КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет комп'ютерних наук та кібернетики
Кафедра теорії та технології програмування

Кваліфікаційна робота
на здобуття ступеня магістра

за спеціальністю 122 Комп'ютерні науки
на тему:

РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ПЛАНУВАННЯ РОЗКЛАДУ ЗАНЯТЬ В ЗАКЛАДАХ ВИЩОЇ ОСВІТИ

Виконала студентка 2-го курсу магістратури
Мирошниченко Марія Миколаївна

(підпис)

Науковий керівник:
доцент, кандидат фіз.-мат. наук
Омельчук Людмила Леонідівна

_____
(підпис)

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

(підпис)

Роботу розглянуто й допущено до захисту на засіданні кафедри теорії та технології програмування
«____» _____ 2021 р., протокол № ___
Завідувач кафедри
М. С. Нікітченко _____
(підпис)

Київ – 2021

РЕФЕРАТ

Звіт до кваліфікаційної роботи магістра складається з вступу, 6 розділів, висновків, списку використаних джерел (25 найменувань) та 1 додатку.

Робота містить 16 рисунків, 10 таблиць. Загальний обсяг роботи становить 62 сторінок, основний текст викладено на 58 сторінках.

РОЗКЛАД, ЗАДАЧА ВИКОНАННЯ ОБМЕЖЕНЬ, ВЕБЗАСТОСУНОК, БАЗА ДАНИХ, SPRING FRAMEWORK, MONGODB.

Об’єктом дослідження є процес планування розкладу в закладах вищої освіти. Предметом дослідження є програмне забезпечення для планування розкладу в закладах вищої освіти.

Метою роботи є аналіз, проектування та розробка програмного забезпечення для планування розкладу в закладах вищої освіти.

Методи розроблення: комп’ютерне моделювання, комп’ютерне програмування, аналіз та синтез. Інструменти розроблення: інтегроване середовище розробки IntelliJ Idea, Spring Framework, мова програмування Java, шаблонізатор Freemarker та документоорієнтовна система керування базами даних MongoDB .

Результати роботи: в роботі представлено планування розкладу як задачу виконання обмежень та було розглянуто основні методи розв’язання задач виконання обмежень, зроблено загальний огляд методів розв’язання задач виконання обмежень, було застосовано один з методів для генерації розкладу на основі вхідних даних.. Розроблено вебзастосунок, що дозволяє зберігати згенерований результат до бази даних та відобразити його для користувача.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

API – Applied programming interface (прикладний програмний інтерфейс);

CRUD – Create, Read, Update and Delete (операції створення, читання, оновлення та видалення даних);

JSON – JavaScript Object Notation (текстовий формат обміну даними);

SQL – Structured query language (мова структурованих запитів);

NoSQL - non SQL (клас баз даних);

POJO – plain old Java object (простий старий Java-об’єкт);

MVC – Model-View-Controller (архітектурний шаблон);

HTTP – HyperText Transfer Protocol (протокол прикладного рівня передачі даних);

URL – Uniform Resource Locator (стандартизована адреса певного ресурсу);

REST – Representational State Transfer (архітектурний стиль розподілених компонентів додатку в мережі);

RESTful - системи, що підтримують REST;

UML – Unified Modeling Language (уніфікована мова моделювання);

БД – база даних;

ЗВО – задача виконання обмежень;

ПЗ – програмне забезпечення;

ПІБ – прізвище, ім’я по-батькові;

СКБД – система керування базами даних;

SHA – алгоритм криптографічного шифрування Secure Hash Algorithm;

MD5 – алгоритм хешування;

CSP – constraint satisfaction problem (задача виконання обмежень);

ВСТУП

Оцінка сучасного стану об’єкта розробки. Станом на сьогодні сфера застосування комп’ютерних наук невпинно зростає. Стрімкий розвиток апаратного забезпечення робить інформатизацію багатьох процесів ще більш доступною. Не є винятком і сфера освіти. Більшість навчальних закладів потребують роботи з розкладом, але, як правило, в закладах вищої освіти він, як правило, формується вручну. Автоматизація даного процесу знизила би і без того високе навантаження працівників освіти, що відповідають за цей процес. Таким чином, методист має змогу зберегти цінний ресурс часу, а студенти і викладачі отримують доступ до готового розкладу. Найбільш оптимальним з точки зору доступності є програмне забезпечення у вигляді вебзастосунку, адже воно не потребує попередньої інсталяції та краще задовольняє задачі відображення великої кількості даних про заняття.

Актуальність роботи та підстави для її виконання. На даний момент програмне забезпечення, яке б вирішувало поставлену проблему в Київському національному університеті імені Тараса Шевченка відсутнє. Саме тому було запропоноване вирішення поставленої проблеми шляхом розробки вебзастосунку, який надав би методистам інструмент для автоматичного складання розкладу, а студентам і викладачам інструмент для його перегляду.

Мета й завдання роботи. Основною метою кваліфікаційної роботи є розробка вебзастосунку, який надавав би методисту можливість генерувати розклад за завантаженими вхідними файлами, а студентам і викладачам переглядати готовий розклад. Для досягнення цієї мети поставлено наступні **завдання**:

- дослідити існуючі методи генерації розкладу за вхідними даними;
- обрати та реалізувати метод адаптований для предметної області закладу вищої освіти;
- дослідити існуючі системи планування розкладу;
- закріпити та поглибити теоретичні і практичні знання у сфері розробки вебзастосунків та використання баз даних;
- спроектувати та розробити зручний інтерфейс вебзастосунку для методистів, що дозволив би після авторизації завантажувати файли з вхідними даними та автоматично генерувати розклад;
- спроектувати та розробити зручний інтерфейс вебзастосунку для студентів, що дозволив би без авторизації переглядати розклад для груп;
- спроектувати та розробити базу даних для збереження даних про розклад;
- спроектувати та розробити вебзастосунок, що надавав би необхідний функціонал відповідно до ролі користувача.

Об'єкт, методи й засоби розроблення. Об’єктом розроблення вебзастосунку «Навчальний розклад» є процес планування та відображення розкладу в закладах вищої освіти.

Розробці програмного засобу передувало передувало дослідження систем автоматизації навчального процесу та формування розкладу та аналіз методів генерації розкладу.

Предмет дослідження – вебзастосунок «Навчальний розклад», призначений для формування розкладу на основі вхідних даних, відображення навчального розкладу для викладачів та здобувачів вищої освіти на освітніх програмах закладів вищої освіти.

Під час розробки програмного продукту використана еволюційна модель розробки. В роботі використано такі **методи дослідження**, як спостереження, порівняння, аналіз та синтез.

Можливі сфери застосування. Розроблений програмний продукт може застосовуватися для формування та відображення розкладу занять в закладах вищої освіти.

РОЗДІЛ 1

ОГЛЯД ІНФОРМАЦІЙНИХ СИСТЕМ, ПРИЗНАЧЕНИХ ДЛЯ ФОРМУВАННЯ РОЗКЛАДУ

1.1. Вебсайт MyTimetable

На сьогодні існує не так багато систем, які б дозволили автоматизувати складання розкладу у закладах вищої освіти. Більшість програмного забезпечення спрямована на автоматизацію інших процесів, або просто на відображення інформації. Розглянемо деякі такі системи.

Основною системою для перегляду розкладу є вебсайт https://mytimetable.live/ (рис. 1) [1].

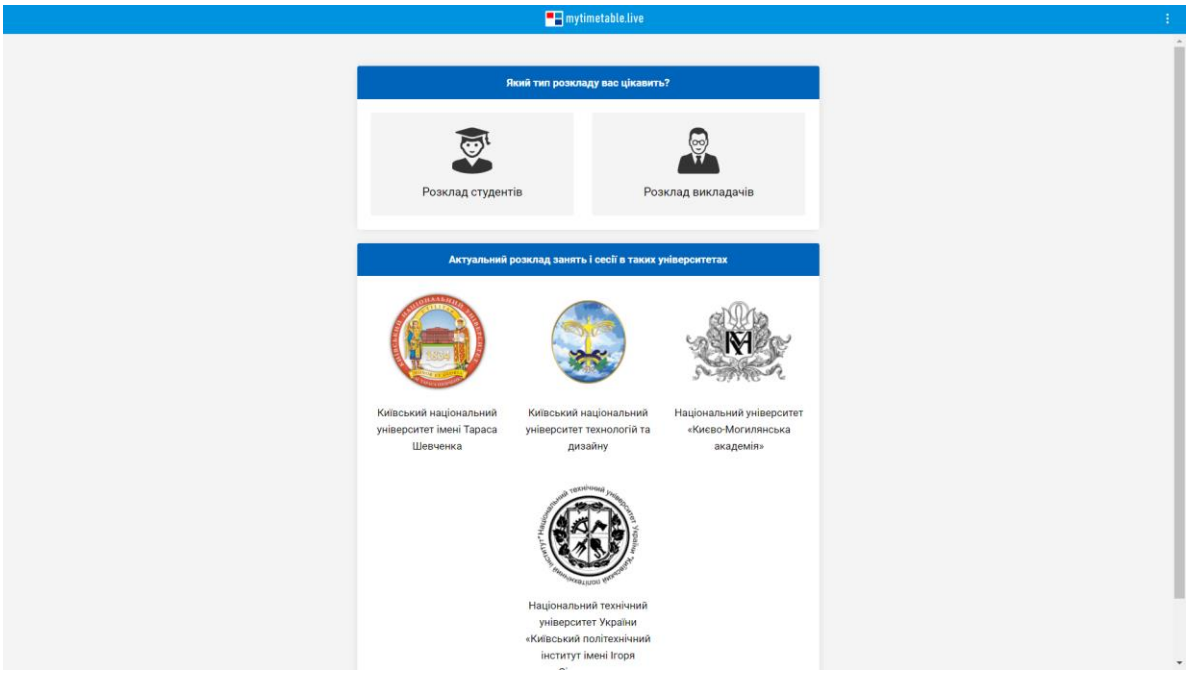

Рисунок 1 - Головна сторінка вебсайту mytimetable.live

Дана система дає можливість переглянути розклад для низки університетів. Варто відзначити, що розклад можливо переглянути як студентів, так і викладів, що є дуже зручною функцією.

Припустимо, ми хочемо побачити розклад для студентів. Обравши тип розкладу та університет, користувачу пропонується обрати факультет, групу і тоді відкривається сторінка з необхідним розкладом (див. рис. 2).

