Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра інформатики факультету інформатики

Розробка системи управління освітніми програмами університету

Текстова частина до кваліфікаційної роботи
за спеціальністю “Інженерія програмного забезпечення”

Керівник кваліфікаційної роботи доктор технічних наук, професор, Глибовець А. М.
_____
(підпис)
“___” _____ 2025 р.
Виконав студент Синяк О. Я.
“___” _____ 2025 р.

Київ 2025

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра інформатики факультету інформатики

ЗАТВЕРДЖУЮ

Зав. кафедри інформатики,
доцент, кандидат наук
_____ Гороховський С.С
(підпис)
“____” _____ 2025 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на кваліфікаційну роботу

студенту Синяк Олександру 3-го курсу факультету інформатики

ТЕМА: Розробка системи управління освітніми програмами університету

Зміст ТЧ до курсової роботи:

Індивідуальне завдання
Анотація
Вступ

1. Аналіз предметної області та постановка задачі
2. Структура проєкту та опис використаних технологій
3. Розробка системи управління освітніми програмами

Висновок
Список використаних джерел
Додатки

Дата видачі “___” _____ 2025 р.
Керівник _____ Завдання отримано _____

Календарний План Виконання Кваліфікаційної Роботи

Тема: Розробка системи управління освітніми програмами університету

№ з/п	ПЕРЕЛІК РОБОТ	Термін Виконання	Примітка
1	Отримання теми кваліфікаційної роботи	Листопад - Грудень 2024	
2	Створення плану роботи	Кінець грудня 2024	
3	Ознайомлення з проєктом	Січень 2025	
4	Розробка практичної частини проєкту	Лютий - Березень 2025	
5	Написання першого розділу роботи	Початок квітня 2025	
6	Описання практичної частини роботи	Середина квітня 2025	
7	Перегляд змісту роботи з керівником	Кінець квітня 2025	
8	Внесення змін відповідно до зауважень наукового керівника	Кінець квітня 2025	
9	Створення презентації	Початок травня 2025	
10	Захист дипломної роботи	10.05.2025	

Студент Синяк О. Я. _____

Керівник Глибовець А.М. _____

“_____” _____ 2025 р.

ЗМІСТ

АНОТАЦІЯ 6

ВСТУП 7

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ 8

1.1. Поняття та роль освітньої програми та навчального плану 8

1.2. Визначення структури даних освітнього процесу 9

1.2.1. Характеристики освітньої програми 9

1.2.2. Характеристики навчального плану 13

1.2.3. Функціонал копіювання навчального плану 14

1.3. Аналіз існуючих бізнес-процесів управління ОП 14

1.4. Формулювання функціональних вимог до системи 16

1.5. Підсумок та постановка задачі 17

2. СТРУКТУРА ПРОЄКТУ ТА ОПИС ВИКОРИСТАНИХ ТЕХНОЛОГІЙ 19

2.1. Архітектура проєкту SmartUkma та місце модуля управління ОП 19

2.2. Технологічний стек 21

2.3. Проєктування бази даних 23

2.4. Підсумок розділу 26

3. РОЗРОБКА СИСТЕМИ УПРАВЛІННЯ ОСВІТНІМИ ПРОГРАМАМИ 27

3.1. Загальний підхід до реалізації 27

3.2. Реалізація серверної частини (Backend) 27

3.3. Реалізація інтерфейсу користувача (Frontend) 31

3.4. Підсумки реалізації *33*

ВИСНОВОК **35**

СПИСОК ЛІТЕРАТУРИ **37**

ДОДАТКИ **38**

Додаток А. ER діаграма *38*

Додаток Б. Приклад визначення класу EducationalProgramEntity *39*

Додаток В. Приклад методу сервісу з управлінням транзакцією *40*

Додаток Г. Приклад роботи контролера та використання DTO *41*

Додаток Ґ. Схема навігації в модулі освітніх програм *42*

Додаток Д. Загальний вигляд інтерфейсу модуля *42*

Додаток Е. Знімок екрану детального перегляду освітньої програми *43*

Додаток Є. Знімок екрану форми редагування освітньої програми *44*

Додаток Ж - Інтерфейс управління навчальними планами та кнопка копіювання *46*

АНОТАЦІЯ

Кваліфікаційна робота присвячена розробці системи управління освітніми програмами в інформаційній системі SmartUKMA. Робота розглядає проблеми ручного управління освітніми програмами та обґрунтовує необхідність їх автоматизації.

В першому розділі, проведено аналіз предметної області, визначено функціональні та нефункціональні вимоги до системи.

В другому розділі, описано архітектуру розробленого модуля в контексті мікросервісної архітектури проєкту SmartUKMA, представлено використаний технологічний стек, спроєктовано структуру бази даних та REST API.

В останньому розділі, детально описано практичну реалізацію серверної та клієнтської частин програми, включаючи функції створення, редагування, перегляду освітніх програм та управління пов'язаними даними (вимоги ЗНО, переваги, гаранти, файли).

Результатом роботи є частина функціоналу, що автоматизує процеси управління освітніми програмами, підвищує ефективність роботи відповідних підрозділів, підтримує узгодженість і доступність інформації.

ВСТУП

Успішна діяльність сучасного університету неможлива без ефективного управління освітнім процесом. Освітні програми є його основою, визначаючи зміст, мету та якість підготовки студентів. В умовах динамічного розвитку суспільства, ринку праці та освітніх стандартів, університети стикаються з необхідністю постійного оновлення та адаптації своїх освітніх програм. Однак, традиційні підходи до управління цією сферою, що часто спираються на паперовий документообіг та ручну обробку даних, демонструючи свою неефективність.

Сучасна цифровізація освіти ставить перед університетами нові виклики щодо ефективної організації навчального та адміністративного процесів. Перехід до автоматизованих систем управління стає не просто бажаним, а необхідним кроком для забезпечення конкурентоспроможності та підвищення якості освітніх послуг.

Метою даної кваліфікаційної роботи є розробка програмного модуля, призначеного для автоматизації роботи з освітніми програмами, та його подальша інтеграція в загальну систему SmartUKMA. Це дозволить оптимізувати процеси створення, супроводу, оновлення та представлення інформації про освітні програми університету.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Поняття та роль освітньої програми та навчального плану

В основі освітнього процесу будь-якого сучасного університету лежить освітня програма (ОП). Це комплексний нормативний документ, що визначає цілі навчання, перелік, зміст, послідовність та організаційні форми вивчення навчальних дисциплін, практик, а також форм підсумкової атестації здобувачів вищої освіти певного рівня (бакалаврського, магістерського, докторського) та спеціальності. Освітня програма є ключовим інструментом реалізації освітньої політики університету та забезпечення якості вищої освіти. Вона окреслює очікувані результати навчання, яких має досягти випускник.

Тісно пов'язаним з освітньою програмою є поняття навчального плану (НП). Якщо освітня програма визначає що вивчати та які компетентності набути, то навчальний план деталізує як і коли це буде відбуватися для конкретної групи студентів конкретного року вступу. Навчальний план є документом, що розробляється на основі освітньої програми та стандарту вищої освіти і містить детальний перелік навчальних дисциплін із зазначенням обсягу їх вивчення в кредитах Європейської кредитної трансферно-накопичувальної системи (ЄКТС), послідовності вивчення за семестрами, видів навчальних занять (лекції, семінари, практичні), форм підсумкового контролю (іспит, залік) та графіку навчального процесу. Таким чином, одна освітня програма може мати декілька навчальних планів, що актуалізуються для різних років набору студентів.

1.2. Визначення структури даних освітнього процесу

Для побудови ефективної системи управління освітніми програмами важливо детально визначити структуру даних ключових елементів освітнього процесу. Це включає не лише саму освітню програму, але й тісно пов'язані з нею сутності, такі як навчальний план, дисципліни та інформація про студентів. Цей підрозділ надає детальний опис мета-інформації для цих об'єктів, визначаючи необхідні атрибути, їх характеристики та призначення.

1.2.1. Характеристики освітньої програми

Освітня програма (ОП) є центральним об'єктом системи, що акумулює комплексну інформацію про підготовку фахівців певного рівня та спеціальності. Для її однозначної ідентифікації та повного опису в системі передбачено наступний набір характеристик:

- Ідентифікатор програми - обов'язкове системне поле; являє собою унікальний числовий ідентифікатор, який автоматично генерується системою при створенні нового запису про ОП і використовується для посилань та внутрішньої ідентифікації.
- Назва програми - обов'язкове поле; текстове значення, що містить повну офіційну назву освітньої програми.
- Короткий опис - обов'язкове поле; текстова анотація програми, що надає стислу характеристику її змісту та мети. Використовується для попереднього ознайомлення, наприклад, у списках програм.
- Повний опис - необов'язкове поле; розгорнутий текстовий опис програми, що може включати детальну інформацію про мету, цілі, структуру, методики навчання, унікальні особливості, кар'єрні перспективи тощо.
- Спеціальність - обов'язкове поле; посилання на відповідний запис у системному довіднику спеціальностей. Визначає код та офіційну назву спеціальності згідно з державним класифікатором, а також галузь знань, до якої вона належить. Це забезпечує стандартизацію та можливість формування звітності за спеціальностями.

- Спеціалізація - необов'язкове поле; текстове значення, що вказує на конкретне фахове спрямування в межах основної спеціальності, якщо таке передбачено освітньою програмою.
- Тип програми - обов'язкове поле; вибір зі стандартизованого переліку значень, що визначають основну мету та спрямованість програми. Можливі значення:
  Освітньо-професійна,
  Освітньо-наукова,
  Освітньо-творча.
- Рівень програми - обов'язкове поле; вибір зі стандартизованого переліку рівнів вищої освіти, на здобуття якого спрямована програма. Можливі значення: Бакалавр , Магістр, Доктор філософії.
- Факультет - обов'язкове поле; посилання на запис у довіднику факультетів. Визначає основний структурний підрозділ університету, відповідальний за адміністрування та забезпечення програми.
- Кафедра - необов'язкове поле; посилання на запис у довіднику кафедр. Вказує на випускову кафедру, що реалізує основну частину фахової підготовки. Важливою є перевірка належності обраної кафедри до факультету, вказаного для програми.
- Тривалість навчання - необов'язкове поле; числове значення, що вказує на нормативний термін здобуття освіти за даною програмою (наприклад, у роках або місяцях).
- Вартість навчання - необов'язкове поле; числове значення, що відображає вартість навчання за певний період (зазвичай, рік) для студентів контрактної форми.
- Кількість бюджетних місць - необов'язкове поле; ціле числове значення, що вказує на обсяг державного замовлення для даної програми на певний рік.
- Кількість контрактних місць - необов'язкове поле; ціле числове значення, що визначає ліцензійний обсяг для навчання на контрактній основі.

