

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіoeлектроніки,
Факультет комп'ютерних наук і технологій
(повне найменування інституту, назва факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавр

(ступінь вищої освіти)

на тему ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ЗБОРУ ДАНИХ ДЛЯ
ФОРМУВАННЯ РОЗКЛАДУ ЗАКЛАДУ ВИЩОЇ ОСВІТИ. DATA
COLLECTION SOFTWARE FOR TIMETABLING AT A HIGHER EDUCATION
INSTITUTION

Виконав: студент(ка) 4 курсу, групи КНТ-117
Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)
Інженерія програмного забезпечення

Марков О.С.

(прізвище та ініціали)

Керівник Каплієнко Т.І.

(прізвище та ініціали)

Рецензент Шитікова О.В.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
 (повне найменування закладу вищої освіти)

Інститут, факультет ІРЕ, ФКНТ
 Кафедра програмних засобів
 Ступінь вищої освіти бакалавр
 Спеціальність 121 Інженерія програмного забезпечення
 (код і найменування)
 Освітня програма (спеціалізація) Інженерія програмного забезпечення
 (назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
 “ ” 20__ року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Маркова Олександра Сергійовича
 (прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Програмне забезпечення для збору даних для формування розкладу закладу вищої освіти. Data Collection Software for Timetabling at a Higher Education Institution
- керівник проєкту (роботи) Каплієнко Тетяна Ігорівна, к.т.н., доцент,
 (прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
- затверджені наказом закладу вищої освіти від “30” березня 2021 року № 103
2. Строк подання студентом проєкту (роботи) 25 травня 2021 року
3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Розробка архітектури програми. 3. Основні рішення щодо реалізації компонентів системи. 4. Керівництво програміста. 5. Керівництво оператора. Висновки. Перелік джерел посилань. Додатки.
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	Каплієнко Т.І., доцент		
Нормоконтроль	Камінська Ж.К., асистент		

7. Дата видачі завдання 15 лютого 2021 р.**КАЛЕНДАРНИЙ ПЛАН**

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	2–3 тижні	Розділ 1
3	Розробка архітектури програми.	4 тиждень	Розділ 2
4	Розробка програми.	5–6 тижні	Розділ 3
5	Тестування та експериментальне дослідження програми.	7 тиждень	Розділ 4
6	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування	8 тиждень	Додатки
7	Захист роботи.	9 тиждень	

Студент (ка)

(підпис)

Марков О.С.

(прізвище та ініціали)

Керівник проєкту (роботи)

(підпис)

Каплієнко Т.І.

(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра:
110 с., 17 табл., 20 рис., 2 дод., 14 джерел.

ЗБІР ДАНИХ, КОМП'ЮТЕРНА ПРОГРАМА, ФОРМУВАННЯ РОЗКЛАДУ, MYSQL, C#, WINDOWS PRESENTATION FOUNDATION.

Об'єкт дослідження – програмні засоби для збору даних для формування розкладу закладу вищої освіти.

Предмет дослідження – комп'ютерні програми для збору даних для формування розкладу закладу вищої освіти.

Мета роботи – створення програмної системи для автоматизації та прискорення збору даних, необхідних для формування розкладу закладу вищої освіти.

Матеріали, методи та технічні засоби: структурне та об'єктно-орієнтоване програмування, мова програмування C#, система для побудови клієнтських Windows-застосунків WPF, інтегроване середовище розробки SharpDevelop 5.1, система управління базами даних MySQL, персональний комп'ютер з процесором AMD Athlon під управлінням операційної системи Microsoft Windows 10.

Результати. Створено програмний Windows-застосунок, який виконує збір даних, необхідних для формування розкладу закладу вищої освіти.

Висновки. Розроблено програмну систему, яка автоматизує та прискорює збір даних для формування розкладу закладу вищої освіти.

Галузь використання – збір даних для формування розкладів в закладах вищої освіти диспетчерами.

ABSTRACT

Explanatory note to the bachelor's thesis: 110 pages, 20 figures, 17 tables, 2 appendixes, 14 sources.

DATA COLLECTION, COMPUTER PROGRAM, TIMETABLING, MYSQL, C#, WINDOWS PRESENTATION FOUNDATION.

Object of study are software tools for data collection for timetabling at a higher education institution.

Subject of study are computer programs for data collection for timetabling at a higher education institution.

The purpose of the work is to create a software system for the automation and acceleration of data collection for timetabling at a higher education institution.

Materials, methods, and tools: structured and object-oriented programming, programming language C#, WPF client application building system, integrated development environments SharpDevelop, MySQL database management system, personal computer with an Intel Core 2 Duo processor under the control of the Microsoft Windows 10 operating system.

Results. A Windows software application has been created that collects the data necessary for timetabling at a higher education institution.

Conclusions. A software system has been developed that automates and accelerates data collection for time tabling at a higher education institution.

The field of use is a data collection for timetabling at higher education institutions by dispatchers.

ЗМІСТ

	С.
Перелік скорочень та умовних позначок	8
Вступ	9
1 Аналіз предметної області	10
1.1 Опис предметної області	10
1.2 Огляд існуючих методів вирішення завдання	13
1.3 Огляд існуючих аналогів та їх порівняльний аналіз	14
1.4 Постановка завдання до роботи	18
2 Розробка архітектури системи	19
2.1 Опис структури розроблюваної програми	19
2.2 Вибір інструментів розробки програмного забезпечення	19
2.3 Вибір СУБД	21
2.4 Проектування бази даних	21
2.5 Висновки з розділу	27
3 Основні рішення щодо реалізації компонентів системи	28
3.1 Основні рішення з реалізації класів	28
3.2 Основні рішення щодо використаних сервісів	30
3.3 Основні рішення щодо даних	30
3.4 Основні рішення щодо реалізації користувацького інтерфейсу	33
3.5 Висновки з розділу	34
4 Керівництво програміста	35
4.1 Призначення і умови застосування програми	35
4.2 Характеристики програми	35
4.3 Звернення до програми	36
4.4 Вхідні і вихідні дані	36
4.5 Повідомлення	36
4.6 Висновки з розділу	37
5 Керівництво оператора	38

5.1 Призначення і умови виконання програми.....	38
5.2 Формат вхідних даних	38
5.3 Виконання програми	43
5.4 Повідомлення оператору	46
5.5 Висновки з розділу	46
Висновки	47
Перелік джерел посилань	49
Додаток А Технічне завдання	50
Додаток Б Текст програми	55
Додаток В Слайди презентації	102

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

FET	– Free Timetabling Software;
IDE	– Integrated Development Environment;
VB	– Visual Basic;
WPF	– Windows Presentation Foundation;
XAML	– eXtensible Application Markup Language;
XML	– eXtensible Markup Language;
БД	– база даних;
ВД	– відомість доручень;
ЗВО	– заклад вищої освіти;
НУ	– національний університет;
ОС	– операційна система;
ПЗ	– програмне забезпечення;
ПІБ	– прізвище, ім'я, по батькові;
СУБД	– система управління базами даних;
ФР	– формування розкладу.

ВСТУП

Одним із основних документів, що визначають функціонування будь-якого навчального закладу, є розклад занять. Якість складеного розкладу впливає не тільки на засвоєння матеріалу учнями, а звідси і на якість підготовки фахівців, але і на ефективність використання матеріальних ресурсів навчального закладу [1].

Очевидно, що успішність процесу збору усіх потрібних даних впливає на якість вирішення задачі формування розкладу (ФР) заклад вищої освіти (ЗВО) в цілому. В наш час автоматизація цього процесу є доволі складним завданням. Це зумовлено великим об'ємом неоднорідних, часто оновлюваних даних, які надходять із різних структурних підрозділів ЗВО, таких як навчальний відділ, деканати, кафедри, відділ матеріально-технічного постачання тощо. Зазвичай усі ці дані охоплюють навчальне навантаження, відомості щодо факультетів, кафедр, викладачів, груп студентів, дисциплін, а також інші категорії, що залежать від конкретного ЗВО. Як правило, вони містяться у різних файлах різних форматів, що, звісно, гальмує процес їхнього «витягання» для подальшого внесення в базу даних.

На цей час у багатьох ЗВО застосовується програмне забезпечення (ПЗ), яке вирішує задачу збору необхідних даних та ФР комплексно, тобто, заздалегідь вважається, що вхідні дані будуть структуровані певним чином, наприклад, у вигляді таблиці або словника певного формату. Очевидно, що така позиція не враховує складнощів, які виникають при ручному упорядкуванні даних диспетчерами ЗВО та їхнє побажання мати програмний продукт, з простим інтерфейсом та не перенасичений за функціоналом, що вирішуватиме конкретне завдання – збір даних.

Розроблене ПЗ має автоматизувати процес збору даних, необхідних для ФР ЗВО, а саме – завантаження необхідної інформації з файлів до БД. В подальшому вона може бути використана як компонент іншого ПЗ, орієнтованого на ФР ЗВО з урахуванням особливостей вхідних даних.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис предметної області

Насамперед необхідно окреслити проблему, що розглядається в даній роботі – процес збору даних, необхідних для ФР ЗВО. В контексті ЗВО в даній роботі виступає НУ «Запорізька політехніка».

Розклад занять є складовою частиною організації учбового процесу, що регламентує трудовий ритм, впливає на творчу віддачу викладачів, тому його можна розглядати як фактор оптимізації використання обмежених ресурсів – викладацького складу та аудиторного фонду [2].

ФР починається із введення даних про учбове навантаження. Джерелами цих даних є відомості доручень (ВД), складені на кафедрах ЗВО. ВД, як правило, формуються у вигляді електронних таблиць за шаблоном, що визначається для кожного конкретного ЗВО. В них містяться записи даних, кожна з яких складається з:

- назви учбової дисципліни, що викладається на кафедрі;
- групи або переліку груп, яким буде викладатися ця дисципліна;
- виду занять (наприклад, семінару або лекції);
- загальної кількості годин за видом занять;
- форми контролю (наприклад, екзамену або заліку);
- ПІБ викладача або групи викладачів за даною дисципліною;
- пропозицій до складання розкладу занять (наприклад, номери тих аудиторій, де процес викладання дисципліни вважається найбільш продуктивним).

Варто зауважити, що останній пункт даного переліку не є обов'язковим, а також те, що в різних ЗВО можуть додаватися або змінюватись інші пункти.

Окрім цього, до складу необхідних даних входять відомості щодо аудиторного фонду ЗВО, які можна умовно розділити на існуючі типи аудиторій та, власне кажучи, самі аудиторії, які описуються наступними параметрами:

- назва аудиторії;
- номер корпусу;
- тип аудиторії;
- кількість місць;
- належність до підрозділу (кафедри);
- значення, що показує, чи використовується дана аудиторія в
учбовому процесі.

Також необхідно виділити переліки факультетів, кафедр та учбових груп ЗВО, вказавши взаємозв'язки між ними, а також перелік учбових дисциплін.

І наостанок добуваються дані з відомостей про викладацький склад. Їх найбільш важливими параметрами є:

- ПІБ викладача;
- належність до підрозділу (кафедри);
- стать;
- посада;
- науковий ступінь;
- значення, що показує, чи працює викладач в період дії розкладу;
- формалізовані побажання викладачів типу «Бажані дні», «Бажаний
час»;
- значення, що показує пріоритет викладача при ФР.

Кількість та якість вищеописаних даних має бути достатньою для їхнього використання при ФР ЗВО. Схему зі збору даних для ФР ЗВО представлено на рис. 1.1.

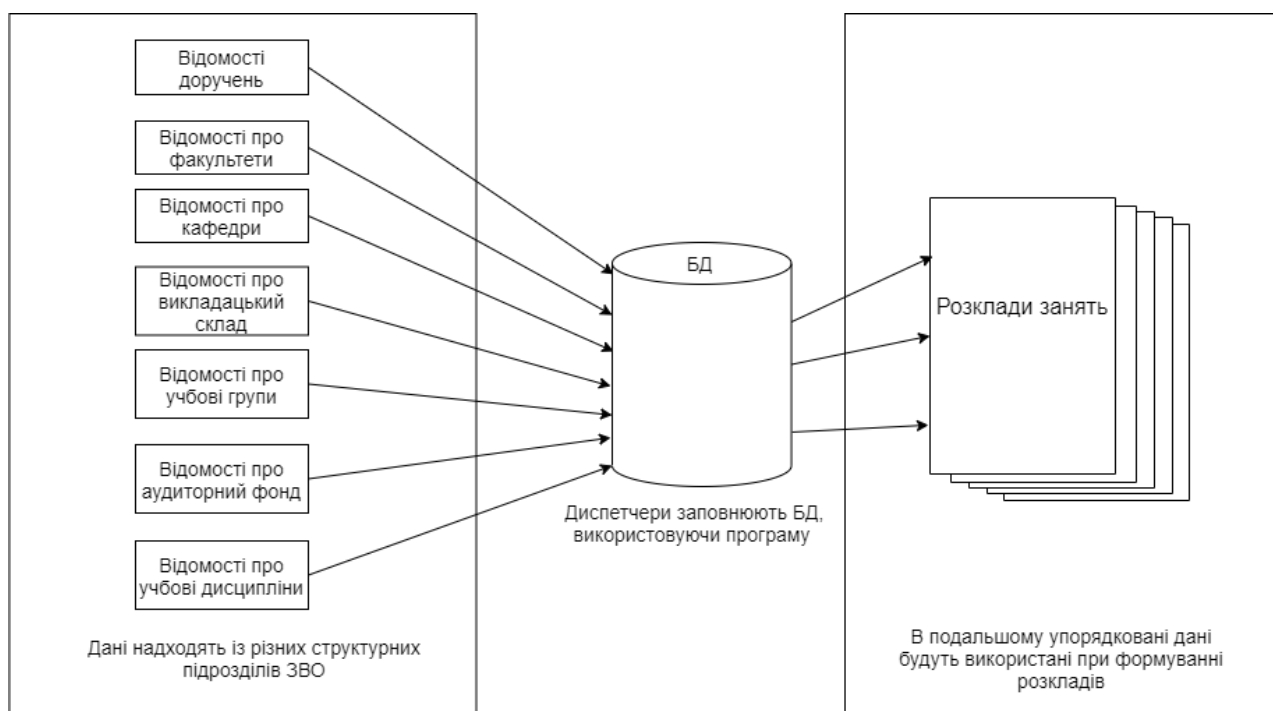


Рисунок 1.1 – Схема зі збору даних для ФР ЗВО

З точки зору кінцевого користувача – диспетчера ЗВО, більш прийнятний є такий алгоритм роботи (блок-схему наведено на рис. 1.2):

- перегляд файлів з даними, аби впевнитись, що саме їх потрібно буде завантажити до БД;
- вибір необхідного файлу для завантаження;
- завантаження до БД «в один клік»;
- перегляд такого собі «Звіту помилок», де описано, які саме неточності містили файли, що були джерелами конкретного завантаження.

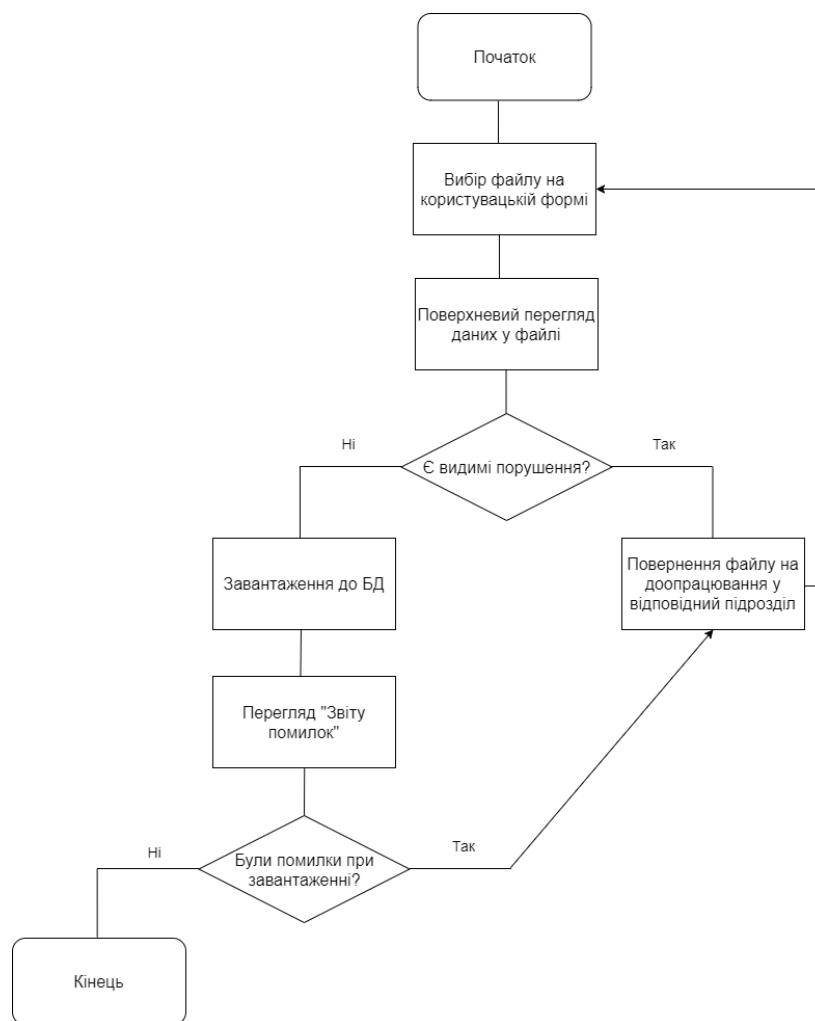


Рисунок 1.2 – Алгоритм роботи диспетчера при зборі даних

Останній крок алгоритму диспетчера є необов'язким, адже зазвичай очікується, що вхідні дані з електронних таблиць не мають неточностей, таких як дублікати, пропуски тощо. Проте наявність «Звіту помилок» допоможе диспетчерові оцінити, в яких саме файлах криються причини потенційних помилок, щоб повернути їх на доопрацювання.

1.2 Огляд існуючих методів вирішення завдання

Класичним способом вирішення проблеми збору даних, необхідних для ФР ЗВО, є їхнє ручне виписування з файлів на аркуші. В подальшому ці аркуші займають місця в таблицях розкладу у певних комбінаціях, визначених диспетчерами, що займаються ФР. Таким чином, процес ФР відбувається без

використання БД в сучасному розумінні цього визначення з усіма наслідками, що випливають з цього:

- перегляд такого собі «Звіту помилок», де описано, які саме неточності містили файли, що були джерелами конкретного завантаження.
- плутанина з аркушами, що сповільнює процес;
- необхідність врахування ситуацій дублювання даних або прогалин в них;
- виникнення потреби перепису аркушів вже в процесі роботи над ФР тощо.

На жаль, даний спосіб досі використовується багатьма диспетчерами в ЗВО України, зокрема і в НУ «Запорізька політехніка».

Іншим, більш привабливим способом вирішення проблеми є використання спеціалізованого ПЗ, що надає можливості ручного, автоматизованого або напівавтоматизованого введення даних з подальшим ФР. Очевидними перевагами цього способу є підвищення надійності зберігання даних, їхня структурованість та швидкість процесу збору, і, звісно, вбудований алгоритм ФР. Але і він має і ряд недоліків:

- комерційність ПЗ;
- необхідність зберігати усі дані у визначених шаблонах заданих форматів для кожного конкретного ПЗ;
- дещо завищені вимоги до кваліфікації диспетчерів тощо.

1.3 Огляд існуючих аналогів та їх порівняльний аналіз

1.3.1 Програма «FET»

Офіційний сайт програми: <https://lalescu.ro/liviu/fet/>

Дане ПЗ, форма додання активності якого представлена на рис. 1.3 [3], призначено для ФР уроків в середній школі, університеті тощо.

Рисунок 1.3 – Форма додання активності [3]

Його базовий функціонал:

- імпорт/експорт даних у форматі CSV (Comma-Separated Values);
- експорт розкладу в форматі HTML (HyperText Markup Language);
- автоматичний алгоритм ФР;
- можливості напіваавтоматичного або ручного ФР.

Переваги даного ПЗ:

- багатоплатформність;
- багатомовність;
- автоматичний алгоритм ФР;
- зручність та лаконічність інтерфейсу.

Його недоліки:

- орієнтація на застосування і в школах, і в ЗВО (не враховуються нюанси ЗВО);
- необхідність редагування вхідних даних вручну (у випадку, якщо користувача не влаштує інтерфейс);
- вимога щодо навичок роботи з редактором XML;

– можлива потреба в компіляції для отримання вихідної версії продукту.

1.3.2 Програма «1С: Автоматизоване складання розкладу. Університет»

Офіційний сайт програми: solutions.1c.ru/catalog/asp_univer/features

Дане ПЗ, вікно формування учбового розкладу якого представлено на рис. 1.4 [4], призначено для вирішення задач автоматизованого ФР і оперативного керування приміщеннями в ЗВО.

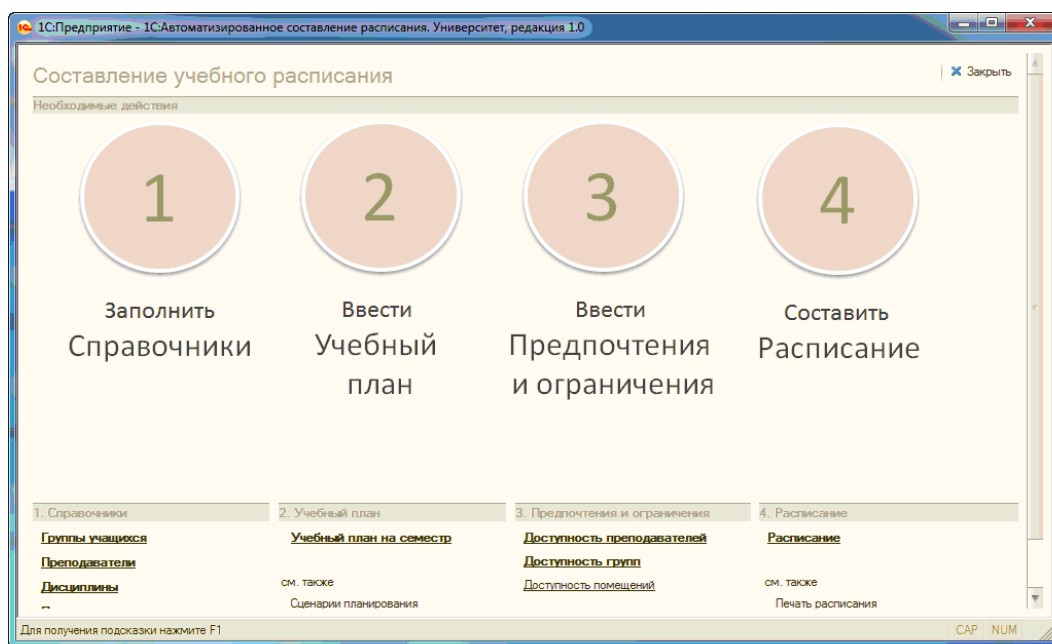


Рисунок 1.4 – Формування учбового розкладу [4]

Його базовий функціонал:

- ФР в ручному, автоматичному, змішаному режимах, а також в режимах «по приміщенням», «по групам», «по викладачам»;
- вибір найкращого розкладу із складених (можливості оптимізації з урахуванням місткості приміщень, мінімізації кількості «вікон» тощо);

– генерація звітів, можливості сповіщення щодо змін розкладу по e-mail для студентів та викладачів тощо.

Переваги даного ПЗ:

- всеохопний функціонал;
- врахування побажань та можливостей окремих категорій (викладачів, груп студентів, приміщень тощо);
- пристосованість до динамічного змінювання вхідних даних.

Його недоліки:

- вхідні дані повинні подаватися у формі довідників форматів Excel і XML, що, звісно, вимагає попередньої ручної обробки файлів;
- велике візуальне навантаження інтерфейсу;
- комерційність.

1.3.3 Порівняльний аналіз аналогів

Готові рішення, описані в попередньому розділі, було проаналізовано та порівняно з точки зору збору даних. Результати (табл. 1.1) засвідчують, що обидва рішення мають очевидні переваги та недоліки, а також важливість розробки власного ПЗ.

