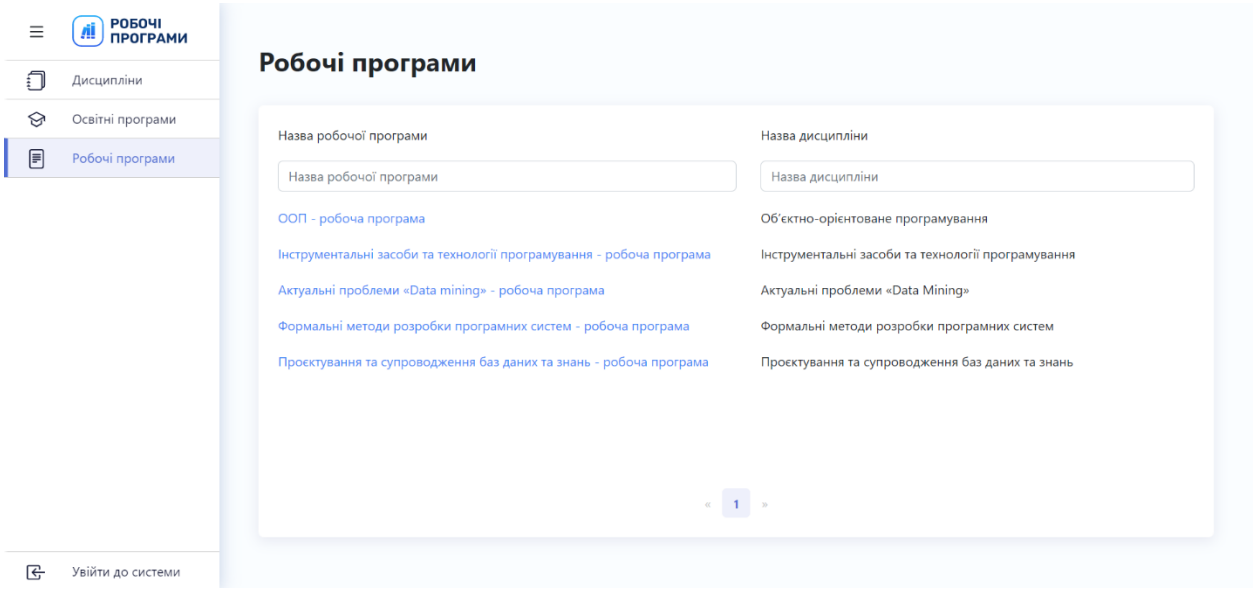


Рисунок 33 – Перелік РПНД

Перейшовши на сторінку РПНД, користувач може переглянути коротку інформацію про ОП, дисципліну, ким дана робоча програма була створена та підтверджена. Також є можливість завантажити файл РПНД (див. рис. 34).

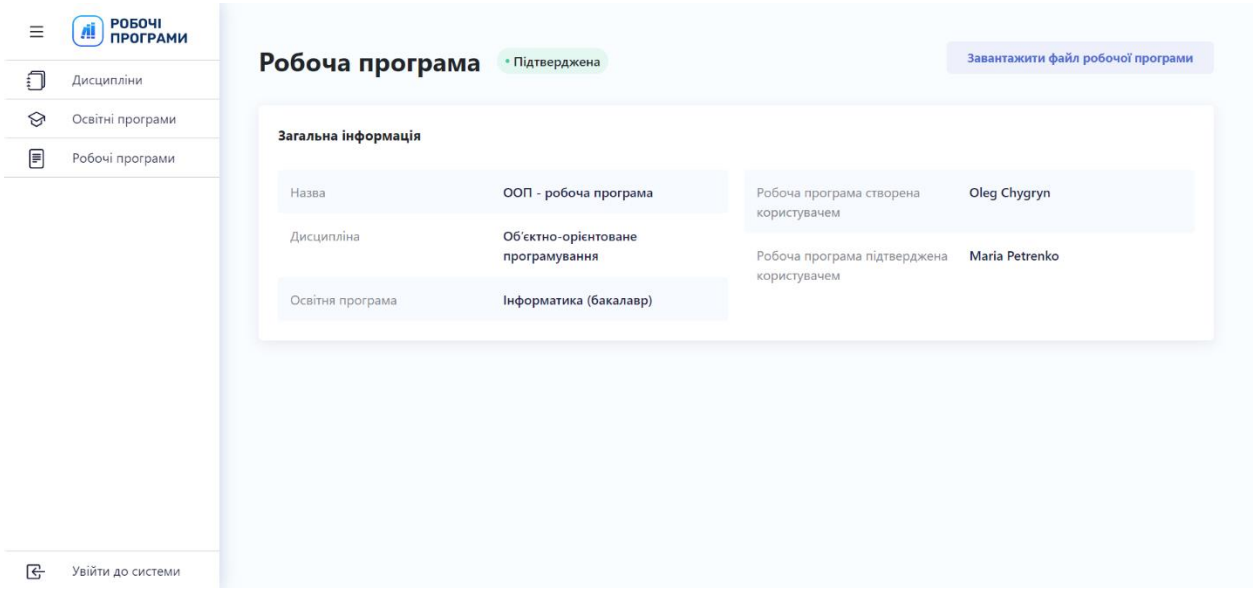


Рисунок 34 – Детальна інформація про РПНД

Для авторизації до систем необхідно ввести електронну пошту та пароль. У випадку, коли якісь поля пропущенні або введені некоректно,

система відображає повідомлення про помилку (див. рис. 35). Якщо необхідні для авторизації дані введено коректно, проте в системі їх комбінація не існує, то користувачу відображається повідомлення про помилку (див. рис. 36).

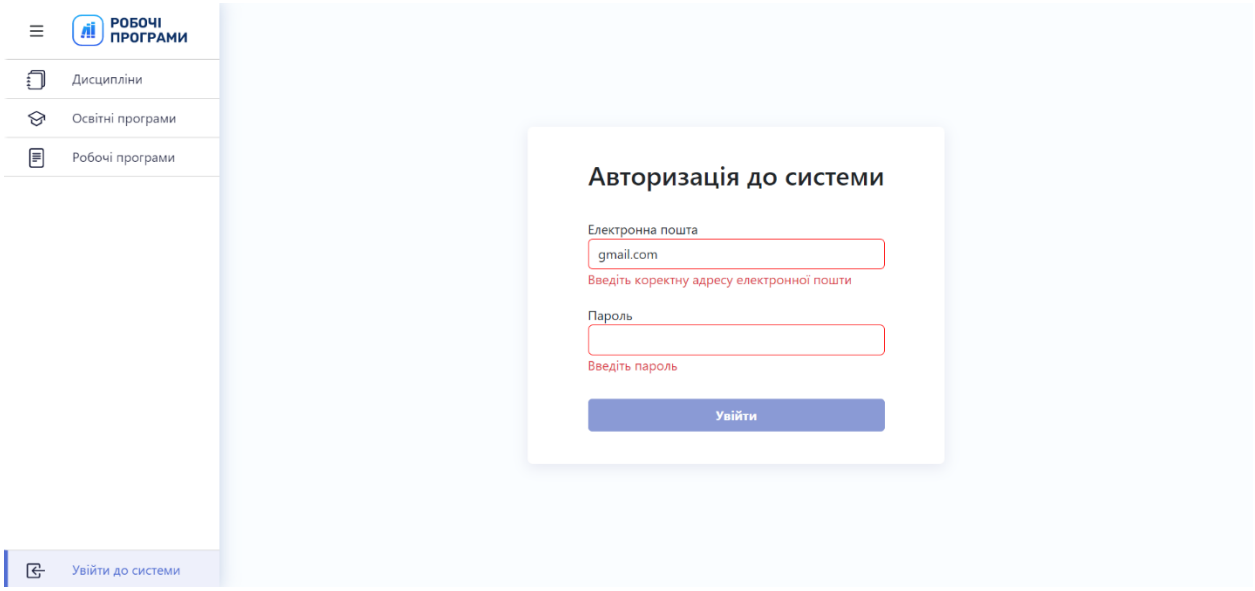


Рисунок 35 – Сторінка авторизації (клієнтська валідація)

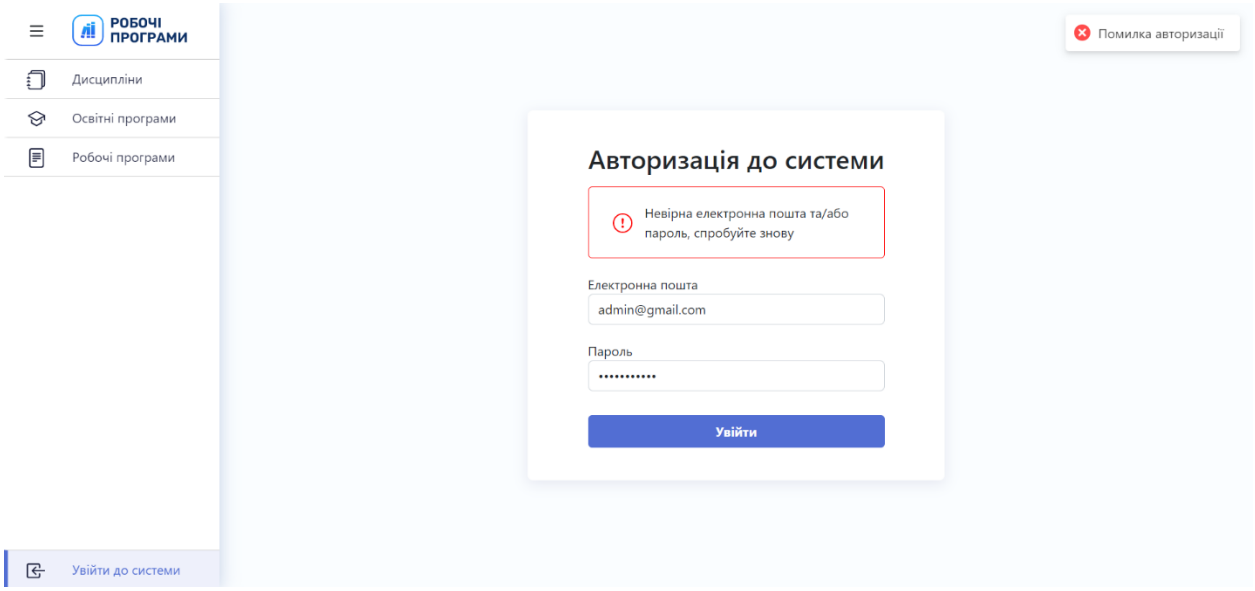


Рисунок 36 – Сторінка авторизації (серверна валідація)

5.2 Інструкція викладача

Після успішної авторизації до системи, з’являється можливість увійти до електронного кабінету, в якому знаходиться інформація про користувача, його роль у системі, перелік створених та підтверджених РПНД (див. рис. 37). Також є можливість зміни паролю. У випадку некоректності введених даних, система відображає повідомлення про помилку (див. рис. 38).

