База даних успішності ЗВО

Учасники курсової роботи

Сидько Нікіта

Паламарчук Максим

Саєнко Андрій

Огляд проекту

Досліджуваний предмет

Предметом дослідження – система централізованого зберігання успішності студентів, даних студентів та даних курсів навчального процесу; з можливістю отримання звітів, друку звітів у файл; з можливістю контролю діяльності викладачів. Також в програмі має бути присутня можливість форматувати звіти та таблиці: задавати різне сортування, виводити різні комбінації стовпців.

Досліджувана система може бути частиною робочого процесу ЗВО і бути використана як система зберігання успішності та як система електронного кампусу факультетів.

Які проблеми вирішує?

У закладах сучасної вищої освіти існує потреба, пов'язана із зберіганням інформації про студентів.

- Студенти повинні мати доступ до перегляду своїх успіхів з окремих дисциплін.
- Викладачі повинні мати змогу просто та зручно виставляти, редагувати оцінки
- Адміністрація в свою чергу повинна мати швидкий та безпечний доступ до інформації як студентів, так і викладачів.

Виникає проблема доступності, швидкості та безпечності віддаленого контролю за студентами і їх успішністю .

Відповідно для успішної співпраці студентів та робітників ЗВО потрібна система централізованого зберігання успішності студентів.

Функціональність програми

Зміна оцінок студента

Призначення стипендії

Переглянути успішність

Переглянути користувача

Призначення групи

Створити користувача

Переглядання курсу

Редагування курсу

Аналоги

Система підтримки навчального процесу Національниго Технічного Університету України "Київський політехнічний інститут імені Ігоря Сікорського".

Навчальна платформа, призначена для об'єднання педагогів, адміністраторів і учнів в одну надійну, безпечну та інтегровану систему для створення персоналізованого навчального середовища.

Чому саме ми?

Ми надаємо декілька варіантів використання нашого софту які є тільки в нас:

- Варіант використання 1
  - Додавання нового студента, вказавши основні особові дані, курс та інші необхідні дані.
  - Перевірка участі студента в громадській роботі та вніс відповідний статус.
  - Новий студент доданий до бази даних.
  - Звіт про стипендії.
- Варіант використання 2
  - Надано опцію "Введення оцінок".
  - Перевірка і внесення необхідної корекції в інформацію про студента, якщо це потрібно.
  - Надано опцію "Перегляд інформації про студента"
- Варіант використання 3
  - Опція "Перегляд оцінок".
  - Перевірка оцінок за поточний семестр.

Чому ми краще?

У порівнянні з аналогами, ми пропонуємо наступні можливості

1.Персоналізований доступ до інформації: розширення особистого кабінету студента, щоб включити додаткові функції та аналітику стосовно його академічної діяльності, наприклад, детальні статистики по конкретних дисциплінах або порівняння з групою.

2.Автоматизована система нагород та стипендій: впровадження системи автоматичного нагородження студентів, які досягають високих успіхів. Студентам з високими оцінками "5" може автоматично надаватися підвищена стипендія.

3.Розширення функцій для викладачів: розробка інструментів для викладачів, які дозволяють їм детально аналізувати та моніторити успішність своїх студентів, а також швидко взаємодіяти з ними для надання додаткової підтримки.

4.Розширення модулів аналітики: додавання нових модулів аналітики для статистичного аналізу успішності груп, порівняльного аналізу між факультетами, ідентифікації трендів та можливостей для оптимізації навчального процесу.

5.Зручний інтерфейс користувача: розробка зручного та інтуїтивно зрозумілого інтерфейсу користувача для полегшення навігації та отримання необхідної інформації.

Набір технологій що були застосовані

Java – мова програмування, яку ми обрали для реалізації роботи

Java Fx – платформа, яка була використана для реалізації графічного інтерфейсу

Docker — програмне забезпечення для автоматизації розгортання та керування програмами в середовищах з підтримкою контейнеризації, контейнеризатор додатків

Maven – фреймворк, який було застосовано для реалізації тестування функцій програми