Таблиця 1.1 – Результати порівняння

Критерій порівняння	«FET»	«1С: Автоматизоване складання розкладу. Університет»
Різноманітність режимів введення	+	+
Оцінка можливих неточностей у вхідних даних	–	–
Завантаження до бази даних «в один клік»	–	–
Зручність інтерфейсу	+	+
Багатомовність	+	+
Безкоштовність	+	–

1.4 Постановка завдання до роботи

В результаті дослідження проблеми збору даних, необхідних для ФР ЗВО, було формалізовано завдання до роботи, яке полягає в створенні десктопного застосунку, якість якого визначається за критеріями, наведеними в попередньому підрозділі.

За результатами аналізу було сформовано технічне завдання, яке наведено в додатку А.

2 РОЗРОБКА АРХІТЕКТУРИ СИСТЕМИ

2.1 Опис структури розроблюваної програми

Структуру розроблюваного проєкту складають:

- глобальні файли, що відповідають за формування головного вікна програми, яке відкривається при запуску;
- файли розмітки;
- файли пов'язаного коду;
- файли з класами моделі програми.

Прийняті рішення щодо реалізації окремих компонентів програми наведено в наступному розділі.

2.2 Вибір інструментів розробки програмного забезпечення

Наступним кроком після аналізу предметної області є вибір необхідних інструментів та технологій, які буде задіяно на етапі розробки (кодування) – мови програмування, Integrated Development Environment (IDE), технології для побудови графічного інтерфейсу, системи управління базами даних (СУБД).

2.2.1 Вибір мови програмування

Мовою розробки програми було обрано C#. C# є основною мовою розробки програм на платформі .NET корпорації Microsoft. В ній вдало поєднуються випробувані засоби програмування з останніми нововведеннями і надається можливість для ефективного і дуже практичного написання програм, призначених для обчислювального середовища сучасних підприємств [5].

Оскільки C# – гібрид із декількох мов, він є таким же синтаксично чистим, як Java (якщо не чистішим), майже настільки ж простим як Visual Basic (VB), і практично таким же потужним та гнучким як C++ [6].

2.2.2 Вибір технології для побудови графічного інтерфейсу

В якості технології для побудови графічного інтерфейсу програми було обрано WPF. Якщо при створенні традиційних застосунків на основі WinForms за отрисовку елементів управління та графіки відповідали такі частини ОС Windows, як User32 (Windows User) і GDI+ (Graphics Device Interface), то застосунки WPF засновані на DirectX. В цьому полягає ключова особливість рендеринга графіки в WPF: використовуючи WPF, значна частина роботи щодо отрисовки графіки, як найпростіших кнопочок, так і складних 3D-моделей, лягає на графічний процесор на відеокарті, що також дозволяє використати апаратне прискорення графіки.

Однією із важливих особливостей є використання мови декларативної розмітки інтерфейсу XAML (eXtensible Application Markup Language), заснованої на XML: можна створювати насичений графічний інтерфейс, використовуючи або декларативне оголошення інтерфейсу, або код на керуючих мовах C# і VB .NET, або поєднувати і те, і те [7].

2.2.3 Вибір IDE

IDE було обрано SharpDevelop 5.1. Це вільне IDE для C#, VB. NET, Boo, IronPython, IronRuby, F#, C++. Використовується як альтернатива Microsoft Visual Studio [8]. Його основними особливостями, окрім підтримки вищевказаних мов програмування, є наявність візуального редактора WPF, інтегрованого налагоджувача, профайлера, підсвічування синтаксису та розширюваність на основі механізму додатків.

Дане середовище розробки доволі часто характеризується як «ідеальне для слабких комп'ютерів», що пояснюється поєднанням достатньо багатого функціоналу, невеликого розміру та високої швидкості завантаження.

2.3 Вибір СУБД

Для роботи з базами даних було обрано СУБД MySQL, яка є однією з найбільш поширених та підтримується багатьма мовами програмування.

MySQL – компактний багато потоковий сервер баз даних. Характеризується високою швидкістю, стійкістю і простотою використання. MySQL вважається гарним рішенням для малих і середніх застосувань. Можливості сервера MySQL:

- простота у встановленні та використанні;
- підтримується необмежена кількість користувачів, що одночасно працюють із БД;
- кількість рядків у таблицях може досягати 50 млн.;
- висока швидкість виконання команд;
- наявність простої і ефективної системи безпеки [9].

Адміністрування сервера MySQL здійснювалось шляхом використання інтерфейсу веб-застосунку phpMyAdmin, що дозволяє керувати БД без вводу SQL-команд [10].

2.4 Проєктування бази даних

До початку проєктування БД слід дати визначення таким важливим поняттям як «реляційна БД», «первинний ключ», «зв'язок між таблицями».

Реляційна БД – БД, яка базується на теорії, що називається реляційною моделлю даних. Інтуїтивно зрозуміло, що окрім всього іншого, в такій системі виконуються як мінімум три умови: структурний аспект, аспект цілісності та аспект обробки [11].

Первинний ключ – один або більше стовпців, які можна використовувати як унікальний ідентифікатор для кожного рядку таблиці [12].

Між двома таблицями існує зв'язок, якщо записам в першій таблиці певним чином співставлені записи іншої таблиці. Самі зв'язки можуть бути

спроектовані за одним із трьох типів: «один до одного», «один до багатьох» і «багато до багатьох» [13].

Концептуальні вимоги й інформаційні потреби до спроектованої БД, можна описати так: БД повинна забезпечувати підготовку, зберігання та обробку даних, необхідних для подальшого ФР ЗВО.

Інформаційне забезпечення БД складається з:

- даних про викладацький склад;
- даних про аудиторний фонд;
- даних про підрозділи ЗВО (факультети та кафедри);
- даних про учбові дисципліни;
- даних про групи студентів;
- динамічно змінювані дані з ВД.

В результаті аналізу предметної області було виділено наступні сутності: «Викладач» («Teacher»), «Факультет» («Faculty»), «Кафедра» («Department»), «Тип аудиторій» («Auditory_type»), «Аудиторія» («Auditory»), «Дисципліна» («Discipline»), «Учбова група» («Study_group»), «Заняття» («Lesson»), «Заняття групи» («Lesson_group»), «Аудиторія для заняття» («Lesson_auditory»), «Викладач, що веде заняття» («Lesson_teacher»), «Час заняття» («Lesson_time»).

Атрибути сутності «Викладач»:

- код викладача (первинний ключ);
- код кафедри;
- ПІБ;
- статъ;
- посада;
- науковий ступінь;
- код (працює/не працює);
- бажані дні роботи;
- бажаний час роботи;
- пріоритет при ФР.

Атрибути сутності «Факультет»:

- код факультета (первинний ключ);
- повне найменування;
- код (скорочена назва).

Атрибути сутності «Кафедра»:

- код кафедри (первинний ключ);
- код факультета;
- найменування;
- скорочена назва.

Атрибути сутності «Тип аудиторій»:

- код типу аудиторій (первинний ключ);
- найменування.

Атрибути сутності «Аудиторія»:

- код аудиторії (первинний ключ);
- код кафедри;
- найменування;
- код (використовується / не використовується);
- тип аудиторії;
- кількість місць;
- номер корпусу.

Атрибути сутності «Дисципліна»:

- код дисципліни (первинний ключ);
- найменування;
- скорочена назва.

Атрибути сутності «Учбова група»:

- код учбової групи (первинний ключ);
- код кафедри;
- найменування;
- код (скорочена назва);
- номер курсу;

- кількість студентів.

Атрибути сутності «Заняття»:

- код заняття (первинний ключ);
- код дисципліни;
- тип заняття;
- кількість годин;
- контроль;
- код кафедри.

Атрибути сутності «Заняття групи»:

- код заняття групи (первинний ключ);
- код заняття;
- код групи.

Атрибути сутності «Аудиторія для заняття»:

- код аудиторії для заняття (первинний ключ);
- код заняття;
- код аудиторії.

Атрибути сутності «Викладач, що веде заняття»:

- код викладача, що веде заняття (первинний ключ);
- код заняття;
- код викладача.

Атрибути сутності «Час заняття»:

- код часу заняття (первинний ключ);
- код заняття;
- номер заняття за розкладом;
- тип тижня;
- день тижня.

Опис інформації, що зберігається в сутностях наведено в табл. 2.1.

Таблиця 2.1 – Опис інформації, що зберігається в сутностях

Сутність	Опис
Викладач	Інформація про викладацький склад ЗВО
Факультет	Інформація про факультети ЗВО
Кафедра	Інформація про кафедри ЗВО
Тип аудиторій	Інформація про типи аудиторій в ЗВО
Аудиторія	Інформація про аудиторний фонд ЗВО
Дисципліна	Інформація про дисципліни
Учбова група	Інформація про учбові групи
Заняття	Інформація про заняття
Заняття групи	Ця сутність зв'язує інформацію сутностей «Учбові групи» та «Заняття»
Аудиторія для заняття	Ця сутність зв'язує інформацію сутностей «Аудиторія» та «Заняття»
Викладач, що веде заняття	Ця сутність зв'язує інформацію сутностей «Викладач» та «Заняття»
Час заняття	Інформація про час проведення заняття

В БД зв'язки між сутностями характеризуються відношенням «один до багатьох», а саме:

- «Faculties» – «Department» (до складу факультету входять декілька кафедр, але кожна кафедра закріплена за конкретним факультетом);
- «Auditory_type» – «Auditory» (багато аудиторій можуть мати той самий тип, але кожна аудиторія має конкретний тип);
- «Discipline» – «Lesson» (одна дисципліна викладається на багатьох заняттях, але на кожному занятті викладається конкретна дисципліна);
- «Department» – «Teacher» (за однією кафедрою закріплено багато викладачів, але кожен викладач закріплений за конкретною кафедрою);
- «Department» – «Auditory» (за однією кафедрою закріплено багато аудиторій, але кожна аудиторія закріплена за конкретною кафедрою);

- «Department» – «Study_group» (за однією кафедрою закріплено багато учбових груп, але кожна учбова група закріплена за конкретною кафедрою);
- «Department» – «Lesson» (однією кафедрою викладається багато занять, але кожне заняття викладається на конкретній кафедрі);
- «Lesson» – «Lesson_teacher» (одне заняття може проводитись багатьма викладачами, але кожен викладач проводить конкретне заняття);
- «Lesson» – «Lesson_group» (одне заняття може проводитись для багатьох груп, але кожна група проходить конкретне заняття);
- «Lesson» – «Lesson_time» (одне заняття може проводитись в різний час, але в кожний час занять ведеться конкретне заняття);
- «Lesson» – «Lesson_auditory» (одне заняття може проводитись в багатьох аудиторіях, але в кожній аудиторії ведеться конкретне заняття);
- «Teacher» – «Lesson_teacher» (один викладач може вести декілька занять, але кожне заняття ведеться конкретним викладачем);
- «Auditory» – «Lesson_auditory» (одна аудиторія може використовуватись для різних занять, але кожне заняття відбувається в конкретній аудиторії);
- «Study_group» – «Lesson_group» (одна учбова група може відвідувати багато занять, але кожне заняття проводиться для конкретної групи).

Схему реляційної БД, що відображає взаємозв'язки між таблицями та є основою для БД розроблюваної програми, представлено на рис. 2.1.

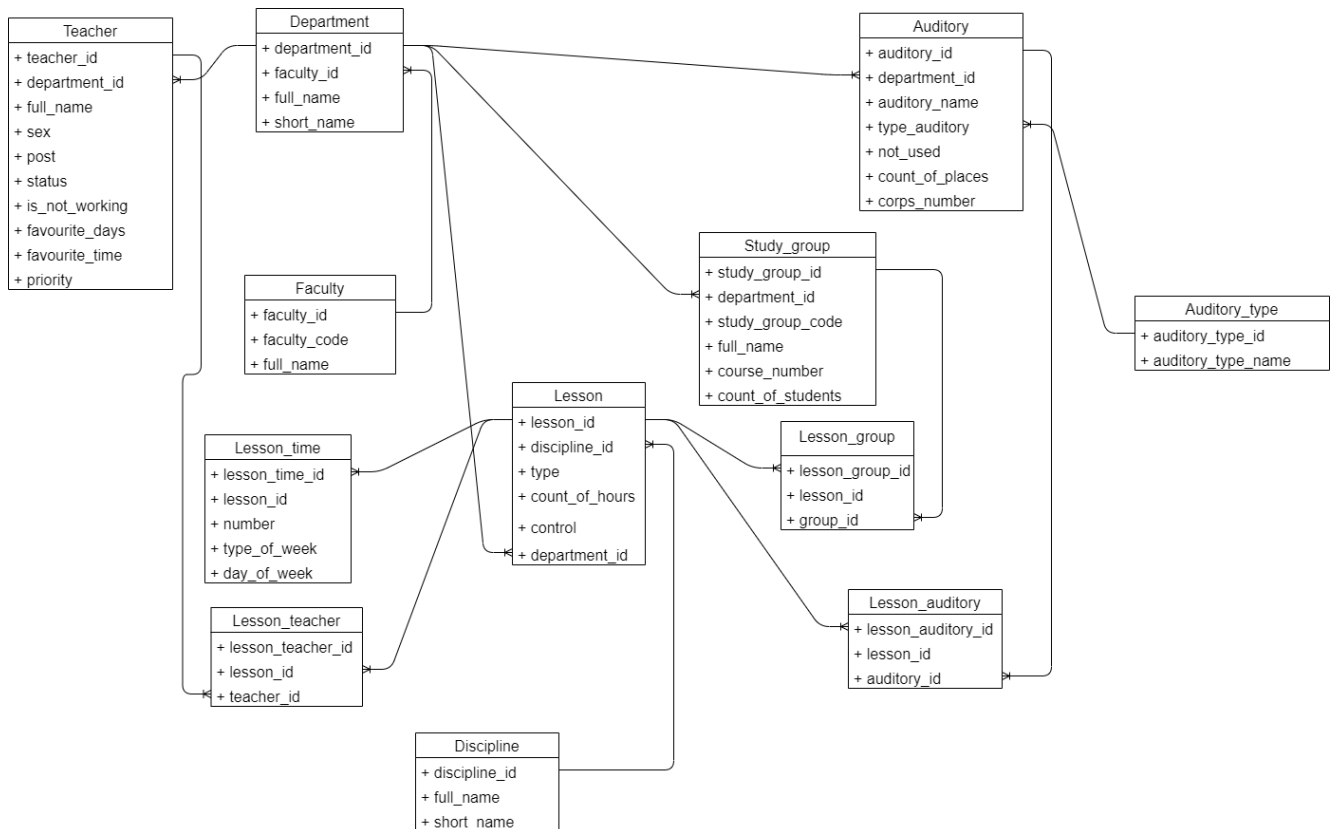


Рисунок 2.1 – Схема БД

2.5 Висновки з розділу

На даному етапі роботи було вибрано структуру системи, що проєктується, необхідні інструменти для розробки (мову програмування C#, IDE SharpDevelop, СУБД MySQL, технологію для побудови графічного інтерфейсу WPF), а також виконано проєктування майбутньої БД.

3 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

3.1 Основні рішення з реалізації класів

Алгоритм «витягання потрібних даних» з файлів та їхнє подальше завантаження до БД можна умовно розділити на 3 окремі кроки:

- зчитування даних з файлів;
- перевірка даних на наявність прогалин та дублювання;
- завантаження до БД.

Очевидно, що при роботі диспетчера з різними файлами, структура цього алгоритму буде загальною, але деталі кроків буде частково перевизначено. Наслідком цього стало рішення щодо застосування патерну поведінки «Шаблонний метод».

Шаблонний метод – це поведінковий патерн проєктування, який визначає кістяк алгоритму, перекладаючи відповідальність за деякі його кроки на підкласи. Патерн дозволяє підкласам перевизначати кроки алгоритму, не змінюючи його загальної структури [14].

На рис. 3.1 представлено реалізацію даного патерну: в класі `ExcelFile` визначено шаблонний метод `sendDataToDB()`, в якому містяться визначення віртуальних функцій `readFromExcelFile()`, `evaluateData()` і `load()`. Їх перевизначено в класах, що успадковуються від класу `ExcelFile`:

- `Auditories`;
- `AuditoryTypes`;
- `Departments`;
- `Faculties`;
- `Disciplines`;
- `StudyGroups`;
- `Teachers`;
- `Statements`.

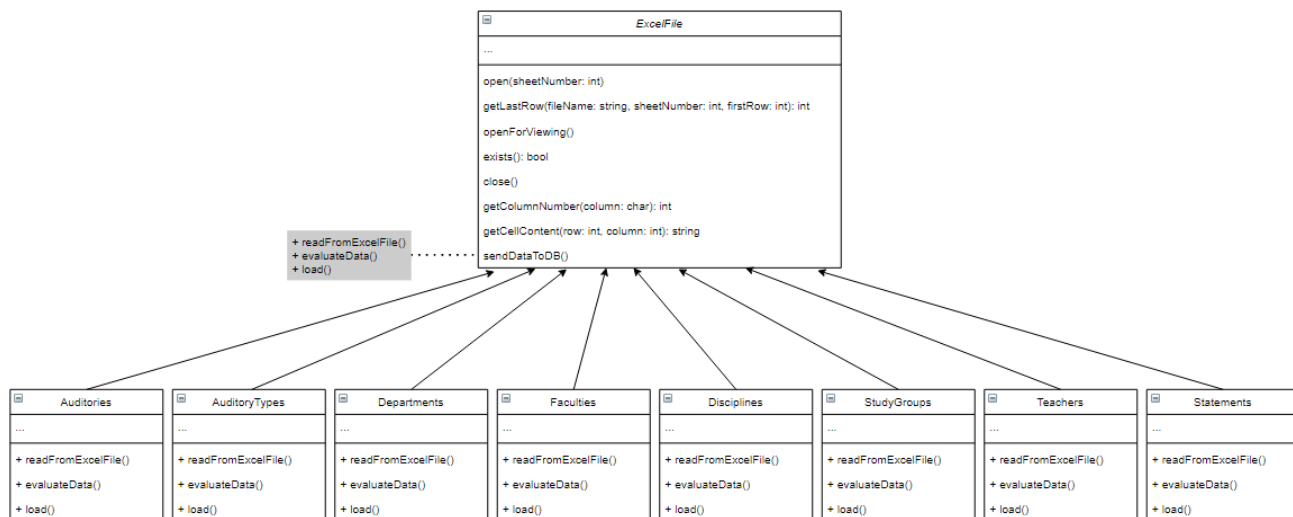


Рисунок 3.1 – Реалізація патерну «Шаблонний метод»

Частини функціоналу, що відповідають за з'єднання з БД, редагування рядків даних тощо винесено в окремі класи. Їх представлено на рис. 3.2.

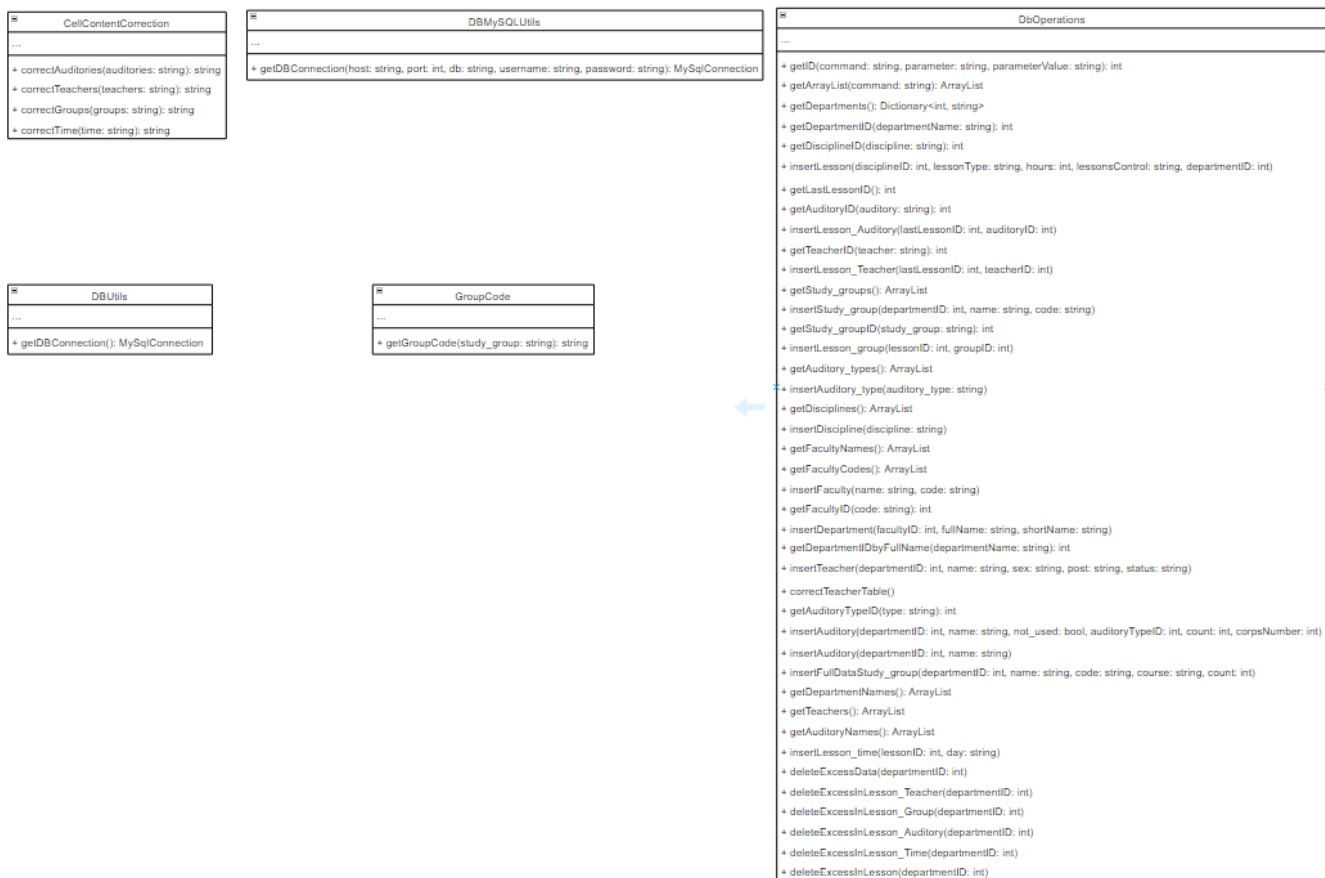


Рисунок 3.2 – Інші класи програми

3.2 Основні рішення щодо використаних сервісів

Для роботи з MySQL в проєкт було підключено посилання на сервіс MySql.Data, що дає можливість працювати з об'єктами типу MySqlConnection, MySqlCommand, MySqlDataReader тощо.

Оскільки вхідними даними є файли з розширенням .xlsx та .xls, для роботи з ними в проєкт було підключено посилання на сервіс Microsoft.Office.Interop.Excel, що дає можливість обробляти зміст файлів даного типу.

3.3 Основні рішення щодо даних

На основі спроектованої схеми даних, описаної у попередньому розділі, було розроблено структури таблиць БД, які наведено в табл. 3.1 – 3.12.

