ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені І.І.МЕЧНИКОВА

(повне найменування вищого навчального закладу)

Факультет математики, фізики та інформаційних технологій

(повне найменування інституту, назва факультету (відділення))

Кафедра математичного забезпечення комп'ютерних систем

(повна назва кафедри (предметної, циклової комісії))

# Кваліфікаційна робота

на здобуття ступеня вищої освіти «бакалавр»

(освітньо-кваліфікаційний рівень)

на тему

Інформаційна система організаційного управління навчальним навантаженням. Підсистема навчального відділу.

Information system for organizational management of educational workload. The educational department subsystem.


Виконав: студент денної форми навчання

спеціальності 126 – Інформаційні системи та технології .

(шифр і назва напряму підготовки, спеціальності)

Освітня програма «Інформаційні системи та технології»

(назва освітньої програми)

Джигов Дмитро Юрійович

(прізвище, ім'я, по-батькові)

Керівник д.т.н., проф. Малахов. Є.В.

(науковий ступінь, вчене звання, прізвище та ініціали, **підпис**)

Рецензент Розновець О.І.

(науковий ступінь, вчене звання, прізвище та ініціали)


Рекомендовано до захисту:

Протокол засідання кафедри

№___ від «___»_____ 2023 р.

Завідувач кафедри

Подписано цифровой подписью: Eugene Malakhov
Дата: 2023.06.16 20:58:07 +03'00'

_Євгеній МАЛАХОВ_

(підпис) (ім'я, прізвище)

Захищено на засіданні ЕК № ___

протокол № __ від «_ _» ____ 2023 р.

Оцінка _____/_____/_____

(за національною шкалою, шкалою ECTS, бали)

Голова ЕК

_____ _Володимир ВИЧУЖАНІН_

(підпис) (ім'я, прізвище)

Одеса - 2023

# АНОТАЦІЯ

Мета роботи полягає в підвищенні ефективності організаційного управління процесом формування і контролю навчального навантаження викладачів університету, шляхом побудови підсистеми навчального відділу інформаційної системи ЗВО.

Користувачами даної системи є співробітники навчального відділу, завідувачі кафедри, гаранти та адміністратори ІС.

Проект виконано з використанням фреймворку Ruby on Rails для реалізації бекенд-серверу та фреймворку React для реалізації фронтенд-застосунку, СУБД PostgreSQL та ORM-бібліотеки ActiveRecord, яка надає можливість об’єктно-реляційного відображення компонентів SQL на компоненти мови Ruby. Архітектура системи відповідає шаблону MVC.

У розробленій підсистемі ІС реалізовано функціонал для організації освітнього процесу в ОНУ імені І.І. Мечникова.

Результатом кваліфікаційної роботи є підсистема навчального відділу ІС організаційного управління навчальним навантаженням, що має простий та інтуїтивний інтерфейс користувача.

# ABSTRACT

Objective: to ensure effective management of the processes of the university's educational department, reduce the time and number of errors in the formation of educational programs and curricula by building a subsystem of the IS of the university.

The users of this system are employees of the educational department, heads of departments, guarantors and IS administrators.

The project was implemented using the Ruby on Rails framework for the backend server and the React framework for the frontend application, the PostgreSQL database, and the ActiveRecord ORM library, which provides the ability to object-relationally map SQL components to Ruby components. The system architecture follows the MVC template.

The developed IS subsystem implements the functionality for organizing the educational process at the Odesa I. I. Mechnikov National University.

The result of the qualification work is the subsystem of the educational department of the IS of organizational management of the educational load, which has a simple and intuitive user interface.

**ЗМІСТ**

Стор.

# СПИСОК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ І ТЕРМІНІВ

ІКТ – інформаційно-комунікаційні технології.

JSON – JavaScript Object Notation.

ORM – Object-Relational Mapping.

SPA – Single-Page Application.

# ВСТУП

Сьогодні інформаційні технології досить швидко розвиваються та використовуються в усіх сферах життя. Особливо важливою стала їх роль у сфері освіти, де системи управління та обробки інформації допомагають ефективніше організовувати навчальний процес та забезпечувати якість навчання студентів [1].

Мета роботи полягає в підвищенні ефективності організаційного управління процесом формування і контролю навчального навантаження викладачів університету, шляхом побудови підсистеми навчального відділу інформаційної системи ЗВО.

Для цього необхідно провести аналіз існуючих систем управління навчальними закладами, визначити їх переваги та недоліки, і на основі цих даних розробити власну підсистему.

Об'єктом розробки є внутрішня структура ЗВО та організація навчального процесу в університетах, а також проблеми, що виникають при управлінні цими процесами та можливості їх вирішення за допомогою інформаційних технологій.

Сучасний стан системи організації управління структурою університетів України визначається високим рівнем децентралізації та відсутністю єдиної системи управління.

Кожен відділ, факультет чи кафедра університету має свої власні внутрішні правила та процедури, що часто не сумісні з політикою та вимогами інших відділів. Така система управління призводить до неефективного використання ресурсів, зменшення якості навчання та збільшення навантаження на персонал.

Недостатня автоматизація та використання інформаційних технологій є ще одним фактором, який ускладнює управління структурою університету. Багато процесів ЗВО досі здійснюються вручну, буквально «на папері» чи у кращому випадку в Excel, що потребує зайвого часу та зусиль від працівників.

Важливо відзначити, що у багатьох університетах відсутня єдина база даних, де б могли зберігатися та оброблятися інформаційні дані. При такій організації часто виникає необхідність запиту викладачами чи будь-якими іншими співробітниками різноманітної інформації від їнших структурних підрозділів, що вимагає додаткового часу обробки, при цьому ця інформація може бути надана неактуальною, оскільки, поки інформація надходила, вже дані могли оновитись. У разі, коли співробітники не можуть покладатись точно на надану інформацію, можуть прийматись неправильні рішення, що можуть негативно позначитися на функціонуванні університету.

Це лише декілька з прикладів комунікації всередині ЗВО за відсутності централізованої системи, але вони дуже добре демонструють потребу ЗВО у комплексній системі обліку та взаємодії з компонентами освітнього процесу.

Тому забезпечення ефективної та злагодженої системи організації управління університетом стає актуальною задачею, яка потребує впровадження нових технологій та методів управління.

Багато закладів вищої освіти в Україні використовують власні інформаційні програмні надбання для організації навчального процесу [2]. Серед таких ЗВО: Київський національний університет імені Т. Шевченка (система «Тритон»), Житомирський державний університет імені Івана Франка (система «Факультет»), Львівському національному університеті Івана Франка, Херсонський державний університет та багато інших.

Вказані ЗВО є прикладами того, як інформаційні системи / часткові програмні модулі можуть позитивно вплинути на навчальний процес та допомогти в управлінні внутрішньою структурою ЗВО, що однозначно зменшує кількість рутинних чи навіть непотрібних операцій.

Однією з головних проблем управління навчальним процесом є необхідність ефективної організації освітнього процесу та навантаження на викладачів. Це може бути дуже складним завданням, особливо у великих університетах, де кількість викладачів, дисциплін та студентських контингентів може досягати значних розмірів.

Для досягнення поставленої мети необхідно вирішити наступний список завдань:

1) проаналізувати предметну область, що охоплює навчальний відділ;

2) сформулювати вимоги предметної області підсистеми до створюваної підсистеми ІС;

3) виконати проектування інформаційної моделі предметної області з врахуванням вимог до підсистеми ІС;

4) розробити необхідні компоненти бізнес-логіки підсистеми ІС;

5) розробити та реалізувати web-застосунок для інтерактивної взаємодії користувача з створеною підсистемою ІС;

6) створити тестовий набір даних для перевірки коректності виконання бізнес-правил системи;

7) провести тестування розробленої системи на відповідність встановленим вимогам.

# 1 ПОСТАНОВКА ЗАДАЧІ СТВОРЕННЯ ПІДСИСТЕМИ НАВЧАЛЬНОГО ВІДДІЛУ ІНФОРМАЦІЙНОЇ СИСТЕМИ ЗВО

## 1.1 Введення в предметну область

Навчально-методичний відділ є однією з ключових структурних одиниць в університеті, що забезпечує організацію та координацію навчально-методичної роботи відповідно до вимог законодавства та стандартів вищої освіти. Його діяльність спрямована на забезпечення високого рівня навчально-методичної роботи, що відповідає вимогам сучасної освіти та забезпечує ефективність навчального процесу університету.

Розуміння того як працює ЗВО, яка його структура та які задачі ставить перед собою навчально-методичний відділ, дають можливість оцінити спектр задач, що можуть бути реалізовані у системі в кінцевому приближенні.

Основні завдання навчально-методичного відділу полягають у забезпеченні якісної підготовки фахівців за різними спеціальностями, формуванні та реалізації освітніх програм та навчальних планів, контролю за дотриманням вимог щодо організації навчального процесу [3].

*Загальні задачі*, що виконує навчально-методичний відділ у ЗВО України, можуть включати:

1) організація розробки навчальних, робочих навчальних планів напрямів/спеціальностей, контроль за їх виконанням;

2) організація формування графіку навчального процесу університету на навчальний рік;

3) організація складання графіків та розкладів навчальних занять, розкладів заліково-екзаменаційних сесій, контроль за їх виконанням;

4) формування та контроль нормативів розрахунку індивідуального навантаження викладачів;

5) організація планування та перевірка обсягів навчального навантаження кафедр, розподілу навантаження між викладачами;

6) контроль виконання навчального навантаження кафедр, індивідуальних планів викладачів;

7) формування та подання статистичної звітності з усіх видів навчальної роботи;

Навчально-методичний відділ у ЗВО виконує ряд задач з організаційної роботи щодо ведення обліку *спеціальностей* та *освітніх програм*, зокрема:

1) розробка та затвердження переліку спеціальностей, що викладаються в університеті, з урахуванням потреб ринку праці та актуальних тенденцій розвитку галузей;

2) організація та координація роботи зі зміною, погодженням та затвердженням навчальних планів та програм з урахуванням змін у вимогах до викладання та навчання, а також внесення змін до них;

3) проведення моніторингу та аналізу якості навчальних програм та планів з метою їх постійного вдосконалення;

4) ведення обліку та статистичного аналізу даних про кількість студентів на кожній спеціальності та їх успішність, що дозволяє оцінити ефективність навчання та приймати рішення щодо його вдосконалення;

5) підготовка документів для акредитації навчальних програм та спеціальностей, що викладаються в університеті, згідно з вимогами державних регуляторних органів;

6) організація та забезпечення роботи відповідних комісій та груп, які відповідають за вдосконалення навчальних програм та планів, викладання певних дисциплін та виконання інших завдань з підвищення якості навчання.

Також навчально-методичний відділ виконує ряд задач з організаційної роботи пов'язаної зі *студентами*. До найбільш важливих можна віднести наступні задачі:

1) забезпечення організації процедури прийому студентів на навчання, реєстрації нових та переведення старих студентів, а також здійснення контролю за дотриманням вимог документації та інших правил, пов'язаних з прийомом студентів;

2) забезпечення організації та контроль за проведенням навчальних занять, відповідно до навчальних планів та програм, а також здійснення контролю за виконанням вимог до організації навчального процесу згідно зі стандартами вищої освіти;

3) забезпечення контролю за навчальними досягненнями студентів, організовуючи систему контролю за знаннями та вміннями, проведенням контрольних та іспитових робіт, складанням та зберіганням академічних рейтингів та іншої документації;

4) забезпечення організації соціальної підтримки студентів, в тому числі наданням інформації про стипендії, гранти, соціальні послуги та допомогу в рішенні соціальних проблем;

5) забезпечення організації та проведення додаткової освіти для студентів, таких як різноманітні курси, семінари, тренінги та інші форми навчання, що дозволяють збільшити професійні навички студентів;

6) забезпечення організації та проведення науково-методичної роботи, спрямованої на підвищення якості навчального процесу, включаючи проведення досліджень та наукових конференцій;

7) забезпечення координації міжнародної діяльності в університеті, зокрема, організацію міжнародної співпраці з іншими університетами, обмін студентами та викладачами.

Серед перелічених задач навчального відділу у рамках головного завдання проекту підсистема має надавати можливості:

1) формування навчального навантаження на підставі модулів створення навчальних планів та освітніх програм;

2) реєстрації доступних спеціальностей ЗВО та їх відповідностей щодо реєстру українських та міжнародних галузей знань;

3) формування нормативів розрахунку навантаження.

У рамках додаткового завдання забезпечення підсистеми кафедри актуальними даними викладацького складу та штатного розкладу підсистема навчального відділу має включати можливість ведення обліку та контролю за даними структурними компонентами.

## 1.2 Огляд систем управління освітнім процесом

### 1.2.1 Проблеми організації освітнього процесу українських ЗВО

Сьогодні кількість якісних автоматизованих систем управління навчальним процесом для ЗВО України не є дуже великою. Але все ж існують як незалежні софтверні рішення, так і деякі власні рішення українських ЗВО.

Однак, зазначимо, що деякі з цих систем можуть бути не такими простими в інтеграції, що відображається у надмірній універсальності систем, чи наявності значної кількості шаблонних задач, що прив’язані до певних стандартів, через що у ЗВО, що використовують специфічний процес управління навчальним процесом, відмінний від наданого рішення, мають або підлаштовувати програмні застосунки під власний технічний процес, або писати власні рішення для розв'язання поставлених задач.

Мета огляду – проаналізувати наявні програмні аналоги та визначити, які з них найбільш ефективні та придатні для використання в ОНУ імені І. І. Мечникова та як власна розробка зможе виконати поставлені задачі краще за існуючі аналоги.

На жаль, велика кількість ЗВО України відрізняються певним рівнем неструктурованості та безконтрольності організації навчального процесу.

До неструктурованості можна віднести відсутність єдиної бази даних, що накопичується ЗВО впродовж багатьох років, як наприклад, навчальні плани за останнє десятиліття, чи контингенти студентського складу за позаминулий рік тощо.

Коли вся інформація зберігається «на папері», чи у вигляді локальних файлів на комп’ютері кафедри, з’являється висока вірогідність втрати цінних та важливих даних, що можуть знадобитись у майбутньому в найбільш необхідний час, тому до питання організації та структуризації додається питання доступності інформації.

Щоб інтенсифікувати роботу ЗВО, потрібно застосовувати інформаційні технології та створювати автоматизовані системи, які допоможуть ефективно збирати, обробляти та використовувати інформацію в процесі управління. Застосування ІКТ в системі управління вищою освітою дозволить підвищити оперативність та якість прийняття рішень.

Впровадження систем автоматизації управління ЗВО вимагає широкого спектру завдань, які варіюються від формалізації процедур збору та зберігання інформації до перерозподілу обов’язків та змін в організаційній структурі управління. Виконання проектів автоматизації має значний вплив на ефективність функціонування ЗВО, тому детальне планування технічних, організаційних та людських аспектів є необхідним для досягнення успіху.

Хоча освітяни вже звикли до використання комп'ютерних програм, які допомагають створювати розклад навчальних занять, розподіляти аудиторії та розраховувати навантаження викладачів, ефективність кожної з цих розробок є недостатньою. Це сталося через відсутність єдиного системного підходу до управління навчальним закладом, тому не дивно, що багато ЗВО в Україні намагаються самостійно розв'язати проблему автоматизації управління навчальним процесом.

### 1.2.2 Приклади аналогів незалежних розробників

Чимало українських незалежних компаній пропонують програмні рішення для розв’язання задач організації освітнього процесу.

Деякі з них пропонують власний фунціонал у вигляді окремих програмних пакетів чи інформаційних систем зі повноцінною архітектурою.

До найбільш успішних та популярних можна віднести наступні приклади компаній та їх розробок:

1) пакет програм «Деканат», розроблена компанією Політек-СОФТ.
2) АСУ ЗВО «Вищий навчальний заклад», розроблена НДІ ПІТ;
3) АСУ ЗВО «Університет», розроблена ТОВ «UNITEX+».

Вказані програмні продукти розв’язують схожі задачі управління навчальним процесом та містять модулі, що допомагають розв’язати задачі навчально-методичного відділу ЗВО.

Серед вказаних продуктів *найбільш популярним* є пакет «Деканат» компанії Політек-СОФТ [4].

Програмний пакет «Деканат» містить кілька програм, що розв’язують різноманітні задачі користувача: «Навчальний процес», «Навчальний план», «Тарифікація», «Викладач» та інші. Серед них навчальному відділу знадобляться тільки «Навчальний процес» та «Навчальний план».

Дані програми надають можливості створення навчальних планів, генерацію звітних документів для навчальних та робочих планів, створення множини занять з урахуванням відповідних норм навантаження та наявних способів проведення занять, аналіз навчального процесу з можливістю перегляду властивостей елементів, фільтрації даних та генерації відповідних звітних документів, порівняння виконання навантаження викладачами з запланованим та генерація відповідних звітних документів;

Пакет «Деканат» можна придбати у різних комплектаціях, що робить його привабливішим вибором на фоні інших ІС. Пакет постачається у декількох версіях, залежно від наданих програм, що містяться у конкретному варіанті придбання. Програми, що вирішують задачі навчального відділу, надаються у рамках майже у всіх версіях пакетів. Оскільки плата за ПЗ виконується в залежності від кількості користувачів на програмний продукт, то майже немає різниці який обирати.

На прикладі даного пакету наочно видно, що придбання готового ПЗ вимагає високого рівня обізнаності в організації процесів та чіткій постановці задач, які має виконувати система.

*Наступний продукт* – АСУ «Вищий навчальний заклад» («ВНЗ»), розроблена у ПрАТ НДІ ПІТ [5]. Дана система надає декілька програмних модулів, серед яких: «Приймальна комісія», «Деканат» та «Студмістечко». Функціональні задачі навчального відділу виконує модуль «Деканат».

Модуль надає функціональні можливості організації навчального процесу (розробка навчальних та робочих планів, закріплення контингенту за робочими планами, формування та розподіл навантаження), роботи зі співробітниками (організація штатного розкладу) та роботи зі студентами (формування контингентів – імпорт з модулю «Приймальна комісія» чи ЄДЕБО, автоматизована генерація індивідуальних навчальних планів на базі загальних навчальних планів, аналіз навчальних досягнень, тощо).

Система має різнопланову направленість та охоплюваність виконуваних задач, тому враховуючи існуючі задачі навчального відділу, інтеграція настільки великої системи може бути занадто потужним рішенням. Придбання даної системи мало б сенс лише при нарощувані потреб ЗВО чи неможливості реалізації поставлених задач власними силами.

*Останній обраний продукт* – АСУ ЗВО «Університет», розроблена компанією «UNITEX+» [6]

Даний продукт інтегровано в ряд українських ЗВО, список яких нажаль є приватним. Дана система також складається з модульних компонентів, деякі з яких знаходяться в стані розробки. Система «Університет» на даний момент складається із наступних завершених автономних модулів: структура ЗВО, WEB-сайт, навчальна частина, кафедра, навчальний розклад та абітурієнт;

Для виконання задач стане у нагоді модуль «Структура ЗВО», як фундамент для обліку структурних компонентів ЗВО, а для виконання задач навчально-методичного відділу – автономний модуль «Навчальна частина».

Модуль «Структура ЗВО» є основним модулем для АСУ «Університет». Через цей модуль формується інформація про структуру ЗВО, кадрове забезпечення, керівний склад тощо.

Модуль «Навчальна частина» забезпечує автоматизацію різних процесів, пов’язаних з навчальною частиною ЗВО, які дозволяють: організовувати та формувати навчальні плани, вести облік та закріплювати навчальні дисципліни за кафедрами, розподіляти навчальні дисципліни об'єднаних академічних потоків із різних спеціальностей, вести облік та

контроль руху студентського контингенту у ЗВО, контролювати професорсько-викладацький складу, нормувати навчальний час викладачів в залежності від освітньо-кваліфікаційного рівня студентів, форми навчання та типу навчального навантаження.