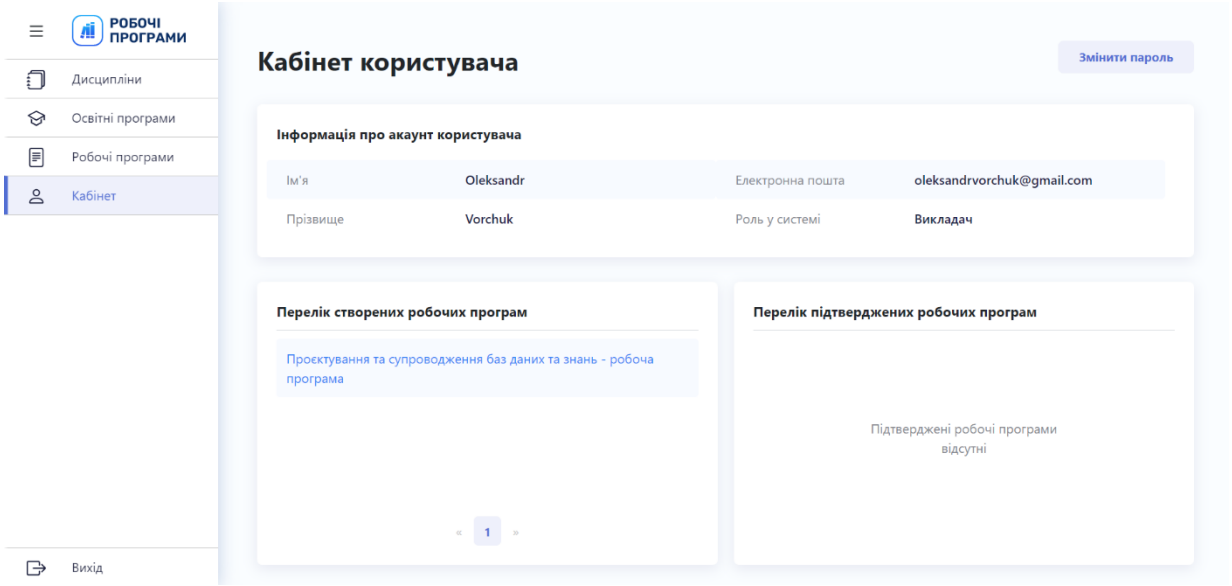


Рисунок 37 – Кабінет користувача

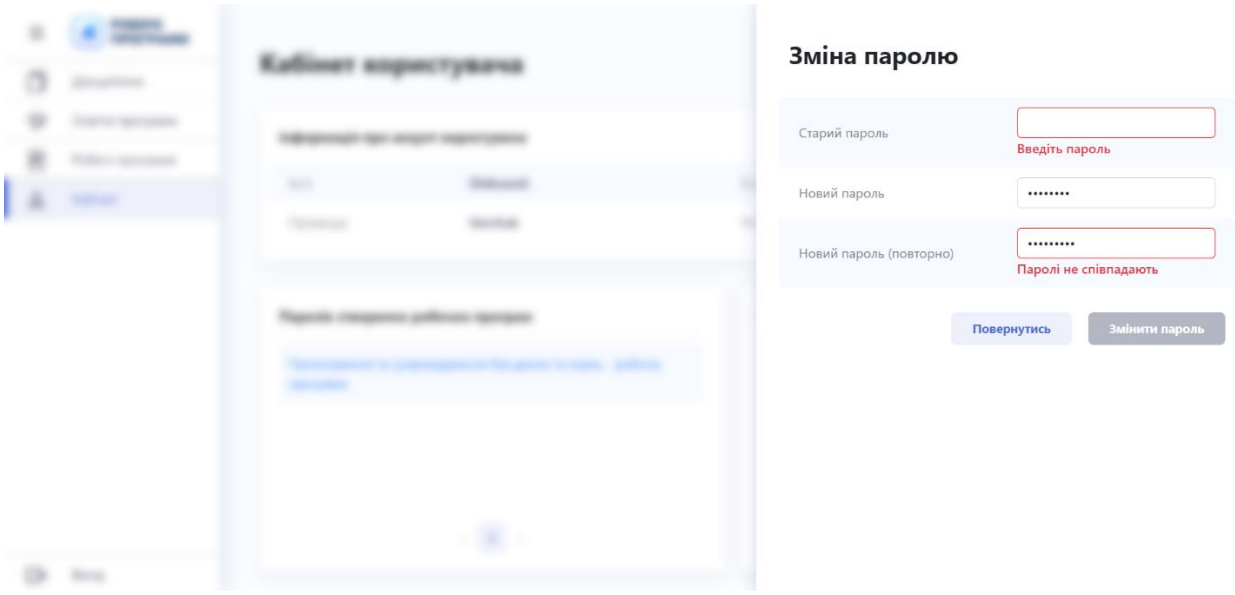


Рисунок 38 – Зміна паролю (клієнтська валідація)

На сторінці детальної інформації про дисципліну з’являється можливість генерації системою шаблону РПНД (див. рис. 39).

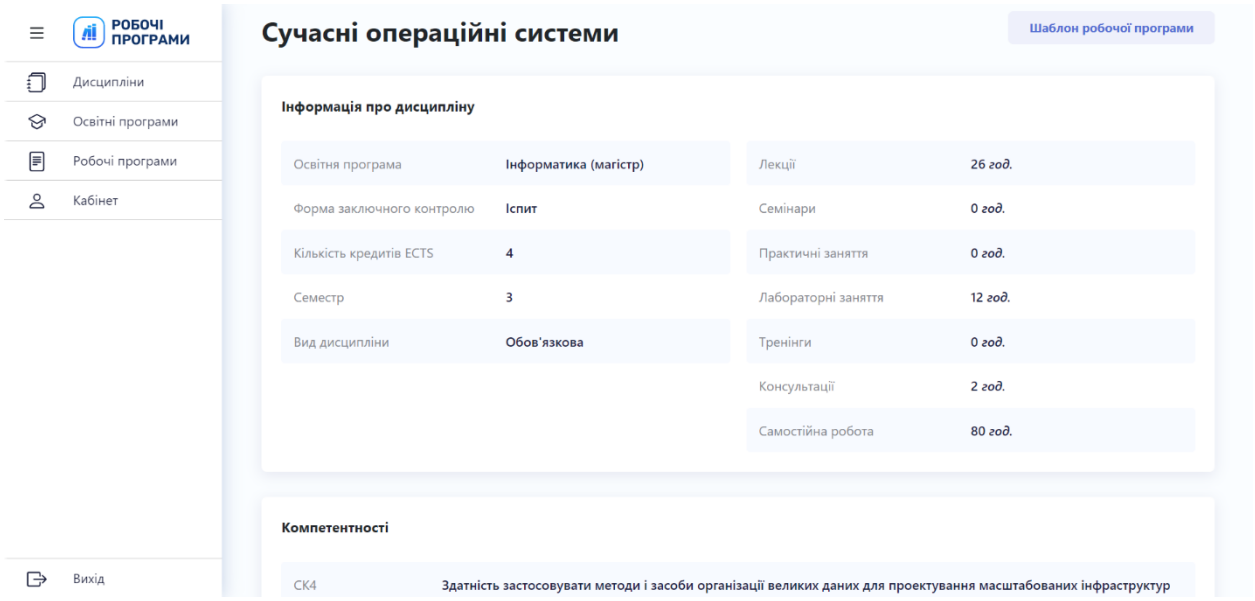


Рисунок 39 – Можливість генерації шаблону РПНД

Після успішного створення шаблону РПНД, файл завантажується користувачу у форматі docx для можливості редагування та внесення додаткової інформації (див. рис 40 – 41).

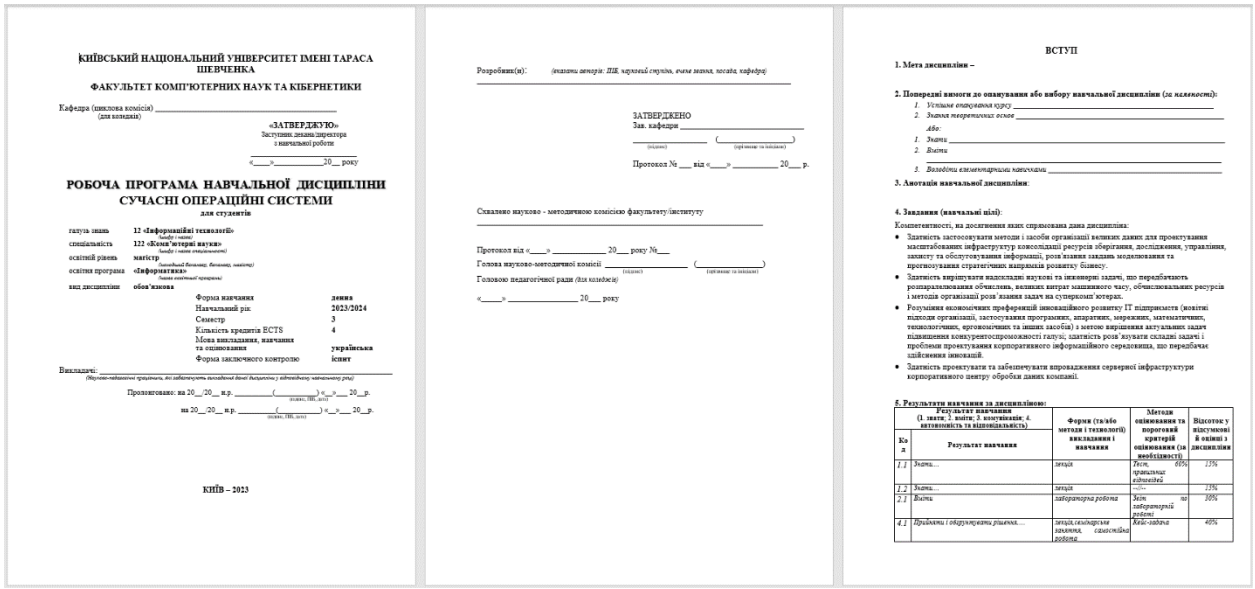
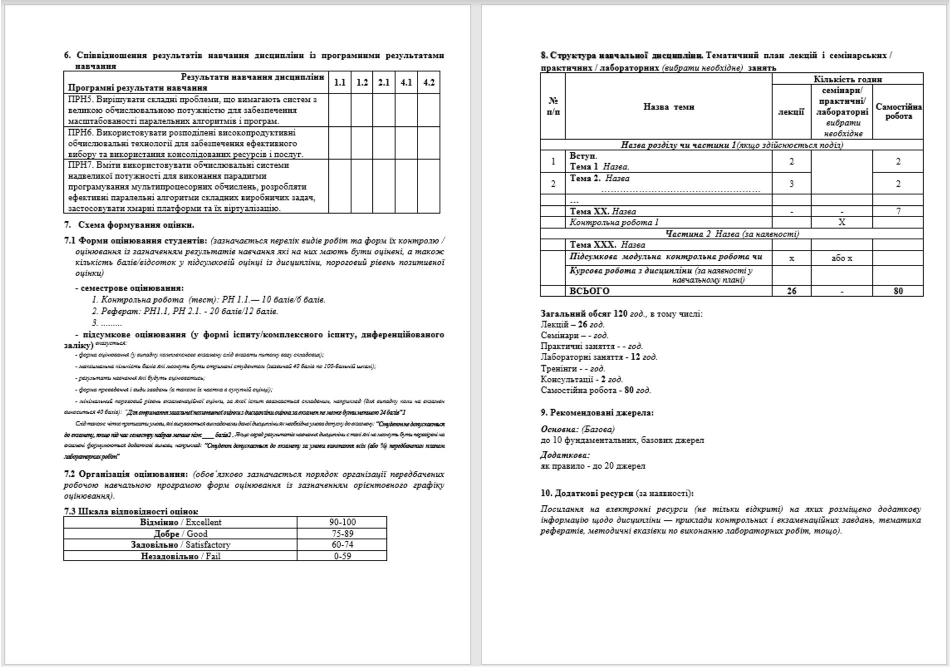

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА
ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ НАУК ТА КІБЕРНЕТИКИ
РОБОЧА ПРОГРАМА НАВЧАЛЬНОЇ ДИСЦИПЛІНИ
СУЧАСНІ ОПЕРАЦІЙНІ СИСТЕМИ
КИЇВ – 2023
ВСТУП

Рисунок 40 – Згенерований шаблон РПНД (частина 1)



6. **Співвідношення результатів навчання дисципліни із програмними результатами навчання**

Результати навчання дисципліни / Програмні результати навчання	1.1	1.2	2.1	4.1	4.2
ПРН5. Вирішувати складні проблеми, що вимагають систем з великою обчислювальною потужністю для забезпечення масштабованості паралельних алгоритмів і програм.					
ПРН6. Використовувати розподілені високопродуктивні обчислювальні технології для забезпечення ефективного вибору та використання консолідованих ресурсів і послуг.					
ПРН7. Вміти використовувати обчислювальні системи надвеликої потужності для виконання парадигми програмування мультипроцесорних обчислень, розробляти ефективні паралельні алгоритми складних виробничих задач, застосовувати хмарні платформи та їх віртуалізацію.					

7. **Схема формування оцінки.**

7.1 Форми оцінювання студентів: *(зазначається перелік видів робіт та форм їх контролю / оцінювання із зазначенням результатів навчання які на них мають бути оцінені, а також кількість балів/відсоток у підсумковій оцінці із дисципліни, пороговий рівень позитивної оцінки)*

- семестрове оцінювання:

1. *Контрольна робота (тест): РН 1.1.— 10 балів/6 балів.*
2. *Реферат: РН1.1, РН 2.1. - 20 балів/12 балів.*
3. ....

- підсумкове оцінювання (у формі іспиту/комплексного іспиту, диференційованого заліку) *вказується:*

- *форма оцінювання (у випадку комплексного екзамену слід вказати питому вагу складових);*
- *максимальна кількість балів які можуть бути отримані студентом (зазвичай 40 балів по 100-бальній шкалі);*
- *результати навчання які будуть оцінюватись;*
- *форма проведення і види завдань (а також їх частка в сукупній оцінці);*
- *мінімальний пороговий рівень екзаменаційної оцінки, за якої іспит вважається складеним, наприклад (для випадку коли на екзамен виноситься 40 балів):* **"Для отримання загальної позитивної оцінки з дисципліни оцінка за екзамен не може бути меншою 24 балів"1**

Слід також чітко прописати умови, які висуваються викладачем даної дисципліни як необхідна умова допуску до екзамену: **"Студент не допускається до екзамену, якщо під час семестру набрав менше ніж ____ балів2**. *Якщо серед результатів навчання дисципліни є такі які не можуть бути перевірені на екзамені формулюються додаткові вимоги, наприклад:* **"Студент допускається до екзамену за умови виконання всіх (або %) передбачених планом лабораторних робіт"**

7.2 Організація оцінювання: *(обов'язково зазначається порядок організації передбачених робочою навчальною програмою форм оцінювання із зазначенням орієнтовного графіку оцінювання).*

7.3 Шкала відповідності оцінок

Відмінно / Excellent	90-100
Добре / Good	75-89
Задовільно / Satisfactory	60-74
Незадовільно / Fail	0-59

8. Структура навчальної дисципліни. Тематичний план лекцій і семінарських / практичних / лабораторних *(вибрати необхідне)* **занять**

№ п/п	Назва теми	Кількість годин: лекції	Кількість годин: семінари/ практичні/ лабораторні *вибрати необхідне*	Кількість годин: Самостійна робота
Назва розділу чи частини 1*(якщо здійснюється поділ)*				
1	**Вступ.** **Тема 1** *Назва.*	2		2
2	**Тема 2.** *Назва*	3		2
	...			
	Тема ХХ. *Назва*	-	-	7
	Контрольна робота 1		Х	
Частина 2 *Назва (за наявності)*				
	Тема ХХХ. *Назва*			
	Підсумкова модульна контрольна робота чи	х	або х	
	Курсова робота з дисципліни *(за наявності у навчальному плані)*			
	ВСЬОГО	**26**	**-**	**80**

Загальний обсяг 120 *год.*, в тому числі:
Лекцій – **26** *год.*
Семінари – - *год.*
Практичні заняття - - *год.*
Лабораторні заняття - **12** *год.*
Тренінги - - *год.*
Консультації - **2** *год.*
Самостійна робота - **80** *год.*

9. Рекомендовані джерела:

Основна: *(Базова)*
до 10 фундаментальних, базових джерел

Додаткова:
як правило - до 20 джерел

10. Додаткові ресурси (за наявності):

Посилання на електронні ресурси (не тільки відкриті) на яких розміщено додаткову інформацію щодо дисципліни — приклади контрольних і екзаменаційних завдань, тематика рефератів, методичні вказівки по виконанню лабораторних робіт, тощо).