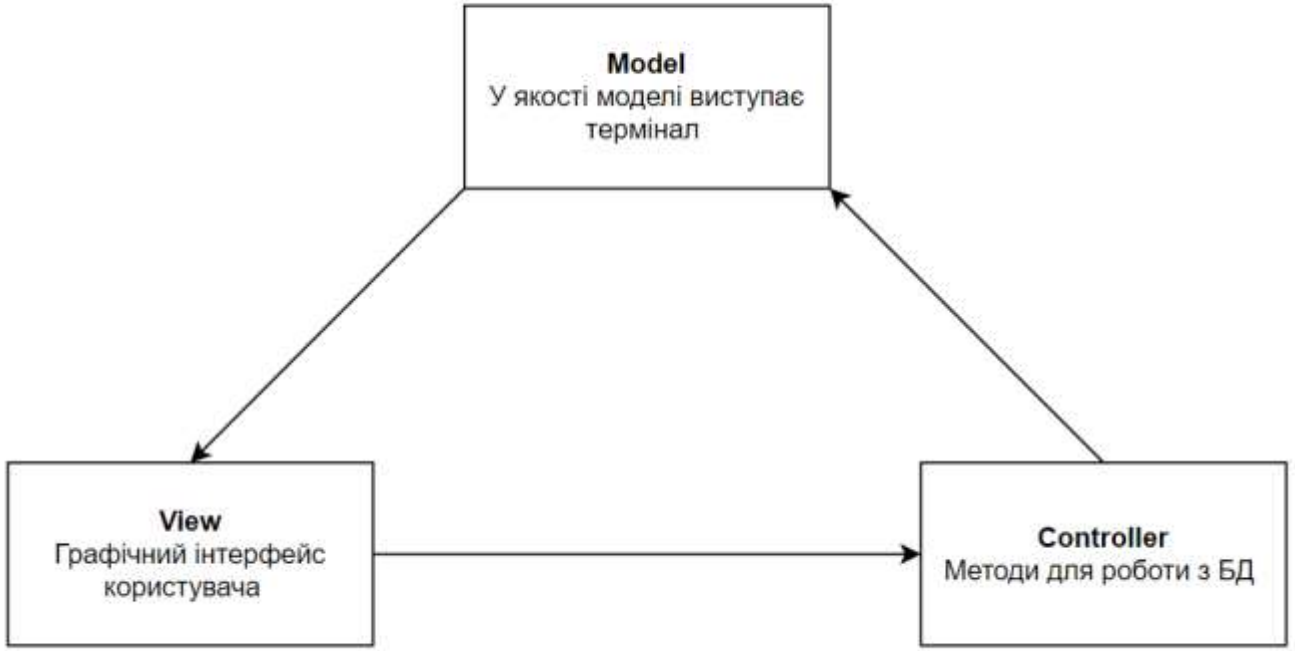
GitHub – система зберігання контролю версій

MongoDB - система документних баз даних

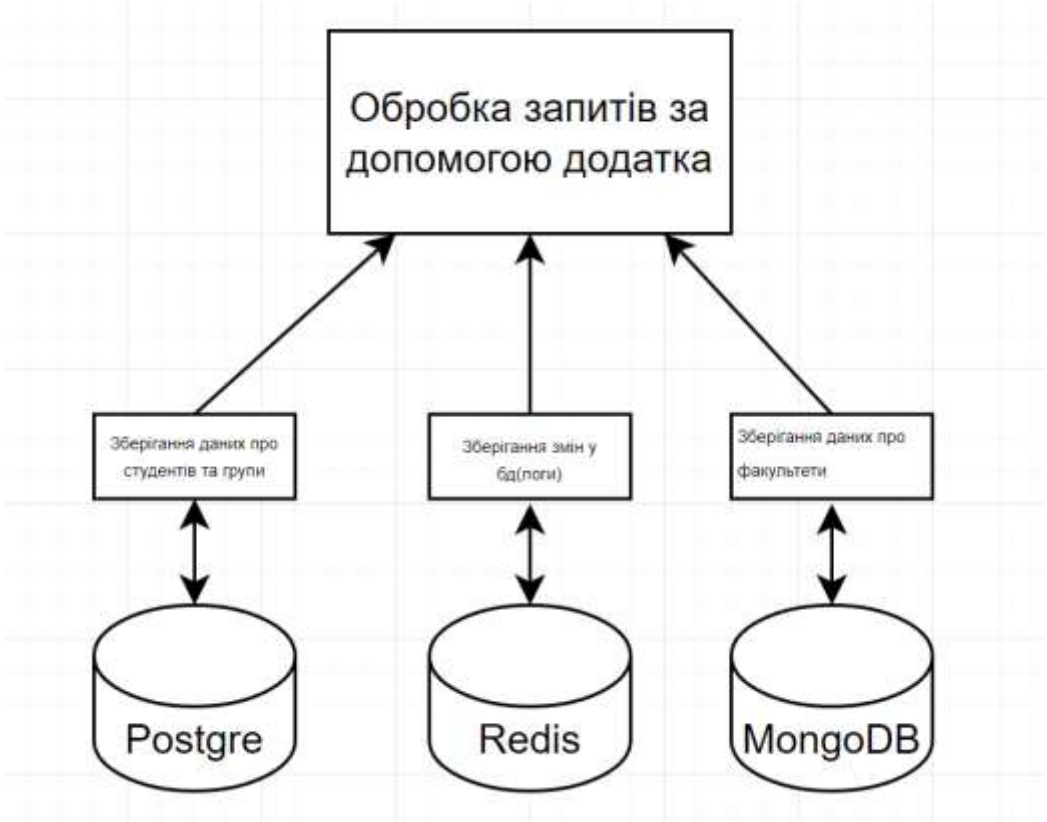
PostgreSQL – система реляційних баз даних