- Мови викладання - необов'язкове поле; список, що містить посилання на одну або декілька мов з відповідного довідника. Визначає мови, якими здійснюється викладання дисциплін програми.
- Вимоги ЗНО/НМТ - необов'язкове поле; структурований список вимог до сертифікатів зовнішнього тестування для вступу. Для кожної вимоги вказується предмет тестування (посилання на довідник предметів ЗНО/НМТ), обов'язковий мінімальний бал та статус обов'язковості (наприклад, обов'язковий конкурсний предмет, вибірковий предмет).
- Мінімальний бал (бюджет) - необов'язкове поле; числове значення (дійсне число) в діапазоні від 0 до 200. Відображає пороговий конкурсний бал для вступу на бюджетну форму навчання, часто базується на статистиці минулих років.
- Мінімальний бал (контракт) - необов'язкове поле; числове значення (дійсне число) в діапазоні від 0 до 200. Аналогічно визначає пороговий бал для вступу на контракт.
- Мінімальний вступний бал - необов'язкове поле; ціле числове значення. Загальний мінімальний бал (наприклад, сума балів з НМТ/ЗНО або середній бал атестату), необхідний для допуску до участі в конкурсному відборі на програму.
- Інші умови вступу - необов'язкове поле; текстовий опис додаткових критеріїв або процедур відбору, таких як творчі конкурси, фахові випробування, співбесіди, вимоги до мотиваційного листа тощо.
- Файл програми вступних випробувань - необов'язкове поле; механізм для завантаження та зберігання одного або декількох файлів (зазвичай у форматі PDF), що містять офіційну програму вступних випробувань (якщо такі передбачені). Система зберігає посилання на файл та його назву.
- Контактна особа (основна) - обов'язкове поле; посилання на запис у довіднику контактних осіб університету. Зазвичай це гарант програми, керівник або інша відповідальна особа.

- Контакт Бадді - необов'язкове поле; посилання на запис у довіднику контактних осіб. Представляє студента-помічника або іншу особу, що консультує абітурієнтів.
- Гаранти програми - необов'язкове поле; список осіб, що є офіційно затвердженими гарантами даної освітньої програми. Для кожного гаранта система зберігає посилання на його обліковий запис (або email) та рік призначення/досягнення.
- Статус програми - обов'язкове поле; вибір зі списку статусів, що визначають стан програми в її життєвому циклі. Основні статуси: Відкрита (OPENED) - програма доступна для вступу, Закрита (CLOSED) – набір не ведеться або програма архівована. За замовчуванням при створенні встановлюється статус CLOSED.
- Посилання на детальний опис - необов'язкове поле; текстове значення, що містить повну URL-адресу сторінки з описом програми на офіційному сайті університету або факультету.
- Посилання на відео - необов'язкове поле; текстове значення, що містить повну URL-адресу презентаційного відеоролика про програму (наприклад, на платформі YouTube).
- Кількість заяв (попередній рік) - необов'язкове поле; ціле числове значення, що відображає кількість поданих заяв на програму під час попередньої вступної кампанії. Має бути невід'ємним.
- Переваги програми - необов'язкове поле; набір структурованих записів, що описують унікальні переваги даної ОП. Кожен запис про перевагу містить:
    - *Назва*
    - *Опис*
- Стипендії - необов'язкове поле; список, що містить посилання на одну або декілька стипендіальних програм з відповідного довідника, які доступні для студентів, що навчаються за цією ОП.

1.2.2. Характеристики навчального плану

Навчальний план — це документ, що забезпечує практичне втілення освітньої програми для конкретного року вступу. Він має наступні характеристики в системі:

- Ідентифікатор плану - обов'язкове системне поле; унікальний числовий ідентифікатор навчального плану.
- Освітня програма - обов'язкове поле; посилання на освітню програму (EducationalProgramEntity), яку цей план деталізує.
- Рік вступу - обов'язкове поле; посилання на запис у довіднику навчальних років, що однозначно ідентифікує групу студентів, для якої розроблено цей план. Важливою вимогою є унікальність комбінації освітньої програми та року вступу – не може існувати двох планів для однієї ОП та одного року.
- Дата створення - обов'язкове системне поле; дата створення запису про НП в системі, встановлюється автоматично.
- Дата підписання - необов'язкове поле; дата офіційного затвердження НП.
- Статус плану - обов'язкове поле; вибір зі списку статусів, що відображають етап життєвого циклу плану:

  У Розробці,

  Підписаний,

  Архівований.

  При створенні плану йому автоматично надається статус Чернетка.
- Дисципліни плану - необов'язкове поле; список посилань на дисципліни, що входять до цього навчального плану. Детальна інформація про кожну дисципліну зберігається окремо.
- Студенти плану - необов'язкове поле; список посилань на студентів, які навчаються за цим планом. Система повинна контролювати відповідність факультету студента та факультету освітньої програми плану.

1.2.3. Функціонал копіювання навчального плану

Для оптимізації роботи зі створення навчальних планів, які часто зберігають свою структуру протягом кількох років з незначними змінами, в системі передбачено функціонал копіювання. Користувач з відповідними правами може обрати існуючий навчальний план як основу (джерело) та вказати новий навчальний рік, на який потрібно створити копію. Система автоматично створює новий запис навчального плану, пов'язаний з тією ж освітньою програмою, що й джерело, але з вказаним новим роком вступу. Новостворений план отримує статус “У Розробці” (DRAFT), що дозволяє внести необхідні корективи перед його затвердженням. Залежно від налаштувань, система може також автоматично копіювати перелік дисциплін з плану-джерела. Цей механізм значно скорочує час на підготовку документації на новий навчальний рік, мінімізуючи рутинні операції.

1.3. Аналіз існуючих бізнес-процесів управління ОП

Поточні процеси управління освітніми програмами в НаУКМА, як і в багатьох інших ЗВО, поєднують елементи паперового документообігу, електронної пошти та використання локальних або розрізнених інформаційних систем. Схематично, життєвий цикл ОП виглядає наступним чином:

1. Ініціація/Розробка: Кафедра або група викладачів розробляє проєкт нової ОП або пропозиції щодо оновлення існуючої. Це включає формування опису, навчальних планів, силабусів дисциплін.
2. Погодження: Проєкт ОП проходить процедуру погодження на рівні кафедри, факультету (науково-методична комісія, Вчена рада факультету), навчально-методичного відділу університету, інших зацікавлених підрозділів. Цей етап часто включає численні ітерації правок та обговорень, що відбуваються через службові записки, електронну пошту, наради.

3. Затвердження: Фінальна версія ОП затверджується Вченою радою університету та вводиться в дію наказом ректора.
4. Внесення даних: Інформація про затверджену ОП (атрибути, навчальні плани, дисципліни) вноситься співробітниками різних підрозділів (деканати, навчальний відділ) у відповідні системи обліку та на офіційний веб-сайт університету.
5. Актуалізація: Протягом життєвого циклу ОП періодично оновлюється відповідно до змін у законодавстві, стандартах освіти, потреб ринку праці або за результатами внутрішнього моніторингу. Процес оновлення повторює етапи погодження та затвердження.
6. Архівування/Закриття: Після завершення терміну дії або припинення набору ОП переводиться в архівний статус.

Ці процеси, особливо при ручному або частково автоматизованому виконанні, мають суттєві недоліки:

- Висока трудомісткість: Багаторівневі погодження, необхідність підготовки паперових копій, ручне внесення даних у різні системи потребують значних затрат часу відповідальних співробітників.
- Тривалість циклу: Процес від розробки до затвердження та публікації ОП може займати тривалий час, що сповільнює реакцію університету на зовнішні зміни.
- Ризик помилок та неузгодженості: Ручне перенесення даних між документами та системами збільшує ймовірність помилок. Інформація в різних джерелах (сайт, внутрішні бази, паперові документи) може відрізнятися та бути неактуальною.
- Складність контролю: Важко відстежити поточний статус погодження документа, відповідальних осіб на кожному етапі, а також керувати версіями ОП та пов'язаних документів.

Ці недоліки підкреслюють гостру необхідність впровадження єдиної інтегрованої системи управління освітніми програмами, яка б автоматизувала

ключові процеси, централізувала дані та забезпечила їхню цілісність, актуальність і доступність.

1.4. Формулювання функціональних вимог до системи

На основі проведеного аналізу предметної області, структури даних та недоліків існуючих бізнес-процесів, можна сформулювати наступні функціональні вимоги до розроблюваної системи управління освітніми програмами:

1. Управління основними даними ОП (CRUD): Система повинна надавати авторизованим користувачам (адміністраторам, співробітникам деканатів/кафедр) інтерфейс для:
    - Створення нових записів про освітні програми з можливістю введення всіх визначених атрибутів.
    - Редагування інформації існуючих освітніх програм.
    - Перегляду детальної інформації про будь-яку ОП, включаючи всі її атрибути та пов'язані дані.
2. Відображення списку ОП та пошук/фільтрація: Система має забезпечити можливість перегляду списку всіх освітніх програм у вигляді таблиці. Необхідно реалізувати гнучкі механізми:
    - Пагінації: Для зручної роботи з великою кількістю програм.
    - Фільтрації: За ключовими атрибутами, такими як факультет, кафедра, рівень програми, статус програми, спеціальність.
    - Повнотекстового пошуку: За назвою.
3. Управління пов'язаними сутностями: В інтерфейсі редагування/створення ОП користувач повинен мати можливість:
    - Додавати, редагувати та видаляти вимоги ЗНО, вказуючи предмет, мінімальний бал та тип обов'язковості.
    - Додавати, редагувати та видаляти переваги програми, вводячи їх назву та опис.

- Додавати, редагувати та видаляти гарантів програми, вказуючи користувача (за email або ПІБ) та рік досягнення.
- Вибирати зі списків (довідників) факультет, кафедру, спеціальність, контактних осіб, бадді, мови викладання, стипендії.
- Завантажувати та оновлювати файл програми вступних випробувань.

4. Управління зв'язком з навчальними планами: Система повинна:
   - Відображати список навчальних планів, що асоційовані з конкретною ОП.
   - Надавати можливість копіювання існуючого навчального плану на новий навчальний рік в контексті даної ОП.
5. Розмежування доступу: Доступ до функціоналу системи має бути чітко регламентований:
   - Перегляд списку програм та їх деталей (з урахуванням статусу OPENED) має бути доступний широкому колу користувачів.
   - Створення та редагування ОП має бути доступне лише визначеним ролям (наприклад, адміністраторам системи, співробітникам, відповідальним за ОП на факультетах/кафедрах).

Виконання цих функціональних вимог дозволить створити систему, що ефективно вирішує завдання автоматизації управління освітніми програмами.

1.5. Підсумок та постановка задачі

Проведений аналіз предметної області управління освітніми програмами в університеті виявив значний потенціал для оптимізації та автоматизації існуючих процесів. Ключовими проблемами є фрагментація даних, висока трудомісткість ручних операцій, ризик виникнення помилок та недостатня оперативність в оновленні та поширенні інформації.

