Одеський національний політехнічний університет

Інститут комп’ютерних систем

Кафедра системного програмного забезпечення

**Пояснювальна записка**

до дипломної роботи бакалавра

на тему «Програма аналізу статистичних даних з процесу навчання для підтримки прийняття рішень про організацію навчальних курсів»

Виконав: студент 4 курсу, групи АС-131

напряму підготовки

6.050103 – Програмна інженерія

Карлов О. С

Керівник: Зіноватна С. Л.

Рецензент:

Одеса – 2017

Одеський національний політехнічний університет

Інститут комп’ютерних систем\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кафедра системного програмного забезпечення \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Освітньо-кваліфікаційний рівень бакалавр\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Напрям підготовки 6.050103 – Програмна інженерія\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ЗАТВЕРДЖУЮ**

**Завідувач кафедри СПЗ**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Крісілов В.А.

“\_\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2017 року

**З А В Д А Н Н Я**

**НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ**

Карлову Олексію Степановичу

1. Тема проекту (роботи) «Програма аналізу статистичних даних з процесу навчання, для підтримки прийняття рішень про організацію навчальних курсів»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

керівник проекту (роботи) Зіноватна Світлана Леонідівна, к. т. н., доцент,

(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом ректора ОНПУ від 04.04.2017 року № 179-в

2. Строк подання студентом проекту (роботи)\_\_\_\_\_13.06.2017\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3. Вихідні дані до проекту (роботи) завдання на розробку, список рекомендованої літератури\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)Специфікація вимог до системи, планування виконання проекту, проектування системи аналізу статистичних даних, програмна реалізація системи, тестування системи.\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

відповідно до слайдів презентації: Ціль та задачі, Архітектура системи, Варіанти використання, Планування, Проектування бази даних, Проектування архітектури, Реалізована архітектура\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

6. Консультанти розділів проекту (роботи)

|  |  |  |  |
| --- | --- | --- | --- |
| Розділ | Прізвище, ініціали та посада  консультанта | Підпис, дата | |
| завдання видав | завдання  прийняв |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

7. Дата видачі завдання 30.01.2017**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| №  з/п | Назва етапів дипломного  проекту (роботи) | Строк виконання етапів проекту  ( роботи ) | Примітка |
| 2 | Склад календарного плану роботи | 30.01.2017 | виконано |
| 3 | Аналіз вимог | 13.02.2017 | виконано |
| 4 | Розробка структур даних | 20.02.2017 | виконано |
| 5 | Проектування системи | 17.03.2017 | виконано |
| 6 | Програмна реалізація системи | 1.05.2017 | виконано |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Студент \_\_\_\_\_\_\_\_\_\_\_\_\_ Карлов О. С.**

( підпис )

**Керівник роботи \_\_\_\_\_\_\_\_\_\_\_\_\_ Зіноватна С. Л.**

( підпис )

РЕФЕРАТ

Пояснювальна записка до дипломної роботи:48 с., 22 рис., 23 табл., 1 додаток, 6 джерел.

Метою роботи є скорочення часу на отримання зведених даних про стан рівня успішності студентів, які наддадуть подальшу можливість реорганізації системи проведення занять з боку керівництва фірми.

Методи розробки базуються на технологіях Java,сервері бази даних MySQL (БД).

Як результат роботи виконана програмна реалізація системи формування вхідних даних для аналізу, за допомогою якої надається можливість реорганізувати процес навчальних курсів.

Ключові слова: зведена таблиця, таблиця фактів, статистика, Java, MySQL, Excel.

# ЗМІСТ

[ВСТУП 6](#_Toc484449757)

[1 СПЕЦИФІКАЦІЯ ВИМОГ ДО СИСТЕМИ ФОРМУВАННЯ ВИХІДНИХ ДАНИХ ДЛЯ АНАЛІЗУ 7](#_Toc484449758)

[1.1 Аналіз існуючих аналогів 7](#_Toc484449759)

[1.1 Опис функціональних вимог](#_Toc484449759) 8

[1.2 Опис нефункціональних вимог](#_Toc484449760) 12

[2 ПЛАНУВАННЯ ВИКОНАННЯ ПРОЕКТУ 1](#_Toc484449761)4

[2.1 Оцінка тривалості розробки](#_Toc484449762) 14

[2.2 Розробка плану робіт проекту](#_Toc484449767) 19

[2.3 Аналіз ризиків 2](#_Toc484449768)1

[3 ПРОЕКТУВАННЯ СИСТЕМИ ФОРМУВАННЯ ВИХІДНИХ ДАНИХ ДЛЯ АНАЛІЗУ](#_Toc484449769) 26

[3.1 Проектування структури та організації класів](#_Toc484449771) 26

[3.2 Проектування системи отримування даних](#_Toc484449770) 27

[3.3 Проектування сховищ даних 30](#_Toc484449772)

[4 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ ФОРМУВАННЯ ВИХІДНИХ ДАНИХ ДЛЯ АНАЛІЗУ](#_Toc484449775) 35

[4.1 Опис програмних класів 37](#_Toc484449776)

[4.2 Алгоритми отримання та підготовлення даних для відображення](#_Toc484449777) 38

[4.3 Опис використаних технологій](#_Toc484449778) 39

[5 ТЕСТУВАННЯ СИСТЕМИ ФОРМУВАННЯ ВИХІДНИХ ДАНИХ ДЛЯ АНАЛІЗУ](#_Toc484449779) 40

[5.1 Тестування функціональності системи](#_Toc484449780) 43

[5.2 Приклад використання системи](#_Toc484449781) 47

[ВИСНОВКИ](#_Toc484449782) 48

[СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ](#_Toc484449783) 49

# ВСТУП

Забезпечення своєчасного доступу до інформації – є необхідна частина нашого життя.

В наш час фірми нерідко організують курси для підвищення кваліфікації співробітників, щоб покращити навички працівників у будь-якій галузі. Керівнику необхідно контролювати процес навчання учнів, успіх працівників, для постійного менеджменту часу, ефективного керування то що.

У цій роботі розглядається проектування та розробка інформаційної системи, яка надає користувачеві можливість отримати статистичні та аналітичні дані у вигляді таблиці фактів та зведеної таблиці Excel, отриманої на основі оперативних даних про стан навчання учнів.

Метою роботи є скорочення часу на отримання зведених даних про стан рівня успішності студентів, які наддадуть подальшу можливість реорганізації системи проведення занять з боку керівництва фірми, завдяки засобу за допомогою якого керівник може швидко і своєчасно отримувати дані стосовно кожного учня и вчителя.

Для досягнення мети в роботі вирішені такі задачі:

* зроблено аналіз вимог до програмної системи;
* виконано планування розробки програмного продукту;
* виконано проектування програмного продукту;
* реалізовано проект, використовуючи сучасні технології розробки;
* проведено тестування роботи проекту.

# 1 СПЕЦИФІКАЦІЯ ВИМОГ ДО СИСТЕМИ ФОРМУВАННЯ ВИХІДНИХ ДАНИХ ДЛЯ АНАЛІЗУ

Важливим етапом розробки інформаційних систем є перед проектний аналіз, одним з ключових завдань якого є виявлення вимог до будованої інформаційної системи. Формулювання вимог дозволяє створити закінчений опис поведінки системи, яку потрібно розробити, та список повністю сформульованих функціональних і нефункціональних вимог, що дає можливість точної оцінки етапів, планів та термінів проекту.

## Аналіз існуючих аналогів

У якості аналога розглянуто автоматизовану інформаційну систему контроля навчального процесу в навчальному закладі (АІС).

Система включає в себе:

– Автоматизоване робоче місце (АРМ) вчителя;

– АРМ батька;

– АРМ учня;

– АРМ адміністратора школи (завуча / директора).

АІС забезпечує:

* ведення бази даних учнів, вчителів та предметів;
* складання навчального плану для класів
* авторизація користувачів з визначенням прав доступу до даних

Система має зручний інтерфейс, що надає найбільш гнучкий спосіб перегляду, додавання, редагування та видалення даних, а також відбір і пошук необхідних даних.

Недоліки:

– відсутня можливість підготовки та формування даних для подальшого аналізу;

– відсутня можливість зберегти дані, у окремий файл чи документ.

Таблиця 1.1 – Опис існуючих програмних продуктів

|  |  |  |  |
| --- | --- | --- | --- |
| Проблеми  Назва  продукту | Розподілення ролей | Планування учбового процесу | Підготовка та формування даних для подальшого аналізу |
| Автоматизована інформаційна система контролю навчального процесу в навчальному закладі. | + | + | - |

Аналізований вище аналог досить непогано вирішує проблему сбіру даних навчання, але не вирішує актуальну проблему, а саме підготовку та сбір даних, для подальшого аналізу. Тому є необхідність розробити нову програмну систему, яка дозволить отримати актуальну інформацію у зручному для аналізу вигляді – своєчасно.

## 1.2 Опис функціональних вимог

Для формалізації вимог до системи використовуються варіанти використання (ВВ) (рис. 1.1).

Кожний сценарій варіанту використання зосереджений на описі того, як досягти конкретної мети або вирішити конкретну задачу. Для більшості програмних проектів це означає, що знадобиться велика кількість сценаріїв варіантів використання, щоб визначити необхідний набір властивостей розроблюваної системи.

Сценарій використання розглядає систему як «чорний ящик», а взаємодія із системою, включаючи системні відповіді, описується с точки зору зовнішнього спостерігача. Такий підхід змушує зосередитись на тому, що система повинна робити.

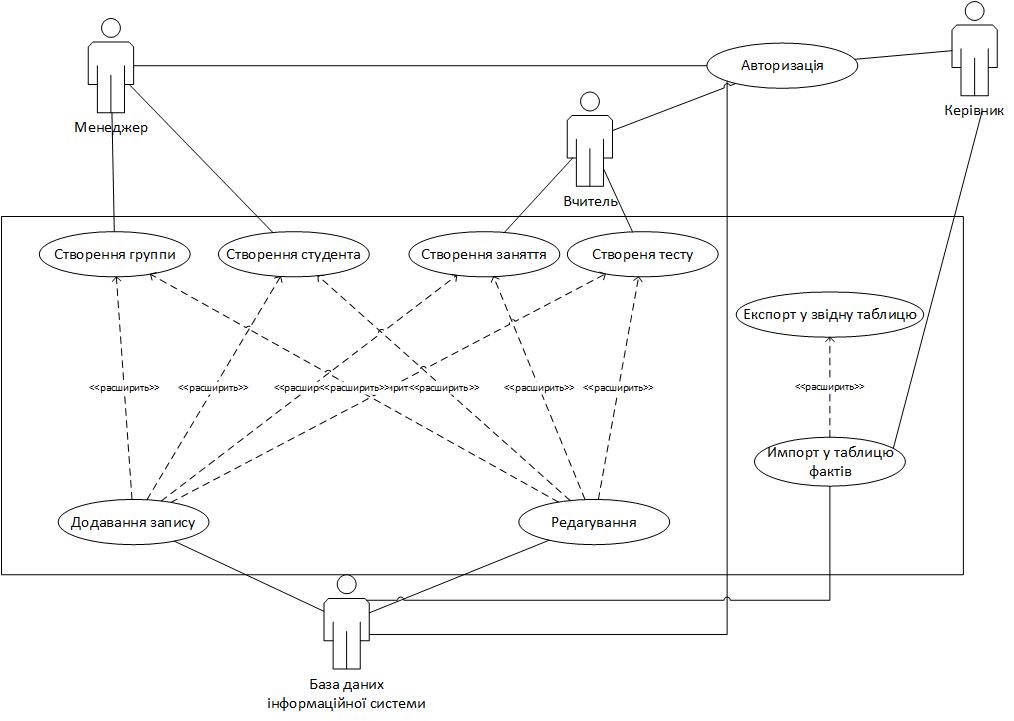


Рисунок 1.1 – Діаграма варіантів використання

Розглянемо докладно сценарії варіантів використання системи аналізу статистичних даних.

**Варіант використання «Авторизація»**

Передумова: користувач не авторизований у системі.

Мінімальні гарантії: користувача не буде авторизовано. Система запропонує авторизуватися повторно

Гарантії успіху: користувач авторизується.

Тригер: користувач повідомляє системі про бажання авторизуватися.

Основний успішний сценарій:

1. Користувач запускає вікно авторизації;

2. Система запитує у користувача дані для авторизації;

3. Користувач вводить дані в систему, система підтверджує.

Альтернативні сценарії:

3а. Користувач ввів некоректні дані;

3а.1. Система сповіщає користувача про некоректність введених даних;

3а.2. Перехід до пункту 2.

**Варіант використання «Додавання запису»**

Передумова:

Мінімальні гарантії: нову групу – не було створено, система продовжить роботу.

Гарантії успіху: у БД створено новий запис.

Тригер: користувач відкрив вікно для введення даних.

Основний успішний сценарій:

1. Система запрошу у користувача введення даних;
2. Користувач вводить необхідні дані;
3. Користувач тисне кнопку «Додати», система додає дані до БД система повідомляє користувача про успіх.

Альтернативні сценарії:

2а. Дані не були введені користувачем;

2а.1. Система повідомляє користувачу про відсутність даних;

3а. Немає з’єднання з інтернетом;

4а.1. Система повідомляє користувачу про відсутність з’єднання;

**Варіант використання «Редагувати запис»**

Передумова: вибрана інформація для редагування.

Мінімальні гарантії: запис – не відредаговано, система продовжить роботу.

Гарантії успіху: системи змінить необхідну інформацію у БД

Тригер: із запропонованих варіантів – користувач вибрав потрібний запис.

Основний успішний сценарій:

1. Система виводить користувачу можливі варіанти запису у таблицю;
2. Користувач редагує необхідну інформацію;
3. Користувач натискає кнопку «Додати», система оновлює дані у БД.

Альтернативний сценарій:

2а.Дані користувача не були введені;

2а.1.Система повідомляє користувачу про відсутність даних;

4а.Немає з’єднання з БД;

4а.1.Система повідомляє користувачу про відсутність з’єднання;

**Варіант використання «Імпорт даних у таблицю фактів»**

Передумова: користувач авторизувався під профілем «керівник» .