Redis – система баз даних «ключ / значення»

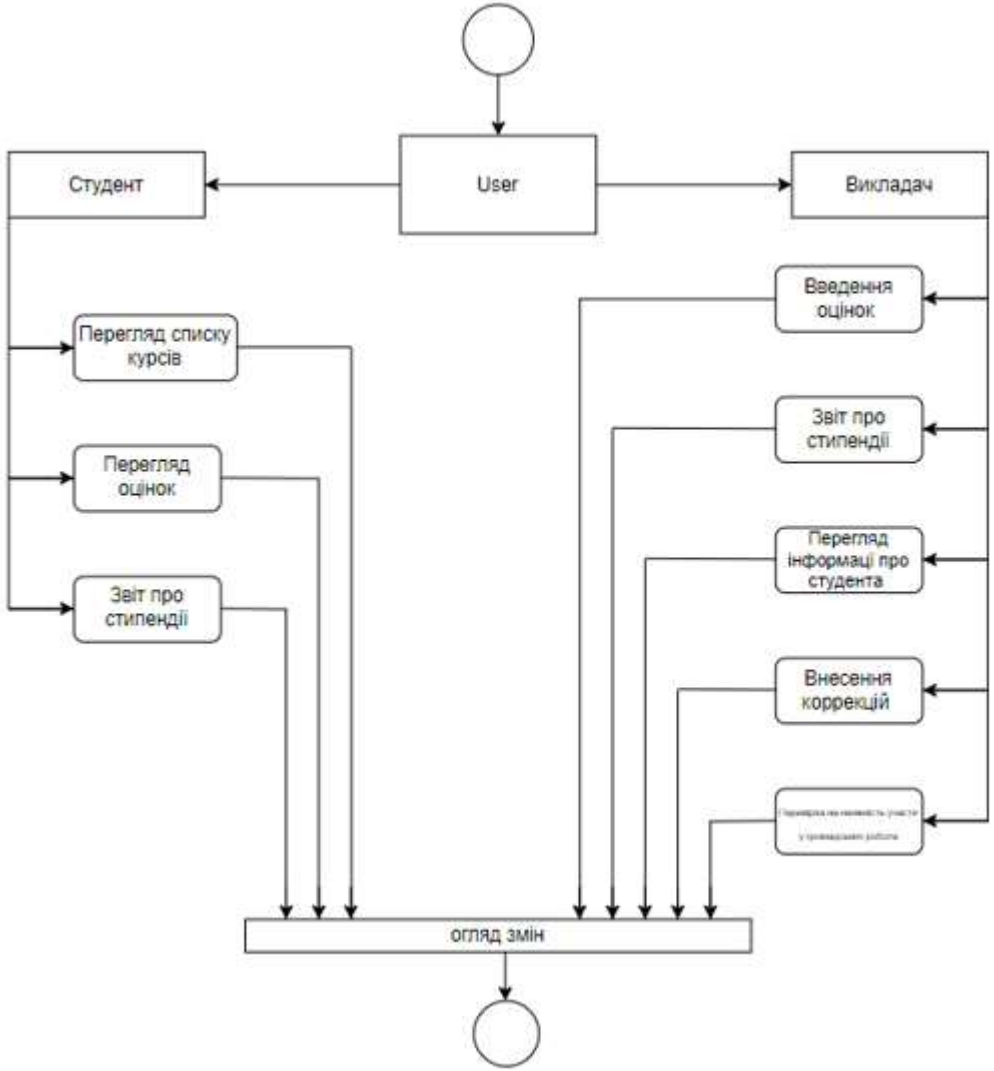
Метод організації коду



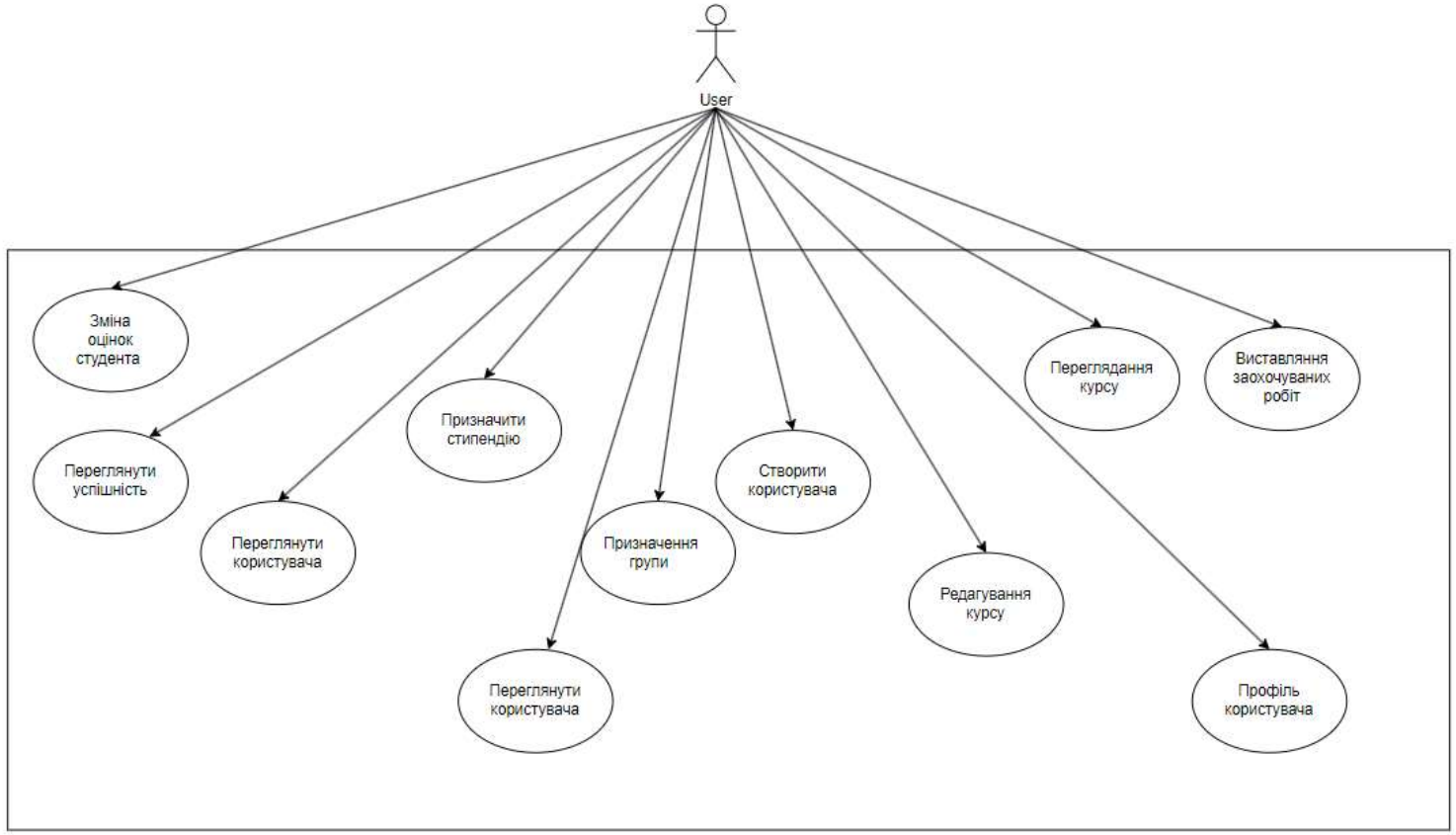
Архітектурно-орієнтовна модель



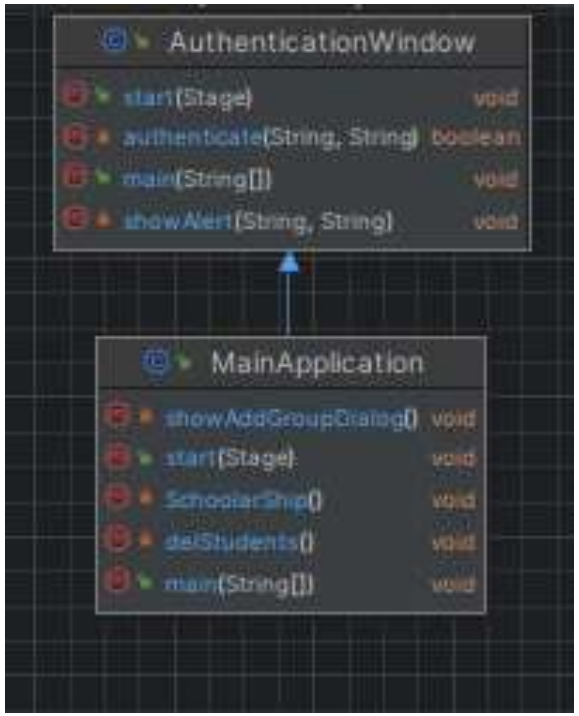
UML-діаграма діяльності

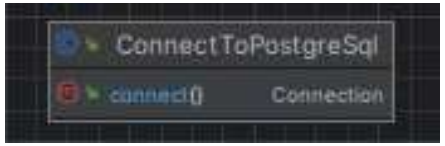


Use-cases diagram

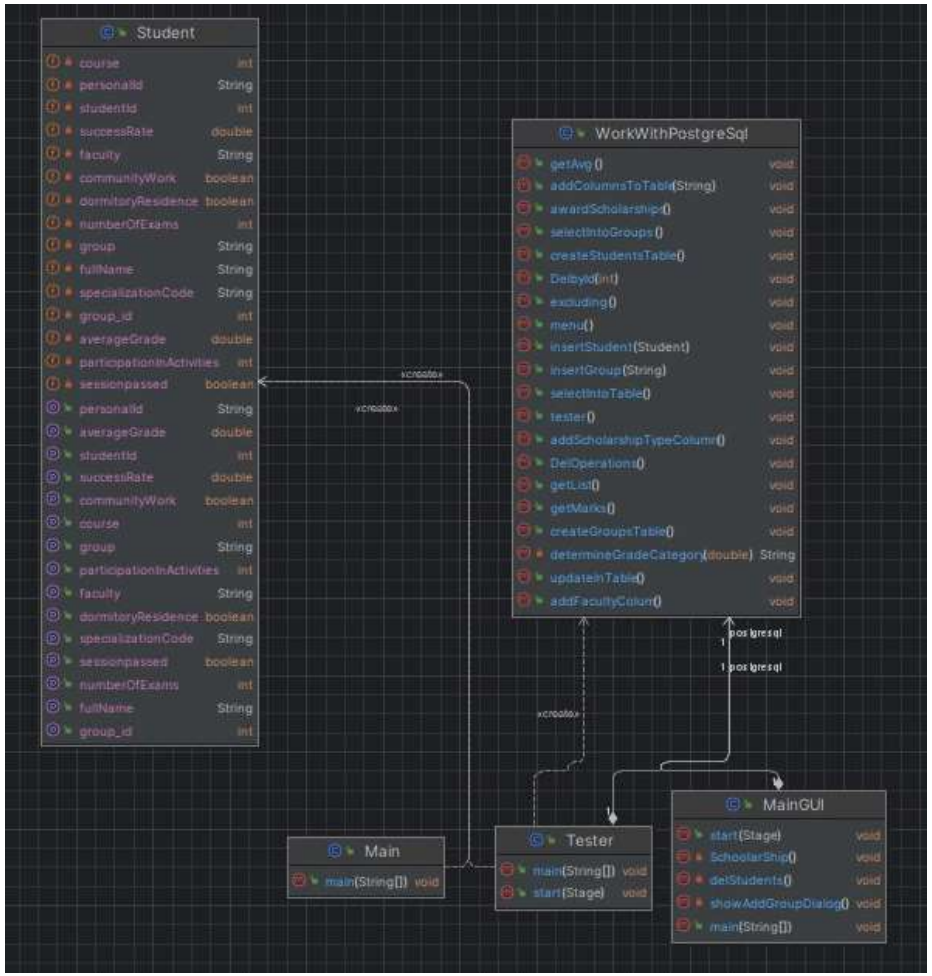


Діаграми класів





Діаграми класів



Тестування

```java
@Test
public void testGetAvg() {

    WorkWithPostgreSql.insertGroup("Group1");
    WorkWithPostgreSql.insertStudent(new Student("John Doe", 1, "CS", 4.0, true, 1, "Computer Science"));
    WorkWithPostgreSql.insertStudent(new Student("Jane Doe", 1, "CS", 5.0, false, 1, "Computer Science"));
```

```java
@Test
public void testInsertStudent() {

    WorkWithPostgreSql.insertGroup("Group1");


    Student student = new Student("John Doe", 1, "CS", 4.0, true, 1, "Computer Science");
    WorkWithPostgreSql.insertStudent(student);