Рисунок 41 – Згенерований шаблон РПНД (частина 2)

На вкладинці «Робочі програми» з'являється кнопка для відкриття форми створення РПНД (див. рис. 42). Дана форма потребує введення назви РПНД, обрання ОП, дисципліни та необхідного файлу. У випадку некоректності введених даних, відображається відповідне повідомлення про помилку (див. рис. 43).

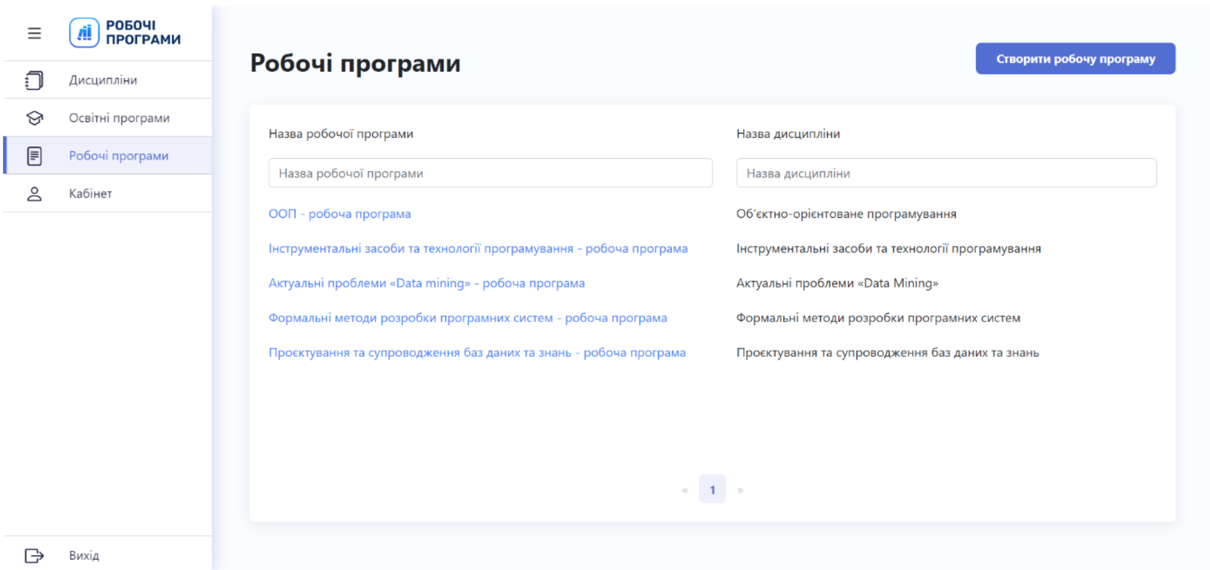


Рисунок 42 – Можливість створення РПНД

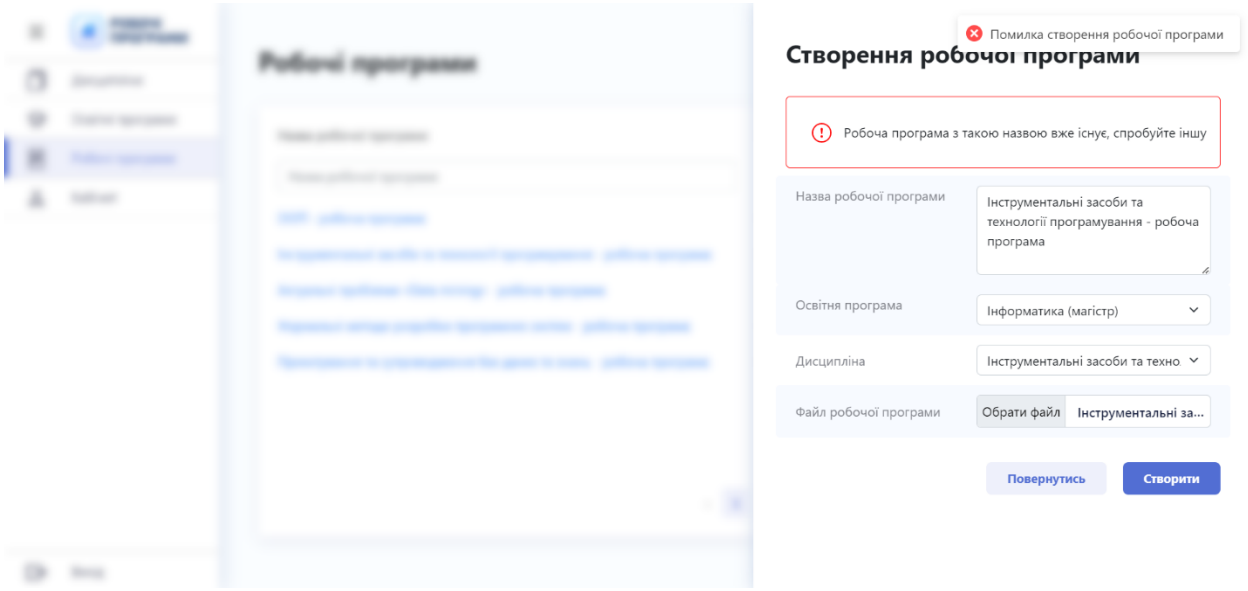


Рисунок 43 – Серверна валідація при створенні РПНД

При натисканні кнопки «Повернутись», система вимагає від користувача підтвердження дії, для унеможливлення випадку помилкового натискання (див. рис. 44).

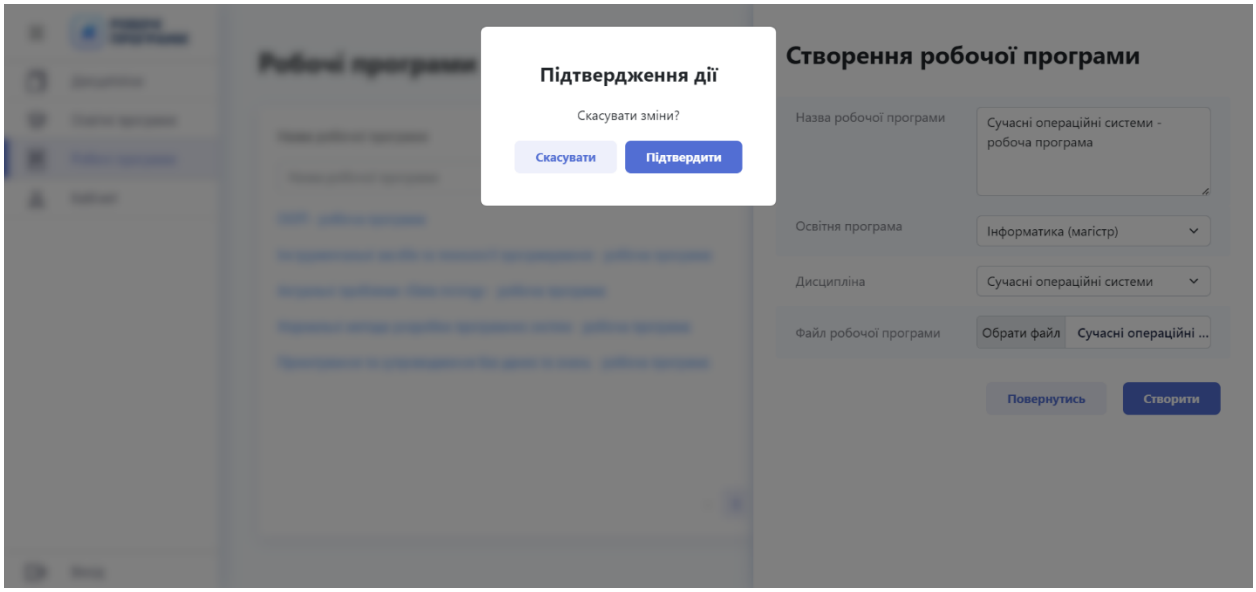


Рисунок 44 – Підтвердження дії у випадку натискання кнопки «Повернутись»

Після успішного створення, відкривається сторінка із детальною інформацією про РПНД та поміткою «Очікує підтвердження» (див. рис. 45). Для того щоб дана робоча програма була доступна всім, необхідно отримати підтвердження методиста.

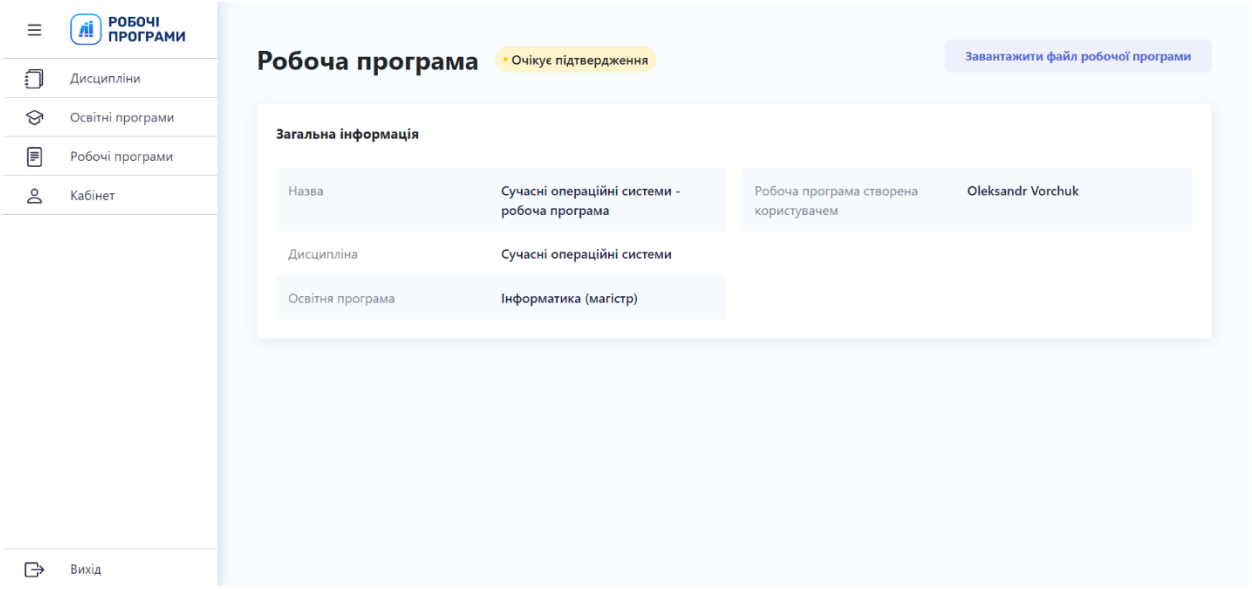


Рисунок 45 – Сторінка РПНД відразу після створення

5.3 Інструкція методиста

Після успішної авторизації користувача, який у системі має роль «Методист», з’являється можливість створення та редагування інформації про дисципліни, додавання та видалення компетентностей та програмних результатів (див. рис. 46 – 47).