Мінімальні гарантії: таблиця не буде сформована, система продовжить роботу.

Гарантії успіху: користувач побачить таблицю фактів, сформовану за вибраними параметрами

Тригер: користувач вибрав режим «Таблиця Фактів».

Основний успішний сценарій:

1. Система пропонує користувачу вибрати фільтри для побудови таблиці фактів;
2. Користувач вибирає фільтри, які йому потрібні;
3. Користувач натискає кнопку «Загрузити таблицю»;
4. Система формує таблицю и виводить її на екран.

Альтернативний сценарій

2а. Користувач не вибрав жодного фільтру;

2а.1.система повідомляє користувачу про відсутність фільтрів;

4а. Немає з’єднання з БД;

4а.1. Система повідомляє користувача про відсутність з’єднання.

**Варіант використання «Експорт таблиці фактів у зведену таблицю»**

Передумова: користувач авторизувався під профілем «Керівник» .

Мінімальні гарантії: звідна таблиця не буде збережена, система продовжить роботу.

Гарантії успіху: система збереже таблицю фактів, за вибраними параметрами

Тригер: користувач сформував таблицю фактів.

Основний успішний сценарій:

1. Користувач натискаю кнопку «Збереження у Excel»;
2. Користувач вибирає параметри звідної таблиці;
3. Користувач натискає кнопку «Загрузити таблицю»;
4. Користувач вибирає путь та ім’я файлу система зберігає файл;
5. Система формує файл таблиці и зберігає на жорсткому диску.

Альтернативний сценарій

2а. Користувач не вибрав жодного параметру;

2а.1. Система повідомляє користувачу про відсутність параметрів;

5а. Системі не вдалося зберегти таблицю;

5а.1.Система повідомляє користувача про помилку збереження.

## Опис нефункціональних вимог

Нефункціональні вимоги– це вимоги до [програмного забезпечення](https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%B5_%D0%B7%D0%B0%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%BD%D1%8F), які задають критерії для оцінки якості його роботи. На відміну від [функціональних вимог](https://uk.wikipedia.org/wiki/%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D1%96%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D1%96_%D0%B2%D0%B8%D0%BC%D0%BE%D0%B3%D0%B8), нефункціональні вимоги визначають якою система повинна бути [3].

Обмеження використання:

1. База даних може бути розгорнута при наявності MySQL4.0

Нефункціональні вимоги:

1. Продуктивність – відповідь на запит від клієнта, не перевищує 5 секунд.
2. Готовність – система працює з даними з БД, тому у випадку недосяжності БД при запиті даних клієнтом, система повинна сповістити клієнта.
3. Безпека – система працює з авторизованими даними користувачів. Кожен користувач має свій пароль.

## Висновки

У даному розділі роботи визначені вимоги до системи, проведено аналіз існуючих рішень, а також описані функціональні і нефункціональні вимогу до системи. Зокрема, була визначена діаграма варіантів використання, з описом сценарію поведінки прецедентів для прецедентів.

# 2ПЛАНУВАННЯ ВИКОНАННЯ ПРОЕКТУ

## 2.1 Оцінка тривалості розробки

Для розрахунків буде використовуватися наведена на рис.2.1 діаграма варіантів використання. Усі обчислення у рамках методу UCP будуть проводитися на її основі.

### 2.1.1 Оцінка акторів

### Як відомо, у рамках методу виділяється 3 типу акторів: прості, середні та складні.

Проаналізувавши акторів системи, маємо наступні висновки:

* Простих акторів: бази даних –1 (вага – 1)
* Складних акторів: користувачі – 3 (вага – 3)

Обчислимо значення нескорегованої оцінки акторів UAW:

(2.1)

де n – кількість акторів,   
ai – вага актору.

### 2.1.2 Оцінка варіантів використання

### У даному випадку буде використатися метод оцінки по кількості транзакцій, відповідних до кожного варіанта використання.

Для підрахунку нескорегованої оцінки варіантів використання (UUCW) слід підрахувати кількість варіантів використання кожного типу, помножити на відповідні вагові коефіцієнти та знайти суму отриманих добутків.

Обчислимо нескореговану оцінку варіантів використання UUCW:

(2.2)

Необхідно оцінити відсоток повторного використання програмних компонентів. Припустимо, що він дорівнює 45%. Такий вибір обґрунтований тим, що майже половина більшість транзакцій є операціями читання/запису з/до бази даних та мають велику схожість.

Обчислимо показникUCP:

(2.3)

### 2.1.3 Оцінка технічних факторів

### Першою категорією, яка підлягає оцінці, є технічні фактори, які, вочевидь, визначають фізичні властивості системи. Вони розглянуті у таблиці 2.1.

Таблиця 2.1 – Технічні фактори

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Фактор | Опис | Вага | Пояснення | Оцінка |
| T1 | Розподіленість системи | 2 | Система не потребує розподілених обчислень | 0 |
| T2 | Час відгуку | 1 | Час відгуку слід мінімізувати, оскільки це впливає на роботу клієнта | 4 |
| T3 | Ефективність кінцевого користувача | 1 | Ефективність користувача не залежить від системи | 2 |

Продовження таблиці 2.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Фактор | Опис | Вага | Пояснення | Оцінка |
| T4 | Складність обробки | 1 | Система обробляє відповіді від бази даних аналізатору та перетворює у необхідний для клієнта формат | 4 |
| T5 | Фокус на повторному використанні коду | 1 | При розробці слід фокусуватися на повторному використанні коду. | 4 |
| T6 | Простота інсталяції | 0,5 | БД та система, має встановлюватися просто, слід забезпечити можливість встановлення з зовнішніх джерел | 4 |
| T7 | Простота використання | 0,5 | Система повинна використовуватися через прості запити до БД | 3 |
| T8 | Портативність | 2 | Система має працювати під усіма сучасними настільними системами | 4 |
| T9 | Простота змін | 1 | Система не має бути простою у змінах в цілому, але окремі частини цього потребують | 1 |
| T10 | Паралельні обчислення | 1 | Система не потребує паралельних обчислень | 0 |
| T11 | Засоби захисту | 1 | Система має використовувати засоби захисту даних користувачів | 4 |
| T12 | Доступ до третьої сторони | 1 | Система буде використовуватися лише клієнтом | 1 |
| T13 | Потреби в спеціальному навчанні | 1 | Для використання системи не потрібно спеціальне навчання | 1 |

Обчислимо показник TFactor як суму добутків значень оцінок факторів на їх вагові коефіцієнти:

i = 32.5(2.4)

де Ti – загальна оцінка технічного фактору,  
n – кількість факторів.

Обчислимо оцінку технічного факторуTCF:

(2.5)

### 2.1.4 Оцінка зовнішніх факторів

### Другою категорією, яку необхідно оцінити, є зовнішні фактори. Ці фактори явно не зв’язані із самою системою та процесом її розробки, але дуже явно можуть впливати на ці два поняття. Зовнішні фактори розглянуті у таблиці 2.2.

Таблиця 2.2 – Зовнішні фактори

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Фактор | Опис | Вага | Пояснення | Оцінка |
| F1 | Знайомство з процесом розробки | 1,5 | Розробник знайомий із предметною областю та її технічними аспектами, але не досконало. | 3 |
| F2 | Досвід подібних проектів | 0,5 | Розробник має достатній досвід розробки баз даних. | 4 |

Продовження таблиці 2.2

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Фактор | | Опис | | Вага | | Пояснення | | Оцінка |
| F3 | | Досвід об’єктно-орієнтованої розробки | | 1 | | Розробник має достатній досвід в проектуванні об’єктно-орієнтованих застосувань | | 4 |
| F4 | | Досвідченість провідного аналітика | | 0,5 | | Аналітик не є досить досвідченим | | 1 |
| F5 | | Мотивація | | 1 | | Виконавець мотивован отриманням диплому | | 5 |
| F6 | | Стабільність вимог | | 2 | | Вимоги можуть змінюватися у дрібницях або не змінюватися зовсім | | 3 |
| F7 | | Часткова зайнятість працівників | | -1 | | Виконавець часткової зайнятості | | 3 |
| F8 | | Складність мови програмування | | -1 | | Мова програмування та обрані технології є в міру складними | | 3 |
|  | |  | |  | |  | 19 |

Обчислимо показник EFactor як суму добуткі взначень оцінок на їх вагові коефіцієнти:

i = 19(2.6)

Далі обчислимо оцінку зовнішнього EF:

(2.7)

Обчислимо скореговане значення UCP за формулою:

(2.8)

Для визначення тривалості розробки також необхідно підрахувати кількість факторів із множини F1-F8, абсолютне значення яких перевищує 3. Таких факторів рівне 3, отже, приймається, що на 1 UCP приходиться 20 робочих часів. Виходячи з цього, обчислимо тривалість розробки:

|  |  |
| --- | --- |
| *TDev = AUCP \* 20 = \* 20 =* 833.6*годин =*  *= 105 робочих днів по 8 годин ~ 3місяця та 2тижденя* | (2.9) |

## 2.2 Розробка плану робіт проекту

У даному розділі розглядаються задачі розбиття проекту на роботи, будування плану проекту та його аналіз. Зокрема, будуть побудовані WBS та діаграма Ганта, яка ілюструє план виконання робіт.

За допомогою виділення самих важливих робіт та розбиття їх на різні рівні, була виділена структура відображена на рис. 2.1.

Після побудови діаграм, було визначено, що критичні операції і шляхи –відсутні. Це відображає, що проект було сплановано з урахуванням усіх нюансів.

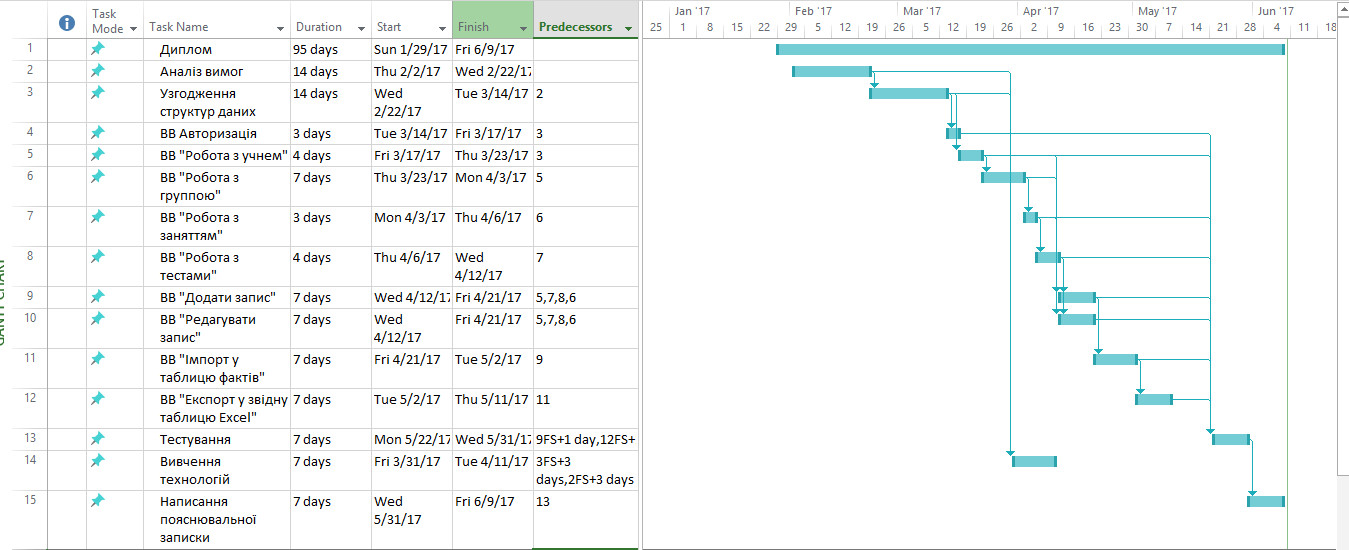


Рисунок 2.1 – WBSпроекту та діаграма Ганта

## 

## 2.3 Аналіз ризиків

Для інформаційної системи небезпечні ризики представлені на діаграмі Ісікави.

Побудуємо діаграми «риб'ячої кістки», що описують причини і слідства.

Це графічний спосіб дослідження найбільш істотних причинно-наслідкових взаємозв'язків між факторами й наслідками в досліджуваній ситуації або проблемі.

Ці діаграми представлені на рис. 2.2– 2.4.