Таблиця 3.1 – Таблиця «Teacher»

Ім'я поля	Ключ	Значення за замовчуванням	Тип	Розмір
teacher_id	Первинний		int	11
department_id			int	11
full_name			varchar	75
sex			text	
post			varchar	50
status			varchar	50
is_not_working		0	tinyint	1
favourite_days		NULL	varchar	255
favourite_time		NULL	varchar	255
priority		NULL	int	11

Таблиця 3.2 – Таблиця «Faculty»

Ім'я поля	Ключ	Значення за замовчуванням	Тип	Розмір
faculty_id	Первинний		int	11
faculty_code			varchar	10
full_name			varchar	255

Таблиця 3.3 – Таблиця «Department»

Ім'я поля	Ключ	Значення за замовчуванням	Тип	Розмір
department_id	Первинний		int	11
faculty_id			int	11
full_name			varchar	255
short_name			varchar	50

Таблиця 3.4 – Таблиця «Auditory_type»

Ім'я поля	Ключ	Значення за замовчуванням	Тип	Розмір
auditory_type_id	Первинний		int	11
auditory_type_name			varchar	50

Таблиця 3.5 – Таблиця «Auditory»

Ім'я поля	Ключ	Значення за замовчуванням	Тип	Розмір
auditory_id	Первинний		int	11
department_id			int	11
auditory_name			varchar	50
not_used		0	tinyint	1
type_auditory		NULL	int	11
count_of_places		NULL	int	11
corps_number		NULL	tinyint	4

Таблиця 3.6 – Таблиця «Discipline»

Ім'я поля	Ключ	Значення за замовчуванням	Тип	Розмір
discipline_id	Первинний		int	11
full_name			varchar	255
short_name		NULL	varchar	10

Таблиця 3.7 – Таблиця «Study_group»

Ім'я поля	Ключ	Значення за замовчуванням	Тип	Розмір
study_group_id	Первинний		int	11
department_id			int	11
study_group_code		NULL	varchar	50
full_name			varchar	50
course_number		NULL	varchar	3
count_of_students		NULL	int	11

Таблиця 3.8 – Таблиця «Lesson»

Ім'я поля	Ключ	Значення за замовчуванням	Тип	Розмір
lesson_id	Первинний		int	11
discipline_id			int	11
type			varchar	25
countOfHours		NULL	float	
control		NULL	varchar	20
department_id			int	11

Таблиця 3.9 – Таблиця «Lesson_group»

Ім'я поля	Ключ	Значення за замовчуванням	Тип	Розмір
lesson_group_id	Первинний		int	11
lesson_group_id			int	11
group_id			int	11

Таблиця 3.10 – Таблиця «Lesson_auditory»

Ім'я поля	Ключ	Значення за замовчуванням	Тип	Розмір
lesson_auditory_id	Первинний		int	11
lesson_id			int	11
auditory_id			int	11

Таблиця 3.11 – Таблиця «Lesson_teacher»

Ім'я поля	Ключ	Значення за замовчуванням	Тип	Розмір
lesson_teacher_id	Первинний		int	11
lesson_id			int	11
teacher_id			int	11

Таблиця 3.12 – Таблиця «Lesson_time»

Ім'я поля	Ключ	Значення за замовчуванням	Тип	Розмір
lesson_time_id	Первинний		int	11
lesson_id			int	11
number		NULL	int	11
type_of_week		NULL	int	11
day_of_week		NULL	varchar	20

3.4 Основні рішення щодо реалізації користувацького інтерфейсу

Вибір елементів керування та їхнє розміщення на формі здійснювалося з міркувань раціонального використання простору, врахування досвіду користувачів в предметній області та їхніх очікувань, сформованих можливим використанням програм-аналогів. Елементи форми, що обираються користувачем, було об'єднано в звичайні списки. Інформацію щодо обраного елемента відображено в текстових полях. Для підтвердження дій користувача було використано кнопки. При наведенні курсору миші на кнопки з'являються підказки з більш детальним поясненням їхнього призначення.

Кольори для GUI (Graphical User Interface) обиралися з метою створити форму, що справляє спокійне, консервативне враження. Для більшості елементів форми було підбрано помірно насичені кольори. Кнопки були виділені більш яскравими кольорами з метою привернення уваги користувача. Також було реалізовано виділення елементів списків за розміром чарунки та шрифту тексту шляхом використання тригерів.

Для текстових елементів було вибрано шрифт Arial, деякі з них виділено за жирним шрифтом та розміром.

3.5 Висновки з розділу

На даному етапі роботи було виконано реалізації попередньо визначених компонентів системи. Було обґрунтовано та застосовано поведінковий патерн, додано в проєкт посилання на необхідні сервіси, розроблено структури таблиць БД, а також прийнято головні рішення основні рішення щодо реалізації графічного інтерфейсу.

4 КЕРІВНИЦТВО ПРОГРАМІСТА

4.1 Призначення і умови застосування програми

Дане ПЗ призначене для автоматизації та прискорення збору даних, необхідних для ФР ЗВО.

4.2 Характеристики програми

Програма може працювати в режимі перегляду файлів та режимі завантаження даних до БД.

Програма написана з використанням мови програмування C# в IDE SharpDeveloper 5.1, для побудови графічного інтерфейсу застосовано технологію WPF, а для роботи з БД використано СУБД MySQL.

Каталог рішення займає 1,59 Мбайт (мегабайт).

Перелік файлів, з яких складається програма:

а) глобальні файли застосунку:

1) App.xaml – файл вікна програми;

2) App.xaml.cs – файл коду, пов'язаний з App.xaml;

б) файл розмітки Window1.xaml та файл пов'язаного коду Window1.xaml.cs;

в) файли з функціоналом з'єднання з БД:

1) DBUtils.cs – файл класу, екземпляр якого створює зв'язок з СУБД за вказаними параметрами (адреса, номер порту, назва БД тощо);

2) DBMySQLUtils.cs – файл класу, статичний метод якого повертає сформований рядок з'єднання з БД;

3) CellContentCorrection.cs – файл класу, методи якого відповідають за обробку рядків, считаних з Excel-файлів;

4) DbOperations.cs – файл класу, методи якого реалізують SQL-запити до БД;

5) GroupCode.cs – файл класу, метод якого повертає сформований код учбової групи;

г) файли класів, що відповідають за обробку та завантаження даних з конкретних джерел наведено в табл. 4.1.

Таблиця 4.1 – Файли класів та відповідні дані

Назва файлу	З якими даними асоціюється
Auditories.cs	Аудиторний фонд
AuditoryTypes.cs	Типи аудиторій
Statements.cs	ВД кафедр
Departments	Кафедри
Disciplines.cs	Учбові дисципліни
Faculties.cs	Факультети
StudyGroups.cs	Учбові групи
Teachers.cs	Викладацький склад

4.3 Звернення до програми

Звернення до програми здійснюється шляхом запуску файлу застосунку DataCollectionApp.exe.

4.4 Вхідні і вихідні дані

Вхідними даними програми є файли типу «Лист Microsoft Excel» з різних структурних підрозділів ЗВО НУ «Запорізька політехніка», сгруповані в окремий каталог data. Дані з ВД є минулорічними.

Вихідні дані, необхідні для ФР ЗВО, додаються та зберігаються в БД у таблицях, описаних в розділі 3.

4.5 Повідомлення

В ході роботи програма може згенерувати повідомлення, описи яких містяться в таблиці 4.2.

Таблиця 4.2 – Повідомлення та їхні описи

Повідомлення	Опис
Не обрано файл для перегляду!	Користувач натиснув кнопку «Переглянути», не обравши файл
Дані завантажено до бази даних	Операція по завантаженню загальних даних пройшла успішно
Відомості доручень завантажено до бази даних	Операція по завантаженню даних з ВД до БД пройшла успішно
Не обрано дані для завантаження!	Користувач натиснув кнопку «Завантажити», не обравши файл
Помилка при отриманні даних з файлу	Помилка при зчитуванні даних пояснюється його неправильним заповненням; такий файл слід надіслати на доопрацювання
Виникла помилка під час завантаження до бази даних!	Найімовірніше, до БД ще не були завантажені залежні дані і необхідно виконати перевірку БД
Дані вже містяться в базі даних!	Спроба завантажити ті самі, ніяк не відредаговані дані

Додаткову інформацію, як вже зазначалося у попередніх розділах, можна переглянути в «Звіті помилок», який представлено текстовим файлом BugsReport.txt.

4.6 Висновки з розділу

На даному етапі роботи було складено «Керівництво програміста», що містить призначення й умови застосування програми, її характеристики, інформацію щодо звернення до неї, опис вхідних та вихідних даних, повідомлення, які може генерувати програма в ході виконання.

5 КЕРІВНИЦТВО ОПЕРАТОРА

5.1 Призначення і умови виконання програми

Дане ПЗ призначене для автоматизації та прискорення збору даних, необхідних для ФР ЗВО.

Характеристики системи для забезпечення стабільної роботи ПЗ:

- ОС Microsoft Windows 7 x32/ 10 x64;
- процесор із тактовою частотою 2.5 ГГц і вище;
- жорсткий диск місткістю не менше, ніж 100 МБ;
- мінімум 2 ГБ обсягу оперативної пам'яті (рекомендується 4 ГБ і більше);
- додаткове ПЗ: SharpDevelop 5.1 / Visual Studio 2017 і вище; MySQL Connector Net 8.0.23 і вище.

5.2 Формат вхідних даних

Вхідними даними програми є файли типу «Лист Microsoft Excel» з різних структурних підрозділів ЗВО НУ «Запорізька політехніка», сгруповані в окремий каталог data.

Формат даних у файлі з типами аудиторій наведено на рис. 5.1. Усі значення записуються в стовпець «А».

	А	В	С
1	лабораторія		
2	комп'ютерний		
3	лекційна		
4	викладацька		
5	кафедра		
6	аспірантська		
7	Зав.Каф.		
8	інші приміщення		
9	кабінет		
10	їдальня		
11	буфет		
12	аудиторія		
13	бібліотека		
14	диспетчерська		
15	зала для креслення		
16	актова зала		
17	обслуговуюча		
18	читальна зала		

Рисунок 5.1 – Формат даних у файлі з типами аудиторій

Формат даних у файлі з дисциплінами наведено на рис. 5.2. В стовпці «G» записуються дисципліни, а в стовпці «F» – відповідні кафедри, де ці дисципліни викладаються.

D	F	G	H
Економіка та митної справа	Економіка та митної справа	Глобальна економіка	Глобальна економіка
Економіка та митної справа	Економіка та митної справа	Соціальна відповідальність	Соціальна відповідальність
Економіка та митної справа	Економіка та митної справа	Методологія наукових досліджень	Методологія наукових досліджень
Економіка та митної справа	Економіка та митної справа	Оцінка персоналу	Оцінка персоналу
Економіка та митної справа	Економіка та митної справа	Планування комерційної діяльності	Планування комерційної діяльності
Економіка та митної справа	Економіка та митної справа	Управління ризиками та страхування	Управління ризиками та страхування
Економіка та митної справа	Економіка та митної справа	Митна політика і митне регулювання	Митна політика і митне регулювання
Економіка та митної справа	Економіка та митної справа	Міжнародні митні та транспортні конвенції і угоди	Міжнародні митні та транспортні конвенції і угоди
Електричних машин	Електричних машин	Основи наукових досліджень та новітні методи пошуку і обробки інформації	Електромагнітні перехідні процеси
Електричних машин	Електричних машин	Теплові, гідравлічні та аеродинамічні розрахунки електромеханічних (електротехнічних) систем та комплексів	Електромагнітні перехідні процеси
Електричних машин	Електричних машин	Основи оптимізації складних електромеханічних (електротехнічних) систем та комплексів	Електромагнітні перехідні процеси
Електричних машин	Електричних машин	Основи функціонування та математичного забезпечення систем діагностики, контролю - вимірювальних приладів і автоматизації	Електромагнітні перехідні процеси
Електричних машин	Електричних машин	Монтаж і налагодження електротехнічних пристроїв	Електромагнітні перехідні процеси
Електричних машин	Електричних машин	Основи автоматизованого проектування ел пристроїв і електромеханічних систем	Електромагнітні перехідні процеси
Електропостачання промислових підприємств	Електропостачання промислових підприємств	Математичні задачі енергетики	Математичні задачі енергетики
Електропостачання промислових підприємств	Електропостачання промислових підприємств	Експлуатація електроустановок	Експлуатація електроустановок
Електропостачання промислових підприємств	Електропостачання промислових підприємств	Основи енергоменеджменту	Основи енергоменеджменту
Електропостачання промислових підприємств	Електропостачання промислових підприємств	Основи наукових досліджень енерговикористання	Основи наукових досліджень енерговикористання
Електропостачання промислових підприємств	Електропостачання промислових підприємств	Моделі та методи оптимізації в системі електропостачання	Моделі та методи оптимізації в системі електропостачання
Електропостачання промислових підприємств	Електропостачання промислових підприємств	Системи керування в енергетичних системах	Системи керування в енергетичних системах
Електропостачання промислових підприємств	Електропостачання промислових підприємств	Сучасний математичний апарат при проведенні наукових досліджень	Сучасний математичний апарат при проведенні наукових досліджень
Електропостачання промислових підприємств	Електропостачання промислових підприємств	Маркетинг енергії та управління проектами енерговикористання	Маркетинг енергії та управління проектами енерговикористання
Електропостачання промислових підприємств	Електропостачання промислових підприємств	Системи автоматичного проектування в енергозбереженні	Системи автоматичного проектування в енергозбереженні
Електропостачання промислових підприємств	Електропостачання промислових підприємств	Енергетичний аудит та економічні аспекти енергоефективності	Енергетичний аудит та економічні аспекти енергоефективності
Комп'ютерних систем та мереж	Комп'ютерних систем та мереж	Аналіз систем та мереж	Аналіз систем та мереж

Рисунок 5.2 – Формат даних у файлі з дисциплінами

Формат даних у файлі з факультетами наведено на рис. 5.3. В стовпці «B» – назви факультетів, в стовпці «D» – їхні коди. Повні назви факультетів зі стовпця «C» не мають значення, тому ігноруються програмою.

A	B	C	D	E
facultet	facul	faculrod	faculcod	
1	Транспортний	транспортного факультету	ТФ	
2	Інженерно-фізичний	інженерно-фізичного факультету	ІФ	
3	Економіка і управління	факультету економіки і управління	ФЕУ	
4	Електротехнічний	електротехнічного факультету	ЕТФ	
5	Машинобудівний	машинобудівного факультету	МФ	
6	Радіоелектроніки і телекомунікацій	радіоелектроніки і телекомунікацій факультету	ФРЕТ	
7	Гуманітарний	гуманітарного факультету	ГФ	
8	Комп'ютерних наук і технологій	факультету комп'ютерних наук і технологій	ФКНТ	
9	Управління фізичною культурою та спортом	факультету управління фізичною культурою та спортом	ФУФКС	
10	Міжнародного туризму та економіки	факультету міжнародного туризму та управління персоналом	ФМТЕ	
11	Соціальних наук	факультету соціальних наук	ФСН	
12	Юридичний	юридичного факультету	ЮФ	
13	Будівництва, архітектури та дизайну	факультету будівництва, архітектури та дизайну	ФБАД	

Рисунок 5.3 – Формат даних у файлі з факультетами

Формат даних у файлі з кафедрами наведено на рис. 5.4. В стовпці «А» – повні назви, в стовпці «В» – скорочені, в стовпці «С» – коди відповідних факультетів.

A	B	C	D
Автомобілів	Автомобілів	ТФ	
Будівельного виробництва та управління проектами	БВтаУП	ФБАД	
Вищої математики	ВМ	МФ	
Військової підготовки	ВП	ЗАПАС	
Двигунів внутрішнього згорання	ДВЗ	ТФ	
Деталей машин і підйомно-транспортних механізмів	ДМіПТМ	МФ	
Дизайну	Дизайну	ФБАД	
Економіки та митної справи	ЕтаМС	ФМТЕ	
Економічної теорії та підприємництва	ЕТтаП	ФЕУ	
Електричних машин	ЕМ	ЕТФ	
Електричних та електронних апаратів	ЕтаЕА	ЕТФ	
Електропостачання промислових підприємств	ЕПП	ЕТФ	
Електроприводу та автоматизації промислових установок	ЕтаАПУ	ЕТФ	
Журналістики	Журналістики	ФСН	
Загальноправових та політичних наук	ЗПтаПН	ЮФ	
Захисту інформації	ЗІ	ФРЕТ	
Іноземних мов	Ін.Мов	ЕТФ	
Іноземної мови професійного спілкування	ІМПС	ФМТЕ	
Інформаційних технологій в туризмі	ІтТ	ФМТЕ	
Інформаційних технологій електронних засобів	ІТЕЗ	ФРЕТ	
Композиційних матеріалів, хімії та технологій	КМХтаТ	ФБАД	
Комп'ютерних систем та мереж	КСтaМ	ФКНТ	
Конституційного, адміністративного та трудового права	КАтаТП	ЮФ	
Кримінального, цивільного та міжнародного права	КЦтаМП	ЮФ	
Маркетингу та логістики	Марк.та Лог.	ФЕУ	
Машин і технологій ливарного виробництва	МіТЛВ	ЮФ	
Менеджменту	Менеджменту	ФЕУ	
Металорізальних верстатів та інструментів	МРВтаІ	МФ	
Механіки	Механіки	ФБАД	
Міжнародних економічних відносин	МЕВ	ГФ	
Мікро- та наноелектроніки	МтаНЕ	ФРЕТ	
Навчальної геометрії, інженерної та комп'ютерної графіки	НГтаКГ	ТФ	

Рисунок 5.4 – Формат даних у файлі з кафедрами

Формат даних у файлі з викладачами наведено на рис. 5.5. В стовпці «А» – повні назви кафедр, в стовпці «J» – ПБ викладачів, в стовпці «К» – значення їхньої статі, в стовпці «L» – їхні посади, а в стовпці «M» – їхні звання. Значення інших стовпців («В» і «N») не мають значення, тому ігноруються програмою.

A	B	J	K	L	M	N
Кафедра (повна назва)	кафедра	Прізвище, ім'я та по батькові		Посада		
Маркетингу та логістики	Каф. Марк.та Лог.	Козицька Галина Валентинівна	ж	Доцент	к.н._доцент	економічних
Маркетингу та логістики	Каф. Марк.та Лог.	Козицька Галина Валентинівна	ж	Доцент	к.н._доцент	економічних
Маркетингу та логістики	Каф. Марк.та Лог.	Кочнова Ірина Валентинівна	ж	Ст. викладач		
Маркетингу та логістики	Каф. Марк.та Лог.	Лифар Владислава Віталіївна	ж	Зав. кафедри	д.н._професор	економічних
Маркетингу та логістики	Каф. Марк.та Лог.	Лифар Владислава Віталіївна	ж	Професор	д.н._професор	економічних
Маркетингу та логістики	Каф. ПТтаБД	Онуфрієнко Наталія Леонідівна	ж	Ст. викладач		
Маркетингу та логістики	Каф. Марк.та Лог.	Павлішина Ніна Михайлівна	ж	Доцент	к.н.	економічних
Маркетингу та логістики	Каф. Марк.та Лог.	Павлішина Ніна Михайлівна	ж	Доцент	к.н.	економічних
Маркетингу та логістики	Каф. Марк.та Лог.	Соколова Юлія Олександрівна	ж	Доцент	к.н._доцент	економічних
Маркетингу та логістики	Каф. Марк.та Лог.	Тиховська Тетяна Миколаївна	ж	Доцент	к.н._доцент	економічних
Економічної теорії та підприємництва	Каф. ЕТтаП	Афендікова Наталія Олександрівна	ж	Доцент	к.н._доцент	економічних
Економічної теорії та підприємництва	Каф. ЕТтаП	Бурма Ольга Григорівна	ж	Пров. фхівець		
Економічної теорії та підприємництва	Каф. ЕТтаП	Бурма Сергій Ігорович	м	Ст. викладач		
Економічної теорії та підприємництва	Каф. ЕТтаП	Венгерська Наталя Сергіївна	ж	Ст. викладач	к.н.	економічних
Економічної теорії та підприємництва	Каф. ЕТтаП	Губань Ольга Василівна	ж	Ст. викладач	к.н._доцент	економічних

Рисунок 5.5 – Формат даних у файлі з викладачами

Формат даних у файлі з аудиторіями наведено на рис. 5.6. В стовпці «Е» – назви аудиторій, в стовпці «G» – кількості місць, в стовпці «H» – види аудиторій, в стовпці «I» – відповідні кафедри, а в стовпці «J» – значення того, чи використовуються аудиторії («1», якщо не використовуються). Значення інших стовпців («A», «B», «C», «D», «F») не мають значення, тому ігноруються програмою.

A	B	C	D	E	F	G	H	I	J
№пп	корпус	літера	№ауд	Площа, м	Кількість місць	Вид аудиторії	Належить підрозділу (кафедрі)		не використ. оеуються
34	3	31	031	64,5	24	лабораторія	Радіотехніки та телекомунікацій		
35	3	32	032	19,6		інші приміщення	Радіотехніки та телекомунікацій		
36	3	33	033	62,9	26	лабораторія	Радіотехніки та телекомунікацій		
37	3	34	034	64,9	26	лабораторія	Радіотехніки та телекомунікацій		
38	3	35	035	20,4		кафедра	БЕЗІМЕННИЙ ПІДРОЗДІЛ		1
39	3	36	036	55,6	24	лабораторія	Захисту інформації		
40	3	37	037	65,1	28	лабораторія	Радіотехніки та телекомунікацій		
41	3	38	038	57,8	24	лабораторія	Захисту інформації		
42	3	39	039	54,7	20	лабораторія	Радіотехніки та телекомунікацій		
43	3	40	040	63,6	22	лабораторія	Інформаційних технологій електронних засобів		
44	3	41	041	21,4		кабінет	БЕЗІМЕННИЙ ПІДРОЗДІЛ		1
45	3	42	042	62,7	30	лабораторія	Інформаційних технологій електронних засобів		
46	3	43	043	39,8		викладацька	Програмних засобів		
47	3	44	044	64,6	24	лабораторія	Інформаційних технологій електронних засобів		
48	3	45	045	83,3	68	лекційна	Програмних засобів		
49	3	46	046	55,5	24	лабораторія	Захисту інформації		
50	3	47	047	64,5	25	комп'ютерний	Інформаційних технологій електронних засобів		
51	3	48	048	56,7	24	комп'ютерний	Інформаційних технологій електронних засобів		
52	3	49	049	55,3	20	лабораторія	Програмних засобів		
53	3	50	050	64,7	20	лабораторія	Комп'ютерних систем та мереж		
54	3	51	051	62,6	22	комп'ютерний	Комп'ютерних систем та мереж		
55	3	52	052	59,6	20	лабораторія	Комп'ютерних систем та мереж		
56	3	53	053a	38,2	40	комп'ютерний	Програмних засобів		
57	3	54	054	65	84	лекційна	Комп'ютерних систем та мереж		
58	3	54	054a	74,5	32	лабораторія	Автомобілі		
59	3	55	055	65,1	84	лекційна	Програмних засобів		
60	3	55	055a	49,6	16	лабораторія	Автомобілі		
61	3	56	056	55,2	30	комп'ютерний	Програмних засобів		
62	3	56	056a	74,5	36	лабораторія	Автомобілі		
63	3	57	057	54,6	36	лабораторія	Програмних засобів		
64	3	57	057a	53,4	20	комп'ютерний	Автомобілі		
65	3	57	057b	53,4	20	комп'ютерний	Автомобілі		

Рисунок 5.6 – Формат даних у файлі з аудиторіями

Формат даних у файлі з учбовими групами наведено на рис. 5.7. В стовпці «В» – назви груп, в стовпці «А» – відповідні кафедри, в стовпці «С» – кількості студентів в групах.

А	В	С
Автомобілів	Т-119	5
Автомобілів	Т-129сп(Б)	17
БВтаУП	БАД-118сп	1
БВтаУП	БАД-119	11
БВтаУП	БАД-119ін	1
БВтаУП	БАД-119сп	35
БВтаУП	БАД-319	2
БВтаУП	БАД-819	4
ДВЗ	Т-419	6
ДВЗ	Т-419сп	9
ДВЗ	Т-429сп(Т)	12
ДМіПТМ	М-319	4
ДМіПТМ	М-319сп	22
ДМіПТМ	М-329сп(Б)	1
ЕтаМС	МТЕ-219	11
ЕтаМС	МТЕ-219сп	8
ЕтаМС	МТЕ-519	2
ЕтаМС	МТЕ-619	2
ЕМ	Е-219	6
ЕМ	Е-219сп	30
ЕМ	Е-229сп	26
ЕМ	Е-239сп	27
ЕМ	Е-249сп	28
ЕтаЕА	Е-419	8
ЕтаЕА	Е-419англ	8
ЕтаЕА	Е-419сп	16
ЕтаЕА	Е-819	2
ЕтаЕА	Е-819сп	2
ЕПП	Е-119	5
ЕПП	Е-119сп	29
ЕПП	Е-519сп	2
ЕтаАПУ	Е-319	3
ЕтаАПУ	Е-319сп	27
ЕтаАПУ	Е-619сп	6
ЕтаАПУ	Е-719сп	3
Журналістики	СН-319	7
ЗПтаПН	Ю-219	18
ЗПтаПН	Ю-219сп	1

Рисунок 5.7 – Формат даних у файлі з учбовими групами

Файли ВД формуються на кафедрах НУ «Запорізька політехніка» за заздалегідь визначеним шаблоном, який описано в табл. 5.1.

Таблиця 5.1 – Шаблон ВД

Стовпець в файлі	Дані в стовпці
В	Назва дисципліни
С	№ груп
Д	Вид занять

Продовження таблиці 5.1

F	Кількість годин на тиждень
G	Контроль
H	Прізвище ім'я по батькові викладачів
I	Пропозиції щодо аудиторій
J	Пропозиції до складання розкладу занять

Формат прикладу ВД наведено на рис. 5.8.

