КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет радіофізики, електроніки та комп'ютерних систем

Кафедра комп'ютерної інженерії

Дипломна робота магістра

Модернізація інструментарію розподілу навчальних дисциплін в системі "Тритон"

виконала студентка II курсу ОР «Магістр»
спеціальності 123«Комп'ютерна інженерія»
ОП «Комп'ютерні системи та мережі»
Юлії СЛІПЧУК

_____

Науковий керівник:
к. т. н., асистент **Євген СЛЮСАР**

_____

До захисту допускаю:
Завідувач кафедрою, к. ф.-м. н.,
доцент **Юрій БОЙКО**

Ухвалено на засіданні кафедри "___" _____ 2022 р., протокол №___

Київ 2023

Реферат

Випускна кваліфікаційна робота магістра: 64 с., 31 рис., 25 джерел.

В ході виконання роботи була модернізовано систему автоматизації розподілу обліку навантаження для підрозділів університету Triton. Було реалізовано виведення поділу та об’єднання дисциплін, а також формування груп та підгруп зі студентів для певних типів занять з дисциплін з інтеграцією в систему triton.

Для імплементації було обрано технології, сумісні з вже наявними в системі для інтеграції, а саме такі як ASP.NET MVC, Entity Framework для взаємодії з системою управління базами даних та Bootstrap treeview для візуалізації.

Було запропоновано та реалізовано алгоритм утворення деревоподібної структури, створено нові моделі, а також їх зв’язки з існуючими моделями для імплементації логіки формування груп та підгруп студентів, а також реалізовано логіку для коректної роботи. Перелічений функціонал було інтегровано імплементацію в наявну систему Triton.

Ключові слова: система автоматизації розподілу обліку навантаження, деревоподібна структура, об’єктно-реляційне відображення, ASP.NET MVC, Entity Framework, MSSQL Server.

Зміст

Вступ

Управління робочими процесами у вищих учбових закладах є важливим аспектом на сьогоднішній день. При цьому такі задачі є досить об’ємними і трудомісткими, відповідно автоматизація таких процесів є важливим нововведенням, яке дозволить зекономити ресурси та упорядкувати і налагодити роботу. В цілому сучасні інформаційні технології дозволяють не тільки підвищити ефективність функціонування учбового закладу, але й, можливо, принципову змінити організаційну та функціональну структуру.

Одним зі завдань, що стоять перед навчальними відділами ВНЗ, які потенційно можна автоматизувати, є розрахунок навчального навантаження викладачів. На його основі якого формуються навчальні доручення, розклад занять та штатний склад закладу. Таке завдання є доволі трудомістким і затратним у випадку виконання людиною.

Метою даної роботи є додання нового функціоналу для системи автоматизації розподілу обліку навантаження для підрозділів університету Triton.

1. Теоретична частина

1.1 Система автоматизації розподілу і обліку навантаження для підрозділів університету

Вищий учбовий заклад є складною активною соціально-економічної системою. Управління такою системою представляє собою трудомісткий процес, потребуючий більших витрат ресурсів не тільки енергетичних, матеріальних, але і людських. Одна з основних завдань управління навчальним закладом полягає в оптимізації освітнього процесу за показниками, що характеризує цей процес.

Навчальне навантаження - це основа робочого часу викладача, встановлювана керівником (завідувачем кафедрою), виходячи з професійної компетентності викладача, кількості годин за навчальним планом та навчальних програм, забезпеченості кадрами, та інших особливостей установи.

Ефективне управління навчальним процесом у ВНЗ сьогодні потребує переходу на якісно нові технології роботи з даними, що відносяться до навчального процесу, засновані на використанні комп'ютерних мереж і баз даних. Як можливість вдосконалення організаційної та навчальної діяльності ВНЗ можна використовувати автоматизацію процесу управління навчальним закладом, а також його частин.

Одним зі завдань, що стоять перед навчальними відділами ВНЗ є розрахунок навчального навантаження на кафедрі. На його основі:

- Формуються навчальні доручення для викладачів
- Складається розклад навчальних занять
- Визначаються штати професорсько-викладацького складу ВНЗ

У той же час, при традиційних способах розрахунку цей процес, в силу своєї трудомісткості, не може бути повторений багаторазово протягом

навчального року і тому, через деякий час перестає відповідати реальному навчальному навантаженню. У зв'язку з цим виникає бажання автоматизувати процес побудови плануючої документації, одночасно виключивши дублювання введення інформації.

Систему автоматизації розподілу і обліку навантаження в основному створюють окремо під певну організацію, ВНЗ для досягнення цілей, які ставить кожен заклад окремо.

1.2 Існуючі рішення

Знайдено лише одне схоже рішення - deanoffice-backend для Черкаського державного технологічного університету (дана система має повну і чітку документацію), але оскільки вона була реалізована лише для співробітників даного університету, то доступу до неї немає. Також, варто зазначити, що подібні системи мають розроблятись для кожного навчального закладу окремо, оскільки кожен заклад має свою унікальну структуру, що зумовлює формування унікальних вимог. Посилання - https://github.com/chdtu-fitis/deanoffice-backend.

1.3 Структура Triton

1.3.1 Навчальні плани

Навчальний план – це те, що входить до навчальної програми. Для конкретної спеціальності є список предметів (з видами роботи для прикладу: лабораторні, практичні і кількістю годин), які потрібно засвоїти. Навчальний план вноситься в Triton лише після затвердження навчальної програми. Також з навчального плану можна згенерувати excel-документ. Для навчального плану можна переглянути історію змін, адже спершу він створюється як чорновий варіант, далі погоджується, вносяться певні корективи і в результаті отримуємо погоджений навчальний план

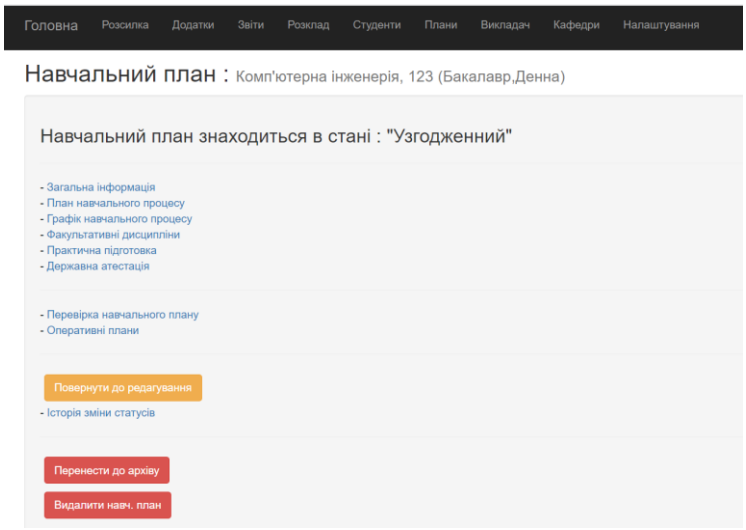

Рисунок 1 Приклад сторінки для навігації навчального плану

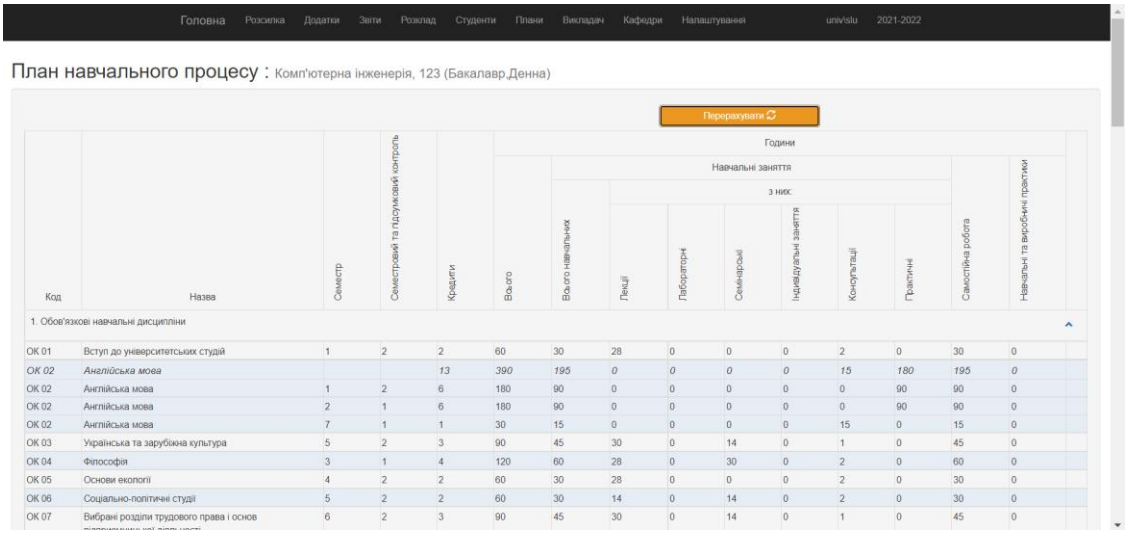

Рисунок 2 Приклад навчального плану

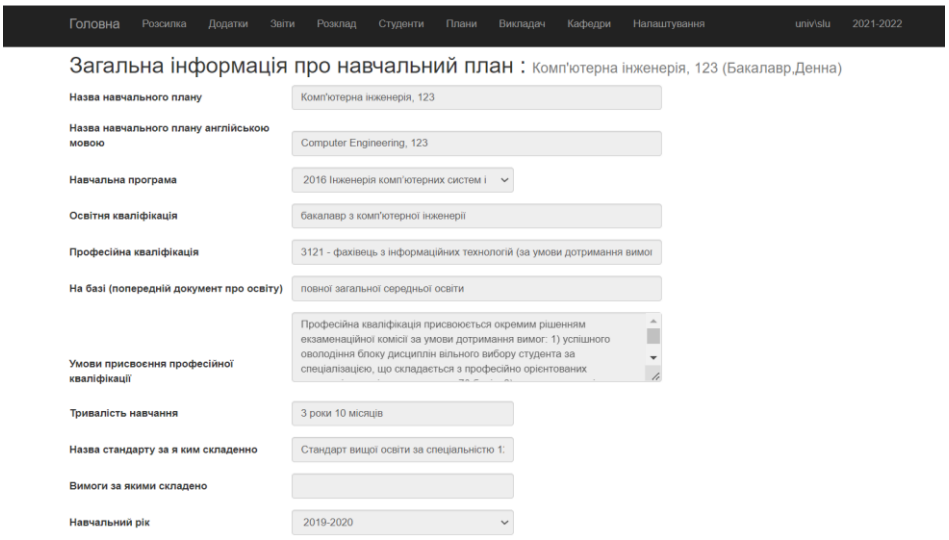

Рисунок 3 Загальна інформація про навчальний план

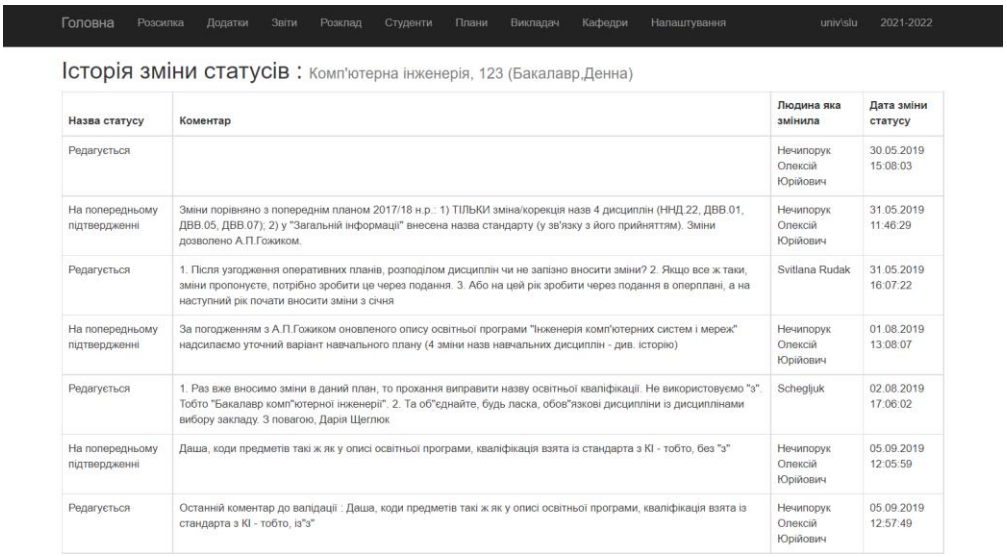

Рисунок 4 Історія зміни статусів навчального плану

1.3.2 Оперативні плани

Якщо відома інформація про те, скільки студентів буде навчатись на курсі – потрібно спланувати навантаження для виконання навчального плану. Тобто на даному етапі формується оперативний план для кожного курсу (план, скільки потрібно підгруп на лабораторні, практичні і по скільки годин вони мають бути, щоб вичитати кожному студенту певну кількість годин предмету). Так само, як і для навчальних планів, оперативні плани мають історію змін, яку можна подивитись в Triton.

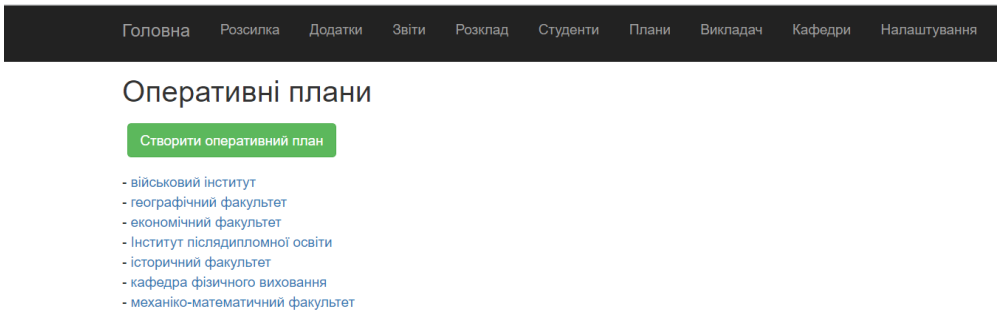

Рисунок 5 Список оперативних планів

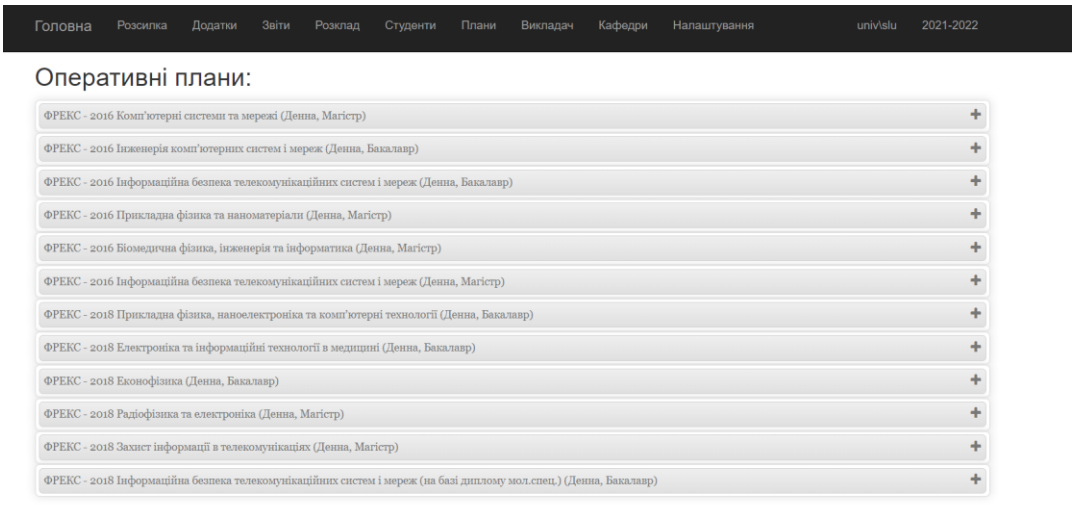

Рисунок 6 Приклад списку оперативних планів для Факультету радіофізики, електроніки та комп'ютерних систем

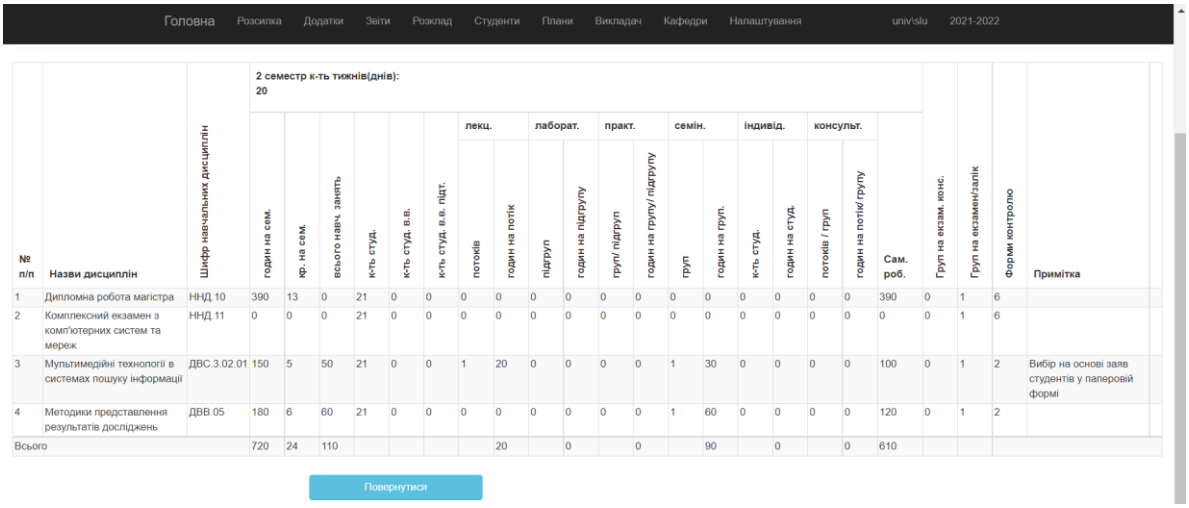

Рисунок 7 Приклад оперативноо плану

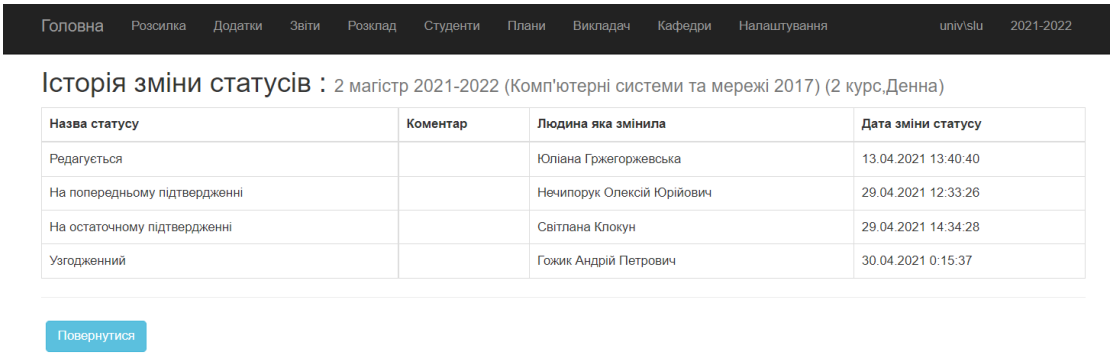

Рисунок 8 Приклад історії змін оперативного плану

1.3.3 Дисципліни

Дисципліни можна об’єднувати або ділити. Об’єднання можливе з іншою дисципліною (злиття з різних оперативних планів, наприклад, часто зливають англійську мову на природничих факультетах, оскільки вона вичитується для всіх однаково). Розділити можна двома методами – за годинами (наприклад, якщо дисципліна складає 120 год, то 60 год на лекції і 60 год на лабораторні (на кожного студента)) та за студентами (наприклад, коли потрібно розділити групу студентів на підгрупи для виконання лабораторних робіт). Поділ на об’єднання можна робити скільки завгодно разів. Кафедра бачить останні результати поділу, тобто найдрібніші частинки і вона в свою чергу також можу ділити або об’єднувати дисципліни. Після того, як зроблено поділ/об’єднання дисциплін, наступний крок – створення заявок.