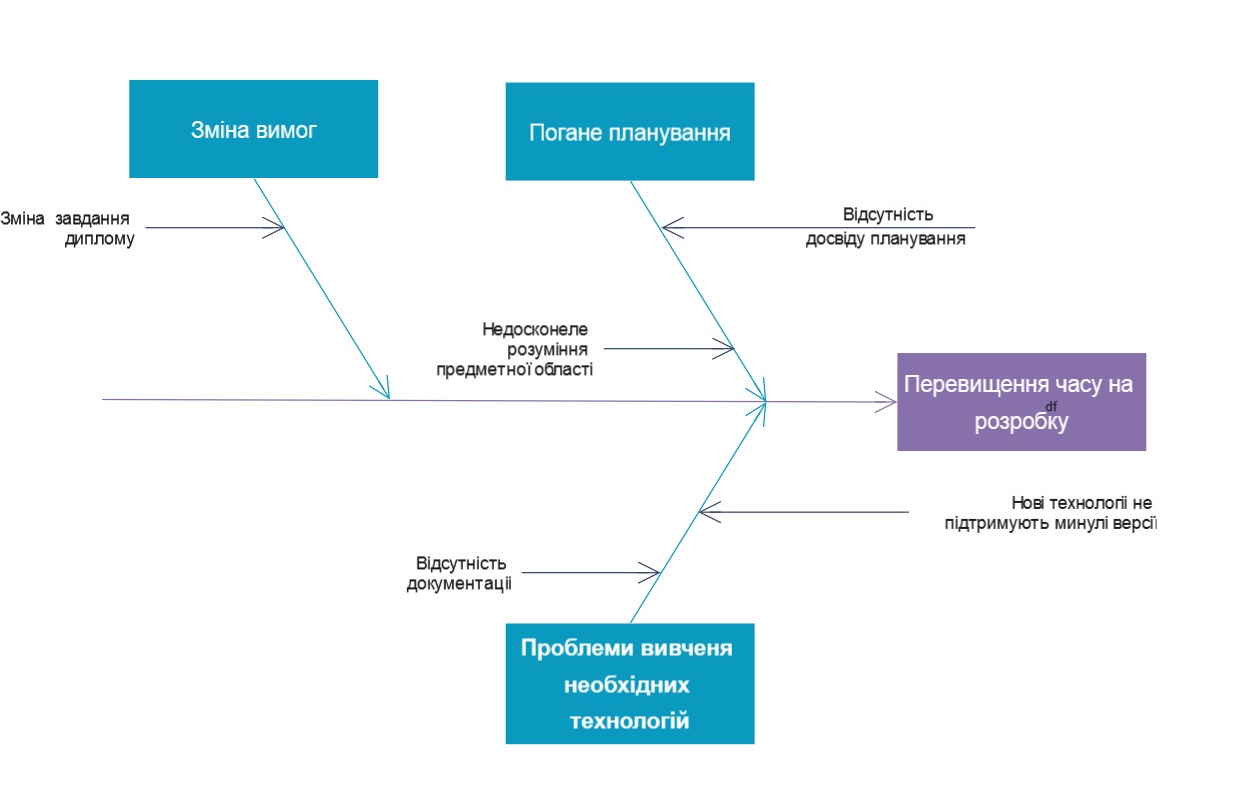


Рисунок 2.2 – Діаграма Ісікави для ризику «Перевищення часу на розробку»

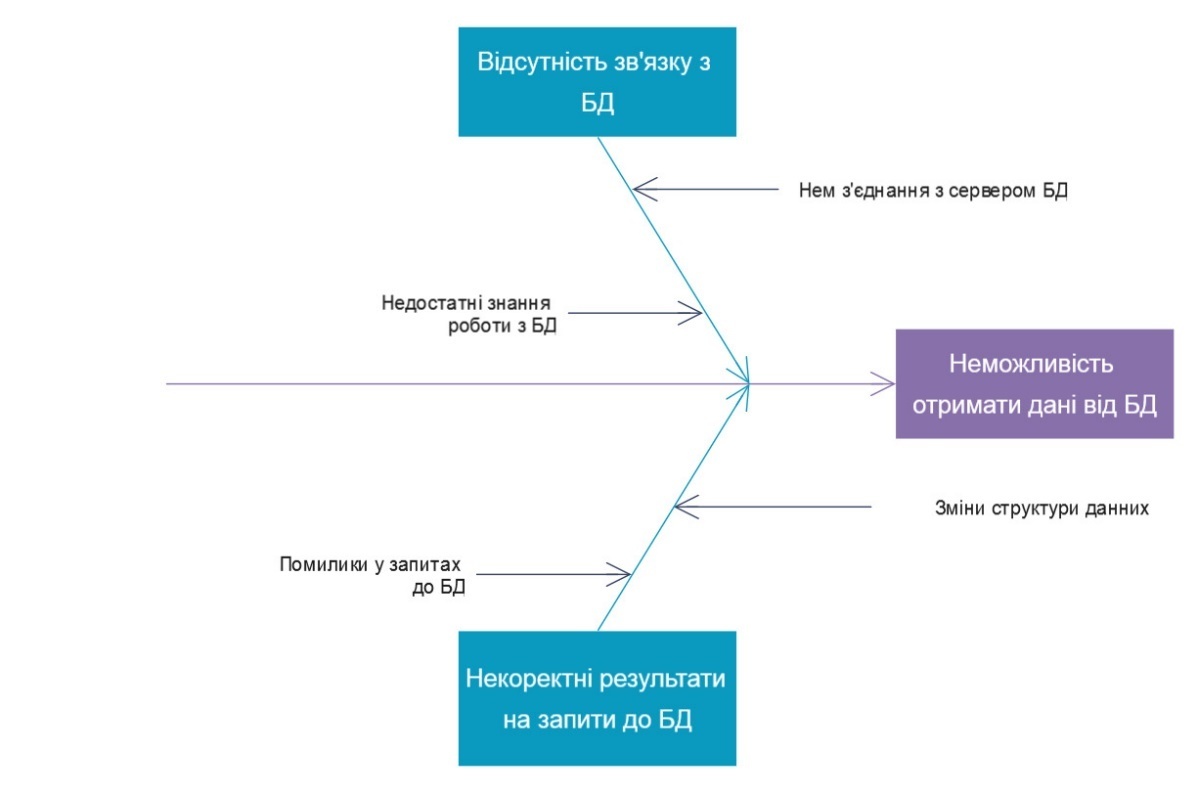


Рисунок 2.3 – Діаграма Ісікави для ризику «Неможливість отримати дані від БД»

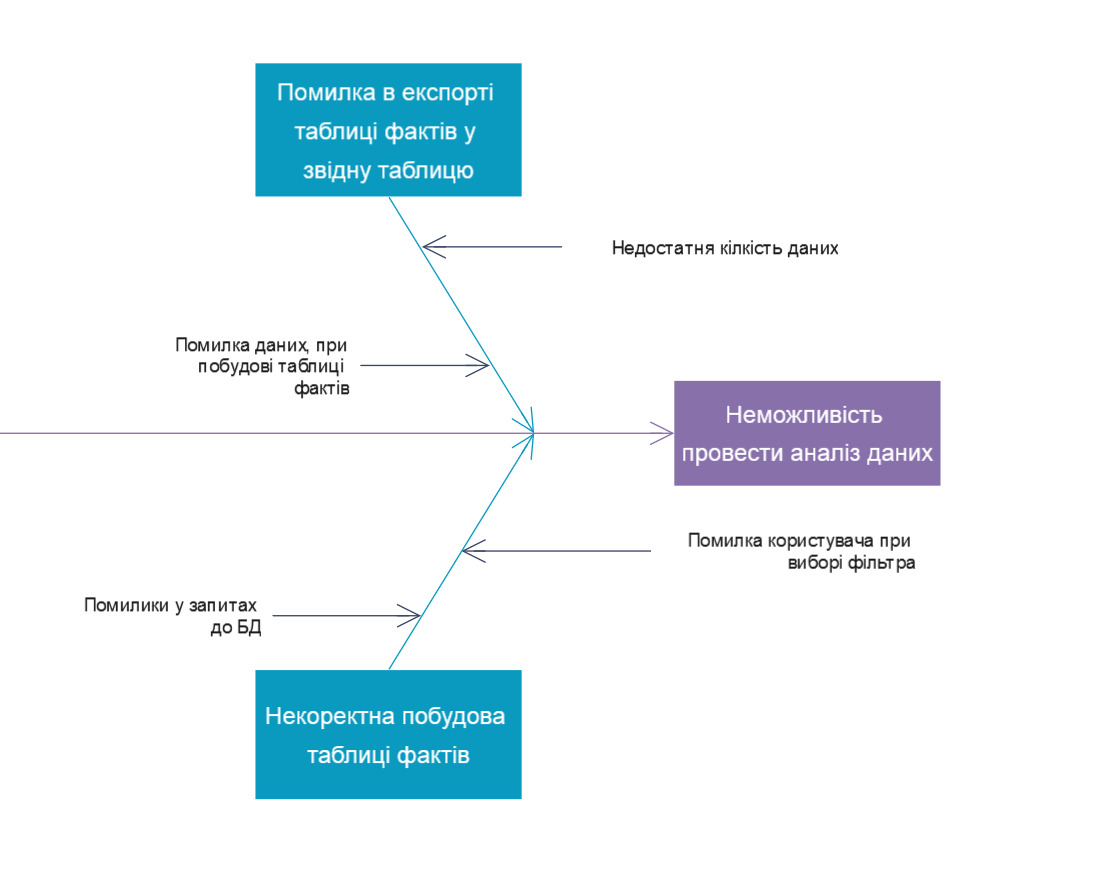


Рисунок 2.4 – Діаграма Ісікави для ризику «Неможливість провести аналіз даних»

У табл. 2.4 показані результати оцінки ризиків проекту.

Таблиця 2.4 – Оцінка ризиків проекту

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Ризик | Вплив | Правдо-подібність | Ранг |
| 1 | Зміна завдання диплому | 3 | 1 | Середній (3) |
| 2 | Відсутність досвіду планування | 3 | 2 | Високий (6) |
| 3 | Недостатнє знання предметної області | 2 | 1 | Низький (2) |
| 4 | Відсутність документації | 2 | 1 | Низький (2) |
| 5 | Нові технології – не підтримають минулі версії | 2 | 2 | Середній (4) |
| 6 | Зміна структури даних | 2 | 1 | Низький (2) |
| 7 | Помилки у запитах до БД. | 3 | 1 | Середній (3) |
| 8 | Відсутність підключення до БД | 3 | 1 | Середній (3) |
| 9 | Недостатні знання проектування БД | 3 | 1 | Середній (3) |
| 10 | Недостатня кількість даних | 3 | 2 | Високий (6) |
| 11 | Помилка користувачів при виборі фільтрів | 3 | 2 | Високий (6) |
| 12 | Помилка при побудові таблиці фактів | 3 | 2 | Високий (6) |

Для 2 ризиків з найбільшими рангами були підготовлені картки з детальним описом ризику та стратегії реакції на даний ризик. Опис ризиків представлено на рис. 2.5-2.6.

|  |  |
| --- | --- |
| Номер: R-2 | Категорія: Технологічний |
| Причина: Недолік досвіду, відсутність мотивації | Умови: Відсутність досвіду планування |
| Наслідки: Зміна завдання на диплом | Вплив: Перевищення часу на розробку |
| Ймовірність: Середня | Ступень впливу: Критична |
| Близькість: Скоро | Ранг: Високий - 6 |
| Стратегії реакції на ризик:  Зниження – повторний аналіз завдання на розробку. | |

Рисунок 2.5 – Опис ризику «Недостатній аналіз завдання на розробку»

|  |  |
| --- | --- |
| Номер: R-5 | Категорія: Технологічний |
| Причина: Недостатня підтримка технологій, невелика популярність технології. | Умови: Відсутня документація з використання технології |
| Наслідки: Складність вивчення технологій | Вплив: Перевищення часу на розробку |
| Ймовірність: Середня | Ступень впливу: Середня |
| Близькість: Дуже скоро | Ранг: Середній - 4 |
| Стратегії реакції на ризик:  Ухилення – зміни технології на іншу  (Ухвалення – продовжити використання технології, будь якою ціною.) | |

Рисунок 2.6 – Опис ризику «Відсутня документація»

## 2.4 Висновки

У даному розділі роботи, було сплановано виконання проекту, проведена оцінка роботи, а також визначені найвпливовіші ризики проекту.

# 3 ПРОЕКТУВАННЯ СИСТЕМИ ФОРМУВАННЯ ВИХІДНИХ ДЛЯ АНАЛІЗУ

## Проектування структури та організації класів

Структура системи представлена діаграмою концептуальних класів, яка показана на рис. 3.6.

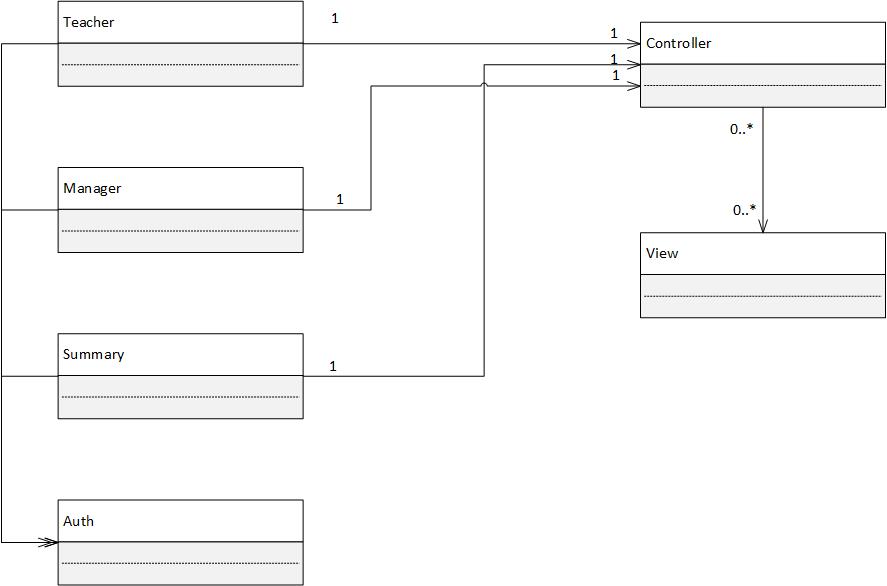


Рисунок 3.6 – Діаграма концептуальних класів

Клас Auth отримує відповідає за авторизацію користувача.

Клас Manager – являє собой модель сутності Менеджер.

Клас Teacher – являє собой модель сутності Вчитель

Клас Summary – являє собой модель сутності Керівник

Клас Controller - є контролером, оброблює інформацію, яку получає від вищееозначенных класів.

## Проектування системи

Проектування здійснювалось з урахуванням варіантів використання. Далі представлені діаграми послідовностей для важливих варіантів використання.

На рис.3.1 показана діаграма послідовностей для ВВ «Авторизація».