Нажаль, про формат надання послуг (чи постачається повна система, що постійно оновлюється, чи постачаються певні автоматизовані модулі) та вартість системи відсутні на сайті самого продукту та в мережі інтернет.

Дана система реалізує чималий набір функціональних можливостей та найбільше за все відповідає вимогам щодо реалізації поставленим задачам навчального відділу та додаткової задачі з обліку структурних компонентів ОНУ імені І. І. Мечникова.

### 1.2.3 Приклади власних аналогів ЗВО

Часто існуючі системи та програмні пакети недостатньо повно виконують поставлені задачі, чи виконують ці задачі іншим шляхом, що не відповідає специфіці обраного ЗВО.

У такому разі ЗВО має або замовляти програмні додатки з урахуванням специфіки організаційних процесів ЗВО, або самим розробляти програмні модулі, що будуть виконувати саме те, що потребують конкретні задачі ЗВО.

Серед ЗВО, що використовують власні розробки, можна виділити:

1) АСУ ЗВО «Тритон» Київського національного університету імені Тараса Шевченка;

2) інформаційно-аналітична система управління ЗВО «Університет» Херсонського державного університету;

3) засоби автоматизації управління навчальним закладом в НУ «Львівська політехніка»;

4) автоматизована інформаційна система «Електронний університет», створена у Хмельницькому національному університеті;

5) система автоматизації управління навчальним процесом, розроблена й введена в експлуатацію у Львівському інституті банківської справи.

Для систем такого типу не характерна публікація документації власних програмних рішень, тому для дослідження цих систем потрібно виконувати офіційний запит до ЗВО. Нажаль, спроба отримати документацію хоча б однієї власної системи чи застосунку зазнала невдачі.

Як приклад інтеграції власних рішень можна навести власну розробку Львівського інституту банківської справи під назвою «Факультет».

У процесі використання пакету програм «Деканат», виявилися проблеми з однією з програм – «ПС-Студент" пакету Політек-СОФТ. Взаємодія з продуктами «Деканату» здійснюється через веб-додатки, які підключаються до централізованої бази даних. Однак, інститут зрозумів, що підхід до централізації даних не є необхідним, оскільки основний рівень, де збирається інформація про стан навчального процесу, є рівнем деканатів. Власна розробка інституту, модуль «Факультет», пропонує більш оптимальні та адаптовані під конкретні потреби рішення, розв'язуючи проблеми, що стояли перед ними.

Працівники навчально-методичного відділу в процесі експлуатації пакету «Деканат» помітили деякі недоліки та незручності, які не відповідали потребам інституту:

1) невідповідний механізм роботи з контингентами студентів заочної форми навчання;

2) некоректна форма роздруку інформації про контингент студентів, наявних у базі;

3) необхідність ручної зміни форм роздруку різноманітних довідок, відомостей та талонів;

4) відсутність автоматизованого заповнення даних для задач обліку.

Ці недоліки спонукали інститут розробити власне рішення, яке забезпечувало більш точну відповідність їх потребам. Власна розробка «Факультет» стала повноцінною системою автоматизації управління областю навчального процесу ЗВО.

Цей приклад ілюструє, що існуючі програмні рішення не завжди вирішують усі потреби організації. У таких випадках може виникати необхідність в розробці власних рішень, які краще відповідають конкретним вимогам та потребам організації. Важливо зрозуміти, що існує багато шляхів досягнення мети, і не завжди єдине рішення відповідає всім потребам.

### 1.2.4 Висновок за аналогами

Огляд аналогів незалежних розробників програмних рішень показав, що автоматизація освітнього процесу – ніша, яку займає чимала кількість компаній, але обрати серед них відповідний продукт дуже важко.

Нажаль, запропоновані приклади аналогів або надають занадто узагальнене або занадто потужне рішення в рамках задачі автоматизації та розв’язання проблем навчального відділу.

Загальні характеристики розглянутих систем управління вказано в таблиці 1.1. Відповідність поставлених задач до програмних рішень, що надають можливість їх вирішення, зображено в таблиці 1.2.

У випадку, коли незалежні розробки не можуть задовольнити потреб ЗВО, маємо можливість реалізувати необхідний функціонал власними силами, спроектувати та реалізувати підсистему навчально-методичного відділу та інтегрувати її в загальну інформаційну систему ЗВО.

Аналогічний досвід інших ЗВО показує, що створення власного програмного рішення за випадку відсутності альтернативних варіантів – не найгірший варіант, оскільки у такому разі не потрібно створювати різноманітні адаптери, виконувати зайву роботи з підгонки існуючого програмного рішення під технологічний процес ЗВО, а одразу спроектувати програмне рішення відповідно до поставлених задач.

Таблиця 1.1 – Матриця аналогів. Загальна характеристика

| # | | Пакет «Деканат» (Політек-СОФТ) | АСУ «Деканат» (НДІ ПІТ) | АСУ «Університет» (Unitex+) | Власні розробки ЗВО |
|---|---|---|---|---|---|
| **Загальні характеристики** | | | | | |
| 1 | Частота зміни платформи | Рідко | Рідко | Середня (активна розробка нових модулів) | Середня (Нові модулі розроблюються залежно від потреб ЗВО) |
| 2 | Гнучкість вибору функціоналу ПЗ | Можна замовити будь-який набір функціоналу за вказаними підписками | Існує можливість обрати тільки певні програмні модулі. | Немає інформації | Надається тільки необхідний функціонал |
| 3 | Сучасність інтерфейсу | Застарілий табличний інтерфейс | Сучасний табличний інтерфейс | Застарілий табличний інтерфейс | Залежить від реалізації конкретним ЗВО |
| 4 | Ціна (підкл./чол.) | ~2,5 тис. грн. | ~5 тис. грн | Інформація публічно не надана. | Безкоштовно, кількість доступних підключень залежить від потужностей ЗВО |
| 5 | Ступінь інтеграції | 375+ ЗВО по усій країні | 70+ ЗВО по усій країні | Інформація публічно не надана. Остання архівна згадка налічує ~20 ЗВО. | Реалізовано та інтегровано в рамках конкретних ЗВО |

Таблиця 1.2 – Матриця аналогів. Виконання задач підсистеми навчального відділу

| **Задача** | **Пакет «Деканат» (Політек-СОФТ)** | **АСУ «Деканат» (НДІ ПІТ)** | **АСУ «Університет» (Unitex+)** | **Власні розробки ЗВО** |
|---|---|---|---|---|
| **Поставлені задачі** | | | | |
| Облік структурних компонентів ЗВО | Так | Так | Так | Залежить від обраного ЗВО.<br><br>При цьому використовується *гібридний підхід,* при якому використовують *придбану систему*, та інтегрують у навчальний процес *власні розробки*, що відповідають особливостям організації роботи ЗВО. |
| Облік спеціальностей | Так | Так | Так | |
| Керування освітніми програмами | Так | Так | Так | |
| Створення та редакція навчальних планів | Так | Так | Так | |
| Керування нормативами для розрахунку навантажень | Частково | Ні | Так | |
| Керування графіками освітнього процесу | Так | Так | Так | |
| Облік викладацького складу та штатного розкладу | Залежно від типу підписки | Ні | Так | |

## 1.3 Постановка задачі

Предметна область, що розглядається в цій роботі – управління навчальним процесом в ОНУ імені І. І. Мечникова.

При аналізі предметної області сформульовані основні задачі навчально-методичного відділу які має вирішувати дана підсистема ІС:

1) облік структурних компонентів ЗВО (списки факультетів, кафедр, викладачів, штатного розкладу);
2) облік та керування спеціальностями та галузями знань;
3) облік та керування нормативами навантаження;
4) облік та керування графіками тривалості семестрів;
5) облік та керування освітніми програмами та навчальними планами;
6) надавати інтерактивний WEB-інтерфейс для взаємодії користувача з підсистемою ІС ЗВО.

В результаті аналізу та детального огляду поставлених задач, виявлено 4 основні групи користувачів:

1) співробітники навчального відділу – мають право створювати нові спеціальності, встановлювати нормативи навантажень та планувати графік тривалості семестрів;
2) гаранти спеціальності – мають право створювати нові освітні програми, планувати на них дисципліни та формувати на основі створених ОП навчальні плани;
3) завідувачі кафедр – мають право назначати нових викладачів на посади в межах своєї кафедри та формувати штатний розклад;
4) адміністратори ІС – мають право виконувати усі операції з керування ресурсами в рамках підсистеми та розпоряджатись делегацією прав доступу.

Усі задачі користувачів перелічені в Додатку А.

# 2 ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

## 2.1 Архітектура застосунку

Для організації взаємодії між компонентами використано патерн MVC (Model-View-Controller) (див. рис. 2.1). Він допомагає реалізувати взаємодію , яка відокремлює логіку, дані та представлення, щоб забезпечити більшу модульність та розширюваність проекту.

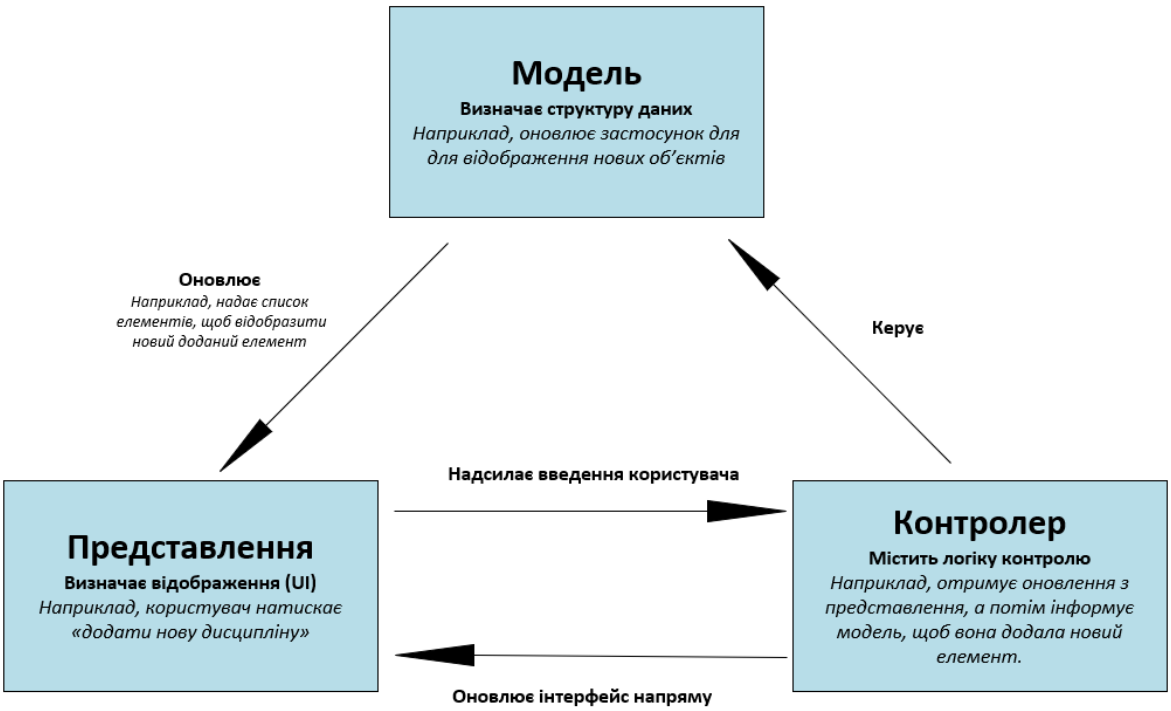


Рисунок 2.1 – Патерн MVC

При реалізації проекту використано трирівневу архітектуру клієнт-сервер (див. рис. 2.2).

Трирівнева архітектура під собою має на увазі, що проект складається з трьох основних компонентів, які взаємодіють між собою. Ці компоненти включають в себе клієнтську частину, серверну частину та базу даних.

Клієнтська частина відповідає за відображення інтерфейсу користувача та обробку його дій. Вона представлена уявленням про те, як користувачі бачать та взаємодіють з проектом.

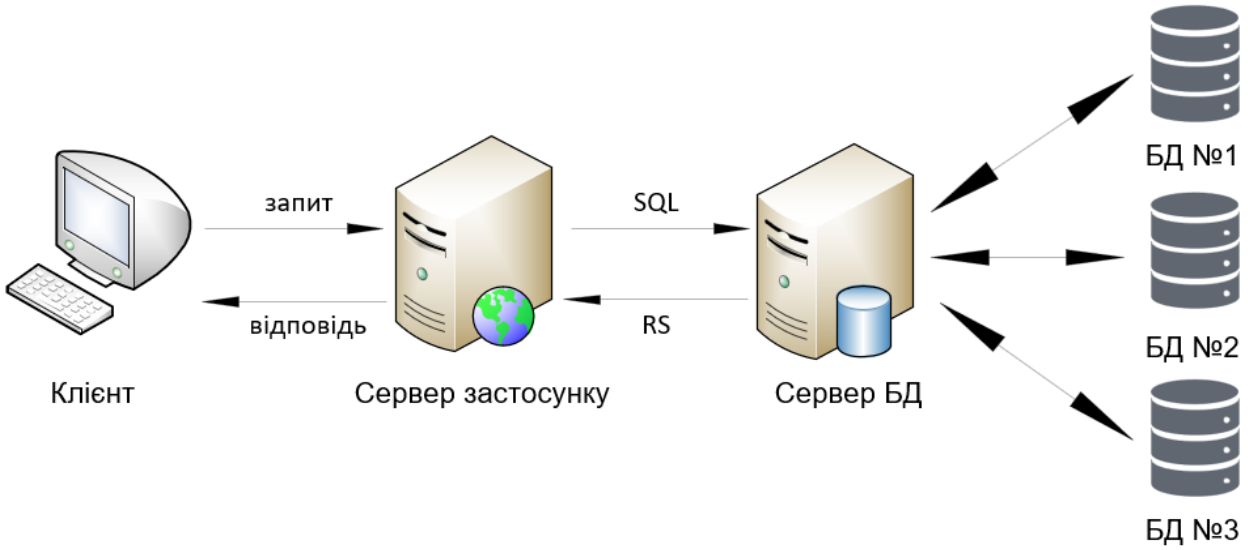


Рисунок 2.2 – Трирівнева архітектура клієнт-сервер

Серверна частина виконує логіку застосунку та обробку запитів від клієнтської частини. Вона приймає запити, обробляє їх та забезпечує клієнта необхідними даними.

База даних забезпечує збереження та доступ до даних, необхідних для роботи проекту. Вона забезпечує постійне зберігання інформації та доступ до неї при необхідності.

Таким чином використання трирівневої архітектури та патерна MVC дозволило ефективно організувати взаємодію компонентів та розподілити відповідальності між ними, що сприяло легшому та швидшому розвитку оновленню застосунку.

Розроблена підсистема складає одну з частин екосистеми ІС організаційного управління навчальним навантаженням.

На актуальний момент ІС складається з двох підсистем:

1) підсистема навчального відділу;
2) підсистема кафедри.

Поділ на підсистеми в даному випадку повністю умовний, оскільки усі таблиці, представлення, тригери, функції та процедури знаходяться в межах однієї бази даних та схеми. Віднесення компонент БД до певної підсистеми повністю залежить від тих задач, що поставлені до відповідної підсистеми.

Вказані підсистеми взаємодіють між собою та використовують інформацію, що може надавати кожна підсистема.

Підсистема навчального відділу надає інформацію щодо нормативів розрахунку навантаження, навчальних планів, викладацького складу та його штатного розкладу. З іншого боку підсистема кафедри надає перелік усіх доступних дисциплін ЗВО та видів робіт, що виконуються на різних етапах освітнього процесу.

## 2.2 Інформаційне моделювання предметної області підсистеми навчального відділу

Сутності та їх атрибути з описом обмежень, що потрібні для розв’язання поставлених задач, наведено в додатку Б.

Для реалізаціїї задач навчального відділу та додаткової задачи керування структурними компонентами ЗВО необхідно створити конкретний ряд сутностей, що дозволять вести облік, формування та керування різноманітними об’єктами освітнього процесу ОНУ імені І. І. Мечникова.

Опишемо сутності, які вони відображають, та їхні зв'язки.

Група таблиць обліку, формування та керування спеціальностями:

1) таблиця «field_of_knowledges» (галузі знань) представляє собою довідник, що містить дані про різні галузі знань за українським реєстром;

2) таблиця «int_field_of_knowledges» (міжнародні галузі знань) представляє собою довідник, що містить дані про різні галузі знань за міжнародним реєстром;

3) таблиця «specialities» (спеціальності) містить інформацію про доступні спеціальності всередині ЗВО. Кожна спеціальність має зв'язок з єдиною галуззю знань українського реєстру (one-to-many);

4) таблиця «infok_specialities» встановлює зв'язок між міжнародними галузями знань і спеціальностями ЗВО. Кожний запис в цій таблиці відображає відповідність між певною міжнародною галуззю знань та спеціальністю за допомогою зовнішніх ключів (many-to-many).

Група таблиць обліку, формування та керування структурними компонентами ЗВО:

1) таблиця «faculties» (факультети) представляє собою довідник, що містить дані про доступні факультети ЗВО;

2) таблиця «departments» (кафедри) містить інформацію про різні кафедри на кожному факультеті. Кожна кафедра має зв'язок з єдиним факультетом (one-to-many);

3) таблиця «users» (користувачі) зберігає список персон-користувачів системи. Кожний користувач, чи то викладач, чи гарант, чи виконавча особа, має тільки один запис персони у БД;

4) таблиця «positions» (посади) представляє собою довідник, що містить дані про різні посади. Кожний запис посади містить її максимально можливе навантаження за ставкою;

5) таблиця «department_assignments» (призначення до кафедри) встановлює зв'язок між користувачами та кафедрами. Кожний запис в цій таблиці відображає призначення персони на певну кафедру з певною виконавчою посадою. Задається зв'язок з відповідною кафедрою та відповідним користувачем (many-to-many);

6) таблиця «educators» (викладачі) містить дані про викладачів ЗВО. Кожний запис має зв'язки з відповідною кафедрою, користувачем та посадою (many-to-many (n = 3)). Один викладач може займати більше однієї посади на одній кафедрі, також один викладач може займати посади на різних кафедрах;

7) таблиця «rates» (ставки) містить дані про штатний розклад викладачів. Кожний запис в цій таблиці відображає ставку на певного викладача за певний рік. Кожний запис має зв'язок з відповідним викладачем (one-to-many).

Група таблиць обліку, формування та керування освітніми програмами:

1) таблиця «faculty_specialities» (спеціальності факультету) встановлює зв'язок між факультетами та спеціальностями. Кожний запис в цій таблиці відображає можливість викладання певної спеціальності на певному

факультеті. Кожний запис задає зв'язок відповідного факультету з відповідною спеціальністю (many-to-many). Та ж сама дисципліна може викладатись на декількох факультетатах;

2) таблиця «educational_programs» (освітні програми) містить дані про різні освітні програми. Кожна освітня програма має зв'язок з відповідною спеціальністю факультету та гарантом спеціальності (many-to-many). Запис в цій таблиці відповідає одній освітній програмі для певної спеціальності на факультеті конкретного освітнього рівня в конкретний рік;

3) таблиця «planned_disciplines» (заплановані дисципліни) визначає перелік дисциплін у рамках освітньої програми. Кожний запис в цій таблиці представляє дисципліну, що зв'язана з відповідною освітньою програмою (many-to-many). Кожна дисципліна може зустрічатись в освітній програмі лише один раз;

4) таблиця «blocked_disciplines» (заблоковані дисципліни) встановлює залежності логічного порядку викладання між дисциплінами в рамках освітньої програми. Кожний запис в цій таблиці вказує, яка дисципліна має викладатись перед іншою. Дана логіка задається зв'язком блокованої та блокуючої дисципліни (many-to-many). За правилом логічного порядку дисципліна не може блокувати саму себе.