    assertTrue(checkIfStudentExists("John Doe"));
}
```

```java
@Test
public void testInsertGroup() {

    WorkWithPostgreSql.insertGroup("Group1");


    assertTrue(checkIfGroupExists("Group1"));
}
```

Результати тестів

✓ Testers	999 ms
✓ testExcluding	521 ms
✓ testGetMarks	45 ms
✓ testAdd	121 ms
✓ testSchoolarship	111 ms
✓ testDelete	53 ms

✓ testGetAvg	36 ms
✓ testSelect	33 ms
✓ testUpdate	32 ms
✓ testdetermineGradeCategory	47 ms

Бази даних

PostgreSQL

- PostgreSQL - було використовано для того щоб зберігати/зчитувати дані про студента.

redis

- Redis - використовувалося для запису логів.

mongoDB

- MongoDB - була використана для додавання студентів до факультетів, та створення груп.

Запити до PostgreSQL

```java
public static void createGroupsTable() {
    try (Connection connection = ConnectToPostgreSql.connect();
         Statement statement = connection.createStatement()) {

        String sql = "CREATE TABLE IF NOT EXISTS GroupsDA (" +
                "GroupID SERIAL PRIMARY KEY," +
                "GroupName VARCHAR(255) NOT NULL);";

        statement.executeUpdate(sql);
        System.out.println("Groups table created successfully.");

    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

```java
public static void insertStudent(Student student) {
    try (Connection connection = ConnectToPostgreSql.connect();
         PreparedStatement preparedStatement = connection.prepareStatement(
                 sql: "INSERT INTO StudentsFac (FullName, COURSE, specializationCode, AVE
                         "VALUES (?, ?, ?, ?, ?, ?,?)")) {

        // Перевірка на null перед встановленням значення для FullName
        if (student.getFullName() != null) {
            preparedStatement.setString( parameterIndex: 1, student.getFullName());
        } else {
            throw new IllegalArgumentException("Помилка: Пусте ім'я студента.");
        }
        preparedStatement.setDouble( parameterIndex: 2, student.getCourse());
        preparedStatement.setString( parameterIndex: 3, student.getSpecializationCode());
        preparedStatement.setDouble( parameterIndex: 4, student.getAverageGrade());
        preparedStatement.setBoolean( parameterIndex: 5, student.isCommunityWork());
        preparedStatement.setInt( parameterIndex: 6, student.getGroup_id());
        preparedStatement.setString( parameterIndex: 7,student.getFaculty());
```

```java
public static void insertGroup(String groupName) {
    try (Connection connection = ConnectToPostgreSql.connect();
         PreparedStatement preparedStatement = connection.prepareStatement( sql: "INSERT INTO

        preparedStatement.setString( parameterIndex: 1, groupName);
        preparedStatement.executeUpdate();
        System.out.println("Group added successfully.");