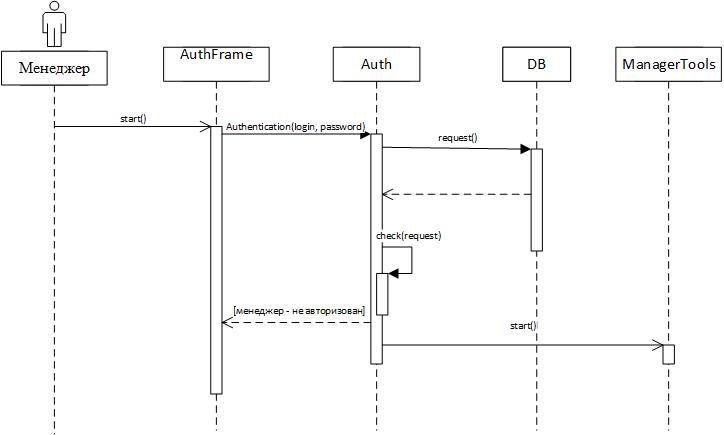


Рисунок 3.1 – Діаграма послідовностей для ВВ «Авторизація»

Користувач ініціює Клас AuthFrame, який являється представником моделі, де вводить свої дані для авторизації,далі викликається функція Authentification(), яка належить класу Auth(), ця функція приймає у логін і пароль у якості параметрів, які далі передаються у запит до БД. Який здійснює перевірку даних для авторизації. Якщо дані введені користувачем і у БД співпадають – то запуститься об’єкт класу ManagerTool.

На рис. 3.2 показана діаграма послідовностей для ВВ «Створити запис»

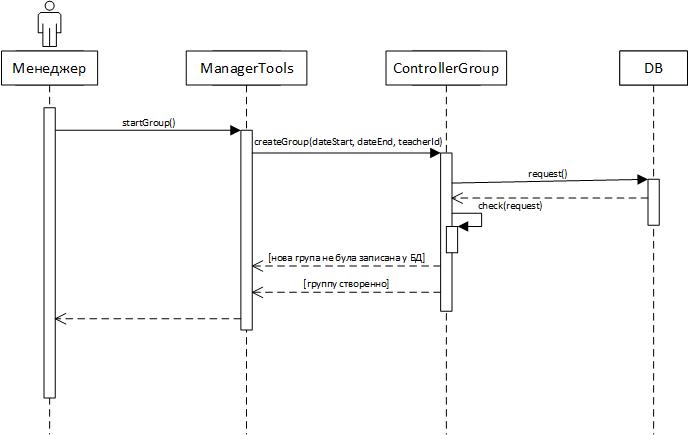


Рисунок 3.2 – Діаграма послідовностей для ВВ «Створити запис»

Коли користувач авторизувався у ролі менеджера, він получить доступ до вікна, де має можливість створювати нових студентів, групи, або редагувати ці записи. У зв’язку з великою кількістю однакових діаграм, буде представлено по одній діаграмі кожного типу.

Коли користувач ввів необхідні дані групи и натиснув кнопку «Додати», клас ManagerTools запустить об’єкт ControllerGroup, який дозволить обробити запит на додавання даних до БД.

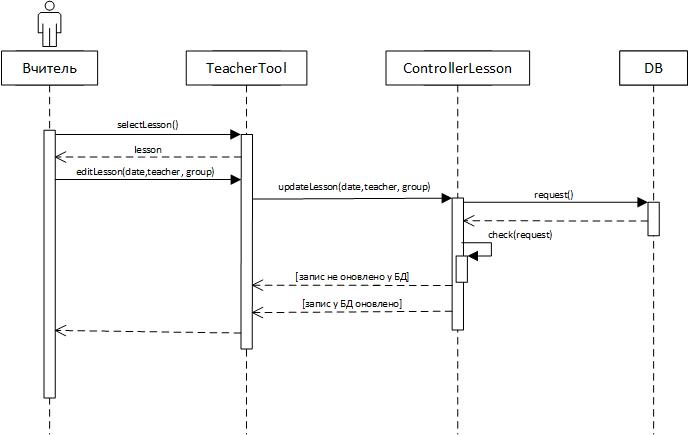


Рисунок 3.3 – Діаграма послідовностей для ВВ «Редагувати запис»

ОбьектклассуControllerLesson, отримає дані, які потрібно обновити у БД, від класу TeacherTool, і за допомогою запиту до БД, оновлює дані.

На рис. 3.4 зображена діаграма послідовностей для ВВ «Імпортувати таблицю фактів».

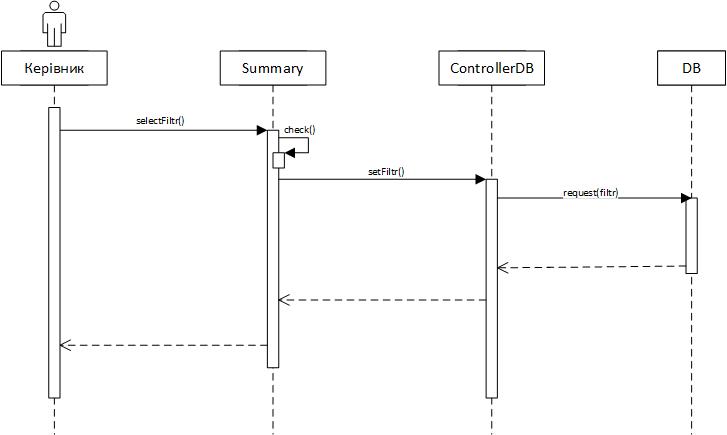


Рисунок 3.4 – Діаграма послідовностей для ВВ «Імпортувати таблицю фактів»

Обьект класу ControllerDBотримає параметри фільтрів від користувача, і після запиту до БД повертає дані для формування таблиці фактів.

На рис. 3.5 зображена діаграма послідовностей для ВВ «Експорт таблиці фактів у звідну таблицю».

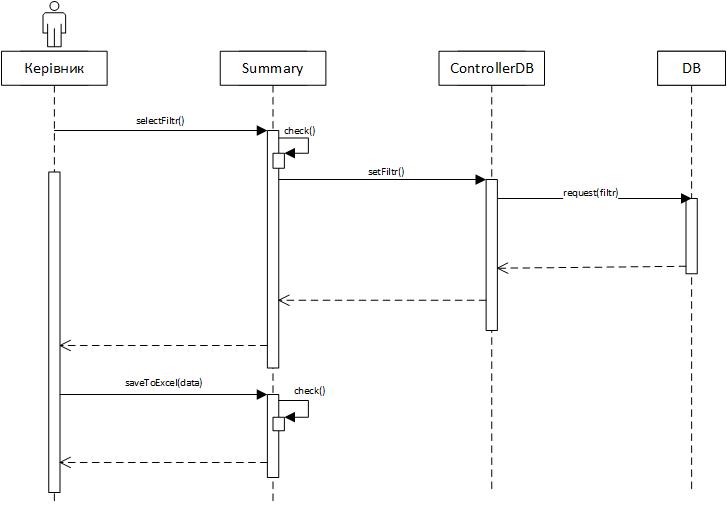


Рисунок 3.5 – Діаграма послідовностей для ВВ «Експорт таблиці фактів у звідну таблицю»

Об’єкт класу Summary отримує запити від клієнта, а саме параметри для формування зведеної таблиці, які потім передає у клас ControllerDB. Який в свою чергу, повертає дані зведеної таблиці.

## 

## Проектування бази даних

### 3.3.1 Структура бази даних з метаданими.

### У таблиці «discipline» зберігається інформація про предмети. Структура даної таблиці наведена у таблиці 3.1.

Таблиця 3.1 – Структура таблиці «discipline»

|  |  |  |  |
| --- | --- | --- | --- |
| Назва поля | Тип даних | Ключ | Призначення |
| id | int | первинний (PK) | Унікальний ідентифікатор предмету |
| name | varchar | - | Назва предмету |

У таблиці «students» зберігається інформація про студентів. Структура даної таблиці наведена у таблиці 3.2.

Таблиця 3.2 – Структура таблиці «students»

|  |  |  |  |
| --- | --- | --- | --- |
| Назва поля | Тип даних | Ключ | Призначення |
| id\_student | int | первинний (PK) | Унікальний ідентифікатор студента |
| first\_name | varchar | - | Ім’я студента |
| last\_name | varchar | - | Фамілія студента |
| date\_birthday | date | - | Дата народження |

У таблиці «teachers» зберігається інформація про вчителів. Структура даної таблиці наведена у таблиці 3.3.

Таблиця 3.3 – Структура таблиці «teachers»

|  |  |  |  |
| --- | --- | --- | --- |
| Назва поля | Тип даних | Ключ | Призначення |
| id | int | первинний (PK) | Унікальний ідентифікатор вчителя |
| first\_name | varchar | - | Ім’я вчителя |
| last\_name | varchar | - | Фамілія вчителя |

У таблиці «groups» зберігається інформація про навчальні групи. Структура даної таблиці наведена у таблиці 3.4.

Таблиця 3.4 – Структура таблиці «groups»

|  |  |  |  |
| --- | --- | --- | --- |
| Назва поля | Тип даних | Ключ | Призначення |
| id | int | первинний(PK) | Унікальний ідентифікатор навчальної групи |
| start\_date | datetime | - | Дата початку занять навчальної групи |
| finish\_date | datetime | - | Дата завершення занять навчальної групи |
| id\_discipline | int | зовнішній (FK) | Ідентифікатор предмету |
| id\_teacher | int | зовнішній (FK) | Ідентифікатор вчителя |
| id\_level | int | зовнішній (FK) | Ідентифікатор рівня |

У таблиці «group\_student» зберігається інформація про студентів та їх належність до певної навчальні групи. Структура даної таблиці наведена у таблиці 3.5.

Таблиця 3.5 – Структура таблиці «group\_student»

|  |  |  |  |
| --- | --- | --- | --- |
| Назва поля | Тип даних | Ключ | Призначення |
| id | int | первинний(PK) | Унікальний ідентифікатор групи та студента |
| id\_group | int | зовнішній (FK) | Ідентифікатор-посилання на відповідну групу |
| id\_student | int | зовнішній (FK) | Ідентифікатор-посилання на відповідного студента |

У таблиці «lesson» зберігається інформація про заняття. Структура даної таблиці наведена у таблиці 3.6.

Таблиця 3.6 – Структура таблиці «lesson»

|  |  |  |  |
| --- | --- | --- | --- |
| Назва поля | Тип даних | Ключ | Призначення |
| id | int | первинний(PK) | Унікальний ідентифікатор заняття |
| name | char | - | Назва(тема) заняття |
| lesson\_date | date | - | Дата проведення заняття |

У таблиці «level» зберігається інформація про рівень навчального курсу. Структура даної таблиці наведена у таблиці 3.7.

Таблиця 3.7 – Структура таблиці «level»

|  |  |  |  |
| --- | --- | --- | --- |
| Назва поля | Тип даних | Ключ | Призначення |
| id\_level | int | первинний(PK) | Унікальний ідентифікатор рівня навчального курсу |
| name | char | - | Назва рівню |

У таблиці «pass» зберігається інформація про пропуски занять студентами. Структура даної таблиці наведена у таблиці 3.8.

Таблиця 3.8 – Структура таблиці «pass»

|  |  |  |  |
| --- | --- | --- | --- |
| Назва поля | Тип даних | Ключ | Призначення |
| id | int | первинний(PK) | Унікальний ідентифікатор групи та студента |
| id\_student | int | зовнішній (FK) | Ідентифікатор-посилання на відповідного студента |
| id\_lesson | int | зовнішній (FK) | Ідентифікатор-посилання на відповідне заняття |

У таблиці «test» зберігається інформація про тестування. Структура даної таблиці наведена у таблиці 3.9.

Таблиця 3.9 – Структура таблиці «test»

|  |  |  |  |
| --- | --- | --- | --- |
| Назва поля | Тип даних | Ключ | Призначення |
| id | int | первинний(PK) | Унікальний ідентифікатор заняття |
| name | varchar | - | Назва або тема тесту |
| dateTest | datetime | - | Дата та час проведення тесту |

У таблиці «testdegrees» зберігається інформація про результати тестуваня. Структура даної таблиці наведена у таблиці 3.10.

Таблиця 3.10 – Структура таблиці «testdegrees»

|  |  |  |  |
| --- | --- | --- | --- |
| Назва поля | Тип даних | Ключ | Призначення |
| id | int | первинний(PK) | Унікальний ідентифікатор результату тестування |

Продовження таблиці 3.10

|  |  |  |  |
| --- | --- | --- | --- |
| Назва поля | Тип даних | Ключ | Призначення |
| id\_student | int | зовнішній (FK) | Ідентифікатор-посилання на відповідного студента |
| id\_test | int | зовнішній (FK) | Ідентифікатор-посилання на відповідний тест |
| degrees | int | - | Бали з тесту |

На рис. 3.1 представлена реляційна модельБД.

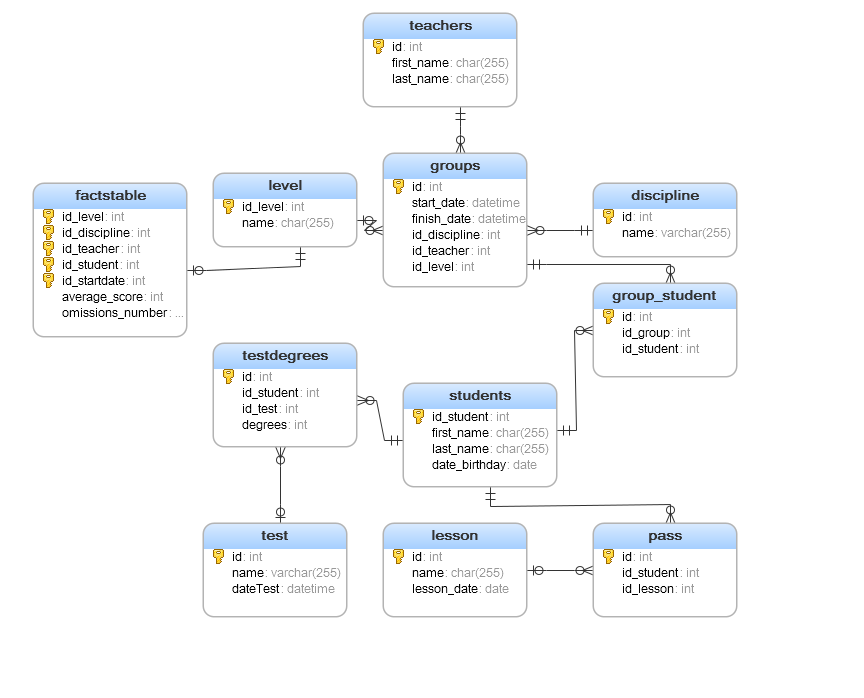


Рисунок 3.1 – реляційна модель БД

## Висновки

В рамках даної цього розділу було спроектовано систему, побудована діаграма класів, а також виконано проектування БД, описано структура БД, у рамках проекту.