B	C	D	E	F	G	H	I	J
Назва дисципліни	№ груп**	Вид занять	Загальна кількість годин за видом занять	Кількість годин на тиждень	Контроль	Прізвище Ім'я По батькові викладачів**	Пропозиції щодо аудиторій	Пропозиції до складання розкладу занять**
2	3	4	5	6	7	8	9	10
Основи наукових досліджень та новітні методи пошуку і обробки інформації	E-219m;E-919m	лекції	180	28	екзамен	Кошур Ігор Михайлович	268; 264	
Основи наукових досліджень та новітні методи пошуку і обробки інформації	E-219m;E-919m	лаб.роб.	180	28	екзамен	Кошур Ігор Михайлович	268; 264	Страница 4
Теплові, гідравлічні та аеродинамічні розрахунки електромеханічних (електротехнічних) систем та комплексів	E-219m;E-919m	лекції	180	28	екзамен	Яримбаш Сергій Тимофійович	268; 264;	
Теплові, гідравлічні та аеродинамічні розрахунки електромеханічних (електротехнічних) систем та комплексів	E-219m;E-919m	лаб.роб.	180	28	екзамен	Солодовнікова Тетяна Павлівна	268; 264	
Основи оптимізації складних електромеханічних (електротехнічних) систем та комплексів	E-219m;E-919m	лекції	180	28	екзамен	Яримбаш Дмитро Сергійович	268; 264;	
Основи оптимізації складних електромеханічних (електротехнічних) систем та комплексів	E-219m;E-919m	лаб.роб.	180	28	екзамен	Кулавіна Євгенія Вікторівна	268; 264	
Основи функціонування та математичного забезпечення систем діагностики, контролю - вимірювальних приладів і автоматики	E-219m	лекції	180	28	екзамен	Кулавіна Євгенія Вікторівна	268; 264	
Основи функціонування та математичного забезпечення систем діагностики, контролю - вимірювальних приладів і автоматики	E-219m	лаб.роб.	180	28	екзамен	Кулавіна Євгенія Вікторівна	268; 264	
Електричні машини	E-219en;E-229en;E-239en;E-249en	лекції	240	48	екзамен	Яримбаш Дмитро Сергійович	240	
Електричні машини	E-219en;E-229en;E-239en;E-249en	практ.роб.	240	14	екзамен	Дичук Тетяна Євгенівна	240	
Електричні машини	E-219en;E-229en;E-239en;E-249en	лаб.роб.	240	14	екзамен	Дичук Тетяна Євгенівна	5; 9a	
Інженерне проектування електричних машин та трансформаторів	E-216;E-227en	лекції	195	16	екзамен	Солодовнікова Тетяна Павлівна	268; 234	
Інженерне проектування електричних	E-216;E-227en	лаб.роб.	195	9	екзамен	Солодовнікова Тетяна Павлівна	268; 234	

Рисунок 5.8 – Формат даних у файлі з ВД кафедри «Електричних машин» НУ «Запорізька політехніка»

5.3 Виконання програми

Після успішного запуску ПЗ відкривається форма, що представлена на рис. 5.9.

Загальні дані

Типи аудиторій
Дисципліни
Факультети
Кафедри
Викладачі
Аудиторії
Учбові групи

Відомості доручень

Автомобілів
Будівельного виробництва та управління проектами
Вищої математики
Військової підготовки
Двигунів внутрішнього згорання
Деталей машин і підійомно-транспортних механізмів
Дизайну

Обраний файл

Назва:

Файл:

Останнє редагування:

Переглянути **Завантажити** **Переглянути звіт про помилки в файлах**

Рисунок 5.9 – Форма програми

Спочатку необхідно обрати файл із представлених переліків: «Загальні дані» і «Відомості доручень», клацнувши його лівою кнопкою миші. Після цього в правій частині форми «Обраний файл», з'явиться описова інформація щодо цього файлу (приклад наведено на рис. 5.10): його назва в списку форми («Назва»), назва файлу в каталозі («Файл») і дата останнього редагування («Останнє редагування»). Таким чином, оператор може оцінити, що собою представляє отриманий файл, і, найголовніше, коли востаннє було здійснення внесення правок.

Загальні дані

Типи аудиторій
Дисципліни
Факультети
Кафедри
Викладачі
Аудиторії
Учбові групи

Відомості доручень

Міжнародних економічних відносин
Мікро- та наноелектроніки
Нарисної геометрії, інженерної та комп'ютерної графіки
Обладнання та технології зварювального виробництва
Облік і оподаткування
Обробки металів тиском
Охорони праці і навколишнього середовища

Обраний файл

Назва:

Облік і оподаткування

Файл:

Облік і оподаткування_ВІДОМІСТЬ ДОРУЧЕНЬ - 2020.xlsx

Останнє редагування:

19.05.2021 22:35:44

Переглянути **Завантажити** **Переглянути звіт про помилки в файлах**

Рисунок 5.10 – Описова інформація обраного файлу

Після цього можна здійснити перегляд файлу (приклад наведено на рис. 5.11), натиснувши відповідну кнопку в нижній частині форми, щоб поверхнево оцінити його зміст – Excel-файл буде відкрито в режимі читання.

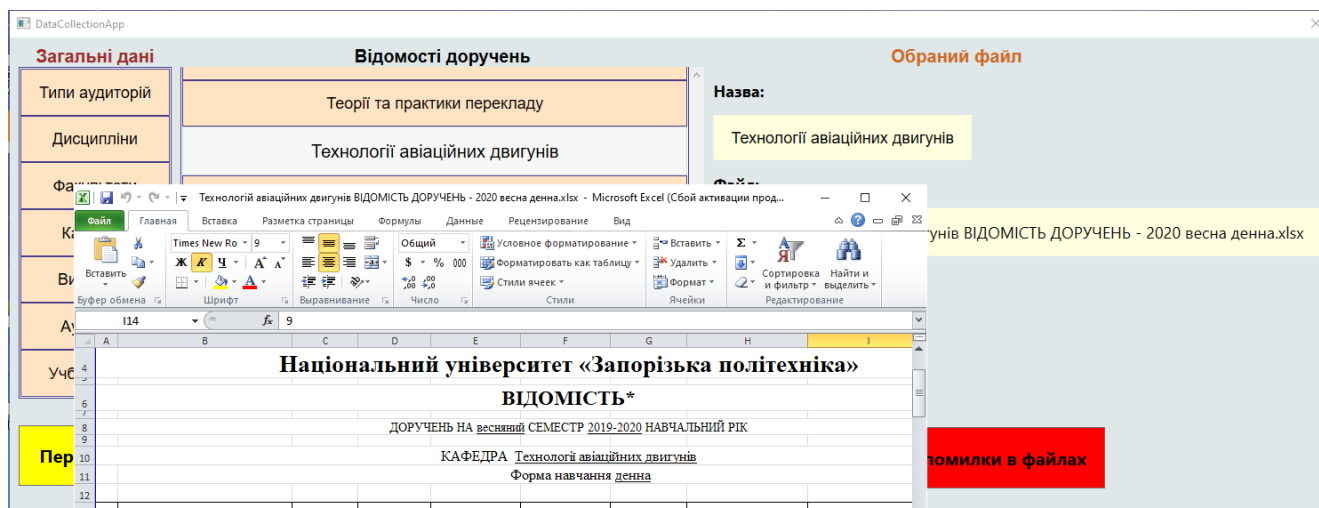


Рисунок 5.11 – Перегляд обраного файлу

Далі можна виконати завантаження файлу, натиснувши відповідну кнопку в нижній частині форми. Після натискання розпочнеться процес завантаження, що триває приблизно 10-15 секунд (в залежності від розміру файлу), після чого з'явиться повідомлення про результат операції (приклад наведено на рис. 5.12).

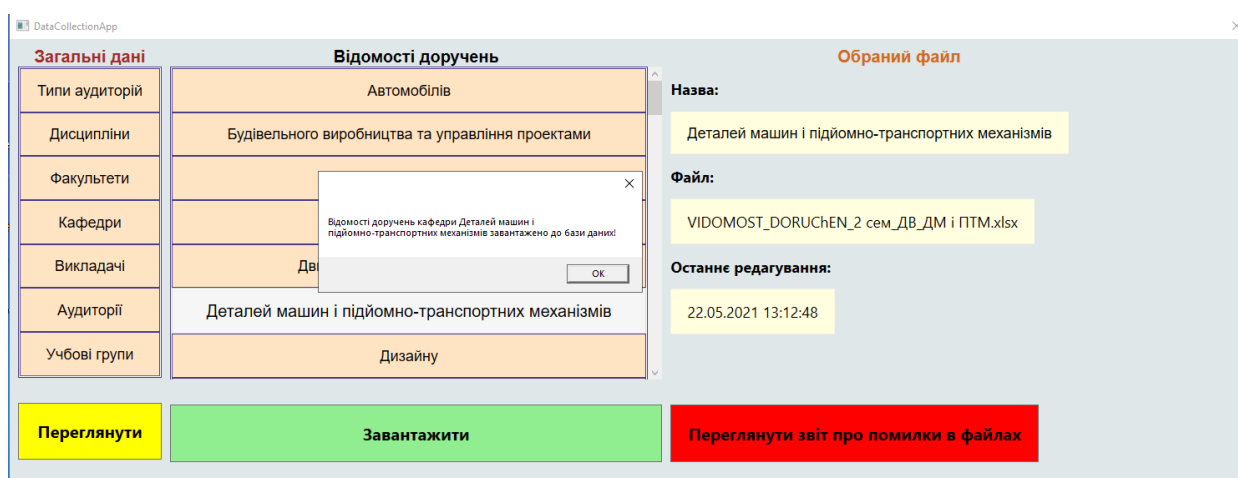


Рисунок 5.12 – Повідомлення про завантаження ВД до БД

Окрім роботи з файлами, можна ознайомитись зі «Звітом помилок» (приклад якого наведено на рис. 5.13) – файл з неточностями, дата операції з ним, перелік неточностей.

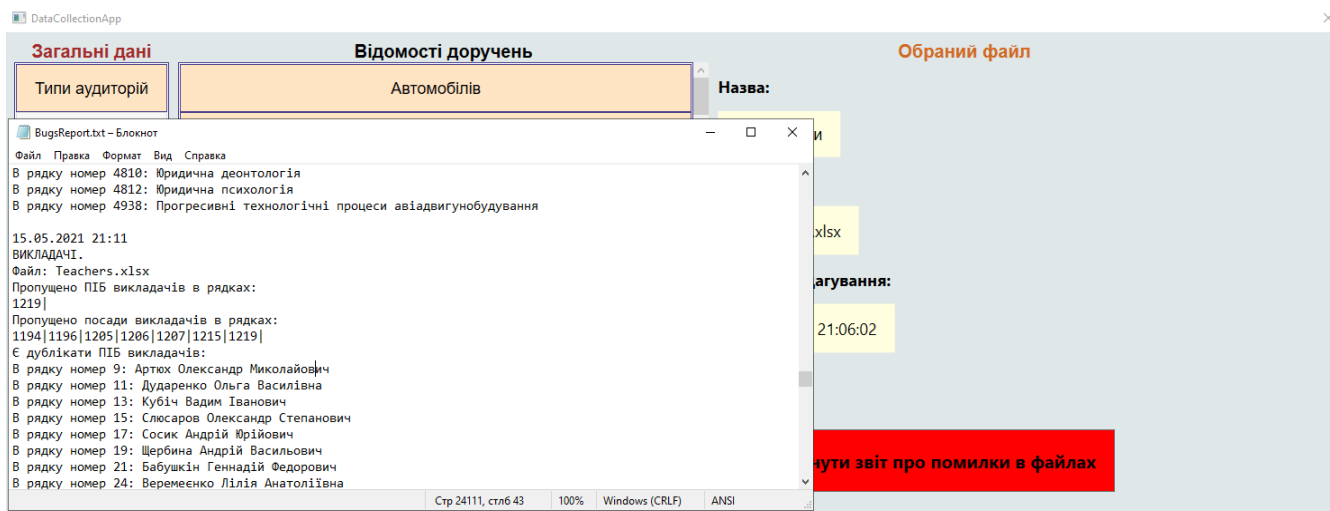


Рисунок 5.13 – Перегляд «Звіту помилок»

5.4 Повідомлення оператору

Під час виконання програми генеруються повідомлення, які інформують користувача щодо результатів виконання операцій, зокрема, можливих помилок та їхніх причин тощо. Перелік повідомлень та їхні описи наведено в попередньому розділі в таблиці 4.2.

5.5 Висновки по розділу

На даному етапі роботи було складено «Керівництво оператора», що містить призначення і умови виконання програми, описано процес виконання програми, а також приведено перелік повідомлень та їхні описи.

ВИСНОВКИ

В ході виконання роботи було виконано аналіз предметної області, виконано розробку архітектури програми, реалізовано її компоненти, а також оформлено такі програмні документи як «Керівництво програміста», «Керівництво оператора» та «Технічне завдання».

На першому етапі роботи було ретельно проаналізовано предметну область: сформовано проблему та актуальність роботи, розглянуто існуючі методи вирішення розв’язуваної проблеми та готові рішення, виявлено їх переваги та недоліки. Наступним кроком стала розробка архітектури системи, в ході якої було описано структуру розроблюваної програми, обрано інструменти для розробки ПЗ, а також спроектовано майбутню БД. На наступному етапі було реалізовано компоненти програми на основі «Технічного завдання» та надано їхнє обґрунтування. Після цього було складено «Керівництво програміста» та «Керівництво оператора».

В результаті роботи було отримано програмний продукт, що дозволяє виконати поставлене завдання, а саме – автоматизацію та прискорення процесу збору даних, необхідних для ФР ЗВО. Роботу програми було протестовано на реальних даних, отриманих зі структурних підрозділів ЗВО НУ «Запорізька політехніка». За необхідності програма може бути модифікована та використана під час розробки більш складної, комплексної системи, що вирішуватиме завдання ФР ЗВО НУ «Запорізька політехніка».

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Симоненко В.П. Метод пошагового конструирования для составления расписания занятий в учебных заведениях / В.П. Симоненко, С.И. Симоненко // Систем. дослідж. та інформ. технології. – 2008. – №4. – с.76-85. – Бібліогр.: 5 назв. – рос.
2. Алгоритм составления расписания занятий / Ю.В. Береговых, Б.А. Васильев, Н.А. Володин // Штучний інтелект. – 2009. – №2. – с.50-56 – Бібліогр.: 7 назв. – рос.
3. Форма додання активності [Електрон. ресурс]. – Режим доступу: <https://lalescu.ro/liviu/fet/screenshots/english-1/007-08-addActivity.png>
4. Формування учбового розкладу [Електрон. ресурс]. – Режим доступу: <https://solutions.1c.ru/upload/v8/UserFiles/Image/abis-bsc/1asp.png>
5. C# 4.0: полное руководство.: Пер. с англ. – М.: ООО «И.Д. Вильямс», 2011. – 1056 с.: ил. – Парал. тит. англ.
6. Язык программирования C# 7 и платформы .NET и .NET Core, 8-е изд. : Пер. с англ. – СПб. : ООО «Диалектика», 2018. – 1328 с. : ил. – Парал. тит. англ.
7. Введение в WPF. Особенности платформы WPF. [Электрон. ресурс]. – Режим доступу: <https://metanit.com/sharp/wpf/1.php>
8. SharpDevelop [Електрон. ресурс]. – Режим доступу: <https://uk.wikipedia.org/wiki/SharpDevelop>
9. MySQL [Електрон. ресурс]. – Режим доступу: <https://uk.wikipedia.org/wiki/MySQL>
10. phpMyAdmin [Електрон. ресурс]. – Режим доступу: <https://uk.wikipedia.org/wiki/PhpMyAdmin>
11. Введение в системы баз данных, 8-е издание.: Пер. с англ. – М.: Издательский дом «Вильямс», 2005. – 1328 с.: ил. – Парал. тит. англ.
12. Изучаем SQL. – Пер. с англ. – СПб: Символ-Плюс, 2007. – 312 с., ил.

13. SQL-запросы для простых смертных. Практическое руководство по манипулированию данными в SQL. – Пер. с англ. – Издательство «Лори», 2003. – 459 с., ил.

14. Шаблонный метод [Электрон. ресурс]. – Режим доступа: <https://refactoring.guru/uk/design-patterns/template-method>

ДОДАТОК А
Технічне завдання

A.1 Підстави для розробки

Підставою для розробки є завдання на дипломний проєкт на тему «Програмне забезпечення для збору даних для формування розкладу закладу вищої освіти», затверджене наказом по НУ «Запорізька політехніка» № 103 від 30 березня 2021 року.

A.2 Призначення розробки

Дане ПЗ призначено для використання в ЗВО, зокрема, в НУ «Запорізька політехніка».

Було встановлено, що успішність процесу ФР ЗВО залежить від якості процесу збору усіх необхідних даних. Використовуючи розроблене ПЗ, диспетчери ЗВО можуть скоротити час роботи за рахунок автоматизації процесу збору даних, а також уникнути проблем, що виникають в результаті ручного опрацювання даних.

A.3 Вимоги до програми

A.3.1 Вимоги до функціональних характеристик

Вимоги до функціональних характеристик:

- реалізація графічного інтерфейса користувача;
- реалізація можливості завантаження даних до БД;
- реалізація можливості перегляду файлів;
- реалізація ведення «Звіту помилок» щодо неточностей у вхідних даних.

А.3.2 Вимоги надійності

Програма повинна реалізовувати наступні вимоги по надійності:

- захист від спроби завантажити дані, що не відповідають встановленим взаємозв'язкам між сутностями предметної області;
- захист від спроби повторного завантаження тих самих даних;
- обробка виняткових ситуацій (наприклад, помилка відкриття до Excel-файлу);
- стабільне функціонування при безперервній роботі комп'ютера.

А.3.3 Умови експлуатації

Умови експлуатації програми відповідають вимогам до експлуатації технічних засобів (комп'ютера, сервера тощо). Для використання програми не потребується додаткова кваліфікація користувача.

А.3.4 Вимоги до складу і параметрів технічних засобів

Вимоги до складу і параметрів технічних засобів:

- ОС Microsoft Windows 7 x32/ 10 x64;
- процесор із тактовою частотою 2.5 ГГц і вище;
- жорсткий диск місткістю не менше, ніж 100 МБ;
- мінімум 2 ГБ обсягу оперативної пам'яті (рекомендується 4 ГБ і більше);
- додаткове ПЗ: SharpDevelop 5.1 / Visual Studio 2017 і вище; MySQL Connector Net 8.0.23 і вище.

A.3.5 Вимоги до маркування та упакування

Програмне забезпечення не потребує додаткових операцій щодо маркування та упакування.

A.3.6 Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання програми відповідають аналогічним вимогам носія, на якому вона записана.

A.3.7 Вимоги до програмної документації

Склад програмної документації:

- технічне завдання;
- керівництво програміста;
- керівництво оператора.

A.3.8 Стадії і етапи розробки

Стадії і етапи розробки програми:

- аналіз предметної області;
- проєктування;
- реалізація;
- тестування.

A.3.9 Порядок контролю та приймання

Порядок контролю та приймання відповідає затвердженому порядку контролю та приймання дипломних кваліфікаційних робіт студентів денної та заочної форми навчання освітнього ступеня «бакалавр» НУ «Запорізька політехніка».

ДОДАТОК Б
Текст програми

Б.1 Файл App.xaml

```
<Applicationx:Class="DataCollectionApp.App"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    StartupUri="Window1.xaml">
    <Application.Resources></Application.Resources>
</Application>
```

Б.2 Файл App.xaml.cs

```
using System;
using System.Windows;
using System.Data;
using System.Xml;
using System.Configuration;
namespace Data Collection App
{ public partial class App :Application  {} }
```

Б.3 Файл Auditories.cs

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.IO;
using System.Windows;
namespace DataCollectionApp{
    public class Auditories : ExcelFile{
        ArrayList names = new ArrayList();
        ArrayList types = new ArrayList();
        ArrayList departments = new ArrayList();
        ArrayList notUsed = new ArrayList();
        ArrayList places = new ArrayList();
        ArrayList corpsNumbers = new ArrayList();
        const char corpsNumberColumn = 'B';
        const char namesColumn = 'E';
        const char typesColumn = 'H';
        const char departmentsColumn = 'I';
        const char notUsedColumn = 'J';
```

```

const char placesColumn = 'G';
bool reading = true;
int row = 2;
ArrayList missingValuesOfNames = new ArrayList();
ArrayList missingValuesOfTypes = new ArrayList();
ArrayList missingValuesOfDepartments = new ArrayList();
Dictionary <int, string> duplicatesOfNames = new Dictionary<int, string>();
DbOperations dbo = new DbOperations();
public Auditories(string fileName): base(fileName)
{ this.fileName = fileName; }
public override void readFromExcelFile(){
    try{
        open(1);
        for(int col = getColumnNumber(namesColumn), i = row; i <= rowsCount; i++){
            cellContent = getCellContent(i, col);
            string trimmedCellContent = cellContent.TrimStart('0');
            if (!string.IsNullOrEmpty(cellContent) && string.IsNullOrEmpty(trimmedCellContent))
            { trimmedCellContent = "0"; }
            if (names.Contains(trimmedCellContent))
            { duplicatesOfNames.Add(i, trimmedCellContent); }
            if(string.IsNullOrEmpty(trimmedCellContent))
            { missingValuesOfNames.Add(i); }
            else
            { names.Add(trimmedCellContent); } }
        for(int col = getColumnNumber(corpsNumberColumn), i = row; i <= rowsCount; i++){
            cellContent = getCellContent(i, col);
            if(string.IsNullOrEmpty(cellContent)){
                corpsNumbers.Add(null);}
            else{
                corpsNumbers.Add(cellContent); } }
        for(int col = getColumnNumber(typesColumn), i = row; i <= rowsCount; i++){
            cellContent = getCellContent(i, col);
            if(string.IsNullOrEmpty(cellContent))
            { missingValuesOfTypes.Add(i); }
            else
            { types.Add(cellContent); } }
        for(int col = getColumnNumber(departmentsColumn), i = row; i <= rowsCount; i++){
            cellContent = getCellContent(i, col);
            if(string.IsNullOrEmpty(cellContent))
            { missingValuesOfDepartments.Add(i); }
            else
            { departments.Add(cellContent); } }
    }
}

```

```

for(int col = getColumnNumber(notUsedColumn), i = row; i <= rowCount; i++){
    cellContent = getCellContent(i, col);
    if(string.IsNullOrEmpty(cellContent))
    { notUsed.Add(false); }
    else
    { notUsed.Add(true); }}
for(int col = getColumnNumber(placesColumn), i = row; i <= rowCount; i++){
    cellContent = getCellContent(i, col);
    if(string.IsNullOrEmpty(cellContent))
    { places.Add(null); }
    else
    { places.Add(Convert.ToInt32(cellContent)); }}
close();}
catch (Exception ex){
    reading = false;
    MessageBox.Show("Помилка при зчитуванні даних з файлу " + FileName + " " + ex.Message);} }
public override void evaluateData(){
    if(reading){
        const string path = @"E:\BACHELORS WORK\TIMETABLE\DataCollectionApp\BugsReport.txt";
        if(missingValuesOfNames.Count != 0 || missingValuesOfTypes.Count !=0 ||
        missingValuesOfDepartments.Count != 0 || duplicatesOfNames.Count != 0){
            using (StreamWriter sw = new StreamWriter(path, true, System.Text.Encoding.Default)){
                sw.WriteLine("{0:g}", DateTime.Now);
                sw.WriteLine("АУДИТОРІЇ.");
                sw.WriteLine("Файл: " + FileName);}}
        if(missingValuesOfNames.Count != 0){
            using (StreamWriter sw = new StreamWriter(path, true, System.Text.Encoding.Default)){
                sw.WriteLine("Пропущено назви аудиторій в рядках: ");
                foreach (int value in missingValuesOfNames)
                { sw.Write(value + "|"); }
                sw.WriteLine();}}
        if(missingValuesOfTypes.Count != 0){
            using (StreamWriter sw = new StreamWriter(path, true, System.Text.Encoding.Default)){
                sw.WriteLine("Пропущено типи аудиторій в рядках: ");
                foreach (int value in missingValuesOfTypes)
                { sw.Write(value + "|"); }
                sw.WriteLine();}}
        if(missingValuesOfDepartments.Count != 0){
            using (StreamWriter sw = new StreamWriter(path, true, System.Text.Encoding.Default)){
                sw.WriteLine("Пропущено назви кафедр в рядках: ");
                foreach (int value in missingValuesOfDepartments)
                { sw.Write(value + "|"); }

```