Група таблиць обліку, формування та керування навчальними планами:

1) таблиця «educational_plans» (навчальні плани) зберігає дані про навчальні плани для конкретної освітньої програми. Кожний запис в цій таблиці відповідає окремому навчальному плану і має зв'язок з відповідною освітньою програмою (one-to-many). Запис в цій таблиці відповідає одному навчальному плану за певною ОП в конкретний рік;

2) таблиця «educational_disciplines» (навчальні дисципліни) містить дані про дисципліни, що включені до навчальних планів. Кожний запис в цій таблиці відображає зв'язок між навчальним планом та запланованою на ОП дисципліною (many-to-many);

3) таблиця «ed_per_semesters» (навчальні дисципліни на семестрі) визначає розподіл викладання дисциплін за семестрами в рамках конкретної дисципліни НП. Кожний запис в цій таблиці відповідає конкретній розподіленій на семестр частці дисципліни НП (one-to-many);

4) таблиця «ed_worktypes» (навчальні види робіт) встановлює зв'язок між семестровою дисципліною та видами роботи, що на ній виконуються. Кожний запис в цій таблиці відображає вид роботи, який виконується в рамках семестрої дисципліни НП (many-to-many);

5) таблиця «worktypes» (види робіт) виступає довідником, що містить інформацію про різні види робіт, що виконуються на різних рівнях освітнього процесу (навчальних планах, робочих планах, розрахунку навантаження).

Група таблиць обліку, формування та керування нормативами розрахунку навантаження:

1) таблиця «normative_groups» (нормативні групи) зберігає дані про існуючі групи нормативів;

2) таблиця «normatives» (нормативи) містить дані про конкретні версії нормативу, що належать до певної нормативної групи. Кожний запис в цій таблиці представляє окреме правило розрахунку навантаження виду роботи пов’язаного з конкретною нормативною групою (one-to-many).

Група таблиць обліку, формування та керування графіками тривалості семестрів освітнього процесу:

1) таблиця «semester_duration» (тривалість семестру) містить дані про тривалість семестру у тижнях. Кожний запис в цій таблиці представляє вказівку на тривалість відповідного семестру для конкретної ОКР.

ER-діаграма, отримана в результаті формалізації зв’язків підсистеми навчального відділу, приведена на рисунку 2.3. Для повноти відображення зв’язків до ER-діаграми додано декілька таблиць підсистеми кафедри. Для її побудування використано програмний застосунок Navicat 16 з використанням нотації «вороняча лапка».

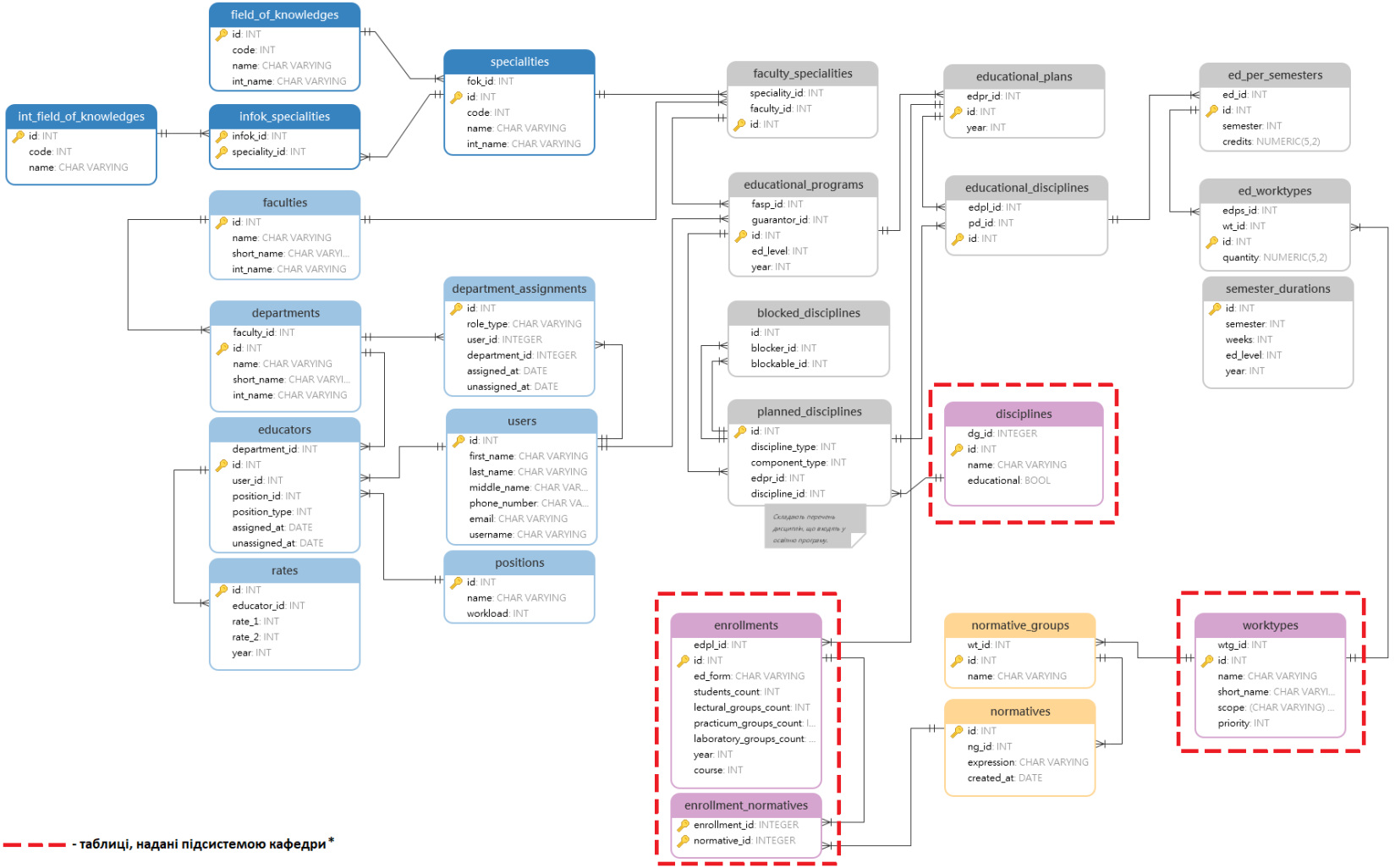


Рисунок 2.3 – ER-діаграма БД підсистеми «Навчальний відділ»

## 2.3 Вибір програмного забезпечення для створення ІС

При реалізації використано наступні технології та інструменти:

1) на бекенд-стороні використано фреймворк Rails v.7 на мові програмування Ruby. Даний фреймворк надає зручні інструменти для розробки серверної логіки та обробки запитів від клієнтів;

2) на фронтенд-стороні використано фреймворк React v.18 на мові програмування Javascript. Використання React дозволяє побудувати інтерфейси за допомогою компонентів, що спрощує розробку та повторне використання коду. Проект реалізовано як SPA (односторінковий додаток), що дозволяє більш плавно та ефективно взаємодіяти з користувачем. Для реалізації маршрутизації на фронтенд-стороні використано бібліотеку React Router v.6.4;

3) для доступу до даних використано бібліотеку ActiveRecord, яка надає можливості ORM для Ruby on Rails. Дана бібліотека спрощує взаємодію з базою даних та дозволяє виконувати операції з даними за допомогою об'єктно-орієнтованого підходу;

4) для взаємодії з базою даних використовується СУБД PostgreSQLv.15. Ця система обрана через свою безкоштовну ліцензію, доступну документацію та гнучкість управління даними;

5) для ініціювання та прийому HTTP-запитів бекенд-серверу використовується веб-сервер Puma. Він дозволяє ефективно обробляти багатопоточні запити та забезпечує стабільну роботу додатку під великим навантаженням;

6) взаємодія застосунку виконана за архітектурою REST API. Одночасно, незалежно один від одного, функціонують фронтенд та бекенд застосунки: бекенд-сервер, що взаємодіє з базою даних та отримає дані, підготовляє (сортує, фільтрує тощо) та надсилає на фронтенд-застосунок дані у JSON-форматі та фронтенд-застосунок, що виконує запити необхідних даних за потребою та використовує їх відповідно до місця, звідки ці дані запитані;

7) для розробки використано середу Visual Studio Code 2023, яка забезпечує зручну роботу розробника та підтримку необхідних розширень для Ruby та JavaScript;

8) розробка виконана на платформі Linux Ubuntu, що надає найбільш зручне робоче середовище для розробки та тестування проекту.

Використання даних технологій та інструментів (див. рис. 2.4) дозволило забезпечити ефективну розробку підсистеми навчального відділу, спростити процес взаємодії з користувачем та забезпечити необхідну функціональність.

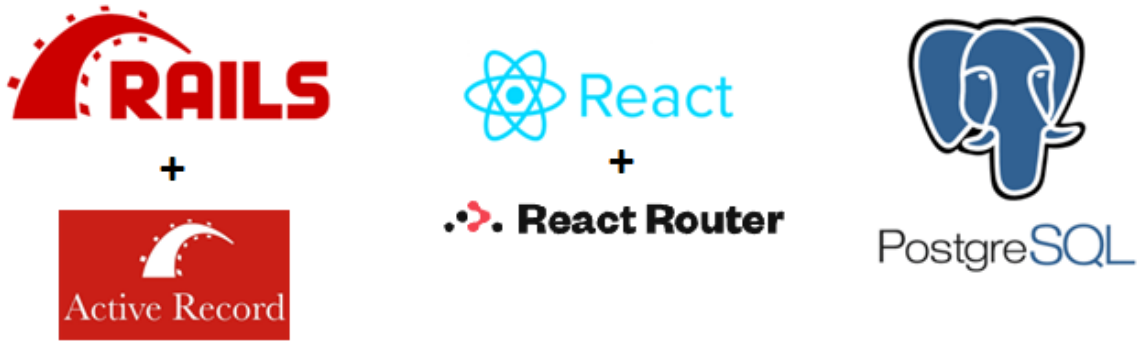


Рисунок 2.4 – Стек технологій застосунку

## 2.4 Програмна модель застосунка

Модель взаємодії рівня бізнес-логіки представлена різноманітними контролерами, які надають механізм отримання ресурсів (resources) через відповідні маршрути (routes).

Кожний ресурс відповідає одному контролеру та надає специфічний для Rails набір методів доступу, що відповідають певним стандартним HTTP-методам: [index (GET), show (GET/:id), create (POST), update (PATCH), destroy (DELETE)]. За бажанням список методів контролерів можна розширити, чи навіть створювати вкладені чи користувацькі маршрути, що можуть

знадобитись при проектуванні нестандартних контролерів. Приклад методу контролеру, що повертає ресурси освітніх програм та пов’язані з ним заплановані дисципліни надано у лістингу 2.1:

```
Rails.application.routes.draw do
  # /educational_programs, /educational_programs/1
  resources :educational_programs do
    # /articles/1/comments, /comments/1
    resources :planned_disciplines
  end

  root to: "users#index" # /
end
```

Лістинг 2.1 – Код методу-маршрутизатора бекенд-серверу

Для взаємодії з БД контролери використовують моделі бібліотеки ActiveRecord, що реалізують механізм ORM. Моделі надають зручний інтерфейс доступу до даних, що дозволяє виконувати на бекенд-стороні запити за допомогою різноманітних методів моделі, які при виконанні транслюються в SQL-запити, що звільняє розробника від необхідності створення великої кількості функцій для часткових випадків отримання даних, чи написання SQL-запитів прямо на бекенд-стороні.

Також ActiveRecord дозволяє напряму звернутись до бази даних з запитом на виконання конкретної процедури чи функції. Методи для SQL-запитів захищені від SQL-ін’єкцій за допомогою механізмів екранування строки запиту.

Рівень представлення являє собою окремий SPA-застосунок. SPA представляє собою веб-застосунок, в якому весь інтерфейс користувача генерується динамічно на клієнтському боці. При переході між сторінками не відбувається повного перезавантаження сторінки, а лише міняється вміст, що дозволяє забезпечити більш плавну і швидку навігацію.

У SPA є єдина точка входу – зазвичай це файл index.html, який містить основний шаблон сторінки. При першому завантаженні сторінки, браузер отримує цей файл, а потім виконується JavaScript-код, який завантажує необхідні компоненти і відображає їх на сторінці.

Після цього, при взаємодії користувача з додатком, зазвичай за допомогою навігаційних посилань або кнопок, спрацьовують маршрутизатори, які перехоплюють URL-шляхи та відповідно визначають, який компонент потрібно відобразити на сторінці. Виконується лише асинхронне завантаження необхідних даних з сервера при переході між шляхами, а не повне перезавантаження сторінки. Це дозволяє швидко та без перерв переходити між різними сторінками застосунку.

Використання SPA-застосунків є популярним підходом у сучасному веб-розробці. Вони дозволяють створювати багатосторінкові додатки зі зменшеним часом завантаження між сторінками та поліпшеним користувацьким досвідом. Більш того, SPA можуть бути побудовані з використанням різних JavaScript-фреймворків, таких як React, Angular або Vue.js.

Повна ієрархія маршрутів фронтенд-застосунку надана на рис. 2.5. На рисунку зображені усі доступні сторінки в залежності від обраної групи відображення.

У SPA-застосунках весь процес відображення сторінок відбувається на клієнтському боці, що означає, що бекенд-сервер має надавати API для обміну даними з клієнтом. Це реалізовано за допомогою розробки API бекенд-серверу, який відповідає на запити клієнта та забезпечує необхідну функціональність для роботи з даними.

Також, при розробці SPA-застосунків важливо враховувати аспекти безпеки, оскільки частину логіки та обробки даних перенесено на клієнтський бік. Забезпечення аутентифікації, авторизації та захисту даних є важливими аспектами при проектуванні SPA-застосунків.

У підсумку, SPA-застосунки забезпечують більш ефективну та зручну навігацію для користувачів, зменшують завантаження сервера та спрощують розробку шляхом повторного використання компонентів. Використання API бекенд-серверу є необхідним для взаємодії між клієнтом та сервером у SPA-застосунках.

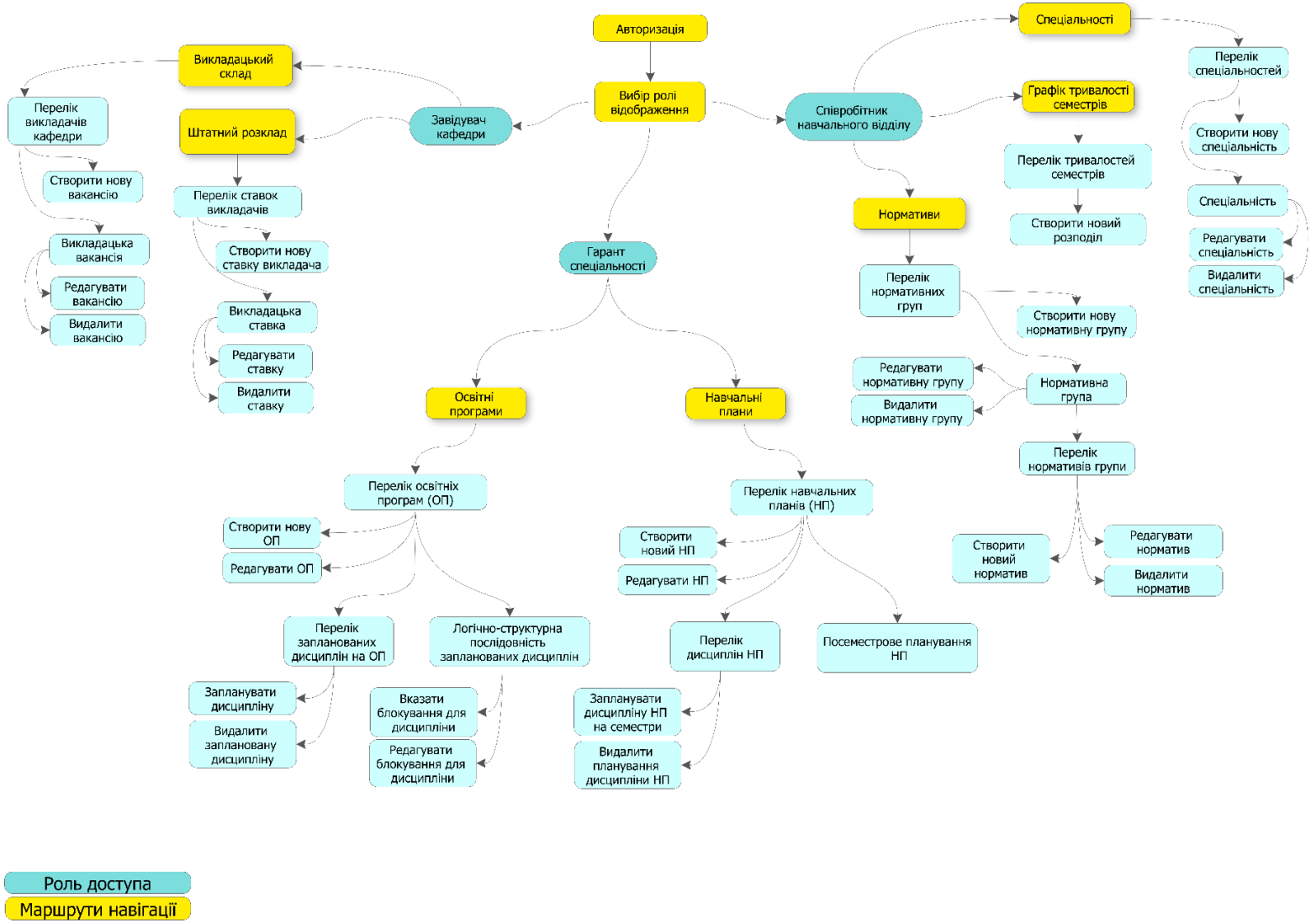


Рисунок 2.5 – Ієрархія маршрутів фронтенд-застосунку підсистеми навчального відділу

# 3 ПРОГРАМНА РЕАЛІЗАЦІЯ

## 3.1 Розв’язання задач користувачів за допомогою SQL-запитів

Для вирішення задач підсистеми навчального відділу необхідні різноманітні SQL-запити, функції та тригери.

SQL-запити на створення таблиць підсистеми БД надано у додатку В.

SQL-запити на створення додаткових функцій та тригерів збереження цілістності даних підсистеми БД надано у додатку Г.

Більшість SQL-запитів, що надсилаються до бази даних є автоматично сгенеровані за допомогою ORM бібліотеки ActiveRecord. У даному випадку, усі операції, що потребують звернення до даних, описані у вигляді послідовного виклику методів на класі моделі, що в результаті транслюється у SQL-запит.

Наведемо приклад операції для отримання суми кредитів по усім семестрам викладання дисципліни навчального плану. В оточенні Rails Console дану задачу можна виконати наступним чином:

```
pry(main)> total_credits = EducationalDiscipline
     .joins(:ed_per_semesters)
     .where(ed_per_semesters: { semester: semester })
     .sum(:credits)
=> SELECT SUM("educational_disciplines"."credits") AS
sum_credits
FROM "educational_disciplines"
JOIN "ed_per_semesters" ON
"ed_per_semesters"."educational_discipline_id"="educational_disc
iplines"."id"
WHERE "ed_per_semesters"."semester" = <значення семестру>
=> 120
```

Наглядно видно, що оператор *joins, where* та *sum* моделі трансформувалися в оператори JOIN, WHERE та SUM SQL-запиту у БД. Обмежень на запити для бібліотеки немає – можна виконувати групування, агрегацію, отримання унікальних значень, поєднувати таблиці між собою тощо.