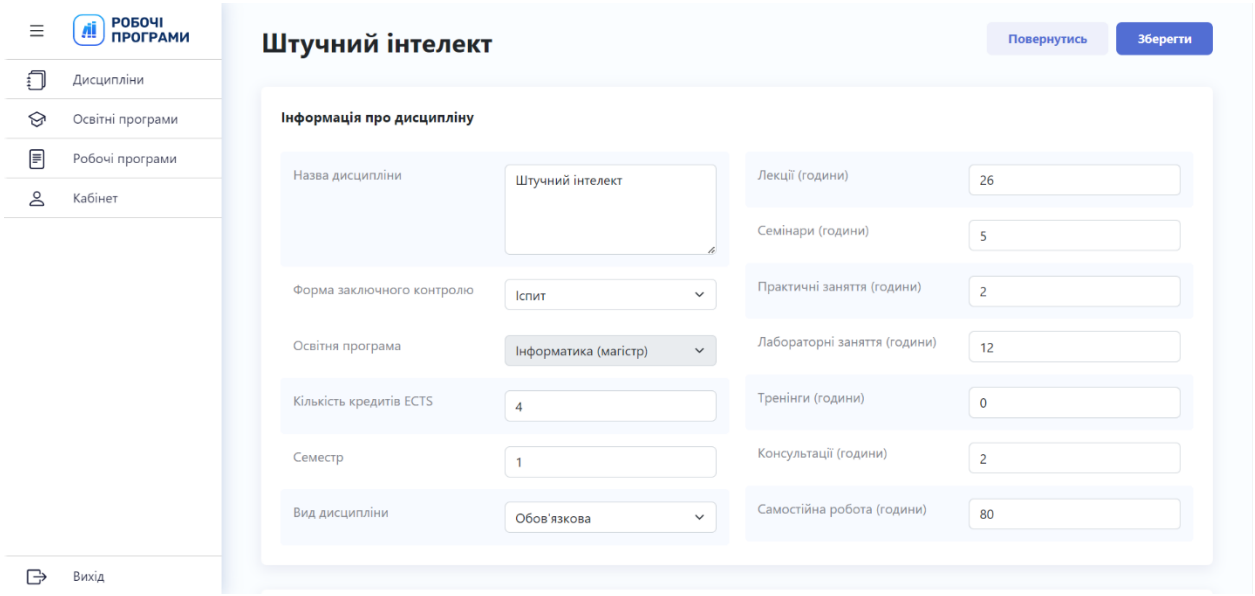


Рисунок 46 – Сторінка редагування дисципліни (основна інформація)

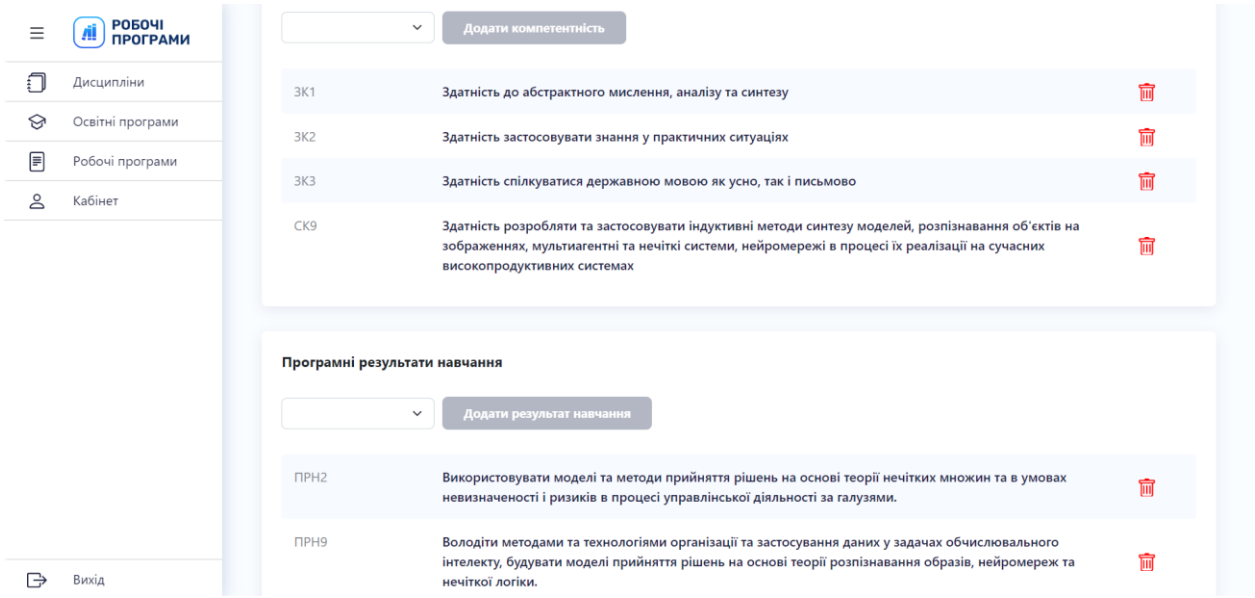


Рисунок 47 – Сторінка редагування дисципліни (компетентності та програмні результати)

Якщо при створенні чи редагуванні дисципліни користувачем вводиться некоректна інформація, то система відображає відповідне повідомлення про помилку (див. рис. 48).

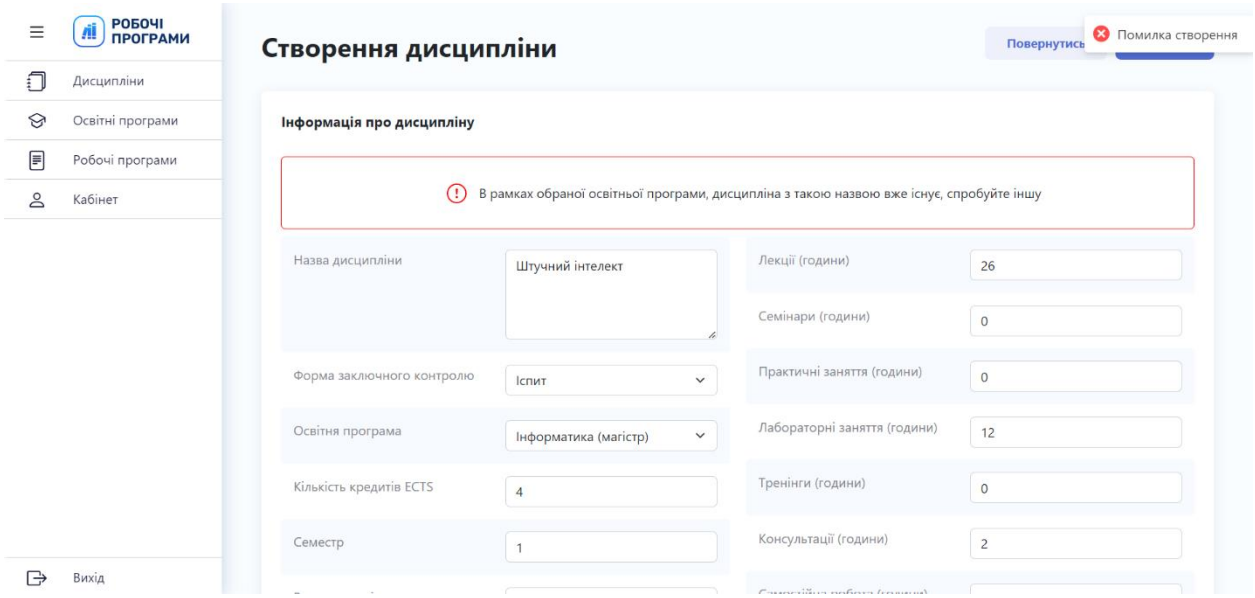


Рисунок 48 – Серверна валідація при створенні дисципліни

Також методист може створювати та редагувати інформацію про ОП. При редагуванні є можливість створювати та видаляти компетентності та програмні результати на рівні ОП (див. рис. 49). При видаленні система запитує підтвердження дії (див. рис. 50). Дані зміни впливають на відповідні дисципліни та генерацію системою шаблонів РПНД.

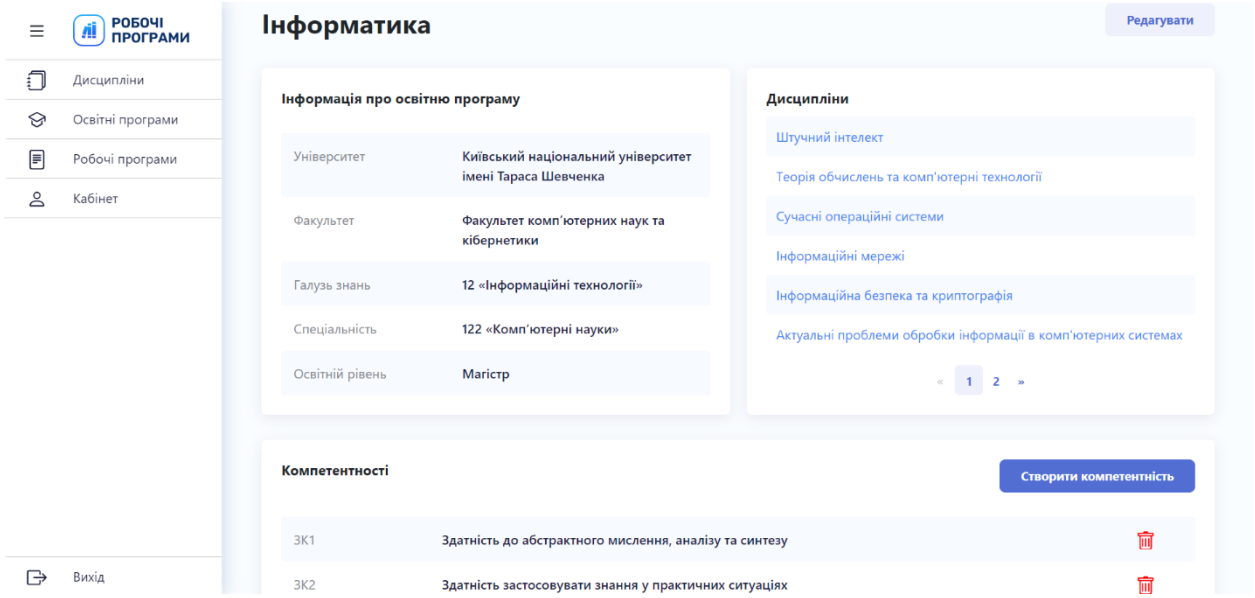


Рисунок 49 – Сторінка редагування ОП

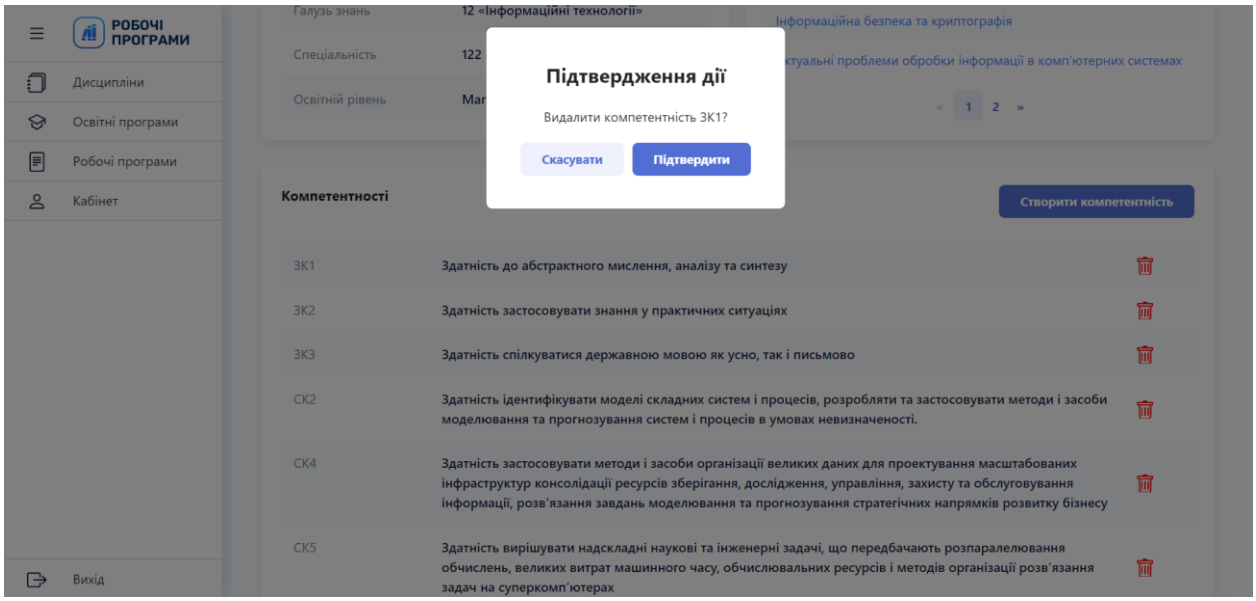


Рисунок 50 – Підтвердження дії при видаленні компетентності

На сторінці детальної інформації РПНД методист може завантажувати для перевірки файл РПНД, підтверджувати та відхиляти (див. рис. 51). Для відхилення необхідно обов’язково вказати причину (див. рис. 52), після чого відбудеться видалення РПНД та автору надішлеться відповідний електронний лист.