```

        sw.WriteLine();}}
if(duplicatesOfNames.Count != 0){
    using (StreamWriter sw = new StreamWriter(path, true, System.Text.Encoding.Default)){
        sw.WriteLine("Є дублікати назв аудиторій: ");
        foreach (KeyValuePair<int, string> duplicate in duplicatesOfNames)
        { sw.WriteLine("В рядку номер " + duplicate.Key + ": " + duplicate.Value); }
        sw.WriteLine();}}
bool noSensetoReload = true;
ArrayList auditoriesInDB = dbo.getAuditoryNames();
foreach(string name in names){
    if(!auditoriesInDB.Contains(name)){
        noSensetoReload = false;
        break;}}
if (noSensetoReload){
    reading = false;
    MessageBox.Show("Дані про аудиторії вже містяться в базі даних!"); } }
public override void load(){
    if(reading){
        try{
            for(int i = 0; i < rowsCount - row + 1; i++){
                int auditoryTypeID = dbo.getAuditoryTypeID(types[i].ToString());
                int departmentID = dbo.getDepartmentIDbyFullName(departments[i].ToString());
                dbo.insertAuditory(departmentID, names[i].ToString(), Convert.ToBoolean(notUsed[i]),
                    auditoryTypeID, Convert.ToInt32(places[i]), Convert.ToInt32(corpsNumbers[i])); }
                MessageBox.Show("Дані про аудиторії завантажено до бази даних!"); }
            catch (Exception ex){
                MessageBox.Show(ex.Message); } } } } }

```

Б.4 Файл AuditoryTypes.cs

```

using System;
using System.Collections;
using System.Collections.Generic;
using System.IO;
using System.Windows;
namespace DataCollectionApp{
    public class AuditoryTypes : ExcelFile{
        ArrayList records = new ArrayList();
        ArrayList missingValues = new ArrayList();
        Dictionary <int, string> duplicates = new Dictionary<int, string>();
        int row = 1;

```

```

const char column = 'A';
bool reading = true;
DbOperations dbo = new DbOperations();
public AuditoryTypes(string fileName): base(fileName)
{ this.fileName = fileName; }
public override void readFromExcelFile()
{
    try{
        open(1);
        for(int col = getColumnNumber(column), i = row; i <= rowCount; i++){
            cellContent = getCellContent(i, col);
            if(records.Contains(cellContent))
            { duplicates.Add(i, cellContent); }
            else
            { records.Add(cellContent); }
            if(string.IsNullOrEmpty(cellContent))
            { missingValues.Add(i); }
        }
        close();
    }
    catch(Exception ex){
        reading = false;
        MessageBox.Show("Помилка при отриманні даних з файлу " + FileName + ". " + ex.Message);
    }
}
public override void evaluateData()
{
    if (reading){
        const string path = @"E:\BACHELORS WORK\TIMETABLE\DataCollectionApp\BugsReport.txt";
        if(missingValues.Count != 0 || duplicates.Count != 0){
            using (StreamWriter sw = new StreamWriter(path, true, System.Text.Encoding.Default)){
                sw.WriteLine("{0:g}", DateTime.Now);
                sw.WriteLine("ТИПИ АУДИТОРІЙ.");
                sw.WriteLine("Файл: " + FileName);
            }
        }
        if (missingValues.Count != 0){
            using (StreamWriter sw = new StreamWriter(path, true, System.Text.Encoding.Default)){
                sw.WriteLine("Є пропуски в рядках: ");
                foreach (int value in missingValues)
                { sw.WriteLine(value + "|"); }
                sw.WriteLine();
            }
        }
        if(duplicates.Count != 0){
            using (StreamWriter sw = new StreamWriter(path, true, System.Text.Encoding.Default)){
                sw.WriteLine("Є дублікати: ");
                foreach (KeyValuePair<int, string> duplicate in duplicates)
                { sw.WriteLine("В рядку номер " + duplicate.Key + ": " + duplicate.Value); }
                sw.WriteLine();
            }
        }
        bool noSensetoReload = true;
        ArrayList auditoryTypesInDB = new ArrayList();
    }
}

```

```

        auditoryTypesInDB = dbo.getAuditory_types();
        foreach (string record in records){
            if(!auditoryTypesInDB.Contains(record)){
                noSensetoReload = false;
                break; } }
        if (noSensetoReload) {
            reading = false;
            MessageBox.Show("Дані про типи аудиторій вже містяться в базі даних!"); } } }
    public override void load(){
        if(reading){
            try{
                foreach (string record in records){
                    dbo.insertAuditory_type(record);}
                MessageBox.Show("Дані про типи аудиторій завантажено до бази даних!"); }
            catch(Exception ex) {
                MessageBox.Show("Виникла помилка під час завантаження даних про типи аудиторій з файлу " +
                FileName + " до бази даних!"); } } } }

```

Б.5 Файл ExcelFile.cs

```

using System;
using System.IO;
using System.Diagnostics;
using System.Windows;
using Excel = Microsoft.Office.Interop.Excel;
namespace DataCollectionApp{
    public class ExcelFile{
        protected string fileName;
        protected const string directoryName = @"E:\BACHELORS WORK\TIMETABLE\data";
        protected Excel.Application app;
        protected Excel.Workbook workbook;
        protected Excel.Worksheet worksheet;
        protected Excel.Range range;
        protected int rowCount;
        protected string cellContent;
        protected string wrongFileName;
        public string FileName{
            get { return fileName; }
            set{
                fileName = null;
                bool containsForbiddenSymbols = false;

```

```

const string forbiddenSymbols = @"\" + "/*?\"<>|";
foreach (char ch in forbiddenSymbols){
    if (value.IndexOf(ch) != -1){
        containsForbiddenSymbols = true;
        break;} }
if (containsForbiddenSymbols == false && (value.EndsWith(".xlsx") || value.EndsWith(".xls"))){
    fileName = value;}
else{
    wrongFileName = value;
    MessageBox.Show("Некоректне ім'я або розширення файлу \"" + wrongFileName +
        "\". Ім'я файлу не повинно містити наступних символів: " + forbiddenSymbols +
        "\nРозширення файлу повинно бути \".xls\" або \".xlsx\""); } }
public string fullPathToFile{
    get { return String.Concat(directoryName, @"\", fileName); } }
public ExcelFile(){ }
public ExcelFile(string fileName)
{ FileName = fileName; }
public void open(int sheetNumber){
    if (fileName == null){
        MessageBox.Show("Помилка відкриття файлу \"" + wrongFileName +
            "\". Перевірте правильність імені файлу та повторіть спробу."); }
    else{
        app = new Excel.Application();
        const int readingMode = 0;
        try{
            workbook = app.Workbooks.Open(fullPathToFile, readingMode, true);
            worksheet = (Excel.Worksheet)workbook.Worksheets[sheetNumber];
            range = worksheet.UsedRange;
            rowsCount = range.Rows.Count; }
        catch (Exception ex) {
            MessageBox.Show("Помилка відкриття файлу \"" + wrongFileName +
                "\". Перевірте правильність імені файлу та повторіть спробу."); } } }
public int sheetsCount{
    get {
        app = new Excel.Application();
        const int readingMode = 0;
        workbook = app.Workbooks.Open(fullPathToFile, readingMode, true);
        return workbook.Worksheets.Count; } }
public int getLastRow (string fileName, int sheetNumber, int firstRow){
    open(sheetNumber);
    int lastRow = firstRow;
    int columnNumber = getColumnNumber('B');

```

```

        cellContent = getCellContent(lastRow, columnNumber);
        while(string.IsNullOrEmpty(cellContent) == false){
            cellContent = getCellContent(lastRow, columnNumber);
            ++lastRow;}

        close();
        return lastRow - 2; }

public void openForViewing()
{    Process.Start(fullPathToFile); }

public string lastWriteTime{
    get{
        FileInfo f = new FileInfo(fullPathToFile);
        f.Refresh();
        return f.LastWriteTime.ToString(); } }

public bool exists(){
    if(!File.Exists(fullPathToFile)){
        MessageBox.Show("Файл не знайдено: " + "\"" + wrongFileName + "\"");
        return false;}
    return true;}

public void close(){
    try{
        workbook.Close(false);
        app.Quit(); }
    catch (Exception ex) {
        MessageBox.Show("Помилка при закритті файлу: \"" + wrongFileName + "\"."); } }

protected int getColumnNumber(char column) {
    if ((int)column >= 'A' && (int)column <= 'Z')
        { return (int)column - 64; }
    else {
        MessageBox.Show("Некоректне ім'я стовпця: " + column + ". Неможливо зчитати дані з файлу " +
        FileName);
        return -1; } }

protected string getCellContent(int row, int column)
{ return ((Excel.Range)worksheet.Cells[row, column]).Text.ToString(); }

public void sendDataToDB() {
    readFromExcelFile();
    evaluateData();
    load(); }

public virtual void readFromExcelFile(){ }
public virtual void evaluateData(){ }
public virtual void load(){ } }

```

Б.6 Файл CellContentCorrection.cs

```
using System;
namespace DataCollectionApp{
    public class CellContentCorrection{
        public string correctAuditories(string auditories){
            auditories = auditories.TrimEnd(new char [] {';', ':'});
            auditories = auditories.Replace(" ", "");
            return auditories;}
        public string correctTeachers(string teachers){
            return teachers.TrimEnd(new char [] {';', ':'}); }
        public string correctGroups(string groups){
            groups = groups.Trim();
            groups = groups.TrimEnd(new char [] {';', ':'});
            groups = groups.Replace(" ", "");
            groups = groups.Replace("\n", "");
            return groups; }
        public string correctTime(string time) {
            time = time.Trim();
            time = time.TrimEnd(new char [] {';', ':'});
            time = time.Replace(" ", "");
            time = time.Replace("\n", "");
            return time; } } }
```

Б.7 Файл DbOperations.cs

```
using System;
using System.Collections;
using System.Collections.Generic;
using MySql.Data.MySqlClient;
namespace DataCollectionApp{
    public class DbOperations{
        MySqlConnection mySqlConnection = DBUtils.getDBConnection();
        MySqlCommand mySqlCommand;
        MySqlDataReader mySqlDataReader;
        public int getID(string command, string parameter, string parameterValue){
            mySqlConnection.Open();
            mySqlCommand = new MySqlCommand(command, mySqlConnection);
            mySqlCommand.Parameters.AddWithValue(parameter, parameterValue);
```

```

        mySqlCommand.ExecuteNonQuery();
        int ID = Convert.ToInt32(mySqlCommand.ExecuteScalar().ToString());
        mySqlConnection.Close();
        return ID;}

public ArrayList getArrayList(string command){
    ArrayList arrayList = new ArrayList();
    mySqlConnection.Open();
    mySqlCommand = new MySqlCommand(command, mySqlConnection);
    mySqlDataReader = mySqlCommand.ExecuteReader();
    while(mySqlDataReader.Read()){
        arrayList.Add(mySqlDataReader[0].ToString());}
    mySqlConnection.Close();
    return arrayList;}

public Dictionary<int, string> getDepartments(){
    Dictionary <int, string> departments = new Dictionary<int, string>();
    mySqlConnection.Open();
    const string getDepartments = "SELECT department_id, full_name FROM department WHERE department_id
< 60";
    mySqlCommand = new MySqlCommand (getDepartments, mySqlConnection);
    using(mySqlDataReader = mySqlCommand.ExecuteReader()){
        while(mySqlDataReader.Read()){
            departments.Add(Convert.ToInt32(mySqlDataReader[0]), mySqlDataReader[1].ToString());}
    mySqlConnection.Close();
    return departments; }

public int getDepartmentID(string departmentName){
    const string command = "SELECT department_id FROM department WHERE short_name =
@DEPARTMENT";
    return getID(command, "@DEPARTMENT", departmentName); }

public int getDisciplineID(string discipline){
    const string command = "SELECT discipline_id FROM discipline WHERE full_name = @DISCIPLINE";
    return getID(command, "@DISCIPLINE", discipline); }

public void insertLesson(int disciplineID, string lessonsType, int hours, string lessonsControl, int departmentID){
    const string command = "INSERT INTO lesson (discipline_id, type, countOfHours, control, department_id) "
        + "VALUES (@DISCIPLINE_ID, @TYPE, @HOURS, @CONTROL, @DEPARTMENT_ID)";
    mySqlConnection.Open();
    mySqlCommand = new MySqlCommand(command, mySqlConnection);
    mySqlCommand.Parameters.AddWithValue("@DISCIPLINE_ID", disciplineID);
    mySqlCommand.Parameters.AddWithValue("@TYPE", lessonsType);
    mySqlCommand.Parameters.AddWithValue("@HOURS", hours);
    mySqlCommand.Parameters.AddWithValue("@CONTROL", lessonsControl);
    mySqlCommand.Parameters.AddWithValue("@DEPARTMENT_ID", departmentID);
    mySqlCommand.ExecuteNonQuery();

```

```

        mySqlConnection.Close();}

public int getLastLessonID(){
    const string command = "SELECT lesson_id FROM lesson ORDER BY lesson_id DESC LIMIT 1";
    mySqlConnection.Open();
    mySqlCommand = new MySqlCommand(command, mySqlConnection);
    mySqlCommand.ExecuteNonQuery();
    int lessonID = Convert.ToInt32(mySqlCommand.ExecuteScalar().ToString());
    mySqlConnection.Close();
    return lessonID;}

public int getAuditoryID(string auditory){
    const string command = "SELECT auditory_id FROM auditory WHERE auditory_name = @AUDITORY";
    return getID(command, "@AUDITORY", auditory);}

public void insertLesson_Auditory(int lastLessonID, int auditoryID){
    const string command = "INSERT INTO lesson_auditory (lesson_id, auditory_id) "
        + "VALUES (@LESSON_ID, @AUDITORY_ID)";
    mySqlConnection.Open();
    mySqlCommand = new MySqlCommand(command, mySqlConnection);
    mySqlCommand.Parameters.AddWithValue("@LESSON_ID", lastLessonID);
    mySqlCommand.Parameters.AddWithValue("@AUDITORY_ID", auditoryID);
    mySqlCommand.ExecuteNonQuery();
    mySqlConnection.Close();}

public int getTeacherID(string teacher){
    const string command = "SELECT teacher_id FROM teacher WHERE full_name = @TEACHER";
    return getID(command, "@TEACHER", teacher);}

public void insertLesson_Teacher(int lastLessonID, int teacherID){
    const string command = "INSERT INTO lesson_teacher (lesson_id, teacher_id) "
        + "VALUES (@LESSON_ID, @TEACHER_ID)";
    mySqlConnection.Open();
    mySqlCommand = new MySqlCommand(command, mySqlConnection);
    mySqlCommand.Parameters.AddWithValue("@LESSON_ID", lastLessonID);
    mySqlCommand.Parameters.AddWithValue("@TEACHER_ID", teacherID);
    mySqlCommand.ExecuteNonQuery();
    mySqlConnection.Close(); }

public ArrayList getStudy_groups(){
    const string command = "SELECT full_name FROM study_group";
    ArrayList studyGroups = new ArrayList();
    mySqlConnection.Open();
    mySqlCommand = new MySqlCommand(command, mySqlConnection);
    mySqlDataReader = mySqlCommand.ExecuteReader();
    while(mySqlDataReader.Read()){
        studyGroups.Add(mySqlDataReader[0].ToString()); }
    mySqlConnection.Close();
}

```

```

        return studyGroups; }

public void insertStudy_group(int departmentID, string name, string code){
    const string command = "INSERT INTO study_group (department_id, full_name, study_group_code) "
    + "VALUES (@DEPARTMENT_ID, @NAME, @CODE)";
    MySqlConnection.Open();
    MySqlCommand = new MySqlCommand(command, MySqlConnection);
    MySqlCommand.Parameters.AddWithValue("@DEPARTMENT_ID", departmentID);
    MySqlCommand.Parameters.AddWithValue("@NAME", name);
    MySqlCommand.Parameters.AddWithValue("@CODE", code);
    MySqlCommand.ExecuteNonQuery();
    MySqlConnection.Close();}

public int getStudy_groupID(string study_group){
    const string command = "SELECT study_group_id FROM study_group WHERE full_name = @GROUP";
    return getID(command, "@GROUP", study_group);}

public void insertLesson_group(int lessonID, int groupID){
    const string command = "INSERT INTO lesson_group (lesson_id, group_id) "
    + "VALUES (@LESSON_ID, @GROUP_ID)";
    MySqlConnection.Open();
    MySqlCommand = new MySqlCommand(command, MySqlConnection);
    MySqlCommand.Parameters.AddWithValue("@LESSON_ID", lessonID);
    MySqlCommand.Parameters.AddWithValue("@GROUP_ID", groupID);
    MySqlCommand.ExecuteNonQuery();
    MySqlConnection.Close();}

public ArrayList getAuditory_types(){
    const string command = "SELECT auditory_type_name FROM auditory_type";
    return getArrayList(command); }

public void insertAuditory_type(string auditory_type){
    const string command = "INSERT INTO auditory_type (auditory_type_name) VALUES (@TYPE)";
    MySqlConnection.Open();
    MySqlCommand = new MySqlCommand(command, MySqlConnection);
    MySqlCommand.Parameters.AddWithValue("@TYPE", auditory_type);
    MySqlCommand.ExecuteNonQuery();
    MySqlConnection.Close(); }

public ArrayList getDisciplines(){
    const string command = "SELECT full_name FROM discipline";
    return getArrayList(command);}

public void insertDiscipline(string discipline){
    const string command = "INSERT INTO discipline (full_name) VALUES (@FULL_NAME)";
    MySqlConnection.Open();
    MySqlCommand = new MySqlCommand(command, MySqlConnection);
    MySqlCommand.Parameters.AddWithValue("@FULL_NAME", discipline);
    MySqlCommand.ExecuteNonQuery();

```

```

        mySqlConnection.Close();}

public ArrayList getFacultyNames(){
    const string command = "SELECT full_name FROM faculty";
    return getArrayList(command); }

public ArrayList getFacultyCodes(){
    const string command = "SELECT faculty_code FROM faculty";
    return getArrayList(command);}

public void insertFaculty(string name, string code){
    const string command = "INSERT INTO faculty (full_name, faculty_code) VALUES (@FULL_NAME,
@CODE)";
    mySqlConnection.Open();
    mySqlCommand = new MySqlCommand(command, mySqlConnection);
    mySqlCommand.Parameters.AddWithValue("@FULL_NAME", name);
    mySqlCommand.Parameters.AddWithValue("@CODE", code);
    mySqlCommand.ExecuteNonQuery();
    mySqlConnection.Close(); }

public int getFacultyID(string code){
    const string command = "SELECT faculty_id FROM faculty WHERE faculty_code = @CODE";
    return getID(command, "@CODE", code);}

public void insertDepartment(int facultyID, string fullName, string shortName){
    const string command ="INSERT INTO department (faculty_id, full_name, short_name) "
+ "VALUES (@FACULTY_ID, @FULL_NAME, @SHORT_NAME)";
    mySqlConnection.Open();
    mySqlCommand = new MySqlCommand(command, mySqlConnection);
    mySqlCommand.Parameters.AddWithValue("@FACULTY_ID", facultyID);
    mySqlCommand.Parameters.AddWithValue("@FULL_NAME", fullName);
    mySqlCommand.Parameters.AddWithValue("@SHORT_NAME", shortName);
    mySqlCommand.ExecuteNonQuery();
    mySqlConnection.Close();}

public int getDepartmentIDbyFullName(string departmentName){
    const string command = "SELECT department_id FROM department WHERE full_name =
@DEPARTMENT";
    return getID(command, "@DEPARTMENT", departmentName);}

public void insertTeacher(int departmentID, string name, string sex, string post, string status){
    const string command = "INSERT INTO teacher (department_id, full_name, sex, post, status) VALUES(@ID,
@NAME, @SEX, @POST, @STATUS)";
    mySqlConnection.Open();
    mySqlCommand = new MySqlCommand(command, mySqlConnection);
    mySqlCommand.Parameters.AddWithValue("@ID", departmentID);
    mySqlCommand.Parameters.AddWithValue("@NAME", name);
    mySqlCommand.Parameters.AddWithValue("@SEX", sex);
    mySqlCommand.Parameters.AddWithValue("@POST", post);

```

```

mySqlCommand.Parameters.AddWithValue("@STATUS", status);
mySqlCommand.ExecuteNonQuery();
mySqlConnection.Close();}

public void correctTeacherTable(){
    const string createTemporaryTable = "CREATE TEMPORARY TABLE teacher2 AS (SELECT * FROM
teacher GROUP BY department_id, full_name)";
    mySqlConnection.Open();
    mySqlCommand = new MySqlCommand(createTemporaryTable, mySqlConnection);
    mySqlCommand.ExecuteNonQuery();
    const string deleteTrash = "DELETE FROM teacher WHERE teacher.teacher_id NOT IN (SELECT
teacher2.teacher_id FROM teacher2)";
    mySqlCommand = new MySqlCommand(deleteTrash, mySqlConnection);
    mySqlCommand.ExecuteNonQuery();
    mySqlConnection.Close(); }

public int getAuditoryTypeID(string type) {
    const string command = "SELECT auditory_type_id FROM auditory_type WHERE auditory_type_name =
@TYPE";
    return getID(command, "@TYPE", type); }

public void insertAuditory(int departmentID, string name, bool not_used, int auditoryTypeID, int count, int
corpsNumber) {
    const string command = "INSERT INTO auditory (department_id, auditory_name, not_used, type_auditory,
count_of_places, corps_number) " +
        "VALUES(@ID, @AUDITORY_NAME, @NOT_USED, @TYPE_ID, @COUNT, @CORPS_NUMBER)";
    mySqlConnection.Open();
    mySqlCommand = new MySqlCommand(command, mySqlConnection);
    mySqlCommand.Parameters.AddWithValue("@ID", departmentID);
    mySqlCommand.Parameters.AddWithValue("@AUDITORY_NAME", name);
    mySqlCommand.Parameters.AddWithValue("@NOT_USED", not_used);
    mySqlCommand.Parameters.AddWithValue("@TYPE_ID", auditoryTypeID);
    mySqlCommand.Parameters.AddWithValue("@COUNT", count);
    mySqlCommand.Parameters.AddWithValue("@CORPS_NUMBER", corpsNumber);
    mySqlCommand.ExecuteNonQuery();
    mySqlConnection.Close(); }

public void insertAuditory(int departmentID, string name) {
    const string command = "INSERT INTO auditory (department_id, auditory_name) " +
        "VALUES(@ID, @AUDITORY_NAME)";
    mySqlConnection.Open();
    mySqlCommand = new MySqlCommand(command, mySqlConnection);
    mySqlCommand.Parameters.AddWithValue("@ID", departmentID);
    mySqlCommand.Parameters.AddWithValue("@AUDITORY_NAME", name);
    mySqlCommand.ExecuteNonQuery();
    mySqlConnection.Close(); }

```