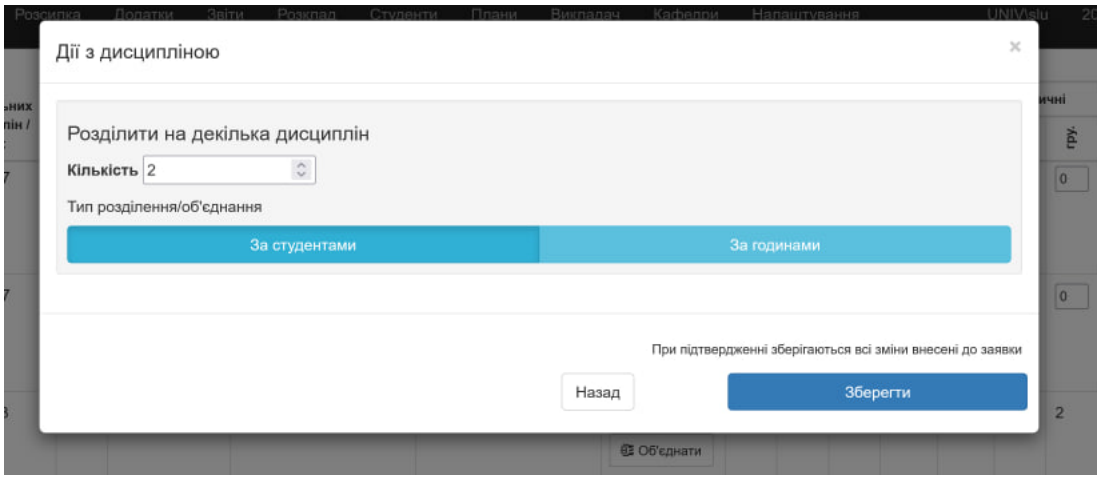

Рисунок 9 Спливаюче вікно для поділу/об’єднання дисциплін

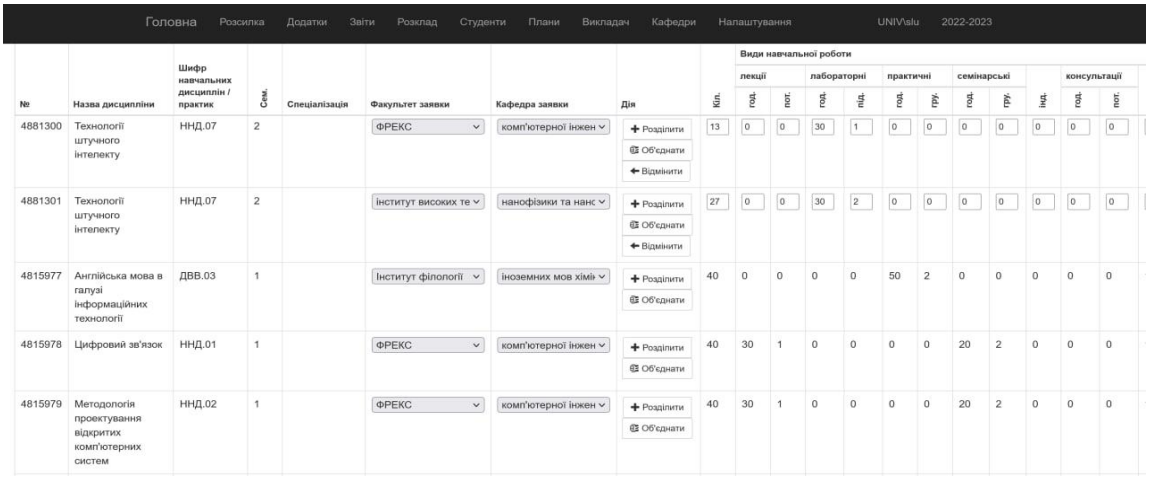

Рисунок 10 Редагування заявок оперативного плану (місце, де можна робити об'єднання та поділ дисциплін)

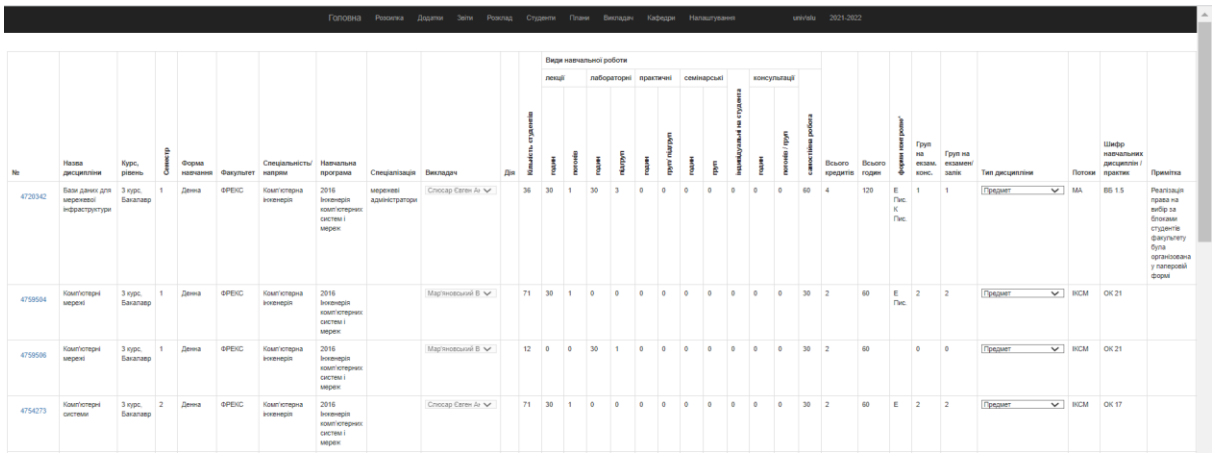

Рисунок 11 Навантаження кафедри

1.3.4 Заявки

Далі постає задача перед деканатом – вирішити, хто буде виконувати дане навантаження (англійська мова – кафедра іноземних мов, фізика – кафедра електрофізики). Можливий варіант, що частина предмету викладається на одній кафедрі, а інша частина – на іншій, для забезпечення цього робиться поділ. Нерозподілені дисципліни – ті дисципліни, які не призначенні жодній кафедрі, але вкінці формування навантаження, вони повинні бути розподілені. Кафедра може прийняти або не прийняти заявку. Після того, як кафедра прийняла певну множину заявок, то далі вона розподіляє їх між викладачами, що в свою чергу формує навантаження для викладачів.

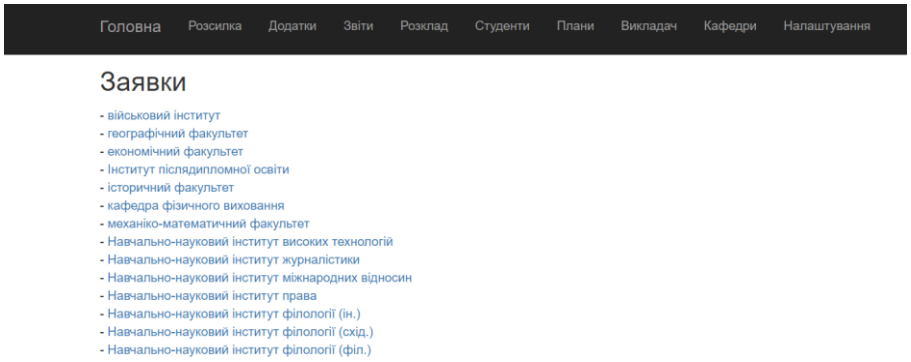

Рисунок 12 Список факультетів, які створюють заявки

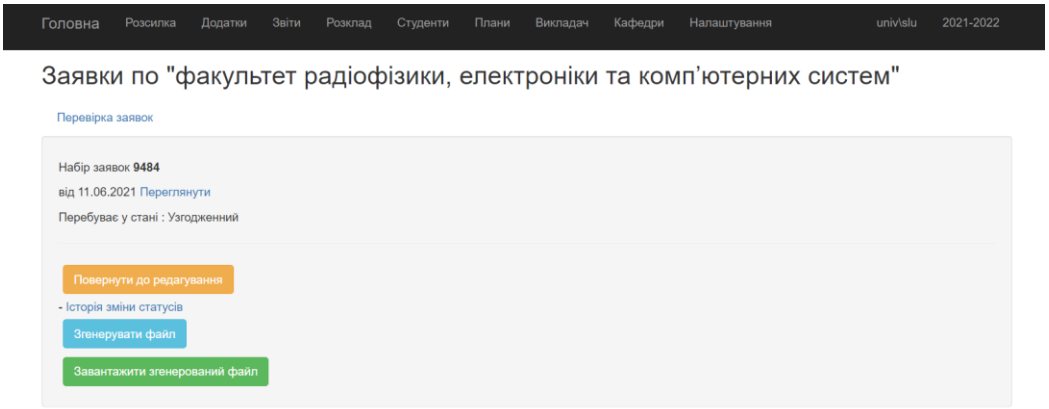

Рисунок 13 Заявки Факультету радіофізики, електроніки та комп'ютерних систем

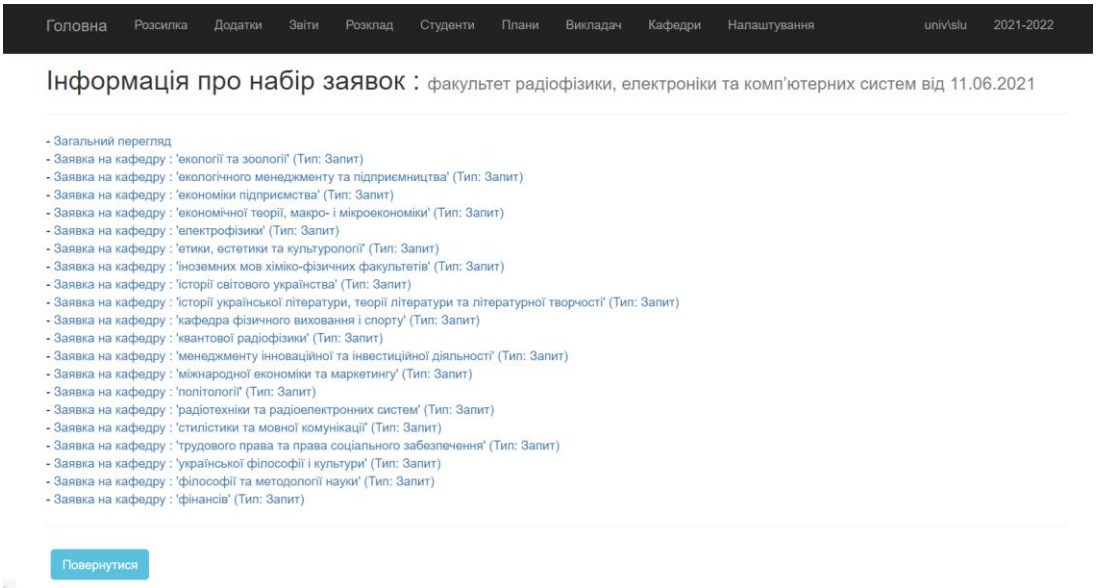

Рисунок 14 Інформація про набір заявок

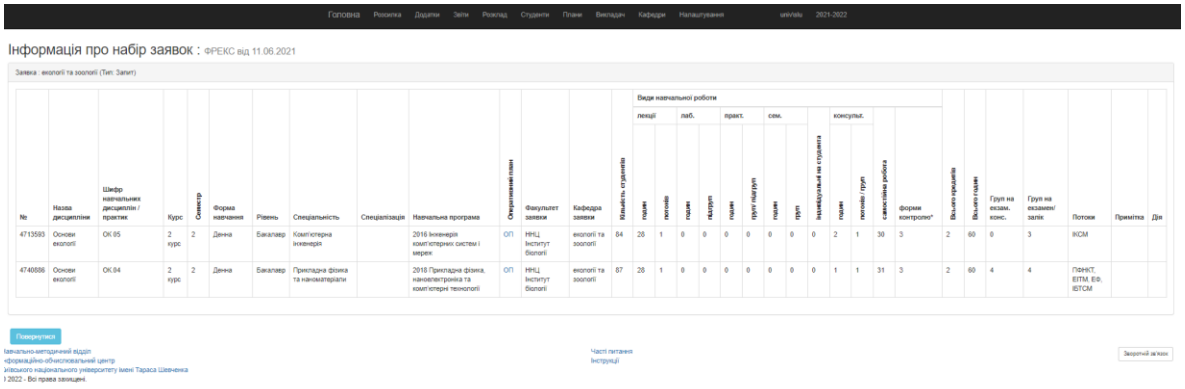

Рисунок 15 Приклад заявки

2. Огляд технологій

1.1 ASP.NET

ASP.NET - це платформа веб-додатків на стороні сервера з відкритим кодом, призначена для веб-розробки для створення динамічних веб-сторінок. Він був розроблений Microsoft, щоб дозволити програмістам створювати динамічні веб-сайти, програми та служби. Назва розшифровується як Active Server Pages Network Enabled Technologies.

ASP.NET розширює платформу .NET інструментами та бібліотеками, спеціально призначеними для створення веб-програм.

Ось деякі речі, які ASP.NET додає до платформи .NET:

- Базова структура для обробки веб-запитів у C# або F#
- Синтаксис шаблонів веб-сторінок, відомий як Razor, для створення динамічних веб-сторінок за допомогою C#
- Бібліотеки для загальних веб-шаблонів, як-от контролер представлення моделі (MVC)
- Система автентифікації, яка включає бібліотеки, базу даних і сторінки шаблонів для обробки входів, включаючи багатофакторну автентифікацію та зовнішню автентифікацію за допомогою Google, Twitter тощо.
- Розширення редактора для підсвічування синтаксису, доповнення коду та інших функцій спеціально для розробки веб-сторінок

2.1.1 ASP.NET - архітектура

Архітектура фреймворку ASP.Net базується на трьох компонентах: мова програмування, бібліотека та CLR.

Мова: .NET Framework підтримує різні мови для написання кодів. Можна використовувати C# або VB.net для розробки веб-додатків.

Бібліотека: структура .NET складається з набору бібліотек класів. System.Web - найпоширеніша бібліотека класів, яка використовується в розробці веб-додатків ASP.NET.

CLR: Common Language Runtime є основною частиною архітектури .NET. Воно виконує всі основні дії фреймворку, такі дії, як обробка винятків і збирання сміття.

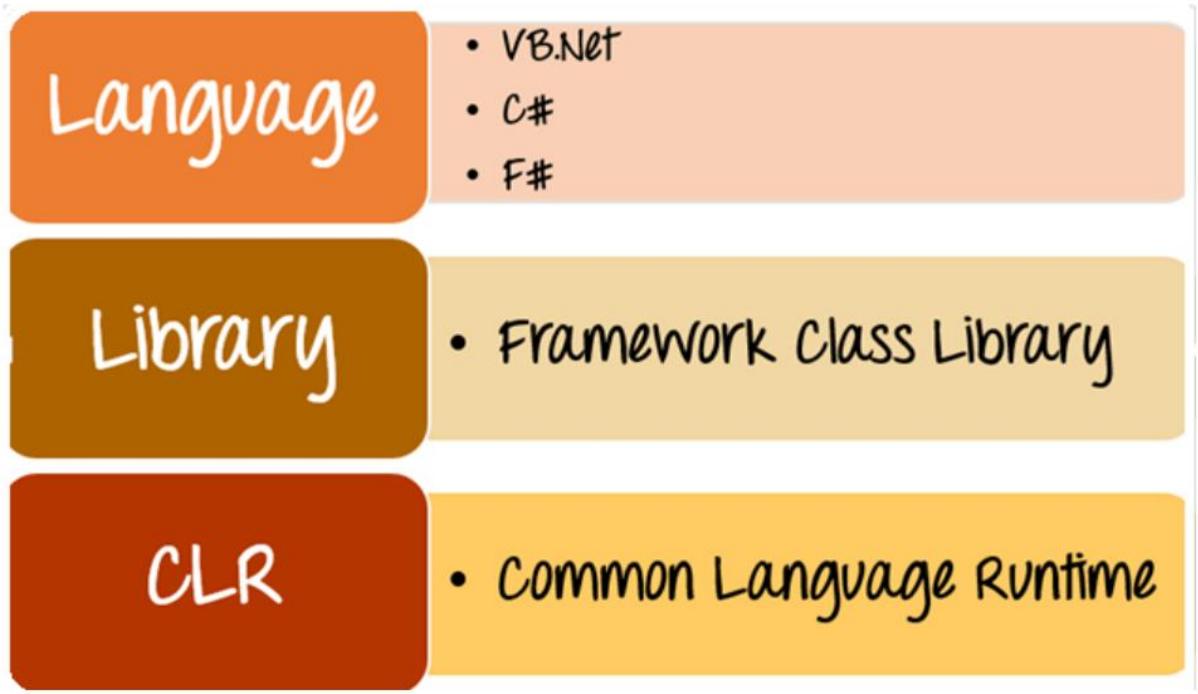

Рисунок 16Архітектура ASP.NET

2.1.2 ASP.NET - характеристики

Code Behind File: ця концепція розділяє дизайн і кодування. Завдяки цьому поділу додаток ASP.Net легко підтримувати. Загальний тип файлу ASP.Net – .aspx. Якщо є веб-сторінка під назвою MyWebPage.aspx, то створиться ще один файл під назвою MyWebPage.aspx.cs, який позначає кодову частину сторінки. Visual Studio IDE створює окремі файли для кожної веб-сторінки, один для дизайну та інший для коду.