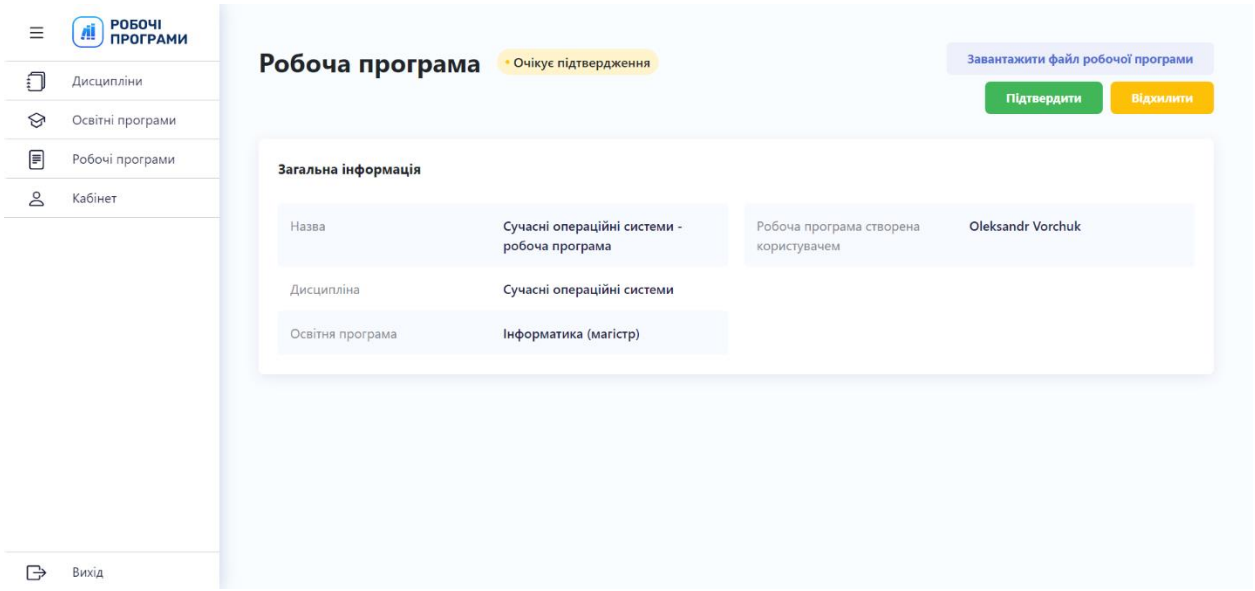


Рисунок 51 – Сторінка детальної інформації РПНД

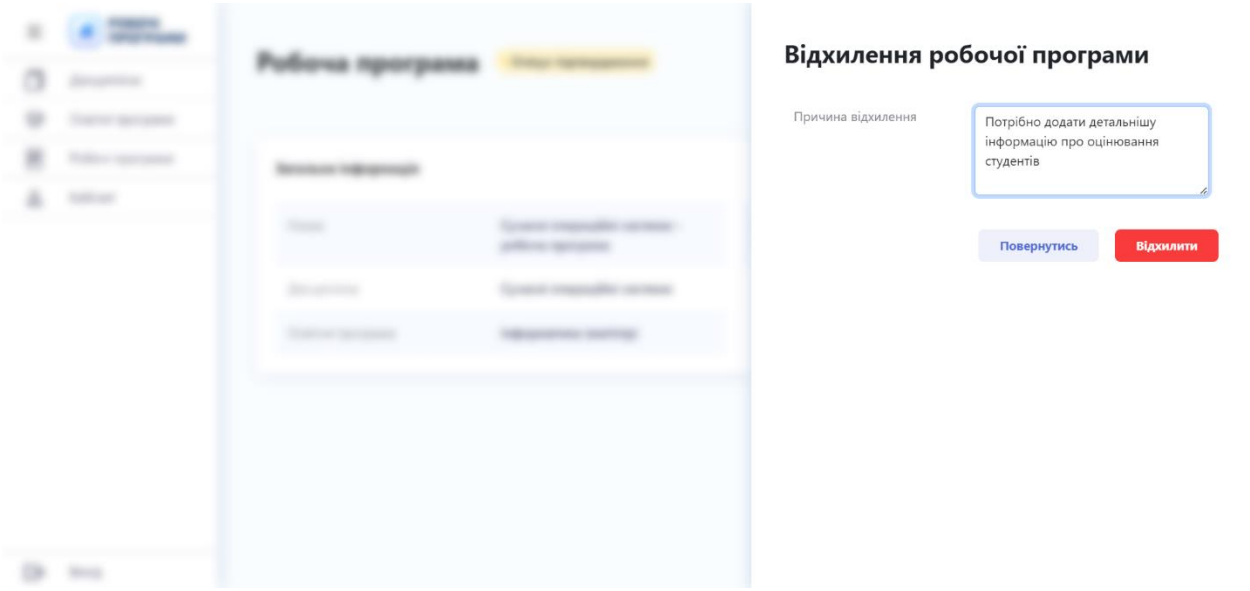


Рисунок 52 – Відхилення РПНД

Після підтвердження система сигналізує про успішність операції та на сторінці РПНД відображається інформація про методиста що здійснив дану операцію (див. рис. 53). Підтверджені РПНД методист має можливість видаляти.

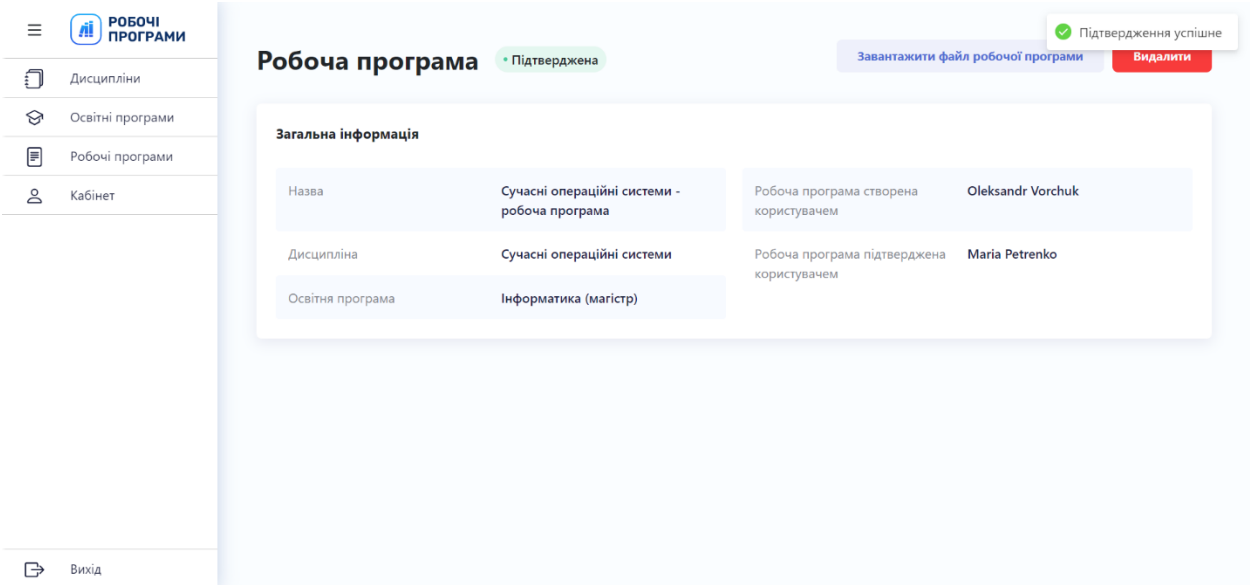


Рисунок 53 – Підтверджена РПНД

5.4 Інструкція адміністратора

Після успішної авторизації користувача, який у системі має роль «Адміністратор», з'являється можливість переглядати детальну інформацію про користувачів та створювати нові акаунти (див. рис. 54). Для створення нового акаунту необхідно ввести електронну пошту, ім'я, прізвище та обрати роль у системі. У випадку некоректності введених даних, система відображає повідомлення про помилку (див. рис. 55).

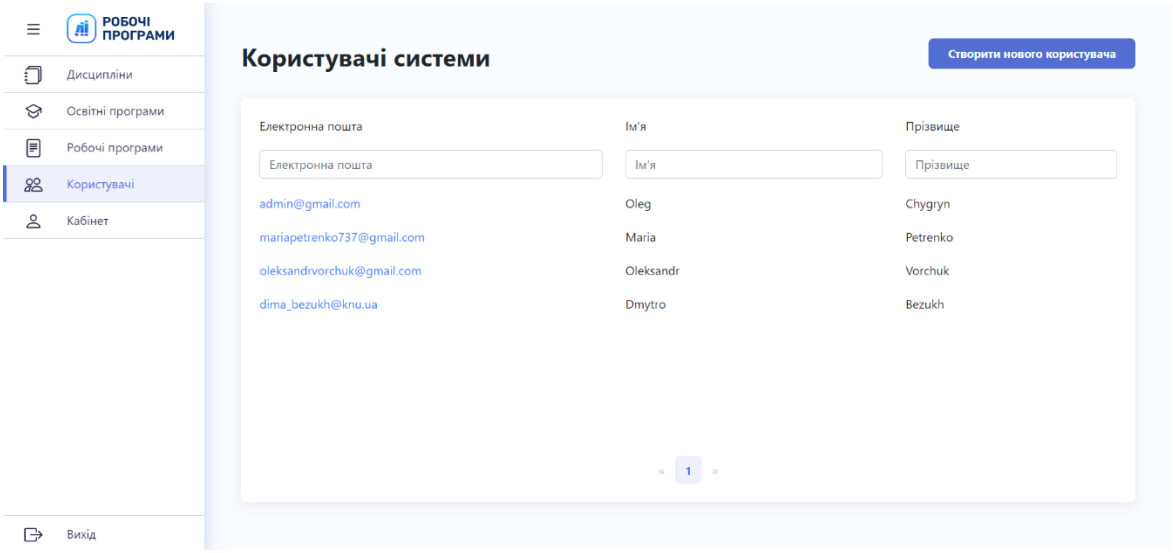


Рисунок 54 – Перелік користувачів системи

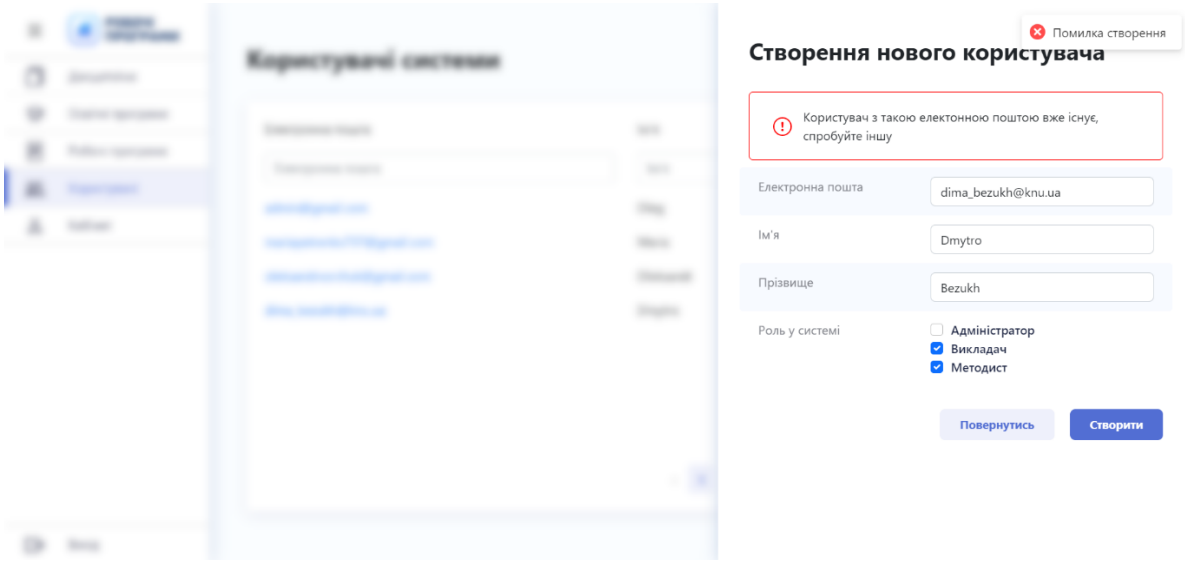


Рисунок 55 – Створення користувача (серверна валідація)

Після успішного створення, система відображає відповідне повідомлення та перенаправляє адміністратора на сторінку електронного кабінету (див. рис. 56). На електронну пошту створеного користувача надсилається лист, в якому міститься посилання для входу на платформу, логін та пароль (див. рис. 57).