```

public void insertFullDataStudy_group(int departmentID, string code, string name, string course, int count) {
    const string command = "INSERT INTO study_group (department_id, study_group_code, full_name," +
        " course_number, count_of_students) VALUES(@ID, @CODE, @NAME, @COURSE, @COUNT)";
    MySqlConnection.Open();
    MySqlCommand = new MySqlCommand(command, MySqlConnection);
    MySqlCommand.Parameters.AddWithValue("@ID", departmentID);
    MySqlCommand.Parameters.AddWithValue("@CODE", code);
    MySqlCommand.Parameters.AddWithValue("@NAME", name);
    MySqlCommand.Parameters.AddWithValue("@COURSE", course);
    MySqlCommand.Parameters.AddWithValue("@COUNT", count);
    MySqlCommand.ExecuteNonQuery();
    MySqlConnection.Close(); }

public ArrayList getDepartmentNames() {
    ArrayList departmentNames = new ArrayList();
    MySqlConnection.Open();
    const string command = "SELECT full_name FROM department WHERE department_id < 60";
    MySqlCommand = new MySqlCommand (command, MySqlConnection);
    using(mySqlDataReader = MySqlCommand.ExecuteReader()) {
        while(mySqlDataReader.Read()) {
            departmentNames.Add(mySqlDataReader[0].ToString()); }
    MySqlConnection.Close();
    return departmentNames; }

public ArrayList getTeachers(){
    const string command = "SELECT full_name FROM teacher";
    return getArrayList(command); }

public ArrayList getAuditoryNames(){
    const string command = "SELECT auditory_name FROM auditory";
    return getArrayList(command); }

public void insertLesson_time(int lessonID, string day){
    const string command = "INSERT INTO lesson_time (lesson_id, day_of_week) "
        + "VALUES (@LESSON_ID, @DAY)";
    MySqlConnection.Open();
    MySqlCommand = new MySqlCommand(command, MySqlConnection);
    MySqlCommand.Parameters.AddWithValue("@LESSON_ID", lessonID);
    MySqlCommand.Parameters.AddWithValue("@DAY", day);
    MySqlCommand.ExecuteNonQuery();
    MySqlConnection.Close();}

public void deleteExcessData(int departmentID){
    deleteExcessInLesson_Teacher(departmentID);
    deleteExcessInLesson_Group(departmentID);
    deleteExcessInLesson_Auditory(departmentID);
    deleteExcessInLesson_Time(departmentID);

```

```

        deleteExcessInLesson(departmentID); }

public void deleteExcessInLesson_Teacher(int departmentID){
    const string command = "DELETE FROM lesson_teacher WHERE lesson_id IN (SELECT lesson_id FROM
lesson WHERE department_id = @DEPARTMENT)";
    MySqlConnection.Open();
    MySqlCommand = new MySqlCommand(command, MySqlConnection);
    MySqlCommand.Parameters.AddWithValue("@DEPARTMENT", departmentID);
    MySqlCommand.ExecuteNonQuery();
    MySqlConnection.Close();}

public void deleteExcessInLesson_Group(int departmentID){
    const string command = "DELETE FROM lesson_group WHERE lesson_id IN (SELECT lesson_id FROM
lesson WHERE department_id = @DEPARTMENT)";
    MySqlConnection.Open();
    MySqlCommand = new MySqlCommand(command, MySqlConnection);
    MySqlCommand.Parameters.AddWithValue("@DEPARTMENT", departmentID);
    MySqlCommand.ExecuteNonQuery();
    MySqlConnection.Close(); }

public void deleteExcessInLesson_Auditory(int departmentID){
    const string command = "DELETE FROM lesson_auditory WHERE lesson_id IN (SELECT lesson_id FROM
lesson WHERE department_id = @DEPARTMENT)";
    MySqlConnection.Open();
    MySqlCommand = new MySqlCommand(command, MySqlConnection);
    MySqlCommand.Parameters.AddWithValue("@DEPARTMENT", departmentID);
    MySqlCommand.ExecuteNonQuery();
    MySqlConnection.Close(); }

public void deleteExcessInLesson_Time(int departmentID){
    const string command = "DELETE FROM lesson_time WHERE lesson_id IN (SELECT lesson_id FROM
lesson WHERE department_id = @DEPARTMENT)";
    MySqlConnection.Open();
    MySqlCommand = new MySqlCommand(command, MySqlConnection);
    MySqlCommand.Parameters.AddWithValue("@DEPARTMENT", departmentID);
    MySqlCommand.ExecuteNonQuery();
    MySqlConnection.Close(); }

public void deleteExcessInLesson(int departmentID){
    const string command = "DELETE FROM lesson WHERE department_id = @DEPARTMENT";
    MySqlConnection.Open();
    MySqlCommand = new MySqlCommand(command, MySqlConnection);
    MySqlCommand.Parameters.AddWithValue("@DEPARTMENT", departmentID);
    MySqlCommand.ExecuteNonQuery();
    MySqlConnection.Close(); } } }

```

Б.8 Файл Departments.cs

```

using System;
using System.Collections;
using System.Collections.Generic;
using System.IO;
using System.Windows;
namespace DataCollectionApp{
    public class Departments : ExcelFile{
        ArrayList full_names = new ArrayList();
        ArrayList short_names = new ArrayList();
        ArrayList faculty_codes = new ArrayList();
        ArrayList missingValuesOfFullNames = new ArrayList();
        ArrayList missingValuesOfShortNames = new ArrayList();
        ArrayList missingValuesOfFacultyCodes = new ArrayList();
        Dictionary <int, string> duplicatesOfFullNames = new Dictionary<int, string>();
        Dictionary <int, string> duplicatesOfShortNames = new Dictionary<int, string>();
        int row = 1;
        const char fullNamesColumn = 'A';
        const char shortNamesColumn = 'B';
        const char facultyCodesColumn = 'C';
        bool reading = true;
        DbOperations dbo = new DbOperations();
        public Departments(string fileName): base(fileName)
        { this.fileName = fileName; }
        public override void readFromExcelFile(){
            try{
                open(1);
                for(int col = getColumnNumber(fullNamesColumn), i = row; i <= rowsCount; i++){
                    cellContent = getCellContent(i, col);
                    if(full_names.Contains(cellContent))
                        { duplicatesOfFullNames.Add(i, cellContent); }
                    else
                        { full_names.Add(cellContent); }
                    if(string.IsNullOrEmpty(cellContent))
                        { missingValuesOfFullNames.Add(i); } }
                for(int col = getColumnNumber(shortNamesColumn), i = row; i <= rowsCount; i++){
                    cellContent = getCellContent(i, col);
                    if(short_names.Contains(cellContent))
                        { duplicatesOfShortNames.Add(i, cellContent); }

```

```

else
{ short_names.Add(cellContent); }
if(string.IsNullOrEmpty(cellContent))
{ missingValuesOfShortNames.Add(i); } }
for(int col = getColumnNumber(facultyCodesColumn), i = row; i <= rowsCount; i++){
cellContent = getCellContent(i, col);
if(string.IsNullOrEmpty(cellContent))
{ missingValuesOfFacultyCodes.Add(i); }
else
{ faculty_codes.Add(cellContent); } }
close(); }
catch (Exception ex){
reading = false;
MessageBox.Show("Помилка при отриманні даних з файлу " + FileName + " " + ex.Message); } }
public override void evaluateData() {
if(reading) {
const string path = @"E:\BACHELORS
WORK\TIMETABLE\DataCollectionApp\BugsReport.txt";
if (missingValuesOfFullNames.Count !=0 || missingValuesOfShortNames.Count != 0
|| missingValuesOfFacultyCodes.Count != 0 || duplicatesOfFullNames.Count != 0
|| duplicatesOfShortNames.Count != 0) {
using (StreamWriter sw = new StreamWriter(path, true, System.Text.Encoding.Default)){
sw.WriteLine("{0:g}", DateTime.Now);
sw.WriteLine("КАФЕДРИ.");
sw.WriteLine("Файл: " + FileName); } }
if(missingValuesOfFullNames.Count != 0) {
using (StreamWriter sw = new StreamWriter(path, true, System.Text.Encoding.Default)) {
sw.WriteLine("Пропущено назви кафедр в рядках: ");
foreach (int value in missingValuesOfFullNames)
{ sw.Write(value + "|"); }
sw.WriteLine(); } }
if(missingValuesOfShortNames.Count != 0) {
using (StreamWriter sw = new StreamWriter(path, true, System.Text.Encoding.Default)) {
sw.WriteLine("Пропущено скорочені назви кафедр в рядках: ");
foreach (int value in missingValuesOfShortNames)
{ sw.Write(value + "|"); }
sw.WriteLine(); } }
if(missingValuesOfFacultyCodes.Count != 0) {
using (StreamWriter sw = new StreamWriter(path, true, System.Text.Encoding.Default)) {
sw.WriteLine("Пропущено коди факультетів в рядках: ");
foreach (int value in missingValuesOfFacultyCodes)
{ sw.Write(value + "|"); }

```

```

        sw.WriteLine(); } }
    if(duplicatesOfFullNames.Count != 0) {
        using (StreamWriter sw = new StreamWriter(path, true, System.Text.Encoding.Default)) {
            sw.WriteLine("Є дублікати назв кафедр: ");
            foreach (KeyValuePair<int, string> duplicate in duplicatesOfFullNames)
                { sw.WriteLine("В рядку номер " + duplicate.Key + ": " + duplicate.Value); }
            sw.WriteLine(); } }
    if(duplicatesOfShortNames.Count != 0){
        using (StreamWriter sw = new StreamWriter(path, true, System.Text.Encoding.Default)){
            sw.WriteLine("Є дублікати скорочених назв кафедр: ");
            foreach (KeyValuePair<int, string> duplicate in duplicatesOfShortNames)
                { sw.WriteLine("В рядку номер " + duplicate.Key + ": " + duplicate.Value); }
            sw.WriteLine(); } }
    bool noSensetoReload = true;
    ArrayList departmentsInDB = dbo.getDepartmentNames();
    foreach(string name in full_names){
        if(!departmentsInDB.Contains(name)){
            noSensetoReload = false;
            break; } }
    if (noSensetoReload){
        reading = false;
        MessageBox.Show("Дані про кафедри вже містяться в базі даних!"); } } }
    public override void load(){
        if(reading){
            try{
                for(int i = 0; i < faculty_codes.Count; i++){
                    int facultyID = dbo.getFacultyID(faculty_codes[i].ToString());
                    dbo.insertDepartment(facultyID, full_names[i].ToString(), short_names[i].ToString()); }
                MessageBox.Show("Дані про кафедри завантажено до бази даних!"); }
            catch(Exception ex){
                MessageBox.Show("Виникла помилка під час завантаження даних про кафедри з файлу " + FileName
                    + " до бази даних!"); } } } } }

```

Б.9 Файл Disciplines.cs

```

using System;
using System.Collections;
using System.Collections.Generic;
using System.IO;
using System.Windows;
namespace DataCollectionApp{

```

```

public class Disciplines : ExcelFile{
    ArrayList records = new ArrayList();
    ArrayList missingValues = new ArrayList();
    Dictionary <int, string> duplicates = new Dictionary<int, string>();
    int row = 2;
    const char column = 'G';
    bool reading = true;
    DbOperations dbo = new DbOperations();
    public Disciplines(string fileName): base(fileName)
    { this.fileName = fileName; }
    public override void readFromExcelFile(){
        try{
            open(1);
            for(int col = getColumnNumber(column), i = row; i <= rowsCount; i++){
                cellContent = getCellContent(i, col);
                if(records.Contains(cellContent))
                { duplicates.Add(i, cellContent); }
                else
                { records.Add(cellContent); }
                if(string.IsNullOrEmpty(cellContent))
                { missingValues.Add(i); } }
            close(); }
        catch(Exception ex) {
            reading = false;
            MessageBox.Show("Помилка при отриманні даних з файлу " + FileName + ". " + ex.Message); } }
    public override void evaluateData(){
        if (reading){
            const string path = @"E:\BACHELORS WORK\TIMETABLE\DataCollectionApp\BugsReport.txt";
            if(missingValues.Count !=0 || duplicates.Count !=0){
                using (StreamWriter sw = new StreamWriter(path, true, System.Text.Encoding.Default)){
                    sw.WriteLine("{0:g}", DateTime.Now);
                    sw.WriteLine("ДИСЦИПЛІНИ.");
                    sw.WriteLine("Файл: " + FileName);} }
            if (missingValues.Count != 0){
                using (StreamWriter sw = new StreamWriter(path, true, System.Text.Encoding.Default)){
                    sw.WriteLine("Є пропуски в рядках: ");
                    foreach (int value in missingValues)
                    { sw.Write(value + "|"); }
                    sw.WriteLine();}}
            if(duplicates.Count != 0){
                using (StreamWriter sw = new StreamWriter(path, true, System.Text.Encoding.Default)){
                    sw.WriteLine("Є дублікати: ");

```

```

        foreach (KeyValuePair<int, string> duplicate in duplicates)
        { sw.WriteLine("В рядку номер " + duplicate.Key + ": " + duplicate.Value); }
        sw.WriteLine(); } }

bool noSensetoReload = true;
ArrayList disciplinesInDB = dbo.getDisciplines();
foreach (string record in records){
    if(!disciplinesInDB.Contains(record)){
        noSensetoReload = false;
        break;} }

if (noSensetoReload){
    reading = false;
    MessageBox.Show("Дані про дисципліни вже містяться в базі даних!"); } } }

public override void load(){
    if(reading){
        try{
            foreach (string record in records){
                dbo.insertDiscipline(record);}

            MessageBox.Show("Дані про дисципліни завантажено до бази даних!"); }

            catch(Exception ex){
                MessageBox.Show("Виникла помилка під час завантаження даних про дисципліни з файлу " +
FileName + " до бази даних!"); } } } }

```

Б.10 Файл Faculties.cs

```

using System;
using System.Collections;
using System.Collections.Generic;
using System.IO;
using System.Windows;

namespace DataCollectionApp{
    public class Faculties : ExcelFile{
        ArrayList names = new ArrayList();
        ArrayList codes = new ArrayList();
        ArrayList missingValuesOfNames = new ArrayList();
        ArrayList missingValuesOfCodes = new ArrayList();
        Dictionary <int, string> duplicatesOfNames = new Dictionary<int, string>();
        Dictionary <int, string> duplicatesOfCodes = new Dictionary<int, string>();
        int row = 2;
        const char namesColumn = 'B';
        const char codesColumn = 'D';
        bool reading = true;

```

```

DbOperations dbo = new DbOperations();
public Faculties(string fileName): base(fileName)
{ this.fileName = fileName; }
public override void readFromExcelFile(){
    try{
        open(1);
        for(int col = getColumnNumber(namesColumn), i = row; i <= rowsCount; i++){
            cellContent = getCellContent(i, col);
            if(names.Contains(cellContent))
            { duplicatesOfNames.Add(i, cellContent); }
            else
            { names.Add(cellContent); }
            if(string.IsNullOrEmpty(cellContent))
            { missingValuesOfNames.Add(i); }
        }
        for(int col = getColumnNumber(codesColumn), i = row; i <= rowsCount; i++){
            cellContent = getCellContent(i, col);
            if(codes.Contains(cellContent))
            { duplicatesOfCodes.Add(i, cellContent); }
            else
            { codes.Add(cellContent); }
            if(string.IsNullOrEmpty(cellContent))
            { missingValuesOfCodes.Add(i); } }
        close();}
    catch (Exception ex){
        reading = false;
        MessageBox.Show("Помилка при отриманні даних з файлу " + FileName + " " + ex.Message);}
    public override void evaluateData(){
        if(reading){
            const string path = @"E:\BACHELORS WORK\TIMETABLE\DataCollectionApp\BugsReport.txt";
            if(missingValuesOfNames.Count != 0 || duplicatesOfNames.Count != 0
            || missingValuesOfCodes.Count != 0 || duplicatesOfCodes.Count != 0){
                using (StreamWriter sw = new StreamWriter(path, true, System.Text.Encoding.Default)){
                    sw.WriteLine("{0:g}", DateTime.Now);
                    sw.WriteLine("ФАКУЛЬТЕТИ.");
                    sw.WriteLine("Файл: " + FileName); }
            }
            if(missingValuesOfNames.Count != 0){
                using (StreamWriter sw = new StreamWriter(path, true, System.Text.Encoding.Default)){
                    sw.WriteLine("Пропущено назви факультетів в рядках: ");
                    foreach (int value in missingValuesOfNames)
                    { sw.Write(value + "|"); }
                    sw.WriteLine();}
            }
            if(duplicatesOfNames.Count != 0){

```


Б.11 Файл GroupCode.cs

```
using System;
namespace DataCollectionApp{
    public class GroupCode{
        public string getGroupCode(string study_group){
            int hyphen = study_group.IndexOf('-');
            string code = study_group.Substring(0, hyphen + 2);
            return code; }}}

```

Б.12 Файл Statements.cs

```
using System;
using System.Collections;
using System.Collections.Generic;
using Excel = Microsoft.Office.Interop.Excel;
using MySql.Data.MySqlClient;
using System.Windows;
using System.IO;
namespace DataCollectionApp{
    public class Statements : ExcelFile{
        int firstRow;
        public int lastRow;
        ArrayList disciplines = new ArrayList();
        ArrayList groups = new ArrayList();
        ArrayList lessonsType = new ArrayList();
        ArrayList hours = new ArrayList();
        ArrayList lessonsControl = new ArrayList();
        ArrayList teachers = new ArrayList();
        ArrayList auditories = new ArrayList();
        ArrayList days = new ArrayList();
        const char disciplinesColumn = 'B';
        const char groupsColumn = 'C';
        const char typesColumn = 'D';
        const char hoursColumn = 'F';
        const char controlColumn = 'G';
        const char teachersColumn = 'H';
        const char auditoriesColumn = 'I';
    }
}

```

```

const char daysColumn = 'J';
int sheetNumber;
string departmentShortName;
bool reading = true;
public Statements(string fileName, int firstRow, int sheetNumber, string departmentShortName): base(fileName){
    this.fileName = fileName;
    this.firstRow = firstRow;
    this.sheetNumber = sheetNumber;
    this.departmentShortName = departmentShortName;
    this.lastRow = getLastRow(fileName, sheetNumber, firstRow); }
DbOperations dbo = new DbOperations();
CellContentCorrection ccc = new CellContentCorrection();
public override void readFromExcelFile(){
    try{
        open(sheetNumber);
        for(int col = getColumnNumber(disciplinesColumn), i = firstRow; i <= lastRow; i++){
            cellContent = getCellContent(i, col);
            disciplines.Add(cellContent); }
        for(int col = getColumnNumber(groupsColumn), i = firstRow; i <= lastRow; i++){
            cellContent = getCellContent(i, col);
            groups.Add(cellContent); }
        for(int col = getColumnNumber(typesColumn), i = firstRow; i <= lastRow; i++){
            cellContent = getCellContent(i, col);
            lessonsType.Add(cellContent); }
        for(int col = getColumnNumber(hoursColumn), i = firstRow; i <= lastRow; i++){
            cellContent = getCellContent(i, col);
            if (string.IsNullOrEmpty(cellContent)){
                hours.Add(null);}
            else{
                char comma = ',';
                int index = cellContent.IndexOf(comma);
                if(index != -1)
                    { cellContent = cellContent.Substring(0, cellContent.Length - index - 1); }
                double h = Convert.ToDouble(cellContent);
                hours.Add(h);} }
        for(int col = getColumnNumber(controlColumn), i = firstRow; i <= lastRow; i++){
            cellContent = getCellContent(i, col);
            lessonsControl.Add(cellContent);}
        for(int col = getColumnNumber(teachersColumn), i = firstRow; i <= lastRow; i++){
            cellContent = getCellContent(i, col);
            teachers.Add(cellContent);}
        for(int col = getColumnNumber(auditoriesColumn), i = firstRow; i <= lastRow; i++){

```

```

        cellContent = getCellContent(i, col);
        auditories.Add(cellContent);}
    for(int col = getColumnNumber(daysColumn), i = firstRow; i <= lastRow; i++){
        cellContent = getCellContent(i, col);
        days.Add(cellContent);}
    close(); }
    catch (Exception ex) {
        reading = false;
        MessageBox.Show("Помилка при отриманні даних з файлу " + FileName + " " + ex.Message);} }
    public override void evaluateData(){
        int departmentID = dbo.getDepartmentID(departmentShortName);
        if (sheetsCount == 1){
            dbo.deleteExcessData(departmentID); }}
    public override void load(){
        if(reading){
            try{
                const string path = @"E:\BACHELORS WORK\TIMETABLE\DataCollectionApp\BugsReport.txt";
                int departmentID = dbo.getDepartmentID(departmentShortName);
                for(int i = 0; i < disciplines.Count; i++){
                    int disciplineID = dbo.getDisciplineID(disciplines[i].ToString());
                    dbo.insertLesson(disciplineID, lessonsType[i].ToString(), Convert.ToInt32(hours[i]),
                        lessonsControl[i].ToString(), departmentID);
                    int lastLessonID = dbo.getLastLessonID();
                    string suggestedAuditories = auditories[i].ToString();
                    if(suggestedAuditories.Length != 0){
                        suggestedAuditories = ccc.correctAuditories(suggestedAuditories);
                        string [] separatedAuditories = suggestedAuditories.Split(new char[] {',', ';'});
                        for (int j = 0; j < separatedAuditories.Length; j++){
                            ArrayList auditoriesInDB = dbo.getAuditoryNames();
                            if(auditoriesInDB.Contains(separatedAuditories[j]) == false){
                                dbo.insertAuditory(departmentID, separatedAuditories[j]);
                                using (StreamWriter sw = new StreamWriter(path, true, System.Text.Encoding.Default)){
                                    sw.WriteLine("{0:g}", DateTime.Now);
                                    sw.Write("Аудиторію \"\" + separatedAuditories[j] + "\" кафедри \"\" + departmentShortName
+ "\" завантажено до бази даних з файлу \"\" + FileName + "\"");
                                    sw.WriteLine(); } }
                                int auditoryID = dbo.getAuditoryID(separatedAuditories[j]);
                                dbo.insertLesson_Auditory(lastLessonID, auditoryID);} }
                    string teachersRecord = teachers[i].ToString();
                    teachersRecord = ccc.correctTeachers(teachersRecord);
                    string [] separatedTeachers = teachersRecord.Split(new char[] {',', ';'});
                    for(int j = 0; j < separatedTeachers.Length; j++){

```

```

        separatedTeachers[j] = separatedTeachers[j].Trim();
        int teacherID = dbo.getTeacherID(separatedTeachers[j]);
        dbo.insertLesson_Teacher(lastLessonID, teacherID);}
    string groupsInCell = groups[i].ToString();
    if(groupsInCell.Length != 0){
        groupsInCell = ccc.correctGroups(groupsInCell);
        string [] separatedGroups = groupsInCell.Split(new char[] {','});
        ArrayList studyGroups = dbo.getStudy_groups();
        for(int j = 0; j < separatedGroups.Length; j++){
            if (studyGroups.Contains(separatedGroups[j]) == false){
                GroupCode gc = new GroupCode();
                string code = gc.getGroupCode(separatedGroups[j]);
                dbo.insertStudy_group(departmentID, separatedGroups[j], code);
                using (StreamWriter sw = new StreamWriter(path, true, System.Text.Encoding.Default)){
                    sw.WriteLine("{0:g}", DateTime.Now);
                    sw.Write("Учбову групу \"" + separatedGroups[j] + "\" кафедри \"" + departmentShortName
+ "\" завантажено до бази даних з файлу \"" + FileName + "\"");
                    sw.WriteLine(); } }
                int groupID = dbo.getStudy_groupID(separatedGroups[j]);
                dbo.insertLesson_group(lastLessonID, groupID); } }
    string daysInCell = days[i].ToString();
    if(daysInCell.Length != 0){
        daysInCell = ccc.correctTime(daysInCell);
        string [] separatedDays = daysInCell.Split(new char[] {','});
        for(int j = 0; j < separatedDays.Length; j++){
            dbo.insertLesson_time(lastLessonID, separatedDays[j]); } }
    catch(Exception ex){
        MessageBox.Show("Виникла помилка під час завантаження відомостей доручень до бази даних!" +
        "\n" + ex.Message); } } }

```

Б.13 Файл StudyGroups.cs

```

using System;
using System.Collections;
using System.Collections.Generic;
using System.IO;
using System.Windows;
namespace DataCollectionApp{
    public class StudyGroups : ExcelFile{
        public StudyGroups(string fileName): base(fileName)
        { this.fileName = fileName; }
    }
}

```