# 4ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ ФОРМУВАННЯ ВИХІДНИХ ДЛЯ АНАЛІЗУ

## 4.1Опис програмних класів

Логічне представлення системи відображено діаграмою програмних класів, яка показана на рис. 4.1.

Клас Controllerотримує запити від користувача та повертає відповіді користувача.

Клас ControllerGroup отримує запити від ManagerTool, пов’язані створенням та редагуванням. Являється нащадком класу Controller.

Методи класу:

* createGroup() – створення нової групи у БД;
* editGroup() – редагування групи;
* loadData() - отримання даних груп, для відображення у таблиці.

Клас ControllerStudent отримує запити від класів ManagerTooliTeacherTool, пов’язані створенням та редагуванням студентів. Являється нащадком класу Controller.

Методи класу:

* createStudent () – створення нового студента у БД;
* editStudent () – редагування студента;
* getTable() - отримання даних студента, для відображення у таблиці.

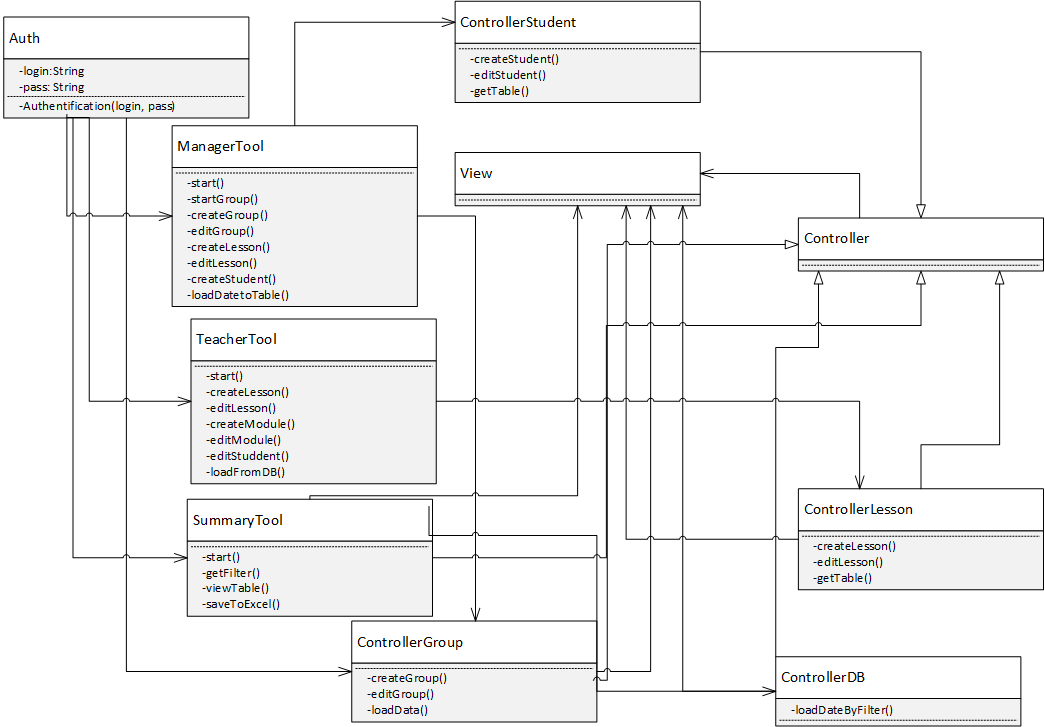


Рисунок 4.1 – Діаграма програмних класів

Клас ControllerLesson отримує запити від TeacherTool, які пов’язані з створенням та редагуванням занять. Являється нащадком класуController.

Методи класу:

* createLesson () – створення нового заняття у БД;
* editLesson () – редагування заняття ;
* getTable() - отримання даних занять, для відображення у таблиці.

Клас ControllerDB отримує запити від SummaryTool, пов’язані з формуванням звідної таблиці.

Методи класу:

* loadDataByFilter () – отримання даних з БД;

Клас View відповідає за генерацію та оновлення інтерфейсу.

## 4.2 Отримання зведених даних

Для побудови таблиці фактів використовується наступний запит до БД, який повертає набір необхідних даних

SELECT discipline.name as Предмет, teachers.last\_name as Имя\_преподавателя, teachers.first\_name as Фамилия\_преподавателя, students.last\_name as Имя\_студента, students.first\_name as Фамилия\_студента, "level.name as Уровень, groups.start\_date as Дата\_начала, AVG(testdegrees.degrees) AS AverageScore, COUNT(DISTINCT pass.id) AS PassCount FROM discipline LEFT OUTER JOIN groups ON discipline.id = groups.id\_discipline LEFT OUTER JOIN teachers ON teachers.id = groups.id\_teacher LEFT OUTER JOIN group\_student ON group\_student.id\_group = groups.id LEFT OUTER JOIN students ON students.id\_student = group\_student.id\_student LEFT OUTER JOIN level ON level.id\_level = groups.id\_level LEFT OUTER JOIN testdegrees ON testdegrees.id\_student = group\_student.id\_student LEFT OUTER JOIN pass ON pass.id\_student = group\_student.id\_student

## 4.3 Алгоритми отримання та підготовлення даних для відображення

На рисунку 4.2 представлено алгоритм функції для формування таблиці фактів і збереження звідної таблиці.

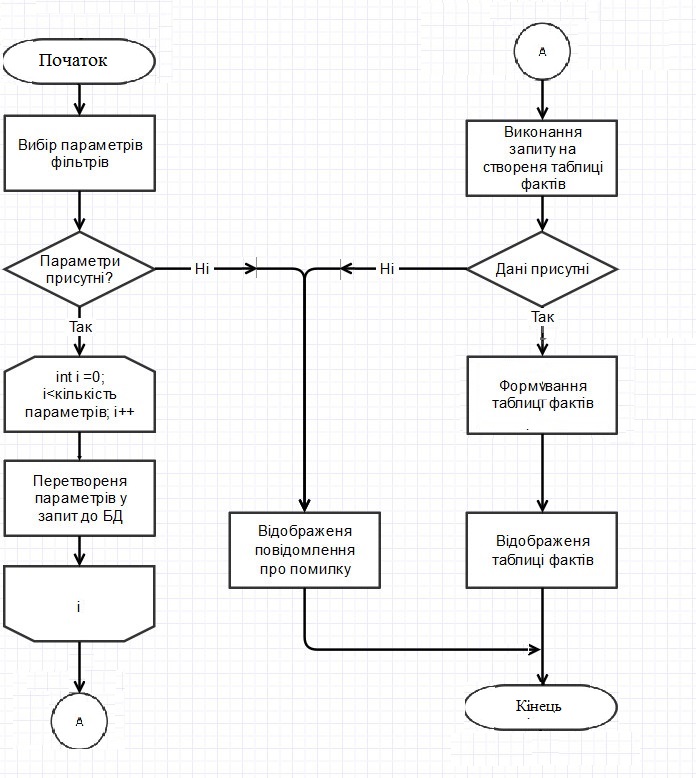


Рисунок 4.2 – Алгоритм функції «Для формування таблиці фактів і збереження звідної таблиці»

Складність алгоритму дорівнює, де *n* – кількість вхідних елементів.

## 4.4Опис використаних технологій

В якості мови програмування була обрана мова програмування Java[4]. Вибір мови програмування обумовлюється наступними причинами: незалежність від операційної системи, що дозволяє не змінювати код при переносу програми з одної машини на іншу; популярність та розмір спільноти мови позначає багатий вибір бібліотек, готових рішень та зрозумілу документацію.

Java – об'єктно-орієнтована мова програмування, що розробляється компанією SunMicrosystems з 1991 року і офіційно випущена 23 травня 1995 року. Спочатку нова мова програмування мала назву Oak (JamesGosling) і розроблявся для побутової електроніки, але згодом була перейменована в Java і стала використовуватися для написання аплетів, застосувань і серверного програмного забезпечення.

В якості робочого середовища розробки, була використано середовище NetBeans IDE [5]

В якості сховища даних було вирішено використовувати систему управління базами даних MySQL [7], яка забезпечує високій рівень надійності та гарантує цілісність даних за рахунок транзакцій.

Після аналізу отриманої на етапі проектування архітектури було вирішено на етапі реалізації використовувати шаблон MVC.

## 4.5 Висновки

У даному розділі було описана програмна реалізація системи формування даних для подальшого аналізу. Був проведений аналіз програмних класів, описано алгоритми доступу до даних, з подальшим їх отриманням.

# 5ТЕСТУВАННЯ СИСТЕМИ ФОРМУВАННЯ ВИХІДНИХ ДЛЯ АНАЛІЗУ

## Тестування функціональності системи

Для тестування функціональності системи проведено функціональне тестування. Функціональне тестування базується на аналізі специфікацій функціональності окремих компонентів або програмного продукту в цілому. Функціональні тести створюються на основі опису прецедентів з урахуванням основного та альтернативного сценаріїв й описуються у вигляді тестових варіантів.

Тестові варіанти описують сукупність кроків, конкретних умов та параметрів, необхідних для перевірки реалізації функції або її частини.

У табл. 5.1-5.7 представлені тестові варіанти для варіантів використання системи формування даних

Таблиця 5.1 – Тестові варіанти ВВ «Авторизація»

|  |  |  |
| --- | --- | --- |
| Дія | Очікуваний результат | Результат тестування |
| Запит авторизації | Інтерфейс вводу даних для авторизації | Успішно |
| Введення некоректних даних | Помилка про некоректність даних | Успішно |
| Введення коректних даних | Доступ до функціоналу авторизованого користувача | Успішно |

Таблиця 5.2 – Тестові варіанти ВВ «Створити запис»

|  |  |  |
| --- | --- | --- |
| Дія | Очікуваний результат | Результат тестування |
| Створити запис | Інтерфейс вводу даних для додавання у БД | Успішно |
| Дані не були введені | Помилка про відсутність даних | Успішно |
| Введення повного набору даних | Запис до БД було додано | Успішно |

Таблиця 5.3 – Тестові варіанти ВВ «Редагувати запис»

|  |  |  |
| --- | --- | --- |
| Дія | Очікуваний результат | Результат тестування |
| Запит на оновлення даних | Інтерфейс редагування даних | Успішно |
| Інформація для оновлення, не була змінена | Повідомлення, що оновлення не відбувається | Успішно |
| Нова інформація – невірного формату | Повідомлення, про помилкові дані. | Успішно |
| Введення інформації вірного формату і унікальній від старої. | Повідомлення що данні оновлені | Успішно |

Таблиця 5.4 – Тестові варіанти ВВ «Імпортувати данні у таблицю фактів»

|  |  |  |
| --- | --- | --- |
| Дія | Очікуваний результат | Результат тестування |
| Імпорт даних у таблицю фактів | Інтерфейс таблиці фактів. | Успішно |
| Вибрано недостатньо фільтрів для таблиці фактів | Помилка про недостатню кількість параметрів | Успішно |
| Введення усіх необхідних параметрів | Таблиця фактів сформована. | Успішно |

Таблиця 5.5 – Тестові варіанти ВВ «Експорт у звідну таблицю Excel»

|  |  |  |
| --- | --- | --- |
| Дія | Очікуваний результат | Результат тестування |
| Запит на збереження | Інтерфейс таблиці фактів. | Успішно |
| Вибір параметрів звідної таблиці | Збереження таблиці факті з вибраними параметрами | Успішно |
| Параметри звідної таблиці не вибрані | Звідна таблиця збережена з параметрами по умовчанню. | Успішно |

## 5.2Приклад використання системи

Початок роботи програми починається з вікна авторизації, де користувач має ввести свій логін та пароль (рис. 5.1).