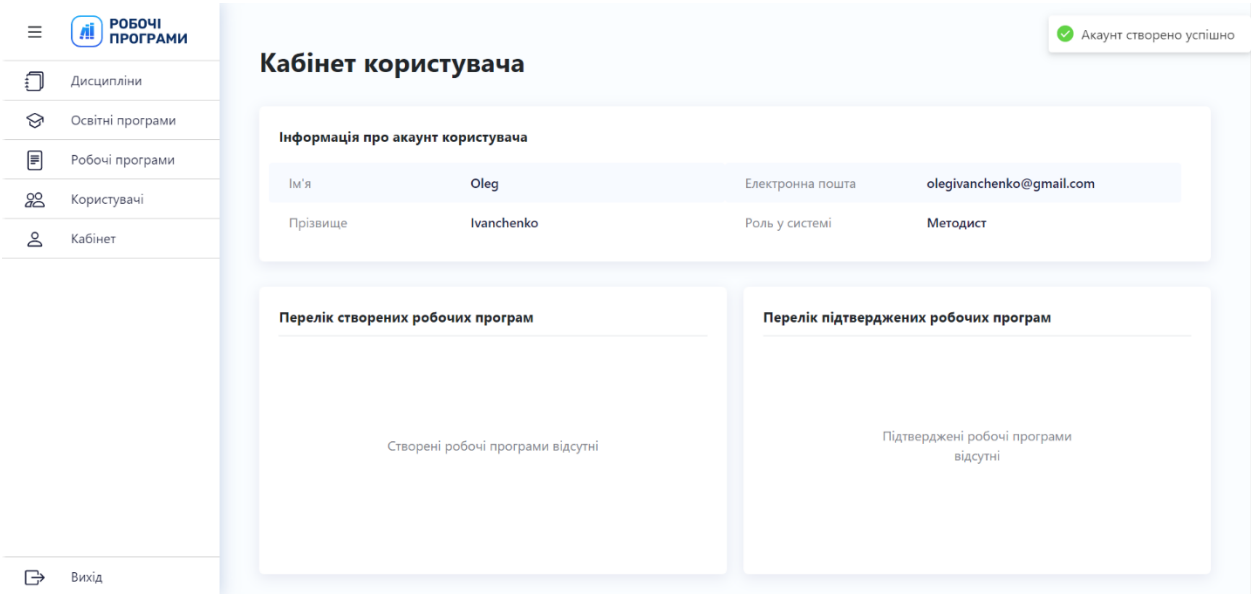


Рисунок 56 – Кабінет створеного користувача

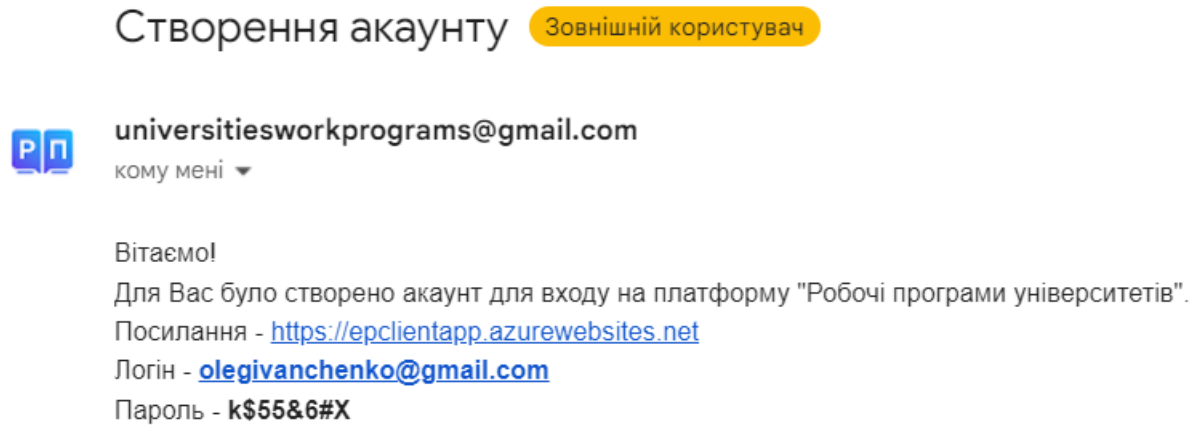


Рисунок 57 – Електронний лист із деталями створеного акаунту

ВИСНОВКИ

Під час розробки системи було досліджено процес розробки РПНД, проаналізовано існуючі додатки, які пов’язані з автоматизацією діяльності ЗВО, визначено переваги та недоліки кожного з них та поставлено відповідні задачі для розробки системи для автоматизації процесу створення РПНД та реалізації в ній основних засобів збереження, обробки та передачі інформації.

Спираючись на поставлені задачі, було сформовано вимоги до функціоналу системи та побудовано моделі баз даних, використовуючи які, було розроблено серверну частину системи, функціонал якої розширювався. В якості шаблону проектування серверної частини було обрано трирівневу архітектуру. Розроблена система була успішно розгорнута в хмарному середовищі Microsoft Azure.

Розробляючи систему, було прочитано необхідний обсяг документації, закріплено навички робити із платформою Microsoft Azure, технологіями ASP.NET CORE 6.0, фреймворком Angular, взаємодією з базою даних та верстки інтернет-сторінок.

За підсумками розробки проєкту, можна відзначити, що результатом роботи є повністю працездатна система, яка може використовуватись для підтримки освітнього процесу у ЗВО.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

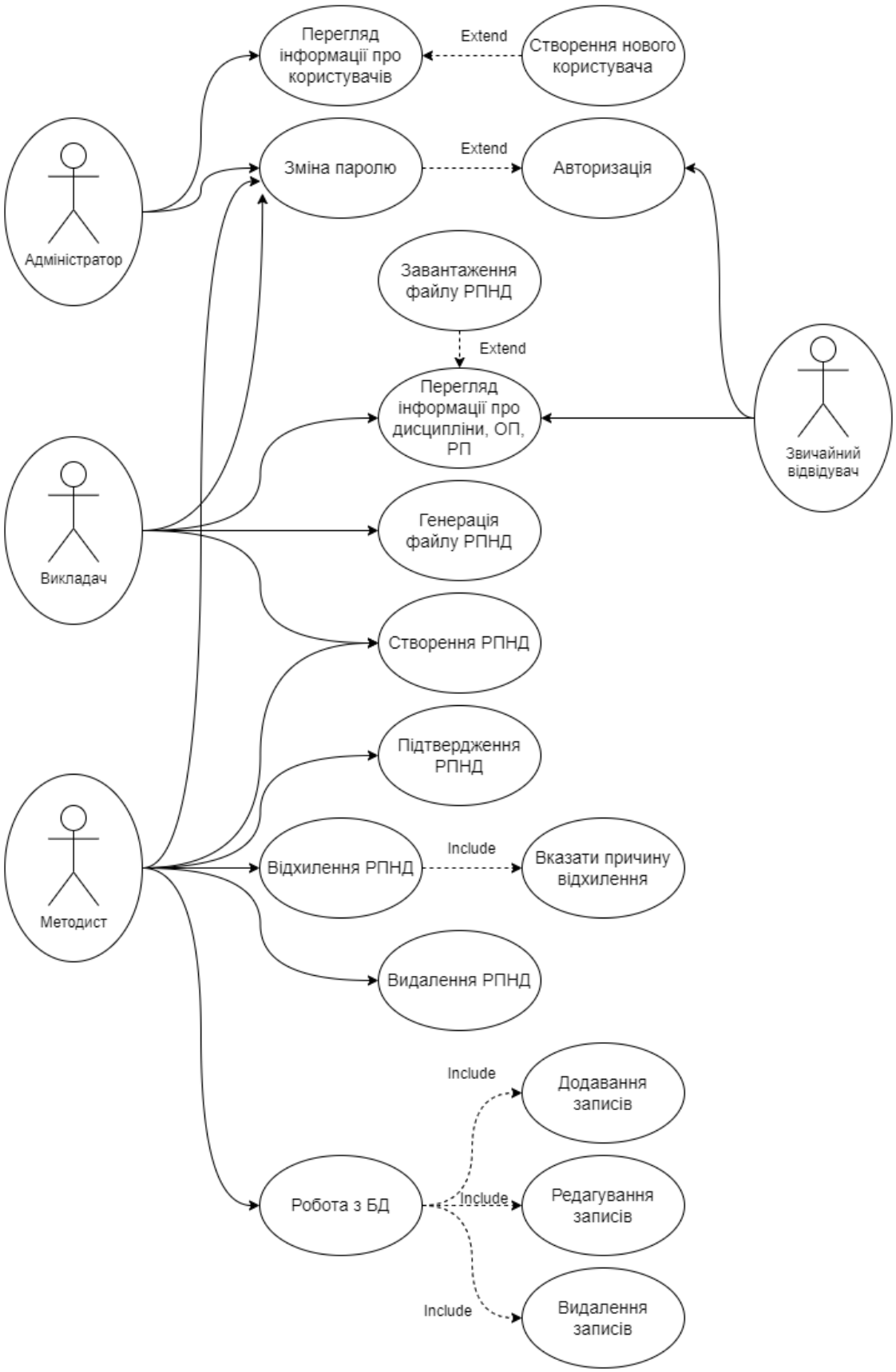
1. M. Elhoseny, N. Metawa, A. E. Hassanien, “An automated information system to ensure quality in higher education institutions” in Proc. 12th International Computer Engineering Conference (ICENCO) – 2016.
2. J. Khalid, B. R. Ram, M. Soliman, A. J. Ali, M. Khaleel, M. S. Islam, “Promising digital university: A pivotal need for higher education transformation”, International Journal of Management in Education. ст. 264-275 – 2018.
3. B. Bygstad, E. Øvrelid, S. Ludvigsen, M. Dæhlen, “From dual digitalization todigital learning space: Exploring the digital transformation of higher education”, Computers & Education – 2022.
4. Косіюк М. М. Automated Information Management System of Higher Education Institution «Electronic University» [Електронний ресурс] / М. М. Косіюк, К. Е. Більовський, В. М. Лисак. – 2023. – Режим доступу до ресурсу: https://journal.iitta.gov.ua/index.php/itlt/article/view/5107.
5. Безух Д. Система підтримки процесу розробки робочих програм навчальних дисциплін для закладів вищої освіти / Дмитро Безух // Всеукраїнська науково-практична інтернет-конференція «Автоматизація та комп’ютерно-інтегровані технології у виробництві та освіті: стан, досягнення, перспективи розвитку» / Дмитро Безух. – Черкаси, 2023. – С. 124.
6. Автоматизована система управління вищим навчальним закладом III - IV рівня акредитації [Електронний ресурс] – Режим доступу до ресурсу: https://www.unitex.com.ua/products/commercial-software/automated-system-for-higher-education-institution/.
7. Автоматизована система управління «Вищий навчальний заклад» [Електронний ресурс] – Режим доступу до ресурсу: https://vuz.osvita.net/.

8. Система автоматизації Київського національного університету імені Тараса Шевченка (Triton Student) [Електронний ресурс]. Режим доступу до ресурсу: https://student.triton.knu.ua/.
9. Overview of ASP.NET Core [Електронний ресурс]. – 2022. – Режим доступу до ресурсу: https://learn.microsoft.com/en-us/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-6.0.
10. ASP.NET documentation [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/aspnet/core/.
11. Code Maze - ASP.NET Core [Електронний ресурс] – Режим доступу до ресурсу: https://code-maze.com/net-core-series/.
12. What's new in ASP.NET Core 6.0 [Електронний ресурс]. – 2022. – Режим доступу до ресурсу: https://learn.microsoft.com/en-us/aspnet/core/release-notes/aspnetcore-6.0?view=aspnetcore-6.0.
13. Entity Framework Core [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/ef/core/.
14. Entity Framework Core [Електронний ресурс] – Режим доступу до ресурсу: https://www.entityframeworktutorial.net/efcore/entity-framework-core.aspx.
15. Introduction to Angular concepts [Електронний ресурс] – Режим доступу до ресурсу https://angular.io/guide/architecture.
16. Using Angular in Visual Studio Code [Електронний ресурс] – Режим доступу до ресурсу https://code.visualstudio.com/docs/nodejs/angular-tutorial.
17. Code Maze - Angular [Електронний ресурс] – Режим доступу до ресурсу: https://code-maze.com/angular-series/.
18. Build fast, responsive sites with Bootstrap [Електронний ресурс] – Режим доступу до ресурсу: https://getbootstrap.com/.
19. Bootstrap documentation [Електронний ресурс] – Режим доступу до ресурсу: https://getbootstrap.com/docs/4.1/getting-started/introduction/.

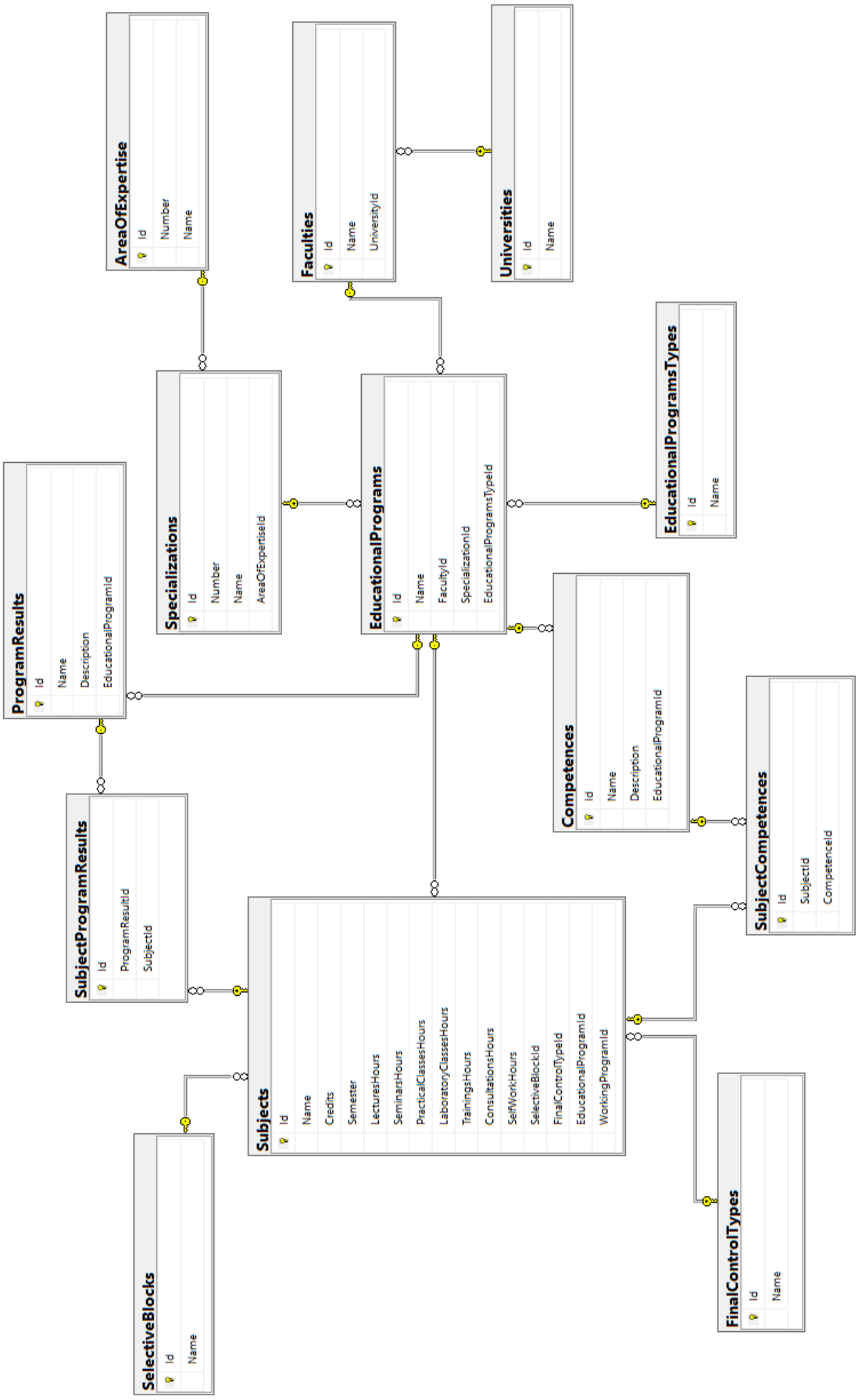
20. Three-Tier Architecture [Електронний ресурс] – Режим доступу до ресурсу https://www.ibm.com/cloud/learn/three-tier-architecture.
21. Microsoft Azure products [Електронний ресурс]. Режим доступу до ресурсу: https://azure.microsoft.com/en-us/products/.
22. Pros and Cons of Microsoft Azure 2023 [Електронний ресурс] – Режим доступу до ресурсу: https://www.trustradius.com/products/microsoft-azure/reviews?qs=pros-and-cons#overview.
23. Browser Statistics [Електронний ресурс] – Режим доступу до ресурсу: https://www.w3schools.com/browsers/.
24. Swagger: API Documentation & Design Tools [Електронний ресурс] – Режим доступу до ресурсу: https://swagger.io/.