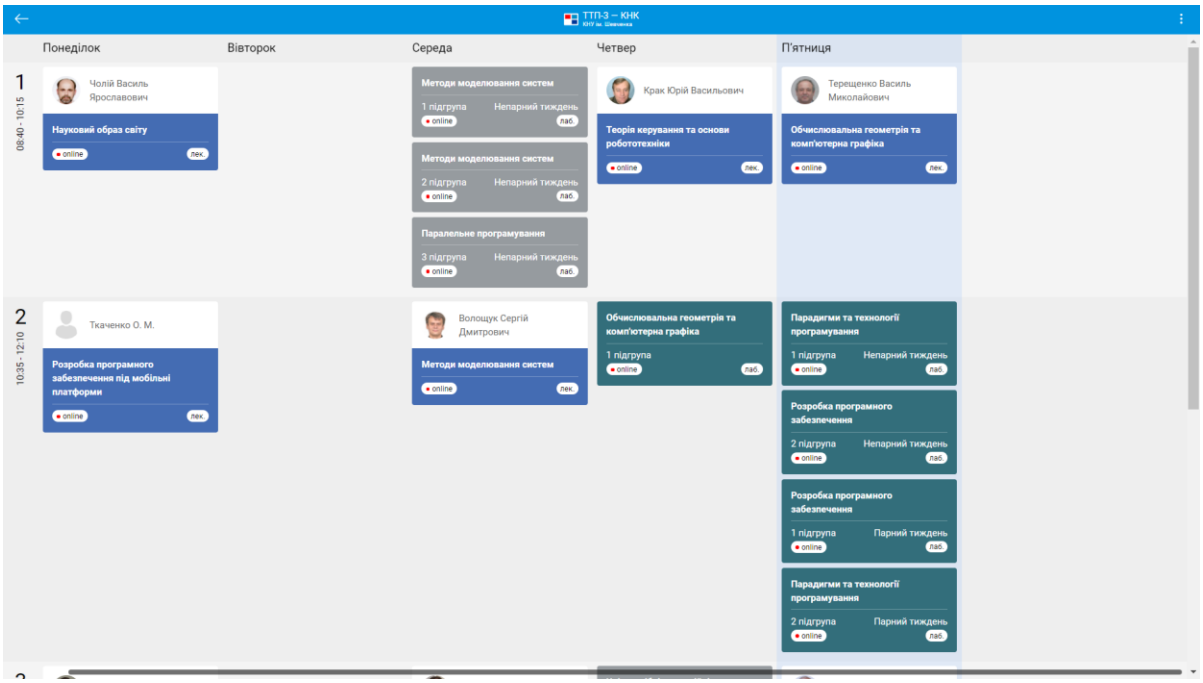

Рисунок 2 - Розклад для групи

З сильних сторін можна виділити наявність двох типів розкладу – для студентів та викладачів. Це є безперечним плюсом як для студентів, так і викладачів. Також перевагою є зручний інтерфейс. Варто відзначити, що той факт, що система є веб сайтом, а не мобільним додатком і відсутність авторизації надають високий рівень доступності для користувачів.

З недоліків можна виділити відсутність можливості складання розкладу вручну, або автоматично.

1.2. Система Triton

Розглянемо другу систему для навчальних закладів - https://triton.knu.ua/ (див. рис. 3, рис. 4) [2].

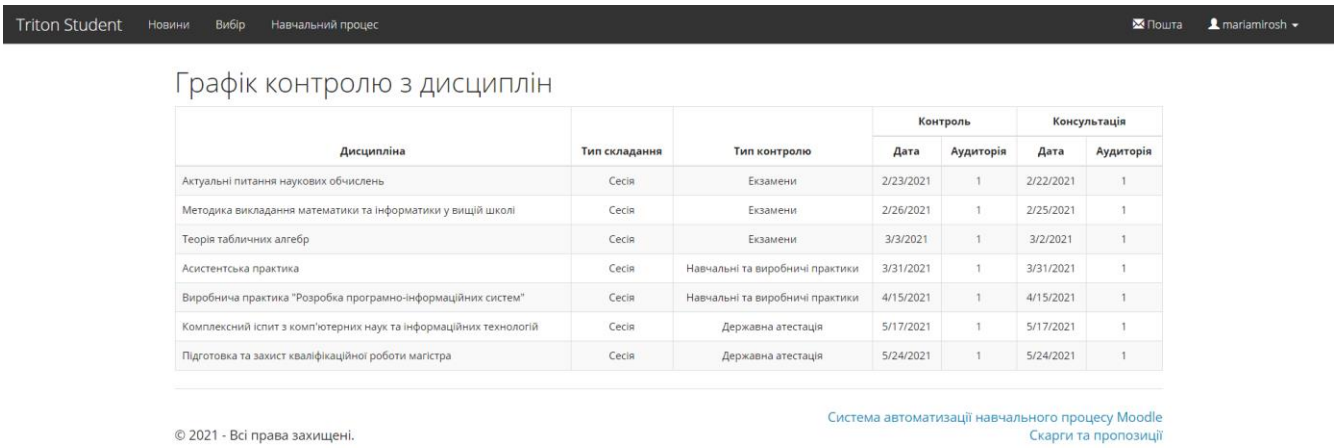

Рисунок 3 – Сторінка графіку контролю з дисциплін Triton

Дана система дозволяє переглядати інформацію про навчальний процес (графік сесії, вибір предметів, навчальні програми, тощо), вносити дані (наприклад вибір дисципліни за вибором) та деякі інші супутні процес (створення університетської корпоративної пошти).

З переваг можна виділити багатий функціонал та існування кількох паралельних підсистем для різних ролей – студентів, викладачів, деканату.

З недоліків можна виділити те, що хоч дана система і автоматизує деякі навчальні процеси, але не складання розкладу. В Triton немає підтримки розкладу.

1.3. Автоматизована система «Деканат»

Автоматизована система «Деканат» – це програмно-технологічний комплекс управління навчальним процесом навчального закладу, призначений для організації роботи методистів і зменшення кількості документації на паперових носіях [3].

Дана система має наступні функції пов’язані з начальним процесом:

- розробка навчальних, робочих планів на навчальний рік, закріплення навчальних груп за планами;
- генерація та відновлення робочих навчальних планів за навчальними планами;
- закріплення контингенту для навантаження по робочим навчальним планам;
- навантаження кафедри: агрегація і розподіл викладачами;
- індивідуальний робочий план викладача кафедри;
- створення розкладу, веброзкладу.

Крім цього, система має багатий функціонал пов’язаний з адмініструванням роботи зі співробітниками та студентами.

З переваг можна виділити велику кількість функцій специфічних для роботи деканату в вищому навчальному закладі. Плюсом є те, що можливе розширення даної системи шляхом інтеграції з іншими системами спрямованими на автоматизацію різноманітних процесів пов’язаних з вищими навчальними закладами. Наприклад, «АС «Приёмная комиссия»» та АС «Студгородок».

Із недоліків можна виділити достатньо завантажений інтерфейс для кожної з підсистем та відсутність функції автоматичного формування розкладу.

Дане програмне забезпечення розроблювалось з огляду на вищенаведену інформацію.

РОЗДІЛ 2
ОГЛЯД АЛГОРИТМІВ ТА МЕТОДІВ ФОРМУВАННЯ РОЗКЛАДУ

2.1. Постановка задачі

Для того щоб автоматизувати процес складання розкладу, необхідно спочатку детальніше розглянути предметну область розкладів та формалізувати її.

Розклад являє собою набір пар <заняття, час та місце>. Розглянемо детальніше кожний елемент. Заняття включає в себе наступну інформацію:

- назва дисципліни;
- група або групи, для яких проводиться заняття;
- викладач, який проводить заняття;
- тип заняття (лекційне, лабораторне, практичне, семінарське).

Варто зауважити, що заняття є однією конкретною парою, а не дисципліною в цілому.

Час та місце містять наступну інформацію:

- день (понеділок, вівторок і т.д.);
- номер заняття (перша пара, друга пара, тощо);
- номер аудиторії.

Так як для генерації розкладу ми маємо враховувати певні обмеження для кожного заняття, то тоді можна представити задачу складання розкладу як задачу виконання обмежень.

Задача виконання обмежень (ЗВО) – це тип математичних проблем, що визначаються як набір об’єктів, стан який має задовольняти певним обмеженням. ЗВО представляє сутності проблеми як однорідний набір обмежень, що накладаються на змінні, які розв'язуються методами виконання обмежень [4]. Дана задача є однієї з важливих задача штучного інтелекту.

Більш формально ЗВО представляє собою кортеж $P =< X, D, C >$, де

- $X = \{X_1, \dots, X_n\}$ - множина змінних;
- $D = \{D_1, \dots, D_n\}$ - множина доменів значень змінних;
- $C = \{C_1, \dots, Cm\}$– множина обмежень.

Кожна змінна X_i може приймати значення з непорожнього домену D_i. Кожне обмеження C_i в свою чергу є парою $< t_j, R_j >$, де $t_j \subset X$ є підмножиною k змінних, а R_j – k-арне відношення на відповідній підмножині доменів D_j. Оцінка v задовольняє обмеження $< t_j, R_j >$, якщо значення, що були присвоєні змінним t_j задовольняють відношенню R_j [5].

Оцінка є не суперечною, якщо вона не порушує жодного з обмежень. Оцінка є повною, якщо вона включає всі змінні. Оцінка – це розв’язок, якщо вона є не суперечною і повною. Така оцінка вирішує проблему задоволення обмежень.

Таким чином можна представити процес складання розкладу як ЗВО. Тоді кортеж $P =< X, D, C >$ матиме наступний вигляд:

- $X = \{X_1, \dots, X_n\}$ - множина занять.
- $D = \{D_1, \dots, D_n\}$ - множина можливих комбінацій часу та місця;
- $C = \{C_1, \dots, Cm\}$ - множина обмежень.

Розглянемо детальніше обмеження. Покладемо такі типи обмежень:

1. Місткість аудиторій. Це унарне обмеження, яка забезпечує, що для кожного заняття місткість аудиторії буде більшою або дорівнюватиме кількості студентів на занятті.

2. Дні роботи викладача. Це унарне обмеження, яка забезпечує, що для кожного заняття, кожний викладач зможе працювати в ті дні, які він вказав як зручні.

3. Не конфліктність часу для викладачів або студентів. Це бінарне обмеження. Не може існувати таких двох занять, які будуть проходити в один і той самий час, які проводяться одним і тим же викладачем, або для тих самих студентів.

2.2. Огляд алгоритмів вирішення задач виконання обмежень

Проблеми задоволення обмежень у скінченних доменах, як правило, вирішуються за допомогою певної форми пошуку. Найбільш використовувані методи – це варіанти пошуку з поверненням (backtracking), поширення обмежень (constraint propagation) та локальний пошук.

Пошук з поверненням – пошук в глибину, який вибирає значення для однієї змінної за раз і повертається назад, коли у змінної не залишилось жодних допустимих значень, які можна призначити [6].