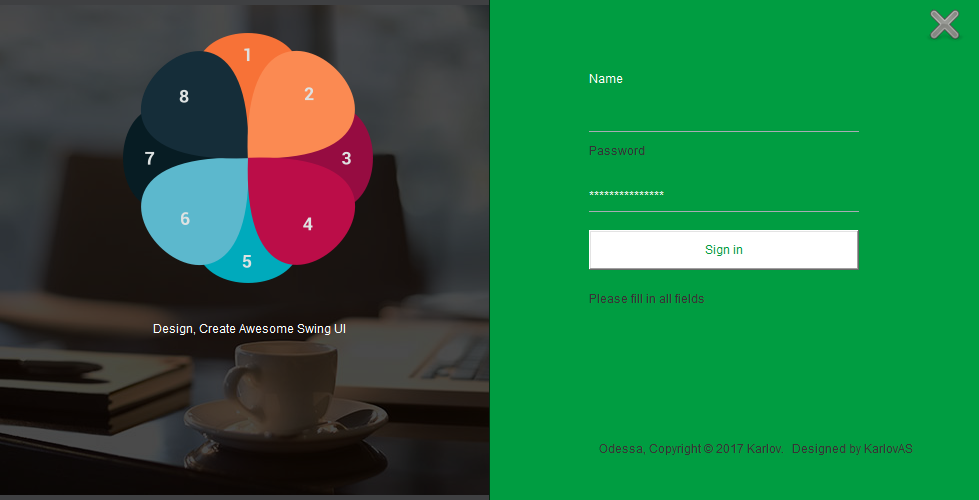


Рисунок 5.1 – Скріншот екрану програми з вікном авторизації

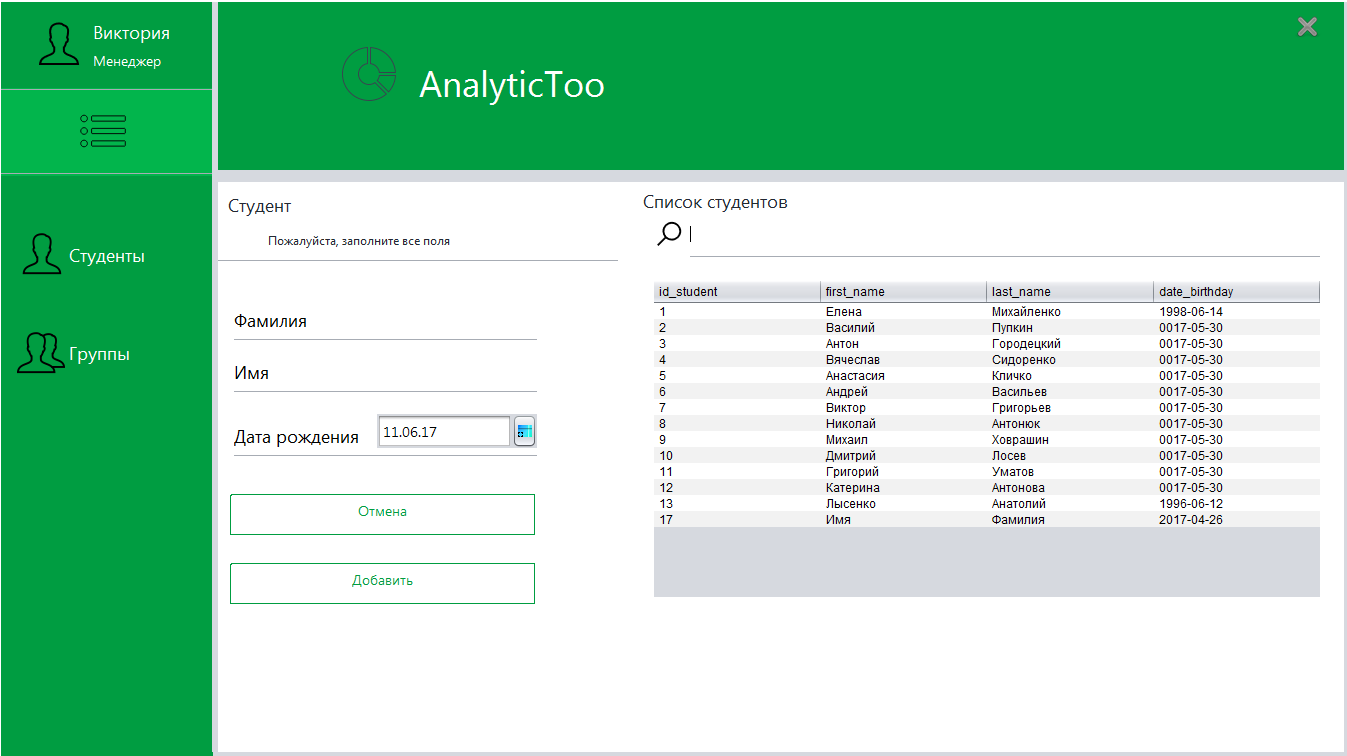
Після авторизації, якщо користувач зайшов в систему в якості менеджера, йому відкривається вікно роботи зі списком студентів, де він може додати нового студента до списку, або знайти та редагувати дані вже присутнього у базі даних (рис. 5.1). 

Рисунок 5.2 – Скріншот екрану програми для менеджера в режимі роботи зі списком студентів

Також, вибравши пункт «Группы» менеджер може перейти до роботи з групами – створення або редагування групи. В режимі створення нової групи менеджер може обрати викладача, предмет, дату початку та закінчення курсу (рис. 5.3).

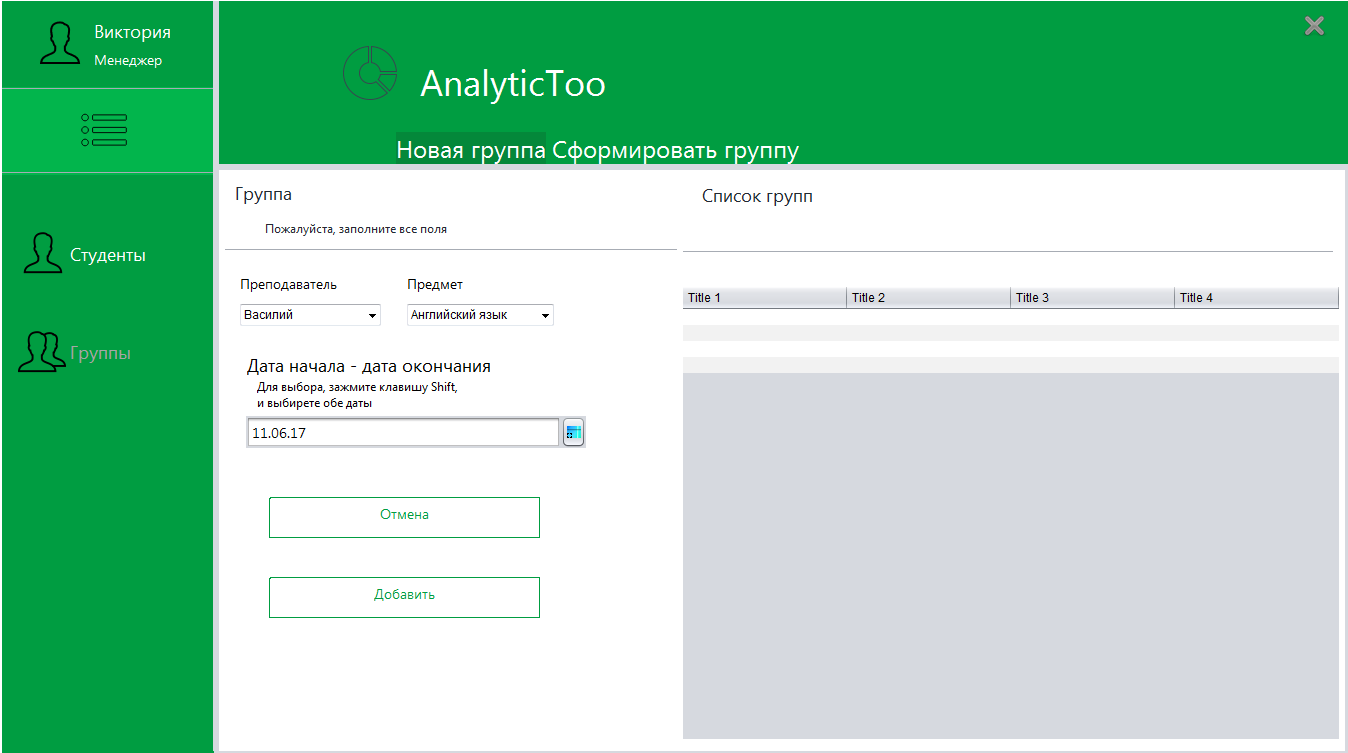


Рисунок 5.3 – Скріншот екрану програми для менеджера в режимі роботи зі списком груп

Натиснувши зверху пункт меню «» менеджер переходить у режим формування нової групи, де він може обрати групу та додати до неї студентів (рис. 5.7).

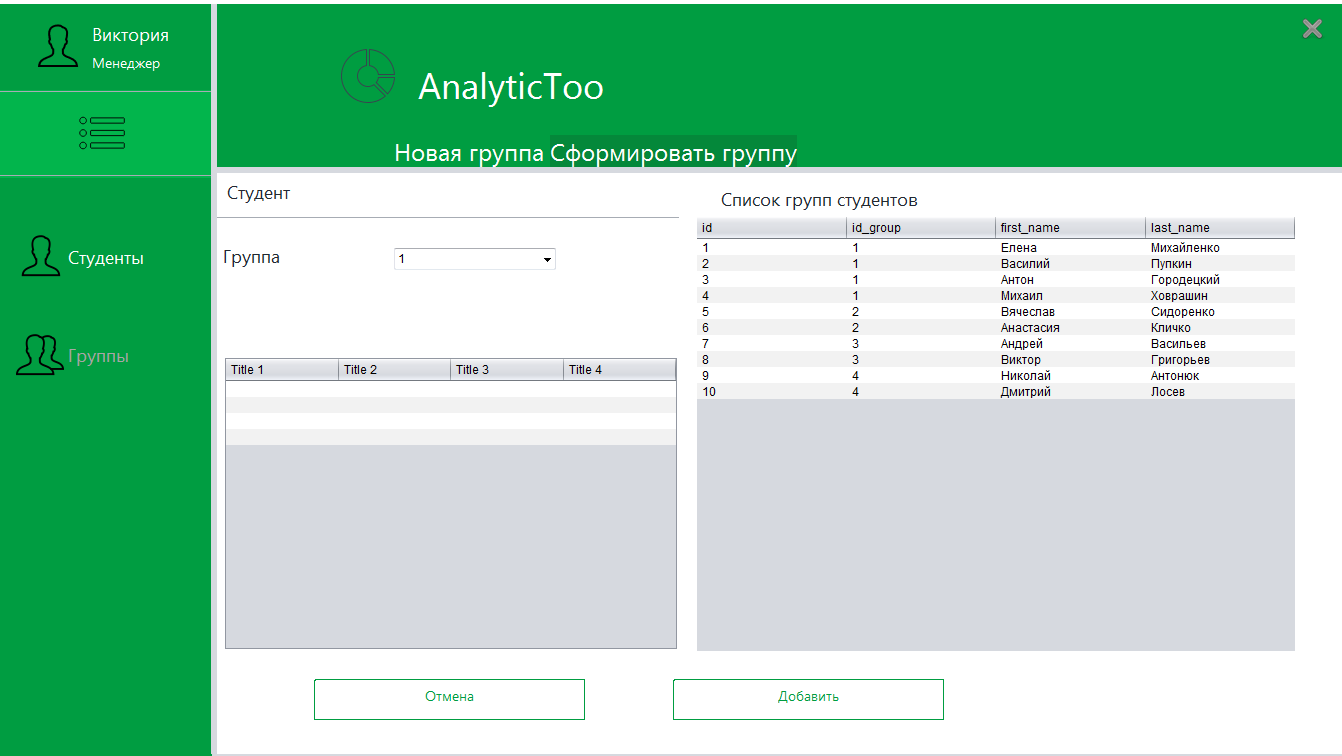


Рисунок 5.4 – Скріншот екрану програми для менеджера в режимі створення нової групи

Також, менеджер має можливість отримати підсумкову та зведену таблиці, обравши необхідні параметри для відповідного сортування результатів (рис 5.5).

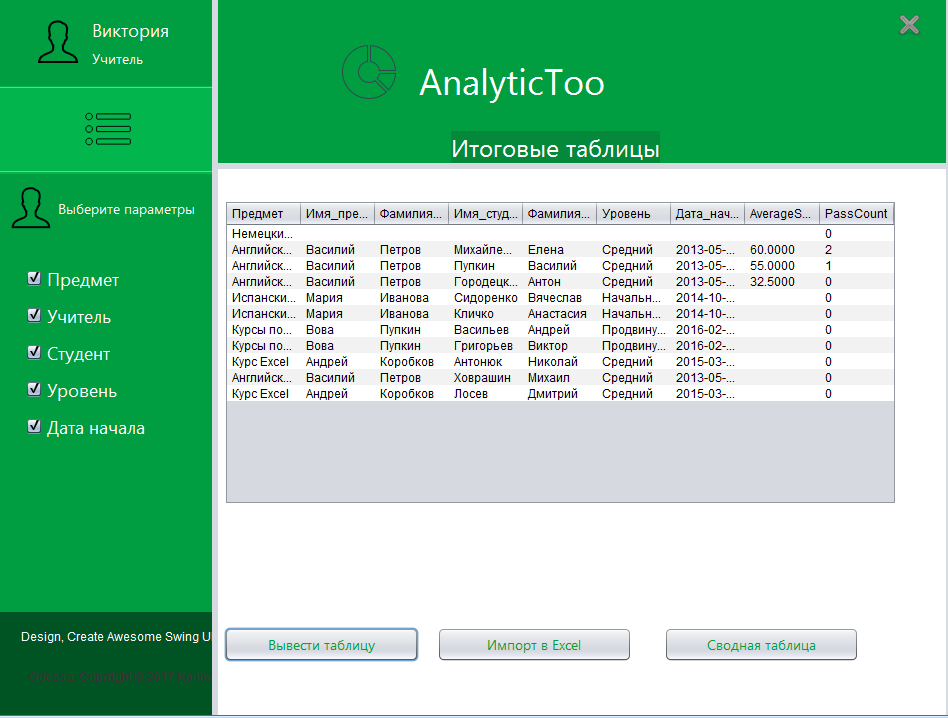


Рисунок 5.5 – Скріншот екрану програми в режимі перегляду підсумкової таблиці

За необхідності, отримані результати можна імпортувати в Microsoft Excel (рис 5.6).

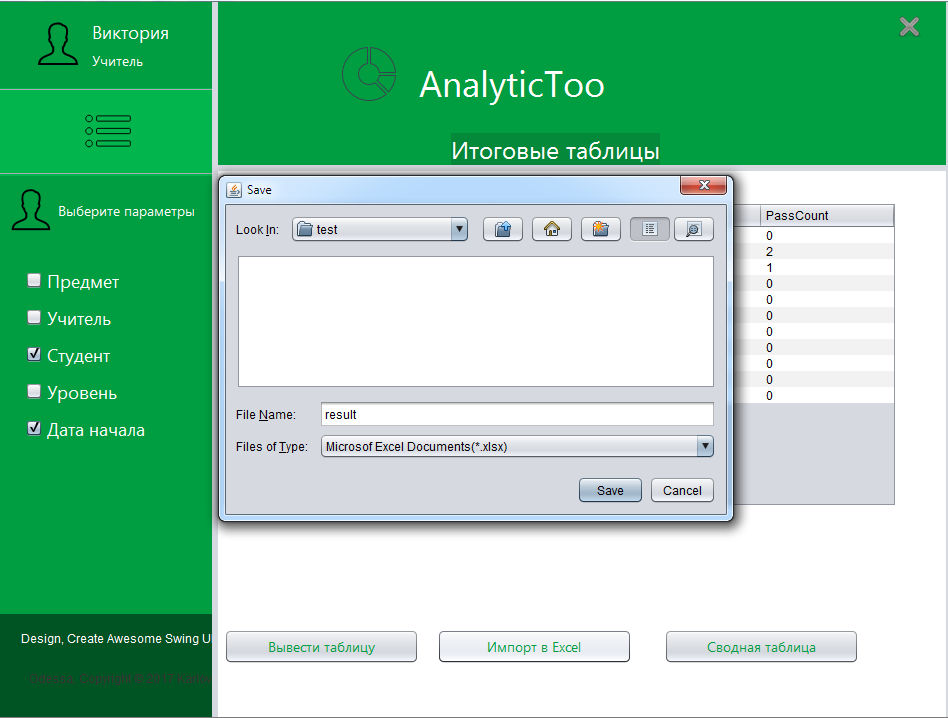


Рисунок 5.6 – Скріншот екрану програми, імпорт підсумкової таблиці в Microsoft Excel

Також, є можливість побудувати звідну таблицю Microsoft Excel, обравши необхідні параметри її побудови (рис 5.7).