ДОДАТКИ

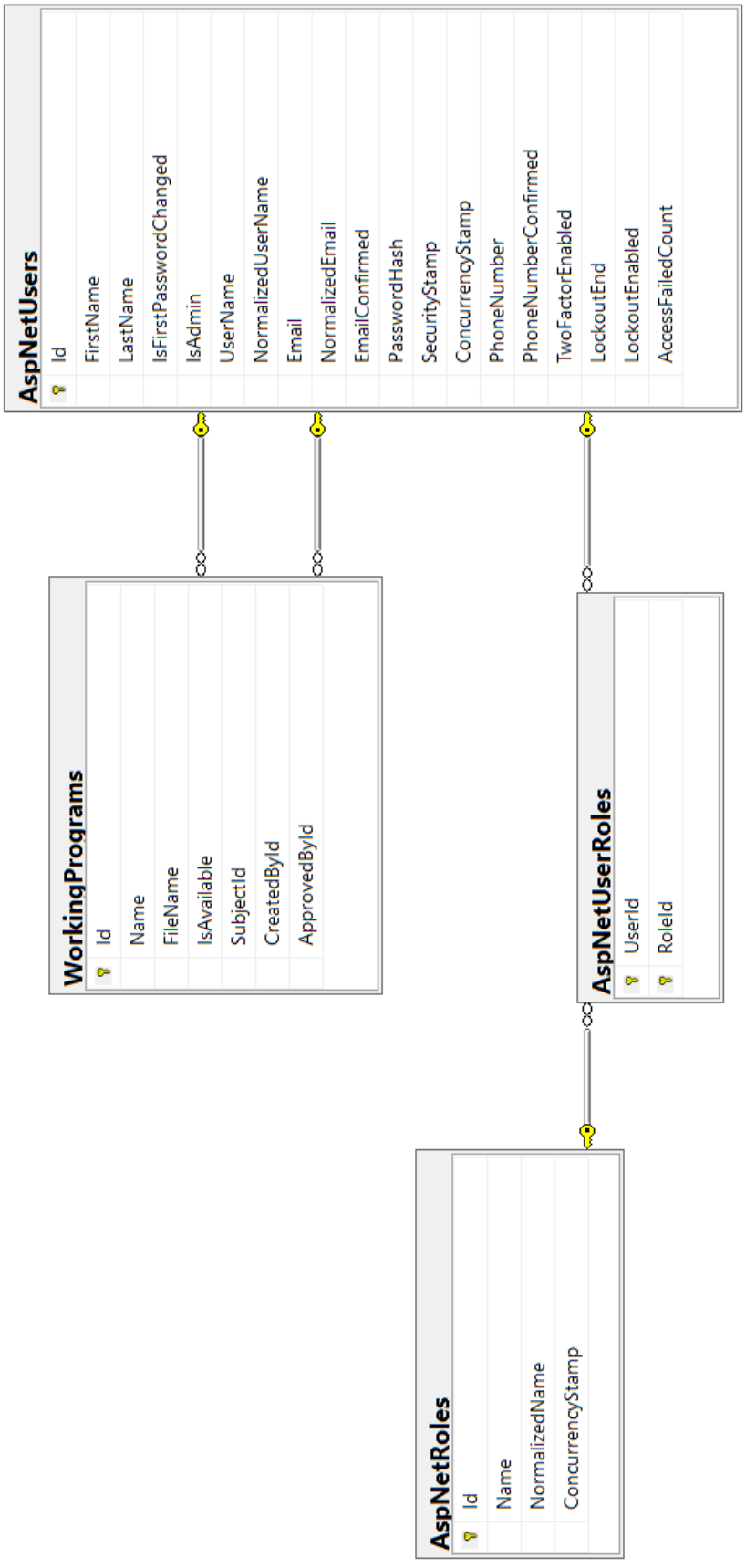
Додаток А Діаграма прецедентів



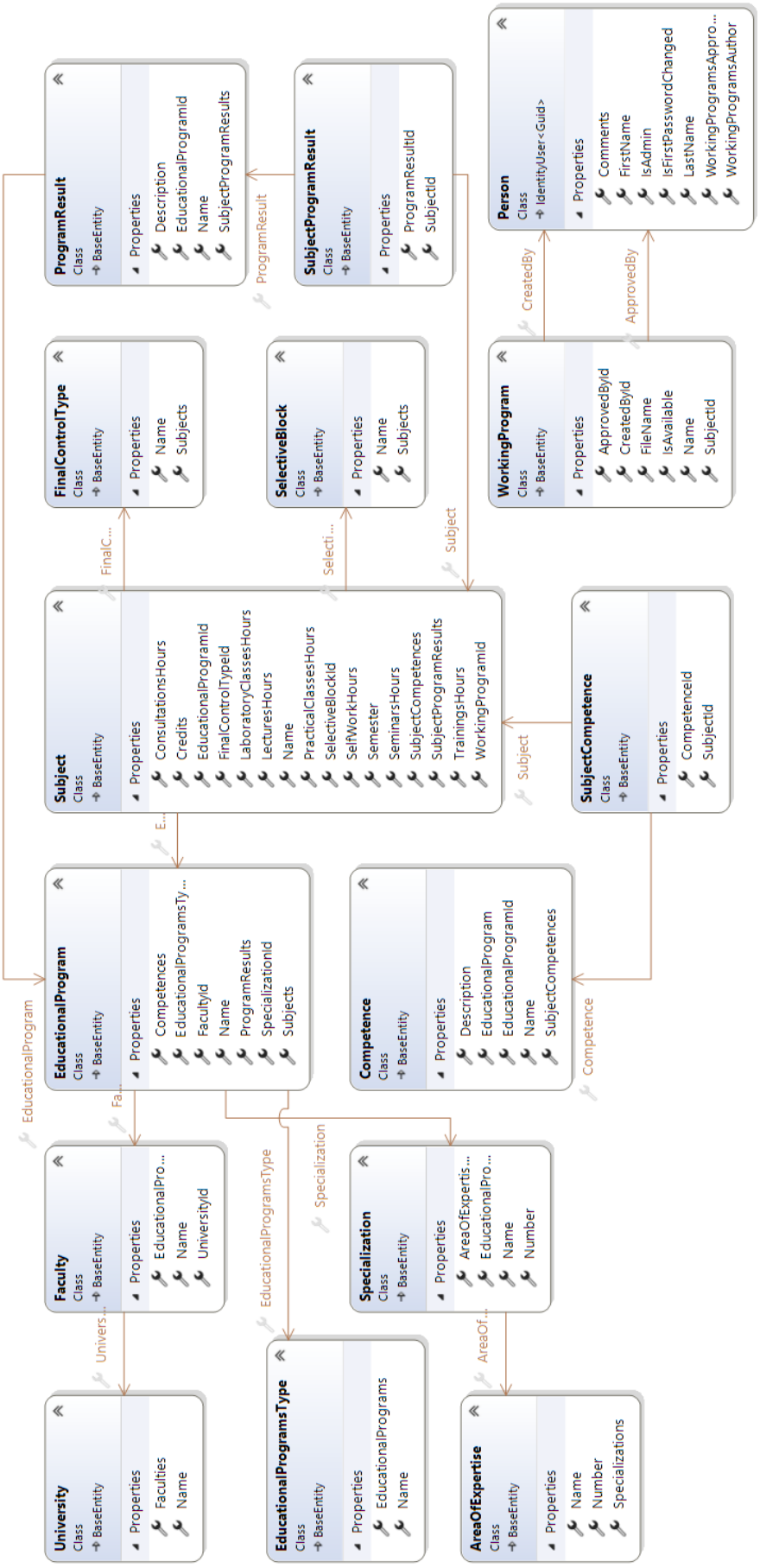
Додаток Б.1 Діаграма бази даних освітніх програм



Додаток Б.2 Діаграма бази даних робочих програм та ідентифікації користувачів

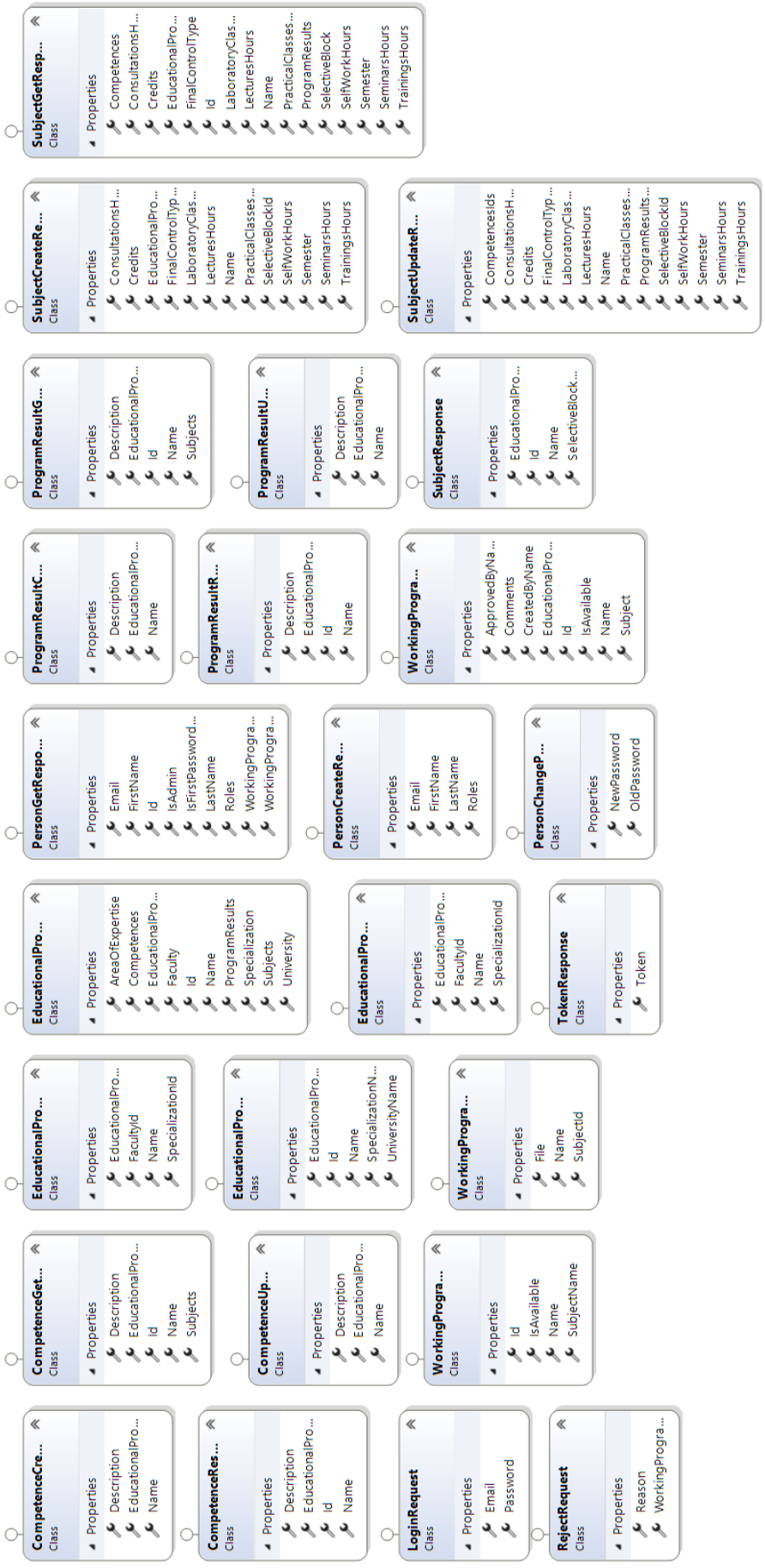


Додаток В.1 Діаграма класів (рівень DAL)