Для вирішення задач користувачів виконуються наступні SQL-запити:

1) для вирішення задач НВ1, НВ5, НВ8, Г1, Г3, Г5, Г7, Г9, ЗК1 та ЗК4 виконується запит з підстановкою відповідних імен таблиць, списку стовпців та умов, коли вони необхідні типу:

```
SELECT список_стовпців FROM таблиця [WHERE умова];
```

2) для вирішення задач НВ2, НВ4, НВ6, НВ9, Г2, Г4, Г6, Г8, Г10, Г12, ЗК2, ЗК5, А1-А5, А7 та А9 виконується запит з підстановкою відповідних імен таблиць та списку стовпців типу:

```
INSERT INTO таблиця(список_стовпців) VALUES (список_стовпців);
```

3) для вирішення задач НВ3, НВ7, Г11, ЗК3, ЗК6, А6, А8 та А10 виконується запит з підстановкою відповідних імен таблиць та списку необхідних стовпців. типу:

```
UPDATE таблиця SET стовпець_1 = значення_1 […, стовпець_n = значення_n] [WHERE умова]
```

Зазначені вище типи SQL-запитів відображають конкретні CRUD-операції. Хоча окремі запити можуть виглядати просто, фактичний процес обробки даних виявляється більш складним. Наприклад, при вибірці даних для отримання штатного розкладу викладачів за певний рік, застосовується фільтрація та групування даних у базі даних за допомогою трансльованого SQL-запиту.

```
SELECT e.id AS educator_id, r.year, r.rate_1, r.rate_2
FROM educators e
JOIN rates r ON e.id = r.educator_id
GROUP BY e.id, r.year, r.rate_1, r.rate_2;
```

Даний процес відбувається в межах дій контролерів, які використовують моделі для отримання відповідних ресурсів (таблиць). Такі операції виконуються безпосередньо на рівні бази даних, що дозволяє ефективно обробляти великі обсяги даних і повертати потрібні результати «на льоту».

Для вирішення деяких задач застосовуються тригери. Наприклад, для реалізації задачі Г8 використовується тригер (див. додаток Д, лістинг 1), який після створення запису нового навчального плану автоматично формує перелік дисциплін навчального плану на основі спланованого переліку дисциплін освітньої програми. Дана можливість спрощує взаємодію користувача та позбавляє його від необхідності вручну підв’язувати дисципліни з ОП.

## 3.2 Програмна реалізація інтерфейсу користувача

### 3.2.1 Реалізація на стороні фронтенд-застосунку

Для побудування інтерфейсу взаємодії на стороні фронтенд-застосунку використовується компонентний підхід [7].

Компонентний підхід у React є однією з ключових концепцій цієї бібліотеки і грунтується на ідеї поділу інтерфейсу користувача на маленькі частини, звані компонентами.

Компоненти React можуть бути *класовими* або *функціональними*. Функціональні компоненти є більш простим та зручним способом створення компонентів у сучасному підході до розробки React-застосунків, оскільки вони використовують хуки для побудови взаємодії. З вказаних причин для використання обрано саме функціональні компоненти.

Компоненти взаємодіють один з одним шляхом включення одних компонентів усередину інших. Це дозволяє будувати ієрархічну структуру компонентів, де компоненти можуть бути вкладені інші компоненти, формуючи деревоподібну структуру.

Дана структура дозволяє створювати складні інтерфейси користувача, де кожен компонент відповідає за свою ізольовану частину функціональності.

Компоненти також можуть містити різні функції, які виконують певні задачі у контексті компонента. Функція може змінювати стан компонента або виконувати інші дії, пов'язані з взаємодією користувача. У додатку Ж.1 надано типовий приклад функціонального компоненту в React, що представляє форму для створення та оновлення нормативної групи.

Даний функціональний компонент надає наочний опис елементів логіки, що може містити компонент на фронтенд: використання хуків для збереження стану компоненту та автоматичного заповнення даних форми, власні функції для обробки даних, валідацій форм, асинхронних API-запитів та валідацію зі сторони бекенду.

### 3.2.2 Реалізація на стороні бекенд-серверу

*Моделі* в Ruby on Rails – це ключовий аспект розробки веб-додатків з використанням даного фреймворку [8].

Основні функціональні можливості моделей в Rails включають:

1) автоматичну генерацію SQL-запитів. Моделі надають велику кількість методів, що дозволяють виконувати такі оператори як where, group_by, select, join та інші;

2) валідацію даних. Моделі можуть містити правила валідації, які дозволяють перевіряти правильність введених даних перед збереженням в базу даних;

3) асоціації між моделями. Між моделями можна встановити взаємодії за допомогою асоціацій, таких як «один-до-одного», «один-до-багатьох» та «багато-до-багатьох». Це дозволяє виконувати зв'язані запити до бази даних та працювати з пов'язаними об'єктами зручним способом;

4) фільтрацію та сортування даних. Моделі надають можливість виконувати фільтрацію, сортування та обмеження вибірки даних з БД;

5) подібність до об'єктів. Моделі в мають поведінку, подібну до звичайних класів в ООП. Вони можуть мати методи, змінні і статичні члени, що дозволяє легко взаємодіяти з даними, агрегувати дані та виконувати бізнес-логіку в контексті моделі.

У додатку Ж.2 надано типовий приклад моделі в Rails, що представляє таблицю спеціальностей. Дана модель надає наочний опис функціональних елементів, що може містити модель: асоціації між декількома моделями, функції-агрегатори властиві класу моделі, фільтратор записів за включенням тексту до відповідних полів та валідації на рівні бекенду.

*Контролери* в Ruby on Rails виконують роль посередників між моделями та відображенням застосунку.

Основні функціональні можливості контролерів в Rails включають:

1) маршрутизацію HTTP-запитів. Контролери використовуються для відповіді на HTTP-запити від користувачів. Вони співставляються з визначеними маршрутами, що визначають, які контролери та дії викликати для кожного типу запиту;

2) виклик дій моделей. Контролери викликають відповідні дії моделей для взаємодії з базою даних. В рамках дій вони виконують CRUD-операції, валідацію даних, встановлюють асоціації та інші операції, необхідні для роботи з даними;

3) обробка параметрів. Контролери отримують параметри, передані в HTTP-запиті, та обробляють їх. Вони можуть проводити перевірку та валідацію параметрів, встановлювати їх у моделі для подальшої обробки;

4) методи-фільтри. Контролери надають можливість виконувати визначені у ньому методи на певних етапах обробки HTTP запиту (до, після, чи одночасно обидва). Наприклад, можна виконати підготовку даних моделі «Specialities» перед початком дії контролеру, що використовує ці дані. Дані методи дозволяють перевикористовувати код та притримуватись DRY-підходу (don`t-repeat-yourself).

У додатку Ж.3 надано типовий приклад контролеру в Rails, що виконує операції з ресурсом моделі «Discipline». Даний контролер надає наочний опис функціональних елементів та дій, що може містити контролер.

## 3.3 Безпека інформаційної системи

### 3.3.1 Обмеження доступу на рівні бази даних

Для забезпечення безпеки на рівні бази даних відбувається створення облікових записів кожного користувача у вигляді ролей бази даних та видача їм привілеїв певної групової ролі.

Кожен користувач у базі даних автоматично отримує власний обліковий запис, який створюється при додаванні нового запису до таблиці Users.

Кожний новий користувач за замовченням не належить до жодної з рольових груп та не має прав на взаємодію з елементами БД.

Адміністратор системи може надати права конкретної групової ролі будь-якому користувачу.

Список доступних групових ролей:

1) викладач: educator_group;
2) завідувач кафедри: departhead_group;
3) відповідальна особа: departexec_group;
4) гарант спеціальності: guarantor_group;
5) співробітник навчального відділу: educdepart_group;
6) адміністратор ІС: admin_group.

Кожній груповій ролі в базі даних встановлені права доступу до відповідних компонентів (таблиць, представлень, функцій тощо).

В межах підсистеми навчального відділу взаємодіють користувачі з груповими ролями: співробітник навчального відділу, завідувач кафедри, гарант спеціальності та адміністратор ІС.

Ознайомитись зі списком привілеїв ролей на об’єкти БД для ролей підсистеми навчального відділу можна в додатку Л.

### 3.3.2 Обмеження доступу на рівні бекенд-серверу

За замовчуванням Rails-сервер виконує запити до бази, що надходять на відповідні контролери, від імені користувача за замовченням.

Під час авторизації на фронтенд-стороні, користувач вводить логін та пароль у відповідній формі, після чого відправляється запит на авторизацію до *контролеру авторизації*.

Отримуючи дані користувача, авторизаційний контроллер налаштовує конфігурацію підключення до бази даних в межах одного запиту та робить спробу підключення.

Якщо користувача з такими обліковими даними не існує, або підключення не відбулось, то на фронтенд повертається повідомлення про помилку авторизації.

Якщо користувача знайдено, тоді генерується JWT-токен [9], що містить логін та пароль, що буде використовуватись для наступних звернень до бекенд-серверу.

Немає необхідності піклуватись про те, що дані містять приватну інформацію, оскільки токен надає інформацію у зашифрованому вигляді за допомогою алгоритму HMAC256. Без секретного ключа, що зберігається у приватному сховищі бекенд-серверу, отримати вміст токену просто неможливо. До того ж токен має обмежений термін дії, що зменшує можливість компрометування приватних даних, оскільки виникає необхідність оновити сесію через 24 години після видачи минулого токену.

Токен може бути збережений в локальному сховищі фронтенду для подальшої перевірки авторизації.

До усіх наступних звернень до бекенд-серверу буде додано токен у заголовку Authorization у форматі стандарту Bearer-токен (строка «Bearer #{token}»). Бекенд-сервер буде перевіряти наявність токену на усіх контролерах, окрім авторизаційного. Якщо токен не буде надано – клієнт отримує відмову запиту та помилку 401 Unauthorized.

Після успішної авторизації, на бекенді встановлюється з'єднання з базою даних за обліковими даними користувача, і всі наступні запити виконуються в рамках цього з'єднання з відповідними правами доступу.

### 3.3.3 Обмеження доступу на рівні фронтенд-застосунку

Під час авторизації, користувач вводить логін та пароль у формі на сторінці /login та відправляє запит на авторизацію бекенд-серверу.

Якщо процес авторизації виконується коректно, то користувач отримує JWT-токен, що ініціює його сесію.

Разом з токеном на фронтенд передається інформація про термін дії токену, що запланувує процес його видалення з сховища.

Після авторизації користувач отримує доступні групові ролі користувача та переходить на сторінку /choose-view, де надається можливість обрати формат відображення додатку залежно від тих прав користувача, що відповідають наданим в базі даних.

Якщо користувач не належить до жодної з груп, то за замовчуванням вибирається група, яка має доступ лише до власного кабінету.

Обрана опція зберігається в контекстному сховищі React-застосунку та залишається активною в рамках сесії. Якщо сторінка повністю перезавантажується, то виконується повторний запит доступних груп з серверу та вибір бажаного відображення. Залежно від обраної ролі, фронтенд відображає відповідний інтерфейс та взаємодії доступні тільки даній ролі. Зміна відображення можлива в процесі роботи без перезавантаження сесії та переавторизації.

Для ліпшого користувацького досвіду реалізовано механізм правової авторизації (Permission-based authorization), що дозволяє відображати тільки ті компоненти інтерфейсу, як, наприклад, кнопки та форми, або навіть цілі сторінки, до ресурсів яких вони мають доступ.

### 3.4 Демонстрація функціонування підсистеми

Під час першого відвідування веб-сайту користувач має авторизуватись (див. рис. 3.1). Для перегляду функціонуючої підсистеми поглянемо на неї від обличчя гаранта.

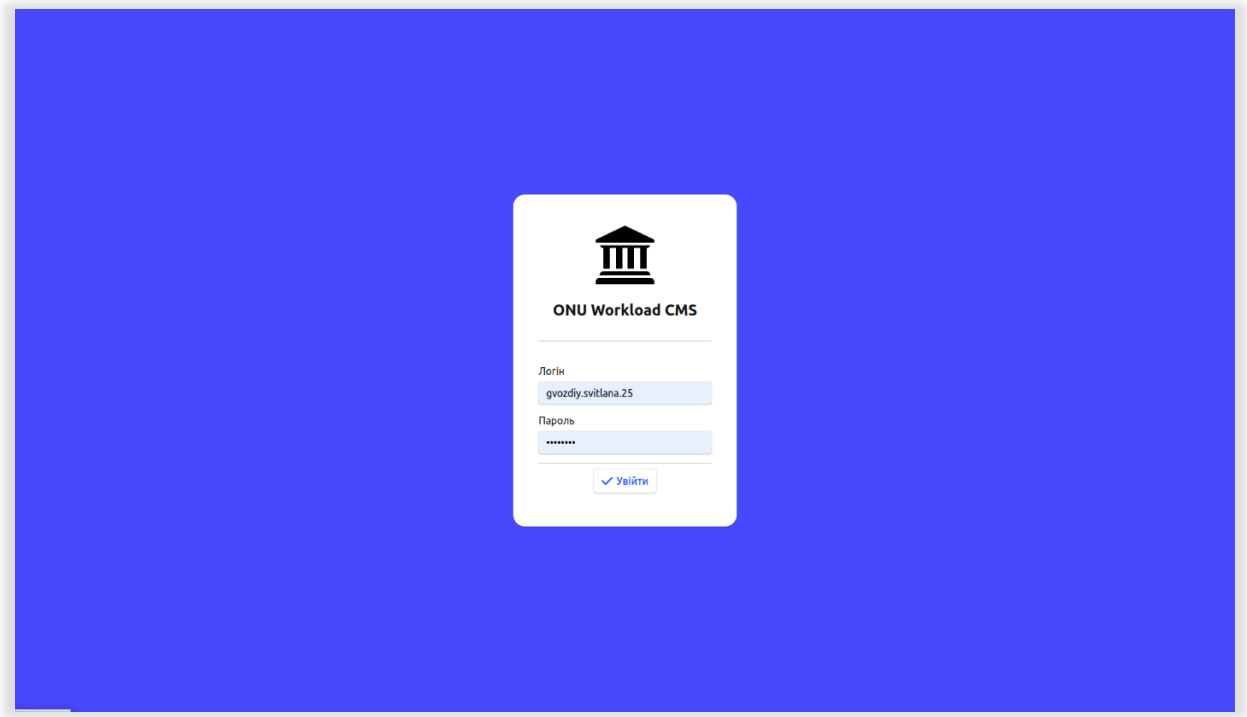


Рисунок 3.1 – Форма авторизації до системи

Після успішної авторизації в системі, користувачу необхідно обрати одну з доступних ролей, які він може переглядати (див. рис. 3.2). Список доступних ролей залежить від груп, до яких належить авторизований користувач. Якщо користувач ще не належить до жодної групи, йому буде доступна лише роль «гість».

Після вибору ролі, інтерфейс користувача буде змінюватись відповідно до обраного формату відображення, та будуть надаватись лише компоненти, що є специфічними для обраної ролі. Наприклад, якщо користувач обрав роль «адміністратора», то йому будуть доступні компоненти та функціонал, які дозволяють керувати системою, додавати або видаляти усі ресурси.

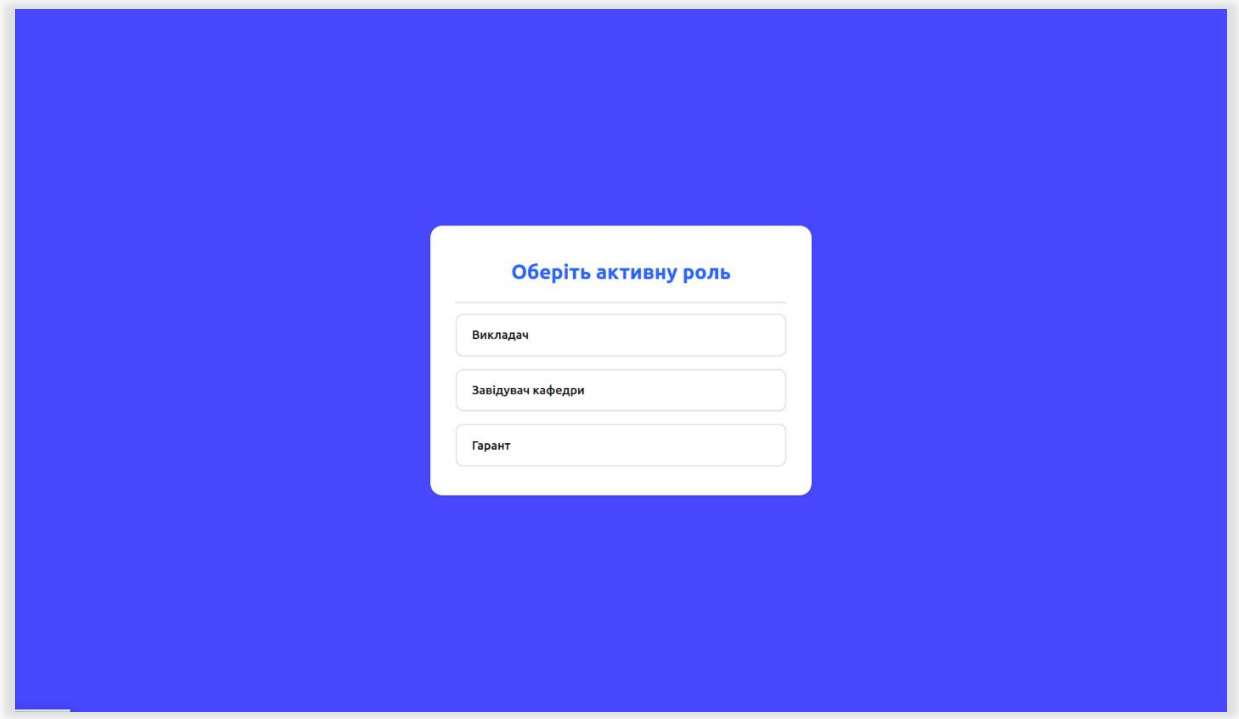


Рисунок 3.2 – Вікно вибору формату відображення інтерфейсу

Після авторизації користувач може виконувати навігацію за допомогою маршрутних посилань та кнопок навігації у боковій панелі (див. рис. 3.3).

На рисунку зображені інтерфейсні елементи, які надають користувачу можливості:

1) завершити сесію користувача;
2) змінити формат представлення, вибравши одну серед доступних наданому користувачу опцій;
3) перейти у особистий кабінет викладача;
4) перейти на будь-яку доступну для даної ролі сторінку

Користувач із правами гаранта може отримати доступ до сторінки, що містить процеси керування освітніми програмами або навчальними планами та виконувати задачі керування навчальним навантаженням.

Через навігаційну панель гарант може перейти до списку усіх доступних даному користувачу освітніх програм (див. рис. 3.4).

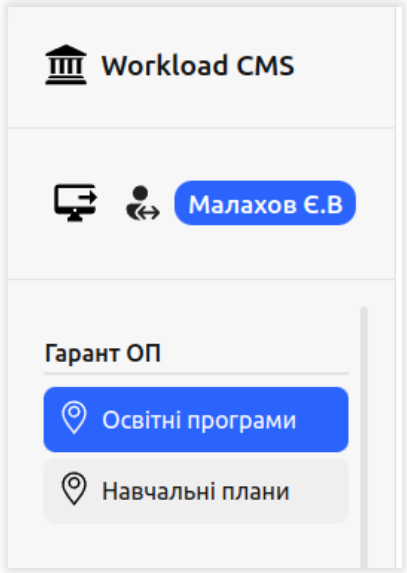


Рисунок 3.3 – Навігаційна панель ролі «Гарант ОП»

Користувач із правами гаранта може отримати доступ до сторінки, що містить процеси керування освітніми програмами або навчальними планами.

Гарант через навігаційну панель може перейти до списку освітніх програм (див. рис. 3.4).