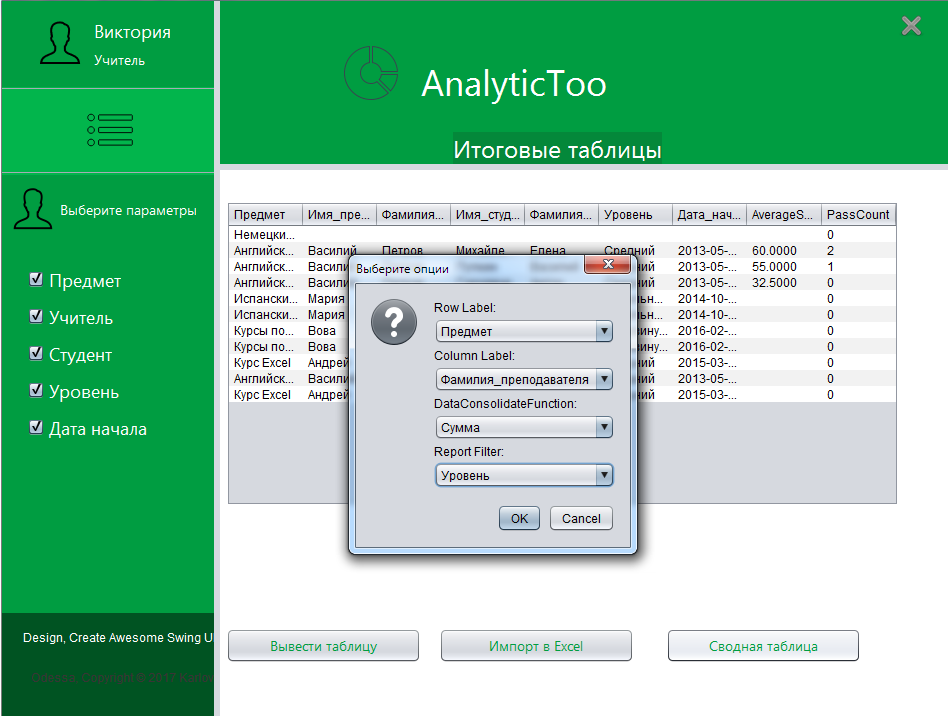


Рисунок 5.7 – Скріншот екрану програми, налаштування параметрів побудови звідної таблиці Microsoft Excel

## 5.3 Експеримент

Після впровадження системи у використання було проведено експеримент.

Протягом місяця сбір та формування даних проводився трьома способами.

* всі дані збирались вручну, записувалися на папері і здавалися керівнику;
* частина даних була збережена на папері, частина заносились до таблиць Excel і також здавалось керівнику;
* усі дані були занесені у БД за допомогою системи.

За умови використання програми час, потрачений на сбір даних, зменшився на 80%. Керівнику достатньо було створити таблицю фактів за вибраними параметрами, та імпортувати у зведену таблицю, на даний момент відпала потреба витрати часу на створення паперових звітів, а також їх подальше зберігання. Таким чином, було експериментально доведено, що система підвищує ефективність управління та аналізу навчальних курсів.

# ВИСНОВКИ

В ході роботі спроектована та розроблена система формування вхідних даних для збереження, збору та аналізу даних. Спочатку була виконана специфікація вимог до системи, а саме були сформульовані та описані функціональні вимоги у вигляді варіантів використання та нефункціональні вимоги до системи.

Також виконано планування виконання проекту. Проведена оцінка тривалості розробки, складений план робіт проекту, та проаналізовані ризики проекту.

Виконане проектування системи, в ході якого описана архітектура серверної частини та спроектоване сховище даних.

Також виконана програмна реалізація системи, в ході якої описані програмні класи, складені алгоритми основних функцій системи та доступу до даних, оцінена алгоритмічна складність, а також описані використані технології.

Також проведено тестування програми. Проведено функціональне тестування системи, результати якого показали, що система відповідає усім вимогам. Також представлений приклад використання системи.

Визначено у експерименті, програма показала приріст ефективності у 80%, на даний момент після впровадження програми, відпала потреба у формуванні великої кількості паперових звітів, а також їх структурування для подальшого аналізу.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. UML Специальный справочник. usecase (вариант использования, прецедент)[Електронний ресурс] – Режим доступу: www / URL: http://ooad.asf.ru/standarts/UML/spr/Use\_case.aspx– Загол. з екрану.
2. Chen, L. Characterizing Architecturally Significant Requirements /

C. Lianping, А. Muhammad, BasharNuseibeh– 2013.

1. Java (programminglanguage):[Електронний ресурс] – Режим доступу: <https://en.wikipedia.org/wiki/Java_(programming_language)>– Загол. з екрану.
2. NetBeansIDE [Електронний ресурс] – Режим доступу: www / URL: https://www.jetbrains.com/phpstorm/ – Загол. з екрану.
3. ApacheHTTPServerProject [Електронний ресурс] – Режим доступу: <https://netbeans.org/>– Загол. з екрану.
4. MySQLDocumentation [Електронний ресурс] – Режим доступу: www / URL: https://dev.mysql.com/doc/– Загол. з екрану.
5. Ларман К. Застосування UML і шаблонів проектування. Введення в ГО аналіз, проектування і уніфікований процес / Ларман К. – : Пер. С англ. 2-е видання, – М.:Видавничий будинок«Вильямс», 2002. – 624с.
6. Зандстра М.Об'єкти, шаблони і методики програмування /

Зандстра М.: Пер. С англ. 2-е видання. – М.:Видавничий будинок«Вильямс», 2011. – 724с.

1. Шилдт Г. Java 8. Полное руководство, 9-е видання/ Герберт Шилдт – М: Видавничий будинок«Вильямс», 2015. – 1376 с.

# ДОДАТОК

# private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

# // TODO add your handling code here:

# if (!(Level.isSelected()||Course.isSelected()||Teacher.isSelected()||Student.isSelected()||StartDate.isSelected()))

# JOptionPane.showMessageDialog(null,"Выберите хотя бы одно измерение!");

# else

# {

# try {

# String SQLQuery="SELECT ";

# if (Course.isSelected()) SQLQuery+="discipline.name as Предмет, ";

# if (Teacher.isSelected()) SQLQuery+="teachers.last\_name as Имя\_преподавателя, teachers.first\_name as Фамилия\_преподавателя, ";

# if (Student.isSelected()) SQLQuery+="students.last\_name as Имя\_студента, students.first\_name as Фамилия\_студента, ";

# if (Level.isSelected()) SQLQuery+="level.name as Уровень, ";

# if (StartDate.isSelected()) SQLQuery+="groups.start\_date as Дата\_начала, ";

# SQLQuery+="AVG(testdegrees.degrees) AS AverageScore, \n" +

# "COUNT(DISTINCT pass.id) AS PassCount\n"+

# "FROM discipline LEFT OUTER JOIN\n"+

# "groups ON discipline.id = groups.id\_discipline LEFT OUTER JOIN \n"+

# "teachers ON teachers.id = groups.id\_teacher LEFT OUTER JOIN\n" +

# "group\_student ON group\_student.id\_group = groups.id LEFT OUTER JOIN \n" +

# "students ON students.id\_student = group\_student.id\_student LEFT OUTER JOIN\n" +

# "level ON level.id\_level = groups.id\_level LEFT OUTER JOIN\n" +

# "testdegrees ON testdegrees.id\_student = group\_student.id\_student LEFT OUTER JOIN\n"+

# "pass ON pass.id\_student = group\_student.id\_student ";

# if (Student.isSelected()) SQLQuery+="GROUP BY students.id\_student;";

# //if (Student.isSelected()) SQLQuery+="GROUP BY students.id\_student;";

# System.out.println("SQLQuery="+SQLQuery);

# SummaryTable.setModel(DbUtils.resultSetToTableModel(db.loadSummaryTable(SQLQuery, null)));

# }

# catch (Exception ex) {

# JOptionPane.showMessageDialog(null,ex);

# }

# }

# }

# private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {

# // TODO add your handling code here:

# try {

# // TODO add your handling code here:

# Path FileName = getPathtoSaveFile();

# if (FileName!=null)

# importToExcel(SummaryTable,FileName);

# } catch (IOException ex) {

# //JOptionPane.showMessageDialog(null,ex);

# Logger.getLogger(SummaryFrameRework.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

# }

# }

# private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {

# int colCount = SummaryTable.getColumnCount();

# String[] valuesList = new String[colCount];

# for(int i=0;i<colCount;i++)

# //valuesList[i]=String.valueOf(i);

# valuesList[i]=String.valueOf(SummaryTable.getModel().getColumnName(i));

# ArrayList<String> resultList = showComboBoxDialog(valuesList);

# if (resultList!= null)

# {

# try {

# Path FileName = getPathtoSaveFile();

# if (FileName!=null)

# {

# createPivotTable(SummaryTable,resultList,FileName);

# JOptionPane.showMessageDialog(this,"Сводная таблица создана!");

# }

# } catch (IOException ex) {

# Logger.getLogger(SummaryFrameRework.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

# } catch (InvalidFormatException ex) {

# Logger.getLogger(SummaryFrameRework.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

# }

# }

# }

# private Path getPathtoSaveFile()

# {

# JFileChooser SaveFile = new JFileChooser();

# FileNameExtensionFilter mask = new FileNameExtensionFilter("Microsof Excel Documents(\*.xlsx)","xlsx");

# SaveFile.setFileFilter(mask);

# SaveFile.setSelectedFile(new File("result"));

# int result = SaveFile.showSaveDialog(this);

# if (result == JFileChooser.APPROVE\_OPTION) {

# File file = SaveFile.getSelectedFile();

# Path FilePath = Paths.get(file+".xlsx");

# return FilePath;

# }

# return null;

# }

# private static void importToExcel(JTable table, Path path) throws FileNotFoundException, IOException {

# new WorkbookFactory();

# Workbook wb = new XSSFWorkbook(); //Excel workbook

# Sheet sheet = wb.createSheet(); //WorkSheet

# Row row = sheet.createRow(2); //Row created at line 3

# TableModel model = table.getModel(); //Table model

# Row headerRow = sheet.createRow(0); //Create row at line 0

# for(int headings = 0; headings < model.getColumnCount(); headings++){ //For each column

# headerRow.createCell(headings).setCellValue(model.getColumnName(headings));//Write column name

# }

# for(int rows = 0; rows < model.getRowCount(); rows++){ //For each table row

# for(int cols = 0; cols < table.getColumnCount(); cols++){ //For each table column

# row.createCell(cols).setCellValue(String.valueOf(model.getValueAt(rows, cols)));

# //row.createCell(cols).setCellValue(model.getValueAt(rows, cols).toString()); //Write value

# }

# //Set the row to the next one in the sequence

# row = sheet.createRow((rows + 3));

# }

# wb.write(new FileOutputStream(path.toString()));//Save the file

# }

# public static void createPivotTable(JTable table, ArrayList<String> Parameters, Path path) throws FileNotFoundException, IOException, InvalidFormatException {

# XSSFWorkbook wb = new XSSFWorkbook();

# XSSFSheet sheet = (XSSFSheet) wb.createSheet();

# //Create some data to build the pivot table on

# Row row = sheet.createRow(2); //Row created at line 3

# TableModel model = table.getModel(); //Table model

# Row headerRow = sheet.createRow(0); //Create row at line 0

# for(int headings = 0; headings < model.getColumnCount(); headings++){ //For each column

# headerRow.createCell(headings).setCellValue(model.getColumnName(headings));//Write column name

# }

# for(int rows = 0; rows < model.getRowCount(); rows++){ //For each table row

# for(int cols = 0; cols < table.getColumnCount(); cols++){ //For each table column

# row.createCell(cols).setCellValue(String.valueOf(model.getValueAt(rows, cols)));//Write value

# }

# //Set the row to the next one in the sequence

# row = sheet.createRow((rows + 3));

# //System.out.println("1= "+table.getColumnCount());

# String LastRowNum = String.valueOf(table.getRowCount()+2);

# XSSFPivotTable pivotTable = sheet.createPivotTable(new AreaReference("A1:I"+LastRowNum,SpreadsheetVersion.EXCEL2007), new CellReference("K5"));

