Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Автоматики і комп'ютеризованих технологій
(повна назва)

Кафедра Комп'ютерно-інтегрованих технологій, автоматизації та робототехніки
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)
Розроблення інформаційної системи для оптимізації розкладу занять загальноосвітнього навчального закладу
(тема)

Виконав:
здобувач 2 року навчання,
групи КІТПВм-23-2
Пісклов М. А.
(прізвище, ініціали)

Спеціальність 174 Автоматизація, комп'ютерно-інтегровані технології та робототехніка
(код і повна назва спеціальності)
Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)
Освітня програма Комп'ютерно-інтегровані технологічні процеси та виробництва
(повна назва освітньої програми)

Керівник доц. Жарікова І. В.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Невлюдов І. Ш.
(підпис) (прізвище, ініціали)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Автоматики і комп'ютеризованих технологій

Кафедра Комп'ютерно-інтегрованих технологій, автоматизації та робототехніки

Рівень вищої освіти другий (магістерський)

Спеціальність 174 Автоматизація, комп'ютерно-інтегровані технології та робототехніка
(код і повна назва)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Комп'ютерно-інтегровані технологічні процеси та виробництва
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
« 22 » листопада 20 24 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Пісклову Миколі Альбертовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розроблення інформаційної системи для оптимізації розкладу занять загальноосвітнього навчального закладу
затверджена наказом по університету від 22 листопада 2024 р. № 1231 Ст
2. Термін подання здобувачем роботи до екзаменаційної комісії 24 січня 2025 р.
3. Вихідні дані до роботи 3.1 Розробити компоненти інформаційної системи для оптимізації розкладу загальноосвітнього навчального закладу.
3.2 Система – застосунок для платформи Windows з інтерфейсом доступу до бази даних.
3.3 Перелік програмних засобів: ОС Windows 11, Microsoft Visual Studio 2022, SQLite.
3.4 Технічне забезпечення: IBM-сумісний ПК з МП Core i3 та вище.
4. Перелік питань, що потрібно опрацювати в роботі: 4.1 Аналіз предметної області;
4.2 Розробка вимог до системи, 4.3 Обґрунтування вибору алгоритму оптимізації розкладу;
4.4 Логічне та фізичне моделювання даних; 4.5 Розробка математичної моделі системи;
4.6 Розробка алгоритму створення розкладу; 4.7 Опис прийнятих проєктних рішень:
опис архітектури системи, розробка запитів до БД системи, розробка класів та методів класів;
4.8 Заходи з охорони праці під час розробки застосунку;
4.9 Висновки.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) _____

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Керівник (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Аналіз предметної області	29.11.2024	Виконано
2	Розробка вимог до системи	10.12.2024	Виконано
3	Опис прийнятих проєктних рішень	15.12.2024	Виконано
4	Заходи з охорони праці під час розробки застосунку	03.01.2025	Виконано
5	Оформлення пояснювальної записки	03.01.2025	Виконано
6	Оформлення графічної частини та презентації	06.01.2025	Виконано
7	Подання роботи на нормоконтроль	11.01.2025	Виконано
8	Перевірка роботи на унікальність	14.01.2025	Виконано
9	Отримання відгуку керівника та рецензії	17.01.2025	Виконано
10	Отримання допуску до захисту	20.01.2025	Виконано
11	Представлення кваліфікаційної роботи до ЕК	22.01.2025	Виконано

Дата видачі завдання 22 листопада 2024 р.

Здобувач _____
(підпис)

Керівник роботи _____ доц. Жарікова І. В.
(підпис) (посада, прізвище, ініціали)

Я, як здобувач вищої освіти ХНУРЕ, розумію і підтримую політику закладу із академічної доброчесності. Я не надавав і не одержував недозволену допомогу під час підготовки кваліфікаційної роботи. Використання ідей, результатів і текстів інших авторів мають посилання на віповідне джерело.

«14» січня 2025 р. 

Пісклов М. А.

РЕФЕРАТ

Пояснювальна записка: 84 с., 4 табл., 49 рис., 3 дод., 31 джерело.

БАЗА ДАНИХ, СИСТЕМА УПРАВЛІННЯ БАЗАМИ ДАНИХ, SQLITE, ІНФОРМАЦІЙНА СИСТЕМА, ІНТЕРФЕЙС ДОСТУПУ, АЛГОРИТМИ СКЛАДАННЯ РОЗКЛАДУ.

Об’єктом дослідження є процес створення та оптимізації розкладу у рамках розробки інформаційної системи, призначеної для використання загальноосвітніми навчальними закладами.

Предмет дослідження – алгоритми оптимізації розкладу, програмні методи створення клієнтської та серверної частин інформаційної системи, яка дозволяє проводити створення й оптимізацію розкладу занять.

Мета роботи полягає у визначенні алгоритму оптимізації розкладу занять загальноосвітнього навчального закладу та створенні інформаційної системи із графічним інтерфейсом користувача, яка реалізує обраний алгоритм.

Методи дослідження – системний підхід, методи структурного аналізу та моделювання реляційних баз даних, методи реляційної алгебри та реляційного числення, методи об’єктноорієнтованого програмування.

У роботі проведено аналіз предметної області, що належить до діяльності загальноосвітнього закладу. Для визначення й уточнення вимог до розроблюваного застосунку проведено логічне та фізичне моделювання даних (відповідно до стандарту IDEF1X). Проведено проєктування застосунку з використанням архітектури MVVM.

Робота сприяє досягенню однієї з цілей сталого розвитку: «Ціль 4. Якісна освіта». Розробка інформаційної системи оптимізації розкладу загальноосвітнього навчального закладу сприяє виконанню завдання 4.1 – забезпечити доступність якісної шкільної освіти для всіх дітей та підлітків.

ABSTRACT

Explanatory note: 84 pages, 4 tables, 49 figures, 3 appendix, 31 sources.

DATABASE, DATABASE MANAGEMENT SYSTEM, SQLITE, INFORMATION SYSTEM, ACCESS INTERFACE, SCHEDULING ALGORITHMS.

The object of the study is the process for creating and optimizing schedule within the framework of the development of an information system designed for use by general educational institutions.

The subject of the study is scheduling optimization algorithms, software methods for creating the client and server parts of an information system that allows for the creation and optimization of class schedules.

The purpose of the work is to determine an algorithm for optimizing the schedule of a general educational institution and to develop an information system with a graphical user interface that implements the chosen algorithm.

Research methods include a systematic approach, methods of structural analysis and modeling of relational databases, methods of relational algebra and relational calculus, and methods of object-oriented programming.

The work includes an analysis of the subject area related to the activities of a general educational institution. Logical and physical data modeling (according to the IDEF1X standard) was carried out to define and clarify the requirements for the developed application. The application was designed using the MVVM architecture.

The work contributes to achieving one of the Sustainable Development Goals: Goal 4: Quality Education. The development of an information system for optimizing the schedule of a general educational institution supports the achievement of Target 4.1 – ensuring access to quality school education for all children and adolescents.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

БД – база даних;

ІС – інформаційна система;

ПЗ – програмне забезпечення;

СУБД – система управління базами даних;

MVVM – Model-View-Viewmodel;

ORM – Object-Relational Mapping;

SQL – Structured Query Language.

ВСТУП

Сучасні інформаційні технології суттєво вплинули на автоматизацію робочих процесів у різних галузях, зокрема на підприємствах та в організаціях. Їх активно використовують для обліку, зберігання даних, а також для управління такими процесами, як створення розкладів і планів. Освітня сфера не є винятком – сьогодні діяльність навчальних закладів тісно пов’язана з використанням комп’ютерних систем, що значно спрощують управління інформацією й організаційні процеси.

Створення розкладу без залучення інформаційних технологій є складним завданням. Потрібно врахувати велику кількість змінних, таких як кількість класів, уроків, наявність і навантаження вчителів, специфічні потреби учнів і предметів. Це часто призводить до помилок та ускладнює внесення будь-яких змін у розклад, що потребує повного його перегляду.

Використання інформаційних технологій дозволяє значно спростити цей процес. Автоматизація за допомогою спеціальних алгоритмів забезпечує швидке врахування всіх параметрів і обмежень, що сприяє ефективному розподілу ресурсів і мінімізує ризик помилок. Окрім того, корекції можуть вноситися в реальному часі, що підвищує гнучкість і ефективність управління навчальним процесом.

Мета роботи полягає у визначенні алгоритму оптимізації розкладу занять загальноосвітнього навчального закладу та створенні інформаційної системи із графічним інтерфейсом користувача, яка реалізує обраний алгоритм.

Об’єктом дослідження є процес створення та оптимізації розкладів у рамках розробки інформаційної системи, призначеної для використання загальноосвітніми навчальними закладами.

Предмет дослідження – алгоритми оптимізації розкладу, програмні методи створення клієнтської та серверної частин інформаційної системи, яка дозволяє проводити створення й оптимізацію розкладу занять.

Для досягнення мети потрібно:

– провести аналіз предметної області;

– провести розробку вимог до системи. Цей процес включає наступні кроки: обґрунтування вибору алгоритму оптимізації розкладу, логічне й фізичне моделювання даних;

– провести опис прийнятих проєктних рішень. Цей процес включає наступні кроки: опис архітектури системи та опис створених класів та методів класів.

Робота інформаційної системи оптимізації розкладу загальноосвітнього навчального закладу буде сприяти досягненню однієї з цілей сталого розвитку: «Ціль 4. Якісна освіта». Розробка інформаційної системи оптимізації розкладу загальноосвітнього навчального закладу робить вагомий внесок у реалізацію завдання 4.1 – забезпечення доступної та якісної шкільної освіти для всіх дітей і підлітків. Ця розробка спрямована на підвищення ефективності організації навчального процесу, зниження адміністративного навантаження та створення сприятливих умов для навчання, що допомагає зменшити освітню нерівність і сприяє рівним можливостям для всіх учасників освітнього процесу.

Кваліфікаційна робота виконана згідно ДСТУ 3008-15 [1], керуючись методичними вказівками [2], а результати кваліфікаційної роботи отримали апробацію у науковій статті (додаток А) [3].

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз технічного завдання

Використання комп’ютерних інформаційних технологій дозволяє практично повністю автоматизувати робочі процеси виробництва, організацій і підприємств [4-7]. Оптимізація розкладу занять для загальноосвітніх навчальних закладів (ЗНЗ) є складною і важливою задачею. Головна проблема при вирішенні цієї задачі полягає в розподілі обмежених ресурсів (вчителі, аудиторії тощо) на певний час, враховуючи різноманітні обмеження (шкільні предмети, учні, час, бажання викладачів). Інформаційна система (ІС) для оптимізації розкладу має забезпечити автоматизацію процесу складання розкладу, що значно скорочує трудомісткість цього процесу та підвищує його якість.

Процес складання розкладу для ЗНЗ включає кілька ключових аспектів:

– кожен клас має свій перелік обов’язкових предметів згідно з навчальним планом, який потрібно розподілити на тиждень;

– кожен вчитель має певний набір предметів, який він викладає, а також особисті побажання або обмеження щодо часу викладання;

– аудиторії не мають бути зайняті двома класами одночасно;

– існують обмеження по часу проведення уроків – час початку та кінця навчального дня, можливі перерви тощо.

Основна мета процесу створення розкладу ЗНЗ – скласти такий розклад, який забезпечує рівномірне навантаження на вчителів і учнів, мінімізує конфлікти та враховує всі обмеження.

У ЗНЗ існують певні особливості, які ускладнюють процес складання розкладу. По-перше – це велика кількість обмежень. Вчителі можуть мати побажання щодо часу проведення уроків, а певні класи вимагають спеціалізованих аудиторій.

По-друге – нерівномірне навантаження. Необхідно враховувати рівномірний розподіл уроків між класами, щоб уникнути перевантаження учнів або вчителів.

Також однією з особливостей складання розкладу є динамічність навчального процесу. Навчальний план може змінюватися протягом року, тому система повинна мати можливість легко вносити зміни до існуючого розкладу.

Отже, можна виділити основні проблеми ЗНЗ під час складання розкладу занять:

– велика кількість помилок і конфліктів за умови складання розкладу без використання комп'ютерних технологій;

– велика трудомісткість процесу складання розкладу;

– велика кількість обмежень і факторів, які треба враховувати;

– динамічність навчального процесу.

ЗНЗ у своїй діяльності керується наступними нормативними документами:

– Закон України «Про освіту»;

– Закон України «Про захист персональних даних»;

– Закон України «Про загальну середню освіту» [8].

1.2 Аналіз аналогічних систем

Сьогодні за умови виникнення проблеми складання розкладу для навчальних закладів все частіше використовується спеціалізоване програмне забезпечення (ПЗ). Існує кілька відомих рішень, які вже застосовуються для автоматизації й оптимізації цього процесу. Аналіз цих систем дозволяє виявити основні переваги та недоліки, а також визначити ключові особливості для майбутньої розробки власної інформаційної системи.

Free Timetabling Software (FET) – це безкоштовне програмне забезпечення з відкритим кодом, яке надає можливість автоматичного складання розкладу на основі заданих обмежень [9].

Головними особливостями ПЗ є підтримка великої кількості параметрів для налаштування розкладу (включаючи складні обмеження на рівні школи, вчителів, класів, предметів), висока швидкість генерації розкладу, навіть для великих навчальних закладів, підтримка різних типів шкіл і університетів.

Головними недоліками цього програмного рішення є відсутність технічної підтримки та користувацького інтерфейсу високої якості, менша зручність у порівнянні з комерційними продуктами, потреба у знаннях з налаштування параметрів для досягнення бажаних результатів. Скріншот програми наведено на рисунку 1.1.

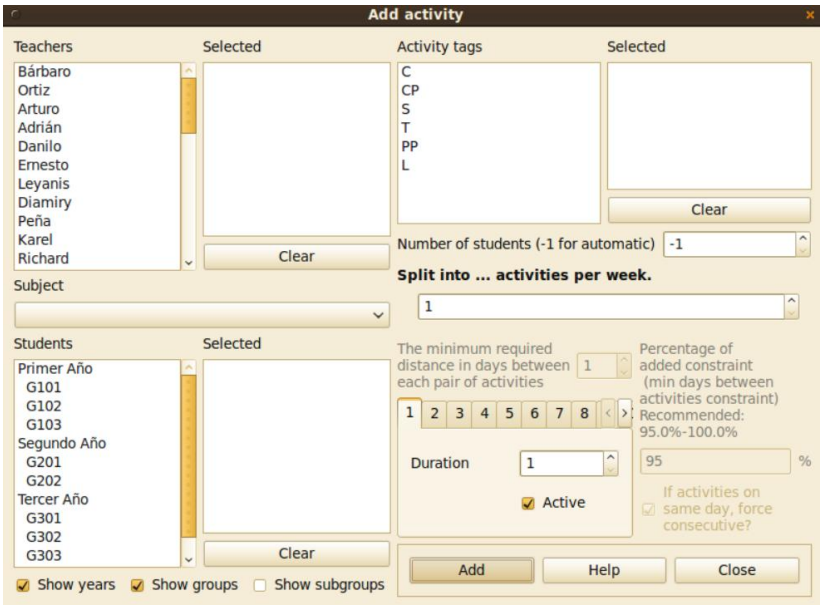

Рисунок 1.1 – FET.Форма «Додати активність»

EduPage – комплексне рішення, що поєднує в собі можливості не тільки для складання розкладу, але й для загального управління навчальним процесом [10].

Особливостями даного ПЗ є інтеграція з електронними журналами, відвідуванням, оцінками, можливість генерувати автоматизовані розклади та надавати учням і вчителям доступ до них через мобільні додатки й управління змінами в розкладі у реальному часі.

Перевагами цього програмного рішення є широка функціональність, що виходить за рамки простої генерації розкладу, взаємодія між усіма учасниками навчального процесу через єдину платформу, сумісність з іншими системами управління навчальним закладом.

Недоліки:

– більший акцент на загальне управління школою, що може бути надмірним для шкіл, які шукають лише інструмент для розкладу;

– висока вартість у випадку використання всього функціоналу.

Знімок екрану застосунку EduPage наведено на рисунку 1.2.

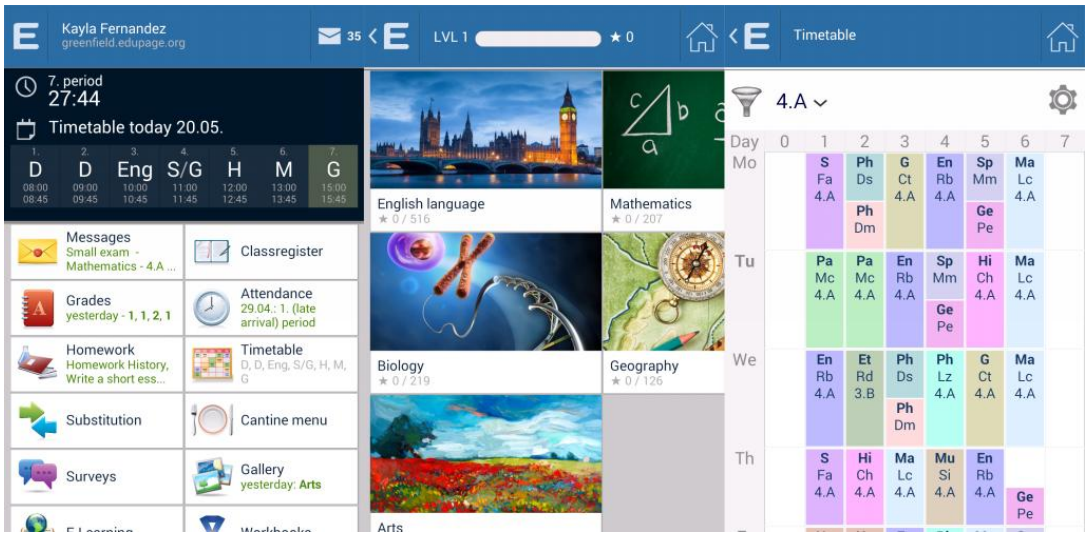

Рисунок 1.2 – Одна зі сторінок застосунку EduPage

Ще одним прикладом ІС, яка направлена на оптимізацію розкладу, є Skolaris. Це платформа для складання розкладу з акцентом на індивідуальний підхід до кожного навчального закладу [11].

Особливості: підтримка складання розкладу для комплексних структур (шкіл з кількома філіями або великою кількість учнів і вчителів), оптимізація з урахуванням побажань вчителів і специфіки навчальних закладів, можливість роботи в хмарі.

Головні недоліки: порівняно висока вартість, обмежена кількість підтримуваних мов інтерфейсу, надмірна ускладненість інтерфейсу.

Скріншот застосунку Skolaris наведено на рисунку 1.3.

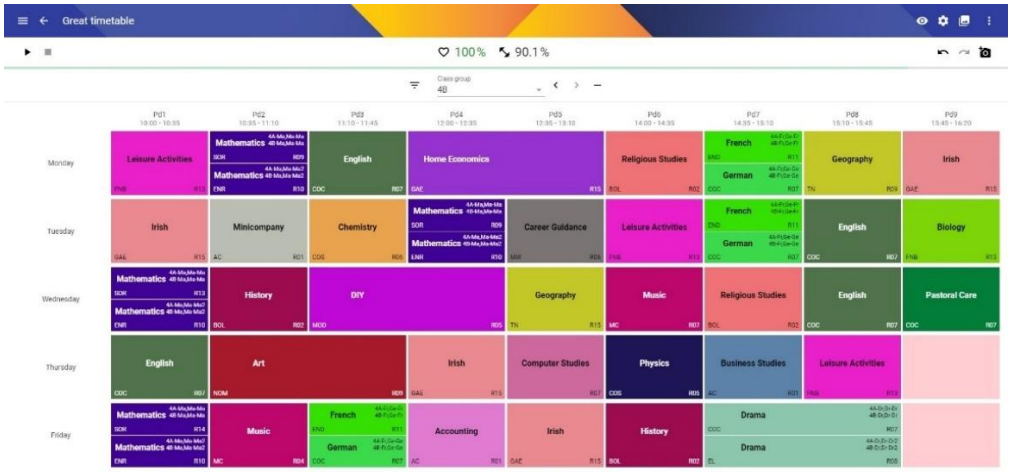

Рисунок 1.3 – Застосунок Skolaris

Безкоштовні рішення, такі як FET, є гнучкими, але вимагають глибокого розуміння процесу складання розкладу (що може потребувати найму окремого працівника, який має досвід складання розкладу в цьому ПЗ) та можуть мати недоліки в плані користувацького інтерфейсу та технічної підтримки.

Системи на зразок EduPage та Skolaris інтегрують управління розкладом у більш комплексні системи управління навчальними процесами, що може бути перевагою для великих ЗНЗ, але надмірним для невеликих закладів.

За результатами аналізу реалізованих систем можна зробити висновки, що у процесі розробки власної ІС для оптимізації розкладу слід врахувати:

– гнучкість налаштувань обмежень і правил;

– простоту у використанні та навчанні користувачів;

– вартість розробки та впровадження, порівняно з готовими рішеннями.

1.3 Постановка задачі розробки програмного забезпечення

Головною задачею інформаційної системи є складання й оптимізація розкладу ЗНЗ. Головна проблема, яку буде вирішувати розроблена система, – це створення й оптимізація розкладу занять ЗНЗ.