Алгоритм пошуку з поверненням обирає не ініціалізовану змінну, а потім намагається по черзі спробувати всі значення в області значень цієї змінної, намагаючись знайти рішення. Якщо виявляється суперечність, алгоритм повертає помилку, в результаті чого попередня ітерація алгоритму намагається підібрати інше значення.

Алгоритму пошуку з поверненням не потрібно надавати специфічний для домену початковий стан, функцію дії, модель переходу або цільовий тест [7]. Алгоритм зберігає лише однин стан і змінює його, а не створює новий.

Зворотне маркування (backmarking) – працює, як пошук з поверненням за допомогою ітераційної оцінки змінних у заданому порядку, наприклад, $x_1, \dots, x_n$ [8]. Даний алгоритм покращує пошук з поверненням за рахунок підтримки інформації про останній раз, коли змінна була проініціалізована значенням та інформацію про те, що було змінено з того часу [9]. Зокрема:

- Для кожної змінної x_i та значення a алгоритм записує інформацію про те, коли востаннє x_i було присвоєно a. Зокрема він зберігає мінімальний індекс $i < j$, такий, після якого присвоєння було суперечним.

- Для кожної змінної x_i алгоритм зберігає інформацію про те, що змінилось з того часу, коли було оцінено x_i. Зокрема, зберігається мінімальний індекс $k < i$ змінної, яка була змінена з того часу.

На відміну від інших варіацій пошуку з поверненням, даний алгоритм не зменшує простір пошуку, але може зменшити кількість обмежень, що задовольняються частковим розв'язком.

Зворотний перехід (backjumping) дозволяє в деяких випадках зменшити простір пошуку. Основна відмінність полягає в тому, що повернення відбувається не на один крок, а одразу на декілька [10].

Вивчення обмежень (constraint learning) – це техніка підвищення ефективності, що діє за допомогою запису нових обмежень, коли знайдена суперечність. Нове обмеження може зменшити простір пошуку, оскільки наступні часткові оцінки можуть виявитися суперечливими без подальшого пошуку [11].

Пошук вперед (look ahead) також часто використовується при пошуку з поверненням. Особливість полягає в тому, що дана техніка намагається передбачити вибору змінної для оцінки одного з її значень. Одною з технік є forward cheking [12].

Методи поширення обмежень (constraint propagation) – це методи, що використовуються для модифікації ЗВО. Точніше, це методи, які забезпечують форму локальної сумісності, яка є умовами, пов'язаними з послідовністю групи змінних та / або обмежень. Поширення обмежень має різне використання. По-перше, це перетворює проблему на рівнозначну, але, як правило, простішу у вирішенні. По-друге, це може довести розв'язність чи нерозв'язність задачі. Це не гарантовано, однак завжди трапляється для деяких форм поширення обмежень та / або для певних видів задач. Найбільш відомими формами локальної сумісності є

сумісність дуги, послідовність гіпердуг та сумісність шляху. Найпопулярнішим методом поширення обмежень є алгоритм AC-3, який забезпечує сумісність дуги [13].

Роботу AC-3 можна представити такими кроками:

1. Перетворити кожну з констант на дві дуги.
2. Додати всі дуги до черги.
3. Повторювати, поки черга не стане порожньою:
    - Обрати та видалити дугу (x, y) з черги.
    - Для кожного значення з домену x, має бути деяке значення з домену y.
    - Зробити x сумісним з y. Для цього необхідно видалити всі значення з домену x, для яких немає відповідних значень в домені y.
    - Якщо домен x було змінено, додати всі дуги виду (k, x) до черги. Де k – змінна, відмінна від y, що має відношення до x.

Локальні методи пошуку працюють шляхом ітеративного вдосконалення повного присвоєння змінним. На кожному кроці змінюється значення невеликої кількості змінних, що має загальну мету збільшити кількість обмежень, задоволених цим присвоєнням. Алгоритм мінімальних конфліктів - це локальний алгоритм пошуку, специфічний для ЗВО, і заснований на цьому принципі. При даному початковому присвоєнні значень усім змінним ЗВО, алгоритм випадковим чином вибирає змінну із набору змінних із конфліктами, що порушують одне або кілька обмежень ЗВО. Далі він присвоює цій змінній значення, яке мінімізує кількість конфліктів.

Якщо є більше одного значення з мінімальною кількістю конфліктів, воно обирається випадковим чином.

Цей процес вибору випадкової змінної та присвоєння мінімальних значень конфлікту повторюється, поки не буде знайдено рішення або не буде досягнуто заздалегідь вибрану максимальну кількість ітерацій.

На задачі про n шахових королев, час виконання алгоритму мінімальних конфліктів не залежить від розміру проблеми. Він вирішує проблему навіть з мільйонами фігур за приблизно 50 кроків (після початкового присвоєння). Це спостереження стало стимулом для подальших досліджень.

Алгоритм мінімальних конфліктів також добре працює для вирішення складних проблем. Наприклад, він був використаний для планування спостереження за космічним телескопом Хаббл, зменшуючи час, необхідний для планування тижневого спостереження від трьох тижнів до приблизно 10 хвилин.

Згідно з дослідженням алгоритм мінімальних конфліктів показав себе як найбільш ефективний на ряді задач ЗВО [14]. Цей алгоритм і був обраний для вирішення поставленої задачі.

РОЗДІЛ 3
ОГЛЯД ВИКОРИСТАНИХ ТЕХНОЛОГІЙ

Для реалізації програмного забезпечення було обрано наступні технології: Spring Framework та СКБД MongoDB. Розглянемо технології детальніше.

3.1. Фреймворк Spring

Spring Framework (або коротко Spring) - це найпопулярніший фреймворк розробки додатків для Java. Мільйони розробників у всьому світі використовують Spring Framework для створення високопродуктивного, легко тестувального та багаторазового коду [15]. Він був написаний Родом Джонсоном і вперше був випущений за ліцензією Apache 2.0 у червні 2003 року. Spring - легкий, якщо говорити про розміри: базова версія Spring framework становить близько 2 МБ. Spring спрямований на полегшення використання J2EE та пропагує гарні практики програмування, зробивши можливою модель програмування на основі POJO [16].

Переваги використання Spring:

- Spring дозволяє розробникам розробляти додатки корпоративного класу за допомогою POJO. Перевага використання POJO полягає в тому, що не потрібен жодний додатковий контейнер, наприклад, сервер застосунків. Є можливість використовувати лише один надійний контейнер сервлетів, такий як Tomcat або якийсь інший комерційний продукт.

- Spring організований модульно. Незважаючи на те, що кількість пакетів та класів значна, доведеться турбуватися лише про ті, які потрібні, а решту ігнорувати.
- Spring використовує деякі існуючі технології, такі як ORM-фреймворки, фреймворки логування, JEE, Quartz та JDK та інші.
- Тестування програми, написаної на Spring, є простим, оскільки в цей фреймворк включено залежний від середовища код. Крім того, за допомогою POJO JavaBeanstyle стає легше використовувати ін'єкцію залежностей для ін'єкції тестових даних.
- Веб-фреймворк Spring - це добре продуманий веб-фреймворк MVC, який забезпечує альтернативу багатьом веб-фреймворкам.
- Spring надає зручний API для трансляції винятків, специфічних для технологій (наприклад, JDBC, Hibernate або JDO), у послідовні, неперевірені винятки [17].
- Spring надає уніфікований інтерфейс управління транзакціями, який може масштабуватися до локальної транзакції (наприклад, з використанням однієї бази даних) та масштабуватися до глобальних транзакцій.

В даній роботі було використано такі проекти Spring як:

1. Spring Boot – спрощена в плані конфігурації версія Spring.
2. Spring Security – потужний фреймворк, що надає велику кількість інструментів для авторизації та аутентифікації.

3.2. Система керування базами даних MongoDB

MongoDB – крос-платформна, документоорієнтована NoSQL система управління базами даних, що забезпечує високу продуктивність, доступність та масштабованість [18]. Замість того, щоб використовувати таблиці та рядки, як в традиційних реляційних БД, MongoDB виокристовує колекції та документи.

Документ складається з пар ключ-значення. Колекції містять набір документів і виконують ту ж задачу, що і таблиці в реляційних БД.

Кожна БД містить колекції, які, в свою чергу, містять документи. Кожний документ може відрізнятися кількістю полів. Розмір та вміст кожного документу також може різнитися [19]. Структура документу більше схожа на те, як розробники створюють класи та об’єкти за допомогою відповідних мов програмування. Тобто можна провести певну аналогію між документами MongoDB та класами, в яких чітко прослідковується структура ключ-значення.

Документи не мають мати наперед визначеної схеми даних. Натомість, поля можуть створюватися динамічно. Модель даних, що доступна в MongoDB дає можливість відобразити ієрархічні відношення, зберігати масиви та інші складні структури в простий спосіб.

Також, варто відзначити, що середовища MongoDB є масштабованими [20].

РОЗДІЛ 4

ПРИЗНАЧЕННЯ ТА ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ «НАВЧАЛЬНИЙ РОЗКЛАД»

4.1. Призначення і цілі створення програми формування розкладу

Призначенням програми є автоматизація процесу формування розкладу.

Застосунок автоматизації формування розкладу створюється з метою:

- забезпечення аналізу та обробки вхідних даних;
- генерації розкладу за вхідними даними;
- запису результату виконання генерації в базу даних;
- забезпечення відображення результатів у вебзастосунку;
- забезпечення авторизації для методистів з правами адміністратора.

В програмному продукті існує три ролі:

1. Користувач. Користувач має можливість лише переглянути розклад
2. Методист. Методист може авторизуватися та спланувати розклад.
3. Адміністратор. Адміністратор може підтверджувати реєстрацію методистів.

Діаграма прецедентів представлена на рисунку нижче (див. рис. 4).