State Management: забезпечує можливість контролювати управління станом. HTTP відомий як протокол без збереження стану. Розглянемо

приклад програми кошика для покупок. Тут, після прийняття рішення про придбання товарів, коли користувач нарешті захоче купити на сайті, він натисне кнопку «Надіслати». Додаток повинен запам'ятати всі товари, які користувач вибрав для покупки. Це відомо як запам'ятовування стану програми в певний момент часу. HTTP є протоколом без стану. Коли користувач переходить на сторінку покупки, HTTP не зберігатиме інформацію про товари в кошику. Необхідно виконати додаткове кодування, щоб гарантувати, що елементи кошика можна буде перенести на сторінку покупки.

Caching: реалізація концепції кешування. Кешування покращує продуктивність програми. За допомогою кешування часто запитувані користувачем сторінки можна зберігати в тимчасовому місці. Ці сторінки можна отримати швидше, а користувачеві можна швидко надіслати відповіді.

2.1.3 ASP.NET – життєвий цикл сторінки

Сторінка ASP.NET проходить життєвий цикл і виконує ряд етапів обробки.

Це:

- Ініціалізація (Initialization)
- Створення екземплярів та елементів керування (Instantiating controls)
- Відновлення та підтримка стану (Restoring and maintaining state)
- Запуск коду обробника подій (Running event handler code)
- Візуалізація (Rendering)

Запит на сторінку (Page Request)

Крок запиту сторінки відбувається перед початком життєвого циклу сторінки. Коли користувач запитує сторінку, ASP.NET вирішує, чи потрібно сторінку аналізувати та скомпілювати, чи у відповідь можна надіслати лише кешовану версію сторінки.

Старт (Start)

Властивості сторінки, такі як запит і відповідь, налаштовуються на початковому етапі. На цьому кроці сторінка також вирішує, чи є запит зворотним відправленням чи новим запитом, і встановлює властивості IsPostBack і UICulture.

Ініціалізація (Initialization)

На етапі ініціалізації всі елементи керування сторінкою доступні, і властивість UniqueID для кожного елемента керування встановлено. Головна сторінка та схеми також застосовуються до сторінки, якщо це можливо.

Завантаження (Load)

Під час завантаження сторінки, якщо поточний запит є зворотнім, то властивості керування завантажуються з інформацією, відновленою зі стану перегляну та стану керування

Обробка подій повернення (Postback event handling)

Обробники подій контролів викликаються лише якщо запит є зворотнім. Потім викликається метод Validate усіх елементів, який в результаті, встановлює властивість IsValid окремих елементів.

Візуалізація (Rendering)

Стан перегляду зберігається для сторінки та всіх елементів керування перед відтворенням. На етапі візуалізації сторінка викликає метод Render для кожного елемента керування, який записує свій результат в об’єкт OutputStream властивості Response сторінки

Розвантаження (Unload)

Подія Unload відбувається після того, як сторінка була повністю відтворена, надіслана клієнту та готова до видалення. У цей момент

властивості сторінки «Response» і «Request» тощо вивантажуються та виконується очищення.

2.2 ASP.NET MVC

ASP.NET MVC – це шаблон проектування, який використовується для відокремлення інтерфейсу користувача (подання), даних (модель) і логіки програми (контролер). Цей шаблон допомагає досягти поділу інтересів. Використовуючи шаблон MVC для веб-сайтів, запити направляються до контролера, який відповідає за роботу з моделлю для виконання дій та/або отримання даних. Контролер вибирає представлення для відображення та надає йому модель. Представлення відображає сторінку на основі даних у моделі. ASP.NET дає потужний, заснований на шаблонах спосіб створення динамічних веб-сайтів за допомогою шаблону MVC, який забезпечує чітке розділення.

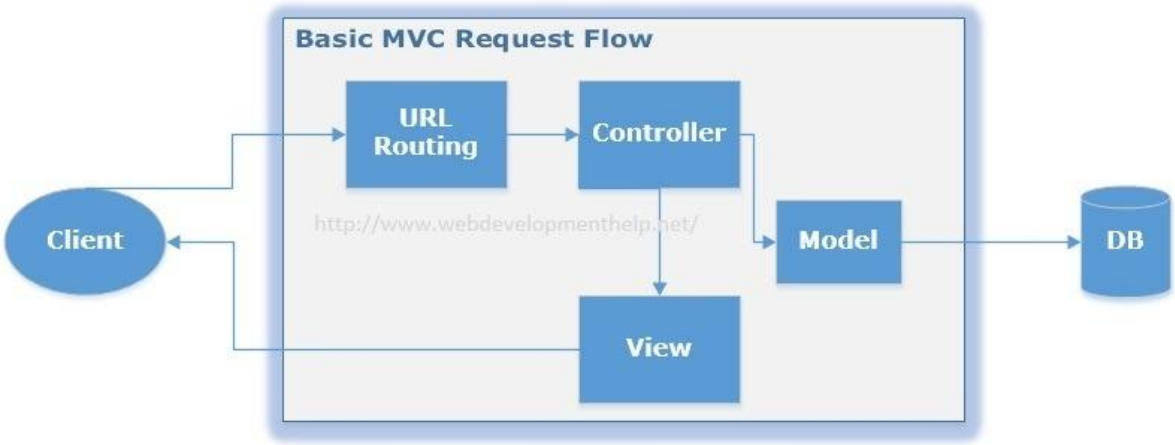

Рисунок 17 Базовий потік запитів MVC

2.2.1 Моделі та дані

Одним з ключевих компонентів патерна MVC є моделі, основна задача яких – опис структури даних, які використовуються. Як правило, всі сутності в додатку виділяються в окремі моделі, які описують структуру кожної сутності. Всі моделі оформлюються як звичайні POCO-класи (plain-old CLR objects), тобто звичайні класи C#. Модель не обов’язково повинна містити

лише властивості. Вона може мати конструктори, методи, поля. В ASP.NET MVC моделі можна поділити по застосуванню на кілька груп:

- Моделі, об’єкти яких зберігаються в спеціальних сховищах даних (напр, базах даних, файлах тощо)
- Моделі, які використовуються на поданнях, їх ще називають view models
- Допоміжні моделі для проміжних обчислень

2.2.2 Контролери

При отриманні запиту, система маршрутизації вибирає для обробки запиту потрібний контролер і передає йому дані запиту. Контролер обробляє ці дані і надсилає результат обробки. В основному, контролер має конструктор, через який за допомогою Dependency Injection передаються сервіси і хелпери. Ключевими елементами контролера є Actions (дії). Дії контролера – це публічні методи, які співставляються з запитами. Співставлення запиту з методом контролера відбувається завдяки маршрутизації. Не всі методи контролера є діями. Контролери можуть мати не публічні методи. Також контролери і методи контролерів можуть мати атрибути. Крім того, методи можуть обробляти різні типи запитів. Для вказання типу запиту HTTP, до методу використовують один (в окремих випадках не один) з атрибутів – [HttpGet], [HttpPost], [HttpPut], [HttpPatch], [HttpDelete], [HttpHead].

2.2.3 Представлення

Зазвичай при зверненні до веб-додатків користувач очікує отримати веб-сторінку з даними. В MVC для цього використовуються View (представлення), які визначають зовнішній вигляд додатку. Існує певна структура зберігання представлень в папці Views:

- Для кожного контролера створюється підкаталог в папці Views, який називається як контролер і зберігає представлення, які використовують методи даного контролера
- Також є папка Shared, яка зберігає спільні представлення для всіх контролерів (в основному це Master pages)

За роботу представлень відповідає об’єкт ViewResult. Він здійснює рендеринг подання на веб-сторінку і повертає її у вигляді відповіді клієнту. Щоб повернути об’єкт ViewResult, метод контролера викликає метод View.

2.3 Bootstrap treeview

Для виконання завдання було використано bootstrap treeview для виводу ієрархічної деревоподібної структури на веб-сторінці. Це досить зручне рішення в даному випадку, оскільки проект використовує bootstrap технологію.

2.4 Зберігання даних

2.4.1 Системи керування базами даних MSSQL

Microsoft SQL Server — це система керування реляційною базою даних (RDBMS), яка підтримує широкий спектр програм обробки транзакцій, бізнес-аналітики та аналітики в корпоративних IT-середовищах. Microsoft SQL Server є однією з трьох провідних на ринку технологій баз даних разом із Oracle Database і DB2 IBM. Як і інше програмне забезпечення RDBMS, Microsoft SQL Server побудовано на основі SQL, стандартизованої мови програмування, яку адміністратори баз даних (DBA) та інші IT-фахівці використовують для керування базами даних і запитів до даних, які вони містять. SQL Server пов’язаний з Transact-SQL (T-SQL), реалізацією SQL від Microsoft, яка додає набір власних розширень програмування до стандартної мови. Основним компонентом Microsoft SQL Server є SQL Server Database Engine, який контролює зберігання, обробку та безпеку даних. Він включає реляційний механізм, який обробляє команди та запити, і механізм зберігання, який керує файлами бази даних, таблицями, сторінками, індексами, буферами даних і транзакціями. Збережені процедури, тригери, подання та інші об’єкти бази даних також створюються та виконуються за допомогою Database Engine. Під обробником баз даних знаходиться операційна система SQL Server або SQLOS. SQLOS обробляє функції нижчого рівня, такі як керування пам’яттю та введенням/виведенням, планування завдань і блокування даних, щоб уникнути конфліктних оновлень. Рівень мережевого інтерфейсу розташований над Database Engine і використовує протокол потоку табличних даних Microsoft для полегшення взаємодії запитів і відповідей із серверами баз даних. А на рівні користувача адміністратори баз даних і розробники SQL Server пишуть оператори T-SQL для створення та модифікації структур бази даних, маніпулювання даними, впровадження засобів захисту та резервного копіювання баз даних, серед інших завдань.

2.4.2 SSDT

Microsoft SQL Server Data Tools (SSDT) — це рішення Visual Studio для розробки реляційних баз даних SQL Server. SSDT — це ширший термін, який включає більше, ніж просто нові засоби баз даних. По суті, це переупаковка продукту Business Intelligence Developer Studio (BIDS) із Visual Studio 2008. На додаток до нових інструментів баз даних, SSDT підтримує звичайні типи проектів BIDS для SQL Server Analysis Services (SSAS), Reporting Services (SSRS) і Служби інтеграції (SSIS). Завдяки SSDT Microsoft тепер об'єднала всі можливості розробки бази даних SQL Server в єдину версію Visual Studio. Інструменти даних SQL Server (SSDT) ефективно об'єднують кілька окремих функцій розробки, які раніше були присутні в окремих інструментах, в єдине інтегроване середовище програмування. Інструменти даних SQL Server ефективно об'єднують кілька окремих функцій розробки, які раніше були присутні в окремих інструментах, в одне інтегроване середовище програмування.

Особливості SSDT:

- Дизайн бази даних на основі декларативної схеми
- Реверсивне проектування баз даних
- Порівняння схем і даних
- Створення XML-звітів
- Рефакторинг коду

2.4.3 Entity Framework

Entity Framework – це структура Object Relational Mapping (ORM), яка пропонує розробникам автоматичний механізм для зберігання та доступу до даних у базі даних. Це основний засіб Microsoft для взаємодії між програмами .NET і реляційними базами даних.

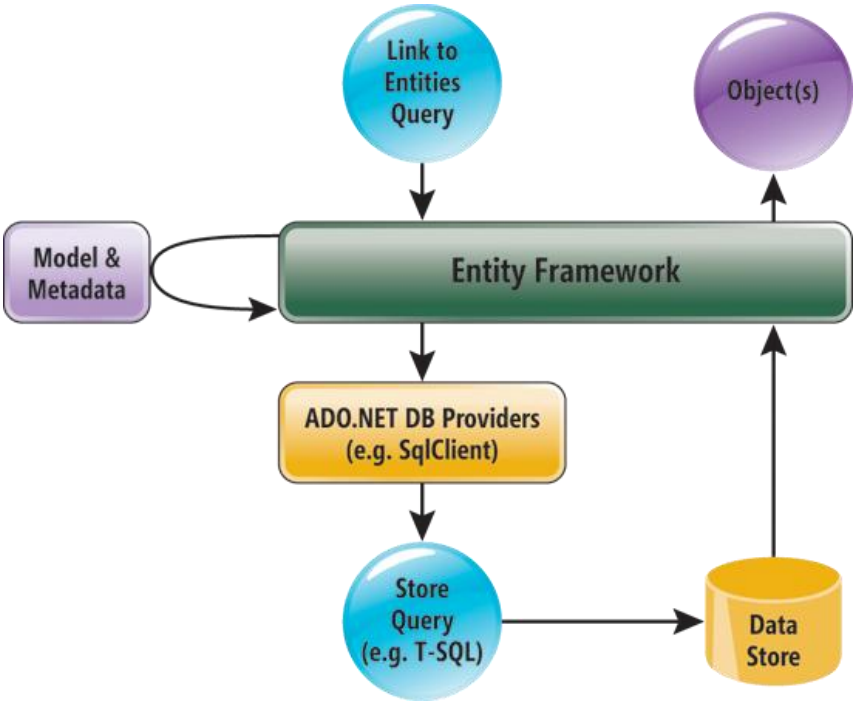

Рисунок 18 Entity Framework виконує запити та обробляє їх результати

Основні характеристики:

- EF може генерувати необхідні команди бази даних для читання або запису даних у базу даних і виконувати їх за вас
- Запити можна виражати за допомогою LINQ до об’єктів
- EF виконає відповідний запит до бази даних, а потім матеріалізує результати в екземпляри об’єктів, щоб з ними можна було працювати в коді
- Можливість відстеження змін об’єктів у пам’яті
- Має збережену підтримку процедур
- Два підходи використання: Database-First (використовуєть вже існуюча база даних, з таблиць якої генеруються моделі, а із зв’язків між таблицями генеруються зв’язки між моделяит) & Code-First (спершу створюються моделі (як представлення таблиць) і зв’язки між моделяит (як зв’язки між таблицями в базі даних))

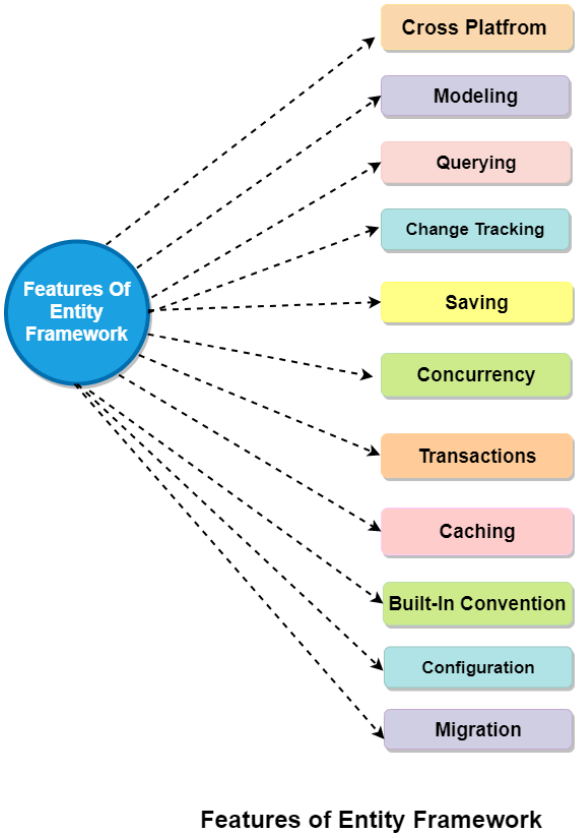

Рисунок 19 Особливості Entity Framework

2.4.4 LINQ

Language-Integrated Query (LINQ) — це назва набору технологій, заснованих на інтеграції можливостей запитів безпосередньо в мову C#. Традиційно запити до даних виражаються у вигляді простих рядків без перевірки типу під час компіляції або підтримки IntelliSense. У LINQ запит є першокласною мовною конструкцією, як і класи, методи, події. Запити пишуться до строго типізованих колекцій об’єктів, використовуючи ключові слова мови та знайомі оператори. Сімейство технологій LINQ забезпечує послідовне виконання запитів до:

- Об’єктів (LINQ to Objects)
- Реляційних баз даних (LINQ to SQL)
- XML (LINQ to XML)

3. Мета і формування технічних завдань

1. Формування і виведення ієрархічної деревоподібної структури поділу та об’єднання дисциплін
2. Імплементація нового функціоналу по створенню зв’язків між студентами та групами/підгрупами навчальних дисциплін
    a. Створення моделей груп та студентів в групі в системі Triton з використанням об’єктно-реляційного відображення Entity Framework
    b. Встановлення зв’язків між новостворенними та існуючими моделями об’єктно-реляційного відображення в системі Triton

4. Практична частина

4.1 Формування і виведення ієрархічної структури

4.1.1 Опис змін в коді та використання бази даних

Щоб виконати поставлене завдання, потрібно вносити зміни до двох проектів – це MainSite - ASP.NET MVC додаток, де знаходяться контролери, які обробляють запити, JS скрипти, CSS файли та представлення, які відмальовуються при поверненні відповіді з сервера та TritonModel - згенеровані моделі з бази даних за допомогою Entity Framework + сервіси + File Generators. В MainSite.