Для вирішення цих проблем поставлено задачу розробки програмного модуля (системи) управління освітніми програмами в рамках інтегрованої

інформаційної системи університету SmartUkma. Головною метою розробки є створення централізованого, надійного та зручного інструменту, який дозволить автоматизувати процеси створення, редагування, зберігання, пошуку та представлення інформації про освітні програми та пов'язані з ними дані на рівні факультету та університету.

2. СТРУКТУРА ПРОЄКТУ ТА ОПИС ВИКОРИСТАНИХ ТЕХНОЛОГІЙ

2.1. Архітектура проєкту SmartUkma та місце модуля управління ОП

Інформаційна система SmartUkma, в рамках якої розробляється модуль управління освітніми програмами, побудована на принципах мікросервісної архітектури. Цей підхід був обраний через низку переваг, критично важливих для складних та багатофункціональних систем університетського рівня. Розбиття системи на невеликі, автономні сервіси, кожен з яких відповідає за окрему бізнес-логіку, дозволяє:

- Незалежно розробляти та оновлювати окремі компоненти системи без значного впливу на інші частини. Це прискорює цикл розробки та полегшує впровадження нових функцій або виправлення помилок.
- Ефективно масштабувати систему, збільшуючи ресурси лише для тих сервісів, які зазнають найбільшого навантаження.
- Підвищити надійність системи загалом, оскільки відмова одного сервісу не обов'язково призведе до зупинки роботи всієї платформи.
- Покращити структуру та підтримуваність коду завдяки чіткому розмежуванню відповідальності.

Модуль управління освітніми програмами (ОП) логічно та функціонально інтегрується в цю архітектуру і тісно взаємодіє з кількома ключовими мікросервісами:

1. user-info-server: Це центральний сервіс, що відповідає за зберігання та управління основною довідковою інформацією університету. Саме в рамках user-info-server реалізовано значну частину логіки, пов'язаної з освітніми програмами, оскільки ОП є фундаментальною інформаційною сутністю. Цей сервіс надає API для роботи з ОП та пов'язаними

довідниками, такими як факультети, кафедри, спеціальності, контактні особи, мови, стипендії, переваги, вимоги ЗНО, гаранти та навчальні плани.

2. ui-server: Виступає як сервер для клієнтської частини, розробленої на Angular, а також як бекенд-проксі. Він приймає HTTP-запити від браузера користувача, обробляє їх та перенаправляє до відповідних мікросервісів (в нашому випадку, переважно до info-server для операцій з ОП).
3. account-server: Відповідає за процеси автентифікації та авторизації. Модуль управління ОП взаємодіє з ним опосередковано (через ui-server або безпосередньо з info-server) для перевірки прав доступу користувача до виконання певних операцій (створення, редагування, видалення ОП).

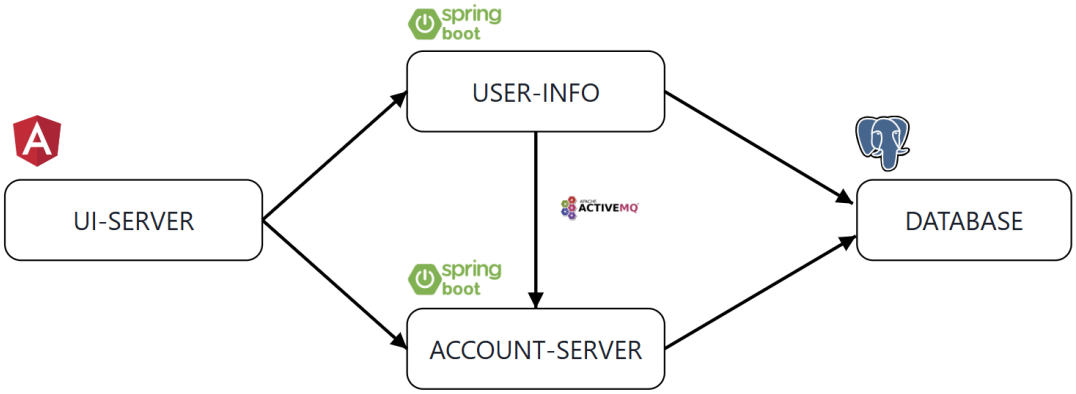


Рисунок 2.1 Діаграма архітектури

Основний механізм комунікації між сервісами – синхронні виклики через REST API. Для забезпечення уніфікації та надійності інтерфейсів використовується підхід “Contract-first” на основі специфікації OpenAPI 3.0. Це дозволяє чітко визначити структуру запитів та відповідей і автоматично генерувати значну частину коду як на серверній (контролери, DTO), так і на клієнтській стороні (сервіси для виклику API) . Для асинхронних завдань, таких як відправка сповіщень, в системі SmartUkma також використовуються черги повідомлень (наприклад, ActiveMQ), хоча для основного функціоналу управління ОП вони менш релевантні.

2.2. Технологічний стек

Вибір технологій для реалізації модуля управління освітніми програмами ґрунтується на загальноприйнятому в проєкті SmartUkma стеку, що забезпечує сучасність, надійність та ефективність розробки:

Серверна частина (Backend):

- Java 11: Як основна мова програмування обрано Java версії 11. Це LTS (Long-Term Support) версія, що гарантує стабільність, широку підтримку спільноти та сумісність з великою кількістю бібліотек. Java є стандартом для багатьох ентерпрайз-систем завдяки своїй надійності та продуктивності.
- Spring Boot 2: Фреймворк, що суттєво прискорює та спрощує розробку Java-застосунків. Він забезпечує автоматичну конфігурацію, вбудований веб-сервер, зручні інструменти для створення REST API, інтеграції з базами даних, забезпечення безпеки та управління залежностями.
- Spring Data JPA / Hibernate: Для взаємодії з базою даних використовується стандарт JPA (Java Persistence API) з реалізацією Hibernate ORM. Це дозволяє абстрагуватися від деталей SQL-запитів, працювати з даними як з Java-об'єктами (Object-Relational Mapping) та ефективно управляти транзакціями й кешуванням.

Клієнтська частина (Frontend):

- Angular 16+: Популярний та потужний TypeScript-фреймворк для створення динамічних односторінкових застосунків (SPA). Його компонентна архітектура, система модулів, вбудовані інструменти для маршрутизації та управління формами (Reactive Forms) сприяють створенню структурованих та підтримуваних інтерфейсів.
- TypeScript: Мова програмування, що розширює JavaScript статичною типізацією. Це значно підвищує надійність коду, полегшує його читання, рефакторинг та виявлення помилок на етапі розробки.

- RxJS: Бібліотека для реактивного програмування з використанням потоків даних (Observables). Активно використовується в Angular для обробки асинхронних операцій, таких як HTTP-запити до API, та для управління станом компонентів.

База даних:

PostgreSQL 13+: Обрана як основна система управління базами даних. Це потужна, реляційна СУБД з відкритим кодом, що відома своєю надійністю, відповідністю стандартам SQL та ACID, підтримкою складних запитів та типів даних, а також хорошою продуктивністю. Її використання є виправданим для зберігання структурованих даних освітніх програм та забезпечення їх цілісності.

API та Інтеграція:

OpenAPI Specification 3.0: Використовується для формального опису REST API у форматі YAML. Це слугує єдиним джерелом правди про контракт між сервером та клієнтом.

OpenAPI Generator: Інструменти (openapi-generator-maven-plugin для Java, openapi-generator-cli для TypeScript) використовуються для автоматичної генерації коду контролерів, моделей даних (DTO/View) та клієнтських сервісів на основі OpenAPI специфікації. Це значно зменшує кількість рутинного коду та ризик помилок при інтеграції.

Інфраструктура та CI/CD:

Docker / Docker Compose: Технології контейнеризації, що дозволяють “упакувати” кожен мікросервіс з усіма його залежностями в ізольований контейнер. Docker Compose використовується для оркестрації запуску декількох контейнерів (база даних, мікросервіси) в середовищі розробки, забезпечуючи консистентність оточення.

Git / GitLab: Система контролю версій Git використовується для управління кодовою базою. GitLab виступає як платформа для хостингу репозиторіїв, управління проєктами та реалізації процесів CI/CD.

GitLab CI/CD: Система безперервної інтеграції та доставки, налаштована для кожного мікросервісу. Вона автоматично виконує збірку проєкту, запуск

тестів, публікацію артефактів (бібліотек) у внутрішній Maven репозиторій (GitLab Package Registry) та розгортання на тестові/продуктивні середовища при змінах у коді.

Maven: Система автоматизації збірки та управління залежностями для Java-проєктів. Використовується для компіляції коду, тестування, пакування та публікації артефактів.

Обраний технологічний стек є збалансованим, використовує перевірені та широко розповсюджені інструменти, що дозволяє створювати надійний, масштабований та сучасний програмний продукт.

2.3. Проєктування бази даних

Для спрощення взаємодії між об'єктно-орієнтованим кодом на Java (серверна частина) та реляційною структурою бази даних PostgreSQL використовується технологія Object-Relational Mapping (ORM), реалізована за допомогою фреймворку Hibernate у поєднанні зі стандартом JPA (Java Persistence API). ORM дозволяє розробникам працювати з даними бази даних як зі звичайними Java-об'єктами (Entities), абстрагуючись від деталей написання SQL-запитів для більшості операцій. Hibernate автоматично генерує необхідні SQL-запити для створення, читання, оновлення та видалення даних (CRUD), а також для роботи зі зв'язками між об'єктами.

Центральним елементом спроєктованої схеми даних є таблиця educational_programs. Вона виступає як основне сховище для всіх ключових атрибутів освітньої програми, детально описаних у Розділі 1.2.1. Ця таблиця містить стовпці для зберігання унікального ідентифікатора (id), назв та описів програми різними мовами (title_ua, title_en, description_ua тощо), класифікаційних ознак (program_level, program_type), параметрів навчання (duration, price, budget_places, contract_places), умов вступу, статусів, посилань та

інших характеристик. Саме ця таблиця є точкою зв'язку з більшістю інших таблиць системи, що зберігають довідкову або деталізуючу інформацію.

Для забезпечення консистентності та уникнення дублювання даних активно використовуються довідникові таблиці. Ці таблиці зберігають стандартизовану інформацію, яка може використовуватися багатьма освітніми програмами. З таблицею educational_programs встановлено зв'язки (зазвичай “багато до одного” з боку ОП) до таких довідників:

faculties: Містить перелік факультетів університету.

departments: Містить перелік кафедр із зазначенням їхньої належності до факультету.

knowledge_speciality: Зберігає офіційний перелік спеціальностей та їх кодів.

person_contacts: Довідник контактних осіб (співробітників, студентів), що використовуються для полів “Контактна особа” та “Контакт Бадді”.

languages: Перелік мов викладання.

scholarship: Довідник стипендіальних програм.

year_of_studying: Довідник навчальних років (напр., 2023-2024), необхідний для прив'язки навчальних планів.

zno_discipline: Довідник предметів ЗНО/НМТ.