З цього випливають основні вимоги до ІС для оптимізації розкладу:

– автоматизація складання розкладу. Система має автоматично розподіляти предмети за часом на основі введених даних;

– гнучке налаштування навантаження. Користувачі повинні мати можливість легко керувати поведінкою програми, шляхом зміни навантаження на вчителів та класи;

– графічний інтерфейс. Система повинна мати інтуїтивний інтерфейс для введення даних, перегляду та редагування розкладу.

Оскільки створенням розкладу зазвичай займається адміністрація ЗНЗ, у системі будуть присутні деякі адміністративні функції, а саме додавання, редагування та видалення даних про: класи, розклади, аудиторії, предмети, вчителів, а також прив’язка вчителів до предметів, які вони викладають та до аудиторій.

Основними функціями є складання й оптимізація розкладу занять ЗНЗ, призначення предметів класу та визначення навантаження за цими предметами.

Після запуску система відображає користувачеві головне меню програми з відповідними пунктами меню для кожної з функцій у виді кнопок:

– кнопка «Робота з класами»;

– кнопка «Робота з аудиторіями»;

– кнопка «Робота з предметами»;

– кнопка «Робота з вчителями»;

– кнопка «Робота з розкладами».

Коли користувач натискає на кнопку «Робота з класами», система відображає користувачу форму з наявними в системі класами. Класи наведені у вигляді списку. Наприкінці списку класів буде відображатись кнопка «Додати». Коли користувач обирає потрібний йому клас, то наприкінці списку з’являться кнопки «Редагувати» та «Видалити». При натисканні кожної з наведених кнопок повинна здійснитись операція з даними, яка відповідає назві кнопки. При натисканні кнопки «Додати» користувач повинен побачити форму додавання

нового класу. Форма додавання класу складається з наступних текстових полів: назва класу, рік набору класу та кнопка «Зберегти». У випадку успіху або помилки при виконанні операцій користувач отримує відповідне повідомлення на екрані.

Коли користувач натискає на кнопку «Робота з аудиторіями», система відображає користувачу форму з наявними в системі аудиторіями у вигляді списку. Наприкінці списку буде відображатись кнопка «Додати». Коли користувач обирає потрібну аудиторію, то наприкінці списку з’являться кнопки «Редагувати» та «Видалити». При натисканні кожної з наведених кнопок повинна здійснитись операція з даними, яка відповідає назві кнопки. При натисканні кнопки «Додати» користувач повинен побачити форму додавання аудиторії. Форма додавання аудиторії складається з наступних текстових полів: номер аудиторії та кнопка «Зберегти». У випадку успіху або помилки при виконанні операцій користувач отримує відповідне повідомлення на екрані.

Коли користувач натискає на кнопку «Робота з предметами», система відображає користувачу форму з наявними в системі предметами у вигляді списку. Наприкінці списку буде відображатись кнопка «Додати». Коли користувач обирає потрібний предмет, то наприкінці списку з’являться кнопки «Редагувати» та «Видалити». При натисканні кожної з наведених кнопок повинна здійснитись операція з даними, яка відповідає назві кнопки. При натисканні кнопки «Додати» користувач повинен побачити форму додавання предмета. Форма додавання предмета складається з наступних текстових полів: назва предмета та кнопка «Зберегти». У випадку успіху або помилки при виконанні операцій користувач отримує відповідне повідомлення на екрані.

Коли користувач натискає на кнопку «Робота з вчителями», система відображає користувачу форму з наявними в системі вчителями у вигляді списку. Наприкінці списку буде відображатись кнопка «Додати». Коли користувач обирає потрібного вчителя, то наприкінці списку з’являться кнопки «Редагувати» та «Видалити». Також в цьому випадку користувач побачить список предметів вчителя. Наприкінці кожного з цих списків буде знаходитись

список, що випадає, з аудиторіями та предметами відповідно. Поруч з списками предметів та аудиторій користувач побачить кнопку «Додати», натискання якої призведе до закріплення за вчителем відповідного предмета. При натисканні кожної з наведених кнопок повинна здійснитись операція з даними, яка відповідає назві кнопки. При натисканні кнопки «Додати», яка знаходиться нижче списку вчителів, користувач повинен побачити форму додавання вчителя. Форма додавання вчителя складається з наступних текстових полів: повне ім’я вчителя, список, що випадає з доступними аудиторіями та кнопка «Зберегти». У випадку успіху або помилки при виконанні описаних операцій користувач отримує відповідне повідомлення на екрані.

Коли користувач натискає на кнопку «Робота з розкладами», система відображає користувачу форму з наявними в системі розкладами класів у вигляді списку. Наприкінці списку буде відображатись кнопка «Додати». Коли користувач обирає потрібний розклад класу, то наприкінці списку з’являться кнопки «Редагувати», «Видалити». При натисканні кожної з наведених кнопок повинна здійснитись операція з даними, яка відповідає назві кнопки. При натисканні кнопки «Додати» користувач повинен побачити форму додавання розкладу. Форма додавання розкладу класу складається з наступних текстових полів: рік розкладу, список, що випадає, з класами та кнопка «Зберегти». У випадку успіху або помилки при виконанні операцій користувач отримує відповідне повідомлення на екрані. Також повинна бути присутня кнопка «Перейти до складання розкладу».

При натисканні кнопки «Перейти до складання розкладу» користувач потрапляє на нове вікно, де буде відображатись майбутній розклад. Розклад представлено у вигляді таблиці. В першій колонці таблиці розташовані дні тижня, а в першому рядку назви класів із кнопкою «Назначені предмети». Кожна клітинка таблиці відповідає дню в розкладі певного класу. Наприкінці таблиці розташовуються кнопки «Очистити», «Скласти розклад». Натискання на кнопку «Скласти розклад» запускає алгоритм складання розкладу. Кнопка «Очистити» видаляє створений розклад. При натисканні «Назначені предмети» застосунок

відкриє нове вікно для користувача. У новому вікні користувач побачить назву класу, предмети для якого будуть редагуватись. Також користувач побачить список, що випадає, з співвідношенням вчителя та предмету, який він викладає. Наприкінці цього списку користувач побачить кнопку «Додати». Якщо користувач обрав предмет та натиснув на цю кнопку, то до списку в правому кінці екрану додається предмет та закріплюється за класом. Наприкінці списку предметів, які вже закріплені за класом буде знаходитись кнопка «Видалити», яка відкріплює предмет від класу. У випадку успіху або помилки при виконанні операцій користувач отримує відповідне повідомлення на екрані.

Розробка ІС для складання та оптимізації шкільного розкладу – це складний процес, що вимагає врахування багатьох обмежень і факторів. Така система дозволить автоматизувати значну частину роботи адміністрації, покращити якість розкладу та підвищити ефективність управління навчальним процесом. Використання сучасних методів оптимізації та алгоритмів дозволить вирішувати задачу складання розкладу навіть для великих шкіл із великою кількістю учнів і вчителів.

Розроблюваний застосунок буде представлений у застосунку для платформи Windows, оскільки більшість користувачів персональних комп’ютерів, а також більшість службових ПК у ЗНЗ працюють на цій операційній системі. Така конфігурація застосунку передбачає, що вхідні дані програми будуть потрапляти від користувача через відповідні форми, які були описані раніше.

2 РОЗРОБКА ВИМОГ ДО СИСТЕМИ

2.1 Обґрунтування вибору алгоритму оптимізації розкладу

У розкладі ЗНЗ необхідно скласти графік проведення заданих зустрічей між учнями та вчителями так, щоб отримані розклади були прийнятними та відповідали вимогам усіх учасників навчального процесу. Крім того, освітні системи відрізняються від країни до країни, і всередині конкретної освітньої системи освіта відрізняється на різних рівнях. У зв’язку з цим виникли різні погляди на характеризацію категорій розкладу ЗНЗ, але в рамках цієї роботи буде розглянута система загальноосвітніх навчальних закладів України.

У типовому загальноосвітньому навчальному закладі складання розкладу базується на класах, які формуються шляхом групування учнів за спільною програмою. Приклад сітки розкладу подано на рисунку 2.1.

Розклади не мають містити конфліктів, мають бути компактними та задовольняти різноманітні обмеження на розподіл і послідовність уроків. В абсолютній більшості закладів України аудиторії для проведення занять розташовані в одному будинку, тому час переміщення учнів між корпусами ЗНЗ в процесі складання розкладу враховуватись не буде.

	Mon	Tue	Wed	Thu	Fri
1					
2					
3					
4					
5					
6					

Рисунок 2.1 – Приклад сітки розкладу класу ЗНЗ

Тож процес складання розкладу передбачає планування та розподіл ресурсів з метою створення розкладу для кожного класу. Як додатковий результат, отримуються розклади для кожної аудиторії та вчителя. Простір рішень лімітується обмеженнями, які наведені на рисунку 2.2:

– обмеження доступу до ресурсів: ресурси (учні, вчителі та аудиторії) не мають бути подвійно забронюваними;

– розподіл класу на підгрупи;

– об’єм ресурсів: кількість аудиторій і вчителів, адже один вчитель може викладати кілька предметів;

– обмеження щоденного робочого навантаження;

– обмеження кількості робочих днів;

– обмеження на наявність «вікон»: розклади можуть містити часові слоти, які не зайняті жодним заняттям. Такий часовий слот у середині дня називається вільним, прогалиною або «вікном», якщо перед і після нього заплановані зустрічі. Кількість вільних часових слотів, які виникають у розкладі однієї особи, називається загальним вільним часом цієї особи.

Рисунок 2.2 – Основа обмежень складання розкладу

Існує кілька алгоритмів, які можуть бути застосовані для розв’язання задачі створення розкладу. До них відносяться методи оптимізації, що дозволяють ефективно розподілити уроки, вчителів, предмети та аудиторії з урахуванням різних обмежень. Ось кілька прикладів таких алгоритмів для оптимізації шкільного розкладу:

– генетичні алгоритми;

– методи тваринного світу;

– локальний пошук;

– метод обмежень.

Ці методи можуть використовуватися як окремо, так і в поєднанні, залежно від специфічних вимог та обмежень завдання створення шкільного розкладу. Особливу увагу варто звернути на метод обмежень, оскільки його застосування надає певні переваги у процесі складання та оптимізації розкладу для ЗНЗ.

По-перше, метод обмежень дозволяє гнучко визначати різні обмеження, що стосуються завдання. Користувач може легко додавати обмеження, такі як доступність вчителів, наявність вільних аудиторій, а також кількість уроків для учнів тощо.

По-друге, виразність обмежень спрощує формалізацію вимог користувача, що особливо корисно для складних задач, де важко визначити точну математичну цільову функцію.

По-третє, метод обмежень показує високу ефективність у вирішенні складних комбінаторних проблем, таких як оптимізація розкладу, оскільки дозволяє розглядати значну кількість можливих комбінацій.

Перевагами методу обмежень є можливість урахування різноманітних умов, ітеративний підхід (що дозволяє додавати чи змінювати обмеження під час розробки розкладу), а також зручність у використанні спеціалізованих інструментів. Це робить його корисним для створення розкладів, де важко передбачити всі можливі умови.

Метод обмежень (Constraint Programming, CP) є підходом до програмування та розв’язання задач оптимізації, який полягає у явному визначенні обмежень, що мають виконуватися для рішень. У цьому підході користувач декларативно задає обмеження на припустимі рішення для набору змінних, які підлягають оптимізації.

Основні принципи методу обмежень включають:

– змінні рішення: задача формулюється через визначення змінних, що потребують оптимізації. Для розкладу ЗНЗ це можуть бути час проведення уроків, вчителі, класи, аудиторії тощо;

– обмеження: визначаються умови, яким мають відповідати змінні. Наприклад, обмеження може вимагати, щоб два уроки не проводилися одночасно для одного класу;

– цільова функція: оптимізація виконується з метою максимізувати або мінімізувати певний критерій. У випадку розкладу це може бути, наприклад, мінімізація «вікон» між уроками.

Для реалізації методу обмежень використовуються інструменти, що забезпечують можливість визначення та розв’язання обмежень, наприклад, спеціалізовані бібліотеки.

У задачах складання розкладу ЗНЗ метод обмежень дозволяє враховувати обмеження, такі як доступність вчителів, аудиторій, максимальна кількість уроків на день та інші фактори. Це допомагає знайти оптимальний розклад, який задовольняє всі встановлені умови.

Щоб полегшити застосування технології методу обмежень до проблеми складання розкладу для загальноосвітнього навчального закладу, нам потрібно перетворити високорівневий опис проблеми (у термінах вчителів, учнів та аудиторій) на так звану задачу задоволення обмежень (Constraint satisfaction problem, CSP). Задача задоволення обмежень формулюється у термінах обмежень для змінних з областями значень, і для знаходження рішення потрібно вибрати значення з області значень кожної змінної так, щоб отримане значення задовольняло всі обмеження.

Зазвичай спосіб опису CSP полягає у наданні трійки (X, δ, C) із набором змінних X, набором обмежень C для X та цільовою функцією δ на X, яка пов’язує кожну змінну із її областю значень. Інший спосіб вказати CSP – це надання гіперграфа змінних у вигляді вузлів і обмежень у вигляді гіпердуг. Такий граф називається мережею обмежень.

Обмеження вважається зайвим відносно визначеної проблеми, якщо кожне рішення проблеми задовольняє обмеження, хоча проблема не визначає це обмеження явно.

У контексті оптимізаційних задач, цільова функція визначає той критерій, який необхідно оптимізувати чи мінімізувати. Вона відображає нашу основну мету.

Обмеження, з іншого боку, визначають умови, яким має відповідати оптимальне рішення. Ці обмеження обмежують простір можливих рішень і враховують фактори, які є необхідність врахувати у процесі оптимізації.

Під час оптимізації розкладу із використанням методу обмежень, шукаємо такі значення змінних, які задовольняють всі обмеження та мінімізують (або максимізують, залежно від завдання) значення цільової функції.

Головним критерієм оптимізації для цільової функції було обрано зведення до мінімуму кількості «вікон» між заняттями класів, а у випадку початкового та середнього рівня навчання взагалі відсутність так званих «простоїв» між заняттями чи на початку дня (наприклад, початок занять з другого або нульового уроку, до чого часто прибігають у складанні розкладу старших класів для запобігання конфлікту доступу до ресурсів: вчителів чи аудиторій). Треба зазначити, що у випадку розглядання розкладу вчителя можна не приділяти багато уваги цьому аспекту, оскільки під час «вікна» вчитель може займатися перевіркою контрольних, самостійних чи домашніх робіт, а також підготовкою до майбутніх занять [3, 12-19].

2.2 Особливості складання розкладу ЗНЗ

Розподіл навчального навантаження протягом тижня встановлюють таким чином, щоб найбільший його обсяг припадав на вівторок, середу. На ці дні до розкладу закладу освіти вносять навчальні предмети, які потребують великого розумового напруження або ті, які не вимагають значного навантаження, але у більшій кількості, ніж в інші дні тижня. Для визначення навантаження використовують коефіцієнт оцінки складності предмету [20].

Коефіцієнт оцінки складності предметів ЗНЗ – це числовий показник, який використовується для оцінки рівня складності певного навчального предмета. Цей коефіцієнт дозволяє врахувати вплив складності предмета на розподіл навчального навантаження у розкладі, щоб забезпечити оптимальний баланс між важкими та легкими уроками протягом навчального тижня [21].

Коефіцієнти оцінки складності предметів наведено у таблиці 2.1.

Таблиця 2.1 – Коефіцієнти складності предметів

Предмет	Ступінь складності
Геометрія	6
Алгебра	5,5
Іноземна мова	5,4
Хімія	5,3
Фізика	5,2
Біологія	3,6
Українська мова	3,5
Українська література	1,7
Історія	1,7
Інші	<1,7

Для отримання реалістичних вхідних даних, які будуть використовуватись в програмному застосунку, навантаження для кожного з класів та їх предмети були залучені з типових навчальних програм [22].

Приклад навантаження для учнів 5-9 класів наведено у таблиці 2.2.

Таблиця 2.2 – Навантаження для учнів 5-9 класів.

Освітні галузі	Навчальні предмети	Кількість годин на тиждень у класах				
		5	6	7	8	9
Мови і літератури	Українська мова	3,5	3,5	2,5	2	2
	Українська література	2	2	2	2	2
	Іноземна мова	3	3	3	3	3
	Зарубіжна література	2	2	2	2	2
Суспільство-знавство	Історія України	1	1	1	1,5	1,5
	Всесвітня історія	-	1	1	1	1
	Основи правознавства	-	-	-	-	1
Мистецтво*	Музичне мистецтво	1	1	1	-	-
	Образотворче мистецтво	1	1	1	-	-
	Мистецтво	-	-	-	1	1
Математика	Математика	4	4	-	-	-
	Алгебра	-	-	2	2	2
	Геометрія	-	-	2	2	2
Природо-знавство	Природознавство	2	-	-	-	-
	Біологія	-	2	2	2	2
	Географія	-	2	2	2	1,5
	Фізика	-	-	2	2	3
	Хімія	-	-	1,5	2	2
Технології	Трудове навчання	2	2	1	1	1
	Інформатика	1	1	1	2	2
Здоров’я і фізична культура	Основи здоров’я	1	1	1	1	1
	Фізична культура**	3	3	3	3	3
Разом		26,5	29,5	31	31,5	33

Години, передбачені для фізичної культури (позначення у вигляді **), не враховуються під час визначення гранично допустимого навчального навантаження учнів, але обов’язково фінансуються.

Навантаження для учнів 10-11 класів наведено у таблиці 2.3.

Таблиця 2.3 – Навантаження учнів 10-11 класів

Предмет	Кількість годин на тиждень у класах	
	10 клас	11 клас
Базові предмети	27	26
Українська мова	2	2
Українська література	2	2
Зарубіжна література	1	1
Іноземна мова	2	2
Історія України	1,5	1,5
Всесвітня історія	1	1
Громадянська освіта	2	0
Математика (алгебра і початок аналізу та геометрія)	3	3
Біологія і екологія	2	2
Географія	1,5	1
Фізика і астрономія	3	4
Хімія	1,5	2
Фізична культура	3	3
Захист Батьківщини	1,5	1,5
Вибірково-обов’язкові предмети (інформатика, технології, мистецтво)	3	3
Додаткові години на профільні предмети	8	9
Гранично допустиме навантаження	33	33

Орієнтовна кількість годин для викладання профільних предметів також буде визначатись за допомогою типової освітньої програми та наведена у таблиці 2.4.

Заклад освіти може збільшувати або зменшувати кількість годин на вивчення профільних предметів [23].

Таблиця 2.4 – Орієнтовна кількість годин профільних предметів

Предмет	Кількість годин на тиждень у класах	
	10 клас	11 клас
Українська мова	4	4
Українська література	4	4
Зарубіжна література	3	3
Іноземна мова	5	5
Друга іноземна мова	3	3
Історія України	3	3
Всесвітня історія	3	3
Правознавство	3	3
Економіка	3	3
Математика	9	9
Фізика і астрономія	7	7
Фізика	6	6
Астрономія	2	2
Біологія і екологія	5	5
Хімія	4	6
Географія	5	5
Інформатика	5	5
Технології	6	6
Мистецтво	5	5
Фізична культура	6	6
Захист Батьківщини	5	5

Розглянуті в цьому розділі дані будуть використані для отримання більш реалістичного набору вхідних даних. Це дозволить максимально наблизити моделювання до реальної задачі.

2.3 Логічне та фізичне моделювання даних

У ході проведення логічного моделювання даних було виділено таблиці майбутньої бази даних (БД), визначено домени атрибутів сутностей і зв’язки між сутностями. Розроблену логічну схему БД можна побачити на рисунку 2.3.

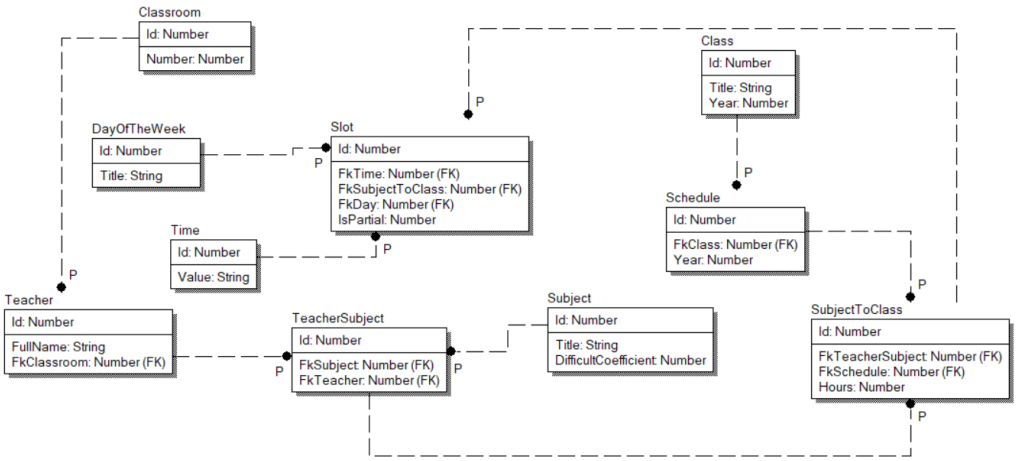

Рисунок 2.3 – Логічна модель даних

На базі проведеного логічного моделювання була побудована фізична модель даних, де визначаються конкретні типи даних, можливість атрибутів зберігати значення NULL тощо. Фізична модель даних представлена на рисунку 2.4.