Також для виконання поставленого завдання було використано кілька таблиць бази даних, а саме:

- Main. OperationPlanDiscipline
- Main.OperationPlan
- Main.Discipline
- Main.Workload
- Main.WorkloadDiscipline
- Main.Syllabus
- Main.Cathedra
- Main.Faculty

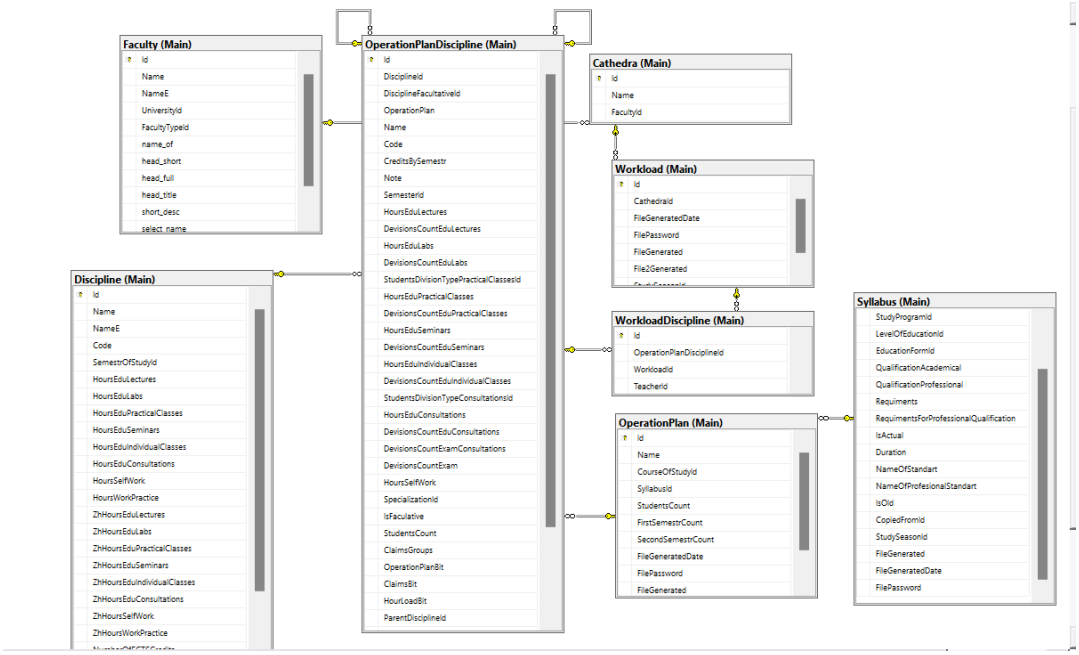

Рисунок 20 Діаграма зв'язків між таблицями

Для формування ієрархічної деревоподібної структури використовувалась таблиця Main.OperationPlanDiscipline. Поділи та об’єднання дисципліни зберігаються у Main.OperationPlanDiscipline. В даній таблиці є поля Id, ParentDisciplineId та ParentDiscipline2Id. Два останні поля посилаються на Id цієї ж таблиці. Розглянемо можливі випадки їх заповнення:

- Якщо ParentDisciplineId та ParentDiscipline2Id = NULL -> це первинна дисципліна навчального плану, її не ділили і не об’єднували
- Якщо ParentDisciplineId != NULL та ParentDiscipline2Id = NULL -> відбувся поділ, дана дисципліна має батьківську дисципліну
- Якщо ParentDisciplineId та ParentDiscipline2Id != NULL -> відбулось об’єдання, дана дисципліна є результатом об’єднання двох інших

В коді було додано метод WorkloadTreeForDiscipline в контролері WorkloadController, який приймає operationPlanDisciplineId та returnUrl і повертає ActionResult з моделлю WorkloadTree.

В сервісі WorkloadService додано кілька методів, які виконують одну задачу – створення ієрархічної деревоподібної структури для виведення її на сторінці.

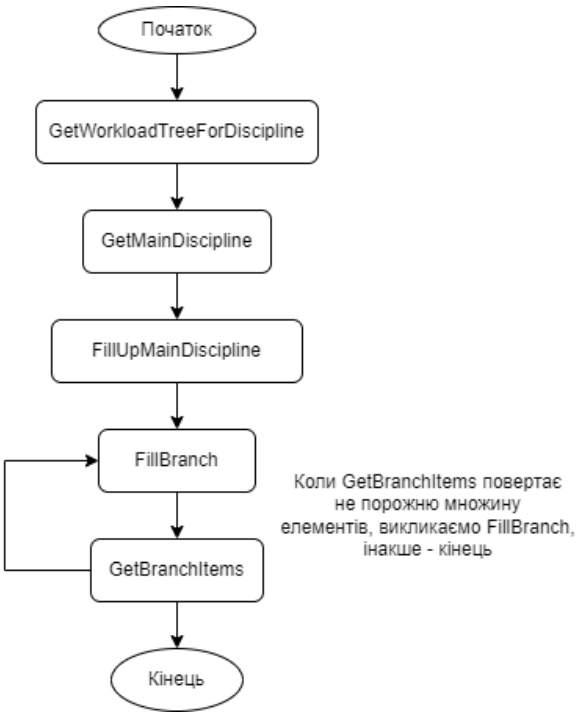

Рисунок 21 Скорочений стек методів для формування ієрархічної деревоподібної структури

2.1.1 Опис алгоритму для формування ієрархічної деревоподібної структури

Перший крок алгоритму полягає в знаходженні первинної дисципліни, тобто тієї, в якої ParentDisciplineId та ParentDiscipline2Id = NULL. Для цього перевіряємо, чи поточна дисципліна не первинна, і якщо ні, то поточній дисципліні присвоюємо батьківську і шукаємо первинну. Все це відбувається у циклі На другому кроці після знаходження первинної дисципліни, отримуємо її дочірні елементи, якщо вони є – присвоїти їх як дочірні вузли для первинної дисципліни в дереві, якщо ж дочірніх дисциплін немає – закінчення формування дерева. Далі робимо ці ж самі кроки для дочірніх дисциплін, допоки не дійдемо до кінця, коли дисципліни не матимуть дочірніх.

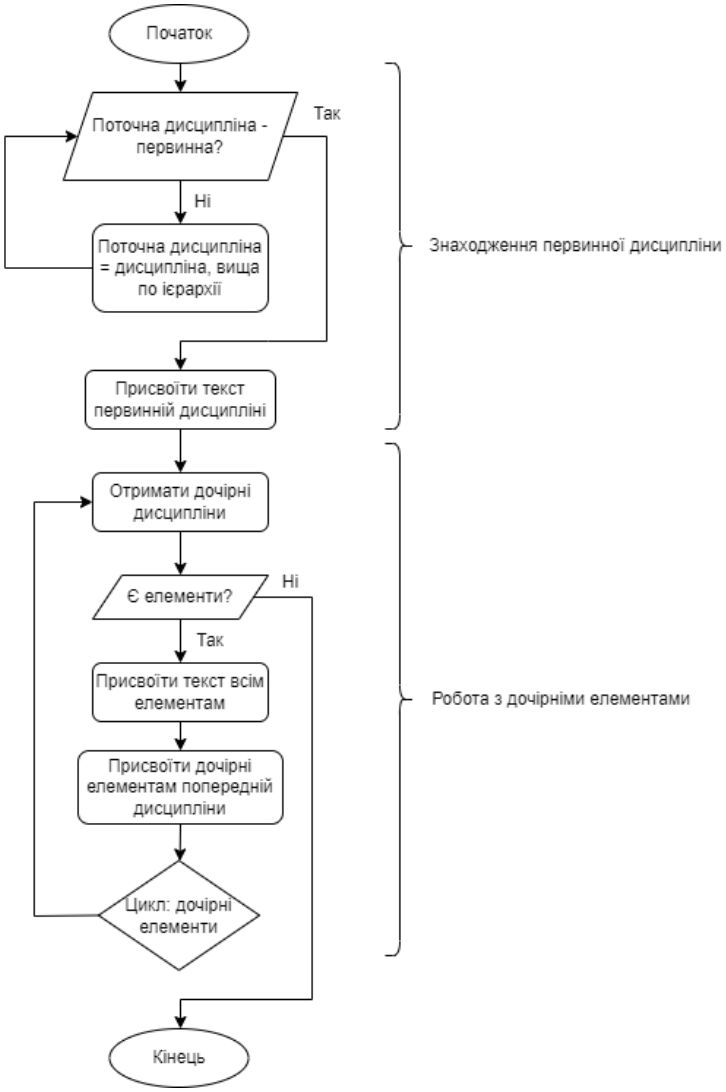

Рисунок 19 Алгоритм утворення деревоподібної структури

2.1.2 Вигляд сторінки з виведенням ієрархічної деревоподібної структури поділу та об’єднання дисципліни

Було створено окрему сторінку, на якій виводиться ієрархічна деревоподібна структура. Деякі дисципліни ділились в межах оперативного плану, деякі – вже на окремій кафедрі, тому вивід для кожного елемента відрізняється (виводимо оперативний план і факультет або кафедру). Та дисципліна, від якої перейшли на цю сторінку, підсвічується синім кольором. Якщо дисципліна була об’єднана, то в дужках буде про це вказано.

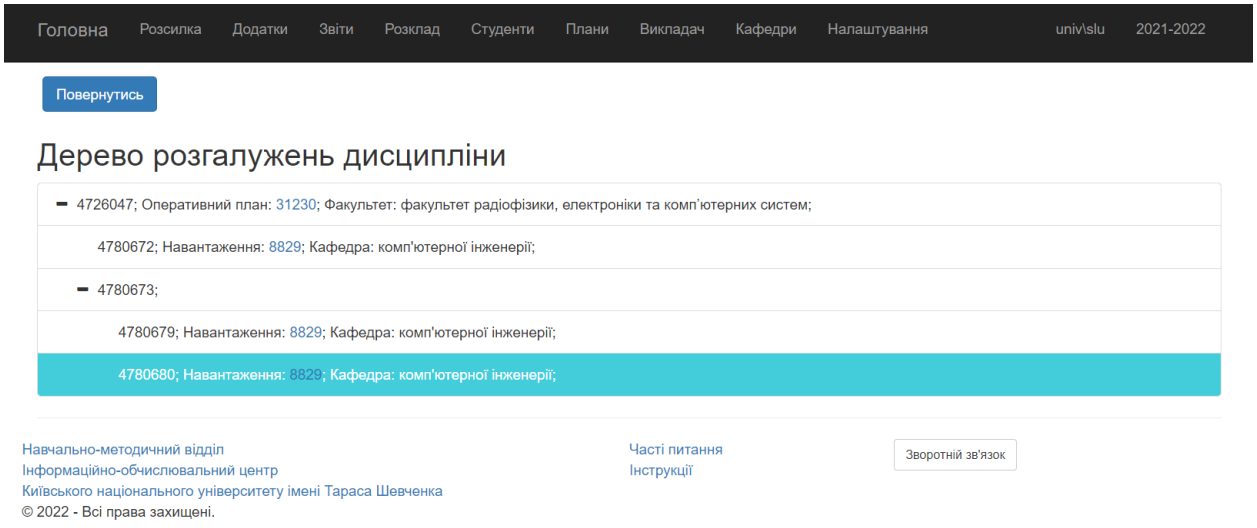

Рисунок 22 Вигляд сторінки з виведенням ієрархічної деревоподібної структури поділу/об'єднання дисципліни

4.2 Створення груп та підгруп

В Triton, який використовується Київським національним університетом імені Тараса Шевченка, на даний момент, відсутній функціонал для створення груп та підгруп для різних видів діяльності та призначення студентів на них, тому вирішено реалізувати цей функціонал.

4.2.1 Опис змін в БД - нові таблиці, зв’язки між ними

Для реалізації даного завдання було створено кілька додаткових таблиць:

- ActivityType (вид роботи, наповнена статичними даними, а саме Лекція, Семінар, Лабораторна робота, Практична робота, Індивідуальне заняття, Консультація, Консультація до іспиту, Іспит)

```
CREATE TABLE [Main].[ActivityType]
(
ID INT NOT NULL IDENTITY PRIMARY KEY,
ActivityName NVARCHAR(30) UNIQUE NOT NULL
)
```

- WorkloadDisciplineStudentGroup (таблиця, яка відображає групи та підгрупи студентів для певної дисципліни. Має зовнішні ключі на ActivityType (ActivityTypeID) та на WorkloadDiscipline (WorkloadDisciplineID))

```
CREATE TABLE Main.WorkloadDisciplineStudentsGroup
(
ID INT NOT NULL IDENTITY PRIMARY KEY,
GroupNumber INT NOT NULL,
WorkloadDisciplineID INT NOT NULL REFERENCES [Main].[WorkloadDiscipline](ID),
ActivityTypeID INT NOT NULL REFERENCES [Main].[ActivityType](ID),
MaxCountOfStudents INT NOT NULL
)
```

- WorkloadDisciplineStudent (таблиця, для відображення студентів в групі/підгрупі. Має зовнішні ключі на WorkloadDisciplineStudentGroup (WorkloadDisciplineStudentGroupID) та на Student (StudentID))

```
CREATE TABLE [Main].[WorkloadDisciplineStudent]
(
ID INT NOT NULL IDENTITY PRIMARY KEY,
StudentID INT NOT NULL REFERENCES [Students].[Student](ID),
WorkloadDisciplineStudentsGroupID INT NOT NULL REFERENCES
[Main].[WorkloadDisciplineStudentsGroup](ID)
)
```

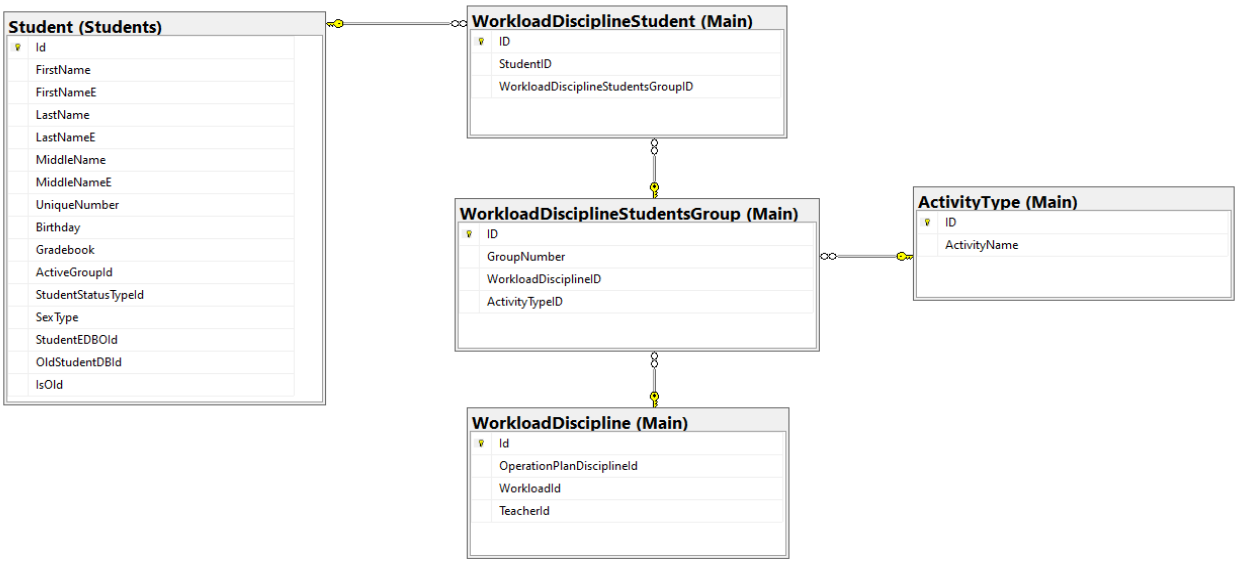

Рисунок 23 Діаграма відношень для новостворених моделей

4.2.2 Опис змін в коді

Для виведення інформації на сторінки, додатково було створено кілька моделей представлення (ViewModels), а саме:

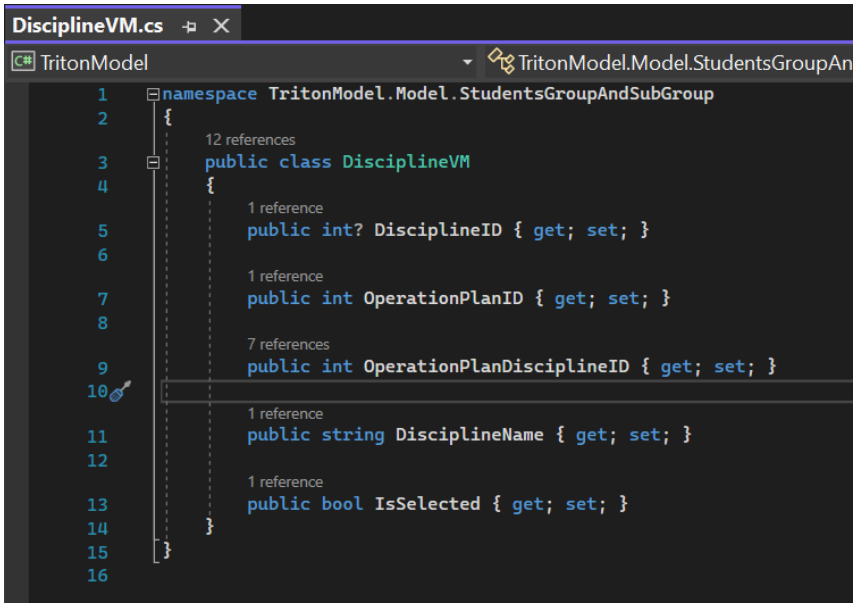

Рисунок 24 DisciplineVM

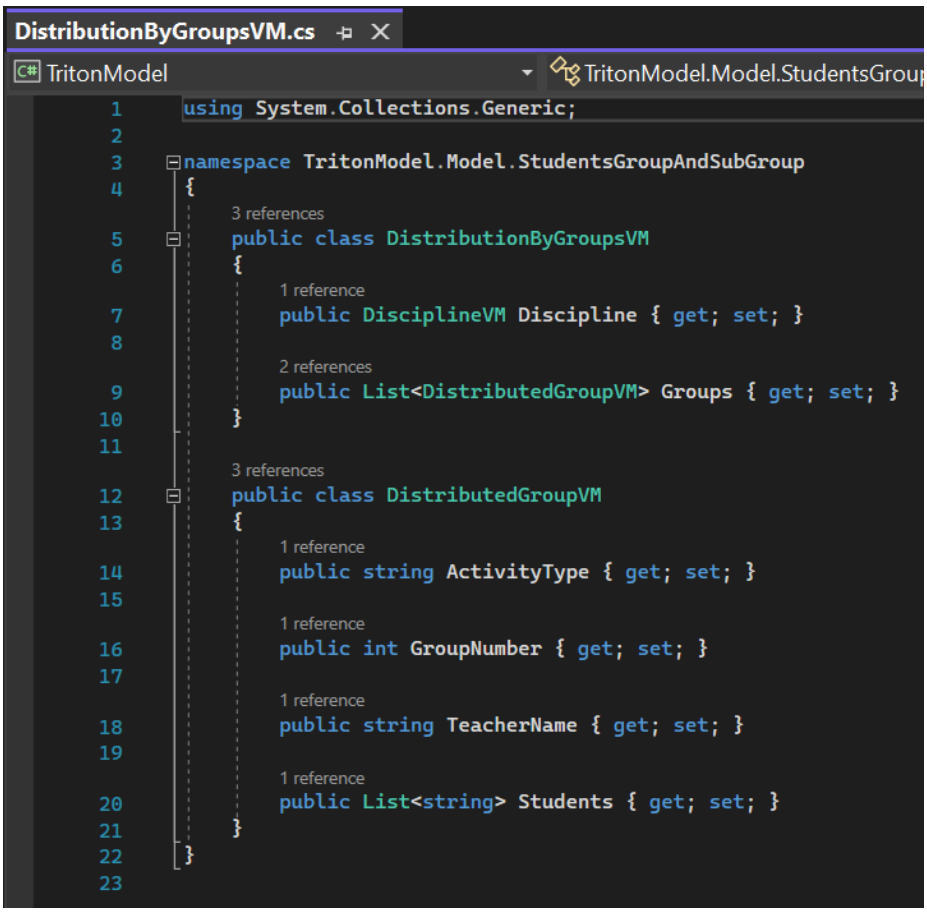

Рисунок 25 DistributionByGroupsVM та DistributedGroupVM

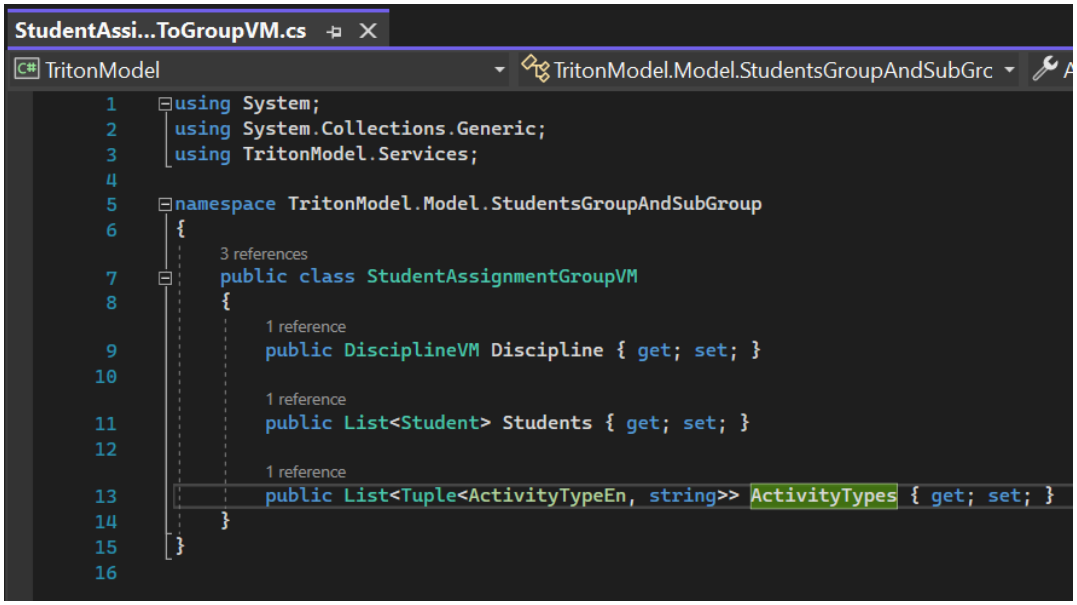

Рисунок 26 StudentAssignmentGroupVM

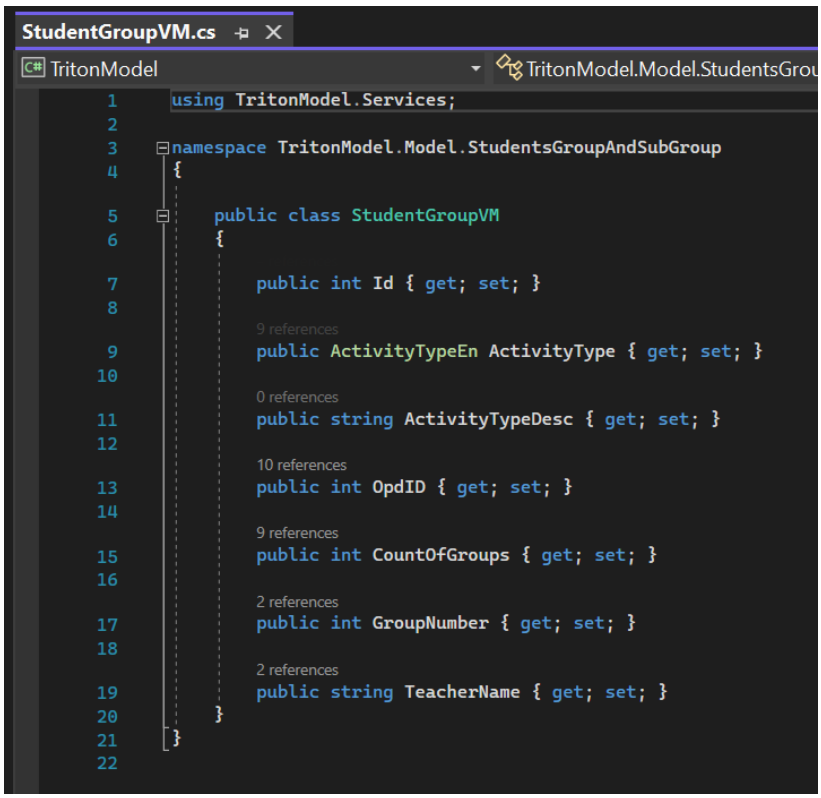

Рисунок 27 StudentGroupVM

Для реалізації другого технічного завдання було додано 8 методів в StudentController:

- CreateGroupsAndSubGroups (метод, який приймає ідентифікатори групи і семестру і повертає сторінку зі списком дисциплін для генерації груп та пігруп)
- GenerateGroupsAndSubGroups (метод, який приймає список обраних дисциплін і викликається для генерації груп та підгруп, повертає ту ж сторінку, що і CreateGroupsAndSubGroups, проте разом з результатом генерації)
- PossibleDisciplinesForGroup (метод, який приймає ідентифікатори групи і семестру і повертає сторінку з дисциплінами, на яких були створенні групи та підгрупи для подальшого призначення студентів)
- AssignmentStudentsPage (метод, який приймає список дисциплін і ідентифікатор групи і повертає сторінку для призначення студентів в групи та підгрупи)
- AssignmentStudentToSpecificGroupsOrSubgroups (метод, який приймає ідентифікатори студента та групи та зберігає призначення студента в групу або підгрупу)
- GetGroupsByDisciplineAndActivityType (метод, який по дисципліні і виду діяльності, повертає можливі групи для призначення в неї студентів)
- GetNonAssignedStudent (метод, який по дисципліні, виду діяльності і індентифікатору групи, повертає студентів, яким не призначена група)

4.2.3 Вигляд сторінок для створення груп/підгруп та призначення студентів в групи/підгрупиу

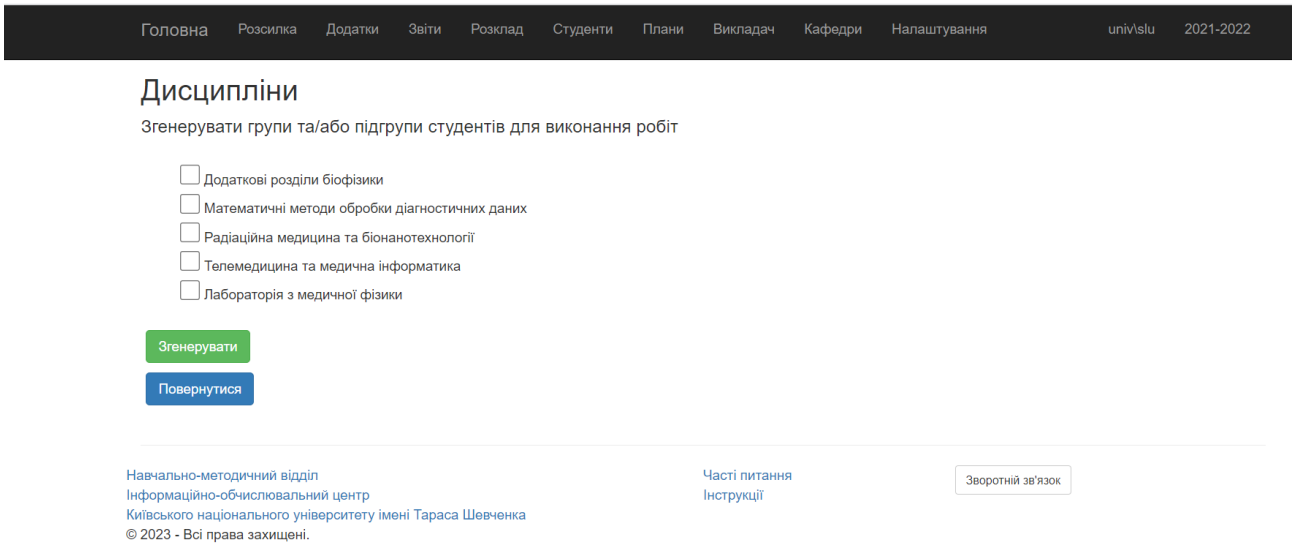

Рисунок 28 Сторінка яка використовується в 2-х місцях: а) для генерації груп та підгруп б) для 8обрання дисциплін перед додаванням студентів в групи та підгрупи

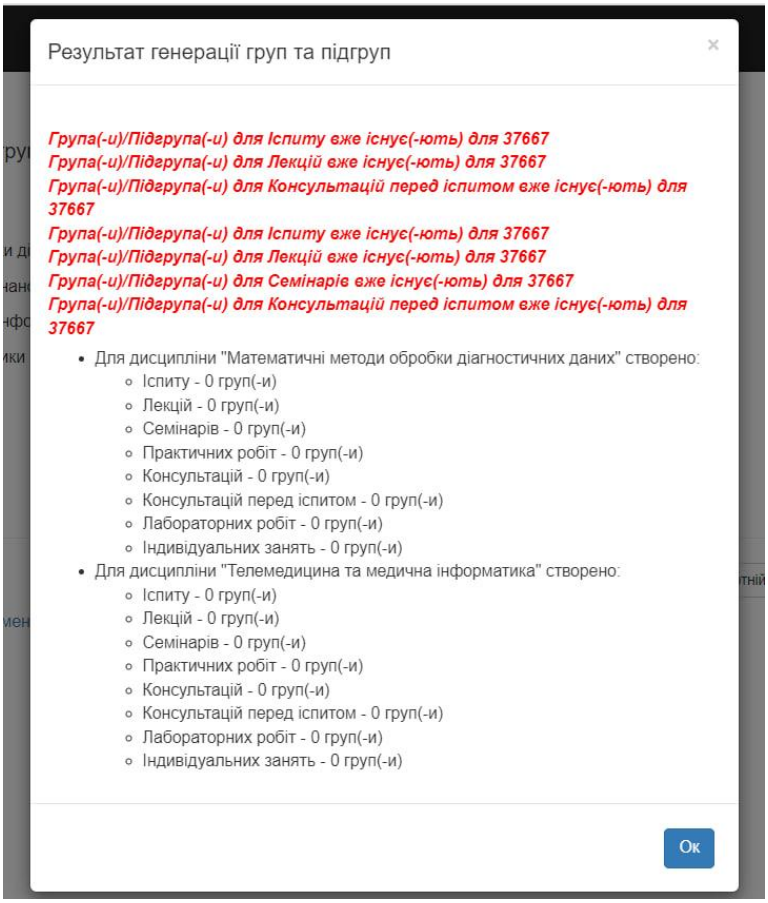

Рисунок 29 Результат генерації груп та підгруп для двох дисциплін (якщо групи і підгрупи вже були згенеровані, то вгорі з'явиться повідомлення про це червоним кольором)

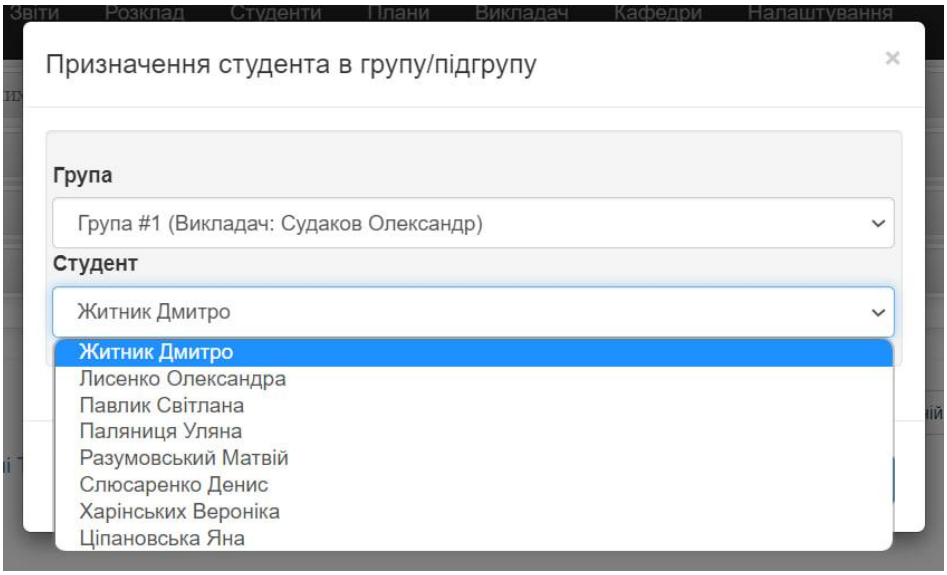

Рисунок 30 Спливаюче вікно для призначення студента в групу або підгрупу

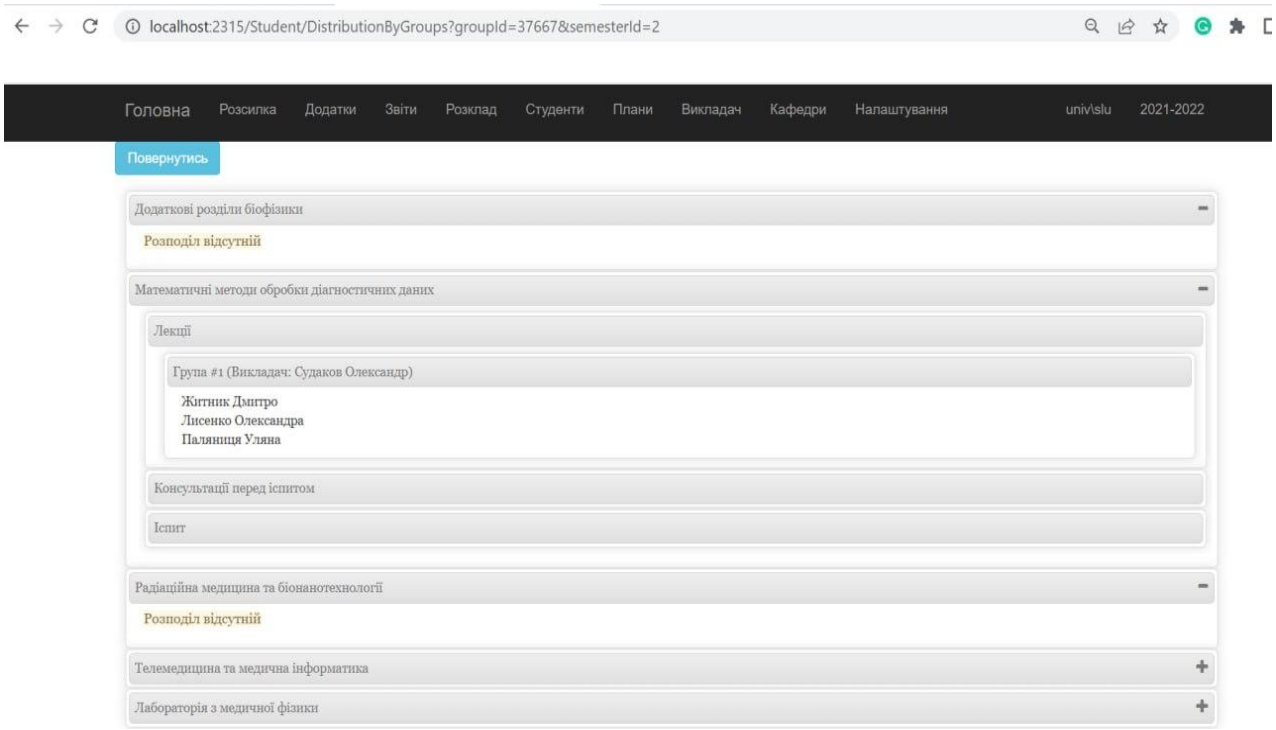

Рисунок 31 Виведення призначень студентів в групи та підгрупи

4.3 Аналіз коду

Аналіз коду рішення в NDepend показав достатньо високий результат (рівень B)