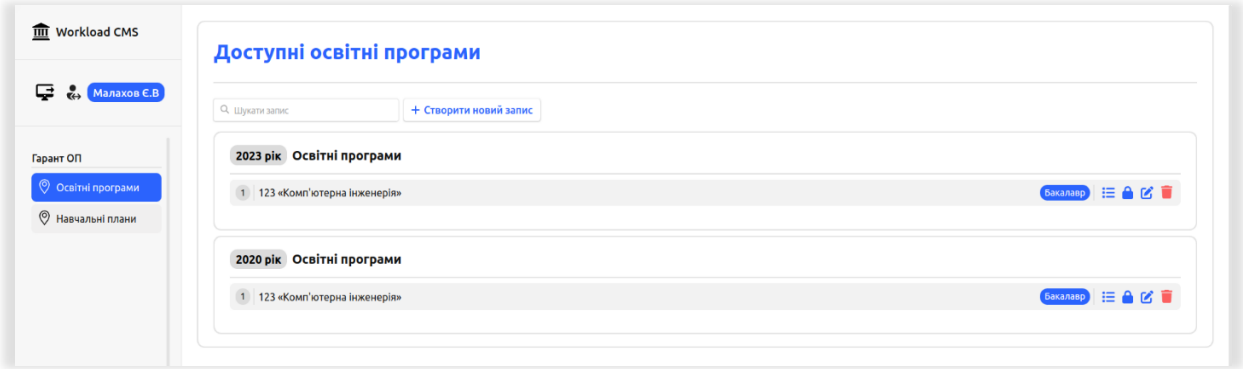


Рисунок 3.4 – Списки доступних освітніх програм гаранта

Обравши конкретну ОП, гарант може потрапити до списку запланованих на ній дисциплін, запланувати нову дисципліну чи видалити відповідне планування (див. рис. 3.5 та 3.6).

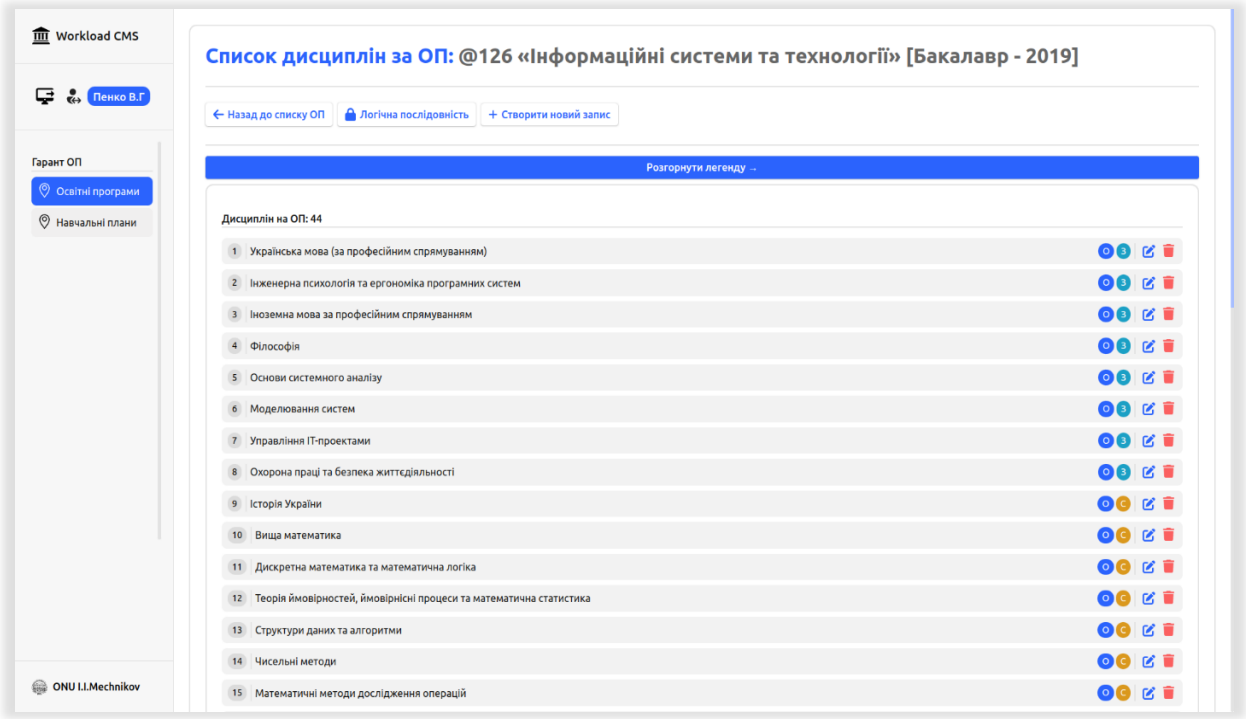


Рисунок 3.5 – Списки запланованих дисциплін на ОП

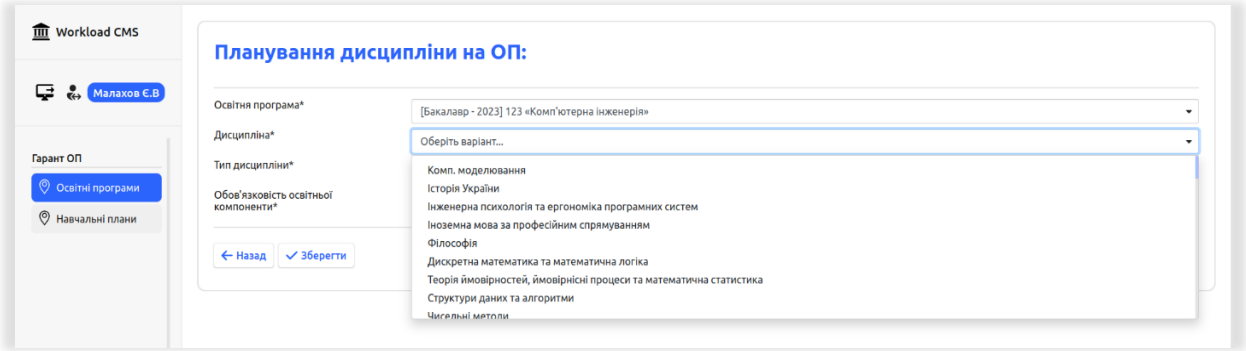


Рисунок 3.6 – Форма створення нової дисципліни на ОП

За відповідною запланованою дисципліною гарант може сформувати структурно-логічну послідовність викладання, вказавши які дисципліни мають бути викладені раніше за іншу дисципліну (див. рис. 3.7). Дані структурно-логічної послідовності використовуються для запобігання порушень у плануванні видів робіт на семестрових дисциплінах на відповідній сторінці.

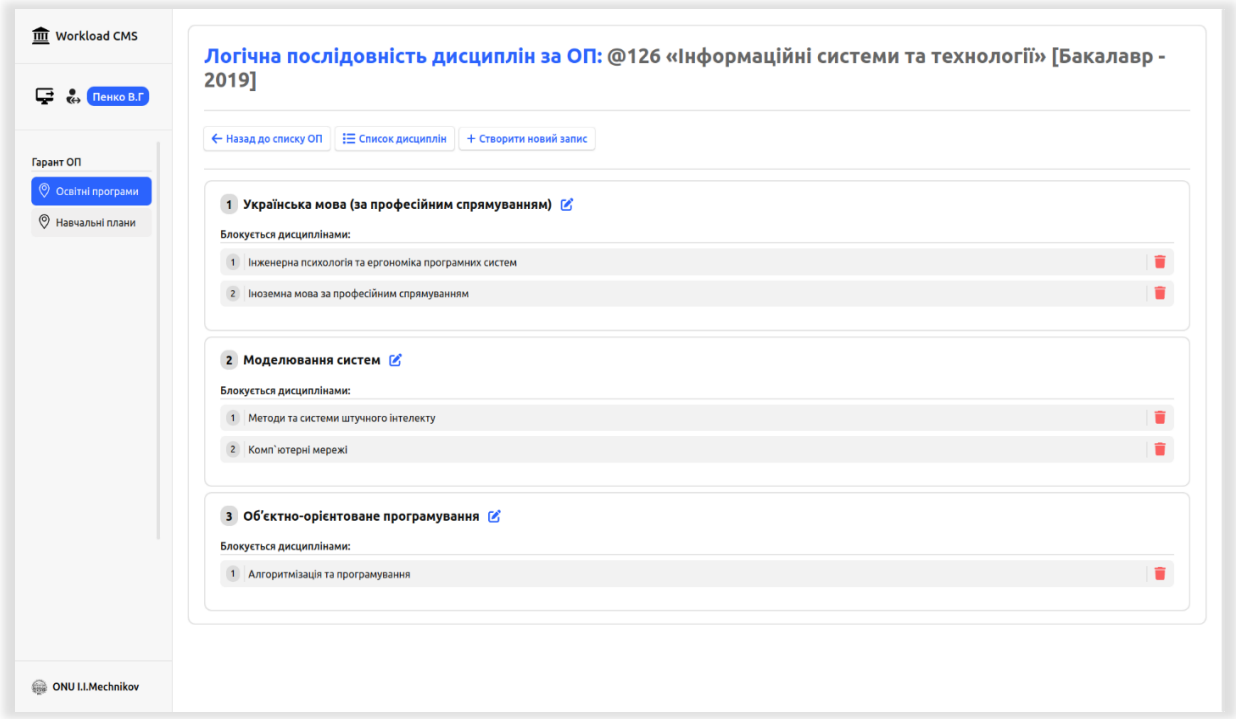


Рисунок 3.7 – Список груп логічно-структурних обмежень дисциплін

Гарант може виконувати керування навчальними планами та займатись розподілом навчального навантаження (див. рис. 3.8).


Рисунок 3.8 – Списки доступних навчальних планів

Процес керування навчальними планами включає розподіл дисциплін навчального плану за відповідними семестрами (див. рис. 3.9). Користувач може використовувати форму для встановлення відповідності викладання дисципліни на множині семестрів.

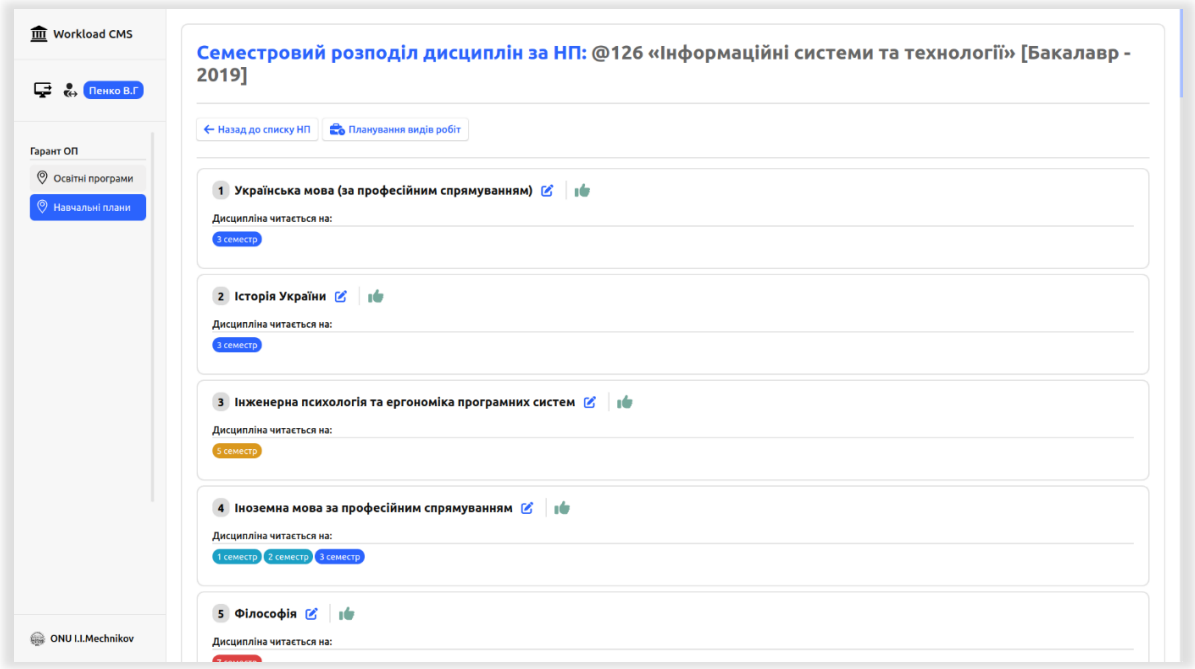


Рисунок 3.9 – Розподіл дисциплін навчального плану за семестрами

Після розподілу дисциплін за семестрами, викладач має можливість займатись плануванням та розподілом навчального навантаження на спеціальній сторінці шляхом встановлення відповідних видів робіт та їх обсягів на конкретний семестр (див. рис. 3.10).

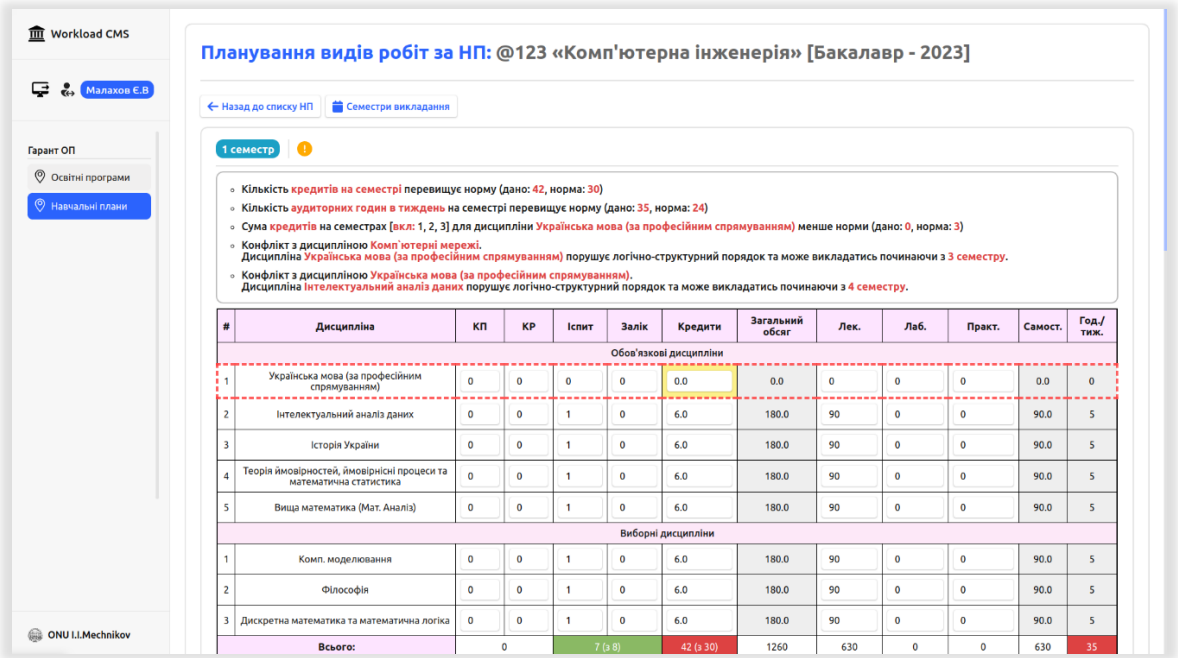


Рисунок 3.10 – Планування та розподіл навчального навантаження

Таблиця розподілу при плануванні видів робіт надає звідну інформацію за переліком показників, що допомагає гаранту відслідковувати без необхідності ручного перерахунку, що значення відповідних видів робіт, загальної кількості робіт, сум кредитів за семестр чи в рамках дисципліни, відсоткового складу аудиторних годин від загального обсягу, правила логічного слідування дисциплін тощо, знаходяться у межах домовленостей.

Дана форма є чудовим прикладом автоматизації, яку надає реалізований функціонал підсистеми. Якщо централізація даних відчувається не одразу, то відсутність необхідності виконувати значну кількість одноманітних та часом «важких» арифметичних операцій допомагає швидше приймати рішення та витрачати час тільки на корисні операції.

# ВИСНОВКИ

В результаті аналізу предметної області сформульовано задачі, що необхідні для досягнення мети проекту, визначено основні особливості функціонування підсистеми навчального відділу у межах ІС організаційного управління навчальним навантаженням, визначено список груп користувачів підсистеми (співробітник навчального відділу, гарант спеціальності, завідувач кафедри та адміністратор).

Для реалізації дипломного проекту обрано трирівневу архітектуру та шаблон проектування MVC, СУБД PostgreSQL, фреймворк Rails (ЯП Ruby) для реалізації бекенду та бібліотеку ActiveRecord для взаємодії з базою даних, фреймворк React (ЯП Javascript) для реалізації фронтенду та бібліотеку React Router для досягнення багатосторінковості застосунку.

У ході розробки створено компоненти бази даних, що складають 22 таблиці, 6 функцій та 6 тригерів.

Більшість тригерних функцій забезпечують цілісність бази даних, а механізм валідацій на моделях на стороні бекенду та користувацьких валідацій на стороні фронтенду додатково відловлюють потенційні помилки та порушення правил бізнес-логіки.

Завдяки рольовому розмежуванню повноважень серед різних груп користувачів забезпечено захист від несанкціонованого доступу у систему.

Створена підсистема навчального відділу успішно реалізує усі функціональні можливості відповідно до задач, визначених на етапі постановки задачі:

1) облік структурної організації ЗВО;
2) облік доступних спеціальностей;
3) керування освітніми програмами та планами;
4) керування нормативами розрахунку навантаження;
5) керування графіками тривалостей семестрів навчання;
6) облік викладацького складу та штатного розкладу.

Реалізація підсистеми навчального відділу має ряд недоліків та недоробок, які можуть бути вирішені у майбутніх версіях застосунку:

1) більшість з задач, що стосуються ведення довідників системи покладена на плечі адміністратора ІС;

2) немає повноцінного механізму видачі групових ролей користувачам окрім мануальної видачі прав адміністратором з Query Tool;

3) завідувач кафедри, щоб бути співставленим з певною кафедрою, має бути записаний до відповідної таблиці бази даних. Дана процедура має виконуватись якимсь керівним органом, наприклад, відділом кадрів;

4) гарант спеціальності не може створити нову дисципліну, щоб запланувати її в ОП;

5) освітні програми та навчальні плани не проходять перевірку / затверджуються навчальним відділом.

Дана підсистема має багато перспектив розвитку:

1) ведення обліку актуальних контингентних груп та керування студентським складом (інтеграція з ЄДЕБО);

2) механізм генерації формування робочих планів;

3) формування розкладу навчання з врахуванням доступних ресурсів ЗВО (професорсько-викладацького складу та аудиторно-технічної бази) та розподілу викладацького навантаження.

Перелік можливих перспектив розвитку можна перелічувати до нескінченності, оскільки створена підсистема навчального відділу у межах ІС організаційного управління навчальним навантаженням надає міцний фундамент для створення нових модулів та підсистем на її основі.

Репозиторії створених застосунків для реалізації фронтенд-сторони [10] та бекенд-сторони [11] проекту інформаційної системи викладені на платформі Github.

За результатами даної роботи опубліковано тези на всеукраїнській конференції та відбулось впровадження в технологічний процес ЗВО (див. додаток М).