    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

PostgreSQL в Docker

```
students=# SELECT * FROM StudentsFac;
 studentid | fullname | course |              specializationcode               | average_mark | communitywork | groupid | faculty
-----------+----------+--------+-----------------------------------------------+--------------+---------------+---------+---------
         3 | Андрей   |      3 | DA                                            |            3 | f             |       1 | mf
         6 | Макс     |      3 | Da-11                                         |            3 | t             |       1 | fm
         7 | Андрей   |      3 | DA-11                                         |            3 | t             |       1 | mf
        10 | Student  |      3 | DA-11                                         |          3.5 | t             |       1 | mf
        11 | Student  |      3 | DA-11                                         |          3.5 | t             |       1 | mf
         9 | Max      |      2 | DA-12                                         |            5 | t             |       2 | mf
(6 rows)
```

```
students=# SELECT * FROM StudentsFac;
 studentid | fullname | course |              specializationcode               | average_mark | communitywork | groupid | faculty |  scholarshiptype
-----------+----------+--------+-----------------------------------------------+--------------+---------------+---------+---------+-------------------
         6 | Макс     |      3 | Da-11                                         |            3 | t             |       1 | fm      | Без стипендії
         7 | Андрей   |      3 | DA-11                                         |            3 | t             |       1 | mf      | Без стипендії
        10 | Student  |      3 | DA-11                                         |          3.5 | t             |       1 | mf      | Без стипендії
        11 | Student  |      3 | DA-11                                         |          3.5 | t             |       1 | mf      | Без стипендії
         9 | Max      |      2 | DA-12                                         |            5 | t             |       2 | mf      | Звичайна стипендія
(5 rows)
```

Запити до MongoDB

```java
private static void addGroupToFaculty(MongoCollection<Document> facultyCollection, Scanner scanner) {
    System.out.print("Enter the faculty ID to add a group: ");
    String facultyId = scanner.next();
    Document existingFaculty = facultyCollection.find(Filters.eq( fieldName: "_id", facultyId)).first();
    if (existingFaculty != null) {
        addGroupToExistingFaculty(existingFaculty, facultyCollection, scanner);
    } else {
        System.out.println("Faculty with ID " + facultyId + " does not exist. Cannot add a group.");
    }
}
```

```java
private static void deleteStudentFromGroup(MongoCollection<Document> groupCollection, Scanner scanner) {
    System.out.print("Enter the _id of the group to delete a student: ");
    String groupId = scanner.next();
    Document existingGroup = groupCollection.find(Filters.eq( fieldName: "_id", groupId)).first();
    if (existingGroup != null) {
        groupCollection.updateOne(Filters.eq( fieldName: "_id", groupId), new Document("$inc", new Document("People", -1)));
        System.out.println("Decreased the number of People for group with _id " + groupId + ".");
    }
    else {
        System.out.println("Group with _id " + groupId + " does not exist. Cannot delete a student.");
    }
}
```

MongoDB в Docker

filter {}

	_id	People
1	DA	2
2	KA	1
3	FG	1
4	QR	1

	_id	Name	group
1	mf	Math	["KA", "QQ"]
2	hf	History	["PA", "MA"]
3	bf	Biology	["FA", "AS", "PA"]

```
> db.facultyCollection.find()
{ "_id" : "mf", "group" : [ "KA", "QQ" ], "Name" : "Math" }
{ "_id" : "hf", "group" : [ "PA", "MA" ], "Name" : "History" }
{ "_id" : "bf", "group" : [ "FA", "AS", "PA" ], "Name" : "Biology" }
```