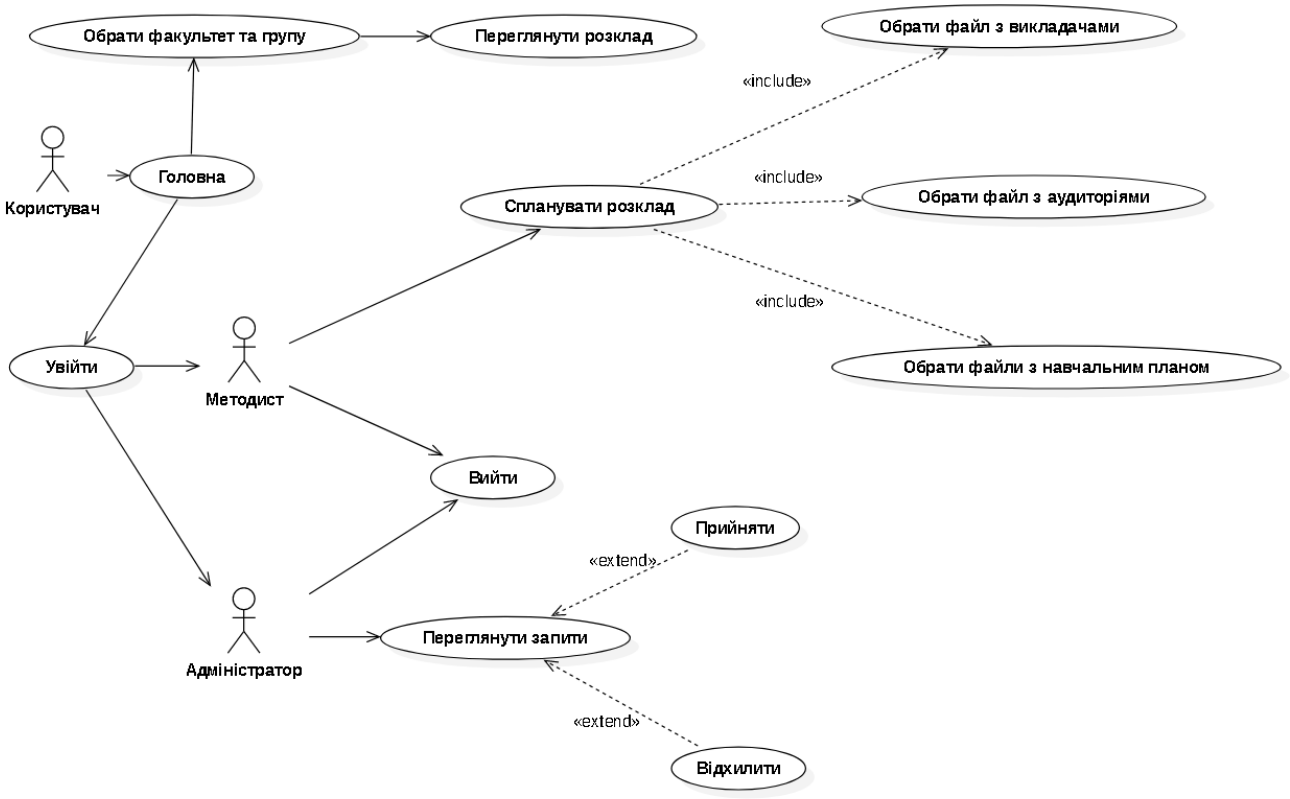

Рисунок 4 - Діаграма прецедентів

4.2. Вимоги до програмного продукту «Навчальний розклад»

4.2.1. Вимоги до структури та функціонування програмного продукту «Навчальний розклад»

Передбачається три функціональні підсистеми:

- Адміністративна, призначена для активації профіля методиста;
- Підсистема методиста, призначена для складання розкладу на основі вхідних даних;
- Підсистема користувача, призначена для перегляду розкладу за обраною групою.

4.2.2 Технічні вимоги до програмного продукту «Навчальний розклад»

Вебзастосунок повинен коректно відображатися в інтернет-браузерах Google Chrome, Edge, Mozilla версії 1.7. і вище, Opera версії 7.54 і вище.

На довільні некоректні дії користувача, пов’язані з веденням невірних даних, не заповнення обов’язкових полів введення в формах та інші, які можуть бути оброблені застосунком, генеруються відповідні повідомлення про помилки українською мовою, в межах загального дизайну вебзастосунку.

Джерелом даних мають бути:

- Наступні файли формату .xls:
  1. Файл, що містить інформацію про аудиторії:
     - унікальний код;
     - місткість.
  2. Файл, що містить інформацію про викладачів:
     - унікальний код;
     - ПІБ;

- пошта;
- робочі дні.

3. Файл, що містить навчальний план, відомості про групи та ставить у відповідність кожному заняттю викладача.

- СКБД MongoDB.

Розглянемо детальніше формат даних для кожного з файлів.

Файл з відомостями про аудиторії (див. рис. 5).

	A	B
1	**id**	**Місткість**
2	1	300
3	2	30
4	3	100

Рисунок 5 - Файл з відомостями про аудиторії

Файл з відомостями про вчителів (див. рис. 6).

	A	B	C	D
1	id	ПІБ	Пошта	Дні роботи
2	В-1	Криволап Андрій Володимирович	krivolap@email.com	ПН ВТ СР ЧТ ПТ
3	В-2	Марченко Олександр Олександрович	marchenko@email.com	ПН ВТ СР ЧТ ПТ
4	В-3	Анісімов Анатолій Васильович	anisimov@email.com	ПН ВТ СР ЧТ ПТ

Рисунок 6 - Файл з відомостями про вчителів

Файл з відомостями про навчальний план (див. рис. 7).

Тижнів: 14																
№	Шифр	Назва	К-ть навчальних занять				Викладачі									
								Лаб / Сем / Прак								
			Лек	Лаб	Сем	Прак	Лек	К-14		К-15		К-16		К-17		
								1	2	1	2	1	2	3	4
1	ОК.01	Вступ до університетських студій	28	0	0	0	В-10								
2	ОК.13	Програмування	28	28	0	0	В-12	В-12	В-13	В-12	В-13	В-12	В-13	В-14	В-15
3	ОК.06	Іноземна мова	0	0	0	84		В-49	В-50	В-49	В-50	В-49	В-50	В-51	В-52
4	ОК.10	Алгебра та геометрія	42	0	0	28	В-26	В-27	В-28	В-27	В-28	В-27	В-28	В-29	В-30
5	ОК.11	Математичний аналіз	42	0	0	42	В-18	В-17	В-22	В-19	В-16	В-17	В-22	В-19	В-16
6	ОК.12	Дискретна математика	28	0	0	28	В-55	В-56	В-57	В-58	В-59	В-56	В-57	В-58	В-59

Рисунок 7 - Файл з відомостями про навчальний план

РОЗДІЛ 5

РЕАЛІЗАЦІЯ ПРОГРАМНОГО ПРОДУКТУ «НАВЧАЛЬНИЙ РОЗКЛАД»

5.1. Структура бази даних

В ході розробки вебзастосунку була використана СКБД MongoDB. Дані записуються у базу даних «timetabledb» у три колекції (аналог таблиць в реляційних СКБД). Структура бази даних «timetabledb» наведена у таблиці 1.

Таблиця 1 - Загальна структура бази даних «timetabledb»

Номер	**Колекція**	**Опис**
1	timetable	Колекція для збереження згенерованого розкладу
2	faculties	Колекція для збереження списку факультетів та груп, що їм належать
3	users	Колекція для збереження авторизаційних даних для адміністратора та методиста

5.1.1. Опис бази даних

Наведемо опис колекції «timetable» у таблиці 2. Дана колекція використовується для зберігання розкладу у вигляді пари <заняття, місце та час>.

Таблиця 2 - Колекція «timetable»

Поле	**Тип**	**Опис**
lesson	object	Заняття
lesson.courseId	String	Шифр дисципліни
lesson.courseName	String	Назва дисципліни
lesson.LessonType	String	Тип заняття
lesson.daysPerWeek	String	Кількість занять на тиждень
lesson.numOfStudents	int	Кількість студентів
lesson.groups	Array	Групи
lesson.groups.groupId	String	Ідентифікатор групи
lesson.groups.faculty	String	Факультет
lesson.groups.specilty	String	Спеціальність
lesson.groups.numOfStudents	int	Кількість студентів
lesson.groups.subgroups	Array	Кількість підгруп
lesson.groups.subgroups.parentId	String	Ідентифікатор групи
lesson.groups.subgroups.numOfSubgroup	int	Кількість студентів
lesson.teacher	Object	Викладач
lesson.teacher.teacherId	String	Ідентифікатор
lesson.teacher.name	String	ПІБ
lesson.teacher.email	String	Пошта
lesson.teacher.preferedDays	Array	Робочі дні

lesson.name	String	Ідентифікатор заняття
roomtimeslot	Object	Час та місце
roomtimeslot.day	int	День тижня
roomtimeslot.timeslot	int	Номер пари
roomtimeslot.room	Object	Аудиторія
roomtimeslot.room.id	String	Назва
roomtimeslot.room.capacity	int	Місткість

Розглянемо структуру колекції «groups» у таблиці 3.

Таблиця 3 - Колекція «groups»

Поле	**Тип**	**Опис**
groupId	String	Ідентифікатор групи
faculty	String	Факультет
specialty	String	Спеціальність
numOfStudents	int	Кількість студентів
subgroups	Array	Кількість підгруп
subgroups.parentId	String	Ідентифікатор групи
subgroups.numOfSubgroup	int	Кількість студентів

Розглянемо структуру колекції «users» у таблиці 4.

Таблиця 4 - Колекція «users»

Поле	**Тип**	**Опис**
email	String	Логін користувача
password	String	Зашифрований пароль
faculty	String	Факультет
active	Boolean	Чи підтверджено профіль адміністратором

5.2. Загальна архітектура програмного продукту «Навчальний розклад»

В ході проектування програми було використано архітектурний шаблон MVC (див. рис. 8). Model-View-Controller (MVC, «Модель-Представлення-Контролер», «Модель-Вид-Контролер») - схема поділу даних програми, призначеного для користувача інтерфейсу і керуючої логіки на три окремих компоненти: модель, уявлення і контролер - таким чином, що модифікація кожного компонента може здійснюватися незалежно [21].

- Модель (Model) надає дані і реагує на команди контролера, змінюючи свій стан.
- Подання (View) відповідає за відображення даних моделі користувачеві, реагуючи на зміни моделі.
- Контролер (Controller) інтерпретує дії користувача, сповіщаючи модель про необхідність змін.