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. Джигов Д.Ю., Підсистема навчально-методичного відділу інформаційної системи управління навчальним процесом / Д.Ю. Джигов, Є.В. Малахов // Інформатика, інформаційні системи та технології: тези доповідей XX Всеукр. конференції студентів і молодих науковців. Одеса, 28 квітня 2023 р. – Одеса, 2023. – С. 110-112.
2. Аналіз деяких вітчизняних інформаційно-аналітичних WEB-орієнтованих систем управління навчальним процесом у вищих школах // «Наукові записки Бердянського державного педагогічного університету». – Випуск 3. – Серія: Педагогічні науки. – Бердянськ : БДПУ, 2017. – 288 с. – С. 275–281.
3. Про навчальний відділ. Склад та задачі навчального відділу. [Електронний ресурс] – Режим доступу: https://onu.edu.ua/uk/acad-office
4. Програмне забезпечення для вищих навчальних закладів України, пакет програм «Деканат» (Політек- СОФТ) [Електронний ресурс]. – Режим доступу: http://www.politek-soft.kiev.ua/index.php?do=products&product=deanery
5. Автоматизована система управління ЗВО усіх рівнів акредитації «Вищий навчальний заклад» (НДІ ПІТ) [Електронний ресурс]. – Режим доступу: https://vuz.osvita.net/as-dekanat/
6. Автоматизована система управління ЗВО III - IV рівня акредитації (Unitex+) [Електронний ресурс]. – Режим доступу: http://www.unitex.com.ua/products/commercial-software/automated-system-for-higher-education-institution/
7. Офіційна документація фреймворку React [Електронний ресурс]. – Режим доступу: https://react.dev/learn
8. Офіційна документація фреймворку Ruby on Rails [Електронний ресурс]. – Режим доступу: https://guides.rubyonrails.org/

9. Introduction to JSON Web Tokens [Електронний ресурс]. – Режим доступу: https://jwt.io/introduction
10. Репозиторій фронтенд-сторони проекту [Електронний ресурс]. – Режим доступу: https://github.com/leew1e/onu-workload-ui
11. Репозиторій бекенд-сторони проекту [Електронний ресурс]. – Режим доступу: https://github.com/leew1e/onu-workload-backend

# ДОДАТОК А

## Задачі користувачів

Таблиця А.1 – Список задач користувачів підсистеми «Навчальний відділ»

| Номер | Задача | Вхідні дані | Вихідні дані |
|---|---|---|---|
| Співробітник навчального відділу | | | |
| НВ1 | Переглядати перелік спеціальностей ЗВО | Критерій відбору (галузь знань, назва спеціальності) | Перелік спеціальностей |
| НВ2 | Реєструвати нову спеціальність | Назва спеціальності, ієрархічний код, інтернаціоналізована назва, відповідна українська галузь знань, список відповідників міжнародних галузей знань | Нова спеціальність |
| НВ3 | Редагувати дані спеціальності | Редагована спеціальність, оновлені: [назва, інтернаціоналізована назва, список відповідників міжнародних галузей знань]. | Оновлена спеціальність |
| НВ4 | Реєструвати спеціальність на факультеті | Конкретна спеціальність, конкретний факультет | Спеціальність, що допускається до формування освітніх програм в ЗВО |
| НВ5 | Переглядати перелік нормативів розрахунку навантаження | Критерій відбору (нормативна група) | Перелік нормативів розрахунку навантаження |
| НВ6 | Створювати нормативи навантаження для виду роботи | *Нормативна група:* назва групи, вид роботи. *Норматив:* вираз розрахунку значення навантаження | Новий норматив розрахунку навантаження для виду роботи |

Продовження таблиці А.1

| Номер | Задача | Вхідні дані | Вихідні дані |
|---|---|---|---|
| НВ7 | Редагувати нормативи навантаження для виду роботи | Редаговані нормативні групи та нормативи, оновлені: [*нормативна група*: назва групи; *норматив*: формула розрахунку] | Оновлений норматив |
| НВ8 | Переглядати графіки тривалості семестрів | Критерій відбору (рік, освітній рівень) | Перелік тривалості семестрів |
| НВ9 | Планувати графіки тривалості семестрів для різних ОКР | Семестр, кількість тижнів, освітньо-кваліфікаційний рівень, рік дії | Графік семестру для конкретного ОКР |
| Гарант спеціальності | | | |
| Г1 | Переглядати нормативи розрахунку навантаження | Критерій відбору | Перелік нормативів розрахунку навантаження |
| Г2 | Створювати нові освітні програми | Доступна спеціальність на факультеті, гарант спеціальності, ОКР, рік створення програми (опціонально). | Нова освітня програма спеціальності |
| Г3 | Переглядати заплановані дисципліни на ОП | Критерій відбору (освітня програма) | Перелік запланованих дисциплін |
| Г4 | Запланувати дисципліну на освітню програму | Дисципліна, освітня програма, тип дисципліни (загальноосвітня чи спеціальна), тип освітньої компоненти (вибірна чи обов’язкова) | Запланована дисципліна на освітній програмі |
| Г5 | Переглядати логічний порядок викладання дисциплін ОП | Критерій відбору (освітня програма) | Перелік дисциплін-блокувальників викладання |

Продовження таблиці А.1

| Номер | Задача | Вхідні дані | Вихідні дані |
|---|---|---|---|
| Г6 | Встановити логічний порядок між запланованими дисциплінами | Запланована дисципліна, список запланованих дисциплін (блокерів), що мають бути викладені до неї | Запланована дисципліна з обмеженням порядку |
| Г7 | Переглядати доступні навчальні плани | Критерій відбору (рік) | Перелік навчальних планів |
| Г8 | Створювати нові навчальні плани | Освітня програма, рік створення (опціонально) | Новий навчальний план |
| Г9 | Переглядати план викладання дисциплін НП за семестрами | Критерій відбору (навчальний план) | Перелік дисциплін НП з вказівкою на семестр викладання |
| Г10 | Запланувати графік викладання дисципліни НП | Дисципліна НП, список семестрів викладання | Дисципліна НП запланована на відповідні семестри |
| Г11 | Перемістити дисципліну НП з одного семестру на інший | Дисципліна НП на конкретному семестрі, новий семестр | Дисципліна НП на новому семестрі |
| Г12 | Запланувати вид роботи на дисципліні НП | Семестрова дисципліна НП, вид роботи, кількість годин/факт наявності | Запланований вид роботи на дисципліні НП |
| Завідувач кафедри | | | |
| ЗК1 | Переглядати вакансії викладачів на кафедрі | Критерій відбору (кафедра) | Перелік викладачів на певних вакансіях |
| ЗК2 | Призначати осіб на посади викладачів конкретної кафедри | Кафедра, особа, посада на кафедрі, дата призначення (опціонально) | Призначення викладача на посаду на кафедрі |
| ЗК3 | Знімати осіб з посади викладачів конкретної кафедри | Конкретне призначення, дата зняття | Викладач з завершеним контрактом |

Продовження таблиці А.1

| Номер | Задача | Вхідні дані | Вихідні дані |
|---|---|---|---|
| ЗК4 | Переглядати штатний розклад викладачів | Критерій відбору (рік) | Перелік ставок викладачів за відповідний рік |
| ЗК5 | Формування штатного розкладу викладача | Викладач, ставка за 1-й семестр, ставка за 2-й семестр, рік дії | Ставка викладача на конкретний рік |
| ЗК6 | Редагування штатного розкладу викладача | Редагований викладач, нові: [ставка за 1-й семестр, ставка за 2-й семестр] | Оновлена ставка викладача на конкретний рік |
| Адміністратор ІС | | | |
| А1 | Реєструвати нові факультети | Назва, коротка назва та інтернаціоналізована назва факультету | Новий факультет |
| А2 | Реєструвати нові кафедри | Назва, коротка назва та інтернаціоналізована назва кафедри, відповідний факультет | Нова кафедра |
| А3 | Реєструвати нові посади | Назва посади, максимальне навантаження | Нова посада |
| А4 | Реєструвати профілі нових користувачів | Ім’я, прізвище, по-батькові, телефонний номер, електронна пошта | Новий профіль користувача |
| А5 | Призначати осіб на виконавчі позиції на кафедрі (Завідувач, Виконавча особа) | Тип позиції призначення, особа, кафедра, дата призначення | Особа на конкретній виконавчій позиції |
| А6 | Знімати осіб з виконавчих позицій на кафедрі | Конкретне призначення, дата зняття | Особа з завершеним контрактом на виконавчій позиції |
| А7 | Реєструвати нові галузі знань | Ієрархічний код, назва, інтернаціоналізована назва галузі знань | Нова галузь знань |

Продовження таблиці А.1

| Номер | Задача | Вхідні дані | Вихідні дані |
|---|---|---|---|
| А8 | Редагувати галузі знань | Редагована галузь знань, оновлені: [назва, інтернаціоналізована назва] | Оновлена галузь знань |
| А9 | Реєструвати нові міжнародні галузі знань | Ієрархічний код, назва міжнародної галузі знань | Нова міжнародна галузь знань |
| А10 | Редагувати міжнародні галузі знань | Редагована міжнародна галузь знань, оновлені: [назва] | Оновлена міжнародна галузь знань |

# ДОДАТОК Б

## Опис сутностей предметної області

Таблиця Б.1 – Опис сутностей ПрО підсистеми навчального відділу

| Ім’я атрибута | Призначення атрибута | Обмеження |
|---|---|---|
| FieldOfKnowledges (Галузь знань) | | |
| id | ідентифікатор галузі знань | первинний ключ |
| code | показник ієрархії галузі знань | унікальне, не порожнє |
| name | назва галузі знань | унікальне, не порожнє |
| int_name | інтернаціоналізована назва галузі знань | унікальне, порожнє за замовченням |
| IntFieldOfKnowledges (Міжнародна галузь знань) | | |
| id | ідентифікатор міжнародної галузі знань | первинний ключ |
| code | показник ієрархії міжнародної галузі знань | унікальне, не порожнє |
| name | назва галузі знань | унікальне, не порожнє |
| Specialities (Спеціальність) | | |
| id | ідентифікатор спеціальності | первинний ключ |
| fok_id | ідентифікатор галузі знань | не порожнє, зовнішній ключ для зв'язку з сутністю *field_of_knowledges* |
| code | показник ієрархії галузі знань | унікальне, не порожнє |
| name | назва спеціальності | унікальне, не порожнє |
| int_name | інтернаціоналізована назва спеціальності | унікальне, порожнє за замовченням |
| | | унікальна комбінація (fok_id, code) |
| InfokSpecialities (Міжнародна галузь знань → Спеціальність) | | |
| infok_id | ідентифікатор міжнародної галузі знань | не порожнє, зовнішній ключ для зв'язку з сутністю *int_field_of_knowledges* |
| speciality_id | ідентифікатор спеціальності | не порожнє, зовнішній ключ для зв'язку з сутністю *specialities* |

Продовження таблиці Б.1

| Ім’я атрибута | Призначення атрибута | Обмеження |
|---|---|---|
| | | унікальна комбінація (infok_id, speciality_id) |
| Faculty (Факультет) | | |
| id | ідентифікатор факультету | первинний ключ |
| name | назва факультету | унікальне, не порожнє |
| short_name | коротка назва факультету | унікальне, не порожнє |
| int_name | інтернаціоналізована назва факультету | унікальне, порожнє за замовченням |
| Departments (Кафедра) | | |
| id | ідентифікатор кафедри | первинний ключ |
| faculty_id | ідентифікатор факультету | не порожнє, зовнішній ключ для зв'язку з сутністю *faculties* |
| name | назва кафедри | унікальне, не порожнє |
| short_name | коротка назва кафедри | унікальне, не порожнє |
| int_name | інтернаціоналізована назва кафедри | унікальне, порожнє за замовченням |
| Positions (Посада) | | |
| id | ідентифікатор посади | первинний ключ |
| name | назва посади | унікальне, не порожнє |
| workload | доступне навантаження на посаду | не порожнє, *workload* > 0 |
| Users (Користувач / Персона) | | |
| id | ідентифікатор персони | первинний ключ |
| first_name | ім’я персони | не порожнє |
| last_name | прізвище персони | не порожнє |
| middle_name | по-батькові персони | не порожнє |
| phone_number | робочий номер телефону | унікальне, не порожнє, обмежене шаблоном (*^[0-9]{8,20}$*) |
| email | робоча пошта викладача | унікальне, не порожнє, обмежене шаблоном (*^[A-Za-z0-9._%\-]+@[A-Za-z0-9]+[.][A-Za-z]+$*) |

Продовження таблиці Б.1

| Ім’я атрибута | Призначення атрибута | Обмеження |
|---|---|---|
| username | логін користувача у системі | унікальне, не порожнє |
| DepartmentAssignments (Назначення на кафедрі) | | |
| id | ідентифікатор назначення | первинний ключ |
| role_type | тип виконавчої позиції (Завідувач кафедри, Виконавча особа) | не порожнє, належить множині (1, 2) |
| user_id | ідентифікатор персони | не порожнє, зовнішній ключ для зв'язку з сутністю *users* |
| department_id | ідентифікатор кафедри | не порожнє, зовнішній ключ для зв'язку з сутністю *departments* |
| assigned_at | дата призначення на виконавчу посаду | не порожнє, за замовчуванням *current_date* |
| unassigned_at | дата зняття з виконавчої посади | порожнє за замовченням |
| Educators (Викладач) | | |
| id | ідентифікатор викладача | первинний ключ |
| department_id | ідентифікатор кафедри | не порожнє, зовнішній ключ для зв'язку з сутністю *departments* |
| user_id | ідентифікатор персони | не порожнє, зовнішній ключ для зв'язку з сутністю *users* |
| position_id | ідентифікатор посади | не порожнє, зовнішній ключ для зв'язку з сутністю *positions* |
| position_type | тип посади (штатна, сумісна, почасова) | не порожнє, належить множині (1, 2, 3) |
| assigned_at | дата призначення на викладацьку посаду | не порожнє, за замовчуванням *current_date* |
| unassigned_at | дата зняття з виконавчої посади | порожнє за замовченням |
| Rates (Ставка) | | |
| id | ідентифікатор ставки | первинний ключ |
| educator_id | ідентифікатор викладача | не порожнє, зовнішній ключ для зв'язку з сутністю *educators* |

Продовження таблиці Б.1

| Ім’я атрибута | Призначення атрибута | Обмеження |
|---|---|---|
| rate_1 | ставка на 1-й семестр | не порожнє, в межах від 0 до 1 |
| rate_2 | ставка на 2-й семестр | не порожнє, в межах від 0 до 1 |
| year | дата встановлення ставки | не порожнє, замовчуванням *[year] з [current_date]* |
| | | унікальна комбінація (educator_id, year) |
| FacultySpecialities (Спеціальність → Факультет) | | |
| id | ідентифікатор спеціальності на факультеті | первинний ключ |
| speciality_id | ідентифікатор спеціальності | не порожнє, зовнішній ключ для зв'язку з сутністю *specialities* |
| faculty_id | ідентифікатор факультету | не порожнє, зовнішній ключ для зв'язку з сутністю *faculties* |
| | | унікальна комбінація (faculty_id, speciality_id) |
| EducationalPrograms (Освітня програма (ОП)) | | |
| id | ідентифікатор ОП | первинний ключ |
| fasp_id | ідентифікатор спеціальності на факультеті | не порожнє, зовнішній ключ для зв'язку з сутністю *faculty_specialities* |
| guarantor_id | ідентифікатор гаранта | не порожнє, зовнішній ключ для зв'язку з сутністю *users* |
| ed_level | освітній рівень | не порожнє, належить множині (1, 2, 3) |
| year | рік створення освітньої програми | не порожнє, замовчуванням *[year] з [current_date]* |
| | | унікальна комбінація (fasp_id, ed_level, year) |
| PlannedDisciplines (Запланована дисципліна) | | |
| id | ідентифікатор запланованої дисципліни | первинний ключ |
| edpr_id | ідентифікатор ОП | не порожнє, зовнішній ключ для зв'язку з сутністю *educational_programs* |

Продовження таблиці Б.1

| Ім’я атрибута | Призначення атрибута | Обмеження |
|---|---|---|
| discipline_id | ідентифікатор дисципліни | не порожнє, зовнішній ключ для зв'язку з сутністю *disciplines*, перевірка: дисципліна у зв’язку повинна бути навчального типу (*educational: true*) |
| discipline_type | тип дисципліни (Загальноосвітня, Спеціальна) | не порожнє, належить множині (1, 2) |
| component_type | тип освітньої компоненти (Обов’язкова, Вибірна) | не порожнє, належить множині (1, 2) |
| | | унікальна комбінація (edpr_id, discipline_id) |
| BlockedDisciplines (Заблокована дисципліна) | | |
| id | ідентифікатор блокування | первинний ключ |
| blockable_id | ідентифікатор планованої дисципліни, що блокується | не порожнє, зовнішній ключ для зв'язку з сутністю *planned_disciplines* |
| blocker_id | ідентифікатор планованої дисципліни, що блокує | не порожнє, зовнішній ключ для зв'язку з сутністю *planned_disciplines* |
| | | перевірка blockable_id <> blocker_id |
| | | унікальна комбінація (blockable_id, blocker_id) |
| EducationalPlans (Навчальний план (НП)) | | |
| id | ідентифікатор НП | первинний ключ |
| edpr_id | ідентифікатор освітньої програми | не порожнє, зовнішній ключ для зв'язку з сутністю *educational_programs* |
| year | рік створення навчального плану | не порожнє, за замовчуванням *[year] з [current_date]* |
| | | унікальна комбінація (edpr_id, year) |

Продовження таблиці Б.1

| Ім’я атрибута | Призначення атрибута | Обмеження |
|---|---|---|
| EducationalDisciplines (Дисципліна НП) | | |
| id | ідентифікатор дисципліни НП | первинний ключ |
| edpl_id | ідентифікатор НП | не порожнє, зовнішній ключ для зв'язку з сутністю *educational_plans* |
| pd_id | ідентифікатор запланованої дисципліни | не порожнє, зовнішній ключ для зв'язку з сутністю *planned_disciplines* |
| | | унікальна комбінація (edpl_id, pd_id) |
| EdPerSemesters (Семестрова дисципліна в НП) | | |
| id | ідентифікатор семестрової дисципліни НП | первинний ключ |
| ed_id | ідентифікатор дисципліни НП | не порожнє, зовнішній ключ для зв'язку з сутністю *educational_disciplines* |
| semester | семестр прочитання дисципліни | не порожнє, *semester* > 0 |
| credits | кількість кредитів | не порожнє, *credits* >= 0, за замовченням 0. |
| | | унікальна комбінація (ed_id, semester) |
| EdWorktypes (Вид роботи на дисципліні НП) | | |
| id | ідентифікатор виду роботи на дисципліні НП | первинний ключ |
| wt_id | ідентифікатор виду робот | не порожнє, зовнішній ключ для зв'язку з сутністю *worktypes*, перевірка: вид роботи у зв’язку повинен належати навчальному плану (*scope: навчальний план*) |
| quantity | *кількість годин* (для аудиторних робіт) / *факт існування* (для екзаменаційних та курсових робіт) | не порожнє, quantity > 0 |

Продовження таблиці Б.1

| Ім’я атрибута | Призначення атрибута | Обмеження |
|---|---|---|
| edps_id | ідентифікатор семестрової дисципліни НП | не порожнє, зовнішній ключ для зв'язку з сутністю *ed_per_semesters* |
| | | унікальна комбінація (edps_id, wt_id) |
| NormativeGroups (Нормативна група) | | |
| id | ідентифікатор нормативної групи | первинний ключ |
| wt_id | ідентифікатор виду роботи | не порожнє, зовнішній ключ для зв'язку з сутністю *worktypes* |
| name | назва нормативної групи | унікальне, не порожнє |
| Normative (Норматив) | | |
| id | ідентифікатор нормативу | первинний ключ |
| ng_id | ідентифікатор нормативної групи | не порожнє, зовнішній ключ для зв'язку з сутністю *normative_groups* |
| expression | вираз розрахунку нормативу | не порожнє |
| created_at | дата створення нормативу | не порожнє, за замовчуванням current_timestamp |
| SemesterDuration (Тривалість семестру) | | |
| id | ідентифікатор семестрової тривалості | первинний ключ |
| semester | номер семестру | не порожнє, semester > 0 |
| weeks | кількість тижнів у семестрі | не порожнє, weeks > 0 |
| ed_level | освітній рівень, для якого визначений термін | не порожнє, належить множині (1, 2, 3) |
| year | рік прийняття постанову про тривалість семестру | не порожнє, замовчуванням *[year] з [current_date]* |
| | | унікальна комбінація (semester, ed_level, year) |