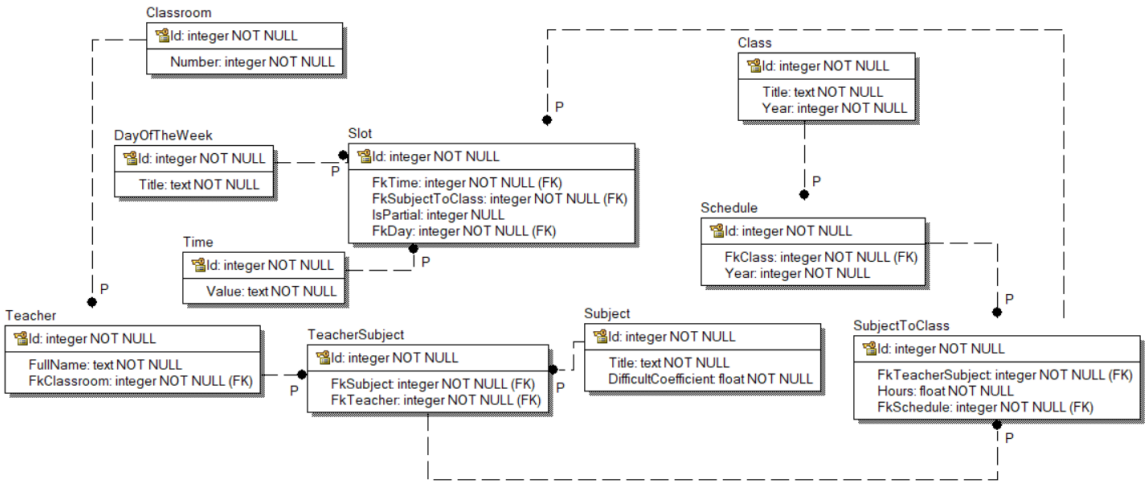

Рисунок 2.4 – Фізична модель даних

Розроблені схеми будувались із використанням стандарту IDEF1X [24-25]. На фізичній моделі даних можна побачити сутності та атрибути.

Наведемо детальний опис кожної таблиці з атрибутами, їх типами даних та інформацією щодо можливості зберігати NULL.

Таблиця Classroom (Аудиторія):

– атрибут Id: унікальний ідентифікатор аудиторії. Має тип integer та не може приймати значення NULL;

– атрибут Number: номер аудиторії. Має тип integer та не може приймати значення NULL.

Таблиця Time (Часовий слот):

– атрибут Id: унікальний ідентифікатор часового слогу. Має тип даних integer та не може бути NULL;

– атрибут Value: чассовий слот у текстовому форматі «HH:MM-HH:MM». Не може бути NULL та має тип даних text.

Таблиця Teacher (Вчитель):

– атрибут Id: унікальний ідентифікатор вчителя. Має тип даних integer та не може бути NULL;

– атрибут FullName: повне ім’я вчителя. Не може бути NULL та має тип даних text;

– атрибут FkClassroom: зовнішній ключ до таблиці Classroom. Не може бути NULL;

Таблиця Subject (Предмет):

– атрибут Id: унікальний ідентифікатор предмета. Має тип даних integer та не може бути NULL;

– атрибут DifficultyCoefficient: коефіцієнт складності предмета. Не може дорівнювати NULL;

– атрибут Title: назва предмета. Не може бути NULL та має тип даних text.

Таблиця Class (Клас):

– атрибут Id: унікальний ідентифікатор класу. Має тип даних integer та не може бути NULL;

– атрибут Title: назва класу (наприклад, 10-А). Не може бути NULL та має тип даних text;

– атрибут Year: рік набору класу. Має тип даних integer та не може бути NULL.

Таблиця Schedule (Розклад):

– атрибут Id: унікальний ідентифікатор розкладу. Має тип даних integer та не може бути NULL;

– атрибут FkClass: зовнішній ключ до таблиці Class. Не може бути NULL;

– атрибут Year: рік навчання. Має тип даних integer та не може бути NULL.

Таблиця TeacherSubject (Вчитель-Предмет):

– атрибут Id: Унікальний ідентифікатор запису. Має тип даних integer та не може бути NULL;

– атрибут FkSubject: Зовнішній ключ до таблиці Subject. Не може бути NULL;

– атрибут FkTeacher: зовнішній ключ до таблиці Teacher. Не може бути NULL.

Таблиця Slot (Слот у розкладі):

– атрибут Id: унікальний ідентифікатор слоту. Має тип даних integer та не може бути NULL;

– атрибут FkSubjectToClass: зовнішній ключ до таблиці SubjectToClass. Не може бути NULL;

– атрибут FkTime: зовнішній ключ до таблиці Time. Не може бути NULL;

– атрибут FkDay: зовнішній ключ таблиці DayOfTHeWeek. Не може приймати значення NULL;

– атрибут IsPartial: відповідає за позначення уроку з неповною кількістю годин (0,5) для того, щоб при необхідності в цей слот можна було розмістити ще одне заняття з кількістю годин 0,5.

Таблиця DayOfTheWeek (День тижня):

– атрибут Id: унікальний ідентифікатор дня тижня. Має тип даних integer та не може бути NULL;

– атрибут Title: назва дня. Не може приймати значення NULL.

Таблиця SubjectToClass (Призначений класу предмет):

– атрибут Id: унікальний ідентифікатор призначеного класу предмету. Має тип даних integer та не може бути NULL;

– атрибут Hours: кількість годин для викладання предмету. Має тип даних float та не може приймати значення NULL;

– атрибут FkTeacherSubject: зовнішній ключ до таблиці TeacherSubject. Не може бути NULL;

– атрибут FkSchedule: зовнішній ключ до таблиці Schedule. Не може приймати значення NULL.

Така структура БД забезпечує цілісність даних, оскільки важливі зв’язки між таблицями не дозволяють існування порожніх значень у ключових полях.

У фізичній моделі зв’язки між таблицями є ключовими для відстеження інформації про розклад, вчителів, класи, предмети й аудиторії. Основні таблиці, такі як Class, Teacher, Subject, Classroom, Time та Schedule, взаємодіють через таблиці зі складними зв’язками, зокрема Slot і TeacherSubject.

Таблиця TeacherSubject відповідає за зв’язок між таблицями Teacher і Subject. Вона зберігає інформацію про те, які вчителі викладають певні предмети. Кожен запис у цій таблиці пов’язує один предмет з одним вчителем через зовнішні ключі до таблиць Teacher і Subject.

Таблиця Slot є центром управління уроками. Вона об’єднує дані з кількох таблиць: Schedule, TeacherSubject, Time, та DayOfTheWeek. Ця таблиця визначає конкретні уроки, які відбуваються у певний час (через зовнішній ключ до таблиці Time), у конкретній аудиторії (через аудиторію вчителя), і викладаються конкретним вчителем на певний предмет (через зовнішній ключ до TeacherSubject). Крім того, кожен урок належить до певного розкладу (через

зовнішній ключ до таблиці Schedule), який, у свою чергу, пов’язаний із класом через таблицю Class.

Таким чином, таблиця Schedule визначає розклад для кожного класу на рік. Кожен запис у таблиці Slot пов’язаний із певним розкладом через зовнішній ключ до Schedule, що дозволяє поєднувати час, предмет, вчителя та аудиторію в конкретні уроки для певного класу. Усі зв’язки мають координальність один до багатьох.

2.4 UML-моделювання ІС

У ході проведення UML-моделювання була розроблена Use-Case діаграма, яка наведена на рисунку 2.5.

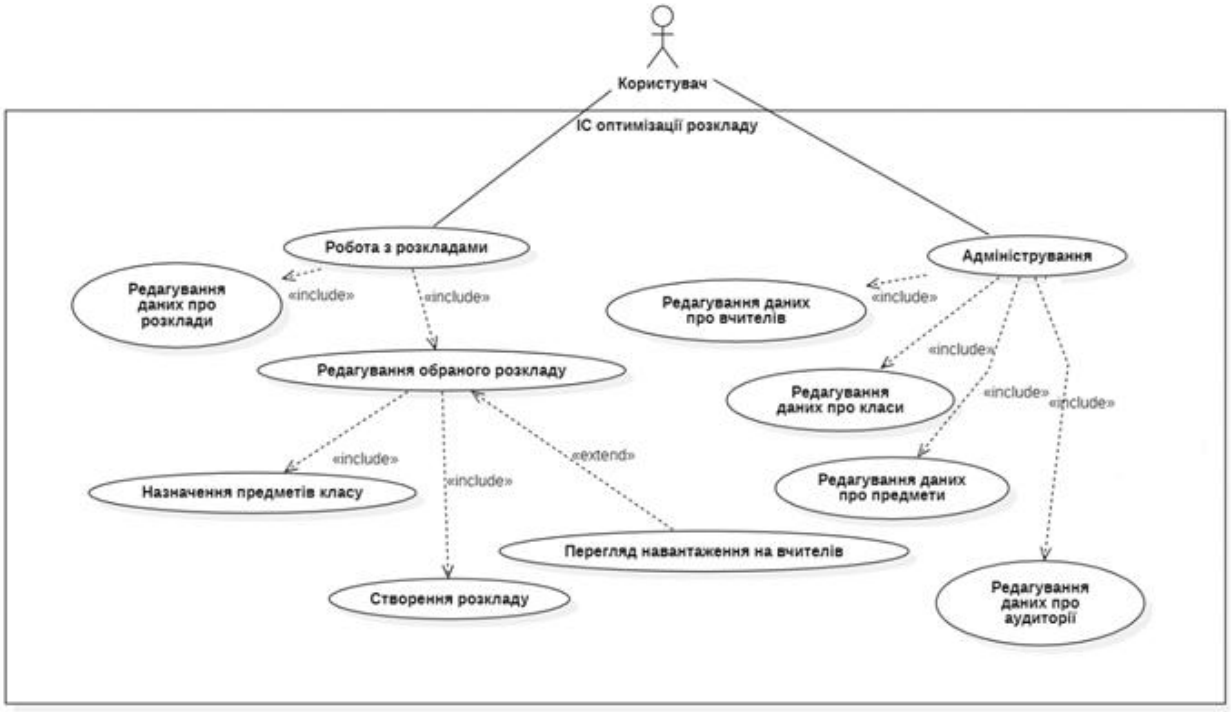

Рисунок 2.5 – Use-Case діаграма ІС

UML-моделювання системи проходило з дотриманням стандарту Unified Modeling Language версії 2.5.1 [26].

В ході аналізу системи та створення діаграми варіантів використання (Use-Case) було визначено одного актора «Користувач» у ролі якого може виступати адміністрація школи, або окремий робітник школи, який відповідальний за складання розкладу занять.

Прецедент «Адміністрування» представляє собою редагування користувачем даних, які є основними для складання розкладу. Кожен з прецедентів, який включено до адміністрування (за допомогою зв’язку include) відбиває редагування відповідної сутності.

Прецедент редагування включає наступні прецеденти:

– прецедент «Редагування даних про вчителів»: додавання, видалення та оновлення даних про вчителів (включає не тільки роботу з особистою інформацією вчителя, але і закріплення предметів та аудиторій за обраним вчителем);

– прецедент «Редагування даних про класи»: додавання, видалення та оновлення даних про класи;

– прецедент «Редагування даних про предмети»: додавання, видалення та оновлення даних про предмети;

– прецедент «Редагування даних про аудиторії»: додавання, видалення та оновлення даних про аудиторії.

Зв’язок include використовувався для кожного з прецедентів, які включені до адміністрування, тому що виконання хоча б одного з них є невід’ємною частиною процесу «Адміністрування».

Прецедент «Робота з розкладами» включає в себе наступні прецеденти:

– прецедент «Редагування даних про розклади»: додавання, видалення та оновлення даних про класи, які включені до розкладу за певний навчальний рік;

– прецедент «Редагування обраного розкладу»: користувач обирає навчальний рік для складання розкладу. Цей прецедент включає декілька інших прецедентів.

Зв’язок include використовувався для кожного з прецедентів, які включені до процесу робота з розкладами, тому що виконання хоча б одного з них є невід’ємною частиною процесу «Робота з розкладами».

Прецедент «Редагування обраного розкладу» включає наступні прецеденти:

– прецедент «Назначення предметів класу»: класу призначаються предмети, які будуть викладатись протягом року з відповідною кількістю годин для обраного предмета;

– прецедент «Створення розкладу»: являє собою створення розкладу. Оскільки цей процес є одним з основних для нього була створена діаграма послідовності, яка буде розглянута далі;

– прецедент «Перегляд навантаження вчителів»: являє собою відкриття нового вікна програми, де будуть відображатись години навантаження вчителів для поточного розкладу.

Зв’язок include використовувався для прецедентів «Назначення предметів класу» та «Створення розкладу», оскільки виконання хоча б одного з них є невід’ємною частиною процесу «Редагування обраного розкладу».

Зв’язок extend використовувався для зв’язку прецедентів «Редагування обраного розкладу» та «Перегляд навантаження вчителів», оскільки виконання прецеденту «Перегляд навантаження вчителів» не є обов’язковим для виконання прецеденту, який він «розширює».

Також в ході UML-проєктування була розроблена діаграма класів, яка наведена на рисунку 2.6.

На діаграмі класів зображено клас DataProviderBase, який буде виступати «основою» для своїх нащадків: ScheduleDataProvider, SlotDataProvider, SubjectToClassDataProvider та TeacherDataProvider. Клас DataProviderBase містить поле, яке зберігатиме шлях до бази даних (БД) та метод для створення з’єднанння з БД.

Клас ScheduleDataProvider містить наступні операції для роботи з БД:

– GetScheduleForAYearAsync: операція буде використовуватись для повернення об’єктів з розширеними даними про розклади за рік. Операція приймає як параметр обраний навчальний рік;

– GetDaysIdsAsync: операція буде використовуватись для отримання списку унікальних ідентифікаторів для кожного дня тижня;

– GetTimeIdsAsync: операція буде використовуватись для отримання списку унікальних ідентифікаторів для кожного часового слота.

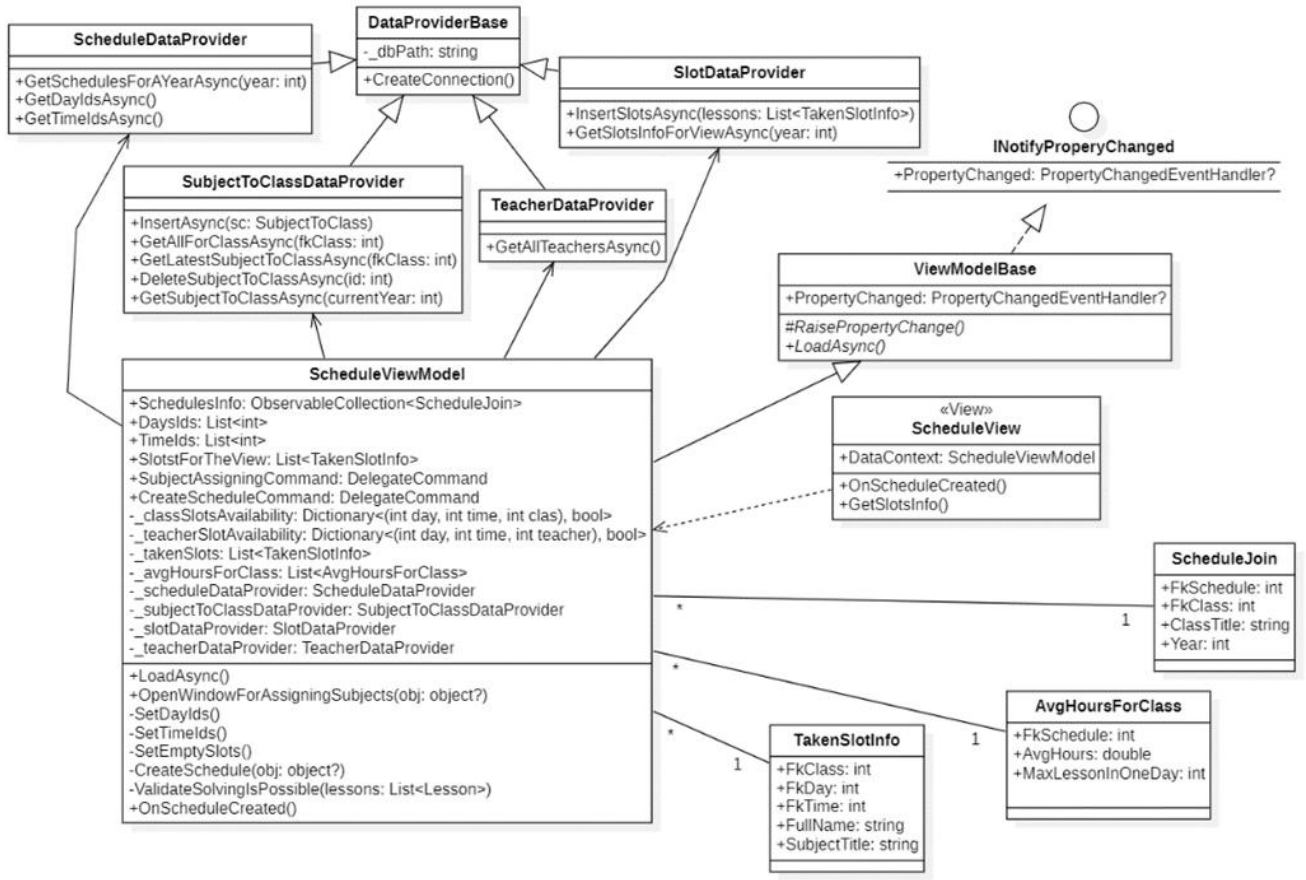

Рисунок 2.6 – Діаграма класів для частини системи

Клас SubjectToClassDataProvider містить наступні операції, які будуть використовуватись для роботи з БД:

– InsertAsync: операція буде використовуватись для додавання запису про призначений предмет для класу до БД. У якості вхідного параметру приймає

об’єкт класу SubjectToClass, поля якого використовуються для внесення даних до БД;

– GetLatestSubjectToClassAsync: операція буде використовуватись для отримання останнього доданого запису;

– DeleteSubjectToClassAsync: операція буде використовуватись для видалення запису про призначений предмет. У якості вхідного параметра приймає унікальний ідентифікатор об’єкту;

– GetSubjectsToClassAsync: операція буде використовуватись для отримання призначених предметів для класу. У якості вхідного параметра приймає обраний навчальний рік для отримання предметів саме за цей рік.

TeacherDataProvider містить наступну операцію для роботи з БД: GetAllTeachersAsync – використовується для повернення даних про вчителів з БД.

SlotDataProvider містить наступну операцію для роботи з БД:

– InsertSlotsAsync: операція буде використовуватись для додавання слотів до розкладу. В якості вхідного параметра приймає список заповнених слотів розкладу;

– GetSlotsForTheViewAsync: операція буде використовуватись для отримання об’єктів, які містять інформацію потрібну для коректного відображення слота для користувача (номер уроку, назва предмета, кабінет тощо).

Також на діаграмі класів зображено уявлення для відображення розкладу. Уявлення містить публічне поле DataContext, яке являє собою контекст даних для роботи уявлення. У випадку уявлення ScheduleView це ScheduleViewModel.

Окрім публічного поля уявлення містить дві операції OnScheduleCreated та GetSlotsInfo. Операція OnScheduleCreated буде викликатись при остаточному та успішному створенні розкладу для відображення зайнятих слотів, а операція GetSlotsInfo буде використовуватись для отримання даних про зайняті слоти розкладу.

Окрім того, на діаграмі класів зображено три моделі: TakenSlotInfo, AvgHoursForClass та ScheduleJoin.

Модель TakenSlotInfo буде використовуватись як об’єкт, який представляє зайнятий слот в розкладі. Модель має наступні публічні поля:

– поле FkClass: являє собою унікальний ідентифікатор класу;

– поле FkDay: являє собою унікальний ідентифікатор дня тижня;

– поле FkTime: являє собою унікальний ідентифікатор часового слоту;

– поле FullName: являє собою повне ім’я вчителя;

– поле SubjectTitle: являє собою назву предмету.

Модель AvgHoursForClass. Об’єкти даного класу будуть використовуватись у якості контейнера з даними про середню кількість годин для класу на день. Модель має наступні публічні поля:

– поле FkSchedule: являє собою унікальний ідентифікатор розкладу класу;

– поле AvgHours: являє собою середню кількість годин на тиждень для класу;

– поле MaxLessonsInOneDay: являє собою максимальну кількість уроків на день для відповідного класу.

Модель ScheduleJoin. Об’єкти даного класу будуть використовуватись для відображення даних про класи в «шапці» таблиці розкладу. Клас має наступні поля:

– поле FkSchedule: являє собою унікальний ідентифікатор розкладу занять класу;

– поле FkClass: являє собою унікальний ідентифікатор класу;

– поле ClassTitle: являє собою назву класу;

– поле Year: являє собою рік набору класу.

Клас ViewModelBase реалізуватиме інтерфейс INotifyPropertyChanged (реалізація даного інтерфейсу необхідна для коректної роботи прив’язки даних моделі-уявлення до елементів уявлення), а також має операцію LoadAsync, яку

будуть використовувати його нащадки для проведення первинного завантаження даних з БД.