Запити до Redis

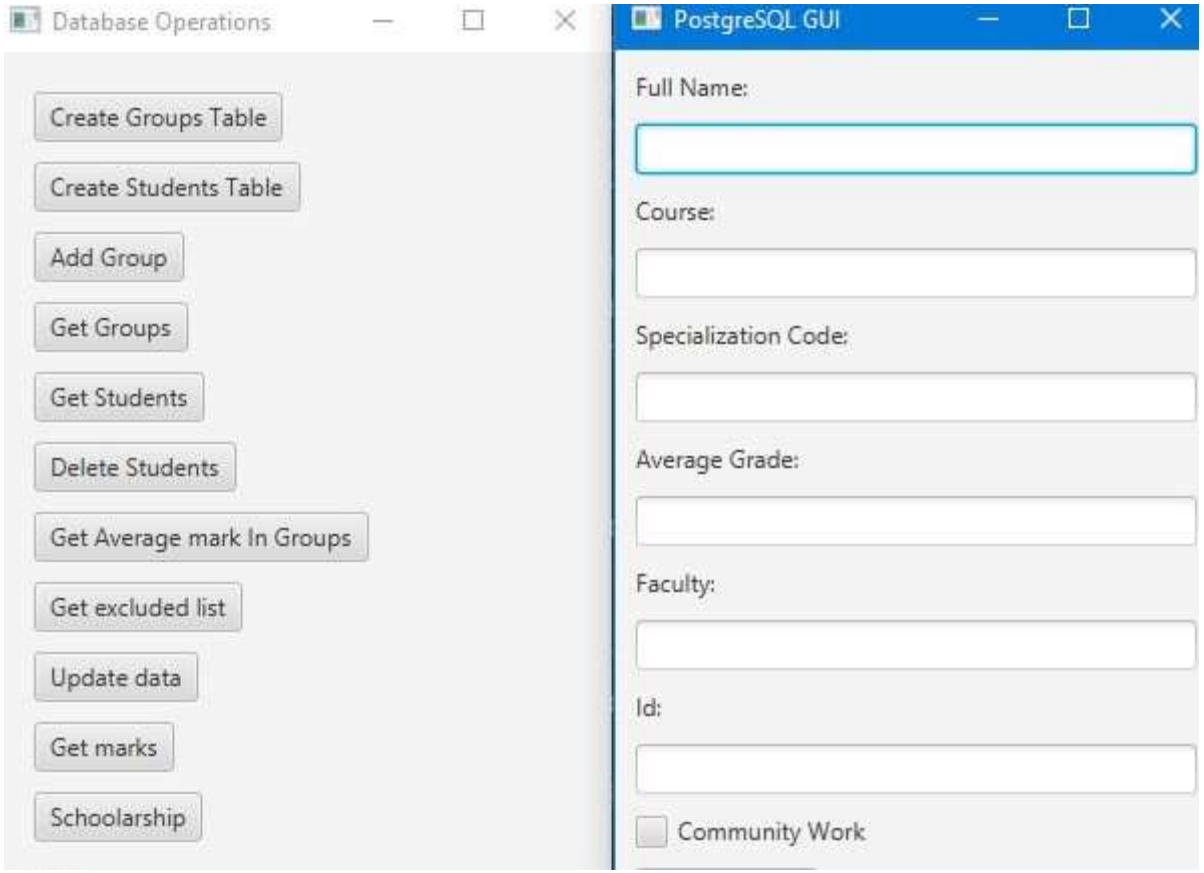
```
ublic static void addScholarshipTypeColumn() {
    try (Connection connection = ConnectToPostgreSql.connect();
         Statement statement = connection.createStatement()) {

        String sql = "ALTER TABLE StudentsFac ADD COLUMN ScholarshipType VARCHAR(50);";

        statement.executeUpdate(sql);
        currentTime = LocalTime.now();
        jedis.rpush(String.valueOf(currentDate.getDayOfYear()), String.valueOf( obj: "ScholarshipType column " + currentTime));
        System.out.println("ScholarshipType column added successfully.");

    } catch (SQLException e) {
        e.printStackTrace();
    }
```

```
public static void addFacultyColum() {
    try (Connection connection = ConnectToPostgreSql.connect();
         Statement statement = connection.createStatement()) {

        String sql = "ALTER TABLE StudentsFac ADD COLUMN Faculty VARCHAR(50);";

        statement.executeUpdate(sql);
        currentTime = LocalTime.now();
        jedis.rpush(String.valueOf(currentDate.getDayOfYear()), String.valueOf( obj: "Faculty column added " + currentTime))
        System.out.println("Faculty column added successfully.");

    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

Redis в Docker

```
# redis-cli
127.0.0.1:6379> LRANGE 358 0 -1
 1) "addedGroup buba3"
 2) "addedGroup BIBA3"
 3) "addedGroup bybe3"
 4) "addedGroup baba3"
 5) "addedGroup jajaja3"
 6) "Addedgroup check 00:00:49.412173900"
 7) "Added student andrii 00:01:01.248359900"
 8) "deleted student1 00:00:49.412173900"
 9) "Scholarships awarded 00:00:49.412173900"
10) "Addedgroup BIBI 00:00:49.412173900"
11) "Added student andrii 00:01:01.248359900"
12) "Added student ANDRII 00:06:01.997322900"
13) "Addedgroup check 00:06:14.210422700"
127.0.0.1:6379>
```

Інтерфейс


Database Operations
Create Groups Table
Create Students Table
Add Group
Get Groups
Get Students
Delete Students
Get Average mark In Groups
Get excluded list
Update data
Get marks
Schoolarship
PostgreSQL GUI
Full Name:
Course:
Specialization Code:
Average Grade:
Faculty:
Id:
Community Work

Дякуємо за увагу