# ДОДАТОК В

## Запити на створення таблиць бази даних

```
CREATE TABLE field_of_knowledges (
  id SERIAL PRIMARY KEY,
  code INT UNIQUE NOT NULL ,
  name VARCHAR UNIQUE NOT NULL,
  int_name VARCHAR UNIQUE NULL
);

CREATE TABLE specialities (
  id SERIAL PRIMARY KEY,
  fok_id INT NOT NULL
  REFERENCES field_of_knowledges(id) ON UPDATE CASCADE ON DELETE
CASCADE,
  code INT UNIQUE NOT NULL,
  name VARCHAR UNIQUE NOT NULL ,
  int_name VARCHAR UNIQUE NULL,

  -- На ту саму галузь знань не можна створити більше
  -- однієї спеціальності з тим самим шифром
  UNIQUE(fok_id, code)
);

CREATE TABLE int_field_of_knowledges (
  id SERIAL PRIMARY KEY,
  code INT NOT NULL UNIQUE,
  name VARCHAR NOT NULL UNIQUE
);

CREATE TABLE infok_specialities (
  id SERIAL PRIMARY KEY,
  infok_id INT NOT NULL
  REFERENCES int_field_of_knowledges(id) ON UPDATE CASCADE ON
DELETE CASCADE ,
  speciality_id INT NOT NULL
  REFERENCES specialities(id) ON UPDATE CASCADE ON DELETE CASCADE
,

  -- відповідність тієї самої міжнародної галузі знань
  -- та спеціальності можна задати лише один раз.
  UNIQUE (infok_id, speciality_id)
);

CREATE TABLE faculties (
  id SERIAL PRIMARY KEY,
  name VARCHAR UNIQUE NOT NULL,
  short_name VARCHAR UNIQUE NOT NULL,
  int_name VARCHAR UNIQUE NULL
);
```

```
CREATE TABLE departments (
  id SERIAL PRIMARY KEY,
  faculty_id INT NOT NULL
  REFERENCES faculties(id) ON UPDATE CASCADE ON DELETE CASCADE,
  name VARCHAR UNIQUE NOT NULL,
  short_name VARCHAR UNIQUE NOT NULL,
  int_name VARCHAR UNIQUE NULL
);

CREATE TABLE users (
  id SERIAL PRIMARY KEY,
  first_name VARCHAR NOT NULL,
  last_name VARCHAR NOT NULL,
  middle_name VARCHAR NOT NULL,
  phone_number VARCHAR UNIQUE NOT NULL CHECK (phone_number ~ '^[0-
9]{8,20}$'),
  email VARCHAR NOT NULL UNIQUE CHECK (email ~ '^[A-Za-z0-9._%\-
]+@[A-Za-z0-9]+[.][A-Za-z]+$'),
  username VARCHAR UNIQUE NOT NULL
);

CREATE TABLE positions (
  id SERIAL PRIMARY KEY,
  name VARCHAR UNIQUE NOT NULL,
  workload INT NOT NULL CHECK(workload > 0)
);

CREATE TABLE department_assignments (
  id SERIAL PRIMARY KEY,
  role_type  INT  NOT  NULL  CHECK(role_type  IN  (1,  2)),  --
[departhead, departexec]
  department_id INT NOT NULL
  REFERENCES departments(id) ON UPDATE CASCADE ON DELETE CASCADE,
  user_id INT NOT NULL
  REFERENCES users(id) ON UPDATE CASCADE ON DELETE CASCADE,
  assigned_at DATE NOT NULL DEFAULT current_date,
  unassigned_at DATE NULL
);

CREATE TABLE educators (
  id SERIAL PRIMARY KEY,
  department_id INT NOT NULL
  REFERENCES departments(id) ON UPDATE CASCADE ON DELETE CASCADE,
  user_id INT NOT NULL
  REFERENCES users(id) ON UPDATE CASCADE ON DELETE CASCADE,
  position_id INT NOT NULL
  REFERENCES positions(id) ON UPDATE CASCADE ON DELETE CASCADE,
  position_type INT NOT NULL CHECK(position_type IN (1, 2, 3)),
  -- [Основна, сумісник, почасова]
  assigned_at DATE NOT NULL DEFAULT current_date,
  unassigned_at DATE NULL
```

```
  -- Той самий викладач може займати БІЛЬШЕ однієї посади на
кафедрі.
  -- У випадку, коли необхідно відробляти частку ставки за іншу
вакансію.
);

CREATE TABLE rates (
  id SERIAL PRIMARY KEY,
  educator_id INT NOT NULL
  REFERENCES educators(id) ON UPDATE CASCADE ON DELETE CASCADE,
  rate_1 NUMERIC(5,2) NOT NULL CHECK(rate_1 BETWEEN 0 AND 1),
  rate_2 NUMERIC(5,2) NOT NULL CHECK(rate_2 BETWEEN 0 AND 1),
  year INT NOT NULL DEFAULT EXTRACT(YEAR FROM current_date),

    -- Розрахунок ставки на того самого викладача
    -- можуть бути встановлені ЛИШЕ один раз за рік.
  UNIQUE (educator_id, year)
);

CREATE TABLE faculty_specialities (
  id SERIAL PRIMARY KEY,
  faculty_id INT NOT NULL
  REFERENCES faculties(id) ON UPDATE CASCADE ON DELETE CASCADE,
  speciality_id INT NOT NULL
  REFERENCES specialities(id) ON UPDATE CASCADE ON DELETE CASCADE,

  -- та сама спеціальність може викладатись на факультеті лише
один раз
  -- (різні формати викладання на різних факультетах)
  UNIQUE (faculty_id, speciality_id)
);

CREATE TABLE educational_programs (
  id SERIAL PRIMARY KEY,
  fasp_id INT NOT NULL
  REFERENCES faculty_specialities(id) ON UPDATE CASCADE ON DELETE
CASCADE,
  guarantor_id INT NOT NULL
  REFERENCES users(id) ON UPDATE CASCADE ON DELETE CASCADE,
  ed_level  INT  NOT  NULL  CHECK(ed_level  IN  (1,  2,  3)),  --
(Бакалаврат, Магістратура, Апірантура)
  year INT NOT NULL DEFAULT EXTRACT(YEAR FROM current_date),

  -- освітня програма на ту саму спеціальність на факультеті
  -- (за тим самим рівнем освітньої програми)
  -- може створюватись НЕ БІЛЬШЕ РАЗУ за РІК
  UNIQUE (fasp_id, ed_level, year)
);

CREATE TABLE educational_plans (
  id SERIAL PRIMARY KEY,
  edpr_id INT NOT NULL
```

```
  REFERENCES educational_programs(id) ON UPDATE CASCADE ON DELETE
CASCADE,
  year INT NOT NULL DEFAULT EXTRACT(YEAR FROM current_date),
  UNIQUE(edpr_id, year)
);

CREATE TABLE planned_disciplines (
  id SERIAL PRIMARY KEY,
  discipline_type INT NOT NULL CHECK(discipline_type IN (1, 2)),
  component_type INT NOT NULL CHECK(component_type IN (1, 2)),
  edpr_id INT NOT NULL
  REFERENCES educational_programs(id) ON UPDATE CASCADE ON DELETE
CASCADE,
  discipline_id INT NOT NULL
  REFERENCES disciplines(id) ON UPDATE CASCADE ON DELETE CASCADE
  CHECK( check_edu_discipline(discipline_id, true) ),
  UNIQUE (edpr_id, discipline_id)
);

CREATE TABLE blocked_disciplines (
  id SERIAL PRIMARY KEY,
  blockable_id INT NOT NULL
  REFERENCES planned_disciplines(id) ON UPDATE CASCADE ON DELETE
CASCADE,
  blocker_id INT NOT NULL
  REFERENCES planned_disciplines(id) ON UPDATE CASCADE ON DELETE
CASCADE,

  -- дисципліна не може блокувати сама себе
  CHECK (blockable_id <> blocker_id),
  -- відповідність блокуючого та блокаваної дисципліни можна
вказати
  -- ЛИШЕ один раз для тих самих дисциплін з осв. програми.
  UNIQUE (blockable_id, blocker_id)
);

CREATE TABLE educational_disciplines (
  id SERIAL PRIMARY KEY,
  edpl_id INT NOT NULL
  REFERENCES educational_plans(id) ON UPDATE CASCADE ON DELETE
CASCADE,
  pd_id INT NOT NULL
  REFERENCES planned_disciplines(id) ON UPDATE CASCADE ON DELETE
CASCADE,
  UNIQUE (edpl_id, pd_id)
);

CREATE TABLE ed_per_semesters (
  id SERIAL PRIMARY KEY,
  ed_id INT NOT NULL
  REFERENCES educational_disciplines(id) ON UPDATE CASCADE ON
DELETE CASCADE,
  semester INT NOT NULL CHECK(semester > 0),
```

```
  credits NUMERIC(5,2) NOT NULL DEFAULT 0 CHECK(credits >= 0),
  -- частка "дисципліни" на семестр
  -- може бути задана лише ОДИН раз.
  UNIQUE(ed_id, semester)
);

CREATE TABLE ed_worktypes (
  id SERIAL PRIMARY KEY,
  edps_id INT NOT NULL
  REFERENCES ed_per_semesters(id) ON UPDATE CASCADE ON DELETE
CASCADE,
  wt_id INT NOT NULL
  REFERENCES worktypes(id) ON UPDATE CASCADE ON DELETE CASCADE
  CHECK( check_worktype_scope(wt_id, 1) ),
  quantity NUMERIC(5,2) NOT NULL CHECK(quantity > 0),

  -- той самий "(навчальний) вид роботи" на "дисципліні"
  -- може бути виділено лише ОДИН раз.
  UNIQUE(edps_id, wt_id)
);

CREATE TABLE normative_groups(
  id SERIAL PRIMARY KEY,
  name VARCHAR UNIQUE NOT NULL,

  wt_id INT NOT NULL
  REFERENCES worktypes(id) ON UPDATE CASCADE ON DELETE CASCADE
  CHECK( check_worktype_scope(wt_id, 3) )
);

CREATE TABLE normatives(
  id SERIAL PRIMARY KEY,
  created_at TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
  expression VARCHAR NOT NULL,

  ng_id INT NOT NULL
  REFERENCES normative_groups(id) ON UPDATE CASCADE ON DELETE
CASCADE
);

CREATE TABLE semester_durations (
  id SERIAL PRIMARY KEY,
  semester INT NOT NULL CHECK(semester > 0),
  weeks INT NOT NULL CHECK(weeks > 0),
  ed_level INT NOT NULL CHECK(ed_level IN (1, 2, 3)),
  year INT NOT NULL DEFAULT EXTRACT(YEAR FROM current_date),

  UNIQUE(semester, ed_level, year)
);
```

# ДОДАТОК Г

## Запити на створення компонентів збереження цілісності

Функції збереження цілісності (*використовуються в операторах CHECK на відповідних таблицях*):

```
-- ##########################################################
--  ФУНКЦІЯ: Обмежити створення записів, що не відповідає конкретному типу дисципліни (field: educational)
-- (для 'planned_discipline': educational=true)
-- ##########################################################
CREATE OR REPLACE FUNCTION check_edu_discipline(_pd_id int, _educational BOOLEAN)
RETURNS BOOLEAN
AS $BODY$
BEGIN
  IF (SELECT educational
    FROM educational_disciplines AS ed
    JOIN planned_disciplines AS pd ON pd.id = ed.pd_id
    JOIN disciplines AS d ON d.id = pd.discipline_id
    WHERE pd.id = _pd_id ) = _educational THEN
    RAISE EXCEPTION 'Дисципліна не є відповідного типу';
    RETURN false;
  END IF;
  RETURN true;
END;
$BODY$ LANGUAGE plpgsql;

-- ##########################################################
--  ФУНКЦІЯ: Обмежити створення записів, що не відповідає конкретній області виборки (field: scope)
-- (для 'ed_worktypes': scope=1 [In Model: 'EducationalPlan"])
-- ##########################################################
CREATE OR REPLACE FUNCTION check_worktype_scope(wt_id int, scope int)
RETURNS BOOLEAN
AS $BODY$
BEGIN
  IF (SELECT w.scope FROM worktypes w WHERE w.id = wt_id ) != scope THEN
    RAISE EXCEPTION 'Ви не можете пов`язати цей вид роботи';
    RETURN false;
  END IF;
  RETURN true;
END;
$BODY$ LANGUAGE plpgsql;
```

Тригери збереження цілісності:

```
-- ########################################################
-- ТРИГЕР: Тригер обмеження ієрархії для відповідності коду "галузі освіти" до коду "спеціальності".
-- ОПИС: Кожний запис спеціальності мусить мати код, що починається кодом освітньої галузі.
-- ########################################################
CREATE FUNCTION check_speciality_code() RETURNS TRIGGER AS $$
DECLARE
  fok_code TEXT;
BEGIN
  -- отримуємо код галузі запису "field_of_knowledge", що відповідає зовнішньому ключу нового запису в "specialities"
  SELECT code INTO fok_code FROM field_of_knowledges WHERE id = NEW.fok_id;

  -- перевіряємо, що код з "speciality" починається з коду з "field_of_knowledge"
  IF NOT NEW.code::TEXT LIKE fok_code || '%' THEN
    RAISE EXCEPTION 'Код спеціальності (code: %) мусить починатись з коду області знань (fok_code: %)!',
          NEW.code, fok_code;
  END IF;
  RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER tr_check_speciality_code
BEFORE INSERT ON specialities FOR EACH ROW
EXECUTE FUNCTION check_speciality_code();

-- ########################################################
-- ТРИГЕР: Обмеження максимальної ставки для особи-викладача.
-- ОПИС: Сума ставок особи-викладача за всіма посадами за рік
-- не має перевищувати MAX_RATE=1.5.
-- ########################################################
CREATE OR REPLACE FUNCTION check_rate_sum()
RETURNS TRIGGER AS $$
DECLARE
    current_user_id INTEGER;
    sum_1_sem NUMERIC;
    sum_2_sem NUMERIC;
    rate_limit CONSTANT NUMERIC := 1.5;
    -- Константа для лимита суми ставок
BEGIN
    -- Отримуємо user_id для даного educator_id
    SELECT e.user_id
    INTO current_user_id
    FROM educators AS e
    WHERE e.id = NEW.educator_id;
    -- Розраховуємо суму ставок для 1 семестру для даного user_id в поточному році
```

```
    SELECT COALESCE(SUM(r.rate_1), 0)
    INTO sum_1_sem
    FROM rates AS r
    JOIN educators AS e ON r.educator_id = e.id
    WHERE e.user_id = current_user_id AND r.year = NEW.year;

    -- Розраховуємо суму ставок для 2 семестру для даного user_id
в поточному році
    SELECT COALESCE(SUM(r.rate_2), 0)
    INTO sum_2_sem
    FROM rates AS r
    JOIN educators AS e ON r.educator_id = e.id
    WHERE e.user_id = current_user_id
      AND r.year = NEW.year;

    -- Перевіряємо, чи не перевищує сума ставок для 1 семестру або
2 семестру ліміт rate_limit
    IF sum_1_sem > rate_limit THEN
        RAISE EXCEPTION 'Сума ставок для 1 семестру для даного
користувача перевищує ліміт %. [Поточна сума: %. Залишок: %. Рік:
%]', rate_limit, sum_1_sem, rate_limit - sum_1_sem, NEW.year;
    END IF;

    IF sum_2_sem > rate_limit THEN
        RAISE EXCEPTION 'Сума ставок для 2 семестру для даного
користувача перевищує ліміт %. [Поточна сума: %. Залишок: %. Рік:
%]', rate_limit, sum_2_sem, rate_limit - sum_2_sem, NEW.year;
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER tr_check_rate_sum
AFTER INSERT OR UPDATE ON rate FOR EACH ROW
EXECUTE FUNCTION check_rate_sum();

--#########################################################
--ТРИГЕР: Сгенерувати та встановити ім'я користувача перед
створенням запису
--#########################################################
CREATE OR REPLACE FUNCTION users_generate_username()
RETURNS TRIGGE AS $$
BEGIN
  NEW.username        :=        generate_username(NEW.last_name,
NEW.first_name, NEW.id);
  RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER tr_users_generate_username
BEFORE INSERT ON users FOR EACH ROW
EXECUTE FUNCTION users_generate_username();
```

```
--#########################################################
--ТРИГЕР: Сгенерувати користувача в базі даних після створення запису, якщо даний користувач ще не існує.
--#########################################################
CREATE OR REPLACE FUNCTION users_create_new_user() RETURNS TRIGGER SECURITY DEFINER AS $$
DECLARE
     default_password TEXT := 'password';
BEGIN
     IF EXISTS (
       SELECT FROM pg_catalog.pg_roles
       WHERE rolname = NEW.username)
       THEN RAISE NOTICE 'Role already exists!';
    ELSE
     EXECUTE 'CREATE ROLE "' || NEW.username || '" WITH LOGIN PASSWORD ''' || default_password || ''';';
    END IF;
     RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER tr_users_create_user
AFTER INSERT ON users
FOR EACH ROW
EXECUTE PROCEDURE users_create_new_user();

--#########################################################
--ТРИГЕР (1): міняти логін користувача БД після зміни 'імені'
--ТРИГЕР (2): міняти ім'я користувача може тільки 'адмін'.
--#########################################################
CREATE OR REPLACE FUNCTION prevent_username_update()
RETURNS TRIGGER AS $$
BEGIN
  IF (SELECT EXISTS (
    SELECT 1
    FROM get_user_roles(current_user::varchar) AS r
    WHERE r.role_name = 'admin_group'
  )) THEN
     EXECUTE 'ALTER ROLE "' || OLD.username || '" RENAME TO "' || NEW.username || '";';
    RETURN NEW;
  ELSE
    RAISE EXCEPTION 'Only users with admin privileges can update the username';
  END IF;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER tr_prevent_username_update
BEFORE UPDATE ON users
FOR EACH ROW
WHEN (OLD.username IS DISTINCT FROM NEW.username)
EXECUTE FUNCTION prevent_username_update();
```

# ДОДАТОК Д

## Запити на створення тригерів для вирішення задач

Тригер для автоматичного заповнення навчального плану дисциплінами освітньої програми після його створення створення. Визначення тригерної функції міститься у лістингу Д.1, а визначення тригера – у лістингу Д.2.

```
--#########################################################
--ТРИГЕР:         Автоматичне         заповнення         записів
"educational_disciplines" при створенні нового навчального плану
("educational_plan")
--ОПИС:    При    створенні    нового    запису    до    таблиці
"educational_disciplines"       вносити       дисципліни       з
"planned_disciplines"
--#########################################################
CREATE OR REPLACE FUNCTION create_educational_disciplines()
RETURNS TRIGGER AS $$
BEGIN
  INSERT INTO educational_disciplines (edpl_id, pd_id)
  SELECT NEW.id, pd.id
  FROM planned_disciplines AS pd
  WHERE pd.edpr_id = NEW.edpr_id;

  RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

Лістинг Д.1 – Код тригерної функції автозаповнення дисциплін навчального плану

```
CREATE TRIGGER tr_create_educational_disciplines
AFTER INSERT ON educational_plans
FOR EACH ROW
EXECUTE FUNCTION create_educational_disciplines();
```

Лістинг Д.2 – Визначення тригеру для автозаповнення дисциплін навчального плану

# ДОДАТОК Е

## Запити на створення функцій для вирішення задач

Допоміжна функція для отримання тривалості освітнього процесу у семестрах за значенням переліку освітньо-кваліфікаційного рівня (лістинг Е.1).