Одним з нащадків класу ViewModelBase буде клас ScheduleViewModel. Клас ScheduleViewModel має наступні публічні поля:

– поле SchedulesInfo: являє собою переглядаєму колекцію та буде використовуватись для відображення інформації про розклади класів для поточного навчального року;

– поле DaysIds: являє собою список унікальних ідентифікаторів кожного дня тижня;

– поле TimeIds: являє собою список унікальних ідентифікаторів часових слотів для проведення занять;

– поле SlotsForTheView: представляє собою колекцію з об’єктами класу TakenSlotInfo, яка зберігає дані про зайняті слоти розкладу занять. Колекція буде використовуватись для відображення на уявленні ScheduleView;

– поле SubjectAssigningCommand: поле представляє собою реалізацію команди, яка дозволить зв’язати виконання операції OpenWindowForAssigningSubjects з елементами користувацького інтерфейсу;

– поле CreateScheduleCommand: поле представляє собою реалізацію команди, яка дозволить зв’язати виконання операції CreateSchedule з елементами користувацького інтерфейсу.

Також клас ScheduleViewModel має наступні приватні поля:

– поле _classSlotsAvailability: представляє собою словник з ключем з трьох значень типу int (унікальний ідентифікатор дня тижня, унікальний ідентифікатор часового слоту, унікальний ідентифікатор розкладу класу) та значенням типу bool (відбиває чи слот в розкладі класу вільний). Словник буде використовуватись при виконанні алгоритму оптимізації розкладу;

– поле _teacherSlotsAvailability: являє собою словник з ключем з трьох значень типу int (унікальний ідентифікатор дня тижня, унікальний ідентифікатор часового слота, унікальний ідентифікатор вчителя) та значенням типу bool

(відбиває чи слот в розкладі вчителя вільний). Словник буде використовуватись при виконанні алгоритму оптимізації розкладу;

– поле _takenSlots: являє собою список із зайнятими слотами розкладу. Список буде використовуватись при виконанні алгоритму оптимізації розкладу;

– поле _avgHours: являє собою список де зберігається інформація про навантаження для кожного класу;

– поле _scheduleDataProvider: об’єкт класу ScheduleDataProvider для надавання доступу до даних таблиці Schedule;

– поле _subjectToClassDataProvider: об’єкт класу SubjectToClass DataProvider для надавання доступу до даних таблиці Subject ToClass;

– поле _slotDataProvider: об’єкт класу SlotDataProvider для надавання доступу до даних таблиці Slot;

– поле _teacherDataProvider: об’єкт класу TeacherDataProvider для надавання доступу до даних таблиці Teacher.

Окрім полів клас ScheduleViewModel має наступні публічні операції:

– операція LoadAsync є публічною та буде викликатись при ініціалізації моделі-представлення для завантаження необхідних даних з БД;

– операція OpenWindowForAssigningSubjects є публічною та буде викликатись для відкриття вікна з можливістю редагування призначених класу предметів;

– операція OnScheduleCreated є публічною та буде викликатись при успішному створенні розкладу занять та слугувати тригером для початку процесу відображення заповненого розкладу для уявлення ScheduleView;

– операція SetDaysIds призначена для заповнення списку з унікальними ідентифікаторами днів тижня;

– операція SetTimeIds призначена для заповнення списку з унікальними ідентифікаторами часових слотів;

– операція SetEmptySlots призначена для заповнення словників з слотами для класів та вчителів із початковим значенням true (позначення, що слот вільний).

На основі діаграм, які були розроблені на попередініх етапах, була створена діаграма послідовності дій (sequence-diagram) для основного прецеденту – «Створення розкладу», яка наведена на рисунку 2.7.

В ході виконання прецеденту «Складання розкладу» користувач натискає кнопку «Скласти розклад». Ця подія викликає команду CreateScheduleCommand, яка вказує на метод CreateSchedule всередині моделі-уявлення ScheduleViewModel. Наступним чином система перевіряє чи можливо вирішення задачі та залежно від результату перевірки надає користувачу відповідний результат.

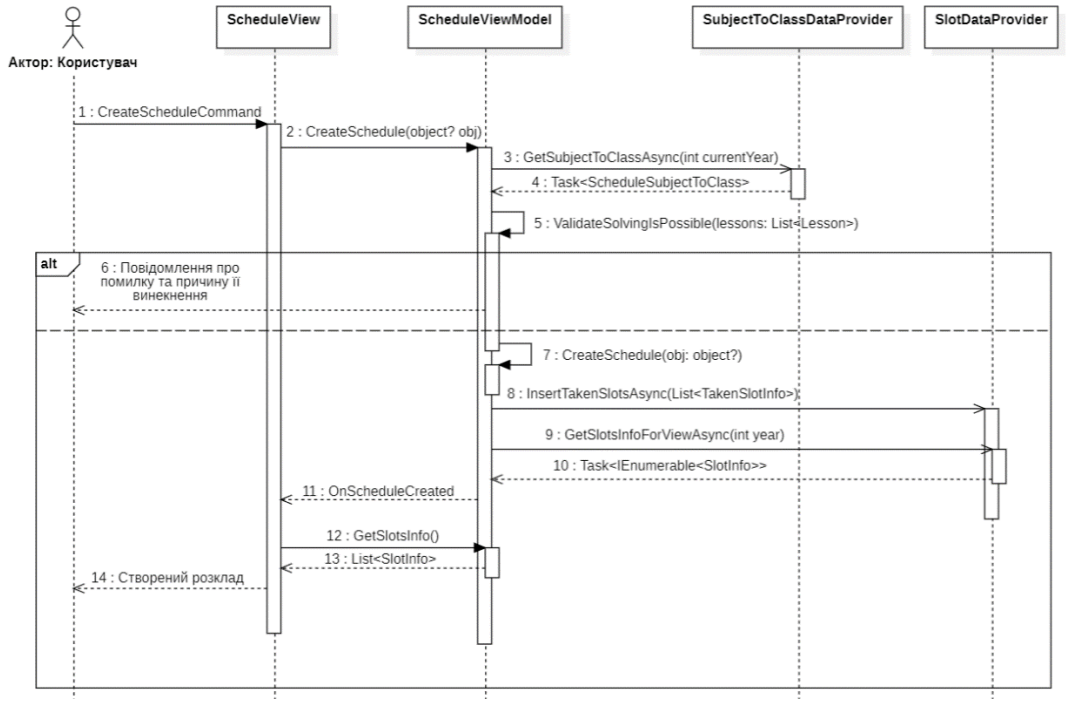

Рисунок 2.7 – Діаграма послідовності дій для прецеденту «Створення розкладу»

Якщо за результатами валідації наданих даних рішення не може бути знайдено, то користувач має побачити спливаюче вікно з помилкою та її

можливою причиною. Саме тому, що вікно з помилкою не є частиною уявлення стрілка відповіді на діаграмі вказує одразу на користувача.

Якщо за результатами перевірки вхідних даних рішення задачі складання розкладу є теоретично можливим, система починає виконувати алгоритм створення розкладу. Після успішного складання розкладу модель-уявлення ScheduleViewModel заносить записи про заповнені слоти до БД за допомогою поля _slotDataProvider, а потім отримує ці дані з потрібними для відображення на уявленні полями (номер уроку, назва предмета, номер дня тощо).

Наступним чином модель-уявлення заповнює список з даними про слоти, який використовується для відображення на уявленні. Після чого спрацьовує подія OnScheduleCreated та уявлення отримує дані про слоти. Останній крок – уявлення відбиває отримані дані за допомогою елементів розмітки на екрані користувача.

2.5 Розробка математичної моделі системи

Перед створенням алгоритму оптимізації розкладу необхідно виразити задачу у математичному вигляді. Отже, у розділі 2.1 наводилась загальна математична модель, яку використовують для розв’язання задач програмування методом обмежень. Її треба підвести до конкретної задачі (створення оптимізованого розкладу).

В ході процесу математичного вираження задачі були виділені наступні змінні:

$$S_k = \{s_k 1, s_k 2, \dots, s_{k_T}\},$$

де S_k – набір доступних слотів у розкладі кожного класу;

T – загальна кількість слотів у розкладі;

$$S_{tch} = \{s_{tch} 1, s_{tch} 2, \dots, s_{tchT}\},$$

де S_{tch} – набір доступних слотів у розкладі кожного вчителя.

Для кожного слоту вчителя та класу буде визначена змінна наступного типу:

$$X_{kt} \in \{0,1\},$$

де X_{kt} – бінарна змінна для слота класу.

Змінна приймає значення 1, якщо слот *t* класу *k* є зайнятим, і приймає значення 0 в іншому випадку.

Для вчителя змінні слотів виглядатимуть наступним чином:

$$Y_{tch_t} \in \{0,1\},$$

де Y_{tch_t} – бінарна змінна для слота вчителя.

Змінна приймає значення 1, якщо слот *t* вчителя *tch* є зайнятим, і приймає значення 0 в іншому випадку.

Після визначення змінних треба визначити обмеження для регулювання вмісту розкладу. Клас може займати свій слот. Якщо слот є зайнятим, то клас не може його використовувати як вільний. Тобто:

$$X_{kt} \leq 1, \forall \mathrm{k}, t.$$

Якщо слот класу зайнято уроком, то слот у розкладі вчителя, який викладає предмет, також повинен бути зайнятим. Таким чином:

$$X_{kt} = Y_{tch_t}, \forall k, t, tch.$$

Обмеження для уникнення «вікон» у розкладі класу виглядає наступним чином:

$$X_{kt1} = 1 \wedge X_{kt2} = 1 \Rightarrow X_{kt} = 1, \forall t \in [t_1 + 1, t_2 - 1].$$

Це є основними обмеженнями, надалі у програмне рішення будуть внесені більш детальні обмеження типу розподілу «складних» предметів на дні з вівторка по середу та інші обмеження, які можуть бути порушені. Обмеження, які наведені в математичній моделі, порушувати не можна.

Також визначено дві цільові функції, за якими буде оцінюватись наскільки оптимізованим є створений розклад.

Перша цільова функція буде направлена на мінімізацію кількості вікон між заняттями. Для кожного класу «вікна» визначаються як кількість проміжків між заняттями за наступною формулою:

$$G_k = \sum_{t=1}^{T-1} \left(1 - X_{kt} \cdot X_{k,t+1}\right).$$

Тоді цільова функція на мінімізацію кількості вікон виглядає наступним чином:

$$min \sum_{k} G_k.$$

Друга цільова функція буде направлена на те, щоб вівторок-середа були самими навантаженими днями. Навантаженість дня для класу k визначається за наступною формулою:

$$K_{d,k} = \sum_{t \in S_k^d} C_{sub},$$

де C_{sub} – коефіцієнт складності предмета у слоті;

d – порядковий номер дня тижня.

Отже, цільова функція для кожного з класів буде виглядати наступним чином:

$$max(K_{2,k} + K_{3,k}).$$

Описана математична модель дозволить мати чітку структуру через розділення множини слотів вчителів і класів, дозволить гнучко додавати обмеження (за необхідності) та добре підходить для визначення алгоритму програмної реалізації.

2.6 Розробка алгоритму створення розкладу

Алгоритм оптимізації розкладу створено, спираючись на міжнародний стандарт ICO 5807-85 [27]. Один з кроків алгоритму наведено на рисунку 2.8.

На початку алгоритму проходить отримання даних з БД про предмети, уроки та кількість годин для кожного класу у вигляді списку для кожної колекції даних.

Наступним чином змінним *i* та *n* присвоюються відповідні значення. Змінній *i* присвоюється значення 0, оскільки вона буде використовуватись у циклі, а змінна *n* дорівнює кількості уроків для всіх класів.

Наступним чином алгоритм попадає до циклу while з умовою виконання нової ітерації, якщо змінна *i* є меншою за змінну *n*. Якщо змінна *i* є меншою, то проводиться процес присвоєння змінним значень для подальшої обробки у циклі. Змінна slotInfo буде використовуватись для отримання даних про урок, який у цей момент оброблюється у циклі while. Змінна availableSlots набуває значення списку вільних слотів класу, урок якого в цей момент оброблюється. А змінна avgHours являє собою об’єкт з даними про середнє навантаження класу.

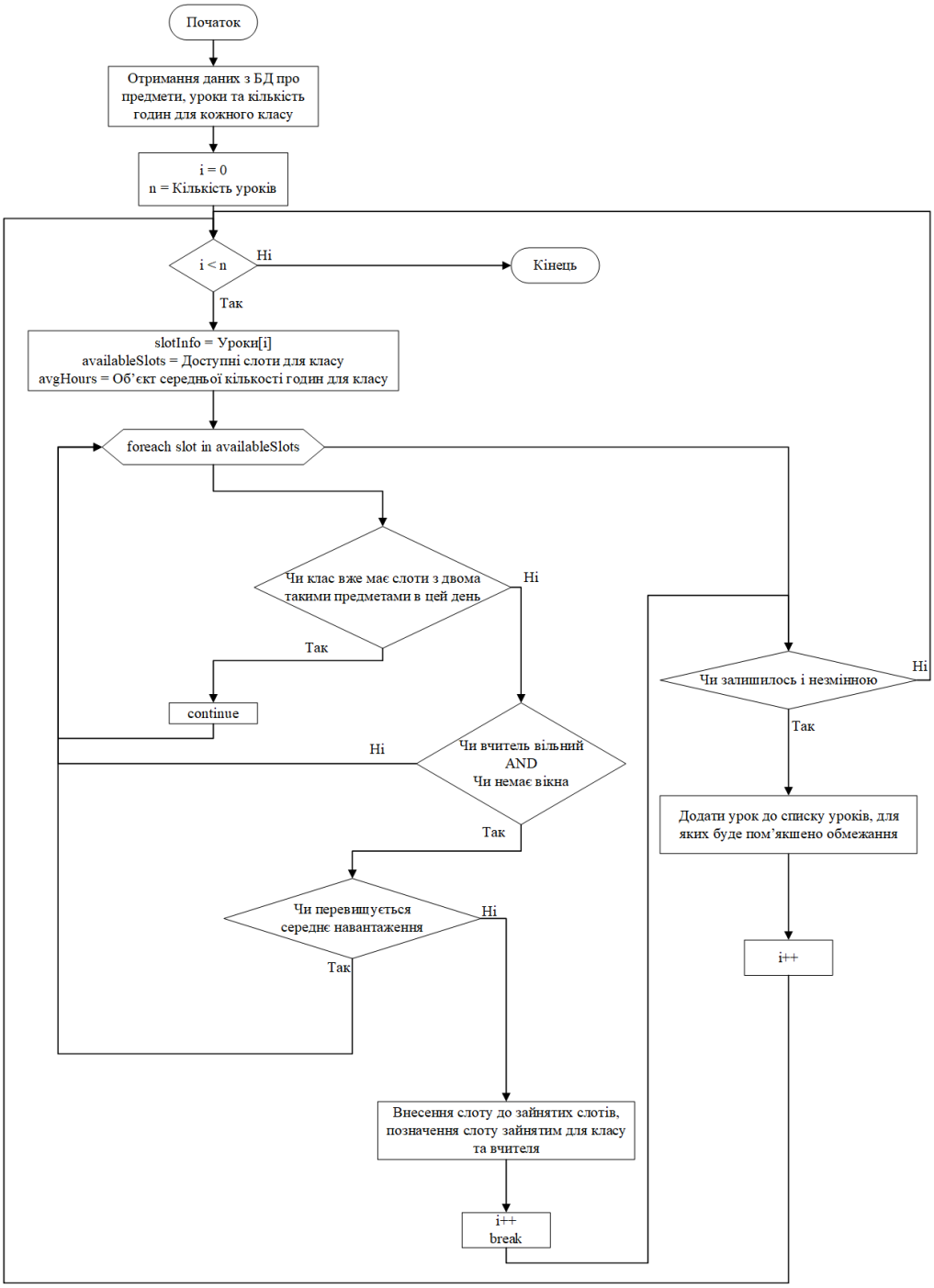

Рисунок 2.8 – Один з кроків алгоритму створення розкладу

Далі алгоритм потрапляє у цикл foreach для кожного слоту зі списку доступних слотів для класу. Спочатку у циклі перевіряється, чи клас вже має два слоти з однаковими предметами за цей день. Якщо умова повертає true, цикл пропускає даний слот і переходить до перевірки іншого слоту. Якщо умова повертає false, то слот проходить наступну перевірку.

Наступна перевірка складається з двох умов, обидві з яких повинні повернути true для того, щоб слот міг пройти останню перевірку. Якщо хоча б одна з умов повертає false, алгоритм пропускає поточний слот і переходить до перевірки наступних. Одна з умов перевіряє, чи є слот вільним у розкладі вчителя, а інша умова перевіряє відсутність вікна між заняттями.

Якщо слот успішно пройшов перевірку, то він переходить до фінальної умови, де перевіряється, чи не перевищується середнє навантаження класу на день. Якщо навантаження перевищується – алгоритм переходить до наступного можливого слоту для поточного уроку. Якщо умова повертає true, то виконується процес внесення слоту до зайнятих слотів, визначення слоту зайнятим у розкладі класу та у розкладі вчителя.

Наступним кроком збільшується значення змінної *i* на одиницю та виконання циклу foreach переривається.

Перед виконанням наступної ітерації циклу while перевіряється, чи залишилась змінна *i* незмінною, якщо умова повертає true, то алгоритм виконує додавання поточного уроку до списку уроків, для яких буде пом’якшено обмеження додавання уроку до розкладу, після чого значення змінної *i* збільшується на одиницю та цикл while починає проводити чергову ітерацію.

Коли змінна *i* досягне значення, яке буде дорівнювати значенню змінної *n*, алгоритм закінчить своє виконання. Описаний алгоритм буде повторюватись, поки всі уроки не знайдуть слот.

Спочатку уроки для класів будуть проходити через найбільш строгі обмеження, та з кожною ітерацією у них буде все більше вільних доступних слотів. Спочатку будуть розподілені складні уроки (уроки, де коефіцієнт складності предмету більший або дорівнює одна ціла сім десятих). Якщо в цьому випадку залишаться уроки, для яких не знайшлося місця, то вони будуть переміщені в іншу колекцію з іншими уроками (коефіцієнт складності предмета в яких менше за одну цілу сім десятих) та ця колекція буде використовуватись у подальшому.

Під час розподілу уроків, які не відповідали умовам, буде виключено обмеження на дні (обмеження для складних уроків – розподіл з вівторка по середу). Всі інші обмеження залишаються незмінними.

Якщо після цього кроку ще залишились уроки, то знімається обмеження на загальну максимальну кількість уроків на день для всіх класів. Таке обмеження потрібне для того, щоб алгоритм не знайшов рішення, за якого, наприклад, для одного з класів кількість уроків в один із днів дорівнювала 2 або менше та навантаження було рівномірним.

Єдиними обмеженнями, які ні в якому разі не можна порушувати та вони завжди повинні виконуватись, є обмеження на відсутність конфліктів і відсутність вікон.

2.7 Висновки до другого розділу

У рамках другого розділу було розглянуто декілька алгоритмів оптимізації розкладу. Як базу для алгоритму було обрано метод обмежень із використання пом’якшення обмежень.

Також у другому розділі розглянуто особливості складання розкладу ЗНЗ, проаналізовані коефіцієнти складності предметів і правила складання оптимального розкладу. У такому розкладі максимальна сума коефіцієнтів складності предметів формується у вівторок-середу.

Окрім того, у другому розділі проведене логічне та фізичне моделювання даних, UML-моделювання, створення математичної моделі й алгоритму створення розкладу занять ЗНЗ. Отримані результати моделювання системи будуть слугувати теоретичною базою під час розробки застосунку, який реалізує алгоритм оптимізації розкладу.

3 ОПИС ПРИЙНЯТИХ ПРОЄКТНИХ РІШЕНЬ

3.1 Опис архітектури системи

Як архітектуру системи обрано архітектуру MVVM (Model-View-ViewModel) з реалізацією на платформі WPF. Архітектура MVVM заснована на чіткому розподілі відповідальностей між трьома основними компонентами: модель, представлення та модель-представлення. Це допомагає розділити бізнес-логіку, керування даними та інтерфейс користувача, роблячи програму гнучкішою та легшою для тестування.

Модель (Model) представляє бізнес-логіку та дані, що використовуються в застосунку. Вона відповідає за доступ до даних, управління ними та виконання логічних операцій. Модель відповідає за роботу з даними та їх зберігання, не взаємодіючи при цьому з інтерфейсом користувача.

Представлення (View) є візуальною частиною програми, яка відображає інформацію для користувача. У WPF це зазвичай XAML-файл, де описується інтерфейс користувача. Представлення не містить логіки бізнесу та не взаємодіє безпосередньо з даними – воно просто показує те, що йому передають інші компоненти, і реагує на події користувача.

Модель-представлення (ViewModel) виступає посередником між моделлю та представленням. Вона містить логіку, яка перетворює дані з моделі у зручний для відображення формат і забезпечує керування командою та взаємодію з інтерфейсом. ViewModel взаємодіє з моделлю для отримання даних та обробки подій, але не залежить від представлення напряму, що дозволяє легко тестувати логіку без необхідності залучення інтерфейсу користувача [28].

Windows Presentation Foundation (WPF) – це графічна підсистема від Microsoft для створення настільних застосунків на платформі Windows. Вона є частиною .NET і дозволяє розробникам створювати інтуітивні та функціональні інтерфейси користувача з використанням декларативного програмування за

допомогою XAML (Extensible Application Markup Language). WPF надає можливості для побудови сучасних інтерфейсів із використанням таких технологій, як 2D та 3D графіка, анімація, стильові шаблони, прив’язка даних (data binding) та інші інструменти для побудови динамічних та інтерактивних інтерфейсів [29].