Зв'язок реалізується через зовнішні ключі в таблиці educational_programs (наприклад, faculty_id, department_id, knowledge_speciality_id, contact_person_id, buddy_contact_id), які посилаються на первинні ключі відповідних довідникових таблиць.

Для моделювання зв'язків типу “багато до багатьох” використовуються проміжні таблиці. Наприклад:

educational_program_lang: Зв'язує educational_programs та languages, дозволяючи вказати кілька мов викладання для однієї програми. Містить два зовнішні ключі: ep_id та lang_id.

education_program_scholarships: Аналогічно зв'язує educational_programs та scholarship, дозволяючи асоціювати програму з кількома стипендіями. Містить ключі education_program_id та scholarship_id.

Існують також таблиці, що зберігають деталізуючу інформацію, яка має відношення “один до багатьох” з освітньою програмою:

curriculum: Таблиця навчальних планів. Кожен запис пов'язаний з однією освітньою програмою (ed_program_id) та одним роком вступу (year_of_entrance_id). Одна ОП може мати багато навчальних планів.

educational_program_zno_disciplines: Зберігає специфічні вимоги ЗНО для конкретної ОП. Кожен запис пов'язаний з однією ОП (educational_program_id) та одним предметом ЗНО (zno_discipline_id) і містить додаткові поля (minimal_bal, obligations_type).

advantages: Зберігає записи про переваги програми, кожен з яких пов'язаний з однією ОП (education_program_id).

entry_step_files: Зберігає метадані (ID файлу, назва) про файли програм вступних випробувань, кожен запис пов'язаний з однією ОП (education_program_id).

educational_program_guarantors: Зберігає інформацію про гарантів програми, кожен запис прив'язаний до однієї ОП (education_program_id).

Таблиця curriculum, у свою чергу, має зв'язок “один до багатьох” з таблицею discipline (через поле curriculum_id в таблиці discipline), що дозволяє визначити перелік дисциплін для кожного навчального плану.

Посилальна цілісність між усіма цими таблицями забезпечується за допомогою обмежень зовнішніх ключів (FOREIGN KEY constraints) на рівні бази даних. Це гарантує, що неможливо створити запис, який посилається на неіснуючий запис в іншій таблиці, або видалити запис, на який існують посилання (без використання каскадних операцій). Для забезпечення швидкого доступу до даних використовуються індекси, які автоматично створюються для первинних ключів та можуть бути додатково створені для зовнішніх ключів та стовпців, що часто використовуються у запитах фільтрації чи сортування.

Візуальне представлення описаної структури таблиць та зв'язків між ними, що стосуються модуля управління освітніми програмами, наведено у ER-діаграмі в *(див. Додаток А - ER діаграма)*. Ця схема даних є основою для

зберігання інформації та забезпечує необхідну гнучкість та цілісність для функціонування системи.

2.4. Підсумок розділу

У даному розділі представлено комплексний огляд архітектурних та технологічних рішень, застосованих при розробці модуля управління освітніми програмами в рамках проєкту SmartUkma. Вибір мікросервісної архітектури для всієї системи забезпечує необхідну гнучкість та масштабованість. Розміщення логіки управління ОП в межах сервісу info-server дозволяє тісно інтегрувати її з основними довідниками університету.

Детально розглянуто проєктування реляційної бази даних, включаючи ключові таблиці (educational_programs, curriculum та ін.) та зв'язки між ними, що забезпечують цілісність та структурованість даних. Також описано підхід до проєктування REST API на основі OpenAPI специфікації, що гарантує чіткий контракт взаємодії між бекендом та фронтендом.

3. РОЗРОБКА СИСТЕМИ УПРАВЛІННЯ ОСВІТНІМИ ПРОГРАМАМИ

3.1. Загальний підхід до реалізації

Цей розділ присвячений висвітленню процесу створення модуля управління освітніми програмами, що ґрунтується на вимогах, сформульованих у Розділі 1, та архітектурно-технологічних рішеннях, детально описаних у Розділі 2. Основною метою даного етапу було перетворення концептуальної моделі та технічних специфікацій у функціональний, надійний та зручний у використанні програмний модуль, інтегрований в інформаційну систему SmartUkma.

Реалізація модуля передбачала паралельну, але взаємоузгоджену розробку двох основних частин: серверної логіки (Backend) та клієнтського інтерфейсу (Frontend).

3.2. Реалізація серверної частини (Backend)

Реалізація серверної частини відбувалася в рамках мікросервісу info-server на платформі Java з використанням фреймворку Spring Boot, що забезпечило тісну інтеграцію з іншими університетськими даними та сервісами.

Фундаментом для роботи з даними стало визначення об'єктної моделі. У коді були створені спеціальні Java-класи, які називаються Entities (Сутності), наприклад, EducationalProgramEntity, CurriculumEntity та інші. Кожен такий клас детально описує структуру відповідного об'єкта (освітньої програми, навчального плану тощо), включаючи всі його атрибути (назви, описи, дати, статуси, числові параметри), як це було визначено в Розділі 1.2. Ці класи є своєрідним “кресленням” для даних. Для того, щоб ці Java-об'єкти могли зберігатися в реляційній базі даних PostgreSQL та завантажуватися з неї, використовується технологія ORM (Object-Relational Mapping), реалізована за

допомогою Hibernate та стандарту JPA. Спеціальні анотації в коді Entity-класів (@Entity, @Table, @Column, @Id та інші) вказують Hibernate, як саме поля класів відповідають таблицям та стовпцям у базі даних *(див. Додаток Б - Приклад визначення класу EducationalProgramEntity)*. Також за допомогою анотацій (@ManyToOne, @OneToMany, @ManyToMany) детально описуються зв'язки між різними сутностями, наприклад, вказується, що освітня програма належить до одного факультету (@ManyToOne) або що вона може мати багато навчальних планів (@OneToMany). Правильне налаштування цих зв'язків та стратегій їх завантаження (Fetch Strategy - чи завантажувати пов'язані дані одразу, чи тільки за потреби) є важливим для забезпечення як цілісності даних, так і оптимальної продуктивності системи.

Для безпосередньої взаємодії з базою даних (виконання операцій читання, запису, оновлення, видалення) використовується підхід Repositories (Сховища даних), реалізований за допомогою Spring Data JPA. Для кожної Entity створюється відповідний інтерфейс-репозиторій (наприклад, EducationalProgramRepository, CurriculumRepository). Перевага цього підходу полягає в тому, що Spring Data JPA автоматично генерує реалізацію для більшості стандартних методів (зберегти, знайти за ідентифікатором, знайти всі, видалити, підрахувати кількість), що значно зменшує кількість шаблонного коду, який довелося б писати вручну. Якщо ж потрібні більш специфічні запити (наприклад, знайти всі освітні програми певного факультету з певним статусом), їх можна легко визначити за допомогою спеціальних імен методів у репозиторії або за допомогою анотації @Query.

Ядром серверної частини є сервісний шар (@Service компоненти, такі як EducationalProgramService, CurriculumService). Саме тут зосереджена основна бізнес-логіка системи. Сервіси виступають у ролі координаторів: вони отримують запити від вищих шарів (контролерів), використовують Репозиторії для доступу до даних, застосовують бізнес-правила (наприклад, встановлення статусу “У Розробці” для новоствореного навчального плану або статусу “Закрита” для нової освітньої програми), викликають механізми валідації даних

та перевірки прав доступу, а також керують транзакціями. Управління транзакціями (@Transactional) гарантує, що складні операції, які можуть змінювати декілька записів у різних таблицях (наприклад, збереження освітньої програми разом зі списком її переваг та вимог ЗНО), будуть виконані атомарно – або всі зміни успішно зберігаються, або, у разі будь-якої помилки, всі зміни скасовуються, підтримуючи таким чином цілісність бази даних *(див. Додаток В - Приклад методу сервісу з управлінням транзакцією)*. Сервіси також відповідають за реалізацію логіки пошуку та фільтрації даних, часто використовуючи Criteria API для побудови динамічних запитів на основі параметрів, отриманих від користувача.

Перед тим, як зберігати будь-які дані, система повинна переконатися в їх коректності та в тому, що користувач має право виконувати відповідну операцію. Для цього реалізовано механізми валідації даних та перевірки прав доступу. Валідація здійснюється як за допомогою стандартних анотацій Java Bean Validation безпосередньо в Entity або DTO класах (перевірка на порожні значення, довжину рядків, діапазони чисел, формат email тощо), так і за допомогою спеціальних класів-валідаторів (EducationalProgramValidation, CurriculumValidation). Ці валідатори містять складніші бізнес-правила, наприклад, перевірку належності кафедри до факультету чи унікальності навчального плану для ОП на певний рік. Перевірка прав доступу реалізована через компонент EducationalProgramPermissionValidator, який взаємодіє з системою безпеки та account-server для визначення ролей та дозволів поточного користувача. Перед виконанням потенційно небезпечних операцій (створення, редагування, видалення) сервісний шар викликає цей валідатор для підтвердження наявності у користувача необхідних прав (наприклад, PermissionsEnum.EDIT_EDUCATIONAL_PROGRAM).

Зовнішнім інтерфейсом серверної частини є REST API, реалізоване за допомогою контролерів (@RestController, наприклад, EducationProgramController). Контролери обробляють HTTP-запити, що надходять від клієнтської частини. Відповідно до принципів REST, кожен ресурс

(освітня програма, навчальний план) має свій набір ендпоінтів для виконання операцій CRUD. Структура API чітко визначена за допомогою OpenAPI Specification, що дозволило використовувати інструменти для автоматичної генерації частини коду контролерів та DTO/View об'єктів (EducationProgramView, CurriculumView тощо). Ці DTO/View об'єкти слугують для передачі даних між клієнтом та сервером, часто маючи структуру, адаптовану для конкретного запиту або відповіді. Контролери отримують дані в DTO/View з тіла запиту, передають їх сервісному шару для обробки, а результат, отриманий від сервісу (можливо, у вигляді Entity), перетворюють назад у DTO/View за допомогою Конвертерів (EducationalProgramConverter) перед відправкою відповіді клієнту. При оновленні даних використовуються Мерджери (EducationalProgramMerger) для перенесення змін з DTO/View в існуючий Entity об'єкт *(див. Додаток Г - Приклад роботи контролера та використання DTO)*.