Додаток В.2 Діаграма класів (рівень BLL)

Додаток В.3 Діаграма класів (рівень PL)



Додаток Г Програмний код методу GenerateFile для створення шаблону РПНД

```
public async Task<Result<WorkingProgramModel>> GenerateFile(Guid
subjectId)
{
    var subjectDetailsModel = await GetDetailsModel(subjectId);
    if (subjectDetailsModel == null)
    {
        return
Result.NotFound<WorkingProgramModel>(BlErrors.NotFound(subjectId));
    }

    var file = await
_fileProvider.GetFileAsync(_programSettings.TemplateFileName);
    using var docx = DocX.Load(new MemoryStream(file.Value.Contents));

    var dictionary = ConvertToDictionary(subjectDetailsModel);
    foreach (var item in dictionary)
    {
        docx.ReplaceText(item.Key, item.Value);
    }

    var format = new Formatting
    {
        Size = 12,
    };

    var bulletList = docx.AddList(null, 0, ListItemType.Bulleted);
    foreach (var item in subjectDetailsModel.Competences)
    {
        docx.AddListItem(bulletList, $"{item.Description}.", 0, formatting:
format);
    }

    var paragraph = docx.Paragraphs.FirstOrDefault(p =>
p.Text.Contains("<COMPETENCES>"));
    paragraph?.InsertListAfterSelf(bulletList);
    docx.ReplaceText(CompetenceTag, CompetenceTagValue);

    foreach (var programResult in subjectDetailsModel.ProgramResults)
    {
        docx.Tables[1].InsertRow();
        docx.Tables[1].Rows.Last().Cells[0].Paragraphs[0]
            .Append($"{programResult.Name}. {programResult.Description}",
format);
```

```
    }

    docx.Tables[1].Design = TableDesign.TableGrid;

    LeaveNecessaryGrades(docx, subjectDetailsModel.FinalControlType);

    var stream = new MemoryStream();
    docx.SaveAs(stream);
    stream.Flush();
    var result = new WorkingProgramModel
    {
        FullFileName = $"{subjectDetailsModel.Subject.Name} - шаблон
робочої програми.docx",
        File = stream,
    };

    return Result.Success(result);
}
```

Додаток Д.1 Програмний код функції Azure GetBlobFile

```
[FunctionName("GetBlobFile")]
public static async Task<IActionResult> Run(
    [HttpTrigger(AuthorizationLevel.Function, "get", Route =
"GetBlobFile/{fullFileName}")]
    HttpRequest req, string fullFileName,
    ILogger log, ExecutionContext context)
{
    var config = new ConfigurationBuilder()
        .SetBasePath(context.FunctionAppDirectory)
        .AddJsonFile("local.settings.json", optional: true, reloadOnChange:
true)
        .AddEnvironmentVariables()
        .Build();

    var blobConnection = config["Storage"];
    var container = config["Container"];
    try
    {
        var containerClient = new BlobContainerClient(blobConnection,
container);
        var blobClient = containerClient.GetBlobClient(fullFileName);

        using var stream = new MemoryStream();
        await blobClient.DownloadToAsync(stream);

        var response = new BlobFileResponse
        {
            Contents = stream.ToArray(),
        };

        return new OkObjectResult(response);
    }
    catch (Exception ex)
    {
        var error = "Failed downloading file from Blob Storage";
        log.LogError(error);

        return new BadRequestErrorMessageResult(ex.Message);
    }
}
```

Додаток Д.2 Програмний код функції Azure PostBlobFile

```
[FunctionName("PostBlobFile")]
public static async Task<IActionResult> Run(
    [HttpTrigger(AuthorizationLevel.Function, "post", Route =
"PostBlobFile")]
    BlobFilePostRequest model,
    HttpRequest req,
    ILogger log, ExecutionContext context)
{
    var config = new ConfigurationBuilder()
        .SetBasePath(context.FunctionAppDirectory)
        .AddJsonFile("local.settings.json", optional: true, reloadOnChange:
true)
        .AddEnvironmentVariables()
        .Build();

    var blobConnection = config["Storage"];
    var container = config["Container"];
    if (string.IsNullOrWhiteSpace(model.FileName) || model.Contents ==
null)
    {
        return new BadRequestErrorMessageResult("File or file name is
missing");
    }

    try
    {
        var containerClient = new BlobContainerClient(blobConnection,
container);
        var blobClient =
containerClient.GetBlobClient($"{model.FileName}.docx");

        using var stream = new MemoryStream(model.Contents);
        await blobClient.UploadAsync(stream);

        return new OkResult();
    }
    catch (Exception ex)
    {
        var error = "Failed uploading file to Blob Storage";
        log.LogError(error);

        return new BadRequestErrorMessageResult(ex.Message);
    }
}
```

Додаток Е Згенерований системою шаблон РПНД

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ НАУК ТА КІБЕРНЕТИКИ

Кафедра (циклова комісія) _____
(для коледжів)

«ЗАТВЕРДЖУЮ»
Заступник декана/директора
з навчальної роботи

_____
«____»_____20__ року

РОБОЧА ПРОГРАМА НАВЧАЛЬНОЇ ДИСЦИПЛІНИ СУЧАСНІ ОПЕРАЦІЙНІ СИСТЕМИ

для студентів

галузь знань	**12 «Інформаційні технології»** *(шифр і назва)*
спеціальність	**122 «Комп'ютерні науки»** *(шифр і назва спеціальності)*
освітній рівень	**магістр** *(молодший бакалавр, бакалавр, магістр)*
освітня програма	**«Інформатика»** *(назва освітньої програми)*
вид дисципліни	**обов'язкова**

Форма навчання	**денна**
Навчальний рік	**2023/2024**
Семестр	**3**
Кількість кредитів ECTS	**4**
Мова викладання, навчання та оцінювання	**українська**
Форма заключного контролю	**іспит**

Викладачі: _____
(Науково-педагогічні працівники, які забезпечують викладання даної дисципліни у відповідному навчальному році)

Пролонговано: на 20__/20__ н.р. _____(_____) «__»___ 20__р.
(підпис, ПІБ, дата)

на 20__/20__ н.р. _____(_____) «__»___ 20__р.
(підпис, ПІБ, дата)

КИЇВ – 2023

Розробник(и): *(вказати авторів: ПІБ, науковий ступінь, вчене звання, посада, кафедра)*
_____

ЗАТВЕРДЖЕНО
Зав. кафедри _____
_____ (_____)
(підпис) (прізвище та ініціали)

Протокол № ___ від «____» _____ 20___ р.

Схвалено науково - методичною комісією факультету/інституту
_____

Протокол від «____» _____ 20___ року №___
Голова науково-методичної комісії _____ (_____)
(підпис) (прізвище та ініціали)
Головою педагогічної ради *(для коледжів)*

«_____» _____ 20___ року

ВСТУП

1. Мета дисципліни –

2. Попередні вимоги до опанування або вибору навчальної дисципліни (*за наявності*):

1. *Успішне опанування курсу* _____
2. *Знання теоретичних основ* _____

Або:

1. *Знати* _____
2. *Вміти* _____
3. *Володіти елементарними навичками* _____

3. Анотація навчальної дисципліни:

4. Завдання (навчальні цілі):

Компетентності, на досягнення яких спрямована дана дисципліна:

- Здатність застосовувати методи і засоби організації великих даних для проектування масштабованих інфраструктур консолідації ресурсів зберігання, дослідження, управління, захисту та обслуговування інформації, розв'язання завдань моделювання та прогнозування стратегічних напрямків розвитку бізнесу.
- Здатність вирішувати надскладні наукові та інженерні задачі, що передбачають розпаралелювання обчислень, великих витрат машинного часу, обчислювальних ресурсів і методів організації розв’язання задач на суперкомп’ютерах.
- Розуміння економічних преференцій інноваційного розвитку ІТ підприємств (новітні підходи організації, застосування програмних, апаратних, мережних, математичних, технологічних, ергономічних та інших засобів) з метою вирішення актуальних задач підвищення конкурентоспроможності галузі; здатність розв’язувати складні задачі і проблеми проектування корпоративного інформаційного середовища, що передбачає здійснення інновацій.
- Здатність проектувати та забезпечувати впровадження серверної інфраструктури корпоративного центру обробки даних компанії.

5. Результати навчання за дисципліною:

Результат навчання (1. знати; 2. вміти; 3. комунікація; 4. автономність та відповідальність)		**Форми (та/або методи і технології) викладання і навчання**	**Методи оцінювання та пороговий критерій оцінювання (за необхідності)**	**Відсоток у підсумковій оцінці з дисципліни**
Код	**Результат навчання**			
1.1	*Знати....*	*лекція*	*Тест, 60% правильних відповідей*	*15%*
1.2	*Знати....*	*лекція*	*--//--*	*15%*
2.1	*Вміти*	*лабораторна робота*	*Звіт по лабораторній роботі*	*30%*
4.1	*Прийняти і обгрунтувати рішення.....*	*лекція,семінарське заняття, самостійна робота*	*Кейс-задача*	*40%*

6. Співвідношення результатів навчання дисципліни із програмними результатами навчання

Результати навчання дисципліни Програмні результати навчання	1.1	1.2	2.1	4.1	4.2
ПРН5. Вирішувати складні проблеми, що вимагають систем з великою обчислювальною потужністю для забезпечення масштабованості паралельних алгоритмів і програм.					
ПРН6. Використовувати розподілені високопродуктивні обчислювальні технології для забезпечення ефективного вибору та використання консолідованих ресурсів і послуг.					
ПРН7. Вміти використовувати обчислювальні системи надвеликої потужності для виконання парадигми програмування мультипроцесорних обчислень, розробляти ефективні паралельні алгоритми складних виробничих задач, застосовувати хмарні платформи та їх віртуалізацію.					

7. Схема формування оцінки.

7.1 Форми оцінювання студентів: *(зазначається перелік видів робіт та форм їх контролю / оцінювання із зазначенням результатів навчання які на них мають бути оцінені, а також кількість балів/відсоток у підсумковій оцінці із дисципліни, пороговий рівень позитивної оцінки)*

- семестрове оцінювання:

1. Контрольна робота (тест): РН 1.1.— 10 балів/6 балів.

2. Реферат: РН1.1, РН 2.1. - 20 балів/12 балів.

3. .....

- підсумкове оцінювання (у формі іспиту/комплексного іспиту, диференційованого заліку) *вказується:*

- форма оцінювання (у випадку комплексного екзамену слід вказати питому вагу складових);

- максимальна кількість балів які можуть бути отримані студентом (зазвичай 40 балів по 100-бальній шкалі);

- результати навчання які будуть оцінюватись;

- форма проведення і види завдань (а також їх частка в сукупній оцінці);

*- мінімальний пороговий рівень екзаменаційної оцінки, за якої іспит вважається складеним, наприклад (для випадку коли на екзамен виноситься 40 балів): "**Для отримання загальної позитивної оцінки з дисципліни оцінка за екзамен не може бути меншою 24 балів"1***

*Слід також чітко прописати умови, які висуваються викладачами даної дисципліни як необхідна умова допуску до екзамену: **"Студент не допускається до екзамену, якщо під час семестру набрав менше ніж____ балів2 .** Якщо серед результатів навчання дисципліни є такі які не можуть бути перевірені на екзамені формулюються додаткові вимоги, наприклад: **"Студент допускається до екзамену за умови виконання всіх (або %) передбачених планом лабораторних робіт"***

7.2 Організація оцінювання: *(обов'язково зазначається порядок організації передбачених робочою навчальною програмою форм оцінювання із зазначенням орієнтовного графіку оцінювання).*

7.3 Шкала відповідності оцінок

Відмінно / Excellent	90-100
Добре / Good	75-89
Задовільно / Satisfactory	60-74
Незадовільно / Fail	0-59

8. Структура навчальної дисципліни. Тематичний план лекцій і семінарських / практичних / лабораторних *(вибрати необхідне)* **занять**

№ п/п	Назва теми	Кількість годин		
		лекції	**семінари/ практичні/ лабораторні** *вибрати необхідне*	**Самостійна робота**
Назва розділу чи частини 1*(якщо здійснюється поділ)*				
1	**Вступ**. **Тема 1** *Назва*.	2		2
2	**Тема 2.** *Назва* ……	3		2
	…			
	Тема ХХ. *Назва*	-	-	7
	Контрольна робота 1		Х	
Частина 2 *Назва (за наявності)*				
	Тема ХХХ. *Назва*			
	Підсумкова модульна контрольна робота чи	х	або х	
	Курсова робота з дисципліни *(за наявності у навчальному плані)*			
	ВСЬОГО	**26**	**-**	**80**

Загальний обсяг 120 *год.,* в тому числі:
Лекцій – **26** *год.*
Семінари – **-** *год.*
Практичні заняття - **-** *год.*
Лабораторні заняття - **12** *год.*
Тренінги - **-** *год.*
Консультації - **2** *год.*
Самостійна робота **- 80** *год.*

9. Рекомендовані джерела:

Основна: *(Базова)*
до 10 фундаментальних, базових джерел

Додаткова:
як правило - до 20 джерел

10. Додаткові ресурси (за наявності)**:**

Посилання на електронні ресурси (не тільки відкриті) на яких розміщено додаткову інформацію щодо дисципліни — приклади контрольних і екзаменаційних завдань, тематика рефератів, методичні вказівки по виконанню лабораторних робіт, тощо).