Основні переваги WPF включають підтримку моделі MVVM (Model-View-ViewModel), потужні можливості для створення анімацій та медіа, а також підтримку темізації та стилізації інтерфейсу.

Завдяки механізмам прив’язки даних (data binding) у WPF, представлення автоматично відображає дані, що містяться у компоненті View Model, та реагує на зміни без необхідності прямої взаємодії з ним. Це дозволяє зберігати інтерфейс і логіку окремо, зберігаючи чистоту коду та полегшуючи його підтримку. Графічне представлення архітектури MVVM наведено на рисунку 3.1.

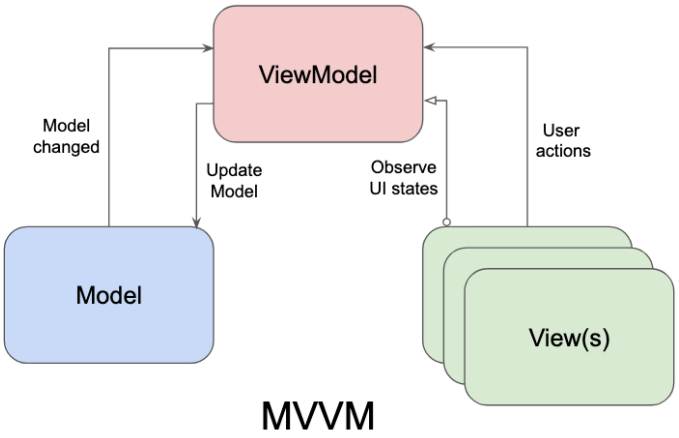

Рисунок 3.1 – Архітектура MVVM

Як система управління БД (СУБД) була обрана SQLite. SQLite – це вбудована реляційна база даних, яка не вимагає окремого серверного програмного забезпечення для управління базами даних [30]. Вона зберігає всю базу даних у одному файлі на диску та використовується переважно для застосунків, де не потрібні великі обсяги даних або складні операції з

багатокористувацьким доступом. SQLite є однією з найпопулярніших баз даних у мобільних застосунках, настільних програмах і для прототипування. Вона забезпечує повноцінний набір можливостей SQL і підтримує транзакції, індекси та різні типи даних.

Для доступу до даних буде використовуватись бібліотека, яка частково реалізує технологію object-relational mapping (ORM), Dapper. Dapper – це бібліотека з відкритим кодом для об'єктно-реляційного відображення (ORM) для застосунків на платформі .NET та .NET Core [31].

3.2 Розробка запитів до БД системи

У ході виконання етапу розробки запитів до БД були розроблені запити, які допоможуть реалізувати наступні функції:

– перегляд навантаження вчителів;

– редагування даних про класи;

– редагування даних про вчителів;

– редагування даних про предмети;

– редагування даних про аудиторії;

– редагування даних про розклади;

– назначення предметів класу;

– створення розкладу.

Запит, який виконує функцію перегляду навантаженості на вчителів наведено на рисунку 3.2.

```
SELECT t.Id, FullName, SUM(Hours) AS SumOfHours FROM Teacher t
INNER JOIN TeacherSubject ts ON t.Id = ts.FkTeacher
INNER JOIN SubjectToClass sc ON ts.Id = sc.FkTs
GROUP BY t.Id
```

Рисунок 3.2 – Запит для наведення даних про навантаження вчителів

Запит для наведення навантаження вчителів використовує оператор SELECT для вибору даних з таблиці Teacher, а також доєднує дані з таблиць про співвідношення вчителів та предметів, таблиці призначення предмета відповідному класу. Запит групує дані за унікальним ідентифікатором вчителя за допомогою оператора GROUP BY. Для розрахунку сумарного часу навантаження вчителя використовується оператор SUM. Полем для розрахунку слугує поле Hours (назначена кількість годин на тиждень відповідного предмета, який викладає вчитель). Приклад результатів виконання запиту наведено на рисунку 3.3.

	Id	FullName	SumOfHours
1	1	Іваненко Іван Іванович	20.0
2	2	Петренко Петро Петрович	20.0
3	3	Сидоренко Сидір Сидорович	20.0
4	4	Коваленко Ольга Миколаївна	22.0
5	5	Гриценко Наталія Василівна	26.0
6	6	Мельник Андрій Вікторович	26.0
7	7	Шевченко Марина Олександрівна	21.0
8	8	Данилюк Олег Анатолійович	19.0

Рисунок 3.3 – Результати виконання запиту для надавання даних навантаження вчителів

Також у ході виконання етапу розробки запитів до системи були розроблені запити, які реалізують функцію «Редагування даних про класи». Запит внесення нового класу до системи наведено на рисунку 3.4.

```
INSERT INTO Class (Title, Year) VALUES ('1-A', 2024)
```

Рисунок 3.4 – Запит для додавання класу до системи

Запит використовує оператор INSERT INTO для додавання запису до таблиці Class та вказує потрібні для цього поля (Title та Year). За допомогою оператора VALUES вказуються значення полів, які будуть внесені до системи.

Значення унікального ідентифікатора Id заповнюється за допомогою властивості поля AUTO-INCREMENT.

Також було розроблено запит оновлення даних про Class. Запит оновлення даних про клас наведено на рисунку 3.5.

```
UPDATE Class SET Title = @Title, Year = @Year WHERE Id = @Id
```

Рисунок 3.5 – Запит оновлення даних про клас

Запит використовує оператор UPDATE для оновлення записів таблиці Class. За допомогою оператора SET встановлюються потрібні значення полів запису в таблиці Class, а за допомогою оператора WHERE запит обирає потрібний запис про клас з таблиці за допомогою унікального ідентифікатора Id.

Також створено запит вибірки класів для їх перегляду в меню редагування даних про класи. Розроблений запит для вибірки класів наведено на рисунку 3.6.

```
SELECT Id, Title, Year FROM Class;
```

Рисунок 3.6 – Запит для вибірки класів

Запит використовує запит SELECT для того, щоб отримати потрібні дані з таблиці Class, а саме: унікальний ідентифікатор класу, назва класу та рік набору класу. За допомогою оператора FROM визначається потрібна таблиця для отримання записів. Результати виконання запиту SELECT для таблиці Class наведено на рисунку 3.7.

	Id	Title	Year
1	1	1-А	2024
2	2	1-Б	2024
3	3	2-А	2023
4	4	2-Б	2023

Рисунок 3.7 – Результати виконання запиту SELECT для таблиці Class

Також створено запит для видалення класів. Розроблений запит для видалення класів наведено на рисунку 3.8.

```
DELETE FROM Class WHERE Id = @Id;
```

Рисунок 3.8 – Запит на видалення класу

Запит використовує оператор DELETE для того, щоб видалити дані про рядок з таблиці. За допомогою оператора FROM визначається потрібна таблиця для видалення даних. За допомогою оператора WHERE та унікального ідентифікатора, який в ньому визначається, обирається запис для видалення.

Також створено запит для перевірки, чи не створено запис у пов’язаній таблиці, який буде використовуватись перед проведенням видалення запису в таблиці Class. Запит наведено на рисунку 3.9.

```
SELECT EXISTS(SELECT 1 FROM Schedule WHERE FkClass = @FkClass);
```

Рисунок 3.9 – Запит на виявлення записів із зовнішнім ключем таблиці Class у пов’язаній таблиці

Запит використовує зв’язку операторів SELECT EXISTS для перевірки, чи існують записи у таблиці Schedule (таблиця розкладів), де використовуються дані для визначеного класу. Запит буде виконуватись перед видаленням класу. Результат роботи запиту буде переведено у тип даних bool програмним шляхом (оскільки SQLite не має типу даних bool) для того, щоб полегшити обробку отриманих даних.

У ході розробки запитів для реалізації функції «Редагування даних про вчителів» були розроблені наступні запити до БД: запит для вибору наявних у системі вчителів, запити для оновлення даних про вчителя, запит для вибору призначених вчителю предметів, а також редагування (додавання, видалення)

предметів, які викладає вчитель. Також були розроблені запити для видалення вчителя та перевірки цілісності даних.

Запит для вибору наявних у системі вчителів наведено на рисунку 3.10.

```
SELECT Id, FullName FROM Teacher;
```

Рисунок 3.10 – Запит для вибору наявних у системі вчителів

Запит використовує оператор SELECT для того, щоб отримати потрібні дані з таблиці Teacher, а саме: унікальний ідентифікатор вчителя та його повне ім’я. За допомогою оператора FROM визначається потрібна таблиця для отримання записів. Результати виконання запиту SELECT для таблиці Teacher наведено на рисунку 3.11.

	Id	FullName
1	1	Іваненко Іван Іванович
2	2	Петренко Петро Петрович
3	3	Сидоренко Сидір Сидорович
4	4	Коваленко Ольга Миколаївна
5	5	Гриценко Наталія Василівна
6	6	Мельник Андрій Вікторович

Рисунок 3.11 – Результати виконання запиту SELECT для таблиці Teacher

Також було розроблено запит оновлення даних таблиці Teacher. Запит оновлення даних про вчителя наведено на рисунку 3.12.

```
UPDATE Teacher SET FullName = @FullName WHERE Id = @Id
```

Рисунок 3.12 – Запит на оновлення даних про вчителя

Запит використовує оператор UPDATE для оновлення записів таблиці Teacher. За допомогою оператора SET оновлюються потрібні значення полів запису у таблиці Teacher, а за допомогою оператора WHERE запит обирає

потрібний запис про вчителя з таблиці за допомогою унікального ідентифікатора Id. Унікальний ідентифікатор запису Id не дозволяється оновлювати для дотримання цілісності даних БД.

Також було розроблено запит для внесення даних до таблиці Teacher. Приклад запиту наведено на рисунку 3.13.

```
INSERT INTO Teacher(FullName) VALUES('Пісклов Микола Альбертович');
```

Рисунок 3.13 – Запит для додавання вчителя до системи

Запит використовує оператор INSERT INTO для додавання запису до таблиці Teacher та вказує потрібні для цього поля (FullName). За допомогою оператора VALUES вказуються значення полів, які будуть внесені до системи. Значення унікального ідентифікатора Id заповнюється за допомогою властивості поля AUTO-INCREMENT.

Розроблений запит для видалення вчителя з системи наведено на рисунку 3.14.

```
DELETE FROM Teacher WHERE Id = @Id;
```

Рисунок 3.14 – Запит видалення вчителя

Запит використовує оператор DELETE для того, щоб видалити дані про рядок з таблиці. За допомогою оператора FROM визначається потрібна таблиця для видалення даних, а саме таблиця Teacher. За допомогою оператора WHERE та унікального ідентифікатора, який в ньому визначається, обирається запис про вчителя для видалення.

Також створено запити для перевірки, чи не створено запис у пов’язаних таблицях, який буде використовуватись перед проведенням видалення запису в таблиці Teacher. Запит наведено на рисунку 3.15.

Запити використовують зв’язку операторів SELECT EXISTS для перевірки, чи існують записи у таблиці TeacherSubject (таблиці співвідношення

вчителя та предмету), де використовуються дані для визначеного вчителя. Запити будуть виконуватись перед видаленням вчителя. Результат роботи запитів буде переведено у тип даних bool для того, щоб полегшити обробку отриманих даних.

```
SELECT EXISTS(SELECT 1 FROM TeacherSubject WHERE FkTeacher = @FkTeacher);
```

Рисунок 3.15 – Запити на виявлення записів із зовнішнім ключем таблиці Teacher у пов’язаних таблицях

Запит для вибору предметів, які викладає вчитель, наведено на рисунку 3.16.

```
SELECT s.Id, s.Title
FROM Subject s
INNER JOIN TeacherSubject t ON t.FkSubject = s.Id
WHERE t.FkTeacher = @TeacherId
```

Рисунок 3.16 – Запит вибору предметів, які викладає вчитель

Запит використовує оператор SELECT для вибору визначених полів таблиці, яка містить дані про предмети, а саме: унікальний ідентифікатор Id та назву предмета. За допомогою оператора FROM визначається початкова таблиця Subject, а за допомогою оператора INNER JOIN обираються відповідні записи в таблиці, яка містить дані про призначені вчителю предмети. Відповідні записи обираються, на основі однакових значень полів Id з таблиці Subject та FkSubject з таблиці TeacherSubject. Також за допомогою оператора WHERE визначається, предмети якого вчителя будуть відображатись на екрані редагування вчителів. Для цього запиту надається унікальний ідентифікатор вчителя.

Також були створені запити для редагування предметів, які викладає вчитель. Запит на додавання предмету на викладання вчителю наведено на рисунку 3.17.

```
INSERT INTO TeacherSubject (FkTeacher, FkSubject)
VALUES (@FkTeacher, @FkSubject);
```

Рисунок 3.17 – Запит для внесення запису у таблицю співвідношення вчителя та предметів

Запит на рисунку 3.17 використовує оператор INSERT INTO для додавання запису до таблиці TeacherSubject та вказує потрібні для цього поля (FkTeacher та FkSubject), які являють собою унікальні ідентифікатори вчителя та предмету відповідно. За допомогою оператора VALUES вказуються значення полів, які будуть внесені до системи. Значення унікального ідентифікатора Id заповнюється за допомогою властивості поля AUTO-INCREMENT.

Запит на видалення закріпленого за вчителем предмета наведено на рисунку 3.18.

```
DELETE FROM TeacherSubject Where Id = @Id
```

Рисунок 3.18 – Запит на видалення запису з таблиці TeacherSubject

Запит використовує оператор DELETE для видалення запису з таблиці співвідношення вчителя та його предметів, а за допомогою оператора WHERE визначається конкретний предмет, який назначено відповідному вчителю за допомогою унікального ідентифікатора запису.

Для підтримки цілісності даних створено запит, який перевіряє, чи призначено цей предмет для одного з класів. Запит буде викликатись перед виконанням видалення запису для того, щоб визначити чи можна видаляти запис, чи це призведе до порушення цілісності даних БД. Код розробленого запиту наведено на рисунку 3.19.

```
SELECT EXISTS(SELECT 1 FROM SubjectToClass WHERE FkTs = @FkTs);
```

Рисунок 3.19 – Запит на виявлення записів в таблиці SubjectToClass з визначеним значенням унікального ідентифікатора запису з TeacherSubject

Запит використовує зв’язку операторів SELECT EXISTS для перевірки, чи існують записи у таблиці SubjectToClass (таблиця призначених класу предметів), де визначений вчитель викладає предмет. Запит буде використовуватись перед видаленням предмета, який викладає вчитель, та результат його роботи буде переведено у тип даних bool для того, щоб полегшити роботу з результатами його роботи.

Запити, які входять до адміністрування, а саме для редагування даних про предмети, аудиторії та розклади, розроблені за описаними раніше прикладами запитів для редагування даних про вчителів і класи.

Також були розроблені запити для прив’язки предметів до класу. Запит для відображення списку предметів, які можна призначити класу, наведено на рисунку 3.20.

```
SELECT ts.Id, FkTeacher, FkSubject, Title AS SubjectTitle, FullName AS TeachersName
FROM TeacherSubject ts INNER JOIN Teacher t ON ts.FkTeacher = t.Id
INNER JOIN Subject s ON s.Id = ts.FkSubject ORDER BY s.Title ASC;
```

Рисунок 3.20 – Запит для відображення списку предметів, які можна призначити класу

Запит використовує оператор SELECT для вибору потрібної інформації, а саме: унікальний ідентифікатор запису в таблиці співвідношення предмету та вчителя, унікальні ідентифікатори вчителя та предмету, назву предмету та ім’я вчителя. За допомогою оператора FROM запит визначає, дані з яких таблиць будуть відображатись. За допомогою оператора INNER JOIN дані з таблиць поєднуються за обмеженням зовнішнього ключа. За допомогою оператора ORDER BY записи сортуються за назвою предметів за абеткою.

Запит для надання списку предметів, які вже призначені класу, наведено на рисунку 3.21.

Запит використовує оператор SELECT для вибору визначених полів таблиць, які містять дані про предмети. За допомогою оператора FROM визначається початкова таблиця SubjectToClass, а за допомогою оператора

INNER JOIN обираються відповідні записи в таблиці, яка містить дані про призначені вчителю предмети. Також використовуються дані з таблиці, яка містить дані про вчителя, предмет і розклад. Відповідні записи обираються на основі однакових значень полів Id з таблиць із відповідними зовнішніми ключами. Також за допомогою оператора WHERE визначається клас, для якого будуть обрані предмети та рік розкладу.

```
SELECT sc.Id AS ScId, FullName, Title, Hours FROM SubjectToClass sc
INNER JOIN  TeacherSubject ts ON sc.FkTs = ts.Id
INNER JOIN Teacher t ON ts.FkTeacher = t.Id
INNER JOIN Subject s ON ts.FkSubject = s.Id
INNER JOIN Schedule sch ON sc.FkSchedule = sch.Id
WHERE FkClass = @FkClass AND sc.Year = @Year;
```

Рисунок 3.21 – Запит надання предметів, призначених відповідному класу

Результати виконання запитів, які наведені на рисунках 3.20-3.21, можна побачити на рисунках 3.22-3.23 відповідно.

Id	FkTeacher	FkSubject	SubjectTitle	TeachersName
25	2	3	Іноземна мова	Петренко Петро Петрович
51	15	3	Іноземна мова	Іноземець Інозем Іноземович
38	8	20	Інформатика	Данилюк Олег Анатолійович
63	21	20	Інформатика	Ломать Іван Іванович
28	3	9	Історія	Сидоренко Сидір Сидорович
59	19	9	Історія	Орлов Микола Іванович
23	1	2	Алгебра	Іваненко Іван Іванович
49	14	2	Алгебра	Наумов Іван Іванович
31	4	6	Біологія	Коваленко Ольга Миколаївна
53	16	6	Біологія	Шилов Іван Петрович

Рисунок 3.22 – Результати запиту для відображення предметів, які можна призначити класу

ScId	FullName	Title	Hours
312	Мельник Андрій Вікторович	Українська література	2.0
313	Іноземець Інозем Іноземович	Іноземна мова	3.0
314	Шликов Ігор Ігорович	Зарубіжна література	2.0
315	Сидоренко Сидір Сидорович	Історія	1.0
316	Сидоренко Сидір Сидорович	Всесвітня історія	1.0
318	Бондар Олена Сергіївна	Образотворче мистецтво	1.0
319	Шевченко Марина Олександрівна	Математика	4.0

Рисунок 3.23 – Результати запиту для відображення призначених предметів

Також були розроблені запити для додавання та видалення призначеного класу предмету, а також запит на перевірку цілісності даних. Запит для додавання призначеного для класу предмету наведено на рисунку 3.24.

```
INSERT INTO SubjectToClass (FkTs, FkSchedule, Hours) VALUES (@FkTs, @FkSchedule, @Hours);
```

Рисунок 3.24 – Запит для додавання призначеного для класу предмету

Запит використовує оператор INSERT INTO для внесення записів до таблиці SubjectToClass. Також вказуються поля, значення яких будуть внесені до рядка таблиці. У випадку запиту з рисунка 3.24 це унікальний ідентифікатор співвідношення вчителя та предмета, унікальний ідентифікатор розкладу та кількість годин на викладання предмета за тиждень. За допомогою оператора VALUES отримуються відповідні значення для вказаних атрибутів. Унікальний ідентифікатор Id набуває значення за допомогою властивості поля AUTO-INCREMENT.

У запиті на видалення використовується оператор DELETE FROM для того, щоб вказати таблицю, запис в якій треба видалити та за допомогою оператора WHERE вказується унікальний ідентифікатор запису в таблиці SubjectToClass, який треба видалити.

Для реалізації створення розкладу розроблено запит, який надає дані про заповнені слоти обраного розкладу. Запит повинен відображати назву предмета, унікальний ідентифікатор дня тижня та часового слота, а також повне ім’я вчителя. Наведені дані будуть використовуватись для коректного відображення заповнених слотів у таблиці розкладу. Запит наведено на рисунку 3.25.

Запит використовує запит SELECT для того, щоб обрати унікальний ідентифікатор класу, ідентифікатор дня, ідентифікатор часового слоту, повне ім’я вчителя та назву предмету. Для співвідношення даних з різних таблиць використовується оператор INNER JOIN, за допомогою якого поєднуються таблиці кабінету, призначених предметів для класу, предметів, які викладають

вчителі, таблиця з даними про предмети, таблиця з даними про вчителів та розклади.

```
SELECT FkClass AS ClassId, FkDay AS DayId, FkTime AS TimeId,
FullName, sub.Title AS SubjectTitle,  cr.Number AS ClassroomNumber  FROM Slot s
INNER JOIN SubjectToClass sc ON s.FkSubjectToClass = sc.Id
INNER JOIN TeacherSubject ts ON sc.FkTs = ts.Id
INNER JOIN Subject sub ON ts.FkSubject = sub.Id
INNER JOIN Teacher t ON ts.FkTeacher = t.Id
INNER JOIN Classroom cr ON t.FkClassroom = cr.Id
INNER JOIN Schedule sch ON sc.FkSchedule = sch.Id
WHERE sch.Year = @Year;
```

Рисунок 3.25 – Запит для надання даних про заповненні слоти

Дані заповнених слотів для поточного навчального року обираються за допомогою оператора WHERE з умовою порівняння поля, яке зберігає рік розкладу з поточним навчальним роком.