Окрім базових операцій, серверна частина реалізує специфічну логіку, необхідну для ефективного управління освітніми програмами. Це, зокрема, стосується роботи з пов'язаними списками (ЗНО, Переваги, Гаранти). При збереженні освітньої програми сервісний шар (часто за допомогою Мерджера) реалізує логіку “delta update”: він аналізує список пов'язаних елементів, що надійшов від клієнта, порівнює його з поточним станом в базі даних і точно визначає, які елементи треба додати, які оновити, а які видалити. Це дозволяє мінімізувати кількість операцій з базою даних та забезпечити коректне оновлення всіх пов'язаних даних в рамках однієї транзакції. Також реалізована серверна логіка для функції копіювання навчального плану, яка передбачає створення нових записів для самого плану та для всіх його дисциплін з коректними посиланнями та статусами.

Отже, реалізація серверної частини охоплює всі необхідні аспекти: від моделювання та зберігання даних до реалізації складної бізнес-логіки, валідації, контролю доступу та надання стандартизованого API.

3.3. Реалізація інтерфейсу користувача (Frontend)

Ключовим елементом системи управління освітніми програмами є її користувацький інтерфейс – те, з чим безпосередньо взаємодіють співробітники університету. Головна мета при розробці інтерфейсу полягала в тому, щоб зробити процес управління освітніми програмами максимально простим, логічним та ефективним.

Для цього функціонал було розділено на декілька основних екранів, навігація між якими здійснюється через систему маршрутизації Angular *(див. Додаток Г - Схема навігації в модулі освітніх програм)*. Візуальне оформлення всіх елементів (таблиць, форм, кнопок) уніфіковано за допомогою стандартного для SmartUkma UI-kit (на базі Bootstrap), що забезпечує єдність зовнішнього вигляду та адаптивність до різних розмірів екранів *(див. Додаток Д - Загальний вигляд інтерфейсу модуля)*.

Екран списку освітніх програм слугує основною точкою входу. Тут користувач бачить таблицю з переліком існуючих програм, де відображається ключова інформація: назва, спеціальність, рівень, статус та факультет. Для зручної роботи з великою кількістю програм реалізовано пагінацію – можливість переглядати список посторінково. Щоб швидко знайти потрібну програму, користувач може скористатися текстовим пошуком за назвою або застосувати фільтри: обрати конкретний факультет, спеціальність (за допомогою поля з автодоповненням), рівень програми (бакалавр, магістр, PhD) або її статус (наприклад, “Відкрита”) з випадаючих списків. При зміні фільтрів список програм автоматично оновлюється. Зі списку можна перейти до перегляду детальної інформації про програму, її редагування, перегляду пов'язаних навчальних планів або видалення (якщо користувач має відповідні права доступу). Перед видаленням система обов'язково запитує підтвердження дії у користувача.

Екран детального перегляду освітньої програми надає вичерпну інформацію про обрану програму. Всі дані, від назви та описів до параметрів

вступу, контактних осіб, переваг, гарантів та списку навчальних планів, представлені у структурованому вигляді для легкого сприйняття. Текстові описи, що можуть містити форматування (списки, виділення тексту), відображаються коректно. Якщо до програми було завантажено файл з програмою вступних випробувань, користувач бачить посилання для його завантаження. З цієї сторінки можна повернутися до списку або перейти до редагування програми (за наявності прав) *(див. Додаток Е - Знімок екрану детального перегляду освітньої програми)*.

Найбільш комплексним є інтерфейс створення та редагування освітньої програми. Він реалізований у вигляді єдиної форми, яка використовується як для додавання нових, так і для зміни існуючих програм. При редагуванні форма автоматично заповнюється поточними даними програми. Для управління великою кількістю полів та забезпечення коректності даних використовується механізм реактивних форм Angular. Користувач заповнює стандартні текстові та числові поля, обирає значення зі списків для таких полів, як факультет, кафедра, спеціальність, контактні особи, мови, стипендії, тип та рівень програми. Ці списки завантажуються з відповідних довідників університету. Для введення детальних описів використовується зручний текстовий редактор з можливостями форматування. Передбачена можливість завантаження файлу програми вступних випробувань.

Особливістю форми є секції для управління пов'язаними даними: вимогами ЗНО/НМТ, перевагами програми та її гарантами. Користувач може динамічно додавати нові записи (наприклад, нову вимогу ЗНО), редагувати існуючі або видаляти непотрібні безпосередньо в процесі редагування основної програми. Система автоматично перевіряє коректність введених даних (наприклад, чи заповнені обов'язкові поля, чи відповідають числові значення заданим діапазонам, чи коректний формат URL) та інформує користувача про помилки. Після заповнення всіх необхідних даних користувач натискає кнопку “Зберегти”. Система збирає всю інформацію з форми та відправляє її на сервер для обробки

та збереження в базі даних *(див. Додаток Є - Знімок екрану форми редагування освітньої програми)*.

Окремий функціонал присвячений управлінню навчальними планами в контексті освітньої програми. Користувач може переглянути список всіх навчальних планів, створених для обраної ОП, бачити їх статус та рік вступу. Ключовою можливістю тут є копіювання навчального плану. Ця функція дозволяє суттєво зекономити час при підготовці документації на новий навчальний рік. Користувач обирає існуючий план як зразок, вказує цільовий навчальний рік у діалоговому вікні, і система автоматично створює повну копію цього плану (включаючи весь перелік дисциплін), але вже для нового року вступу, у статусі “Чернетка”. Після цього залишається лише внести необхідні корективи *(див. Додаток Ж - Інтерфейс управління навчальними планами та кнопка копіювання)*.

Таким чином, розроблений інтерфейс користувача надає повний набір інструментів для всебічного управління освітніми програмами – від створення та наповнення детальною інформацією до перегляду пов'язаних даних та використання допоміжних функцій, таких як копіювання навчальних планів.

3.4. Підсумки реалізації

Практична частина даної кваліфікаційної роботи була зосереджена на розробці та впровадженні функціонального програмного модуля для управління освітніми програмами в рамках інтегрованої інформаційної системи SmartUkma. У цьому розділі було детально описано процес реалізації як серверної, так і клієнтської частин системи, а також їхню взаємодію.

Створений програмний модуль є готовим до впровадження рішенням, що відповідає вимогам предметної області та сучасним стандартам розробки програмного забезпечення. Він надає співробітникам університету ефективний інструмент для управління освітніми програмами, сприяючи підвищенню точності даних, зменшенню трудомісткості рутинних операцій та покращенню доступності інформації про освітній процес.

ВИСНОВОК

У даній кваліфікаційній роботі було успішно вирішено завдання розробки та реалізації системи управління освітніми програмами. Метою роботи була автоматизація та оптимізація процесів, пов'язаних зі створенням, зберіганням, оновленням та представленням інформації про освітні програми університету.

На першому етапі роботи було проведено глибокий аналіз предметної області. Це включало визначення ключових понять, таких як освітня програма та навчальний план, детальний опис їх структури та необхідних атрибутів (від назви та рівня до вимог ЗНО, переваг та гарантів), а також аналіз існуючих бізнес-процесів управління ОП та їх недоліків, таких як трудомісткість, ризик помилок та фрагментація даних. На основі цього аналізу були сформульовані чіткі функціональні вимоги до майбутньої системи.

Другий розділ роботи був присвячений проєктуванню системи та вибору технологічних рішень. Було розглянуто мікросервісну архітектуру проєкту SmartUkma та визначено місце модуля управління ОП в рамках сервісу info-server. Було спроєктовано структуру бази даних для зберігання інформації про ОП та пов'язані сутності, а також визначено ключові ендпоінти REST API для взаємодії компонентів.

У третьому розділі було детально описано процес розробки та реалізації програмного модуля. Реалізовано серверну частину, включаючи моделі даних, сховища даних, сервісний шар з бізнес-логікою, механізми валідації та контролю доступу, а також API контролери. Розроблено клієнтську частину на Angular, що надає користувачам функціональний та зручний інтерфейс для перегляду списку програм з фільтрацією, детального перегляду ОП, а також для їх створення та редагування за допомогою комплексних реактивних форм.

Практичне значення розробленої системи полягає у підвищенні ефективності роботи підрозділів університету, відповідальних за розробку та супровід освітніх програм, покращенні якості та консистентності даних, а також забезпеченні

кращої доступності актуальної інформації про освітні програми для всіх учасників освітнього процесу – співробітників, студентів та абітурієнтів.

СПИСОК ЛІТЕРАТУРИ

1. Закон України «Про вищу освіту» № 1556-VII від 01.07.2014 р. [Електронний ресурс]. - Режим доступу: https://zakon.rada.gov.ua/laws/show/1556-18.

2. Освітня програма // Вікіпедія : вільна енциклопедія. [Електронний ресурс]. - Режим доступу: https://uk.wikipedia.org/wiki/Освітня_програма.

3. Навчальний план // Вікіпедія : вільна енциклопедія. [Електронний ресурс]. - Режим доступу: https://uk.wikipedia.org/wiki/Навчальний_план.

4. Spring Boot Reference Documentation [Електронний ресурс]. - Режим доступу: https://docs.spring.io/spring-boot/index.html.

5. Spring Data JPA - Reference Documentation [Електронний ресурс]. – Режим доступу: https://docs.spring.io/spring-data/jpa/reference/.