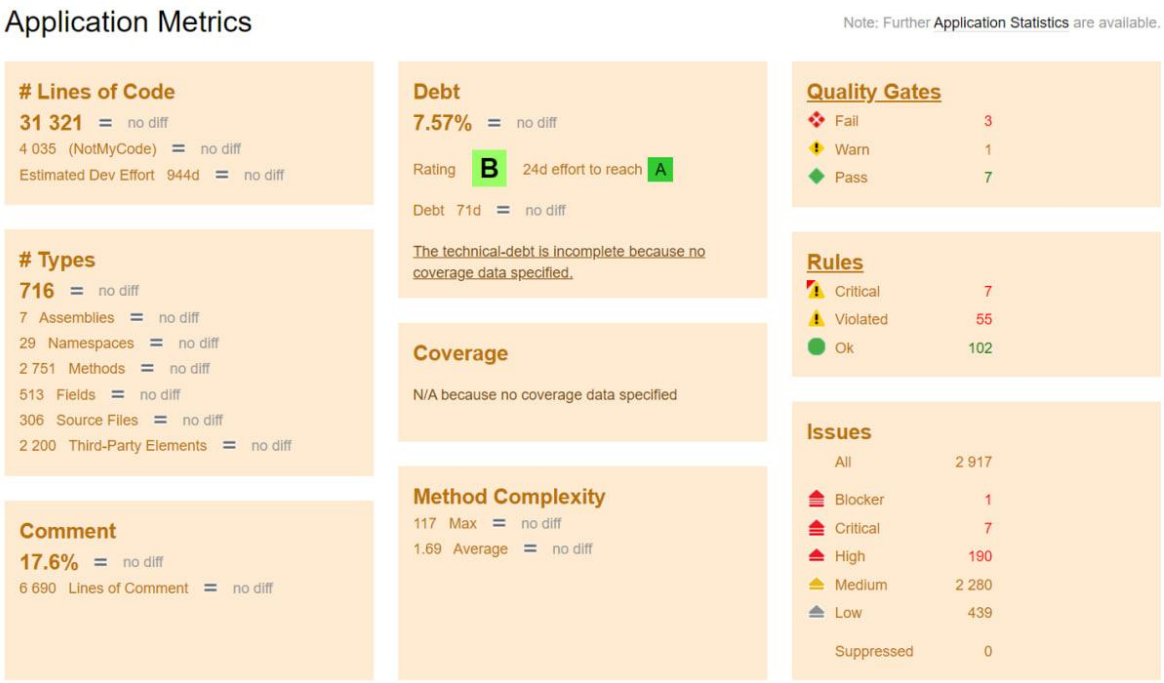

Рисунок 32 Результат аналізу коду

Name	Trend	Baseline Value	Value	Group
Percentage Coverage	=	N/A %	N/A %	Project Rules \ Quality Gates
Percentage Coverage on New Code	=	N/A %	N/A %	Project Rules \ Quality Gates
Percentage Coverage on Refactored Code	=	N/A %	N/A %	Project Rules \ Quality Gates
Blocker Issues	=	1 issues	1 issues	Project Rules \ Quality Gates
Critical Issues	=	7 issues	7 issues	Project Rules \ Quality Gates
New Blocker / Critical / High Issues			2 issues	Project Rules \ Quality Gates
Critical Rules Violated	=	7 rules	7 rules	Project Rules \ Quality Gates
Percentage Debt	=	7.57 %	7.57 %	Project Rules \ Quality Gates
New Debt since Baseline			0 man-days	Project Rules \ Quality Gates
Debt Rating per Namespace	=	0 namespaces	0 namespaces	Project Rules \ Quality Gates
New Annual Interest since Baseline			0 man-days	Project Rules \ Quality Gates

Showing 1 to 11 of 11 entries

Рисунок 33 Результат аналізу коду

Висновки

Аналіз процесу розподілу навчального навантаження та запитів користувачів системи автоматизації "Тритон", яка впроваджена в університеті, виявив недостатню деталізацію процесу розподілу при передачі навантаження між підрозділами та розподілу студентів між підгрупами для різних видів занять в межах навчальних дисциплін. Це дозволило сформувати завдання для модернізації системи, яке дозволить технічно досягти більшої прозорості для всіх учасників процесу розподілу навантаження.

Згідно поставленого завдання було обрано засоби та спроєктовано ряд модулів для розширення функціональності системи автоматизації "Тритон".

Запропонований алгоритм формування деревоподібної структури розподілу дисциплін дозволяє візуалізувати весь багатоступеневий процес передачі навантаження від оперативних планів освітніх програм до конкретних викладачів кафедр, що значно спрощує перевірку та пошук невідповідностей при погодженні навантаження на всіх етапах. Модуль прив'язки студентів до підгруп за видами занять дозволяє вести облік закріплення студентів за певними викладачами безпосередньо в системі та спрощує роботу деканатів.

Спроєктовані модулі були реалізовані в програмних засобах із використанням сучасних технологій та завдяки сумісності були інтегровані до системи "Тритон" та використовують наявну інфраструктуру, розширюючи схему бази даних та додаючи компоненти веб-інтерфейсу для візуалізації. Це дозволяє в майбутньому проводити наступні модернізації, зокрема висвітлення інформації в студентському порталі системи "Тритон".

Список посилань

1. Deanoffice-beckend. [Онлайн]. Доступний за: https://github.com/chdtu-fitis/deanoffice-backend. [Дата звернення: 05.02.2023]
2. Richter, Jeffrey. CLR via c#. Vol. 4. Redmond: Microsoft Press, 2006.
3. Abhilasha, A., & Seekoli, A. (2019). *Dotnet Framework*. LAP Lambert Academic Publishing.
4. Beasley, R. (2018). *Web Application Construction with ASP. Net 4. 6, C#. net, SQL, Ajax, and JavaScript*. Createspace Independent Publishing Platform.
5. Bellinaso, M., Hoffman, K. S., & Martinez, D. (2003). *Asp.net website programming: Problem - design - solution - visual basic .NET edition*. WROX Press.
6. Elk, K. (2018). *SQL Server with C#*. Createspace Independent Publishing Platform.
7. Gorman, B. L. (2020). *Practical entity framework: Database access for enterprise applications* (1st ed.). APress.
8. Hamilton, B., & MacDonald, M. (2003). *Ado.net in a Nutshell* (3rd ed.). O'Reilly Media.
9. Liberty, J., & Hurwitz, D. (2002). *Programming Asp.net*. O'Reilly Media.
10. Richter, J. (2012). *CLR via C#* (4th ed.). Microsoft Press.
11. Smith, L. (2018). *Software architecture with Asp.net core 2.0 MVC revised edition*. Createspace Independent Publishing Platform.

12. Watson, B. (2018). *Writing High-Performance .Net Code* (L. Watson, Ed.).
13. ASP.NET MVC. [Онлайн]. Доступний за: https://learn.microsoft.com/en-US/aspnet/mvc/. [Дата звернення: 06.02.2023]
14. ASP.Net: An Overview. [Онлайн]. Доступний за: https://msatechnosoft.in/blog/asp-net-architecture-life-cycle-events-web-development/. [Дата звернення: 06.02.2023]
15. What is .NET Framework? Explain Architecture & Components. [Онлайн]. Доступний за: https://www.guru99.com/net-framework.html. [Дата звернення: 06.02.2023]
16. A Comprehensive Guide to SQL Server Data Tools for 2023. [Онлайн]. Доступний за: https://hevodata.com/learn/sql-server-data-tools/#features. [Дата звернення: 07.02.2023]
17. Entity Framework documentation. [Онлайн]. Доступний за: https://learn.microsoft.com/en-us/ef/. [Дата звернення: 08.02.2023]
18. Treeview. [Онлайн]. Доступний за: https://mdbootstrap.com/docs/b4/jquery/plugins/treeview/. [Дата звернення: 09.02.2023]
19. CodePen Home - Bootstrap TreeView Example. [Онлайн]. Доступний за: https://codepen.io/paulch/pen/OdLoJR. [Дата звернення: 09.02.2023]
20. Tutorial: Get started with EF Core in an ASP.NET MVC web app. [Онлайн]. Доступний за: https://learn.microsoft.com/en-

us/aspnet/core/data/ef-mvc/intro?view=aspnetcore-7.0. [Дата звернення: 10.02.2023]

21. Синтаксис LINQ. [Онлайн]. Доступний за: https://learn.microsoft.com/ru-ru/dotnet/csharp/linq/. [Дата звернення: 11.02.2023]
22. Мова інтегрованого запиту (LINQ) (C#). [Онлайн]. Доступний за: https://learn.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/linq/. [Дата звернення: 11.02.2023]
23. Galloway, J., Haack, P., Wilson, B., & Allen, K. S. (2012). Professional ASP. NET MVC 4. John Wiley & Sons.
24. Maatuk, A., Ali, A., & Rossiter, N. (2019). An integrated approach to relational database migration
25. Framework, E. (2021). Entity framework. Architecture [edit], 21, 22.

Додатки

Лістинг 1

WorkloadController.cs (змінено):

```
#region Workload Tree

        public ActionResult WorkloadTreeForDiscipline(int operationPlanDisciplineID, string returnUrl)
        {
            WorkloadTree tree = servWorkload.GetWorkloadTreeForDiscipline(operationPlanDisciplineID);
            ViewData["ReturnUrl"] = returnUrl;

            return View(tree);
        }

#endregion
```

ServiceWorkload.cs (змінено):

```
#region Workload Tree

        /// <summary>
        /// Gets workload tree for discipline
        /// </summary>
        /// <param name="operationPlanDisciplineID">Operation plan discipline ID</param>
        /// <returns>Workload Tree</returns>
        /// <exception cref="DllNotFoundException"></exception>
        public WorkloadTree GetWorkloadTreeForDiscipline(int operationPlanDisciplineID)
        {
            var operationPlanDiscipline = db.OperationPlanDiscipline
                .FirstOrDefault(opd => opd.Id == operationPlanDisciplineID);

            if (operationPlanDiscipline == null)
                throw new DllNotFoundException($"{operationPlanDisciplineID} оперативний план не знайдено");

            var tree = GetMainDiscipline(operationPlanDiscipline);
            FillUpMainDiscipline(ref tree, operationPlanDisciplineID);

            return tree;
        }

        private void FillUpMainDiscipline(ref WorkloadTree tree, int selectedOperationPlanDisciplineID)
        {
            var isSuccessParsing = Int32.TryParse(tree.MainDiscipline.text.Split(';')[0], out int mainDisciplineID);
            if (!isSuccessParsing)
                throw new Exception("Неможливо розпарсити OperationPlanDisciplineID як Int");

            var dividedDisciplines = db.OperationPlanDiscipline
                .Where(opd => opd.ParentDisciplineId.HasValue && opd.ParentDisciplineId.Value == mainDisciplineID);
```

```
    if (dividedDisciplines.Count() > 0)
    {
        tree.MainDiscipline.nodes = new List<BranchItem>();

        var branchItems = GetBranchItems(dividedDisciplines);
        tree.MainDiscipline.nodes.AddRange(branchItems);

        tree.MainDiscipline.nodes.ForEach(b => FillBranch(ref b, selectedOperationPlanDisciplineID));
    }
}

private void FillBranch(ref BranchItem branch, int selectedOperationPlanDisciplineID)
{
    var isSuccessParsing = Int32.TryParse(branch.text, out int branchID);
    if (!isSuccessParsing)
        throw new Exception("Неможливо розпарсити OperationPlanDisciplineID як Int");

    var operationPlanDiscipline = db.OperationPlanDiscipline
        .FirstOrDefault(opd => opd.Id == branchID);

    if (selectedOperationPlanDisciplineID.ToString() == branch.text)
    {
        branch.selected = true;
    }

    SetTextOfNode(branch, operationPlanDiscipline);

    var dividedDisciplines = db.OperationPlanDiscipline
        .Where(opd => opd.ParentDisciplineId == branchID);

    if (dividedDisciplines.Count() > 0)
    {
        branch.nodes = new List<BranchItem>();
        branch.nodes.AddRange(GetBranchItems(dividedDisciplines));

        branch.nodes.ForEach(b => FillBranch(ref b, selectedOperationPlanDisciplineID));
    }
}

private WorkloadTree GetMainDiscipline(OperationPlanDiscipline operationPlanDiscipline)
{
    while (!ServiceWorkload.IsPrimaryDiscipline(operationPlanDiscipline))
    {
        operationPlanDiscipline = db.OperationPlanDiscipline
            .FirstOrDefault(opd => opd.Id == operationPlanDiscipline.ParentDisciplineId);
    }

    WorkloadTree tree = new WorkloadTree()
    {
        MainDiscipline = new BranchItem()
    };

    SetTextOfNode(tree.MainDiscipline, operationPlanDiscipline);

    return tree;
}
```

```
        private void SetTextOfNode(BranchItem branch, OperationPlanDiscipline operationPlanDiscipline)
        {
            string union = operationPlanDiscipline.ParentDiscipline2Id.HasValue ? $" було сформовано через об'єднання з {operationPlanDiscipline.ParentDiscipline2Id.Value};" : string.Empty;
            string cathedra = operationPlanDiscipline.HourLoadBit ? GetCathedraName(operationPlanDiscipline.Id) : string.Empty;
            string faculty = operationPlanDiscipline.ClaimsBit ? GetFacultyName(operationPlanDiscipline.Id) : string.Empty;
            string operationPlan = operationPlanDiscipline.OperationPlanBit ? GetOperativePlan(operationPlanDiscipline) : string.Empty;
            string workload = operationPlanDiscipline.HourLoadBit ? GetWorkloadText(operationPlanDiscipline) : string.Empty;

            branch.text = $"{operationPlanDiscipline.Id};{union}{workload}{operationPlan}{cathedra}{faculty}";
        }

        private List<BranchItem> GetBranchItems(IQueryable<OperationPlanDiscipline> operationPlanDiscipline)
        {
            return operationPlanDiscipline
                .Select(opd => new BranchItem()
                {
                    text = opd.Id.ToString()
                })
                .ToList();
        }

        private string GetWorkloadText(OperationPlanDiscipline operationPlanDiscipline)
        {
            var workloadId = db.WorkloadDiscipline.FirstOrDefault(wd => wd.OperationPlanDisciplineId == operationPlanDiscipline.Id).WorkloadId;
            var cathedraId = db.Workload.FirstOrDefault(w => w.Id == workloadId).CathedraId;
            var courseId = db.OperationPlan.FirstOrDefault(op => op.Id == operationPlanDiscipline.OperationPlan).CourseOfStudyId;
            var syllabusId = db.OperationPlan.FirstOrDefault(op => op.Id == operationPlanDiscipline.OperationPlan).SyllabusId;
            var levelId = db.Syllabus.FirstOrDefault(s => s.Id == syllabusId).LevelOfEducationId;

            return $" Навантаження: <a href='/Workload/Edit?cathedraId={cathedraId}&courseId={courseId}&levelId={levelId}'>{workloadId}</a>;";
        }

        private string GetOperativePlan(OperationPlanDiscipline discipline)
        {
            var semester = db.SemesterOfStudy.FirstOrDefault(s => s.Id == discipline.SemesterId).SemestOfStudyTypeId;
            var type = discipline.DisciplineTypeId;

            return $" Оперативний план: <a href='/OperativePlan/EditOP?planId={discipline.OperationPlan}&semester={semester}&type={type}'>{discipline.OperationPlan}</a>;";
        }

        private string GetCathedraName(int disciplineId)
```

```
        {
            var workloadId = db.WorkloadDiscipline.FirstOrDefault(wd => wd.OperationPlanDisciplineId ==
disciplineId).WorkloadId;
            var cathedraId = db.Workload.FirstOrDefault(w => w.Id == workloadId).CathedraId;
            var cathedraName = db.Cathedra.FirstOrDefault(c => c.Id == cathedraId).Name;

            return $" Кафедра: {cathedraName};";
        }

        private string GetFacultyName(int disciplineId)
        {
            ServiceClaims claimsService = new ServiceClaims();
            var cathedraId = claimsService.GetLastDisciplineClaim(disciplineId).CathedraId;
            var facultyId = db.Cathedra.FirstOrDefault(c => c.Id == cathedraId).FacultyId;
            var facultyName = db.Faculty.FirstOrDefault(f => f.Id == facultyId).Name;

            return $" Факультет: {facultyName};";
        }

        private static bool IsPrimaryDiscipline(OperationPlanDiscipline operationPlanDiscipline)
        {
            return operationPlanDiscipline != null
                && !operationPlanDiscipline.ParentDisciplineId.HasValue
                && !operationPlanDiscipline.ParentDiscipline2Id.HasValue;
        }

    #endregion
```

WorkloadTreeForDiscipline.cshtml (додано):

```
@model TritonModel.Model.WorkloadTree.WorkloadTree

@using Newtonsoft.Json

@{
    ViewBag.Title = "WorloadTreeForDiscipline";
}

<script type="text/javascript" src="~/Scripts/jquery-3.4.1.min.js"></script>
<script type="text/javascript" src="~/Scripts/bootree.min.js"></script>
<link rel="stylesheet" href="~/Content/bootree.min.css" />

<style>
    .btn.btn-primary {
        margin-top: 20px;
    }

    #treeview li.list-group-item.node-treeview.node-selected:not(.selected) {
        background-color: unset !important;
        color: black !important;
    }

    #treeview li.list-group-item.node-treeview.selected {
        background-color: #43ccda;
        color: white;
    }
```

```
</style>

<span>
    <a class="btn btn-primary" href="@ViewData["ReturnUrl"]">Повернутись</a>
</span>

<h2>Дерево розгалужень дисциплі ни</h2>

<div>
    <div id="treeview" class="treeview"></div>
</div>

<script type="text/javascript">
    $(function () {
        var treeModel = unescape('@Html.Raw(JsonConvert.SerializeObject(Model.MainDiscipline, new
JsonSerializerSettings { StringEscapeHandling = StringEscapeHandling.EscapeHtml }))');
        treeModel = [eval("(" + treeModel + ")")];

        $('#treeview').treeview(
            {
                data: treeModel,
                state: {
                    expanded: true
                }
            });

        $('#treeview').data('treeview').expandAll();
    });
</script>
```

Лістинг 2:

StudentController.cs (змінено):

```
#region Create Groups and SubGroups

        [HttpGet]
        public ActionResult CreateGroupsAndSubGroups(int groupId, int semesterId)
        {
            var listOfDiscipline = serviceStudentGroup.GetSemesterDisciplinesForGroup(groupId,
semesterId);

            ViewData["GroupID"] = groupId;
            ViewData["SemesterID"] = semesterId;

            return View("GetDisciplesForGenerateOrAssignment", listOfDiscipline);
        }

        [HttpPost]
        public ActionResult
GenerateGroupsAndSubGroups(List<StudentGroupAndSubGroupVM.DisciplineVM> disciplines, int groupId, int
semesterId)
        {
            var selectedDisciplines = serviceStudentGroup.GetSelectedDisciplines(disciplines);
```

```
            var studentGroupsAndSubGroups =
serviceStudentGroup.GenerateStudentGroupsAndSubGroups(selectedDisciplines, groupId);
            var listOfDiscipline = serviceStudentGroup.GetSemesterDisciplinesForGroup(groupId,
semesterId);

            ViewData["GroupID"] = groupId;
            ViewData["SemesterID"] = semesterId;
            ViewData["ResultOfGenerating"] = studentGroupsAndSubGroups;
            ViewData["ErrorMessage"] = serviceStudentGroup.ErrorMessage;

            return View("GetDisciplinesForGenerateOrAssignment", listOfDiscipline);
        }

        [HttpGet]
        public ActionResult PossibleDisciplinesForGroup(int groupId, int semesterId)
        {
            var listOfDisciplines = serviceStudentGroup.GetPossibleDisciplinesForGroup(groupId,
semesterId);

            ViewData["GroupID"] = groupId;
            ViewData["SemesterID"] = semesterId;
            ViewData["IsAssignmentStudents"] = true;

            return View("GetDisciplinesForGenerateOrAssignment", listOfDisciplines);
        }

        [HttpPost]
        public ActionResult AssignmentStudentsPage(List<StudentGroupAndSubGroupVM.DisciplineVM>
disciplines, int groupId)
        {
            var selectedDisciplines = serviceStudentGroup.GetSelectedDisciplines(disciplines);
            var studentAssignmentGroups =
serviceStudentGroup.GetStudentAssignmentGroups(selectedDisciplines, groupId);

            ViewData["GroupID"] = groupId;

            return View(studentAssignmentGroups);
        }

        [HttpGet]
        public ActionResult GetGroupsByDisciplineAndActivityType(int operationPlanDisciplineId,
ActivityTypeEn activityType)
        {
            var groups =
serviceStudentGroup.GetGroupsByDisciplineAndActivityType(operationPlanDisciplineId, activityType)
                .Select(g => new SelectListItem()
                {
                    Text = String.Concat("Група #", g.GroupNumber, " (Викладач: ", g.TeacherName, ")"),
                    Value = g.Id.ToString()
                });
            return Json(groups, JsonRequestBehavior.AllowGet);
        }

        [HttpGet]
        public ActionResult GetNonAssignedStudent(int operationPlanDisciplineId, ActivityTypeEn
activityType, int groupId)
```

```
        {
            var students = serviceStudentGroup.GetNonAssignedStudent(operationPlanDisciplineId,
activityType, groupId)
                .Select(s => new SelectListItem()
                {
                    Text = String.Concat(s.LastName, " ", s.FirstName),
                    Value = s.Id.ToString()
                });
            return Json(students, JsonRequestBehavior.AllowGet);
        }

        [HttpPost]
        public ActionResult AssignmentStudentToSpecificGroupsOrSubgroups(int studentId, int
workloadDisciplineGroupId)
        {
            var result = serviceStudentGroup.SaveAssingmentStudentToGroup(studentId,
workloadDisciplineGroupId);

            return Json(new { isSuccess = result.Item1, error = result.Item2 });
        }

        public ActionResult DistributionByGroups(int groupId, int semesterId)
        {
            var listOfDisciplines = serviceStudentGroup.GetSemesterDisciplinesForGroup(groupId,
semesterId);
            var result = serviceStudentGroup.GetDistributionByGroups(listOfDisciplines);

            return View(result);
        }

    #endregion
```

StudentGroupService.cs (додано):

```
    #region Public Properties

        public string ErrorMessage { get; set; }

    #endregion
    #region Public Methods

        /// <summary>
        /// Get ALL semecter disciplines for students group
        /// </summary>
        /// <param name="groupId">Group id</param>
        /// <param name="semesterId">Semecter id</param>
        /// <returns>List of disciplines</returns>
        public List<ViewModels.DisciplineVM> GetSemesterDisciplinesForGroup(int groupId, int
semesterId)
        {
            return db.OperationPlanDiscipline
                .Join(db.StudentsGroup, opd => opd.OperationPlan, sg => sg.OperationPlanId, (opd, sg) =>
new { StudentGroupID = sg.Id, OperationPlanDiscipline = opd })
```

```
                .Where(d => d.StudentGroupID == groupId && d.OperationPlanDiscipline.SemesterId ==
semesterId && d.OperationPlanDiscipline.HourLoadBit && d.OperationPlanDiscipline.ParentDisciplineId ==
null && d.OperationPlanDiscipline.ParentDiscipline2Id == null)
                .Select(d => new ViewModels.DisciplineVM
                {
                    DisciplineID = d.OperationPlanDiscipline.DisciplineId,
                    OperationPlanID = d.OperationPlanDiscipline.OperationPlan,
                    OperationPlanDisciplineID = d.OperationPlanDiscipline.Id,
                    DisciplineName = d.OperationPlanDiscipline.Name
                })
                .ToList();
            }

            /// <summary>
            /// The first method that is called to generate groups and subgroups
            /// </summary>
            /// <param name="disciplines">List of disciplines</param>
            /// <param name="groupId">Group id</param>
            /// <returns>Dictionary, where key is opdId and value is list of tuples, where first item -
activityType as string and second item - count of created groups</returns>
            public Dictionary<string, List<Tuple<string, int>>>
GenerateStudentGroupsAndSubGroups(List<ViewModels.DisciplineVM> disciplines, int groupId)
            {
                var leafsFromOperationPlanDiscipline = new List<OperationPlanDiscipline>();

                foreach (var discipline in disciplines)
                {
                    var opd = db.OperationPlanDiscipline
                        .First(x => x.Id == discipline.OperationPlanDisciplineID);

                    if (!IsWorkloadTreeLeaf(discipline.OperationPlanDisciplineID))
                    {
                        SetWorkloadTreeLeafs(opd, leafsFromOperationPlanDiscipline);
                    }
                    else
                    {
                        leafsFromOperationPlanDiscipline.Add(opd);
                    }
                }

                var leafsGroupedByDiscipline = leafsFromOperationPlanDiscipline
                    .GroupBy(l => l.Name);

                return GenerateStudentGroupsAndSubGroups(leafsGroupedByDiscipline, groupId);
            }

            /// <summary>
            /// General method to generate students groups and subgroups
            /// </summary>
            /// <param name="leafsGroupedByDiscipline">Grouped leafs by discipline name</param>
            /// <param name="groupId">Group id</param>
            /// <returns>ey is opdId and value is list of tuples, where first item - activityType as string and
second item - count of created groups</returns>
            public Dictionary<string, List<Tuple<string, int>>> GenerateStudentGroupsAndSubGroups
                (IEnumerable<IGrouping<string, OperationPlanDiscipline>> leafsGroupedByDiscipline, int
groupId)
```

```
        {
            Dictionary<string, List<Tuple<string, int>>> result = new Dictionary<string, List<Tuple<string, int>>>();

            this.ErrorMessage = string.Empty;

            foreach (var leafs in leafsGroupedByDiscipline)
            {
                var resultList = new List<Tuple<string, int>>();

                foreach (var activityTypeId in db.ActivityType1Set.Select(at => at.ID))
                {
                    var activityTypeEn = (ActivityTypeEn)activityTypeId;

                    var groupsForActivityType = GetStudentGroupsFromDisciplineLeafs(activityTypeEn, leafs);
                    var activityTypeLabel = GetActivityTypeEnDescription(activityTypeEn);

                    resultList.Add(new Tuple<string, int>(activityTypeLabel, GenerateStudentGroupsAndSubGroupsForSpecificActivityType(activityTypeLabel, groupsForActivityType, groupId)));
                }

                result.Add(leafs.Key, resultList);
            }

            return result;
        }

        /// <summary>
        /// Specified generate student groups and subgroups method, that creates groups for specified discipline and specified activity type
        /// </summary>
        /// <param name="activityTypeStr">Activity type for group/subgroup that should be created</param>
        /// <param name="groups">List of groups</param>
        /// <param name="groupId">Student group id</param>
        /// <returns>Count of created groups/subgroups</returns>
        public int GenerateStudentGroupsAndSubGroupsForSpecificActivityType(string activityTypeStr, List<ViewModels.StudentGroupVM> groups, int groupId)
        {
            if (groups.Count == 0)
                return 0;

            var activityType = groups.First().ActivityType;

            // check whether groups/subgroups already exist
            var operationPlanDisciplines = groups.Select(g => g.OpdID);
            var workloadDisciplines = db.WorkloadDiscipline
                .Where(wd => operationPlanDisciplines.Contains(wd.OperationPlanDisciplineId))
                .Select(wd => wd.Id);
            var isGroupsExisting = db.WorkloadDisciplineStudentsGroups
                .Any(wdsg => workloadDisciplines.Contains(wdsg.WorkloadDisciplineID)
                    && wdsg.ActivityTypeID == (int)activityType);

            if (isGroupsExisting)
            {
```

```
                this.ErrorMessage += $"<br />Група(-и)/Пі дгрупа(-и) для {activityTypeStr} вже
і існує(-ють) для {groupId}";

                return 0;
            }

            var countOfGroups = groups
                .Sum(g => g.CountOfGroups);
            var countOfStudents = db.Student
                .Where(s => s.ActiveGroupId == groupId)
                .Count();

            for (int i = 0; i < groups.Count(); i++)
            {
                var currentGroup = groups[i];
                var maxCountOfStudentsInGroup = countOfStudents / countOfGroups;

                if (i == groups.Count() - 1)
                {
                    maxCountOfStudentsInGroup = (countOfStudents / countOfGroups)
                        + (countOfStudents % countOfGroups);
                }

                var workloadDisciplineID = GetWorkloadDisciplineByOpdID(currentGroup.OpdID).Id;

                db.WorkloadDisciplineStudentsGroups.Add(new WorkloadDisciplineStudentsGroup()
                {
                    GroupNumber = i + 1,
                    WorkloadDisciplineID = workloadDisciplineID,
                    ActivityTypeID = (int)activityType,
                    MaxCountOfStudents = maxCountOfStudentsInGroup
                });
            }

            db.SaveChanges();

            return countOfGroups;
        }
    /// <summary>
        /// Set workload tree leafs
        /// </summary>
        /// <param name="operationPlanDiscipline">Operation plan discipline object</param>
        /// <param name="leafs">List of leafs</param>
        public void SetWorkloadTreeLeafs(OperationPlanDiscipline operationPlanDiscipline,
List<OperationPlanDiscipline> leafs)
        {
            var unionAndDividedDisciplines = (db.OperationPlanDiscipline
                    .Where(opd => (opd.ParentDisciplineId == operationPlanDiscipline.Id &&
opd.ParentDiscipline2Id != null)
                                || (opd.ParentDiscipline2Id == operationPlanDiscipline.Id &&
opd.ParentDisciplineId != null)))
                    .Union(db.OperationPlanDiscipline
                    .Where(opd => opd.ParentDisciplineId == operationPlanDiscipline.Id &&
opd.ParentDiscipline2Id == null));

            foreach (var unionDiscipline in unionAndDividedDisciplines)
```

```
                {
                    if (IsWorkloadTreeLeaf(unionDiscipline.OperationPlan))
                        leafs.Add(unionDiscipline);
                    else
                        SetWorkloadTreeLeafs(unionDiscipline, leafs);
                }
            }

            /// <summary>
            /// Get possible disciplines for group (only when student groups/subgroups were created) on
which you can add students
            /// </summary>
            /// <param name="groupId">Group id</param>
            /// <param name="semesterId">Semester id</param>
            /// <returns>List of possible disciplines</returns>
            public List<ViewModels.DisciplineVM> GetPossibleDisciplinesForGroup(int groupId, int
semesterId)
            {
                var allDisciplies = GetSemesterDisciplinesForGroup(groupId, semesterId);

                var operationPlanDisciplines = allDisciplies
                    .Select(d => d.OperationPlanDisciplineID)
                    .Where(opd => db.WorkloadDiscipline.Any(wd => wd.OperationPlanDisciplineId == opd
                        && db.WorkloadDisciplineStudentsGroups.Any(wdsg => wdsg.WorkloadDisciplineID ==
wd.Id)));

                return allDisciplies.Where(d =>
operationPlanDisciplines.Contains(d.OperationPlanDisciplineID)).ToList();
            }

            /// <summary>
            /// Get student assignment groups
            /// </summary>
            /// <param name="disciplines">List of disciplines</param>
            /// <param name="groupId">Group id</param>
            /// <returns>List of student assignment groups</returns>
            public List<ViewModels.StudentAssignmentGroupVM>
GetStudentAssignmentGroups(List<ViewModels.DisciplineVM> disciplines, int groupId)
            {
                var result = new List<ViewModels.StudentAssignmentGroupVM>();
                var students = db.Student
                    .Where(s => s.ActiveGroupId == groupId)
                    .ToList();

                foreach (var discipline in disciplines)
                {
                    var workloadDiscipline =
GetWorkloadDisciplineByOpdID(discipline.OperationPlanDisciplineID);

                    var resultActivityTypes = new List<Tuple<ActivityTypeEn, string>>();
                    var activityTypes = db.WorkloadDisciplineStudentsGroups
                        .Where(wdsg => wdsg.WorkloadDisciplineID == workloadDiscipline.Id)
                        .Select(wdsg => (ActivityTypeEn)wdsg.ActivityTypeID)
                        .Distinct()
                        .ToList();
                    activityTypes.ForEach(at => resultActivityTypes.Add(
```

```
                    new Tuple<ActivityTypeEn, string>(at, GetActivityTypeEnDescription(at, false))));

                result.Add(new ViewModels.StudentAssignmentGroupVM()
                {
                    Discipline = discipline,
                    Students = students,
                    ActivityTypes = resultActivityTypes
                });
            }

            return result;
        }
        /// <summary>
        /// Get groups by discipline and activity type
        /// </summary>
        /// <param name="opdID">Operation plan discipline id</param>
        /// <param name="activityType"> Activity type</param>
        /// <returns>List of student groups</returns>
        public List<ViewModels.StudentGroupVM> GetGroupsByDisciplineAndActivityType(int opdID,
ActivityTypeEn activityType)
        {
            var result = new List<ViewModels.StudentGroupVM>();
            var workloadDiscipline = GetWorkloadDisciplineByOpdID(opdID);
            var teacherName = GetTeacherNameByWorkloadDisciplineId(workloadDiscipline.TeacherId);

            var workloadDisciplineStudentGroups = db.WorkloadDisciplineStudentsGroups
                .Where(wdsg => wdsg.WorkloadDisciplineID == workloadDiscipline.Id &&
wdsg.ActivityTypeID == (int)activityType)
                .ToList();

            var workloadDisciplineStudentGroupIds = workloadDisciplineStudentGroups
                .Select(wdsg => wdsg.ID);
            var workloadDisciplineStudentsCount = db.WorkloadDisciplineStudents
                .Where(wds =>
workloadDisciplineStudentGroupIds.Contains(wds.WorkloadDisciplineStudentsGroupID))
                .Count();

            foreach (var group in workloadDisciplineStudentGroups
                .Where(wdsg => wdsg.MaxCountOfStudents < workloadDisciplineStudentsCount))
            {
                result.Add(new ViewModels.StudentGroupVM()
                {
                    Id = group.ID,
                    GroupNumber = group.GroupNumber,
                    TeacherName = teacherName
                });
            }

            return result;
        }

        /// <summary>
        /// Get Non assigned student for discipline
        /// </summary>
        /// <param name="opdID">Operation plan discipline id</param>
        /// <param name="activityType">Activity type</param>
```

```
        /// <param name="groupId">Group id</param>
        /// <returns>List of students</returns>
        public List<Student> GetNonAssignedStudent(int opdID, ActivityTypeEn activityType, int groupId)
        {
            var result = new List<Student>();

            var workloadDisciplineId = GetWorkloadDisciplineByOpdID(opdID).Id;
            var workloadDisciplineStudentGroupIds = db.WorkloadDisciplineStudentsGroups
                .Where(wdsg => wdsg.WorkloadDisciplineID == workloadDisciplineId &&
wdsg.ActivityTypeID == (int)activityType)
                .Select(wdsg => wdsg.ID)
                .ToList();

            var allStudents = db.Student.Where(s => s.ActiveGroupId == groupId).ToList();

            return allStudents
                .Where(s => !db.WorkloadDisciplineStudents.Any(wds => wds.StudentID == s.Id
                    &&
workloadDisciplineStudentGroupIds.Contains(wds.WorkloadDisciplineStudentsGroupID)))
                .ToList();
        }

        /// <summary>
        /// Save assignment student to group
        /// </summary>
        /// <param name="studentId">Student id</param>
        /// <param name="workloadDisciplineGroupId">Workload discipline group id</param>
        /// <returns>Tuple, where first item - isSuccess and second item - error message</returns>
        public Tuple<bool, string> SaveAssingmentStudentToGroup(int studentId, int
workloadDisciplineGroupId)
        {
            bool isSuccess = false;
            string error = string.Empty;

            try
            {
                var countOfStudentsInGroup =
GetCountOfStudentsInWorkloadDisciplineGroup(workloadDisciplineGroupId);

                var workloadDisciplineStudentGroup =
GetWorkloadDisciplineStudentGroupByID(workloadDisciplineGroupId);
                if (workloadDisciplineStudentGroup == null ||
workloadDisciplineStudentGroup.MaxCountOfStudents <= countOfStudentsInGroup)
                    throw new Exception();

                db.WorkloadDisciplineStudents.Add(new WorkloadDisciplineStudent()
                {
                    StudentID = studentId,
                    WorkloadDisciplineStudentsGroupID = workloadDisciplineGroupId
                });

                db.SaveChanges();

                isSuccess = true;
            }
            catch
```

```
                {
                    error = "Сталась помилка при збереженні даних!";
                }

                return new Tuple<bool, string>(isSuccess, error);
            }

            /// <summary>
            /// Get distribution by groups
            /// </summary>
            /// <param name="disciplines">List of disciplines</param>
            /// <returns>List of distribution groups</returns>
            public List<ViewModels.DistributionByGroupsVM>
GetDistributionByGroups(List<ViewModels.DisciplineVM> disciplines)
            {
                var result = new List<ViewModels.DistributionByGroupsVM>();

                foreach (var discipline in disciplines)
                {
                    var workloadDiscipline =
GetWorkloadDisciplineByOpdID(discipline.OperationPlanDisciplineID);

                    var groups = db.WorkloadDisciplineStudentsGroups
                        .Where(wdsg => wdsg.WorkloadDisciplineID == workloadDiscipline.Id);

                    var item = new ViewModels.DistributionByGroupsVM()
                    {
                        Discipline = discipline,
                        Groups = new List<ViewModels.DistributedGroupVM>()
                    };

                    foreach (var group in groups)
                    {
                        var studentIds = db.WorkloadDisciplineStudents
                            .Where(wds => wds.WorkloadDisciplineStudentsGroupID == group.ID)
                            .Select(wds => wds.StudentID);
                        var students = db.Student
                            .Where(s => studentIds.Contains(s.Id))
                            .Select(s => string.Concat(s.LastName, " ", s.FirstName))
                            .ToList();

                        var teacherName =
GetTeacherNameByWorkloadDisciplineId(workloadDiscipline.TeacherId);

                        item.Groups.Add(new ViewModels.DistributedGroupVM()
                        {
                            ActivityType = GetActivityTypeEnDescription((ActivityTypeEn)group.ActivityTypeID,
false),
                            GroupNumber = group.GroupNumber,
                            TeacherName = teacherName,
                            Students = students
                        });
                    }

                    result.Add(item);
                }
```

```
            return result;
        }
    /// <summary>
        /// Returns list of selected disciplines
        /// </summary>
        /// <param name="disciplines">List of disciplines</param>
        /// <returns>List of disciplines</returns>
        public List<ViewModels.DisciplineVM> GetSelectedDisciplines(List<ViewModels.DisciplineVM>
disciplines)
        {
            return disciplines
                .Where(d => d.IsSelected)
                .ToList();
        }

    #endregion

    #region Private Methods

        private bool IsWorkloadTreeLeaf(int opdID)
        {
            return !db.OperationPlanDiscipline
                .Any(opd => opd.ParentDisciplineId == opdID || opd.ParentDiscipline2Id == opdID);
        }

        private List<ViewModels.StudentGroupVM> GetStudentGroupsFromDisciplineLeafs(
            ActivityTypeEn activityType,
            IGrouping<string, OperationPlanDiscipline> leafs)
        {
            switch (activityType)
            {
                case ActivityTypeEn.Lecture:
                    return leafs.Where(l => l.DevisionsCountEduLectures > 0)
                                .Select(l => new ViewModels.StudentGroupVM
                                {
                                    ActivityType = ActivityTypeEn.Lecture,
                                    OpdID = l.Id,
                                    CountOfGroups = l.DevisionsCountEduLectures
                                })
                                .ToList();
                case ActivityTypeEn.Lab:
                    return leafs.Where(l => l.DevisionsCountEduLabs > 0)
                                .Select(l => new ViewModels.StudentGroupVM
                                {
                                    ActivityType = ActivityTypeEn.Lab,
                                    OpdID = l.Id,
                                    CountOfGroups = l.DevisionsCountEduLabs
                                })
                                .ToList();
                case ActivityTypeEn.Practical:
                    return leafs.Where(l => l.DevisionsCountEduPracticalClasses > 0)
                                .Select(l => new ViewModels.StudentGroupVM
                                {
                                    ActivityType = ActivityTypeEn.Practical,
                                    OpdID = l.Id,
```

```
                    CountOfGroups = l.DevisionsCountEduPracticalClasses
                })
                .ToList();
        case ActivityTypeEn.Seminar:
            return leafs.Where(l => l.DevisionsCountEduSeminars > 0)
                .Select(l => new ViewModels.StudentGroupVM
                {
                    ActivityType = ActivityTypeEn.Seminar,
                    OpdID = l.Id,
                    CountOfGroups = l.DevisionsCountEduSeminars
                })
                .ToList();
        case ActivityTypeEn.Individual:
            return leafs.Where(l => l.DevisionsCountEduIndividualClasses > 0)
                .Select(l => new ViewModels.StudentGroupVM
                {
                    ActivityType = ActivityTypeEn.Individual,
                    OpdID = l.Id,
                    CountOfGroups = l.DevisionsCountEduIndividualClasses
                })
                .ToList();
        case ActivityTypeEn.Consultation:
            return leafs.Where(l => l.DevisionsCountEduConsultations > 0)
                .Select(l => new ViewModels.StudentGroupVM
                {
                    ActivityType = ActivityTypeEn.Consultation,
                    OpdID = l.Id,
                    CountOfGroups = l.DevisionsCountEduConsultations
                })
                .ToList();
        case ActivityTypeEn.ExamConsultation:
            return leafs.Where(l => l.DevisionsCountExamConsultations > 0)
                .Select(l => new ViewModels.StudentGroupVM
                {
                    ActivityType = ActivityTypeEn.ExamConsultation,
                    OpdID = l.Id,
                    CountOfGroups = l.DevisionsCountExamConsultations
                })
                .ToList();
        case ActivityTypeEn.Exam:
            return leafs.Where(l => l.DevisionsCountExam > 0)
                .Select(l => new ViewModels.StudentGroupVM
                {
                    ActivityType = ActivityTypeEn.Exam,
                    OpdID = l.Id,
                    CountOfGroups = l.DevisionsCountExam
                })
                .ToList();
        default:
            throw new Exception("Невідомий вид діяльності !");
    }
}

private WorkloadDiscipline GetWorkloadDisciplineByOpdID(int opdId)
{
    return db.WorkloadDiscipline
```

```
                .First(wd => wd.OperationPlanDisciplineId == opdId);
        }

        private string GetTeacherNameByWorkloadDisciplineId(int teacherId)
        {
            var teacher = db.Teacher.First(t => t.Id == teacherId);
            return string.Concat(teacher.LastName, " ", teacher.FirstName);
        }

        private WorkloadDisciplineStudentsGroup GetWorkloadDisciplineStudentGroupByID(int
workloadDisciplineStudentGroupId)
        {
            return db.WorkloadDisciplineStudentsGroups
                .FirstOrDefault(wdsg => wdsg.ID == workloadDisciplineStudentGroupId);
        }

        private int GetCountOfStudentsInWorkloadDisciplineGroup(int
workloadDisciplineStudentGroupId)
        {
            var students = db.WorkloadDisciplineStudents
                .Where(wds => wds.WorkloadDisciplineStudentsGroupID ==
workloadDisciplineStudentGroupId);

            return students == null ? 0 : students.Count();
        }

        private string GetActivityTypeEnDescription(ActivityTypeEn activityType, bool genitiveCase = true)
        {
            switch (activityType)
            {
                case ActivityTypeEn.Lecture:
                    return genitiveCase ? "Лекції" : "Лекції";
                case ActivityTypeEn.Lab:
                    return genitiveCase ? "Лабораторних робіт" : "Лабораторні роботи";
                case ActivityTypeEn.Practical:
                    return genitiveCase ? "Практичних робіт" : "Практичні роботи";
                case ActivityTypeEn.Seminar:
                    return genitiveCase ? "Семінарів" : "Семінари";
                case ActivityTypeEn.Individual:
                    return genitiveCase ? "Індивідуальних занять" : "Індивідуальні заняття";
                case ActivityTypeEn.Consultation:
                    return genitiveCase ? "Консультацій" : "Консультації";
                case ActivityTypeEn.ExamConsultation:
                    return genitiveCase ? "Консультацій перед іспитом" : "Консультації перед
іспитом";
                case ActivityTypeEn.Exam:
                    return genitiveCase ? "Іспиту" : "Іспит";
            }

            return string.Empty;
        }
```