Результати виконання запиту для відображення заповнених слотів розкладу наведено на рисунку 3.26.

	ClassId	DayId	TimeId	FullName	SubjectTitle	ClassroomNumber
1	9	2	1	Гриценко Наталія Василівна	Українська мова	105
2	11	2	1	Мельник Андрій Вікторович	Українська мова	106
3	19	2	1	Шилов Іван Петрович	Хімія	302
4	17	2	1	Шевченко Марина Олександрівна	Фізика	201
5	17	2	2	Сидоренко Сидір Сидорович	Історія	103
6	15	2	1	Іноземець Іноземович Іноземович	Іноземна мова	301
7	15	2	2	Шевченко Марина Олександрівна	Фізика	201
8	19	3	1	Шевченко Марина Олександрівна	Фізика	201
9	18	3	1	Шилов Іван Петрович	Біологія	302
10	13	2	1	Коваленко Ольга Миколаївна	Хімія	104
11	10	3	1	Іноземець Іноземович Іноземович	Іноземна мова	301
12	16	2	1	Іваненко Іван Іванович	Геометрія	101
13	14	2	1	Сидоренко Сидір Сидорович	Історія	103
14	14	2	2	Гриценко Наталія Василівна	Українська мова	105
15	11	2	2	Мельник Андрій Вікторович	Українська література	106
16	14	2	3	Іваненко Іван Іванович	Геометрія	101

Рисунок 3.26 – Результати запиту для відображення даних про заповнені слоти розкладу

Також для реалізації функції створення розкладу розроблено запит, який додає запис про слот розкладу. Запит для додавання слота наведено на рисунку 3.27.

```
INSERT INTO Slot (FkSubjectToClass, FkDay, FkTime, FkClassroom)
VALUES (@FkSubjectToClass, @FkDay, @FkTime, @FkClassroom)
```

Рисунок 3.27 – Запит додавання запису до таблиці зі слотами розкладу

Запит використовує оператор INSERT INTO для додавання запису до таблиці Slot та вказує потрібні для цього поля (FkSubjectToClass, FKDay, FkTime та FkClassroom), які являють собою унікальні ідентифікатори предмету, який призначено класу, ідентифікатор дня тижня, ідентифікатор часового слоту та ідентифікатор кабінету відповідно. За допомогою оператора VALUES вказуються значення полів, які будуть внесені до системи. Значення унікального ідентифікатора Id заповнюється за допомогою властивості поля AUTO-INCREMENT.

3.3 Розробка класів та методів класів

Основою програмного застосунку є наступні розроблені класи:

– DataProviderBase;

– ViewModelBase та ValidationViewModelBase;

– MainViewModel;

– DelegateCommand.

Всі інші розроблені класи є або нащадками наведених вище класів, або знаходяться у тісній взаємодії з ними.

Наприклад, клас DataProviderBase надає шлях до локальної БД SQLite та створює зв’язок із нею, що можуть використовувати його нащадки. Код розробленого класу наведено на рисунку 3.28.

```
using System.Data;
using System.IO;
using Microsoft.Data.Sqlite;

namespace Schedule.DataProviders
{
    6 references
    public class DataProviderBase
    {
        private string _dbPath = Path
            .Combine(AppDomain.CurrentDomain.BaseDirectory, @"..\..\..\Schedule.db");
        20 references
        public IDbConnection CreateConnection()
        {
            return new SqliteConnection($"Data Source={_dbPath}");
        }
    }
}
```

Рисунок 3.28 – Код класу DataProviderBase.cs

На початку підключаються необхідні простори імен: System.Data для загальних інтерфейсів роботи з базами даних, таких як IDbConnection, System.IO для роботи з файловою системою, та Microsoft.Data.Sqlite для підключення до бази даних SQLite.

У класі визначена приватна змінна _dbPath, яка зберігає шлях до файлу бази даних SQLite. Цей шлях формується за допомогою методу Path.Combine, що поєднує базовий каталог програми (отриманий через AppDomain.CurrentDomain.BaseDirectory) із відносним шляхом до файлу бази даних, який знаходиться на три рівні вище відносно поточної директорії.

Метод CreateConnection використовується для створення з'єднання з базою даних SQLite. Він повертає новий екземпляр SqliteConnection, який ініціалізується рядком підключення, де вказується шлях до бази даних. Цей метод реалізує інтерфейс IDbConnection, що робить його універсальним для різних типів баз даних, якщо клас буде змінено для використання іншої бази. На рисунку 3.29 наведено приклад, як нащадок може використовувати створення зв'язку з БД.

```
2 references
public async Task<IEnumerable<Class>?> GetAllAsync()
{
    using(var connection = CreateConnection())
    {
        var sql = "SELECT Id, Title, Year FROM Class;";
        var classes = await connection.QueryAsync<Class>(sql);
        return classes.ToList();
    }
}
```

Рисунок 3.29 – Метод класу ClassDataProvider.cs

Метод GetAllAsync є асинхронним методом, який отримує всі записи з таблиці Class у базі даних.

Метод використовує ключове слово async, що вказує на його асинхронність, а також повертає Task<IEnumerable<Class>?>, що означає, що метод повертає колекцію об’єктів типу Class, обгорнуту у завдання (Task), яке виконується асинхронно.

У тілі методу за допомогою оператору using створюється з’єднання з базою даних через метод CreateConnection(), який створює новий об’єкт з’єднання з SQLite.

SQL-запит, що зберігається у змінній sql, вибирає стовпці Id, Title та Year з таблиці Class. Метод QueryAsync<Class> викликається для виконання цього SQL-запиту асинхронно, і результати зберігаються у змінній classes. Цей метод повертає список об’єктів типу Class, створених на основі результатів запиту.

Нарешті, результати запиту конвертуються у список за допомогою ToList() і повертаються як результат виконання методу.

Клас ViewModelBase є базовим класом для всіх об’єктів типу модель-представлення у застосунку, який реалізує інтерфейс INotifyPropertyChanged для забезпечення механізму повідомлення про зміни властивостей. Цей інтерфейс необхідний для автоматичного оновлення інтерфейсу користувача, коли змінюються властивості у ViewModel.

Клас має подію PropertyChanged, яка сигналізує про те, що певна властивість була змінена.

Клас також містить три захищені константи, які можна використовувати у похідних класах для спрощення роботи з вікнами повідомлень:

– _iconSuccess – відображає іконку успішного результату (MessageBoxImage.Asterisk);

– _iconFail – відображає іконку помилки (MessageBoxImage.Error);

– _cancelButton – встановлює кнопку підтвердження (MessageBoxButton.OK).

Метод RaisePropertyChange є захищеним і віртуальним, що дозволяє похідним класам перевизначати його. Цей метод викликається для підняття події PropertyChanged і приймає параметр propertyName, що визначає назву властивості, яка змінилася. За допомогою атрибута [CallerMemberName] значення цього параметра автоматично заповнюється ім’ям викликаного члена, якщо метод викликається без аргументів.

Метод LoadAsync є віртуальним і повертає завершене завдання за замовчуванням, реалізуючи асинхронну функцію, яку можуть перевизначати дочірні класи для завантаження даних або виконання будь-яких асинхронних операцій.

Таким чином, цей клас забезпечує базовий функціонал для ViewModel в архітектурі MVVM, спрощуючи реалізацію прив’язки даних і асинхронного завантаження під час ініціалізації ViewModel.

Клас ValidationViewModelBase є нащадком ViewModelBase та реалізує логіку валідації даних.

Клас MainViewModel є основною моделлю-представленням (ViewModel) для застосунку, побудованого за архітектурою MVVM. Він наслідує базовий клас ViewModelBase, що забезпечує функціональність для прив’язки даних і повідомлення про зміни властивостей.

Клас містить властивість SelectedViewModel, яка відповідає за вибір поточної ViewModel для відображення у представленні (View). Ця властивість має приватне поле _selectedViewModel, яке зберігає поточну ViewModel. При встановленні нового значення для SelectedViewModel викликається метод RaisePropertyChange(), що сигналізує інтерфейсу користувача про зміну властивості, дозволяючи оновити відображення.

У класі також є інші екземпляри ViewModel, які використовуються при перемиканні SelectedViewModel.

DelegateCommand – це команда для виконання певної дії, тут вона використовується для команди SelectMenuItemCommand, яка відповідає за зміну поточного представлення. Команда викликає метод SelectMenuItem, який передає параметр до об’єкту Messanger.Instance для зміни View.

Конструктор MainViewModel ініціалізує команду SelectMenuItemCommand і підписується на події від об’єкта Messanger.Instance.

Метод SelectMenuItem перевіряє, чи переданий параметр не є null, і якщо це так, він надсилає нове значення для зміни представлення через Messanger.Instance.

Метод LoadAsync перевизначає базовий метод і завантажує дані для поточного екземпляра ViewModel.

Метод OnViewChanged викликається при зміні представлення через подію ViewChanged. Він оновлює поле SelectedViewModel на нову ViewModel і асинхронно викликає метод LoadAsync для завантаження даних.

Модель-представлення ScheduleViewModel, в якій представлений основний функціонал програми (створення розкладу для загальноосвітнього навчального закладу), реалізує алгоритм, який було описано у розділі 2.6.

Програмна реалізація алгоритму (додаток Б) використовує цикл while для знаходження слоту кожному з виявлених складних уроків, які зібрані у колекції difficultLessons.

На початку циклу визначається змінна startingI для зберігання стартового значення змінної *i*. Цей крок потрібен для подальшої перевірки в кінці виконання ітерації циклу. Наступним чином об’єкт з даними про урок класу записується у змінну slotInfo. Об’єкт з даними про урок класу обирається за допомогою змінної *i*. Далі алгоритм отримує колекцію доступних для класу слотів та записує їх у змінну availableSlots. Дані про доступні слоти збираються зі словника ClassSlotsAvailability. У випадку розподілу складних предметів обираються тільки слоти з вівторка по середу. Також отримуються дані про середнє навантаження класу (дані записуються до змінної avgHours).

Наступним кроком програма проходить по всіх вільних слотах за допомогою циклу foreach.

Першою умовою перевіряється, чи клас вже має два предмети у цей день. Метод IsClassHaveTwoSameSubjectsInDay є приватним методом класу ScheduleViewModel та має наступні вхідні параметри:

– параметр типу Lesson;

– параметр, який приймає ключ словника ClassSlotsAvailability.

Метод починає виконання з того, що перевіряє колекцію TakenSlots (зайняті слоти розкладу) на наявність слотів, які підходять до поставлених умов. Умови перевіряють співвідношення унікального ідентифікатора предмета, який призначено класу, та дня тижня, слот якого у цей момент перевіряється. У змінну subjectCount записується кількість слотів, зайнятих цим предметом. Описані операції виконуються за допомогою бібліотеки LINQ.

Якщо кількість слотів, зайнятих цим предметом, менша за 2, то метод повертає false. Це позначає, що клас у цей день ще не має більше одного слота, які зайняті поточним предметом. В іншому випадку метод повертає true.

Якщо метод IsClassHaveTwoSameSubjectsInDay повертає true, то цикл, який проходить по доступних слотах для класу, одразу переходить до наступного доступного слоту в розкладі. В іншому випадку виконання циклу переходить до наступних двох умов.

Наступні умови перевіряють, чи є вчитель вільним у цей слот розкладу та чи немає вікна між заняттями.

Метод для перевірки, чи є вчитель вільним, має назву IsTeacherFree. Метод IsTeacherFree є приватним, має повертати значення типу bool та приймає наступні вхідні параметри:

– параметр типу Lesson;

– параметр, який приймає ключ словника ClassSlotsAvailability.

Метод починає виконання з присвоєння змінній teacherKey значення, яке підходить для ключа словника TeacherSlotsAvailability (колекція зі слотами вчителя). Наступним кроком до змінної isTeacherFree записується значення, яке отримано за допомогою ключа у вигляді змінної teacherKey зі словника TeacherSlotsAvailability.

Потім метод IsTeacherFree повертає значення true, якщо вчитель вільний у визначений слот, та false у іншому випадку.

Метод для перевірки вікна між слотами має назву IsPreviousSlotTaken. Метод IsPreviousSlotTaken першим кроком перевіряє, чи є час проведення уроку першим за день. Якщо умова виконується, то вікна бути не може та метод повертає true. Наступним чином метод знаходить попередній урок у розкладі класу за поточний день та записує результат пошуків у змінну previousLesson. Наступним кроком метод віднімає порядковий номер поточного уроку та попереднього. Якшо результат віднімання дорівнює значенню, яке є більшим за одиницю, то метод повертає false (що позначає наявність вікон між заняттями). В іншому випадку метод повертає true.

Наступним кроком метод перевіряє, чи дотримується денна норма навантаження на клас, за допомогою методу IsDayHoursNormal та об’єкту типу double, який представляє середнє навантаження на день для конкретного класу та розраховується всередині об’єкту AvgHours. Список з об’єктами AvgHours, створюється за допомогою методу CalculateAvgHoursForClasses до початку виконання алгоритму створення розкладу.

Якщо потенційний слот пройшов перевірки всіх обмежень, день і час позначаються зайнятим у словнику доступних слотів класу, а також відповідний слот позначається зайнятим для вчителя, який викладає предмет, у словнику зі списком слотів для вчителів. Також урок, для якого було знайдено вільний слот, додається до колекції з об’єктами, які зберігають дані про зайняті слоти.

Метод AddSlotToTakenSlots є приватним та приймає наступний набір вхідних параметрів:

– параметр типу Lesson;

– комбінацію пари унікальних ідентифікаторів дня тижня та часу проведення заняття.

Метод формує об’єкт TakenSlotInfo за допомогою присвоєння значень своїм полям, відповідним полям об’єкта Lesson, та комбінацію пари унікальних ідентифікаторів дня тижня та часу проведення заняття. Після цього створений об’єкт додається до колекції TakenSlots, яка є полем класу ScheduleViewModel.

Останнім етапом перевірки під час розподілу слотів є перевірка змінної *i*. Якщо ця змінна залишилася незмінною впродовж виконання циклу foreach, це сигналізує про те, що уроку не знайшлось місця у розкладі, якщо до нього застосовані поточні обмеження. У такому випадку урок, який в цей момент оброблюється програмою, відправляється до колекції уроків, для яких буде пом’якшено обмеження, а змінна *i* збільшується на одиницю.

Після того, як розклад успішно створено, викликається метод OnScheduleCreated. Метод OnScheduleCreated використовує цикл foreach для внесення зайнятих слотів до БД за допомогою об’єкта, який відповідає за роботу з запитами, до таблиці Slot. Також за допомогою іншого циклу foreach всередині методу проходить підготовка колекції, а саме колекції SlotsForTheView, яка буде використовуватись для відображення даних про слоти користувачу. Після цього за допомогою об’єкту типу Messanger викликається подія, на яку підписане представлення ScheduleView. Виклик цієї події слугує тригером для представлення, після спрацювання якого треба починати відображати слоти для користувача за допомогою елементів інтерфейсу.

Для реалізації функції призначення предметів для класу було розроблено клас SubjectToClassViewModel, який буде відповідати за цей клас функціоналу. Для того, щоб відкрити нове вікно користувача, на якому будуть відображатись відповідні елементи інтерфейсу, було створено метод OpenWindowForAssigningSubjects. На цей метод вказує відповідна команда, яка прив’язана до події натискання на кнопку для кожного класу. Метод створює контекст даних для нового вікна у вигляді об’єкта SubjectToClassViewModel. За допомогою конструктора цього об’єкту визначається потрібний клас та інші важливі поля. Після цього проводиться завантаження потрібних даних та ініціалізація нового вікна з інтерфейсом для редагування даних про призначені класу предмети.

SubjectToClassViewModel має наступний список полів:

– поле типу ObservableCollection<TeacherSubjectInfo> під назвою SubjectsInfo є колекцією, що зберігає інформацію про зв’язки між вчителями та

їхніми предметами. Це дозволяє автоматично відображати зміни в інтерфейсі користувача за допомогою інструментів Data Binding;

– поле типу ObservableCollection<SubjectToClassInfo> під назвою AssignedSubjects представляє колекцію предметів, які призначені класу, також реалізовану як ObservableCollection для підтримки динамічних змін за допомогою інструментів Data Binding;

– властивість Hours є числовим полем типу double, яке використовується під час додавання предмету для викладання класу. Поле прив’язано до відповідного елементу інтерфейсу користувача та динамічно змінює своє значення за допомогою інструментів Data Binding;

– поле SelectedTS типу TeacherSubjectInfo зберігає вибраний зв’язок між вчителем і предметом, який використовується для додавання обраного предмета до призначених предметів. Поле SelectedTS динамічно змінює своє значення за допомогою інструментів Data Binding та прив’язано до елементу зі списку, що випадає інтерфейсу користувача;

– властивість SelectedSubjectToClass типу SubjectToClassInfo зберігає вибраний зв’язок між предметом і класом, який використовується для видалення призначеного предмету. Поле може динамічно змінюватись за допомогою інструментів Data Binding та прив’язано до обраного елементу призначених предметів;

– властивість ScheduleInfo типу ScheduleJoin містить об’єднану інформацію про розклад класу, предмети якого редагуються на даний момент;

– команда InsertCommand є об’єктом типу DelegateCommand, що використовується для додавання даних про призначений предмет і вказує на метод InsertSubjectToClass. Команда прив’язана до відповідної кнопки, яка є одним з елементів користувацького інтерфейсу, за допомогою інструментів Data Binding;

– команда DeleteCommand також є об’єктом типу DelegateCommand. Команда вказує на метод DeleteSubjectToClass та прив’язана до відповідної кнопки за допомогою інструментів Data Binding;

– приватне поле _tsDataProvider є екземпляром інтерфейсу ITeacherSubjectDataProvider і використовується для надання доступу до таблиці, яка містить інформацію про співвідношення вчителів і предметів;

– приватне поле _subjectToClassDataProvider є екземпляром інтерфейсу ISubjectToClassDataProvider і використовується для надання доступу до таблиці, яка містить інформацію про співвідношення класів і предметів;

– приватне поле _hours типу double, доступ до якого надає властивість Hours;

– приватне поле _selectedTS типу TeacherSubjectInfo, доступ до якого надає властивість SelectedTS;

– приватне поле _selectedSubjectToClass типу SubjectToClassInfo, доступ до якого надає властивість SelectedSubjectToClass.

Всередині SubjectToClassViewModel класу були реалізовані методи додавання та видалення предмету, який призначено класу, які використовують об’єкти доступу до таблиці БД з даними про призначені предмети та видаляють (з попередньою перевіркою на цілісність даних) або додають записи до неї. Також розроблені класи для реалізації інших функцій системи.

3.4 Проведення експерименту та тестування програми

Для проведення експерименту буде використовуватись комп’ютер з процесором 12th Gen Intel(R) Core(TM) i7-12700H 2.70 GHz та об’ємом ОЗУ 16 Гб. Як середовище розробки буде використовуватись Visual Studio 2022.

Метою експерименту буде оптимізація розкладу загальноосвітнього навчального закладу (зведення до мінімуму кількості «вікон» у розкладі занять та оптимізація навантаження).

Для перевірки результатів роботи алгоритму розроблена частина програми, яка буде перевіряти показники цільових функцій. Розроблений код програми для тестування наведено на рисунку 3.30.

```
int windowCount = 0;
var tempSlots = ClassSlotsAvailability.Where(x=>x.Value == false).ToList();
for(int i = 0; i < tempSlots.Count-1; i++)
{
    if ((tempSlots[i].Key.time - tempSlots[i+1].Key.time > 1)
        && (tempSlots[i].Key.day == tempSlots[i + 1].Key.day)
        && (tempSlots[i].Key.clas == tempSlots[i + 1].Key.clas))
    {
        windowCount++;
    }
}
Console.WriteLine($"Window Count: {windowCount}");
foreach(var clas in SchedulesInfo)
{
    foreach(var day in DaysIds)
    {
        var diffCount = TakenSlots.Where(x=>x.ScheduleId == clas.ScheduleId && x.DayId == day).Sum(x=>x.Diff);
        Console.WriteLine($"Class: {clas.ClassTitle}, Day: {day}, Diff: {diffCount}");
    }
    Console.WriteLine();
}
```

Рисунок 3.30 – Код, розроблений для тестування роботи алгоритму

Алгоритм для перевірки значень цільових функцій використовує колекцію із заповненими слотами TakenSlots та словник із зайнятими слотами класу ClassSlotsAvailability. Спочатку проходить ініціалізація змінної windowCount, яка буде відповідати за розраховану кількість вікон між заняттями. Наступним чином елементи зі словника ClassSlotsAvailability у вигляді списку надсилаються до змінної tempSlots. До цієї колекції відносяться слоти, значення зі словника для яких дорівнює false. Далі програма проходить за елементами колекції tempSlots за допомогою циклу for. Умови, за якими лічильник вікон буде збільшуватись, є наступними: якщо різниця між номерами уроків є більша за одиницю та дні слотів збігаються, а також класи для слотів збігаються, то між заняттями присутнє вікно. Після виконання циклу кількість вікон у розкладі виводиться до консолі.