6. Angular Documentation [Електронний ресурс]. - Режим доступу: https://angular.io/.

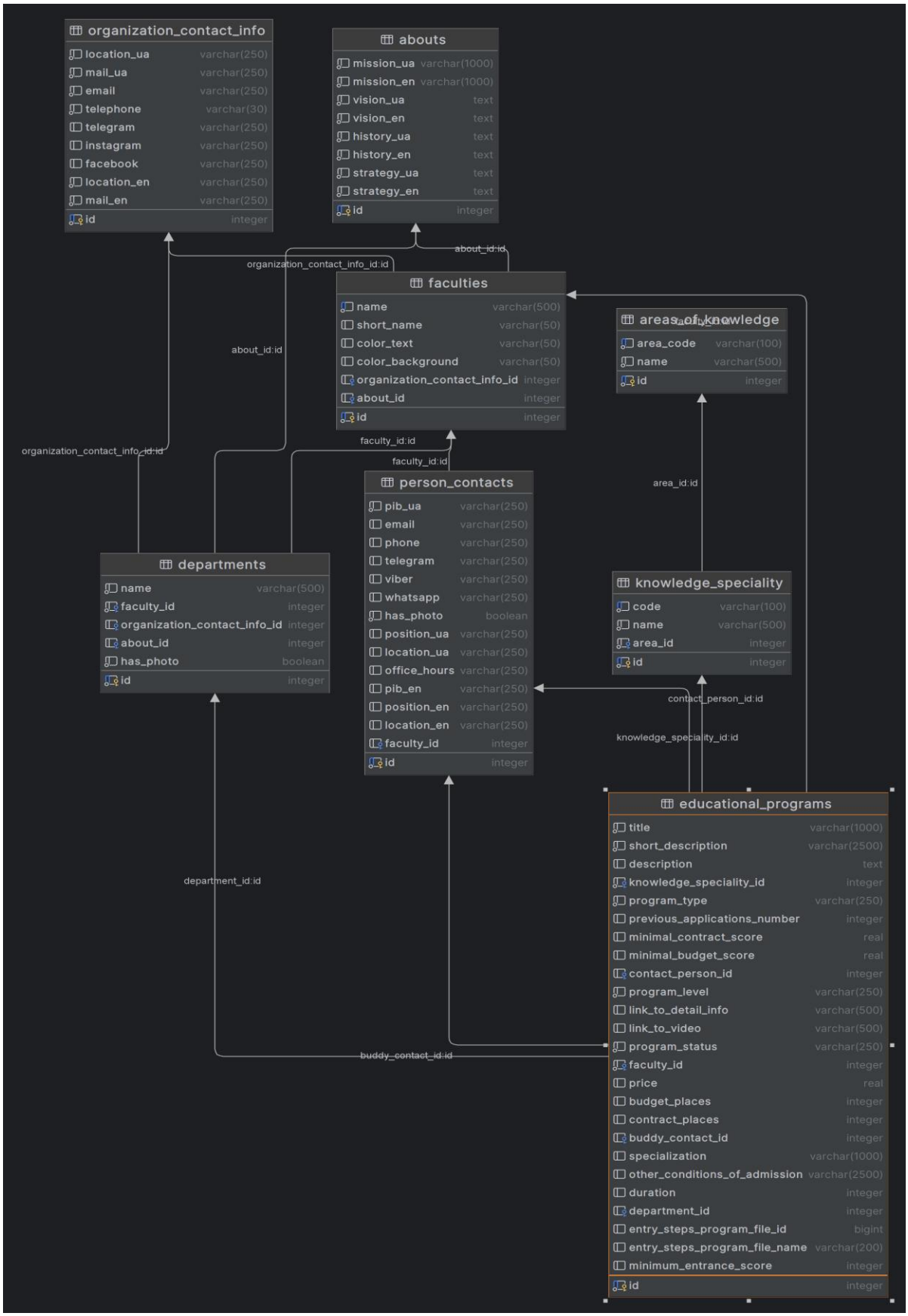
7. RxJS Overview [Електронний ресурс]. - Режим доступу: https://rxjs.dev/guide/overview.

8. PostgreSQL: Documentation [Електронний ресурс]. - Режим доступу: https://www.postgresql.org/docs/.

9. Docker Overview [Електронний ресурс]. - Режим доступу: https://docs.docker.com/get-started/overview/.

ДОДАТКИ

Додаток А. ER діаграма



Додаток Б. Приклад визначення класу EducationalProgramEntity

EducationalProgramEntity.java

```java
package ua.edu.ukma.fin.userinfo.domain.entity;

import ...



@SuppressWarnings("JpaDataSourceORMInspection")
@NoArgsConstructor
@AllArgsConstructor
@Builder
@Entity
@Table(name = "educational_programs")
@Getter
@Setter
public class EducationalProgramEntity implements Serializable, GettableById<Integer> {

    private static final long serialVersionUID = -2065460251799662179L;
    @Id
    @Column(name = "id")
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;
    @Column(name = "title_ua")
    @NotEmpty(message = "error.educational.program.title.empty")
    @Size(max = 1000, message = "error.educational.program.title.size")
    private String titleUa;
    @Column(name = "title_en")
    @NotEmpty(message = "error.educational.program.title.empty")
    @Size(max = 1000, message = "error.educational.program.title.size")
    private String titleEn;
    @Column(name = "short_description_ua")
    @NotEmpty(message = "error.educational.program.short.description.empty")
    @Size(max = 2500, message = "error.educational.program.short.description.size")
    private String shortDescriptionUa;
    @Column(name = "short_description_en")
    @NotEmpty(message = "error.educational.program.short.description.empty")
    @Size(max = 2500, message = "error.educational.program.short.description.size")
    private String shortDescriptionEn;
    @Column(name = "description_ua")
    private String descriptionUa;
    @Column(name = "description_en")
    private String descriptionEn;
    @ManyToOne(fetch = FetchType.EAGER)
    @JoinColumn(name = "knowledge_speciality_id")
```

Додаток В. Приклад методу сервісу з управлінням транзакцією

EducationalProgramService.java

```java
package ua.edu.ukma.fin.userinfo.services.educational.program;

import ...


@Service
@Slf4j
@Transactional(propagation = Propagation.REQUIRED, rollbackFor = BaseException.class)
public class EducationalProgramService extends BaseService<EducationalProgramEntity, EducationProgramView, Integer> {

    2 usages
    private final EducationalProgramRepository educationalProgramRepository;


    @Autowired
    public EducationalProgramService(EducationalProgramRepository educationalProgramRepository) {
        super(EducationalProgramEntity.class, EducationalProgramEntity::new);
        this.educationalProgramRepository = educationalProgramRepository;
    }


    @Override
    public boolean delete(Integer id) {
        EducationalProgramEntity educationalProgram = educationalProgramRepository.findById(id).orElseThrow(
            () -> new NoSuchEntityException(EducationalProgramEntity.class.getName(), "by id: " + id)
        );
        permissionValidator.validForDelete(educationalProgram);
        validationService.validForDelete(educationalProgram);
        repository.delete(educationalProgram);
        log.debug("We delete educational program and all Education ZNO Disciplines. {}", educationalProgram);
        return true;
    }


    public PageResult<Map<String, Object>> getPage(Collection<String> fields, String restrict, ProgramLevel level) {
        EducationalProgramCriteria criteria = parse(restrict);
        if (level != null) {
            criteria.setProgramLevel(level);
        }
        return new PageResult<>(getList(criteria, fields), count(criteria));
```

Додаток Г. Приклад роботи контролера та використання DTO

EducationProgramController.java

```java
package ua.edu.ukma.fin.userinfo.controllers.rest;

import ...


@Slf4j
@RestController
@RequiredArgsConstructor
public class EducationProgramController implements EducationProgramControllerApi {

    private final EducationalProgramService service;

    no usages
    @Override
    public ResponseEntity<CommonResponse.LongResponse> countEducationPrograms(String restrict) {
        return ResponseEntity.ok(CommonResponse.LongResponse.of(service.count(restrict)));
    }

    no usages
    @Override
    public ResponseEntity<CommonResponse.LongResponse> countEducationProgramsOpen(String restrict) {
        return ResponseEntity.ok(CommonResponse.LongResponse.of(service.count(restrict)));
    }


    @Override
    public ResponseEntity<CommonResponse.IntegerResponse> createEducationProgram(EducationProgramView body) {
        log.info("User {} want's to create education program {}", SecurityContextAccessor.getBehalfOnEmail(), body);
        return ResponseEntity.ok(CommonResponse.IntegerResponse.of(service.create(body)));
    }

    no usages
    @Override
    public ResponseEntity<CommonResponse.BooleanResponse> deleteEducationProgramById(Integer id) {
        log.info("User {} want's to delete education program {}", SecurityContextAccessor.getBehalfOnEmail(), id);
        return ResponseEntity.ok(CommonResponse.BooleanResponse.of(service.delete(id)));
    }

```

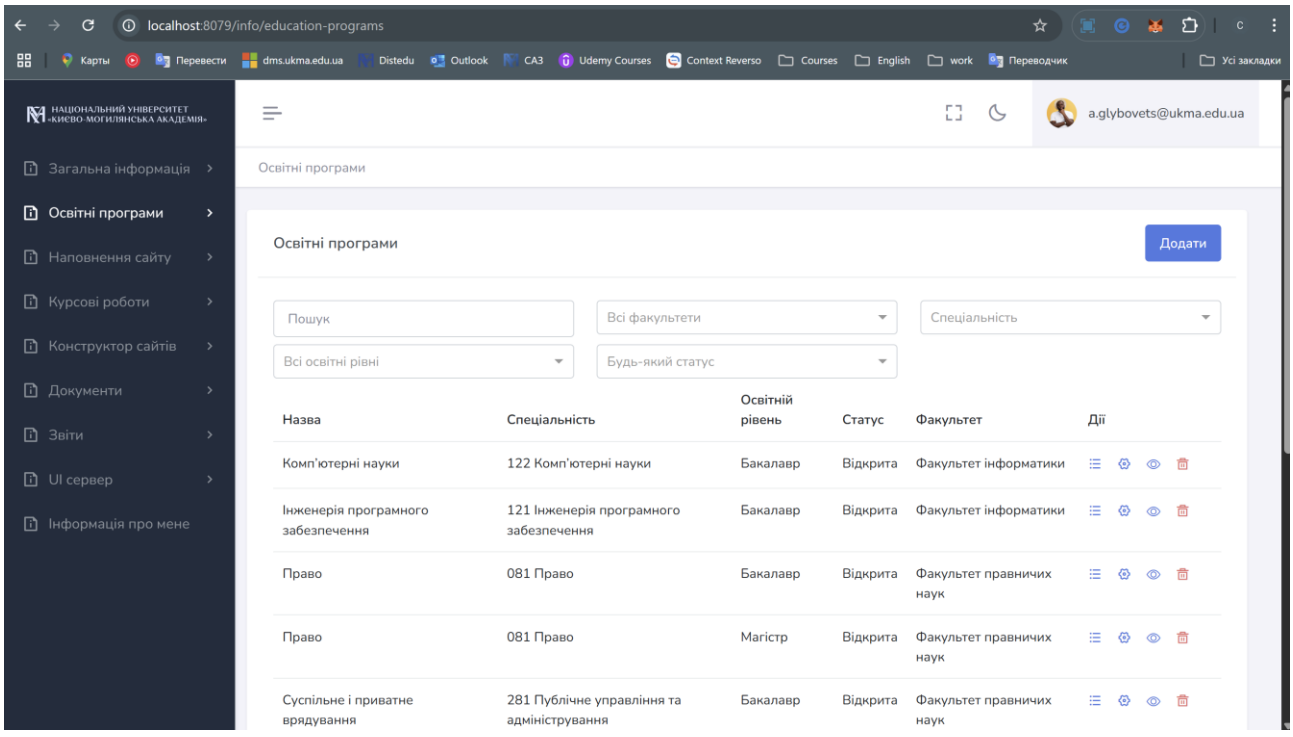
Додаток Ґ. Схема навігації в модулі освітніх програм