# //Configure the pivot table

# //Use first column as row label

# //System.out.println("RowLabelParam= "+Integer.valueOf(Parameters.get(0)));

# pivotTable.addRowLabel(Integer.valueOf(Parameters.get(0)));

# //Sum up the second column

# //System.out.println("ColumnLabelParam= "+Integer.valueOf(Parameters.get(1)));

# switch (Parameters.get(2)){

# case "Сумма":

# pivotTable.addColumnLabel(DataConsolidateFunction.SUM, Integer.valueOf(Parameters.get(1)));

# //System.out.println("Сумма");

# break;

# case "Среднее значение":

# pivotTable.addColumnLabel(DataConsolidateFunction.AVERAGE, Integer.valueOf(Parameters.get(1)));

# //System.out.println("Среднее значение");

# break;

# }

# //Set the third column as filter

# //pivotTable.addColumnLabel(DataConsolidateFunction.AVERAGE, 7);

# //Add filter on forth column

# //System.out.println("ReportFilter= "+Integer.valueOf((Parameters.get(3))));

# pivotTable.addReportFilter(Integer.valueOf((Parameters.get(3))));

# wb.write(new FileOutputStream(path.toString()));

# /\*FileOutputStream fileOut = new FileOutputStream("G:\\ooxml-pivottable.xlsx");

# wb.write(fileOut);

# fileOut.close();\*/

# }

# }

# public static ArrayList<String> showComboBoxDialog(String[] values) {

# ArrayList<String> ParametersList = new ArrayList<String>();

# String[] RowLabelValues = values;

# String[] ColumnLabelValues = values;

# String[] DataConsolidateFunctionValues = {"Сумма","Среднее значение"};

# String[] ReportFilterValues = values;

# JComboBox RowLabel = new JComboBox(RowLabelValues);

# JComboBox ColumnLabel = new JComboBox(ColumnLabelValues);

# JComboBox DataConsolidateFunction = new JComboBox(DataConsolidateFunctionValues);

# JComboBox ReportFilter = new JComboBox(ReportFilterValues);

# Object[] parameters = {

# "Row Label:", RowLabel,

# "Column Label:", ColumnLabel,

# "DataConsolidateFunction: ",DataConsolidateFunction,

# "Report Filter:",ReportFilter

# };

# int option = JOptionPane.showConfirmDialog(null, parameters, "Выберите опции", JOptionPane.OK\_CANCEL\_OPTION);

# if (option == JOptionPane.OK\_OPTION) {

# ParametersList.add(String.valueOf(RowLabel.getSelectedIndex()));

# ParametersList.add(String.valueOf(ColumnLabel.getSelectedIndex()));

# ParametersList.add(DataConsolidateFunction.getSelectedItem().toString());

# ParametersList.add(String.valueOf(ReportFilter.getSelectedIndex()));

# return ParametersList;

# } else {

# System.out.println("Cancelled");

# return null;

# }

# }

# /\*\*

# \* @param args the command line arguments

# \*/

# public static void main(String args[]) {

# /\* Set the Nimbus look and feel \*/

# //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">

# /\* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.

# \* For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html

# \*/

# try {

# for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {

# if ("Nimbus".equals(info.getName())) {

# javax.swing.UIManager.setLookAndFeel(info.getClassName());

# break;

# }

# }

# } catch (ClassNotFoundException ex) {

# java.util.logging.Logger.getLogger(SummaryFrameRework.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

# } catch (InstantiationException ex) {

# java.util.logging.Logger.getLogger(SummaryFrameRework.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

# } catch (IllegalAccessException ex) {

# java.util.logging.Logger.getLogger(SummaryFrameRework.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

# } catch (javax.swing.UnsupportedLookAndFeelException ex) {

# java.util.logging.Logger.getLogger(SummaryFrameRework.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

# }

# //</editor-fold>

# /\* Create and display the form \*/

# java.awt.EventQueue.invokeLater(new Runnable() {

# public void run() {

# new SummaryFrameRework().setVisible(true);

# }

# });

# }

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package Controller;

import Model.Tags;

import MySQL.javaconnect;

import java.sql.Connection;

import java.sql.Date;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.text.DateFormat;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.Calendar;

import java.util.HashMap;

import java.util.Locale;

import java.util.logging.Level;

import java.util.logging.Logger;

import javax.swing.JOptionPane;

/\*\*

\*

\* @author AlexKarlov

\*/

public class DBControllerStudent {

Connection conn = null;

ResultSet rs=null;

PreparedStatement pst =null;

Tags tag;

public DBControllerStudent() {

conn=javaconnect.ConnecrDb();

tag = new Tags();

}

public void studentTableInsert(String name, String lastName, Calendar dateBirthday){

Calendar date = dateBirthday;

java.sql.Date bDate = null;

bDate = new java.sql.Date(date.getTimeInMillis());

try {

String sql="INSERT INTO "+tag.DB\_NAME+"."+tag.STUDENT\_TABLE+"("+tag.NAME+","+tag.LASTNAME+","+tag.STUDENT\_DATEBIRTHDAY+" ) "

+ "VALUES('"+name+"', '"+lastName+"', '"+bDate+"')";

pst =conn.prepareStatement(sql);

pst.executeUpdate();

} catch (SQLException ex) {

JOptionPane.showMessageDialog(null,ex);

}

finally {

try {

rs.close();

pst.close();

}

catch (Exception e){

}

}

}

public ResultSet loadTable(String tableName, ResultSet rs){

try {

String sql = "select \* from "+tableName+" ";

pst = conn.prepareStatement(sql);

rs=pst.executeQuery(sql);

} catch (SQLException ex) {

JOptionPane.showMessageDialog(null,ex);

}

return rs;

}

public void updateTableStudent(int id, String name, String lastName, Calendar birthdayDay){

Calendar date = birthdayDay;

java.sql.Date bDate = null;

bDate = new java.sql.Date(date.getTimeInMillis());

try {

String sql = "UPDATE "+tag.DB\_NAME+"."+tag.STUDENT\_TABLE+" "

+ "set "+tag.NAME+" = '"+name+"', "+tag.LASTNAME+" = '"+lastName+"',"

+ " "+tag.STUDENT\_DATEBIRTHDAY+" = '"+bDate+"' where id\_student = '"+id+"'";

pst = conn.prepareStatement(sql);

pst.execute();

} catch (SQLException ex) {

JOptionPane.showMessageDialog(null,ex);

}

}

public HashMap getValueFromTableStudent(Integer tableClickId){

HashMap<String, Object> listValues = new HashMap<String, Object>();

try {

String sql = "SELECT \* from "+tag.DB\_NAME+"."+tag.STUDENT\_TABLE+" where id\_student='"+tableClickId+"'";

pst = conn.prepareStatement(sql);

rs = pst.executeQuery();

if(rs.next()){

listValues.put("first\_name", rs.getString("first\_name"));

listValues.put("last\_name", rs.getString("last\_name"));

listValues.put("date\_birthday", rs.getDate("date\_birthday"));

listValues.put("id", rs.getInt("id\_student"));

}

} catch (SQLException ex) {

JOptionPane.showMessageDialog(null,ex);

}

finally {

try {

rs.close();

pst.close();

}

catch (Exception e){

}

}

return listValues;

}

public ResultSet searchStudentInTable(String lastName, ResultSet rs){

String sql = "SELECT \* FROM "+tag.DB\_NAME+"."+tag.STUDENT\_TABLE+" where "+tag.LASTNAME+" = '"+lastName+"'";

try {

pst = conn.prepareStatement(sql);

rs=pst.executeQuery(sql);

} catch (SQLException ex) {

JOptionPane.showMessageDialog(null,ex);

}

return rs;

}

}

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package Controller;

import Model.Tags;

import MySQL.javaconnect;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.util.ArrayList;

import java.util.Calendar;

import java.util.HashMap;

import java.util.logging.Level;

import java.util.logging.Logger;

import javax.swing.JOptionPane;

/\*\*

\*

\* @author AlexKarlov

\*/

public class DBControllerGroup {

Connection conn = null;

ResultSet rs=null;

PreparedStatement pst =null;

Tags tag;

public DBControllerGroup() {

conn=javaconnect.ConnecrDb();

tag = new Tags();

}

public ArrayList loadToChoice(String tablename, String value){

ArrayList<String> itemList = new ArrayList<String>();

try {

String sql = "SELECT \* FROM "+tablename+"";

pst = conn.prepareStatement(sql);

rs = pst.executeQuery();

while(rs.next()){

itemList.add(rs.getString(value));

}

} catch (SQLException ex) {

Logger.getLogger(DBControllerGroup.class.getName()).log(Level.SEVERE, null, ex);

}

return itemList;

}

public void insertGroup(String teacher\_id, String discipline\_id, String start\_date, String finish\_date){

String sql = "INSERT INTO "+tag.DB\_NAME+"."+tag.GROUP\_TABLE+"()";

}

public ResultSet loadtoStudentTableGroup(String tableName, ResultSet rs){

try {

String sql = "select \* from "+tableName+" ";

pst = conn.prepareStatement(sql);

rs=pst.executeQuery(sql);

} catch (SQLException ex) {

JOptionPane.showMessageDialog(null,ex);

}

return rs;

}

public ResultSet loadStudentGroup(String id, ResultSet rs){

try {

String sql = "SELECT\n" +

"group\_student.id,\n" +

"group\_student.id\_group,\n" +

"students.first\_name,\n" +

"students.last\_name\n" +

"FROM\n" +

"group\_student ,\n" +

"students\n" +

"WHERE\n" +

"group\_student.id\_student = students.id\_student and group\_student.id\_group = '"+id+"'";

pst = conn.prepareStatement(sql);

rs=pst.executeQuery(sql);

} catch (SQLException ex) {

JOptionPane.showMessageDialog(null,ex);

}

return rs;

}

public ResultSet loadStudentGroupClear(ResultSet rs){

try {

String sql = "SELECT\n" +

"group\_student.id,\n" +

"group\_student.id\_group,\n" +

"students.first\_name,\n" +

"students.last\_name\n" +

"FROM\n" +

"group\_student ,\n" +

"students\n" +

"WHERE\n" +

"group\_student.id\_student = students.id\_student";

pst = conn.prepareStatement(sql);

rs=pst.executeQuery(sql);

} catch (SQLException ex) {

JOptionPane.showMessageDialog(null,ex);

}

return rs;

}

public void studentTableInsert(String id\_group, String id\_student){

try {

String sql="INSERT INTO "+tag.DB\_NAME+"."+tag.STUDENTGROUP\_TABLE+"("+tag.IDGROUP+","+tag.IDSTUDENT+") "

+ "VALUES('"+id\_group+"', '"+id\_student+"')";

pst =conn.prepareStatement(sql);

pst.executeUpdate();

} catch (SQLException ex) {

JOptionPane.showMessageDialog(null,ex);

}

finally {

try {

rs.close();

pst.close();

}

catch (Exception e){

}

}

}

}

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package Controller;

import Model.Tags;

import MySQL.javaconnect;

import java.sql.Connection;

import java.sql.Date;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.text.DateFormat;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.Calendar;

import java.util.HashMap;

import java.util.Locale;

import java.util.logging.Level;

import java.util.logging.Logger;

import javax.swing.JOptionPane;

/\*\*

\*

\* @author AlexKarlov

\*/

public class DBController {

Connection conn = null;

ResultSet rs=null;

PreparedStatement pst =null;

Tags tag;

public DBController() {

conn=javaconnect.ConnecrDb();

tag = new Tags();

}

public void studentTableInsert(String name, String lastName, Calendar dateBirthday){

Calendar date = dateBirthday;

java.sql.Date bDate = null;

bDate = new java.sql.Date(date.getTimeInMillis());

if (!(name.equals("Имя") )&&(lastName.equals("Фамилия"))){

try {

String sql="INSERT INTO "+tag.DB\_NAME+"."+tag.STUDENT\_TABLE+"("+tag.NAME+","+tag.LASTNAME+","+tag.STUDENT\_DATEBIRTHDAY+" ) "

+ "VALUES('"+name+"', '"+lastName+"', '"+bDate+"')";

pst =conn.prepareStatement(sql);

pst.executeUpdate();

} catch (SQLException ex) {

JOptionPane.showMessageDialog(null,ex);

}

finally {

try {

rs.close();

pst.close();

}

catch (Exception e){

}

}

}

else {

JOptionPane.showMessageDialog(null,"Пожжалуйста, ввидете действительніе данные");

}

}

public ResultSet loadTable(String tableName, ResultSet rs){

try {

String sql = "select \* from "+tableName+" ";

pst = conn.prepareStatement(sql);

rs=pst.executeQuery(sql);

} catch (SQLException ex) {

JOptionPane.showMessageDialog(null,ex);

}

return rs;

}

public void updateTableStudent(int id, String name, String lastName, Calendar birthdayDay){

Calendar date = birthdayDay;

java.sql.Date bDate = null;

bDate = new java.sql.Date(date.getTimeInMillis());

try {

String sql = "UPDATE "+tag.DB\_NAME+"."+tag.STUDENT\_TABLE+" "

+ "set "+tag.NAME+" = '"+name+"', "+tag.LASTNAME+" = '"+lastName+"',"

+ " "+tag.STUDENT\_DATEBIRTHDAY+" = '"+bDate+"' where id\_student = '"+id+"'";

pst = conn.prepareStatement(sql);

pst.execute();

} catch (SQLException ex) {

JOptionPane.showMessageDialog(null,ex);

}

}

public HashMap getValueFromTableStudent(Integer tableClickId){

HashMap<String, Object> listValues = new HashMap<String, Object>();

try {

String sql = "SELECT \* from "+tag.DB\_NAME+"."+tag.STUDENT\_TABLE+" where id\_student='"+tableClickId+"'";

pst = conn.prepareStatement(sql);

rs = pst.executeQuery();

if(rs.next()){

listValues.put("first\_name", rs.getString("first\_name"));

listValues.put("last\_name", rs.getString("last\_name"));

listValues.put("date\_birthday", rs.getDate("date\_birthday"));

listValues.put("id", rs.getInt("id\_student"));

}

} catch (SQLException ex) {

JOptionPane.showMessageDialog(null,ex);

}

finally {

try {

rs.close();

pst.close();

}

catch (Exception e){

}

}

return listValues;

}

public ResultSet searchStudentInTable(String lastName, ResultSet rs){

String sql = "SELECT \* FROM "+tag.DB\_NAME+"."+tag.STUDENT\_TABLE+" where "+tag.LASTNAME+" = '"+lastName+"'";

try {

pst = conn.prepareStatement(sql);

rs=pst.executeQuery(sql);

} catch (SQLException ex) {

JOptionPane.showMessageDialog(null,ex);

}

return rs;

}

public ResultSet loadSummaryTable(String sql, ResultSet rs){

try {

pst = conn.prepareStatement(sql);

rs = pst.executeQuery(sql);

} catch (SQLException ex) {

JOptionPane.showMessageDialog(null,ex);

}

return rs;

}

}