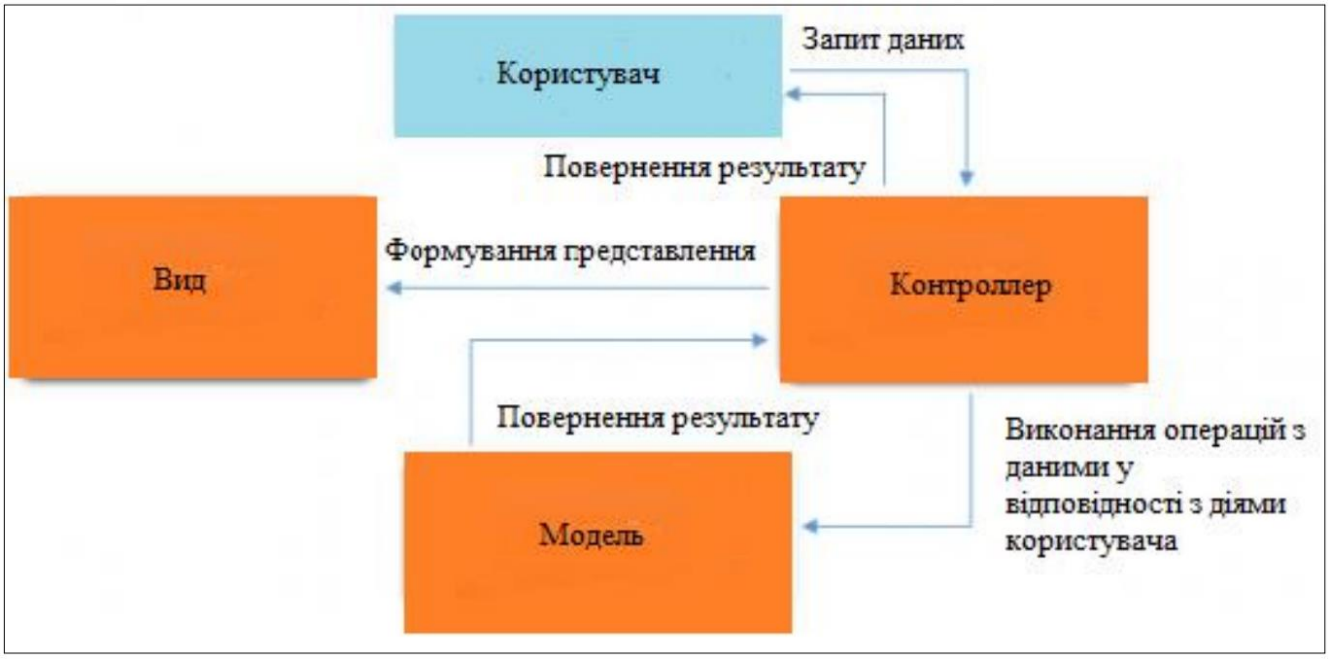

Рисунок 8 - Архітектура MVC

Розглянемо детальніше модель на прикладі діаграми класів (див. рис. 9).

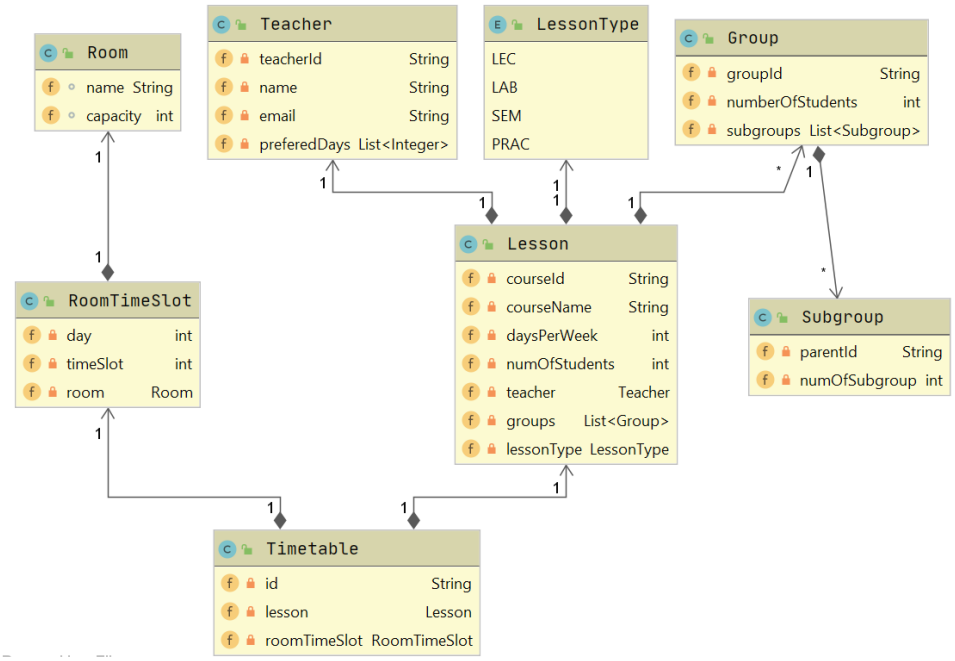

Рисунок 9 - Діаграма класів

Діаграму не сутнісних класів можна переглянути у додатку А.

Для проектування та реалізації вебзастосунку також було використано архітектурний стиль REST. Це стиль архітектури програмного забезпечення для розподілених систем, що як правило використовується для побудови веб-служб.

Системи, що підтримують REST, називаються RESTful-системами. У загальному випадку REST є простим інтерфейсом управління інформацією без використання додаткових внутрішніх прошарків.

Кожна одиниця інформації однозначно визначається глобальним ідентифікатором, таким як URL. Кожна URL в свою чергу має строго заданий формат. Як відбувається управління інформацією сервісу цілком і повністю залежить від протоколу передачі даних. Найбільш поширений протокол - це HTTP. Для HTTP дія над даними задається за допомогою методів: GET (отримати), PUT

(додати, замінити), POST (додати, змінити, видалити), DELETE (видалити). Таким чином, дії CRUD (Create-Read-Updtae-Delete) можуть виконуватися як з усіма 4-ма методами, так і тільки за допомогою GET і POST [22].

5.3. Реалізація алгоритму мінімальних конфліктів

Розглянемо псевдокод алгоритму мінімальних конфліктів, де failure позначає те, що розв'язок не був знайдений:

algorithm MIN-CONFLICTS **is**

input: *csp* - ЗВО,

max_steps – кількість кроків,

current_state – початкове присвоєння,

output – множина значень для змінних або failure:

for *i* ← 1 **to** *max_steps* **do**

if *current_state* є рішенням *csp* **then**

return *current_state*

set *var* ← випадкова змінна з множини коонфліктуючих змінних CONFLICTED[*csp*]

set *value* ← значення *v* для змінної *var* що мінімізує конфлікти CONFLICTS(*var,v,current_state,csp*)

set *var* ← *value* **in** *current_state*

return *failure*

Розглянемо детальніше реалізацію на Java.

Було створено наступні класи:

- CSP;
- Variable;
- Value;
- Domain;

- Constraint;
- Allocation;
- Lesson;
- RoomTimeslot;
- InputParser;
- CSPSolver.

Розглянемо детальніше деякі з них. Клас Allocation представляє собою деяке присвоєння і містить єдине поле variabaleValue типу HashMap<Variable, Value>, яке ставить у відповідність кожній змінній значення.

У таблиці 5. можна побачити опис класу CSP. Даний клас необхідний для того, щоб описати ЗВО.

Таблиця 5 - Опис класу CSP

Поле	**Тип**	**Опис**
variables	List<Variable>	список змінних
domains	List<Domain<Value>>	список доменів
constraints	List<Constraint<Variable, Value>>	список обмежень
varConstraint	Hashtable<Variable, List<Constraint<Variable, Value>>>	таблиця, ставить у відповідність змінним значення

Класи Variable, Value є загальними. На їх місце підставляються їх нащадки. Для Variable Lesson, Value RoomTimeSlot. Constraint – це інтерфейс, який імплементують наступні класи - CapacityConstraint, PreferedDayConstraint, LessonTimeConstraint, LecturePracticeConstraint, PracticeConstraint.

Розглянемо клас Lesson у таблиці 6 та інші класи, що використовуються в ньому у таблицях 7, 8 та 9.

Таблиця 6 - Опис класу Lesson

Поле	**Тип**	**Опис**
lessonId	String	унікальний код дисципліни
lessonName	String	назва дисципліни
lessonType	LessonType	тип заняття, що приймає значення: LEC (лекційне), LAB (лабораторне), PRAC (практичне), SEM (семінарське).
daysPerWeek	int	кількість занять на тиждень
numOfStudents	int	кількість студентів, що має відвідувати заняття
groups	List<Group>	перелік груп, що мають відвідувати заняття
subGroup	int	номер підгрупи, де 0 – відсутній поділ на підгрупи, 1 та більше – позначає номер підгрупи
teacher	Teacher	викладач

Таблиця 7 - Опис класу Group

Поле	**Тип**	**Опис**
groupId	String	ідентифікатор групи
numberOfStudents	int	кількість студентів
subGroups	List<Subgroup>	перелік підгруп

Таблиця 8 - Опис класу SubGroup

Поле	**Тип**	**Опис**
parentId	String	ідентифікатор групи, якій належить підгрупа
numOfSubgroups	int	кількість підгруп

Таблиця 9 - Опис класу Teacher

Поле	**Тип**	**Опис**
teacherId	String	ідентифікатор вчителя
name	String	ПІБ
email	String	пошта
preferredDays	List<Integer>	перелік робочих днів

Розглянемо детальніше клас-нащадок RoomTimeSlot у таблиці 10.

Таблиця 10 - Опис класу RoomTimeSlot

Поле	**Тип**	**Опис**
day	int	номер дня тижня
timeslot	int	номер пари
room	Room	аудиторія

Клас Room містить два поля: name – назва аудиторії та capacity – місткість.

Далі, оглянемо кожного з нащадків класу Constraint. В кожному з цих класів є метод isSatisfied з параметром Allocation<Variable, Value> a. Він повертає істину тоді, коли виповнюється певна умова обмеження. Є два типи обмежень: унарні та бінарні.

До унарних обмежень належать:

1. CapacityConstraint. Обмеження накладається на кожну змінну. Його можна представити таким чином: ∄ a ∈ Allocation: numberOfStudent ∈ Lesson ∈ a > capacity Room ∈RoomTimeSlot ∈ a. Тобто забезпечується те, що аудиторія має вмістити всіх студентів.

2. PreferedDayConstraint. Обмеження накладається на кожну змінну. Воно забезпечує те, що пара, яку веде певний викладач буде відбуватися в один з його робочих днів.

Наведемо бінарні обмеження:

1. LessonTimeConstraint. Дане обмеження накладається лише на ту пару змінних, в якій одна з них є лекційним заняттям. Воно забороняє проведення будь-яких двох занять в один і той самий час та місці, якщо у цій парі приймають участь або одні й ті ж самі студенти, або однаковий виклад.

2. PracticeConstraint. Дане обмеження накладається лише на ту пару змінних, в якій обидва заняття є не лекційними. Обмеження дозволяє проведення двох занять одночасно за таких умов:

а). Якщо йдеться про спільний предмет;

б). Якщо йдеться про одну й ту ж саму підгрупу;

в). Якщо цю пару занять ведуть різні викладачі;

г). Якщо заняття будуть проведені в різних аудиторіях.

Інакше, заняття мають відбутися у різний час.

Наступний клас – InputParser. Даний клас відповідає за зчитування даних з вхідних файлів.