route.ts

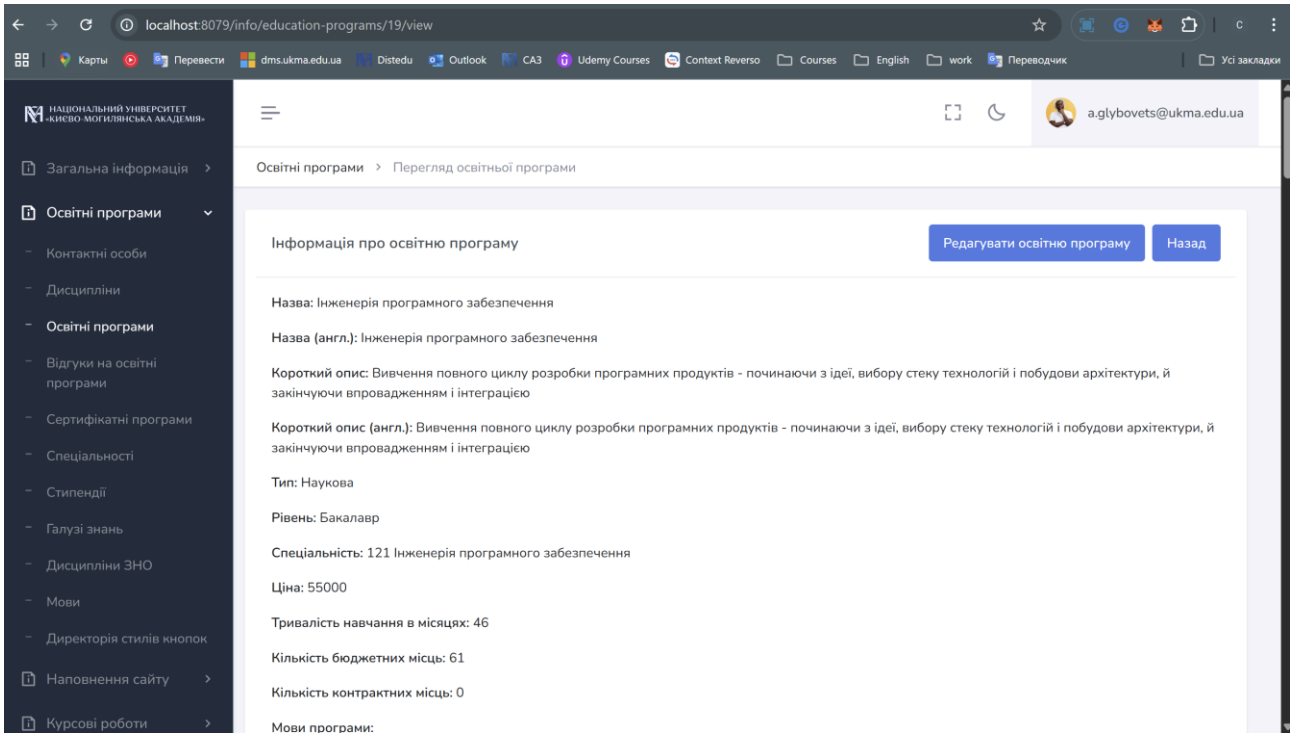
```ts
import {Routes} from "@angular/router";
import {hasInfoServerPermissionsGuard} from "@core/services/session/session-access.guard";
import {InfoServerPermission} from "@model/permissions/permissions";
import {withBreadcrumb} from "../../../layouts/breadcrumbs/utils/with-breadcrumb.util";
import {CreateEducationProgramStrategy, UpdateEducationProgramStrategy} from "./components/upsert/strategy";
import {BaseDisciplineListStrategy, CurriculumDisciplinesListStrategy} from "../disciplines/list/strategy";

export const ROUTES: Routes = [
  {
    path: "",
    pathMatch: "full",
    loadComponent: () => import('./components/list/education-program-list.component').then(m=>m.EducationProgramListComponent),
  },
  {
    path: ':id/view',
    loadComponent: () => import('./components/view/education-program-view.component').then(m=>m.EducationProgramViewComponent),
    data: withBreadcrumb( label: 'Перегляд освітньої програми'),
  },
  {
    path: ':id/curriculums',
    loadComponent: () => import('./components/education-program-curriculums/education-program-curriculums.component').then(m=>m.EducationProgramCurriculumsComponent),
    data: withBreadcrumb( label: 'Перегляд навчальних планів для освітньої програми'),
  },
  {
    path: "create",
    loadComponent: () => import('./components/upsert/education-program-upsert-page.component').then(m=>m.EducationProgramUpsertPageComponent),
    resolve: {strategy: () => new CreateEducationProgramStrategy()},
    canActivate: [hasInfoServerPermissionsGuard(InfoServerPermission.CREATE_EDUCATIONAL_PROGRAM)],
    data: withBreadcrumb( label: 'Створення освітньої програми'),
  },
  {
    path: ":id/edit",
    loadComponent: () => import('./components/upsert/education-program-upsert-page.component').then(m=>m.EducationProgramUpsertPageComponent),
    resolve: {strategy: (route) => new UpdateEducationProgramStrategy(Number(route.params['id']))},
    canActivate: [hasInfoServerPermissionsGuard(InfoServerPermission.EDIT_EDUCATIONAL_PROGRAM)],
    data: withBreadcrumb( label: 'Редагування освітньої програми'),
  }
]
```