```
--#########################################################
--ФУНКЦІЯ: Отримувати розмір у семестрах для відповідних ОКР
--#########################################################
CREATE FUNCTION get_semesters_count(education_level INTEGER)
RETURNS INTEGER AS $$
DECLARE
     -- [TODO]: Перенести тривалості за відповідною ОКР
('ed_level') до таблиці 'Options'.
     bachelor_duration INTEGER := 8;
     master_duration   INTEGER := 3;
     phd_duration      INTEGER := 6;
     semesters_count   INTEGER;
BEGIN
  IF education_level = 1 THEN
    semesters_count := bachelor_duration;
  ELSIF education_level = 2 THEN
    semesters_count := master_duration;
  ELSIF education_level = 3 THEN
    semesters_count := phd_duration;
  ELSE
     semesters_count := 0;
  END IF;

  RETURN semesters_count;
END;
$$ LANGUAGE plpgsql;
```

Лістинг Е.1 – Код функції, що повертає тривалість у семестрах

Допоміжна функція для отримання списку групових ролей певного користувача системи (лістинг Е.2). Дана функція приймає на вхід ім’я користувача у БД та звертається до системних таблиць Postgres. В результаті функція повертає набір усіх доступних групових ролей користувача.

```
--#########################################################
--ФУНКЦІЯ: Повернути список груп користувача
--#########################################################
CREATE OR REPLACE FUNCTION get_user_roles(user_name VARCHAR)
RETURNS TABLE (role_name NAME) AS
$$
BEGIN
    RETURN QUERY SELECT r.rolname AS role_name
                 FROM pg_auth_members m
                 JOIN pg_roles r ON (m.roleid = r.oid)
                 JOIN pg_roles u ON (m.member = u.oid)
                 WHERE u.rolname = user_name;
END;
$$ LANGUAGE plpgsql;
```

Лістинг Е.2 – Код функції, що повертає список групових ролей користувача

Допоміжна функція для отримання транслітерованої версії тексту. Використовується для транслітерації імені користувача при реєстрації (лістинг Е.3).

```
--#########################################################
-- ФУНКЦІЯ: Транслітерувати ім’я особи
--#########################################################
CREATE OR REPLACE FUNCTION translate_to_latin(text TEXT)
RETURNS TEXT
AS $$
DECLARE
  -- UA-alphabet ['абвгдеєжзиіїйклмнопрстуфхцчшщьюя']
  mapping_table TEXT[] := ARRAY[
    'а=>a',  'б=>b',  'в=>v',  'г=>g',  'ґ=>g',  'д=>d',  'е=>e',
'є=>e', 'ж=>zh', 'з=>z',
    'и=>i',  'і=>i',  'ї=>i',  'й=>y',  'к=>k',  'л=>l',  'м=>m',
'н=>n', 'о=>o', 'п=>p',
    'р=>r',  'с=>s',  'т=>t',  'у=>u',  'ф=>f',  'х=>kh',  'ц=>ts',
'ч=>ch', 'ш=>sh',
    'щ=>shch', 'ь=>', 'ю=>yu', 'я=>ya'
  ];
  i           INTEGER;
  latin_text  TEXT := lower(text);
  char_pair   TEXT[];
  char_map    TEXT[];
BEGIN
```

Лістинг Е.3 – Код функції, що повертає транслітеровану версію тексту

```
FOR i IN 1 .. cardinality(mapping_table) LOOP
    char_pair := string_to_array(mapping_table[i], '=>');
    char_map  := string_to_array(char_pair[2], '');
    latin_text     :=     replace(latin_text,     char_pair[1],
coalesce(char_map[1],''));
    IF cardinality(char_map) > 1 THEN
      latin_text   :=   replace(latin_text,   char_pair[1]   ||
char_map[1], coalesce(char_map[2],''));
    END IF;
  END LOOP;

  RETURN latin_text;
END;
$$ LANGUAGE plpgsql;
```

Лістинг Е.3, лист 2

Допоміжна функція для генерації імені користувача. Функція приймає ім’я, прізвище та по-батькові особи (лістинг Е.4).

```
--#########################################################
--ФУНКЦІЯ:  Створити  'ім'я'  користувача  з  полей  ("ім'я",
"прізвище", "id").
--#########################################################
CREATE  OR  REPLACE  FUNCTION  generate_username(last_name  TEXT,
first_name TEXT, id INT)
RETURNS TEXT
AS $$
DECLARE
  username TEXT;
BEGIN
  username := translate_to_latin(last_name);
  username := username || '.' || translate_to_latin(first_name);
  username := username || '.' || id;
  RETURN username;
END;
$$ LANGUAGE plpgsql;
```

Лістинг Е.4 – Код функції, що повертає згенероване ім’я користувача

# ДОДАТОК Ж

## Приклади програмних компонентів інтерфейсу користувача

```
import { useCallback, useEffect, useMemo, useState } from "react";
import {
  Form,
  redirect,
  useActionData,
} from "react-router-dom";

import                        HSeparator                        from
"../../components/UI/HSeparator/HSeparator";
import ButtonBack from "../../components/UI/Button/ButtonBack";
import                      ButtonSubmit                        from
"../../components/UI/Button/ButtonSubmit";
import Panel from "../../components/UI/Panel/Panel";
import Menu from "../../components/UI/Menu/Menu";
import Select from "../../components/Form/Select/Select";
import Field from "../../components/Form/Field/Field";
import Input from "../../components/Form/Input/Input";
import {
  createNormativeGroup,
  updateNormativeGroup,
} from "../../utils/api/normatives";
import {
  createOptionList,
  createOption,
} from "../../utils/helpers/createOptionsHelper";
import                      FormErrorList                       from
"../../components/Form/FormErrorList/FormErrorList";
import { validateRequired } from "../../utils/helpers/formHelper";

const NormativeGroupForm = ({ method, normativeGroup, worktypes })
=> {
  const { name, wtId } = normativeGroup || {};
  const [selectedWorktype, setSelectedWorktype] = useState();
  const data = useActionData();

  const handleWorktypeChange = useCallback((data) => {
    setSelectedWorktype(data);
  }, []);

  const buildWorktype = useCallback(
    (obj) => ({ value: obj.id, label: obj.title }),
    []
  );
```

Лістинг Ж.1 – Типове використання функціонального компоненту в React

```
  useEffect(() => {
    const worktypeOption = createOption(
      worktypes.find((item) => item.id === wtId),
      buildWorktype
    );

    handleWorktypeChange(worktypeOption);
  }, []);

  const worktypeOptions = useMemo(
    () => createOptionList(worktypes, buildWorktype),
    [worktypes]
  );

  const isPatching = useMemo(() => method === "patch", [method]);

  return (
    <Form id="form" method={method}>
      <FormErrorList data={data} />
      <Panel>
        <HSeparator />
        <Field label="Вид навантаження*">
          <Select
            name="wtId"
            data={worktypeOptions}
            value={selectedWorktype}
            onChange={handleWorktypeChange}
            readOnly={isPatching}
          />
        </Field>
        <Field label="Нормативна група*">
          <Input
            name="name"
            type="text"
            placeholder="Введіть назву нормативної групи"
            defaultValue={name && name}
          />
        </Field>
        <HSeparator />
        <Menu>
          {/* Depending on the nesting level, you have to go to a
certain level */}
          <ButtonBack level={isPatching ? 2 : 1} />
          <ButtonSubmit />
        </Menu>
      </Panel>
    </Form>
  );
};
```

Лістинг Ж.1, лист 2

```
export default NormativeGroupForm;

export async function action({ request, params }) {
  const formData = await request.formData();
  const updates = Object.fromEntries(formData);

  const { name, wtId } = updates;

  const errors = [];

  if (!validateRequired(wtId)) {
    errors.push("Вид роботи має бути визначений.");
  }

  if (!validateRequired(name)) {
    errors.push("Ім'я не може бути порожнімені");
  }

  if (errors.length) {
    return { errors };
  } else {
    let response;

    if (request.method === "PATCH") {
      response = await updateNormativeGroup(params.groupId,
updates);
    } else {
      response = await createNormativeGroup(updates);
    }

    if (response.status === 422) {
      return response.data;
    }

    return redirect(`/normative-groups`);
  }
```

Лістинг Ж.1, лист 3

```
class Speciality < ApplicationRecord
  has_many :faculty_specialities
  has_many :infok_specialities, foreign_key: :speciality_id
  has_many :int_foks, through: :infok_specialities, source:
:int_fok
  belongs_to :fok, class_name: "FieldOfKnowledge", foreign_key:
:fok_id
```

Лістинг Ж.2 – Типове використання моделі в Ruby on Rails

```
  def title
    "[#{code.to_s.rjust(3, "0")}] #{name}"
  end

  def short_name
    keys = name.split(" ")
    abbreviation = keys.map { |w| w[0].capitalize }.join("")
  end

  scope :filter_by,
    ->(text) do
      where("name ILIKE :text OR code::text ILIKE :text OR
int_name ILIKE :text", text: "%#{text}%")
    end

  validates :code, :name, uniqueness: true, presence: true
end
```

Лістинг Ж.2, лист 2

```
class DisciplinesController < ApplicationController
  before_action :set_disciplines, only: [:index]
  before_action :set_discipline, only: [:show, :update, :delete]

  # GET: [/educational_program/:id]/disciplines
  def index
    if params[:educational].present? && params[:educational] !=
"null"
      @disciplines = @disciplines.filter_by(params[:educational])
    end

    render json: @disciplines
  end

  # GET: /disciplines/:id
  def show
    render json: @discipline
  end

  # POST: /disciplines
  def create
    @discipline = Discipline.new(discipline_params)

    if @discipline.save
      render json: @discipline
    else
```

Лістинг Ж.3 – Типове використання контролеру в Ruby on Rails

```
      render json: { error: "Failed to create discipline!" },
status: :unprocessable_entity
    end
  end

  # POST: /disciplines/:id
  def update
    if @discipline.update(discipline_params)
      render json: @discipline
    else
      render json: { error: "Failed to update discipline!" },
status: :unprocessable_entity
    end
  end

  # DELETE: /disciplines/:id
  def destroy
    @discipline.destroy

    render json: :no_head
  end

  private

  def set_disciplines
    if params[:educational_program_id]
      program                                                =
EducationalProgram.find(params[:educational_program_id])
      @disciplines           =           Discipline.where.not(id:
program.planned_disciplines.pluck(:discipline_id))
    else
      @disciplines = Discipline.all
    end
  end

  def set_discipline
    @discipline = Discipline.find(params[:id])
  end
end
```

Лістинг Ж.3, лист 2

## ДОДАТОК К

## Список привілеїв ролей на об’єкти БД підсистеми

Таблиця К.1 – Список привілеїв ролей на об’єкти БД

| Об’єкти БД | Ролі | | | |
|---|---|---|---|---|
| | Співробітник навчального відділу (educdepart) | Гарант (guarantor) | Завідувач кафедри (departhead) | Адміністратор ІС (admin) |
| Таблиці | | | | |
| FieldOfKnowledge | CRUD | R | Відсутні | CRUD |
| IntFieldOfKnowledge | CRUD | R | Відсутні | CRUD |
| InfokSpecialities | CRUD | R | Відсутні | CRUD |
| Specialities | CRUD | R | Відсутні | CRUD |
| FacultySpecialities | CRUD | R | Відсутні | CRUD |
| EducationalPrograms | RU | CRUD | Відсутні | CRUD |
| PlannedDiscipline | R | CRUD | Відсутні | CRUD |

Продовження таблиці К.1

| Об’єкти БД | Ролі | | | |
|---|---|---|---|---|
| | Співробітник навчального відділу (educdepart) | Гарант (guarantor) | Завідувач кафедри (departhead) | Адміністратор ІС (admin) |
| Disciplines | R | R | Відсутні | CRUD |
| BlockedDisciplines | R | CRUD | Відсутні | CRUD |
| EducationalPlans | RU | CRUD | Відсутні | CRUD |
| EducationalDisciplines | R | CRUD | Відсутні | CRUD |
| EdPerSemesters | R | CRUD | Відсутні | CRUD |
| EdWorktypes | R | CRUD | Відсутні | CRUD |
| SemesterDurations | CRUD | R | Відсутні | CRUD |
| Worktypes | R | R | Відсутні | CRUD |
| Discipline | R | R | Відсутні | |
| NormativeGroups | CRUD | R | Відсутні | CRUD |
| Normatives | CRUD | R | Відсутні | CRUD |
| Faculties | R | R | R | CRUD |

Продовження таблиці К.1

| Об’єкти БД | Ролі | | | |
|---|---|---|---|---|
| | Співробітник навчального відділу (`educdepart`) | Гарант (`guarantor`) | Завідувач кафедри (`departhead`) | Адміністратор ІС (`admin`) |
| `Departments` | R | R | R | CRUD |
| `Users` | R | R | R | CRUD |
| `Positions` | R | R | R | CRUD |
| `Educators` | R | R | CRUD | CRUD |
| `Rates` | R | R | CRUD | CRUD |
| `DepartmentAssignments` | Відсутні | Відсутні | R | CRUD |
| Функції | | | | |
| CheckEduDiscipline | Відсутні | E | Відсутні | E |
| CheckWorktypeScope | Відсутні | E | Відсутні | E |
| GetSemestersCount | E | E | Відсутні | E |
| GetUserRoles | E | E | E | E |
| TranslateToLatin | Відсутні | Відсутні | Відсутні | E |
| GenerateUsername | Відсутні | Відсутні | Відсутні | E |

# ДОДАТОК Л

## Створення ролей БД та призначення їм привілеїв

```
-- ###
-- №1: Співробітник навчального відділу (educdepart)
-- ###

-- Таблиці
GRANT INSERT, SELECT, UPDATE, DELETE ON field_of_knowledge TO
educdepart_group;
GRANT INSERT, SELECT, UPDATE, DELETE ON int_field_of_knowledge TO
educdepart_group;
GRANT INSERT, SELECT, UPDATE, DELETE ON infok_specialities TO
educdepart_group;
GRANT INSERT, SELECT, UPDATE, DELETE ON specialities TO
educdepart_group;
GRANT INSERT, SELECT, UPDATE, DELETE ON faculty_specialities TO
educdepart_group;
GRANT SELECT, UPDATE ON educational_programs TO educdepart_group;
GRANT SELECT ON planned_discipline TO educdepart_group;
GRANT SELECT ON disciplines TO educdepart_group;
GRANT SELECT ON blocked_disciplines TO educdepart_group;
GRANT SELECT, UPDATE ON educational_plans TO educdepart_group;
GRANT SELECT ON educational_disciplines TO educdepart_group;
GRANT SELECT ON ed_per_semesters TO educdepart_group;
GRANT SELECT ON ed_worktypes TO educdepart_group;
GRANT INSERT, SELECT, UPDATE, DELETE ON semester_durations TO
educdepart_group;
GRANT SELECT ON worktypes TO educdepart_group;
GRANT INSERT, SELECT, UPDATE, DELETE ON normative_groups TO
educdepart_group;
GRANT INSERT, SELECT, UPDATE, DELETE ON normatives TO
educdepart_group;
GRANT SELECT ON faculties TO educdepart_group;
GRANT SELECT ON departments TO educdepart_group;
GRANT SELECT ON users TO educdepart_group;
GRANT SELECT ON positions TO educdepart_group;
GRANT SELECT ON educators TO educdepart_group;
GRANT SELECT ON rates TO educdepart_group;

-- Функції
GRANT EXECUTE ON FUNCTION get_semesters_count TO educdepart_group;
GRANT EXECUTE ON FUNCTION get_user_roles TO educdepart_group;

-- ###
-- №2: Гарант спеціальності (guarantor)
-- ###
GRANT SELECT ON field_of_knowledge TO guarantor_group;
GRANT SELECT ON int_field_of_knowledge TO guarantor_group;
GRANT SELECT ON infok_specialities TO guarantor_group;
GRANT SELECT ON specialities TO guarantor_group;
```

```
GRANT SELECT ON faculty_specialities TO guarantor_group;
GRANT INSERT, SELECT, UPDATE, DELETE ON educational_programs TO
guarantor_group;
GRANT INSERT, SELECT, UPDATE, DELETE ON planned_discipline TO
guarantor_group;
GRANT SELECT ON disciplines TO guarantor_group;
GRANT INSERT, SELECT, UPDATE, DELETE ON blocked_disciplines TO
guarantor_group;
GRANT INSERT, SELECT, UPDATE, DELETE ON educational_plans TO
guarantor_group;
GRANT INSERT, SELECT, UPDATE, DELETE ON educational_disciplines TO
guarantor_group;
GRANT INSERT, SELECT, UPDATE, DELETE ON ed_per_semesters TO
guarantor_group;
GRANT INSERT, SELECT, UPDATE, DELETE ON ed_worktypes TO
guarantor_group;
GRANT SELECT ON semester_durations TO guarantor_group;
GRANT SELECT ON worktypes TO guarantor_group;
GRANT SELECT ON normative_groups TO guarantor_group;
GRANT SELECT ON normatives TO guarantor_group;
GRANT SELECT ON faculties TO guarantor_group;
GRANT SELECT ON departments TO guarantor_group;
GRANT SELECT ON users TO guarantor_group;
GRANT SELECT ON positions TO guarantor_group;
GRANT SELECT ON educators TO guarantor_group;
GRANT SELECT ON rates TO guarantor_group;

-- Функції
GRANT EXECUTE ON FUNCTION check_edu_discipline TO guarantor_group;
GRANT EXECUTE ON FUNCTION check_worktype_scope TO guarantor_group;
GRANT EXECUTE ON FUNCTION get_semesters_count TO guarantor_group;
GRANT EXECUTE ON FUNCTION get_user_roles TO guarantor_group;

-- ###
-- №3: Завідувач кафедри (departhead)
-- ###
GRANT SELECT ON departments TO departhead_group;
GRANT SELECT ON users TO departhead_group;
GRANT SELECT ON positions TO departhead_group;
GRANT INSERT, SELECT, UPDATE, DELETE ON educators TO
departhead_group;
GRANT INSERT, SELECT, UPDATE, DELETE ON rates TO departhead_group;
GRANT SELECT ON department_assignments TO departhead_group;

-- Функції
GRANT EXECUTE ON FUNCTION get_user_roles TO departhead_group;
```

# ДОДАТОК М

## Довідка про впровадження

Міністерство освіти і науки України
ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
імені І.І. Мечникова

**ФАКУЛЬТЕТ МАТЕМАТИКИ, ФІЗИКИ та ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**

**Україна, 65057, Одеса, вул. Дворянська 2**
**тел.: + 380 (48) 723-12-23**
**e-mail: fmpit@onu.edu.ua**
**https://onu.edu.ua/uk/structure/faculty/fmfit**

Ministry of Education and Science of Ukraine
ODESSA I.I. Mechnikov NATIONAL UNIVERSITY

**FACULTY of MATHEMATICS, PHYSICS and INFORMATION TECHNOLOGY**

**2, Dvoryanskaya str., 65057, Odessa. Ukraine**
**phones: + 380 (48) 723-12-23**
**e-mail: fmpit@onu.edu.ua**
**https://onu.edu.ua/uk/structure/faculty/fmfit**

# ДОВІДКА
## про впровадження

Інформаційна система організаційного управління навчальним навантаженням, розроблена студентами Одеського національного університету імені І.І. Мечникова Джиговим Дмитром Юрійовичем та Жаром Михайлом Юрійовичем під час виконання дипломної роботи бакалавра на кафедрі математичного забезпечення комп'ютерних систем, проходить виробничі випробування на кафедрах факультету математики, фізики та інформаційних технологій ОНУ імені І.І. Мечникова і планується до впровадження.

Декан факультету математики
фізики, та інформаційних технологій — Юрій НІЦУК

Зав. кафедри математичного
забезпечення комп'ютерних систем — Євгеній МАЛАХОВ