Перейдемо до реалізації самого алгоритму мінімальних конфліктів. Йому відповідає один з методів класу CSPSolver. Клас містить одне поле – кількість кроків до завершення пошуку розв'язку. Розглянемо методи цього класу. Почнемо з головного методу solve, що втілює основу алгоритму мінімальних конфліктів.

```
public Optional<Allocation<Variable, Value>> solve(CSP< Variable, Value > csp)
{
        Allocation< Variable, Value > alloc = generateAllocation(csp);
        for (int i = 0; i < max_steps; i++) {
            if (alloc.isSolution(csp)) {
                return alloc;
            } else
                {
                Set<Variable> vars = getConflicts(alloc, csp);
                alloc.add(getRandom(vars), getMinConflict(var, alloc, csp));
            }
        }
        return alloc;
    }
```

Метод generateAllocation(CSP csp) має наступний вигляд:

```
private Allocation< Variable, Value > generateAllocation(CSP< Variable, Value >
csp) {
        Allocation< Variable, Value > result = new Allocation<>();
        for (Variable var : csp.getVariables()) {
```

```
            result.add(var, GetRandom(csp.getDomain(var).asList()));
        }
        return result;
    }
```

Оглянемо метод isSolution(CSP csp) Allocation. Як було вже згадано, щоб присвоєння було розв'язком воно має бути не суперечним та повним. Щоб розв'язок був не суперечним необхідно щоб виконувалися всі обмеження. Методом, що перевіряє не суперечність є метод класу Allocation isConsistent:

```
public boolean isConsistent(List<Constraint< Variable, Value >> cons) {
        for (Constraint< Variable, Value > c : cons )
            return c.isSatisfiedWith(this) ;
    }
```

Метод повертає істину, якщо всі обмеження виконані.

Для перевірки повноти розв'язку було використано метод класу Allocation isComplete:

```
    public boolean isComplete(List< Variable > variables ) {
        for (Variable v : variables)
            return contains(v);
    }
```

Метод повертає істину, якщо всім змінним було присвоєно значення.

Метод getConflicts обирає конфліктні змінні з переліку змінних.

Метод getMinConflict обирає найменш конфліктне значення і має наступний вигляд:

```
private Value getMinConflict(Assignment< Variable , Value > assignment, CSP< Variable , Value > csp, Variable variable) {
        Assignment< Variable , Value > assignment1 = assignment.clone();
        List< Value > result = new ArrayList<>();
        for (Value value : csp.getDomain(variable )) {
            assignment1.add(variable , value);
            int current = countConflicts(assignment1, csp.getConstraints(var););
           int min = 0;
            if (current <= min ) {
                    if (current != min ){
                            result .add(value);
                    }
                        min = current;
                        result .clear();
            }
        }
           if (result .isEmpty())
                    return null;
           else
                    return getRandom (result);
```

```
}
```

В даному методі функція countConflicts підраховує кількість конфліктів.

Таким чином було реалізовано алгоритм мінімальних конфліктів для вирішення ЗВО. Для підвищення модульності та наочності було створено окремі класи для кожної з сутностей.

5.4. Реалізація вебзастосунку

5.4.1. Авторизація та реєстрація

Як згадувалося раніше, система передбачає два види користувачів:

1. Адміністратор.
2. Методист
3. Звичайний користувач.

Перші дві потребують авторизації.

Для реалізації авторизації та автентифікації використовується Spring Security. Було додано та налаштовано клас WebSecurityConfig, що наслідує WebSecurityConfigurerAdapter. Розглянемо детальніше його конфігурацію. В методі configure(HttpSecurity http) встановлюються ресурси, що не потребують авторизації та сторінка реєстрації.

```
@Override
protected void configure(HttpSecurity http) throws Exception {
    http
            .authorizeRequests()
                .antMatchers("/", "/registration").permitAll()
                .antMatchers("/static/**").permitAll()
                .anyRequest().authenticated()
            .and()
                .formLogin()
                .loginPage("/login")
                .permitAll()
```

```
                .and()
                    .logout()
                    .permitAll();
    }
```

Метод configure(AuthenticationManagerBuilder auth) встановлює джерело авторизаційних даних та шифрувальник паролів.

```
    @Override
        protected void configure(AuthenticationManagerBuilder auth) throws Exception {
            auth.userDetailsService(userService)
                    .passwordEncoder(getPasswordEncoder());
    }
```

Метод getPasswordEncoder() повертає шифрувальник BCryptPasswordEncoder.

BCryptPasswordEncoder – це імплементація шифрувальника, що використовує адаптивну криптографічну хеш-функцію формування ключа bcrypt [23].

BCryptPasswordEncoder – це клас, який реалізує PasswordEncoder і забезпечує хешування паролів. Це кодер, який використовує випадкові 16 байт солі та алгоритм bcrypt. В алгоритмі Bcrypt кількість обчислень навмисно, в порівнянні з обчисленнями стандартних алгоритмів. Отже, у порівнянні зі стандартними алгоритмами шифрування (SHA, MD5), він має деякі особливості, що дозволяють протистояти «атаці грубої сили».

Розглянемо джерело автентифікаційних даних клас UserService.

UserService – це сервісний клас позначений анотацією @Service та імплементуючий інтерфейс UserDetailsService, що є частиною Spring Security.

Даний клас надає функціонал пошуку користувачів та реєстрації нових.

Користувач представлений сутністю User і має наступні поля:

- login;
- password;
- faculty;
- active.

5.4.2. Реалізація основного функціоналу програмного продукту «Навчальний розклад»

Для реалізації архітектурного шаблону MVC необхідно виділити модель, контролер та представлення. Діаграма класів моделі була розглянута на рисунку 8 у розділі 5.2. Оглянемо спочатку представлення.

5.4.2.1. Представлення

В якості представлень було використано шаблонізатор Freemarker. Даний шаблонізатор написано на Java, він дозволяє динамічно генерувати HTML сторінки та є популярним для втілення MVC архітектури.

Наприклад, додамо до традиційного HTML синтаксису опису випадаючого списку конструкцію Freemarker, отримаємо наступний код:

```
<ul class="dropdown-menu" aria-labelledby="navbarDropdown">
        <#list groups as gr>
            <li><a class="dropdown-item" href="/timetable/${gr}">${gr}</a></li>
        </#list>
```

```
</ul>
```

Дана конструкція дає змогу разом із GET-запитом проініціалізувати об’єкт group списком груп і кожний з елементів списку group відобразити як елемент випадаючого списку. В метод класу-контролера необхідно передати параметр model типу Model та додати рядок такого вигляду:

```
model.addAttribute("groups", listOfGroups),
```

де listOfGroups – список об’єктів груп.

5.4.2.2. Доступ до бази даних

Доступ до бази даних MongoDB здійснюється за допомогою допоміжних класів:

1. UserRepository. Містить наступні методи:
   - findUserByLogin (String login). Пошук користувача за логіном.
   - save (User user). Створення або оновлення користувача.
   - updatePassword(String login, String password). Оновлення паролю для користувача.
2. GroupsRepository. Містить наступні методи:
   - findAll(). Пошук списку груп.
3. TimetableRepository. Містить наступні методи:
   - findByTimeslotAndGroup (int day, int timeslot, String group). Пошук занять за групою та часом.
   - findByGroup(String group). Пошук занять за групою.

Кожний з класів відповідає за колекцію з бази даних. Класи такого типу забезпечують підключення та звернення до бази даних.

Наведемо приклад запиту пошуку по колекції «timetable» занять за групою та часом:

```
public List<Timetable<Lesson, RoomTimeSlot>> findByTimeslotAndGroup(int day, int timeslot, String group){
        List<Timetable<Lesson, RoomTimeSlot>> timetableList = new ArrayList<>();
        FindIterable<Document> documentList = timetableCollection.find(and(eq("roomTimeSlot.day", day),
            eq("roomTimeSlot.timeSlot", timeslot),
            eq("lesson.groups.groupId", group)));
        for(Document d : documentList){
           Timetable<Lesson, RoomTimeSlot> timetable= (Timetable<Lesson, RoomTimeSlot>) new Gson().fromJson(d.toJson(), Timetable.class);
           timetableList.add(timetable);
        }
        return timetableList;
    }
```

5.4.2.3. Сервісні класи вебзастосунку

Перед тим, як перейти до контролерів, необхідно описати додаткові сервісні класи.

У Spring Framework сервісом називається клас, що надає певну бізнес функціональність і позначається анотацією @Service.

Виділимо наступні сервіси:

- UserService;

- TimetableService;
- GroupsService;
- GeneratorService.

UserService – сервіс, що відповідає за дії з користувачами: авторизація, реєстрація та зміна паролю користувача. Містить наступні методи:

- findUserByLogin;
- saveUser;
- checkIfValidOldPassword;
- updatePassword.

TimetableService – сервіс, що виконує пошук по занять. Містить два методи:

- findByTimeslotAndGroup;
- findByGroup.

GroupsService – сервіс, що виконує пошук груп. Містить один метод:

- findAll.

GeneratorService – сервіс, що генерує розклад за вхідними файлами та записує його до бази даних. Містить єдиний метод:

- generateTimetable.

Наведемо реалізацію методу findByTimeslotAndGroup сервісу TimetableService:

```
@Autowired
    TimetableRepository timetableRepo;
```

```
public List<Timetable<Lesson, RoomTimeSlot>> findByTimeslotAndGroup(int day, int time, String group){
    return TimetableRepository.findByTimeslotAndGroup(day, time, group);
}
```

5.4.2.4. Контролери вебзастосунку

Тепер розглянемо контролери. В Spring Framework клас-контролер позначається анотацією @Controller.

Контролерів можна виділити три класи:

1. RegistrationController.
2. TimetableController.
3. AdminController.

Розглянемо кожний з них окремо.

RegistrationController необхідний для проведення реєстрації. Даний контролер складається з двох методів, кожному з яких відповідає певний HTTP-запит. За допомогою анотацій @GetMapping (<<Шлях запиту>>) та @PostMapping (<<Шлях запиту>>) перед методом, можна вказати якого типу запит буде виконувати метод.

GET-метод необхідний для відображення форми реєстрації. POST-метод створює нового користувача за параметрами реєстраційної форми.

TimetableController відповідає за відображення розкладу. Містить всього два методи. Обидва з них є GET -методами. Метод getTimetableByGroup передає для відображення в модель шаблонізатора дані про групи та заняття:

```
@GetMapping("/timetable/{groupId}")
public String getTimetableByGroup(@PathVariable(value="groupId") String id, Model model){
```

```
            model.addAttribute("id", id);
            model.addAttribute("groups", groupsServices.findAll());
            String [] days = {"mon", "tue", "wed", "thu", "fri"};


            for(int i=1; i<numberOfLessons; i++){
                for(int j=1; j<=days.length; j++){
                    String name = days[j-1]+ i;
                    model.addAttribute(name,
timetableService.findByTimeslotAndGroup(j, i, id));
                }
            }
            return "timetable";
```

Таким чином модель містить набір атрибутів з іменами формату mon1, mon2 і т.д. для кожного з занять і відповідний їм об’єкт типу Timetable з відомостями про заняття.