Наступним чином програма розраховує сумарну складність кожного дня для кожного класу. Аналіз розкладу виконується, враховуючи класи, дні та коефіцієнт складності предмета. Для цього використовуються два цикли foreach. Перший цикл проходить за кожним класом розкладу, а цикл, який розміщено всередині першого, проходить по кожному дню розкладу. Всередині циклу

визначається сума складності предметів за день, результат записується до змінної diffCount та виводиться на екран консолі. Результати розрахунків наведено на рисунку 3.31.

```
Window Count: 0
Class: 5-А, Day: 1, Diff: 16,6
Class: 5-А, Day: 2, Diff: 10,899999999999999
Class: 5-А, Day: 3, Diff: 17,700000000000003
Class: 5-А, Day: 4, Diff: 7
Class: 5-А, Day: 5, Diff: 7,2

Class: 5-Б, Day: 1, Diff: 12,700000000000001
Class: 5-Б, Day: 2, Diff: 22,400000000000002
Class: 5-Б, Day: 3, Diff: 7,2
Class: 5-Б, Day: 4, Diff: 7,500000000000001
Class: 5-Б, Day: 5, Diff: 9,600000000000001

Class: 6-А, Day: 1, Diff: 18,8
Class: 6-А, Day: 2, Diff: 10,799999999999999
Class: 6-А, Day: 3, Diff: 13
Class: 6-А, Day: 4, Diff: 17,400000000000002
Class: 6-А, Day: 5, Diff: 7,9

Class: 6-Б, Day: 1, Diff: 10,799999999999999
Class: 6-Б, Day: 2, Diff: 21,2
Class: 6-Б, Day: 3, Diff: 19,8
Class: 6-Б, Day: 4, Diff: 10,5
Class: 6-Б, Day: 5, Diff: 5,6000000000000005

Class: 7-А, Day: 1, Diff: 19,2
Class: 7-А, Day: 2, Diff: 16,6
Class: 7-А, Day: 3, Diff: 28,2
Class: 7-А, Day: 4, Diff: 23,5
Class: 7-А, Day: 5, Diff: 12,9

Class: 7-Б, Day: 1, Diff: 17,3
Class: 7-Б, Day: 2, Diff: 22,999999999999996
Class: 7-Б, Day: 3, Diff: 31,6
Class: 7-Б, Day: 4, Diff: 12
Class: 7-Б, Day: 5, Diff: 16,5

Class: 8-А, Day: 1, Diff: 27,1
Class: 8-А, Day: 2, Diff: 8,5
Class: 8-А, Day: 3, Diff: 11,799999999999999
Class: 8-А, Day: 4, Diff: 24,7
Class: 8-А, Day: 5, Diff: 26,799999999999997

Class: 8-Б, Day: 1, Diff: 18,5
Class: 8-Б, Day: 2, Diff: 32
Class: 8-Б, Day: 3, Diff: 23,1
Class: 8-Б, Day: 4, Diff: 15
Class: 8-Б, Day: 5, Diff: 10,3

Class: 9-А, Day: 1, Diff: 17,8
Class: 9-А, Day: 2, Diff: 30,9
Class: 9-А, Day: 3, Diff: 24,6
Class: 9-А, Day: 4, Diff: 16,200000000000003
Class: 9-А, Day: 5, Diff: 13,100000000000001

Class: 9-Б, Day: 1, Diff: 23,3
Class: 9-Б, Day: 2, Diff: 25,7
Class: 9-Б, Day: 3, Diff: 26,9
Class: 9-Б, Day: 4, Diff: 14,499999999999998
Class: 9-Б, Day: 5, Diff: 12,200000000000003

Class: 10-А, Day: 1, Diff: 16,099999999999998
Class: 10-А, Day: 2, Diff: 26,5
Class: 10-А, Day: 3, Diff: 28,2
Class: 10-А, Day: 4, Diff: 15,299999999999999
Class: 10-А, Day: 5, Diff: 21,400000000000002

Class: 11-А, Day: 1, Diff: 17,3
Class: 11-А, Day: 2, Diff: 31,700000000000003
Class: 11-А, Day: 3, Diff: 16,8
Class: 11-А, Day: 4, Diff: 14,1
Class: 11-А, Day: 5, Diff: 10
```

Рисунок 3.31 – Результати розрахунків тестування

За результатами розрахунків можна зазначити, що кількість вікон у розкладі дорівнює 0, що є одним з основних критеріїв оцінки успішності складання розкладу занять загальноосвітнього навчального закладу.

Вимоги щодо навантаження класів полягали у тому, що найбільше навантаження на клас повинно припадати на вівторок та середу. Для дотримання реалістичного сценарію складання розкладу більшість вчителів, які викладають предмети у розкладі, мають навантаження мінімум в одну ставку (18 год на

тиждень). Також треба зазначити, що деякі вчителі, які викладають предмети з найбільшою кількістю годин (українська мова, література, математика тощо), мають навантаження більше, ніж на одну ставку. Це є ключовим фактором під час проведення оцінки дотримання вимог навантаження на класи.

Отже, за результатами тестування навантаження більшості класів відповідають поставленим вимогам. Деякі з класів відповідають частково: лише вівторок або середа є днями з найбільшим навантаженням. Тільки розклади двох класів не повною мірою відповідають критеріям, але різниця навантаження між днями навчання не є суттєвою. Описану проблему можна вирішити тільки шляхом розподілення навантаження за більшою кількістю вчителів. Враховуючи описане навантаження на вчителів, можна зробити висновок, що алгоритм відповідає вимогам, тому що навіть в умовах великого навантаження деяких вчителів алгоритм складає розклад ЗНЗ, в якому розклад більшості класів (10 з 12 класів) відповідають вимогам до розкладу, описаним у розділі 2.2.

3.5 Опис інтерфейсу користувача

Інтерфейс користувача розроблено на основі постановки задачі, яка була описана у підрозділі 1.3. Головне вікно програми наведено на рисунку 3.32. У головному вікні програми користувач може побачити кнопки для роботи з відповідними складовими розкладу.

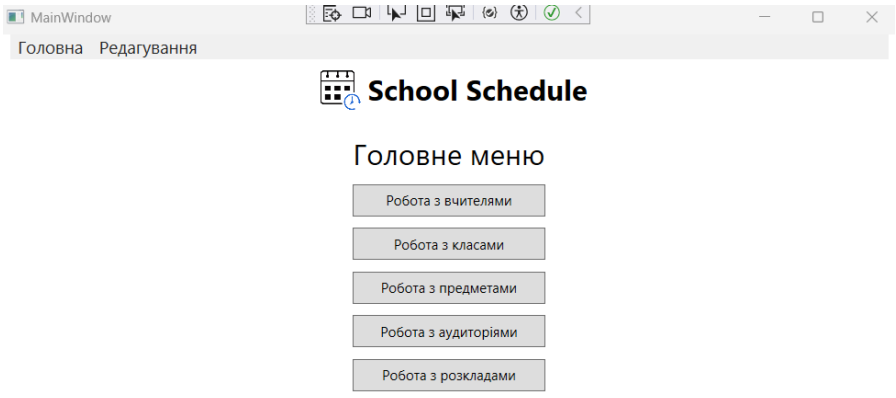

Рисунок 3.32 – Головне вікно програми

Меню у головному вікні програми складається з наступних кнопок:

– кнопка «Робота з вчителями»;

– кнопка «Робота з класами»;

– кнопка «Робота з предметами»;

– кнопка «Робота з аудиторіями»;

– кнопка «Робота з розкладами».

Окрім використання кнопок головного меню, користувач може перемикатися між формами за допомогою кнопок на верхній панелі вікна застосунку. Кнопка «Головна» повертає користувача до головного меню програми. Якщо користувач натисне на кнопку «Редагувати», то побачить список з пунктами, які збігаються з кнопками головного меню програми.

Кнопка «Робота з вчителями» спрямовує користувача до сторінки редагування даних про вчителів. Форма редагування даних вчителів наведена на рисунку 3.33.

Форма для роботи з вчителями складається зі списку вчителів у лівій частині екрану та списку предметів, які викладає обраний вчитель, у правій частині екрана. Наприкінці списку вчителів можна побачити кнопки «Додати», «Редагувати», «Видалити».

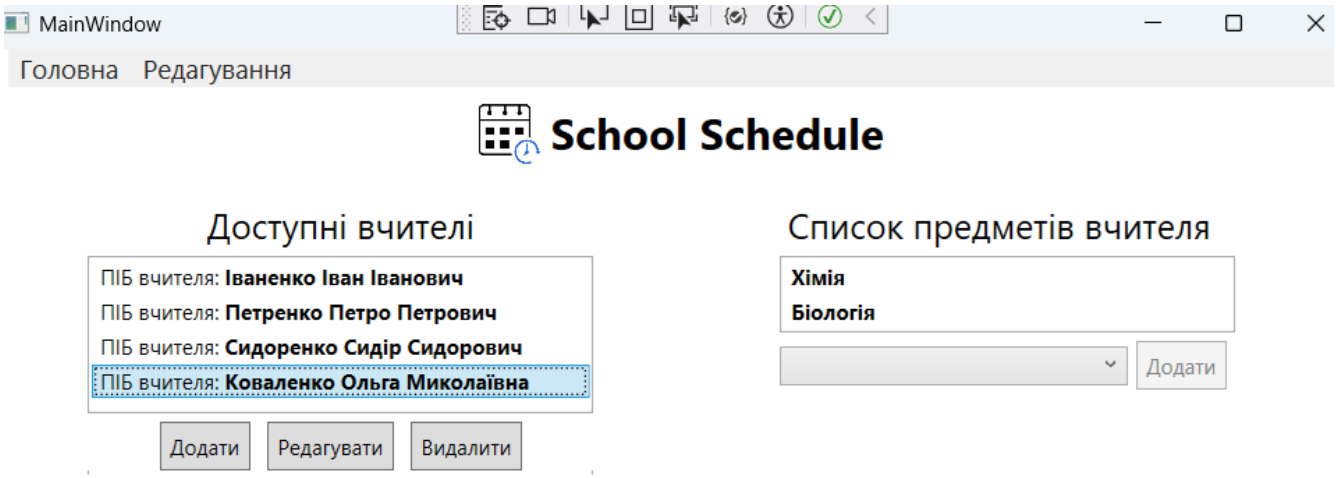

Рисунок 3.33 – Форма редагування даних про вчителів

Після натискання кнопки «Додати» користувача спрямовує до форми додавання вчителя, яка являє собою текстове поле, куди користувач вносить ПІБ вчителя. Окрім того, на формі міститься список, що випадає із даними про номер аудиторії вчителя, а також кнопки «Зберегти» та «Назад». Якщо користувач натискає кнопку «Зберегти», то система додає запис про вчителя до відповідної таблиці БД. Кнопка «Назад» повертає користувача до форми, яка наведена на рисунку 3.33.

Після натискання кнопки «Редагувати» користувача спрямовує до форми редагування вчителя, яка являє собою текстове поле з поточним повним ім’ям вчителя, куди користувач вносить оновлений ПІБ вчителя. Окрім того, на формі міститься список, що випадає із даними про номер аудиторії вчителя, а також кнопки «Зберегти» та «Назад». Якщо користувач натискає кнопку «Зберегти», то система оновлює запис у відповідній таблиці БД. Кнопка «Назад» повертає користувача до форми, яка наведена на рисунку 3.33.

Після натискання кнопки «Видалити» система проводить спробу видалити запис про вчителя з БД. Якщо в ході виконання цієї операції виникає помилка, то користувач побачить спливаюче вікно з відповідною помилкою (наприклад, повідомлення про те, що обраний вчитель використовується в одному з розкладів, ще має призначені предмети тощо). Якщо під час виконання операції видалення вчителя помилок не виникло, то запис про обраного вчителя буде видалено, а список вчителів оновлено.

Коли користувач обирає вчителя зі списку доступних вчителів, список із предметами, які викладає вчитель, оновлюється відповідно до обраного вчителя. Наприкінці списку предметів, які викладає вчитель, присутній список, що випадає, та кнопка «Додати». Кнопка «Додати» є не активною, поки користувач не обере предмети зі списку, що випадає. Після того, як користувач обрав предмет, кнопка змінює свій статус на активний. Якщо користувач натисне на кнопку «Додати», яка знаходиться наприкінці списку предметів, то предмет закріпиться за вчителем і користувач побачить новий предмет у списку.

Натискання на кнопку головного меню «Робота з класами» спрямовує користувача до сторінки редагування даних про класи. Форма редагування даних класів наведена на рисунку 3.34.

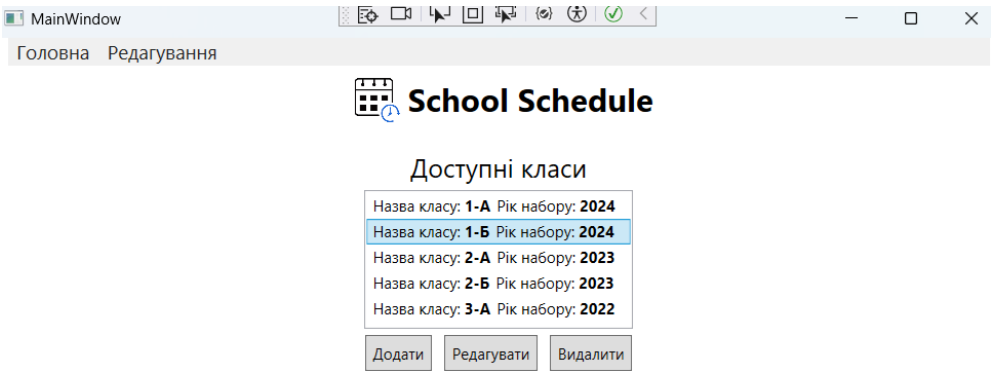

Рисунок 3.34 – Форма редагування даних про класи

Форма для роботи з класами складається зі списку класів. Наприкінці списку класів можна побачити кнопки «Додати», «Редагувати», «Видалити».

Після натискання кнопки «Додати» користувача спрямовує до форми додавання класу, яка являє собою текстове поле, куди користувач вносить назву класу, текстове поле з роком набору класу, кнопки «Зберегти» та «Назад». Якщо користувач натискає кнопку «Зберегти», то система додає запис про клас до відповідної таблиці БД. Кнопка «Назад» повертає користувача до форми, яка наведена на рисунку 3.34.

Після натискання кнопки «Редагувати» користувача спрямовує до форми редагування класу, яка являє собою текстове поле з поточною назвою класу, текстове поле з поточним роком набору класу, куди користувач вносить оновлені дані, кнопки «Зберегти» та «Назад». Якщо користувач натискає кнопку «Зберегти», то система оновлює запис про клас у відповідній таблиці БД. Кнопка «Назад» повертає користувача до форми, яка наведена на рисунку 3.34.

Після натискання кнопки «Видалити» система проводить спробу видалити запис про клас з БД. Якщо в ході виконання цієї операції виникає помилка, то користувач побачить спливаюче вікно з відповідним текстом (наприклад, повідомлення про те, що обраний клас використовується в одному з розкладів).

Якщо при виконанні операції видалення класу помилок не виникло, то запис про обраний клас буде видалено, а список класів оновлено.

Натискання на кнопку головного меню «Робота з предметами» спрямовує користувача до сторінки редагування даних про предмети. Форма для роботи з предметами складається зі списку предметів. Наприкінці списку можна побачити кнопки «Додати», «Редагувати», «Видалити». Форма для роботи з предметами наведена на рисунку 3.35.

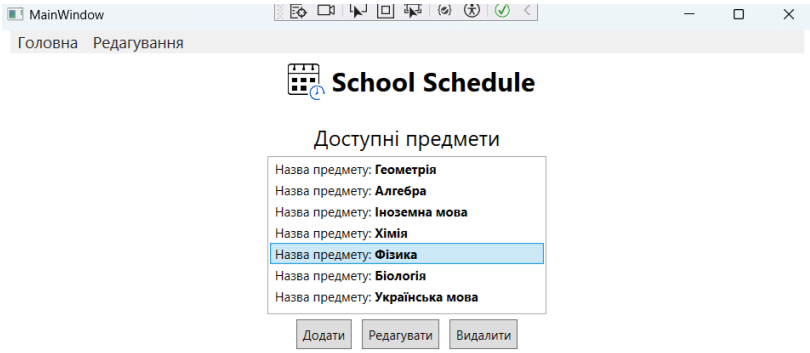

Рисунок 3.35 – Форма редагування даних про предмети

Після натискання кнопки «Додати» користувача спрямовує до форми додавання предмета, яка являє собою текстове поле, куди користувач вносить назву предмета, кнопки «Зберегти» та «Назад». Якщо користувач натискає кнопку «Зберегти», то система додає запис про предмет до відповідної таблиці БД. Кнопка «Назад» повертає користувача до форми редагування даних про предмети.

Після натискання кнопки «Редагувати» користувача спрямовує до форми редагування предмета, яка являє собою текстове поле з поточною назвою предмета, куди користувач вносить оновлені дані, кнопки «Зберегти» та «Назад». Якщо користувач натискає кнопку «Зберегти», то система оновлює запис про предмет у відповідній таблиці БД. Кнопка «Назад» повертає користувача до форми редагування даних про предмети зі списком предметів.

Після натискання кнопки «Видалити» система проводить спробу видалити запис про предмет з БД. Якщо в ході виконання цієї операції виникає помилка,

то користувач побачить спливаюче вікно з відповідним текстом (наприклад, повідомлення про те, що обраний предмет призначено відповідному класу). Якщо під час виконання операції видалення предмета помилок не виникло, то запис про обраний предмет буде видалено, а список предметів оновлено.

Натискання на кнопку головного меню «Робота з аудиторіями» спрямовує користувача до сторінки редагування даних про аудиторії. Форма для роботи з аудиторіями складається зі списку доступних аудиторій. Наприкінці списку можна побачити кнопки «Додати», «Редагувати», «Видалити». Форма для роботи з аудиторіями наведена на рисунку 3.36.

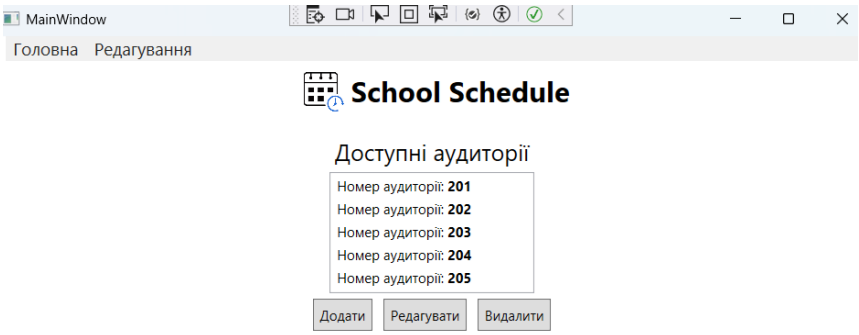

Рисунок 3.36 – Форма редагування даних про аудиторії

Після натискання кнопки «Додати» користувача спрямовує до форми додавання аудиторії, яка являє собою текстове поле, куди користувач вносить номер аудиторії, кнопки «Зберегти» та «Назад». Якщо користувач натискає кнопку «Зберегти», то система додає запис про аудиторію до відповідної таблиці БД. Кнопка «Назад» повертає користувача до форми редагування аудиторій, яка наведена на рисунку 3.36.

Після натискання кнопки «Редагувати» користувача спрямовує до форми редагування аудиторії, яка являє собою текстове поле з поточним номером аудиторії, куди користувач вносить оновлені дані, кнопки «Зберегти» та «Назад». Якщо користувач натискає кнопку «Зберегти», по система оновлює запис про аудиторію у відповідній таблиці БД. Кнопка «Назад» повертає

користувача до форми редагування даних про аудиторії, яка наведена на рисунку 3.36.

Після натискання кнопки «Видалити» система проводить спробу видалити запис про аудиторію з БД. Якщо в ході виконання цієї операції виникає помилка, то користувач побачить спливаюче вікно з відповідним текстом (наприклад, повідомлення про те, що обрану аудиторію закріплено за відповідним вчителем). Якщо під час виконання операції видалення аудиторії помилок не виникло, то запис буде видалено, а список аудиторій оновлено.

Натискання на кнопку головного меню «Робота з розкладами» спрямовує користувача до сторінки редагування даних про розклади класів. Форма складається зі списку розкладів класів, розклади для яких вже були створені. Наприкінці списку можна побачити кнопки «Додати», «Редагувати» та «Видалити». Також на формі присутня кнопка «Перейти до складання розкладу».

Після натискання кнопки «Додати» користувача спрямовує до форми додавання розкладу, яка являє собою список, що випадає, з набором класів, кнопки «Зберегти» та «Назад». Якщо користувач натискає кнопку «Зберегти», то система додає запис про розклад до відповідної таблиці БД. Кнопка «Назад» повертає користувача до форми редагування розкладів.

Після натискання кнопки «Редагувати» користувача спрямовує до форми редагування розкладу, яка являє собою список, що випадає з активним елементом поточного класу, кнопки «Зберегти» та «Назад». Якщо користувач натискає кнопку «Зберегти», по система оновлює запис про розклад у відповідній таблиці БД. Кнопка «Назад» повертає користувача до форми редагування даних про розклади.

Після натискання кнопки «Видалити» система проводить спробу видалити запис про розклад з БД. Якщо в ході виконання цієї операції виникає помилка, то користувач побачить спливаюче вікно з відповідним текстом (наприклад, повідомлення про те, що для розкладу наявні слоти). Якщо під час виконання операції видалення розкладу помилок не виникло, то запис буде видалено, а список розкладів оновлено.