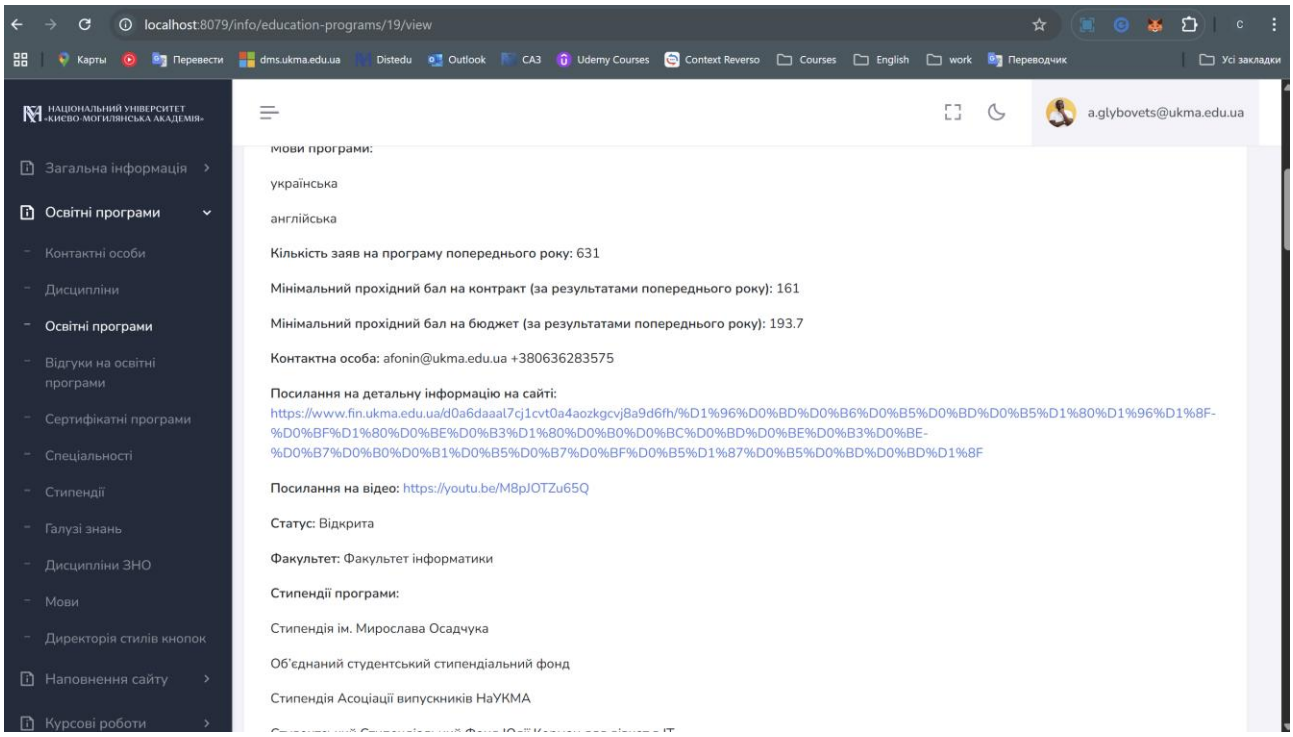
ROUTES

Додаток Д. Загальний вигляд інтерфейсу модуля

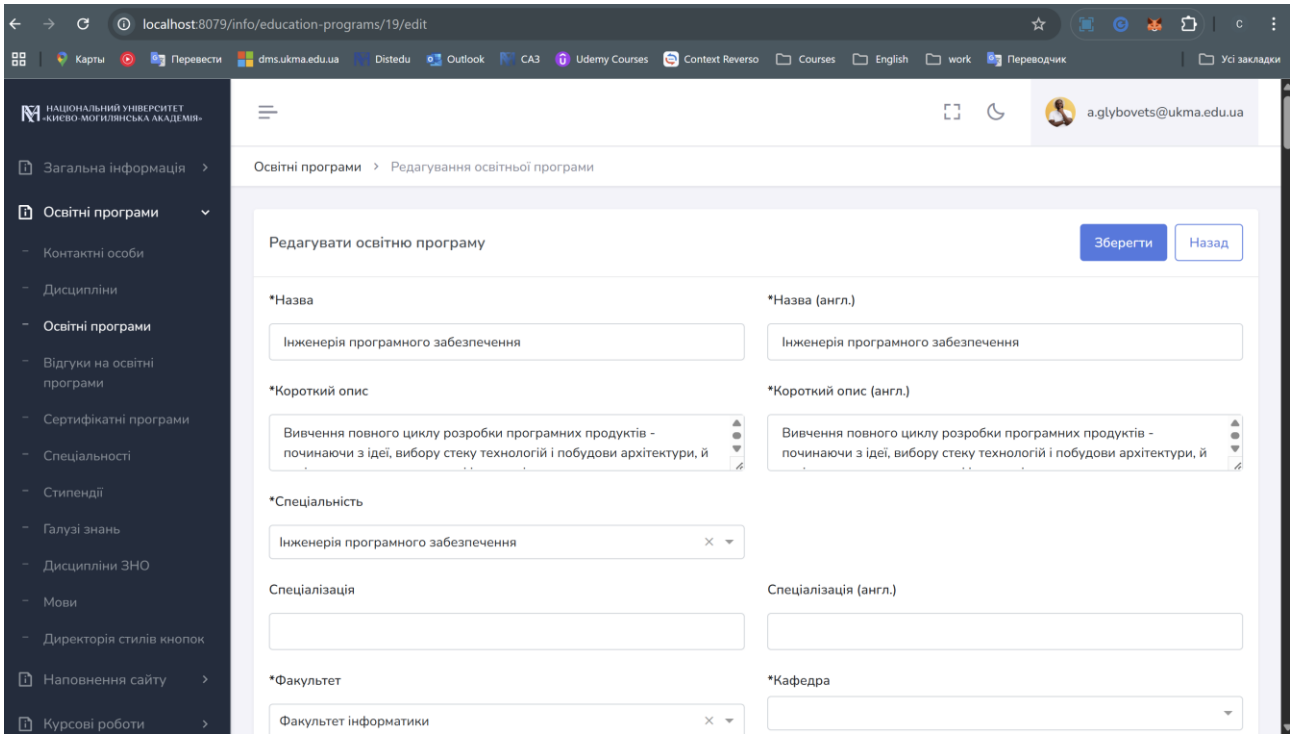


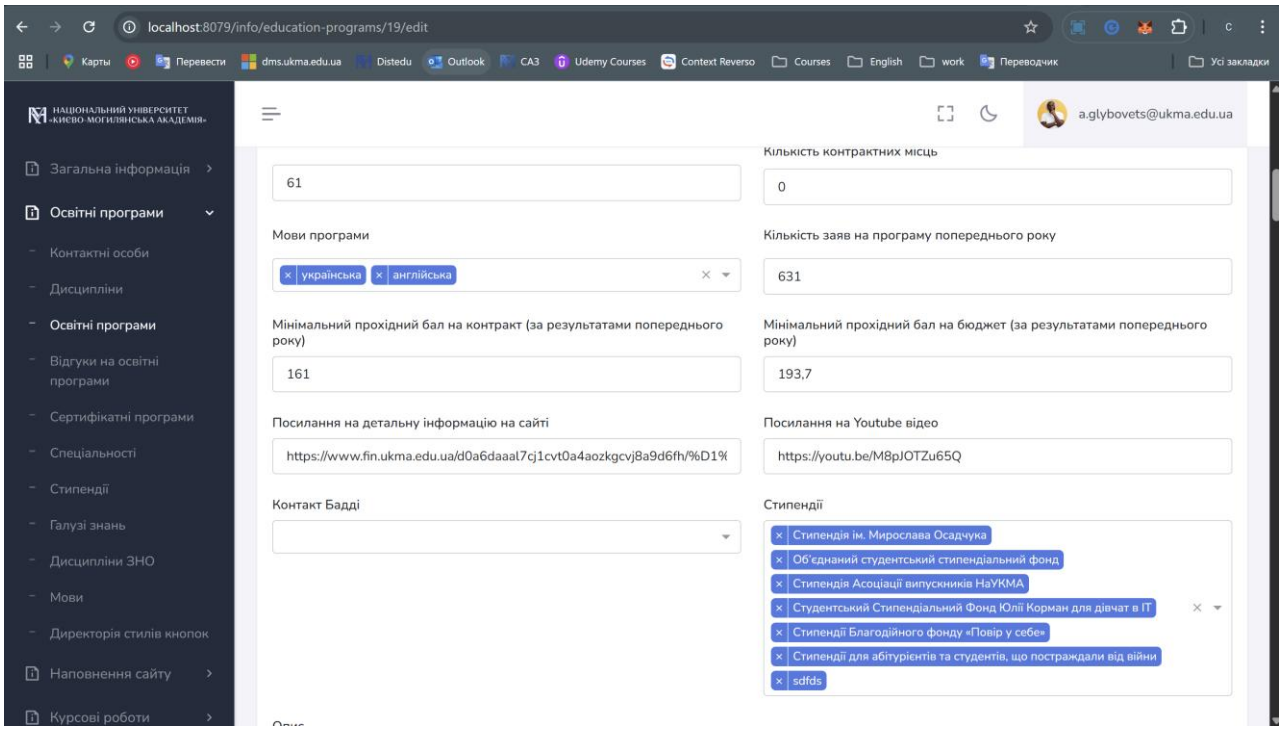
Додаток Е. Знімок екрану детального перегляду освітньої програми

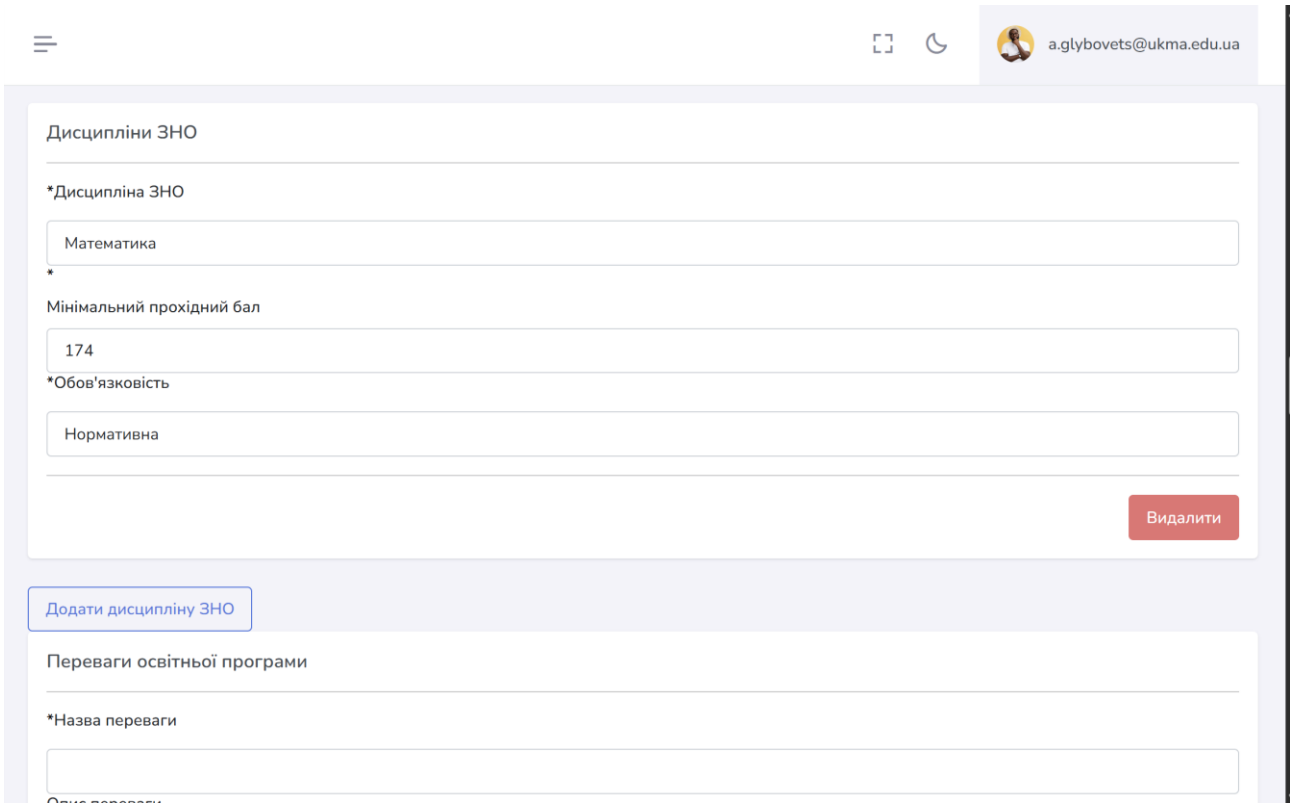


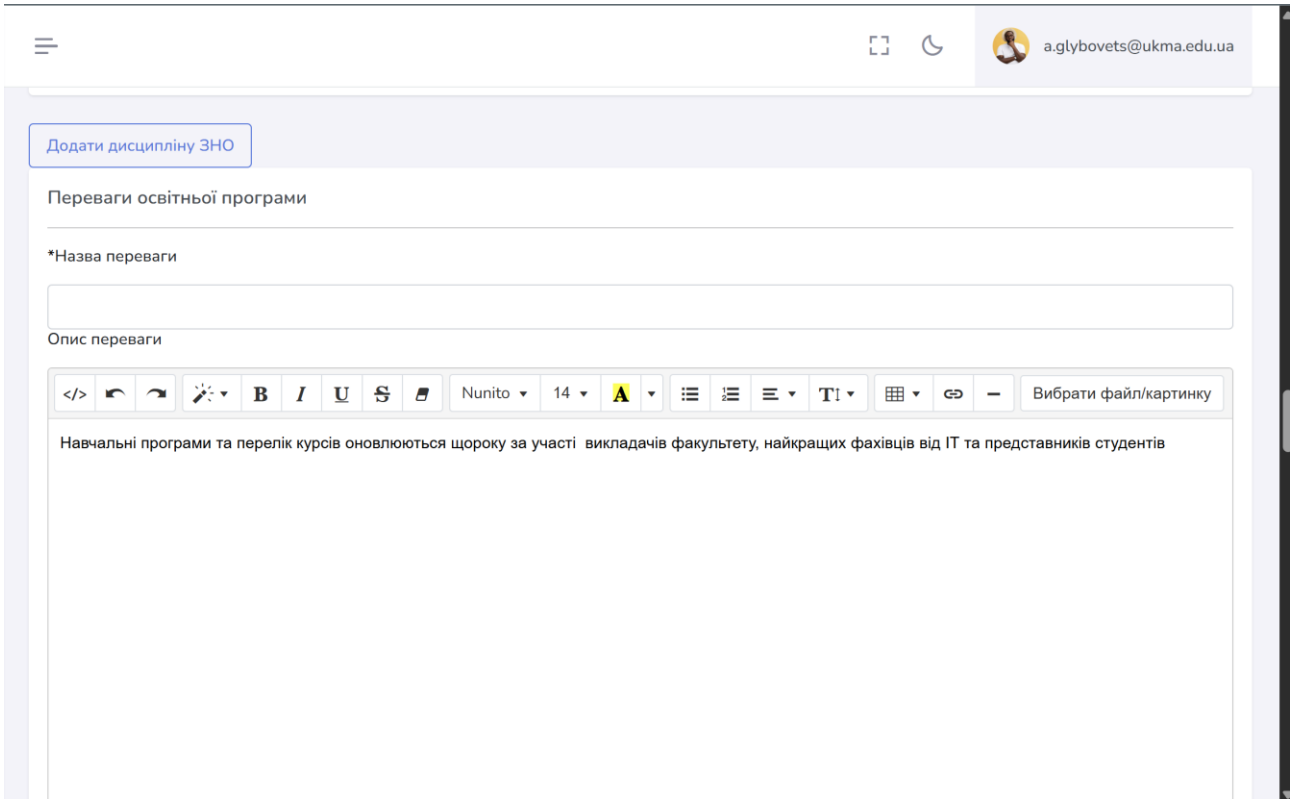


Додаток Є. Знімок екрану форми редагування освітньої програми






a.glybovets@ukma.edu.ua
Дисципліни ЗНО
*Дисципліна ЗНО
Математика
*
Мінімальний прохідний бал
174
*Обов'язковість
Нормативна
Видалити
Додати дисципліну ЗНО
Переваги освітньої програми
*Назва переваги


a.glybovets@ukma.edu.ua
Додати дисципліну ЗНО
Переваги освітньої програми
*Назва переваги
Опис переваги
Nunito
14
Вибрати файл/картинку
Навчальні програми та перелік курсів оновлюються щороку за участі викладачів факультету, найкращих фахівців від IT та представників студентів

Додаток Ж - Інтерфейс управління навчальними планами та кнопка копіювання