#endregion

GetDisciplіesForGenerateOrAssignment (додано):

```
@model List<TritonModel.Model.StudentsGroupAndSubGroup.DisciplineVM>
```

```
@using TritonModel.Services;

@{
    var method = "GenerateGroupsAndSubGroups";
    var header = "Згенерувати групи та/або підгрупи студенті в для виконання робіт";
    var buttonText = "Згенерувати";
    if(ViewData["IsAssignmentStudents"] != null && (bool)ViewData["IsAssignmentStudents"])
    {
        method = "AssignmentStudentsPage";
        header = "Оберіть дисциплі ну(-и) для розподілу студенті в по групам";
        buttonText = "Далі ";
    }
}

<h2>Дисциплі ни</h2>
<h4>@header</h4>
<style>
    form {
        padding-top: 15px;
    }

    .btn.btn-success.btn-block {
        width: unset;
        margin-top: 25px;
    }

    .error-message {
        color: red;
        font-style: italic;
    }
</style>

@using (Html.BeginForm(method, "Student", FormMethod.Post))
{
    <div>
        @Html.Hidden("GroupID", ViewData["GroupID"])
        @Html.Hidden("SemesterID", ViewData["SemesterID"])

        @if (!Model.Any())
        {
            <label class="error-message">Доступних дисцпиплі н для групи не знайдено</label>
        }
        else
        {
            <ul style="list-style: none;" name="disciplines">
                @for (int i = 0; i < Model.Count(); i++)
                {
                    <li>
                        @Html.HiddenFor(m => Model[i].DisciplineID)
                        @Html.HiddenFor(m => Model[i].OperationPlanID)
                        @Html.HiddenFor(m => Model[i].OperationPlanDisciplineID)
                        @Html.HiddenFor(m => Model[i].DisciplineName)
```

```
                        @Html.CheckBoxFor(m => Model[i].IsSelected, new { style = "width: 20px; height: 20px"
})
                        @Html.DisplayFor(m => Model[i].DisciplineName)
                    </li>
                }
            </ul>
            <input type="submit" class="btn btn-success btn-block" value=@buttonText />
        }

    </div>

    @Html.ActionLink("Повернутися", "ActionsByGroup", new { groupId = ViewData["GroupID"] },
new { @class = "btn btn-primary" })
}

@if (ViewData["ResultOfGenerating"] != null)
{
    <script>
        $(document).ready(function () {
            $('#groupsModal').modal('show');
        });

        setTimeout(function () {
            $('#Ok-btn').on('click', function () {
                $('#groupsModal').modal('hide');
            });
        }, 1000);

    </script>
}



<div id="groupsModal" class="modal fade">
    <div class="modal-dialog">
        <div class="modal-content">
            <div class="modal-header">
                <button type="button" class="close" data-dismiss="modal" aria-label="Close"><span aria-
hidden="true">&times;</span></button>
                <h4 class="modal-title">Результат генерації груп та підгруп</h4>
            </div>
            <div class="modal-body">
                @if (ViewData["ErrorMessage"] != null &&
!string.IsNullOrEmpty((string)ViewData["ErrorMessage"]))
                {
                    <div>
                        <label class="error-message">@Html.Raw(@ViewData["ErrorMessage"])</label>
                    </div>
                }
                <div>
                    @if (ViewData["ResultOfGenerating"] != null)
                    {
                        <ul>
                            @foreach (var groups in (Dictionary<string, List<Tuple<string,
int>>>)ViewData["ResultOfGenerating"])
                            {
```

```
                        <li>Для дисциплі ни "@groups.Key" створено: </li>
                        <ul>
                            @foreach (var group in groups.Value)
                            {
                                <li>@group.Item1 - @group.Item2 г руп(-и)</li>
                            }
                        </ul>
                    }
                </ul>
            }
        </div>
    </div>
    <div class="modal-footer">
        <button type="button" id="Ok-btn" class="btn btn-primary">Ok </button>
    </div>
</div><!-- /.modal-content -->
</div><!-- /.modal-dialog -->
</div><!-- /.modal -->
```

AssignmentStudentsPage (додано):

```
@model List<TritonModel.Model.StudentsGroupAndSubGroup.StudentAssignmentGroupVM>
@using TritonModel.Services;
@using Newtonsoft.Json;

@{
    @Styles.Render("~/Content/dropdown.css");
    @Scripts.Render("~/Scripts/dropdown.js");
}

<script>
    $('body').on('click', '.discipline-activityType', function () {
        var opdIdVal = $(this).attr('data-opdID');
        var acvitiyTypeVal = $(this).attr('data-activityType');

        FillDropDown('.groups-dropdown', '/Student/GetGroupsByDisciplineAndActivityType', {
operationPlanDisciplineId: opdIdVal, activityType: acvitiyTypeVal });
        FillDropDown('.students-dropdown', '/Student/GetNonAssignedStudent', {
operationPlanDisciplineId: opdIdVal, activityType: acvitiyTypeVal, groupId: $('#GroupID').val() });

        $('#assignmentStudentModal').modal('show');

        $('#saveAssignment').off('click', saveStudentAssignment).on('click', saveStudentAssignment);
    });

    $('#assignmentStudentModal').on('change', 'select[name="groups"]', function () {
        FillDropDown('.students-dropdown', '/Student/GetNonAssignedStudent', {
operationPlanDisciplineId: opdIdVal, activityType: acvitiyTypeVal, groupId: $('#GroupID').val() });
    });

    function FillDropDown(dropDownSelector, actionName, parameters) {
        $.getJSON(actionName, parameters, function (child) {

            var dropDownElem = $(dropDownSelector);
            dropDownElem.empty();
```

```
                $.each(child, function (_, child) {
                    dropDownElem.append($('<option/>', {
                        value: child.Value,
                        text: child.Text
                    }));
                });

            });
        }

        function saveStudentAssignment() {
            $.ajax({
                url: '/Student/AssignmentStudentToSpecificGroupsOrSubgroups',
                data: { studentId: $('#students-dropdown').val(), workloadDisciplineGroupId: $('#groups-
dropdown').val() },
                type: "POST",
                success: function (data) {
                    if (data.isSuccess) {
                        alert('Успішно збережено!');
                    } else {
                        alert(data.error);
                    }
                }
            });
        }
    </script>

    <div class="block">
        <input id="GroupID" value="@ViewData["GroupID"]" hidden />

        @foreach (var item in Model)
        {
            <div class="box">
                <div class="dropdown-h3">
                    @item.Discipline.DisciplineName
                    <span class="expand glyphicon glyphicon-plus"></span>
                </div>
                <ul class="dropdown-list">
                    @foreach (Tuple<ActivityTypeEn, string> activityType in item.ActivityTypes)
                    {
                        <li class="discipline-activityType" data-
opdID="@item.Discipline.OperationPlanDisciplineID" data-activityType="@activityType.Item1">
                            <div class="box">
                                <div class="dropdown-h3">
                                    @activityType.Item2
                                </div>
                            </div>
                        </li>
                    }
                </ul>
            </div>
        }
    </div>

    <div id="assignmentStudentModal" class="modal fade">
        <div class="modal-dialog">
```

```
            <div class="modal-content">
              <div class="modal-header">
                <button type="button" class="close" data-dismiss="modal" aria-label="Close"><span aria-
hidden="true">&times;</span></button>
                <h4 class="modal-title">Призначення студента в групу/підгрупу</h4>
              </div>
              <div class="modal-body">

                <input name="opdID" hidden />
                <input name="activityType" hidden />

                <div class="well">
                  <div class="row">
                    <label for="groups">Група</label>
                    <select name="groups" id="groups-dropdown" class="groups-dropdown form-
control">
                      <option></option>
                    </select>
                  </div>
                  <div class="row">
                    <label for="students">Студент</label>
                    <select name="students" id="students-dropdown" class="students-dropdown form-
control">
                      <option></option>
                    </select>
                  </div>
                </div>
              </div>
              <div class="modal-footer">
                <button type="button" class="btn btn-default" data-dismiss="modal">Відміна</button>
                <button type="button" id="saveAssignment" class="btn btn-
primary">Зберегти</button>
              </div>
            </div><!-- /.modal-content -->
          </div><!-- /.modal-dialog -->
        </div><!-- /.modal -->
```

DistributionByGroups (додано):

```
@model List<TritonModel.Model.StudentsGroupAndSubGroup.DistributionByGroupsVM>

@{
    @Styles.Render("~/Content/dropdown.css");
    @Scripts.Render("~/Scripts/dropdown.js");
}

<div class="row">
    @Html.ActionLink("Повернутись", "ActionsByGroup", "Student", null, new { @class = "btn btn-
info" })
</div>

<div class="block">
    @foreach (var item in Model)
    {
        <div class="box">
            <div class="dropdown-h3">
```

```
                @item.Discipline.DisciplineName
                <span class="expand glyphicon glyphicon-plus"></span>
            </div>
            <ul class="dropdown-list">
                @if (item.Groups == null || !item.Groups.Any())
                {
                    <span class="alert-warning">Розподіл відсутній</span>
                }
                else
                {
                    foreach (var activity in item.Groups.GroupBy(g => g.ActivityType))
                    {
                        <li>
                            <div class="box">
                                <div class="dropdown-h3">
                                    @activity.Key
                                </div>

                                @if (activity == null || !activity.Any())
                                {
                                    <span class="alert-warning">Групи відсутні для даного виду
занять</span>
                                }
                                else
                                {
                                    <ul class="dropdown-list">
                                        @foreach (var group in activity.ToList())
                                        {
                                            <div class="box">
                                                <div class="dropdown-h3">
                                                    Група #@group.GroupNumber (Викладач: @group.TeacherName)
                                                </div>

                                                @if (group.Students == null || !group.Students.Any())
                                                {
                                                    <span class="alert-warning">Студенті поки немає в цій
групі </span>
                                                }
                                                else
                                                {
                                                    <ul class="dropdown-list">
                                                        @foreach (var student in group.Students)
                                                        {
                                                            <li>@student</li>
                                                        }
                                                    </ul>
                                                }
                                            </div>
                                        }
                                    </ul>
                                }
                            </div>
                        </li>
                    }
                }
            </ul>
```

```
        </div>
    }
</div>
```