```

ArrayList departments = new ArrayList();
ArrayList names = new ArrayList();
ArrayList courseNumbers = new ArrayList();
ArrayList countOfStudents = new ArrayList();
ArrayList codes = new ArrayList();
ArrayList missingValuesOfDepartments = new ArrayList();
ArrayList missingValuesOfNames = new ArrayList();
Dictionary <int, string> duplicatesOfNames = new Dictionary<int, string>();
const char departmentsColumn = 'A';
const char namesColumn = 'B';
const char countOfStudentsColumn = 'C';
const char courseNumbersColumn = 'D';
bool reading = true;
int row = 1;
DbOperations dbo = new DbOperations();
public override void readFromExcelFile(){
    try{
        open(1);
        for(int col = getColumnNumber(departmentsColumn), i = row; i <= rowsCount; i++){
            cellContent = getCellContent(i, col);
            if(string.IsNullOrEmpty(cellContent))
                { missingValuesOfDepartments.Add(i); }
            else{
                cellContent = cellContent.Replace("*", "");
                departments.Add(cellContent); } }
        for(int col = getColumnNumber(namesColumn), i = row; i <= rowsCount; i++){
            cellContent = getCellContent(i, col);
            if(string.IsNullOrEmpty(cellContent))
                { missingValuesOfNames.Add(i); }
            else{
                if(names.Contains(cellContent))
                    { duplicatesOfNames.Add(i, cellContent); }
                names.Add(cellContent);
                int hyphen = cellContent.IndexOf('-');
                codes.Add(cellContent.Substring(0, hyphen + 2)); } }
        for(int col = getColumnNumber(courseNumbersColumn), i = row; i <= rowsCount; i++){
            cellContent = getCellContent(i, col);
            courseNumbers.Add(cellContent); }
        for(int col = getColumnNumber(countOfStudentsColumn), i = row; i <= rowsCount; i++){
            cellContent = getCellContent(i, col);
            if(string.IsNullOrEmpty(cellContent))
                { countOfStudents.Add(null); }

```

```

else
    { countOfStudents.Add(Convert.ToInt32(cellContent)); } }
close();}
catch (Exception ex){
    reading = false;
    MessageBox.Show("Помилка при зчитуванні даних з файлу " + FileName + " " + ex.Message);} }
public override void evaluateData(){
    if(reading){
        const string path = @"E:\BACHELORS WORK\TIMETABLE\DataCollectionApp\BugsReport.txt";
        if(missingValuesOfNames.Count != 0 || missingValuesOfDepartments.Count != 0
            || duplicatesOfNames.Count != 0){
            using (StreamWriter sw = new StreamWriter(path, true, System.Text.Encoding.Default)){
                sw.WriteLine("{0:g}", DateTime.Now);
                sw.WriteLine("УЧБОБИ ГРУПИИ.");
                sw.WriteLine("Файл: " + FileName);} }
        if(missingValuesOfNames.Count != 0){
            using (StreamWriter sw = new StreamWriter(path, true, System.Text.Encoding.Default)){
                sw.WriteLine("Пропущено назви груп в рядках: ");
                foreach (int value in missingValuesOfNames)
                { sw.Write(value + "|"); }
                sw.WriteLine();} }
        if(missingValuesOfDepartments.Count != 0){
            using (StreamWriter sw = new StreamWriter(path, true, System.Text.Encoding.Default)){
                sw.WriteLine("Пропущено назви кафедр в рядках: ");
                foreach (int value in missingValuesOfDepartments)
                { sw.Write(value + "|"); }
                sw.WriteLine();} }
        if (duplicatesOfNames.Count != 0) {
            using (StreamWriter sw = new StreamWriter(path, true, System.Text.Encoding.Default)){
                sw.WriteLine("С дублікати назв груп: ");
                foreach (KeyValuePair<int, string> duplicate in duplicatesOfNames)
                { sw.WriteLine("В рядку номер " + duplicate.Key + ": " + duplicate.Value); }
                sw.WriteLine();} }
        bool noSensetoReload = true;
        ArrayList studyGroupsInDB = dbo.getStudy_groups();
        foreach(string name in names){
            if(!studyGroupsInDB.Contains(name)){
                noSensetoReload = false;
                break;}}
        if (noSensetoReload){
            reading = false;
            MessageBox.Show("Дані про аудиторії вже містяться в базі даних!");} } }

```

```

public override void load(){
    if(reading){
        try{
            for (int i = 0; i < rowCount; i++){
                int departmentID = dbo.getDepartmentIDbyFullName(departments[i].ToString());
                dbo.insertFullDataStudy_group(departmentID, codes[i].ToString(), names[i].ToString(),
                    courseNumbers[i].ToString(), Convert.ToInt32(countOfStudents[i])); }
                MessageBox.Show("Дані про учбові групи завантажено до бази даних!"); }
            catch (Exception ex){
                MessageBox.Show("Виникла помилка під час завантаження даних про учбові групи з файлу " +
                    FileName + " до бази даних!");}}}}

```

Б.14 Файл Teachers.cs

```

using System;
using System.Collections;
using System.Collections.Generic;
using System.IO;
using System.Windows;
namespace DataCollectionApp{
    public class Teachers : ExcelFile{
        ArrayList names = new ArrayList();
        ArrayList sex = new ArrayList();
        ArrayList posts = new ArrayList();
        ArrayList statuses = new ArrayList();
        ArrayList departments = new ArrayList();
        const char departmentsColumn = 'A';
        const char namesColumn = 'J';
        const char sexColumn = 'K';
        const char postsColumn = 'L';
        const char statusesColumn = 'M';
        int row = 8;
        ArrayList missingValuesOfNames = new ArrayList();
        ArrayList missingValuesOfPosts = new ArrayList();
        ArrayList missingValuesOfDepartments = new ArrayList();
        Dictionary <int, string> duplicatesOfNames = new Dictionary<int, string>();
        Dictionary <int, string> wrongValuesOfSex = new Dictionary<int, string>();
        bool reading = true;
        DbOperations dbo = new DbOperations();
        public Teachers(string fileName): base(fileName)
        { this.fileName = fileName; }
    }
}

```

```

public override void readFromExcelFile(){
    try{
        open(1);
        for(int col = getColumnNumber(departmentsColumn), i = row; i <= rowCount; i++){
            cellContent = getCellContent(i, col);
            if(string.IsNullOrEmpty(cellContent))
                { missingValuesOfDepartments.Add(i); }
            else
                { departments.Add(cellContent); }
        }
        for(int col = getColumnNumber(namesColumn), i = row; i <= rowCount; i++){
            cellContent = getCellContent(i, col);
            if(names.Contains(cellContent))
                { duplicatesOfNames.Add(i, cellContent); }
            if(string.IsNullOrEmpty(cellContent))
                { missingValuesOfNames.Add(i); }
            names.Add(cellContent);}
        for(int col = getColumnNumber(sexColumn), i = row; i <= rowCount; i++){
            cellContent = getCellContent(i, col);
            sex.Add(cellContent);
            if ( cellContent != "м" && cellContent != "ж")
                { wrongValuesOfSex.Add(i, cellContent); }}
        for(int col = getColumnNumber(postsColumn), i = row; i <= rowCount; i++){
            cellContent = getCellContent(i, col);
            posts.Add(cellContent);
            if(string.IsNullOrEmpty(cellContent))
                { missingValuesOfPosts.Add(i); }}
        for(int col = getColumnNumber(statusesColumn), i = row; i <= rowCount; i++){
            cellContent = getCellContent(i, col);
            statuses.Add(cellContent);}
        close();}
    catch (Exception ex){
        reading = false;
        MessageBox.Show("Помилка при отриманні даних з файлу " + FileName + " " + ex.Message);}
    public override void evaluateData(){
        if(reading){
            const string path = @"E:\BACHELORS WORK\TIMETABLE\DataCollectionApp\BugsReport.txt";
            if (missingValuesOfDepartments.Count != 0 || missingValuesOfNames.Count != 0 ||
                missingValuesOfPosts.Count !=0 || duplicatesOfNames.Count != 0 ||
                wrongValuesOfSex.Count != 0){
                using (StreamWriter sw = new StreamWriter(path, true, System.Text.Encoding.Default)){
                    sw.WriteLine("{0:g}", DateTime.Now);
                    sw.WriteLine("ВИКЛАДАЧІ.");
                }
            }
        }
    }
}

```

```

        sw.WriteLine("Файл: " + FileName);} }
if (missingValuesOfDepartments.Count != 0){
    using (StreamWriter sw = new StreamWriter(path, true, System.Text.Encoding.Default)){
        sw.WriteLine("Пропущено назви кафедр в рядках: ");
        foreach (int value in missingValuesOfDepartments)
            { sw.Write(value + "|"); }
        sw.WriteLine();} }
if (missingValuesOfNames.Count != 0){
    using (StreamWriter sw = new StreamWriter(path, true, System.Text.Encoding.Default)){
        sw.WriteLine("Пропущено ПІБ викладачів в рядках: ");
        foreach (int value in missingValuesOfNames)
            { sw.Write(value + "|"); }
        sw.WriteLine(); } }
if (missingValuesOfPosts.Count != 0){
    using (StreamWriter sw = new StreamWriter(path, true, System.Text.Encoding.Default)){
        sw.WriteLine("Пропущено посади викладачів в рядках: ");
        foreach (int value in missingValuesOfPosts)
            { sw.Write(value + "|"); }
        sw.WriteLine();} }
if(duplicatesOfNames.Count != 0){
    using (StreamWriter sw = new StreamWriter(path, true, System.Text.Encoding.Default)){
        sw.WriteLine("Є дублікати ПІБ викладачів: ");
        foreach (KeyValuePair<int, string> duplicate in duplicatesOfNames)
            { sw.WriteLine("В рядку номер " + duplicate.Key + ": " + duplicate.Value); }
        sw.WriteLine();} }
if(wrongValuesOfSex.Count != 0){
    using (StreamWriter sw = new StreamWriter(path, true, System.Text.Encoding.Default)){
        sw.WriteLine("Є некоректні значення статі викладачів: ");
        foreach (KeyValuePair<int, string> wrongValue in wrongValuesOfSex)
            { sw.WriteLine("В рядку номер " + wrongValue.Key + ": " + wrongValue.Value); }
        sw.WriteLine();} }
bool noSensetoReload = true;
ArrayList teachersInDB = dbo.getTeachers();
foreach(string name in names){
    if(!teachersInDB.Contains(name)){
        noSensetoReload = false;
        break;}}
if (noSensetoReload){
    reading = false;
    MessageBox.Show("Дані про викладачів вже містяться в базі даних!");} } }
public override void load(){
    if(reading){

```

```

try{
    for(int i = 0; i < rowCount - row + 1; i++){
        int departmentID = dbo.getDepartmentIDbyFullName(departments[i].ToString());
        dbo.insertTeacher(departmentID, names[i].ToString(), sex[i].ToString(),
            posts[i].ToString(), statuses[i].ToString());
        dbo.correctTeacherTable();
        MessageBox.Show("Дані про викладачів завантажено до бази даних!"); }
catch (Exception ex){
    MessageBox.Show("Виникла помилка під час завантаження даних про викладачів з файлу " +
        FileName + " до бази даних!" + "\n " + ex.Message);} } } }

```

Б.15 Файл DBMySQLUtils.cs

```

using System;
using MySql.Data.MySqlClient;
namespace DataCollectionApp{
    public static class DBMySQLUtils{
        public static MySqlConnection
        GetDBConnection(string host, int port, string db, string username, string password){
            String connectionString = "Server=" + host + ";Database=" + db + ";port=" + port + ";User Id=" + username
            + ";password=" + password;
            MySqlConnection connection = new MySqlConnection(connectionString);
            return connection; } } }

```

Б.16 Файл DBUtils.cs

```

using System;
using MySql.Data.MySqlClient;
namespace DataCollectionApp{
    public class DBUtils{
        public static MySqlConnection GetDBConnection(){
            string host = "127.0.0.1";
            int port = 3306;
            string db = "app_db";
            string username = "root";
            string password = "pbx93fq26";
            return DBMySQLUtils.GetDBConnection(host, port, db, username, password); } } }

```

Б.17 Файл Window1.xaml

```

<Window x:Class="DataCollectionApp.Window1"
    x:Name="MainWindow"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local = "clr-namespace:DataCollectionApp"
    Title="DataCollectionApp" Width="1500" Height="575"
    ResizeMode="NoResize" Background="#dfe7e8">
<Window.Resources>
    <Style TargetType="ListBox">
        <Setter Property="FontFamily" Value="Arial"/>
        <Setter Property="FontSize" Value="18"/>
        <Setter Property="HorizontalAlignment" Value="Stretch"/>
        <Setter Property="HorizontalContentAlignment" Value="Center"/>
    </Style>
    <Style TargetType="TextBlock">
        <Setter Property="FontFamily" Value="Arial"/>
        <Setter Property="HorizontalAlignment" Value="Center"></Setter>
        <Setter Property="FontSize" Value="20"/>
    </Style>
    <Style TargetType="Button">
        <Setter Property="FontSize" Value="20"/>
        <Setter Property="FontWeight" Value="Bold"/>
        <Setter Property="Padding" Value="20" />
    </Style>
    <Style x:Key="textBlockPadding1">
        <Setter Property="TextBlock.Padding" Value="20,15" />
    </Style>
    <Style x:Key="textBlockPadding2">
        <Setter Property="TextBlock.Padding" Value="0, 15, 0, 15" />
    </Style>
</Window.Resources>
<Grid x:Name="MainGrid">
    <StackPanel>
        <StackPanel Orientation="Horizontal" HorizontalAlignment="Stretch">
            <StackPanel Margin="10">
                <TextBlock Text="Загальні дані" FontWeight="Bold" Foreground="Brown"/>
                <ListBox x:Name="commonDataListBox" BorderBrush="DarkSlateBlue"
SelectionChanged="commonDataListBox_Changed">
                    <ListBox.ItemContainerStyle>
                        <Style TargetType="{x:Type ListBoxItem}">
                            <Style.Setters>
                                <Setter Property="BorderBrush" Value="DarkSlateBlue" />

```

```

        <Setter Property="Padding" Value="20,15" />
        <Setter Property="FontFamily" Value="Arial"/>
        <Setter Property="FontSize" Value="18"/>
        <Setter Property="Background" Value="Bisque"/>
    </Style.Setters>
    <Style.Triggers>
        <Trigger Property="IsSelected" Value="True">
            <Setter Property="FontSize" Value="20" />
        </Trigger>
        <Trigger Property="IsMouseOver" Value="True">
            <Setter Property="Cursor" Value="Hand"/>
            <Setter Property="FontSize" Value="20" />
        </Trigger>
    </Style.Triggers>
</Style>
</ListBox.ItemContainerStyle>
<ListBoxItem x:Name="lbi_auditoryTypes" Content="Типи аудиторій"/>
<ListBoxItem x:Name="lbi_disciplines" Content="Дисципліни"/>
<ListBoxItem x:Name="lbi_faculties" Content="Факультети"/>
<ListBoxItem x:Name="lbi_departments" Content="Кафедри"/>
<ListBoxItem x:Name="lbi_teachers" Content="Викладачі"/>
<ListBoxItem x:Name="lbi_auditories" Content="Аудиторії"/>
<ListBoxItem x:Name="lbi_groups" Content="Учбові групи"/>
</ListBox>
<Button x:Name="WatchButton" Content="Переглянути" ToolTip="Обраний файл буде відкрито в
режимі читання" Background="Yellow" Margin="0, 30, 0, 0" Click="WatchButton_Click"></Button>
</StackPanel>
<StackPanel Margin="0, 10, 10, 10">
    <TextBlock Text="Відомості доручень" FontWeight="Bold" Foreground="Black"/>
    <ScrollViewer CanContentScroll="True" Height="375">
        <ListBox x:Name="statementsListBox" BorderBrush="DarkSlateBlue"
SelectionChanged="statementsListBox_Changed">
            <ListBox.ItemContainerStyle>
                <Style TargetType="{x:Type ListBoxItem}">
                    <Style.Setters>
                        <Setter Property="BorderBrush" Value="DarkSlateBlue" />
                        <Setter Property="Padding" Value="20,15" />
                        <Setter Property="FontFamily" Value="Arial"/>
                        <Setter Property="FontSize" Value="18"/>
                        <Setter Property="Background" Value="Bisque"/>
                    </Style.Setters>
                    <Style.Triggers>

```

```

        <Trigger Property="IsSelected" Value="True">
            <Setter Property="FontSize" Value="20" />
        </Trigger>
        <Trigger Property="IsMouseOver" Value="True">
            <Setter Property="Cursor" Value="Hand"/>
            <Setter Property="FontSize" Value="20" />
        </Trigger>
    </Style.Triggers>
</Style>
</ListBox.ItemContainerStyle>
</ListBox>
</ScrollViewer>
<Button x:Name="loadButton" Content="Завантажити" ToolTip="Інформація з обраного файлу буде
завантажена до бази даних" Background="LightGreen" Margin="0, 30, 0, 0" Click="loadButton_Click"></Button>
</StackPanel>
<StackPanel HorizontalAlignment="Center" Margin="0, 10, 10, 10">
    <TextBlock Text="Обраний файл" HorizontalAlignment="Stretch" Margin="200,0,0,0"
    FontWeight="Bold" Foreground="Chocolate"/>
    <TextBlock Text="Назва:" FontWeight="Bold" FontSize="18" HorizontalAlignment="Left"
    Style="{StaticResource textBlockPadding2}"/>
    <TextBlock Text="" x:Name="selectedFile" Background="LightYellow" FontSize="18" Padding="20,15"
    HorizontalAlignment="Left"/>
    <TextBlock Text="Файл:" FontWeight="Bold" FontSize="18" HorizontalAlignment="Left"
    Style="{StaticResource textBlockPadding2}"/>
    <TextBlock Text="" x:Name="selectedFileName" Background="LightYellow" FontSize="18"
    HorizontalAlignment="Left" Style="{StaticResource textBlockPadding1}"/>
    <TextBlock Text="Останнє редагування:" FontWeight="Bold" FontSize="18"
    HorizontalAlignment="Left" Style="{StaticResource textBlockPadding2}"/>
    <TextBlock Text="" x:Name="lastWriteTime" Background="LightYellow" FontSize="18"
    HorizontalAlignment="Left" Style="{StaticResource textBlockPadding1}"/>
    <Button Content="Переглянути звіт про помилки в файлах" HorizontalAlignment="Left"
    ToolTip="Буде відкрито текстовий файл із записами про помилки в Excel-файлах з даними" Background="Red"
    Margin="0, 85, 0, 0" Click="WatchBugsReport"></Button>
</StackPanel>
</StackPanel>
</StackPanel>
</Grid>
</Window>

```

Б.18 Файл Window1.xaml.cs

```

using System;
using System.Collections;
using System.Collections.Generic;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Diagnostics;
namespace DataCollectionApp{
    public partial class Window1 : Window{
        const string auditoryTypesText = "Типи аудиторій";
        const string disciplinesText = "Дисципліни";
        const string facultiesText = "Факультети";
        const string auditoriesText = "Аудиторії";
        const string departmentsText = "Кафедри";
        const string teachersText = "Викладачі";
        const string groupsText = "Учбові групи";
        const string machinePartsText = "Деталей машин і підйомно-транспортних механізмів";
        const string mbtText = "Технології машинобудування";
        const string economyAndCustomsText = "Економіки та митної справи";
        const string economicalTheoryText = "Економічної теорії та підприємництва";
        const string electricalMachinesText = "Електричних машин";
        const string industrialEnergySupplyText = "Електропостачання промислових підприємств";
        const string computerSystemsAndNetworksText = "Комп'ютерних систем та мереж";
        const string marketingAndLogisticsText = "Маркетингу та логістики";
        const string internationalEconomicRelationsText = "Міжнародних економічних відносин";
        const string accountingAndAuditText = "Обліку і оподаткування";
        const string appliedMathematicsText = "Прикладної математики";
        const string computerSoftwareText = "Програмних засобів";
        const string psychologyText = "Соціальної роботи та психології";
        const string aviationEngineConstructionTechnologyText = "Технології авіаційних двигунів";
        const string tourismText = "Туристичного, готельного та ресторанного бізнесу";
        AuditoryTypes auditoryTypes = new AuditoryTypes("TypesOfAuditories.xlsx");
        Disciplines disciplines = new Disciplines("Disciplines.xlsx");
        Faculties faculties = new Faculties("Faculties.xlsx");
        Departments departments = new Departments("Departments.xlsx");
        Teachers teachers = new Teachers("Teachers.xlsx");
        Auditories auditories = new

```

```

Auditories("Auditories_недостающие_корпуса_из_путеводителя_ЗНТУ_мои_предположения.xls");
StudyGroups studyGroups = new StudyGroups("Групи 30.04.2021.xlsx");
Statements appliedMathematics = new Statements("Прикладна_математика_Форма 44 ПМ денна 2019-2020.xlsx", 15, 1, "ПМ");
Statements internationalEconomicRelations = new Statements("МІЖНАРОДНИХ ЕКОНОМІЧНИХ ВІДНОСИН денне 44 2020.xlsx", 15, 1, "МЄВ");
Statements machineParts = new Statements("VIDOMOST_DORUCHEN_2 сем_ДВ_ДМ і ПТМ.xlsx", 15, 1, "ДМіПТМ");
Statements mbt = new Statements("ВІДОМІСТЬ ДОРУЧЕНЬ ТМБ денне весна - 2020.xlsx", 15, 1, "ТМБ");
Statements economyAndCustoms_sheet1 = new Statements("Економіки та митної справи_Форма 44 ВІДОМІСТЬ ДОРУЧЕНЬ - 2020_ЕМС.xlsx", 15, 1, "ЕтаМС");
Statements economyAndCustoms_sheet2 = new Statements("Економіки та митної справи_Форма 44 ВІДОМІСТЬ ДОРУЧЕНЬ - 2020_ЕМС.xlsx", 15, 2, "ЕтаМС");
Statements economicalTheory = new Statements("ЕКОНОМІЧНОЇ ТЕОРІЇ ТА ПІДПРИЄМНИЦТВА_ВІДОМІСТЬ ДОРУЧЕНЬ - 2020.xlsx", 15, 1, "ЕТтаП");
Statements electricalMachines = new Statements("Електричних_машин-Форма 44 ВІД ДОРУЧЕНЬ-2020_кафЕМ_ден2 сем.xlsx", 15, 1, "ЕМ");
Statements industrialEnergySupply = new Statements("Електропостачання промислових підприємств_Форма 44 ЕПП - 2020д.xlsx", 15, 1, "ЕПП");
Statements computerSystemsAndNetworks_sheet1 = new Statements("КОМП_ЮТЕРНІ СИСТЕМИ ТА МЕРЕЖІ_ВІДОМІСТЬ ДОРУЧЕНЬ_19_20.xlsx", 15, 1, "КСтаМ");
Statements computerSystemsAndNetworks_sheet2 = new Statements("КОМП_ЮТЕРНІ СИСТЕМИ ТА МЕРЕЖІ_ВІДОМІСТЬ ДОРУЧЕНЬ_19_20.xlsx", 15, 2, "КСтаМ");
Statements marketingAndLogistics = new Statements("МАРКЕТИНГУ ТА ЛОГІСТИКИ_Відомість_денне_II_нова.xls", 15, 1, "Марк.та Лог.");
Statements accountingAndAudit_sheet1 = new Statements("Облік і оподаткування_ВІДОМІСТЬ ДОРУЧЕНЬ - 2020.xlsx", 15, 1, "ОіО");
Statements accountingAndAudit_sheet2 = new Statements("Облік і оподаткування_ВІДОМІСТЬ ДОРУЧЕНЬ - 2020.xlsx", 15, 2, "ОіО");
Statements psychology = new Statements("Форма 44 ВІДОМІСТЬ ДОРУЧЕНЬ - 2020 Денна Соціальна робота та психологія.xlsx", 15, 1, "СоцРтаП");
Statements aviationEngineConstructionTechnology = new Statements("Технологій авіаційних двигунів ВІДОМІСТЬ ДОРУЧЕНЬ - 2020 весна денна.xlsx", 15, 1, "ТАД");
Statements computerSoftware = new Statements("Програмних_засобів_26-12-19_Форма 44_ВІДОМІСТЬ ДОРУЧЕНЬ - 2020.xlsx", 15, 1, "ПЗ");
Statements tourism_sheet1 = new Statements("Туризм_Форма 44 денна заочна 2020.xlsx", 15, 1, "ТГтаРБ");
Statements tourism_sheet2 = new Statements("Туризм_Форма 44 денна заочна 2020.xlsx", 15, 2, "ТГтаРБ");
public Window1(){
    InitializeComponent();
    DbOperations dbo = new DbOperations();
    Dictionary<int, string> departments = dbo.getDepartments();

```