Другий метод при GET-запиті за шляхом /timetable/ переадресовує запит на одну з існуючих груп або показує сповіщення про те, що наразі не додано розклад для жодної групи.

Клас-контролер MethodistController відповідає за дії пов’язані з правами методиста, зокрема відображення сторінки методиста, додавання файлів та запуск автоматичного планування розкладу.

Щоб гарантувати те, що сторінка методиста буде доступною лише користувачеві з правами методиста, Spring Framework передбачив спеціальну

анотацію «@PreAuthorize("hasAuthority(<<Назва авторизаційної ролі>>)")». В нашому випадку такою роллю буде «METHODIST».

За додавання файлів та запуск відповідає POST-метод:

```
@PostMapping("/methodist ")
public String uploadFilesAndGenerate(Model model, File teachersFile, File roomsFile, File planFile){
    model.addAttribute("groups", groupsServices.findAll());
    if (!generatorService.generateTimetable(teachersFile, roomsFile, planFile)){
        model.addAttribute("message", "Не вдалося створити розклад");
    }
    model.addAttribute("message", "Розклад створено");
    return "methodist ";
}
```

В атрибут message передається повідомлення, що відображається на тій самій сторінці та вказує на успішність операції створення розкладу.

Таким чином було створено зручний вебзастосунок з використання архітектурного підходу MVC. Було реалізовано механізми авторизації та реєстрації за допомогою одного з проектів Spring – Spring Security. Було забезпечено надійне зберігання паролей користувачів у зашифрованому вигляді. Також був створений механізм верифікації користувачів адміністратором для уникнення отримання неправомірного доступу до функціоналу методиста.

РОЗДІЛ 6

ІНСТРУКЦІЯ КОРИСТУВАЧА

Фронт-енд вебзастосунку забезпечує простий та зручний інтерфейс. Для перегляду розкладу не потрібні жодні додаткові дії на кшталт реєстрації та/або авторизації. Дана інструкція дасть користувачеві розуміння яким чином працює дане програмне забезпечення.

Після запуску додатку, користувачу пропонується авторизуватися, зареєструватися або переглянути список груп (див. рис. 10).

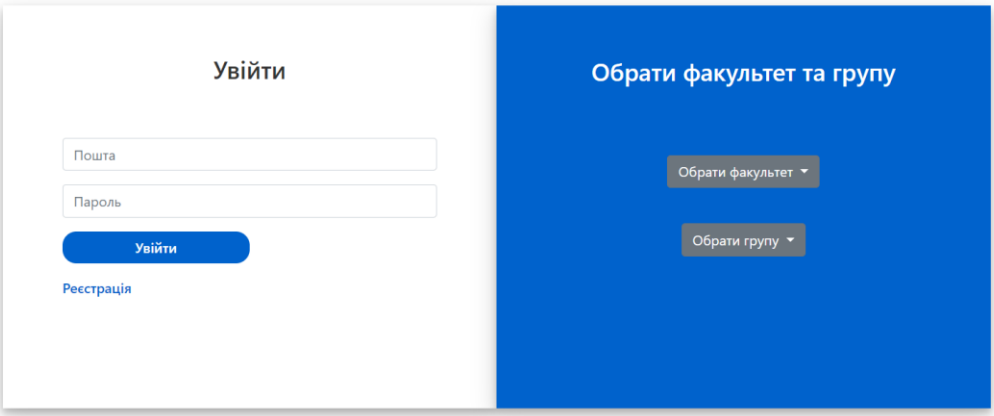

Рисунок 10 - Головна сторінка

Припустимо, методист хоче зареєструватися. Для цього необхідно перейти за посиланням під назвою «Реєстрація» та заповнити всі необхідні поля (див. рис. 11).

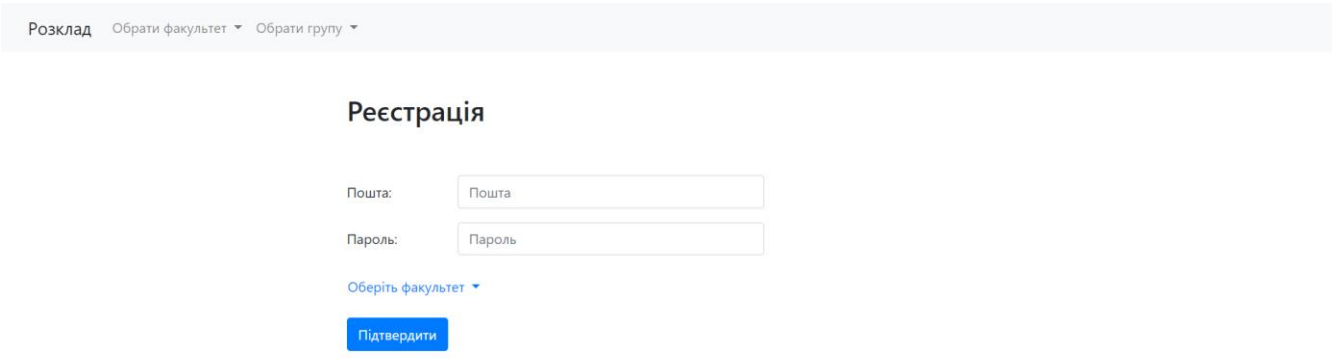

Рисунок 11 - Сторінка реєстрації

Далі, якщо методист авторизується за допомогою форми на головній сторінці (див. рис. 10), він потрапить на сторінку, яка сповіщує про те, що його профіль ще не активований (див. рис. 12).

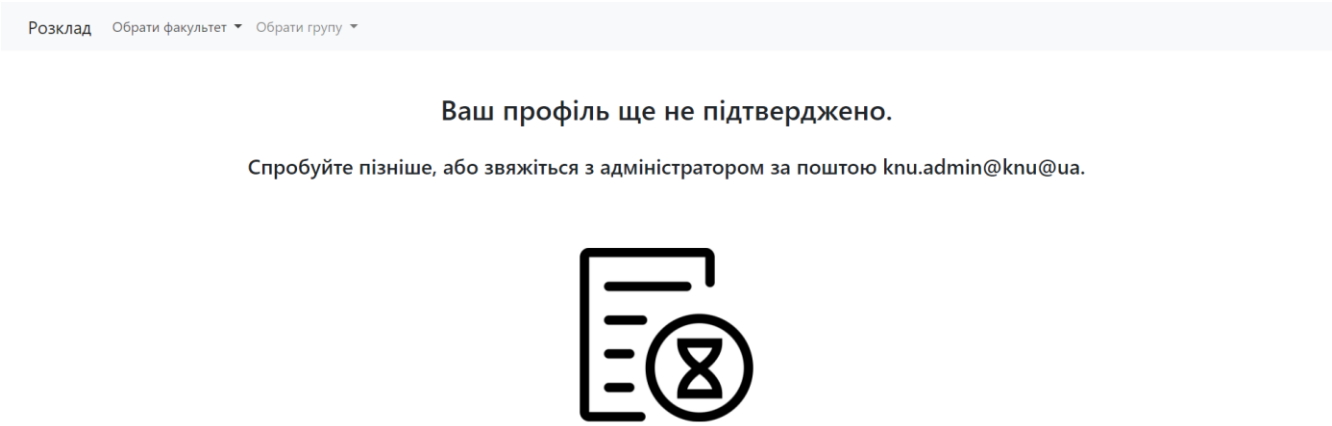

Рисунок 12 – Сторінка зі сповіщенням для методиста

Для того, щоб методист мав змогу продовжити роботу, його профіль має підтвердити адміністратор.

Для цього адміністратор має авторизуватися та перейти на сторінку з запитами на активацію (див. рис. 13).

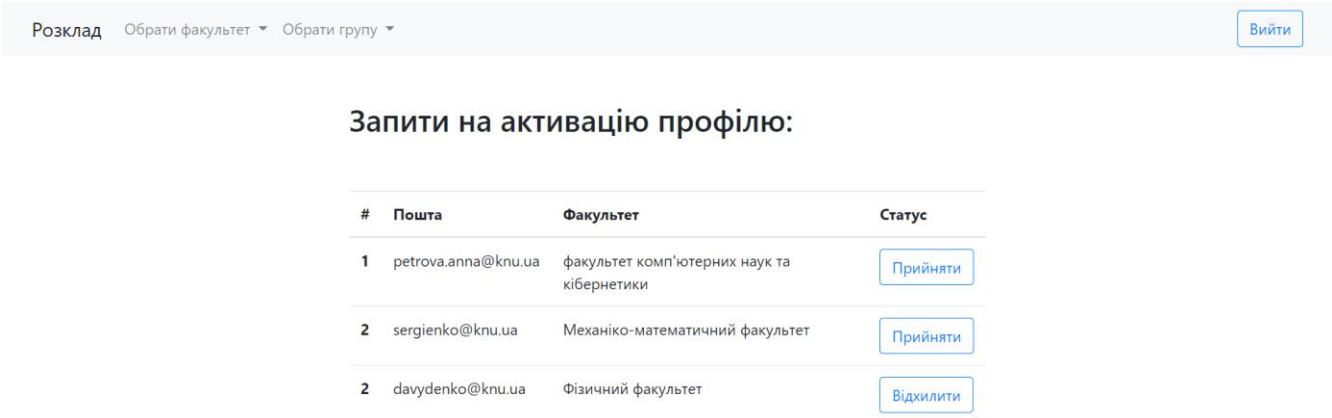

Рисунок 13 - Сторінка адміністратора з запитами на активацію профілів

Після того, як адміністратор прийняв запит методиста на активацію профіля, методист може знову авторизуватися і перейти на сторінку для складання розкладу (див. рис. 14). На цій сторінці методист може завантажити відповідні файли та натиснути кнопку «Завантажити» для підтвердження.