Кнопка «Перейти до складання розкладу» спрямовує користувача до вікна з таблицею для заповнення слотів. У першому рядку користувач побачить назви класів і поруч із кожним класом кнопку «Назначені предмети».

Якщо користувач натисне на кнопку «Назначені предмети», то програма відкриє нове вікно. У новому вікні користувач може побачити предмети, які вже призначені класу. Також у новому вікні присутня форма призначення класу відповідного предмета. Користувач обирає потрібний предмет зі списку, що випадає. Нижче, у текстовому полі, користувач вводить кількість годин на тиждень для викладання обраного предмета. Для того, щоб призначити предмет, користувач натискає кнопку «Призначити предмет». Після цього обраний предмет додається до списку з відповідною кількістю годин.

Користувач має змогу видалити предмет. Він може обрати предмет зі списку та натиснути кнопку «Видалити». Після цього предмет видалиться зі списку призначених предметів, якщо під час проведення операції не виникло помилок. Вікно програми з можливістю призначення предметів для класу наведено на рисунку 3.37.

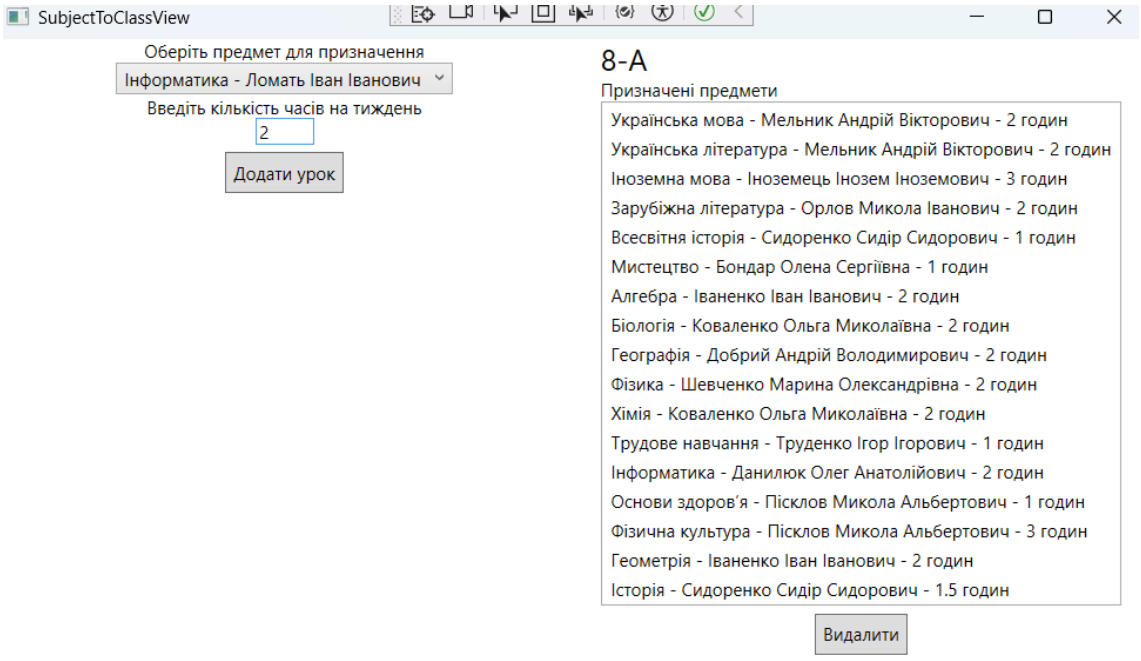

Рисунок 3.37 – Форма призначення предметів для класу

Також у вікні редагування слотів розкладу користувач має доступ до кнопки «Перегляд навантаження на вчителів». Ця кнопка відкриває нове вікно, де користувач може побачити список вчителів та їх тижневе навантаження.

Наприкінці таблиці зі слотами існує кнопка «Створити розклад», яка запускає алгоритм створення розкладу ЗНЗ. Приклад вікна з готовим розкладом наведено на рисунку 3.38.

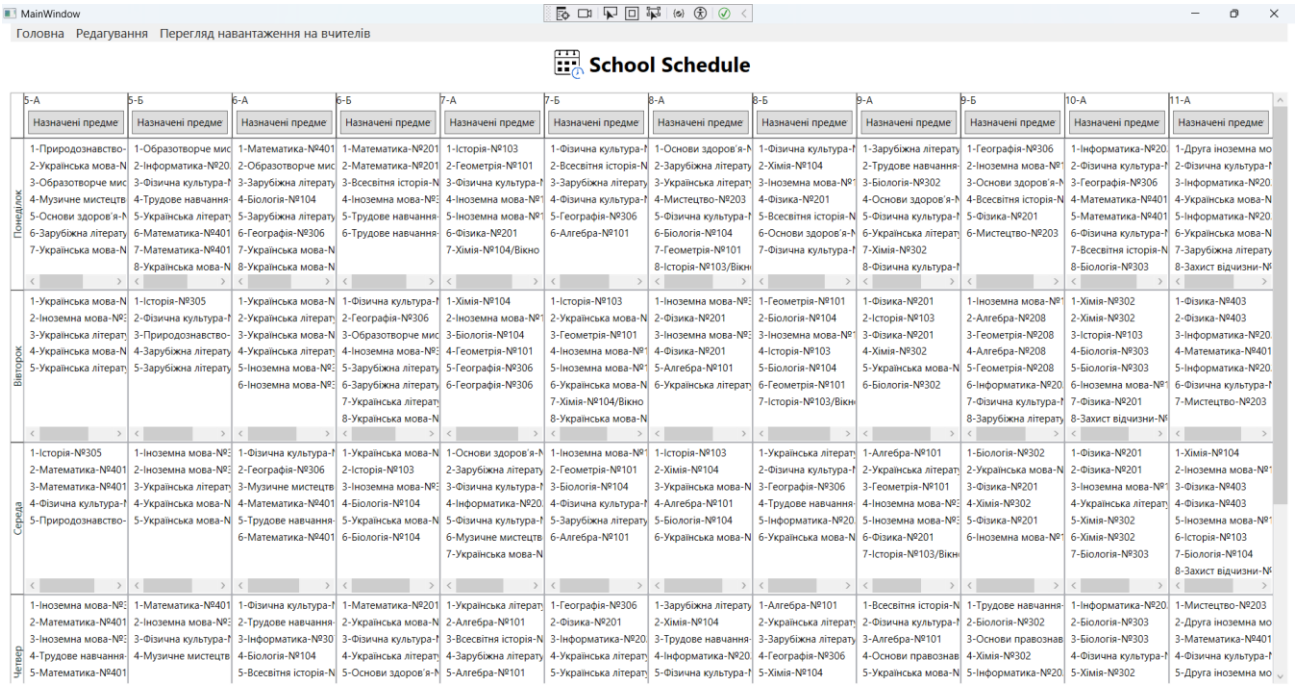

Рисунок 3.38 – Вікно редагування розкладу ЗНЗ

Поруч із кнопкою «Створити розклад» існує кнопка «Очистити». Кнопка «Очистити» видаляє записи з таблиці.

3.6 Висновки до третього розділу

У ході виконання третього розділу створено запити до БД, програмну реалізацію алгоритму, який було описано у другому розділі, та побудовано інтерфейс користувача для взаємодії з БД.

Також було проведено експеримент оптимізації розкладу, за результатами якого можна зробити висновок, що розроблений застосунком розклад занять ЗНЗ

відповідає вимогам цільової аудиторії, а саме: забезпечує відсутність вікон між заняттями та створює розклад з оптимальним навантаженням для більшості класів. Розподіл навантаження на учнів визначався за сумою коефіцієнтів складності предметів кожного з навчальних днів. Під час створення оптимального розкладу «найскладнішими» днями мають бути вівторок і середа. Можна покращити отримані результати шляхом більш рівномірного розподілу навантаження на вчителів (деякі з вчителів мали навантаження більш ніж півтори ставки).

4 ЗАХОДИ З ОХОРОНИ ПРАЦІ ПІД ЧАС РОЗРОБКИ ЗАСТОСУНКУ

Охорона праці є важливою складовою будь-якої професійної діяльності, зокрема й розробки програмного забезпечення. Цей розділ присвячено аналізу умов праці, можливих ризиків для здоров'я та безпеки розробника, а також заходам, спрямованим на мінімізацію цих ризиків. Дотримання норм охорони праці сприяє створенню комфортного та безпечного робочого середовища, що є запорукою продуктивної діяльності.

Робоче місце розробника програмного забезпечення є спеціалізованим середовищем, яке складається з комп’ютерної техніки, периферійного обладнання та меблів. Умови праці мають значний вплив на фізичне та психічне здоров’я співробітників. Особливої уваги потребує забезпечення ергономічності робочого місця, належного освітлення та організація режиму роботи. Також важливим є питання електробезпеки, пожежної безпеки та запобігання психоемоційному вигоранню.

Основним елементом робочого середовища є комп’ютер. Постійна робота за монітором може спричинити напруження зору, головний біль і втому. Для запобігання цим негативним наслідкам важливо забезпечити правильне розташування монітора – на рівні очей на відстані 50–70 см від користувача. Крім того, потрібно використовувати монітори з якісним покриттям для зменшення відблисків і налаштовувати яскравість та контрастність відповідно до умов освітлення.

Стілець розробника має бути ергономічним, з можливістю регулювання висоти, нахилу спинки та підтримки попереку. Стіл має бути таким, щоб руки користувача розташовувалися під прямим кутом під час роботи з клавіатурою. Висота стільця та столу повинна забезпечувати зручну позу, щоб уникнути надмірного навантаження на хребет, шию та суглоби. Також необхідно

забезпечити достатній простір для ніг, щоб уникнути застою крові в нижніх кінцівках.

Освітлення на робочому місці має відповідати санітарним нормам. Природне освітлення бажане, але його необхідно комбінувати з штучним. Для цього використовуються лампи денного світла, які забезпечують рівномірне освітлення без мерехтінь. Робочий стіл має бути розташований таким чином, щоб уникнути прямих сонячних променів або відблисків на моніторі. Важливим є також використання джерел додаткового освітлення, наприклад, настільних ламп, які забезпечують локальне підсвічування без тіні.

Одним з основних ризиків для здоров’я є тривале перебування у статичній позі. Це може призвести до болю в спині, шиї та суглобах. Для запобігання цьому рекомендовано робити перерви кожні 50–60 хв, під час яких виконувати прості фізичні вправи: розтяжку для спини, шиї та рук, а також короткі прогулянки. Крім того, потрібно виконувати вправи для очей, наприклад, методику «20-20-20»: кожні 20 хв роботи дивитися 20 с на об’єкт, розташований на відстані приблизно 6 м.

Режим роботи має бути організований таким чином, щоб уникнути надмірного навантаження. Робочий день повинен включати достатню кількість перерв для відпочинку та харчування. Крім того, важливо забезпечити можливістю гнучкого графіка, що дозволяє регулювати робочий час залежно від їхньої продуктивності та особистих обставин. Такі заходи сприяють зниженню стресу та підвищенню загальної ефективності праці.

Також важливим питанням є дотримання високого рівня електробезпеки. Усі електроприлади на робочому місці мають бути сертифікованими та справними. Робочі місця повинні бути обладнані заземленням, щоб уникнути ураження електричним струмом. Кожен співробітник має бути ознайомлений із правилами користування електрообладнанням і діями у разі аварійної ситуації. Регулярна перевірка стану електромережі та обладнання є обов’язковою для забезпечення безпеки.

Окрему увагу слід приділити психоемоційному стану. Велика кількість задач, дедлайни та складність проєктів можуть спричинити стрес і вигорання. Для запобігання цьому важливо створити комфортну атмосферу, організувати зони для відпочинку тощо. Також корисно використовувати техніки тайм-менеджменту для оптимізації робочого процесу [32].

Таким чином, правильна організація робочого місця, режиму роботи, електробезпека, а також заходи з підтримання психоемоційного здоров’я сприяють зниженню ризиків для здоров’я та підвищенню продуктивності праці.

ВИСНОВКИ

У рамках кваліфікаційної роботи обрано алгоритм оптимізації розкладу для загальноосвітнього навчального закладу, а також здійснено процес проєктування та розробки інформаційної системи.

Під час виконання кваліфікаційної роботи були виконані наступні кроки: проаналізована предметна область, проаналізовані вже реалізовані системи, їх основні переваги та недоліки, сформульована постановка задачі, на основі аналізу проведено обґрунтування вибору алгоритму оптимізації розкладу, були розглянуті особливості складання розкладу ЗНЗ, проведене логічне та фізичне моделювання БД за стандартом IDEF1X, UML-моделювання системи, розроблена математична модель та алгоритм роботи програми. Також проведено обґрунтування вибору архітектури (архітектура MVVM дозволяє забезпечити ефективну розробку та підтримку інформаційної системи), платформи для застосунку та СУБД. Як СУБД виступала SQLite, як платформу застосунку обрано платформу WPF, а для доступу до даних було використано пакет Dapper.

У рамках кваліфікаційної роботи було реалізовано адміністративні функції роботи з базою даних: додавання, оновлення та видалення записів, а також функціонал оптимізації розкладу. Реалізовано користувацький інтерфейс у вигляді XAML-форм.

За результатами проведення експерименту оптимізації розкладу загальноосвітнього навчального розкладу можна зробити наступні висновки: розроблений програмний засіб успішно складає розклад без наявності вікон і забезпечує рівномірне навантаження протягом тижня. Але частина програми, яка стосується розподілу складних предметів, може бути вдосконалена, хоча абсолютна більшість розкладів класів відповідає цьому критерію, шляхом оптимізації алгоритму під велике навантаження вчителів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. ДСТУ 3008-15. Звіти у сфері науки і техніки. Структура та правила оформлення. – Офіц. вид. – Київ : Держстандарт України, 2015. – 31 с. – (Державний стандарт України).

2. Методичні вказівки з підготовки та захисту кваліфікаційної роботи здобувачами другого (магістерського) рівня вищої освіти спеціальності 174 Автоматизація, комп'ютерно-інтегровані технології та робототехніка, освітньо-професійних програм: «Комп'ютерно-інтегровані технологічні процеси і виробництва», «Комп'ютеризовані та робототехнічні системи» / Упоряд. І. Ш. Невлюдов, Р. В. Артюх, В. В. Безкоровайний, Н. П. Демська, В. В. Євсєєв, О. І. Филипенко, О. М. Цимбал. Харків: ХНУРЕ, 2023. 49 с.

3. Пісклов М. А. Алгоритми створення та оптимізації розкладу для загальноосвітніх навчальних закладів. Автоматизація та приладобудування («Automation and Development of Electronic Devices» ADED-2023) [Електронний ресурс]: збірник студентських наукових статей: ХНУРЕ, 2023. – Вип. 2. – С. 275-280.

4. Моделі та методи кіберфізичних виробничих систем в концепції Industry 4.0 : монографія / І. Ш. Невлюдов, В. В. Євсєєв, А. О. Андрусевич, С. С. Максимова. – Oktan Print Prague. 2023. – 321 с.

5. Viktoriia Bortnikova, Vladyslav Yevsieiev, Iryna Botsman, Igor Nevliudov, Kostiantyn Kolesnyk, Nazariy Jaworski. Queries classification using machine learning for implementation in intelligent manufacturing // Chapter 6 in Monograph «Methods and tools in CAD – selected issues». – Białystok (Poland): Publishing House of Bialystok University of Technology. – 2021. – PP. 63-74.

6. V. Bortnikova, I. Nevliudov, I. Botsman and O. Chala, "Search Query Classification Using Machine Learning for Information Retrieval Systems in Intelligent Manufacturing," in CEUR Workshop Proceedings of the 15th International Conference on ICT in Education, Research, and Industrial Applications:

Integration, Harmonization, and Knowledge Transfer (ICTERI’2019), June 12-15, 2019, Kherson, Ukraine.

7. Жарикова И. В. Системологический подход при исследовании параметров РЭС / И. В. Жарикова, В. В. Невлюдова // Технология приборостроения. – 2014. – No2. – С. 40-43.

8. Нормативні документи про освіту, що регламентують діяльність загальноосвітнього навчального закладу. UkrSchool : веб-сайт. URL: https://mstyshyn.ukr.school/vedennya-dokumentatsiyi/normatyvni-dokumenty-pro-osvitu-shho-reglamentuyut-diyalnist-zagalnoosvitnogo-navchalnogo-zakladu/ (дата звернення: 05.09.2024).

9. Сторінка завантаження FET: вебсайт. URL: https://lalescu.ro/liviu/fet/ (дата звернення 05.09.2024).

10. EduPage. Комунікація школи та батьків: вебсайт. URL: https://vs-pfarrplatz-baden.edupage.org/text/?eqa=dGV4dD10ZXh0L3RleHQyNyZzdWJwYWdlP TMmc2tnZHllYXI9MjAyNA%3D%3D (дата звернення 05.09.2024).

11. Офіційний сайт Skolaris: вебсайт. URL: https://skolaris.net/ (дата звернення 05.09.2024).

12. Задача виконання обмежень: вебсайт. URL: https://uk.wikipedia.org/wiki/%D0%97%D0%B0%D0%B4%D0%B0%D1%87%D0%B0_%D0%B2%D0%B8%D0%BA%D0%BE%D0%BD%D0%B0%D0%BD%D0%BD%D1%8F_%D0%BE%D0%B1%D0%BC%D0%B5%D0%B6%D0%B5%D0%BD%D1%8C (дата звернення 20.10.2024).

13. Сайт FoxMinded. Стаття «Імперативне та декларативне програмування і чим вони відрізняються»: вебсайт. URL: https://foxminded.ua/imperatyvne-ta-deklaratyvne-prohramuvannia/ (дата звернення 21.10.2024).

14. Michael Marte. Models and Algorithms for School Timetabling – A Constraint-Programming Approach, 2002.

15. І. Ю. Юрчак Застосування генетичних алгоритмів в автоматизованій системі розподілу навчального навантаження / І. Ю. Юрчак, Т. Р. Москович // Комп’ютерні системи та мережі. – 2018. – Львів, 2018. – С. 149-157.

16. F. Rossi, P. van Beek and T. Walsh. Handbook of Constraint Programming, 2006.

17. High School Class Scheduling Problem: вебсайт. URL: https://levelup.gitconnected.com/high-school-class-scheduling-problem-70d0077ae2ac (дата звернення 21.10.2024).

18. Constraint programming approach for school timetabling: вебсайт. URL: https://www.researchgate.net/publication/222891149_Constraint_programming_approach_for_school_timetabling (дата звернення 22.10.2024).

19. Effective local search algorithms for high school timetabling problems. вебсайт URL: https://www.sciencedirect.com/science/article/abs/pii/S1568494617303903 (дата звернення 29.10.2024).

20. Офіційний сайт Верховної Ради України: вебсайт. URL: https://zakon.rada.gov.ua/laws/show/z1111-20#Text (дата звернення 29.10.2024).

21. Ресурс Studfile.net: вебсайт. URL: https://studfile.net/preview/5110357/page:54/ (дата звернення 29.10.2024).

22. Про внесення змін до типової освітньої програми для 5-9 класів закладів загальної середньої освіти [Електронний ресурс]: Наказ Міністерства освіти і науки України 19.02.2021 №235 – Режи доступу: URL: https://mon.gov.ua/static-objects/mon/sites/1/zagalna%20serednya/programy-5-9klas/2024/09.08.2024/typova-osvitnya-prohrama-dlya-5-9-klasiv-zzso-1120-vid-09082024.pdf.

23. Про внесення змін до типової освітньої програми для 10-11 класів закладів загальної середньої освіти [Електронний ресурс]: Наказ Міністерства освіти і науки України 20.04.2018 №408 – Режи доступу: URL: https://mon.gov.ua/static-objects/mon/sites/1/zagalna%20serednya/typovi programu-2-11/Typova.osv.prohr.ZZSO-III.stupenya.pdf.

24. Офіційний сайт IDEF: вебсайт. URL: https://www.idef.com/idef1x-data-modeling-method/ (дата звернення 29.10.2024).

25. Нотація IDEF1-X: вебсайт. URL: https://www.essentialstrategies.com/publications/modeling/idef1x.htm (дата звернення 29.10.2024).

26. Unified Modeling Language. Version 2.5.1 – published 05–12–2017. – United States: UML, 2017. – 796 с.

27. ICO 5807-85. Information processing – Documentation symbols and conventions for data, program and system flowcharts, program network charts and system resources charts – published 15–02–1985. – Switzerland: ISO, 1985. – 12 с.

28. Pro WPF 4.5 in C#Windows Presentation Foundation in .NET 4.5 / авт. кол. та упоряд. : Меттью Мак Дональд. Нью-Йорк, 2012. 1078 с.

29. Документація WPF: вебсайт. URL: https://learn. microsoft.com/en-us/dotnet/desktop/wpf/ (дата звернення 09.10.2024).

30. Офіційний сайт SQLite: вебсайт. URL: https://www.sqlite.org /index.html (дата звернення 10.10.2024).

31. Ресурс з вивчення ORM Dapper: вебсайт. URL: https://www .learndapper.com/ (дата звернення 10.10.2024).

32. Охорона праці в IT-компаніях: вебсайт. URL: https://pro-op.com.ua/article/17850-okhorona-pratsi-v-it-kompaniyakh (дата звернення 12.10.2024).