```

foreach (var d in departments){
    ListBoxItem lbi = new ListBoxItem();
    lbi.Content = d.Value;
    lbi.Uid = d.Key.ToString();
    statementsListBox.Items.Add(lbi);}}
void WatchButton_Click(object sender, RoutedEventArgs e){
    string fileName = "";
    ListBoxItem statementsFile = (ListBoxItem)statementsListBox.SelectedItem;
    ListBoxItem commonDataFile = (ListBoxItem)commonDataListBox.SelectedItem;
    if(statementsFile == null && commonDataFile != null){
        fileName = commonDataFile.Content.ToString();
        switch(fileName){
            case auditoryTypesText:
                auditoryTypes.openForViewing();
                break;
            case disciplinesText:
                disciplines.openForViewing();
                break;
            case facultiesText:
                faculties.openForViewing();
                break;
            case departmentsText:
                departments.openForViewing();
                break;
            case teachersText:
                teachers.openForViewing();
                break;
            case auditoriesText:
                auditories.openForViewing();
                break;
            case groupsText:
                studyGroups.openForViewing();
                break; }
        fileName = "";
        commonDataListBox.SelectedItem = null;}
    else{
        if(statementsFile != null && commonDataFile == null){
            fileName = statementsFile.Content.ToString();
            switch(fileName){
                case machinePartsText:
                    machineParts.openForViewing();
                    break;

```

```

case mbtText:
    mbt.openForViewing();
    break;
case economyAndCustomsText:
    economyAndCustoms_sheet1.openForViewing();
    economyAndCustoms_sheet2.openForViewing();
    break;
case economicalTheoryText:
    economicalTheory.openForViewing();
    break;
case electricalMachinesText:
    electricalMachines.openForViewing();
    break;
case industrialEnergySupplyText:
    industrialEnergySupply.openForViewing();
    break;
case computerSystemsAndNetworksText:
    computerSystemsAndNetworks_sheet1.openForViewing();
    computerSystemsAndNetworks_sheet2.openForViewing();
    break;
case marketingAndLogisticsText:
    marketingAndLogistics.openForViewing();
    break;
case internationalEconomicRelationsText:
    internationalEconomicRelations.openForViewing();
    break;
case accountingAndAuditText:
    accountingAndAudit_sheet1.openForViewing();
    accountingAndAudit_sheet2.openForViewing();
    break;
case appliedMathematicsText:
    appliedMathematics.openForViewing();
    break;
case computerSoftwareText:
    computerSoftware.openForViewing();
    break;
case psychologyText:
    psychology.openForViewing();
    break;
case aviationEngineConstructionTechnologyText:
    aviationEngineConstructionTechnology.openForViewing();
    break;

```

```

        case tourismText:
            tourism_sheet1.openForViewing();
            tourism_sheet2.openForViewing();
            break; }
        fileName = "";
        statementsListBox.SelectedItem = null;}
    else {
        if(statementsFile == null && commonDataFile == null)
            { MessageBox.Show("Не обрано файл для перегляду!"); } } }
void loadButton_Click(object sender, RoutedEventArgs e){
    string fileName = "";
    ListBoxItem statementsFile = (ListBoxItem)statementsListBox.SelectedItem;
    ListBoxItem commonDataFile = (ListBoxItem)commonDataListBox.SelectedItem;
    if(statementsFile == null && commonDataFile != null){
        fileName = commonDataFile.Content.ToString();
        switch(fileName){
            case auditoryTypesText:
                auditoryTypes.sendDataToDB();
                break;
            case disciplinesText:
                disciplines.sendDataToDB();
                break;
            case facultiesText:
                faculties.sendDataToDB();
                break;
            case departmentsText:
                departments.sendDataToDB();
                break;
            case teachersText:
                teachers.sendDataToDB();
                break;
            case auditoriesText:
                auditories.sendDataToDB();
                break;
            case groupsText:
                studyGroups.sendDataToDB();
                break; }
        fileName = "";
        commonDataListBox.SelectedItem = null;}
    else{
        if(statementsFile != null && commonDataFile == null){
            fileName = statementsFile.Content.ToString();

```

```

switch(fileName) {
    case machinePartsText:
        machineParts.sendDataToDB();
        MessageBox.Show("Відомості доручень кафедри " + fileName +
" завантажено до бази даних!");
        break;
    case mbtText:
        mbt.sendDataToDB();
        MessageBox.Show("Відомості доручень кафедри " + fileName +
" завантажено до бази даних!");
        break;
    case economyAndCustomsText:
        economyAndCustoms_sheet1.sendDataToDB();
        economyAndCustoms_sheet2.sendDataToDB();
        MessageBox.Show("Відомості доручень кафедри " + fileName +
" завантажено до бази даних!");
        break;
    case economicalTheoryText:
        economicalTheory.sendDataToDB();
        MessageBox.Show("Відомості доручень кафедри " + fileName +
" завантажено до бази даних!");
        break;
    case electricalMachinesText:
        electricalMachines.sendDataToDB();
        MessageBox.Show("Відомості доручень кафедри " + fileName +
" завантажено до бази даних!");
        break;
    case industrialEnergySupplyText:
        industrialEnergySupply.sendDataToDB();
        MessageBox.Show("Відомості доручень кафедри " + fileName +
" завантажено до бази даних!");
        break;
    case computerSystemsAndNetworksText:
        computerSystemsAndNetworks_sheet1.sendDataToDB();
        computerSystemsAndNetworks_sheet2.sendDataToDB();
        MessageBox.Show("Відомості доручень кафедри " + fileName +
" завантажено до бази даних!");
        break;
    case marketingAndLogisticsText:
        marketingAndLogistics.sendDataToDB();
        MessageBox.Show("Відомості доручень кафедри " + fileName +
" завантажено до бази даних!");

```

```

        break;
    case internationalEconomicRelationsText:
        internationalEconomicRelations.sendDataToDB();
        MessageBox.Show("Відомості доручень кафедри " + fileName +
" завантажено до бази даних!");
        break;
    case accountingAndAuditText:
        accountingAndAudit_sheet1.sendDataToDB();
        accountingAndAudit_sheet2.sendDataToDB();
        MessageBox.Show("Відомості доручень кафедри " + fileName +
" завантажено до бази даних!");
        break;
    case appliedMathematicsText:
        appliedMathematics.sendDataToDB();
        MessageBox.Show("Відомості доручень кафедри " + fileName +
" завантажено до бази даних!");
        break;
    case computerSoftwareText:
        computerSoftware.sendDataToDB();
        MessageBox.Show("Відомості доручень кафедри " + fileName +
" завантажено до бази даних!");
        break;
    case psychologyText:
        psychology.sendDataToDB();
        MessageBox.Show("Відомості доручень кафедри " + fileName +
" завантажено до бази даних!");
        break;
    case aviationEngineConstructionTechnologyText:
        aviationEngineConstructionTechnology.sendDataToDB();
        MessageBox.Show("Відомості доручень кафедри " + fileName +
" завантажено до бази даних!");
        break;
    case tourismText:
        tourism_sheet1.sendDataToDB();
        tourism_sheet2.sendDataToDB();
        MessageBox.Show("Відомості доручень кафедри " + fileName +
" завантажено до бази даних!");
        break;
    default:
        MessageBox.Show("Відомості доручень цієї кафедри ще не завантажили...");
        break;}
fileName = "";

```

```

        statementsListBox.SelectedItem = null; }
else {
    if(statementsFile == null && commonDataFile == null){
        MessageBox.Show("Не обрано дані для завантаження!"); } } }
void WatchBugsReport(object sender, RoutedEventArgs e){
    const string path = @"E:\BACHELORS WORK\TIMETABLE\DataCollectionApp\BugsReport.txt";
    Process.Start(path); }
void statementsListBox_Changed(object sender, SelectionChangedEventArgs e){
    try{
        selectedFile.Text = ((ListBoxItem)statementsListBox.SelectedItem).Content.ToString(); }
    catch (Exception ex){
        selectedFile.Text = null;}
    switch(selectedFile.Text){
        case machinePartsText:
            selectedFileName.Text = machineParts.FileName;
            lastWriteTime.Text = machineParts.lastWriteTime;
            break;
        case mbtText:
            selectedFileName.Text = mbt.FileName;
            lastWriteTime.Text = mbt.lastWriteTime;
            break;
        case economyAndCustomsText:
            selectedFileName.Text = economyAndCustoms_sheet1.FileName;
            lastWriteTime.Text = economyAndCustoms_sheet1.lastWriteTime;
            break;
        case economicalTheoryText:
            selectedFileName.Text = economicalTheory.FileName;
            lastWriteTime.Text = economicalTheory.lastWriteTime;
            break;
        case electricalMachinesText:
            selectedFileName.Text = electricalMachines.FileName;
            lastWriteTime.Text = electricalMachines.lastWriteTime;
            break;
        case industrialEnergySupplyText:
            selectedFileName.Text = industrialEnergySupply.FileName;
            lastWriteTime.Text = industrialEnergySupply.lastWriteTime;
            break;
        case computerSystemsAndNetworksText:
            selectedFileName.Text = computerSystemsAndNetworks_sheet1.FileName;
            lastWriteTime.Text = computerSystemsAndNetworks_sheet1.lastWriteTime;
            break;
        case marketingAndLogisticsText:

```

```

        selectedFileName.Text = marketingAndLogistics.FileName;
        lastWriteTime.Text = marketingAndLogistics.lastWriteTime;
        break;
    case internationalEconomicRelationsText:
        selectedFileName.Text = internationalEconomicRelations.FileName;
        lastWriteTime.Text = internationalEconomicRelations.lastWriteTime;
        break;
    case accountingAndAuditText:
        selectedFileName.Text = accountingAndAudit_sheet1.FileName;
        lastWriteTime.Text = accountingAndAudit_sheet1.lastWriteTime;
        break;
    case appliedMathematicsText:
        selectedFileName.Text = appliedMathematics.FileName;
        lastWriteTime.Text = appliedMathematics.lastWriteTime;
        break;
    case computerSoftwareText:
        selectedFileName.Text = computerSoftware.FileName;
        lastWriteTime.Text = computerSoftware.lastWriteTime;
        break;
    case psychologyText:
        selectedFileName.Text = psychology.FileName;
        lastWriteTime.Text = psychology.lastWriteTime;
        break;
    case aviationEngineConstructionTechnologyText:
        selectedFileName.Text = aviationEngineConstructionTechnology.FileName;
        lastWriteTime.Text = aviationEngineConstructionTechnology.lastWriteTime;
        break;
    case tourismText:
        selectedFileName.Text = tourism_sheet1.FileName;
        lastWriteTime.Text = tourism_sheet1.lastWriteTime;
        break;
    default:
        selectedFileName.Text = "Відомості доручень цієї кафедри ще не завантажили...";
        lastWriteTime.Text = "...";
        break;}}
void commonDataListBox_Changed(object sender, SelectionChangedEventArgs e){
    try{
        selectedFile.Text = ((ListBoxItem)commonDataListBox.SelectedItem).Content.ToString(); }
    catch (Exception ex){
        selectedFile.Text = null;}
    switch(selectedFile.Text){
        case auditoryTypesText:

```

```
    selectedFileName.Text = auditoryTypes.FileName;
    lastWriteTime.Text = auditoryTypes.lastWriteTime;
    break;
case disciplinesText:
    selectedFileName.Text = disciplines.FileName;
    lastWriteTime.Text = disciplines.lastWriteTime;
    break;
case facultiesText:
    selectedFileName.Text = faculties.FileName;
    lastWriteTime.Text = faculties.lastWriteTime;
    break;
case departmentsText:
    selectedFileName.Text = departments.FileName;
    lastWriteTime.Text = departments.lastWriteTime;
    break;
case teachersText:
    selectedFileName.Text = teachers.FileName;
    lastWriteTime.Text = teachers.lastWriteTime;
    break;
case auditoriesText:
    selectedFileName.Text = auditories.FileName;
    lastWriteTime.Text = auditories.lastWriteTime;
    break;
case groupsText:
    selectedFileName.Text = studyGroups.FileName;
    lastWriteTime.Text = studyGroups.lastWriteTime;
    break; } } }
```

ДОДАТОК В
Слайди презентації

Національний університет «Запорізька політехніка»

Програмне забезпечення для збору даних для формування розкладу закладу вищої освіти

Виконав:
студент групи КНТ-117

О.С. Марков

Керівник:
доцент, к.т.н.

Т.І. Каплієнко

1

Рисунок В.1 – Слайд 1

Мета, об'єкт і предмет дослідження

Мета дослідження – створення програмної системи для автоматизації та прискорення збору даних, необхідних для формування розкладу закладу вищої освіти.

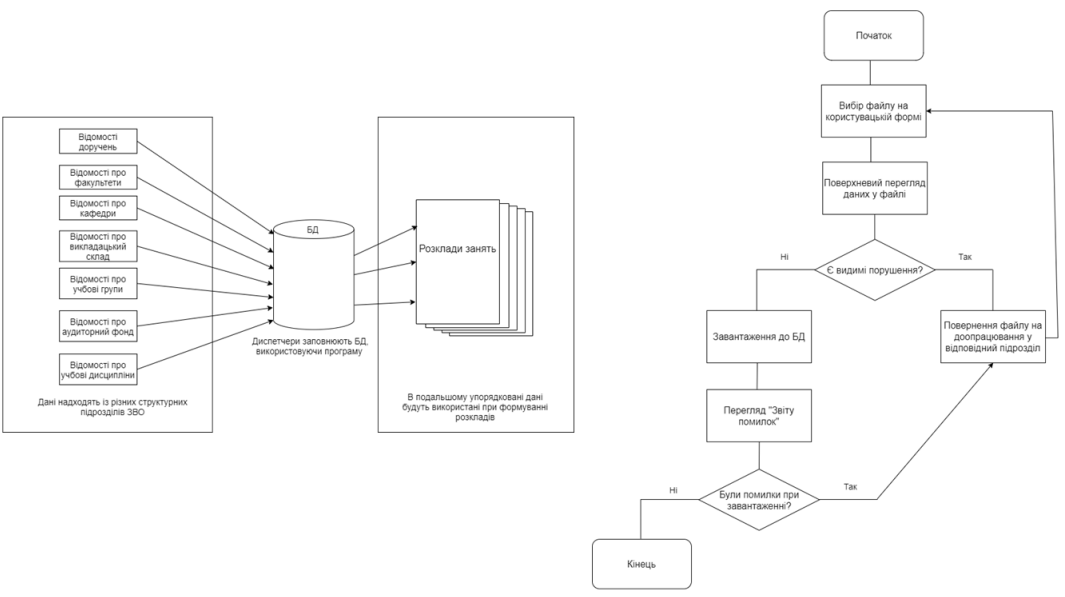
Об'єкт дослідження – програмні засоби для збору даних для формування розкладу закладу вищої освіти.

Предмет дослідження – комп'ютерні програми для збору даних для формування розкладу закладу вищої освіти.

2

Рисунок В.2 – Слайд 2

Схема зі збору даних та алгоритм дій диспетчера



3

Рисунок В.3 – Слайд 3

Сутності предметної області

Сутність	Опис
Викладач	Інформація про викладацький склад ЗВО
Факультет	Інформація про факультети ЗВО
Кафедра	Інформація про кафедри ЗВО
Тип аудиторій	Інформація про типи аудиторій в ЗВО
Аудиторія	Інформація про аудиторний фонд ЗВО
Дисципліна	Інформація про дисципліни
Учбова група	Інформація про учбові групи
Заняття	Інформація про заняття
Заняття групи	Ця сутність зв'язує інформацію сутностей «Учбові групи» та «Заняття»
Аудиторія для заняття	Ця сутність зв'язує інформацію сутностей «Аудиторія» та «Заняття»
Викладач, що веде заняття	Ця сутність зв'язує інформацію сутностей «Викладач» та «Заняття»
Час заняття	Інформація про час проведення заняття

4

Рисунок В.4 – Слайд 4

Огляд існуючих методів вирішення проблеми

Критерій порівняння	Програма «FET»	Програма «ІС: Автоматизоване складання розкладу. Університет»
Різноманітність режимів введення	+	+
Оцінка можливих неточностей у вхідних даних	–	–
Завантаження до бази даних «в один клік»	–	–
Зручність інтерфейсу	+	+
Багатомовність	+	+
Безкоштовність	+	–

5

Рисунок В.5 – Слайд 5

Постановка завдання

В результаті дослідження проблеми збору даних, необхідних для ФР ЗВО, було формалізовано завдання до роботи, яке полягає в створенні десктопного застосунку, якість якого визначається за вище згаданими критеріями.

Вимоги:

- реалізація графічного інтерфейса користувача;
- реалізація можливості завантаження даних до БД;
- реалізація можливості перегляду файлів;
- реалізація введення «Звіту помилок» щодо неточностей у вхідних даних.

Рисунок В.6 – Слайд 6

Вибір інструментів розробки програмного забезпечення

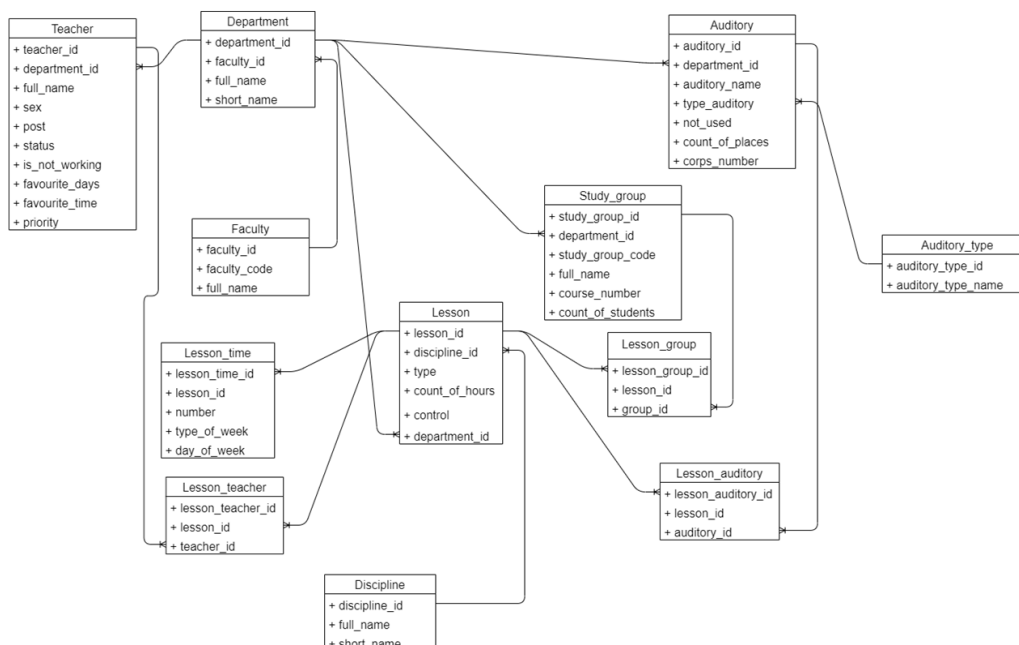
Було обрано наступні інструменти та технології:

- мову програмування C#;
- середовище розробки SharpDevelop;
- технологію для побудови графічного інтерфейсу WPF;
- систему управління базами даних MySQL;
- веб-застосунок phpMyAdmin для адміністрування сервера MySQL.

7

Рисунок В.7 – Слайд 7

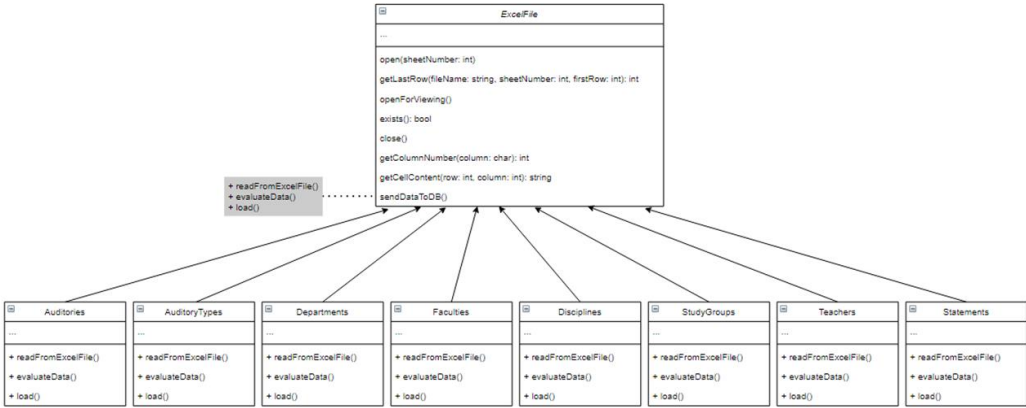
Схема бази даних



8

Рисунок В.8 – Слайд 8

Реалізація патерну «Шаблонний метод»



9

Рисунок В.9 – Слайд 9

Інші класи програми



10

Рисунок В.10 – Слайд 10

Приклад структури реалізованої таблиці бази даних

Структура таблиці Study_group

Ім'я поля	Ключ	Значення за замовчуванням	Тип	Розмір
study_group_id	Первинний		int	11
department_id			int	11
study_group_code		NULL	varchar	50
full_name			varchar	50
course_number		NULL	varchar	3
count_of_students		NULL	int	11

11

Рисунок В.11 – Слайд 11

Повідомлення

Повідомлення	Опис
Не обрано файл для перегляду!	Користувач натиснув кнопку «Переглянути», не обравши файл
Дані завантажено до бази даних	Операція по завантаженню загальних даних пройшла успішно
Відомості доручень завантажено до бази даних	Операція по завантаженню даних з ВД до БД пройшла успішно
Не обрано дані для завантаження!	Користувач натиснув кнопку «Завантажити», не обравши файл
Помилка при отриманні даних з файлу	Помилка при зчитуванні даних пояснюється його неправильним заповненням; такий файл слід надіслати на доопрацювання
Виникла помилка під час завантаження до бази даних!	Найімовірніше, до БД ще не були завантажені залежні дані і необхідно виконати перевірку БД
Дані вже містяться в базі даних!	Спроба завантажити ті самі, ніяк не відредаговані дані

12

Рисунок В.12 – Слайд 12

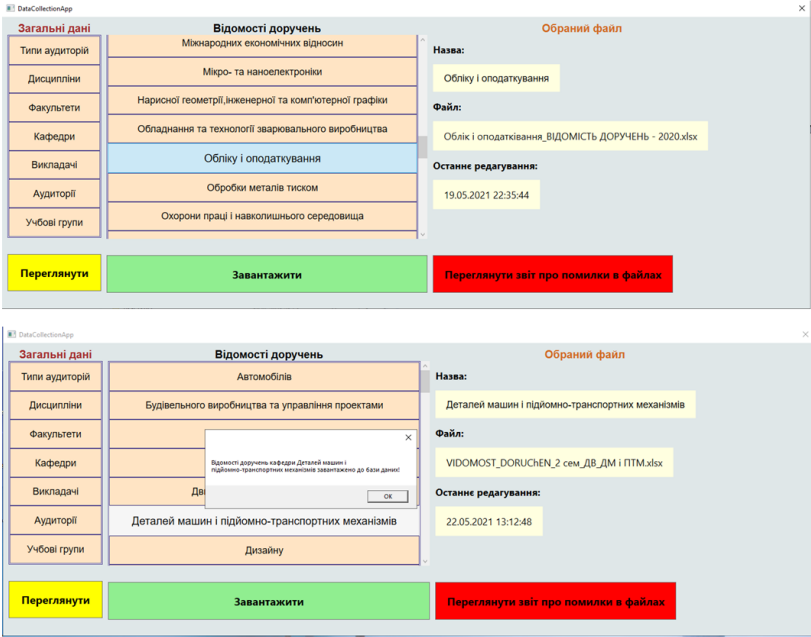
Шаблон файлу відомостей доручень

Стовпець в файлі	Дані в стовпці
B	Назва дисципліни
C	№ груп
D	Вид занять
F	Кількість годин на тиждень
G	Контроль
H	Прізвище ім'я по батькові викладачів
I	Пропозиції щодо аудиторій
J	Пропозиції до складання розкладу занять

13

Рисунок В.13 – Слайд 13

Екранні форми



14

Рисунок В.14 – Слайд 14

Висновки

В ході виконання роботи було виконано аналіз предметної області, виконано розробку архітектури програми, реалізовано її компоненти, а також оформлено такі програмні документи як «Керівництво програміста», «Керівництво оператора» та «Технічне завдання».

В результаті роботи було отримано програмний продукт, що дозволяє виконати поставлене завдання, а саме – автоматизацію та прискорення процесу збору даних, необхідних для ФР ЗВО. Роботу програми було протестовано на реальних даних, отриманих зі структурних підрозділів ЗВО НУ «Запорізька політехніка». За необхідності програма може бути модифікована та використана під час розробки більш складної, комплексної системи, що вирішуватиме завдання ФР ЗВО НУ «Запорізька політехніка».

15

Рисунок В.15 – Слайд 15