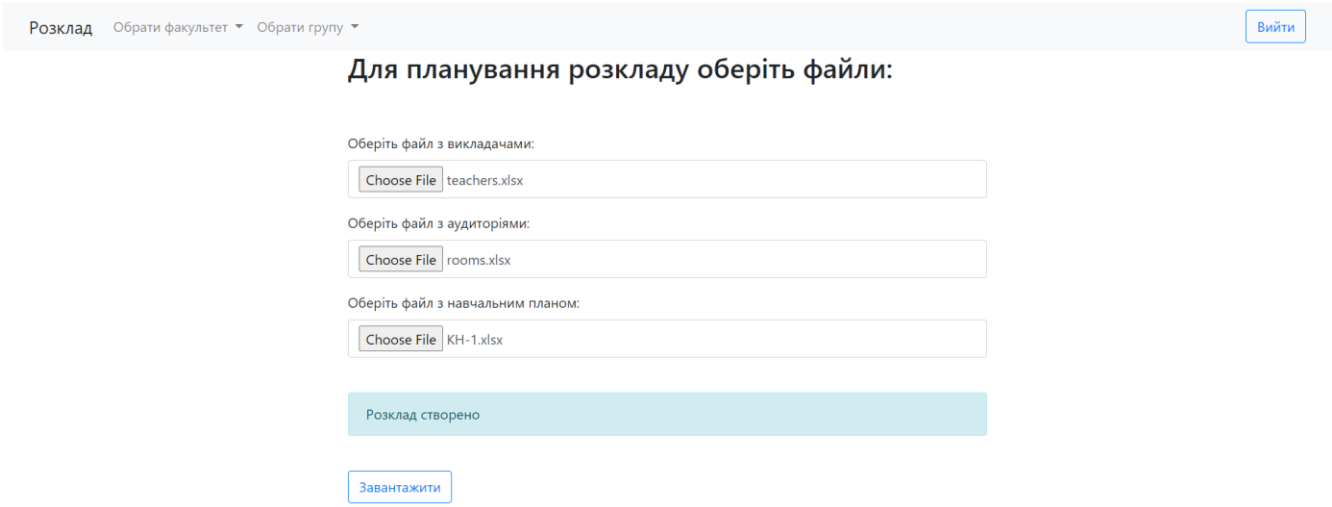

Рисунок 14 - Сторінка для складання розкладу

В випадку, якщо операція пройшла успішно, з'являється інформаційне сповіщення «Розклад створено». В іншому випадку методист побачить попередження «Не вдалося створити розклад».

Далі методист може обрати факультет та групу зі списків на панелі зверху (див. рис. 15) .

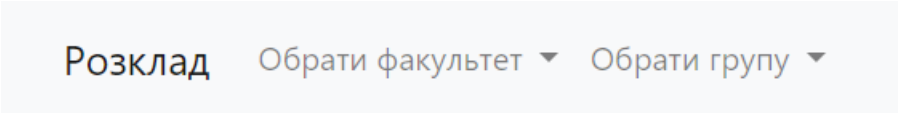

Рисунок 15 – Вибір факультету та групи

Далі користувач переходить на сторінку з розкладом для даної групи (див. рис. 16).

На цю сторінку може потрапити і звичайний користувач з головної сторінки, якщо обрати групу з випадаючого списку.

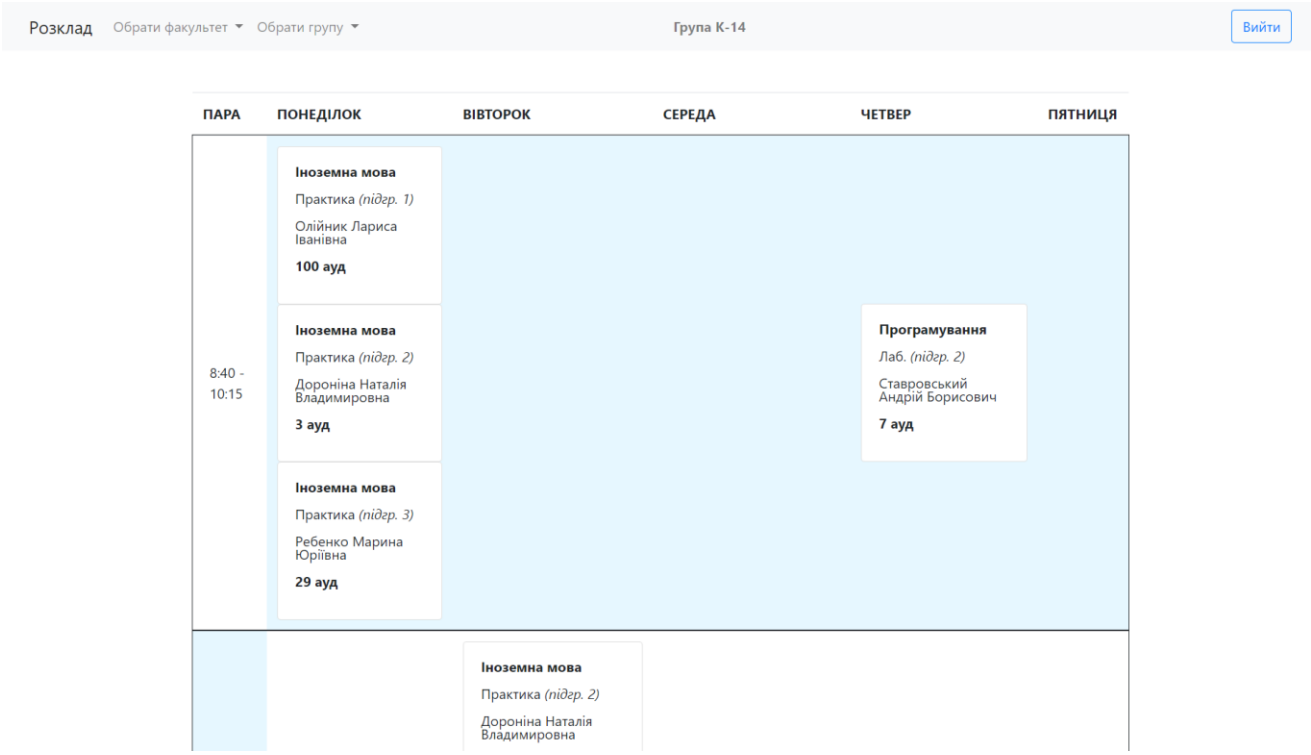

Рисунок 16 – Відображення розкладу

ВИСНОВКИ

В роботі реалізовано вебзастосунок, що надає можливість автоматично формувати розклад за завантаженими вхідними файлами та переглядати готовий розклад.

Для досягнення цієї мети було виконано наступні завдання:

- досліджено існуючі методи генерації розкладу за вхідними даними;
- обрано та реалізовано метод адаптований для предметної області вищого закладу;
- досліджено існуючі системи планування розкладу;
- закріплено та поглиблено теоретичні та практичні знання у сфері розробки вебзастосунків та використання баз даних;
- спроектовано та розроблено зручний інтерфейс вебзастосунку для методистів, що дозволяє після авторизації завантажувати файли з вхідними даними та автоматично генерувати розклад;
- спроектовано та розроблено зручний інтерфейс для студентів, що дозволяє без попередньої авторизації переглядати розклад для груп;
- спроектовано та розроблено базу даних для збереження даних про розклад;
- спроектовано вебзастосунок, що надає необхідний функціонал відповідно до ролі користувача.

Розроблений програмний продукт може застосовуватися для формування та відображення розкладу занять в закладах вищої освіти.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. MyTimetable [Електронний ресурс] – Режим доступу до ресурсу: https://mytimetable.live/.
2. Triton [Електронний ресурс] – Режим доступу до ресурсу: https://student.triton.knu.ua/.
3. АС "Деканат" [Електронний ресурс] – Режим доступу до ресурсу: https://vuz.osvita.net/ru/asu-vuz/as-dekanat/.
4. Tsang E. A Glimpse of Constraint Satisfaction / Edward Tsang. // Journal Artificial Intelligence Review, Kluwer Academic Publishers. – 1999. – №13. – С. 215–227.
5. Tsang E. Foundations of Constraint Satisfaction / Edward Tsang. – Essex: Academic Press Limited, 1993. – 309 с.
6. Karumanchi N. Algorithm Design Techniques: Recursion, Backtracking, Greedy, Divide and Conquer, and Dynamic Programming / Narasimha Karumanchi., 2018. – 488 с.
7. Matuszek D. Backtracking [Електронний ресурс] / David Matuszek – Режим доступу до ресурсу: https://www.cis.upenn.edu/~matuszek/cit594-2012/Pages/backtracking.html.
8. Ghedira K. Constraint Satisfaction Problems: CSP Formalisms and Techniques / Khaled Ghedira., 2013. – 238 с.
9. Dechter R. Constraint processing / Rina Dechter. – San Francisco: Morgan Kaufmann, 2003. – 520 с.
10. Erickson J. Algorithms / Jeff Erickson., 2019. – 472 с. – (1).
11. Rossi F. Handbook of Constraint Programming / F. Rossi, P. van Beek, T. Walsh., 2006. – 978 с. – (1).

12. Apt K. Principles of Constraint Programming / Krzysztof Apt., 2009. – 407 c.

13. Paparrizou A. Efficient Algorithms for Strong Local Consistencies and Adaptive Techniques in Constraint Satisfaction Problems / Anastasia Paparrizou., 2015. – 166 c. – (1).

14. Russell S. Artificial Intelligence / S. Russell, P. Norvig. – New Jersey: Prentice Hall, 2010. – 1151 c. – (3).

15. Pro Spring 5: An In-Depth Guide to the Spring Framework and Its Tools / I.Cosmina, R. Harrop, C. Schaefer, C. Ho., 2017. – 1296 c. – (5).

16. Gutierrez F. Pro Spring Boot 2: An Authoritative Guide to Building Microservices, Web and Enterprise Applications, and Best Practices / Felipe Gutierrez., 2018. – 564 c. – (2).

17. Bloch J. Effective Java / Joshua Bloch., 2013. – 414 c. – (3).

18. Bradshaw S. MongoDB: The Definitive Guide: Powerful and Scalable Data Storage / S. Bradshaw, E. Brazil, K. Chodorow., 2019. – 774 c. – (3).

19. Perkins L. Seven Databases in Seven Weeks: A Guide to Modern Databases and the NoSQL Movement / L. Perkins, E. Redmond, J. Wilson., 2018. – 475 c. – (2).

20. Singh A. Data Modeling With NoSQL Database / A. Singh, S. Ahmad., 2019. – 95 c. – (1).

21. Raj P. Architectural Patterns: Uncover essential patterns in the most indispensable realm of enterprise architecture / P. Raj, A. Raman, H. Subramanian., 2017. – 470 c. – (1).

22. Архитектура REST [Електронний ресурс]. – 2008. – Режим доступу до ресурсу: https://habr.com/ru/post/38730/.

23. Password Hashing [Електронний ресурс]. – 2015. – Режим доступу до ресурсу: https://terasolunaorg.github.io/guideline/1.0.x/en/Security/PasswordHashing.html#:~:text=2.1.-,BCryptPasswordEncoder,of%20salt%20and%20bcrypt%20algorithm.&text=Therefore

%2C%20compared%20to%20standard%20algorithms,%E2%80%9Coffline%20brute%20force%20attack%E2%80%9D.

ДОДАТОК А. ДІАГРАМА КЛАСІВ