вМИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

(МОСКОВСКИЙ ПОЛИТЕХ)

Факультет информационных технологий

Кафедра «Инфокогнитивные технологии»

КУРСОВОЙ ПРОЕКТ

на тему: *«Модели ограничения доступа к записям базы данных»*

Направление подготовки 09.03.03 «Прикладная информатика»

Профиль «Корпоративные информационные системы»

Выполнил:

студент группы 211-363

Шарков Иван Александрович

11.07.2022

(подпись)

Введение

Утечка информации с базы данных способна привести к непоправимым и ужасным последствиям, которые могут в корне изменить ситуацию в жизни человека, организации или целой страны. В настоящее время большинство СУБД не используют в полной мере все подвиды моделей ограничения доступа, из-за чего они легко уязвимы перед внешними кибератаками. Для устранения данной проблемы были разработаны специальные модели ограничения доступа баз данных для защиты от нежелательных проникновений и утечек важной информации. В данном проекте представлен вид: «Кластеризационная модель ограничения доступа», основанный на не пересекаемых подмножествах отношений базы данных.

1. Постановка задачи

Цель работы – разработать простое оконное клиент-серверное приложение, позволяющее пользователю посредством графического интерфейса и согласно предоставляемыми ему правами доступа ролевой модели обрабатывать данные, хранящиеся на сервере.

Основные этапы работы:

1. Настроить и подключить OpenVPN для успешной установки соединения с базой данных.
2. Создать и подключить удаленный репозиторий на GitHub.
3. Подключить все необходимые библиотеки к проекту.
4. Создать интерфейсные окна.
5. Создать классы контроллеры для окон.
6. Скомпилировать исполняемый jar файл.

2. Проектирование и разработка

2.1. Модель базы данных

Отношения базы данных, хранящиеся на сервере, представлены в следующем виде:

1. users = {id, login, password, study_group, rank};
2. teacher_quotes = {id, user_id, quote, last_name, first_name, second_name, lesson, publication_date}.

Все отношения соответствуют первой нормальной форме, поскольку все атрибуты являются атомарными, а также нормальной форме Бойса-Кода (которая покрывает вторую и третью нормальные формы), так как все детерминанты являются потенциальными ключами.

Физическая модель данных представлена в приложении (2).

Реляционная модель данных представлена на Рисунок 1.

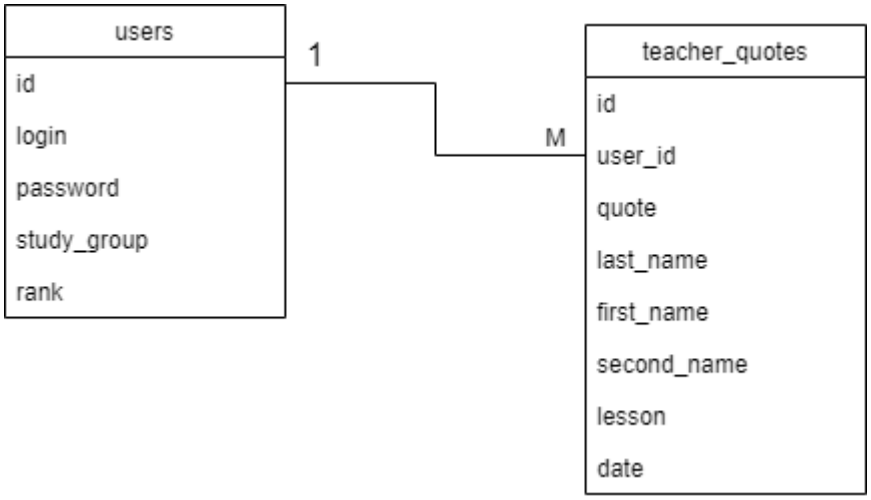

Рисунок 1 — Реляционная модель базы данных

2.2. Архитектура проекта

Приложение написано на языке программирования Java с использованием библиотеки JavaFX для создания интерфейса и mysql-connector для установки соединения с базой данных в IDE IntelliJ IDEA. Код разработан в концепции MVC. Разработка сопровождалась ведением удаленного репозитория посредством системы контроля версий Git.

Содержание проекта:

1. Пакет resources – содержит файлы формата fxml, отвечающие за интерфейс различных окон.
2. Пакет Java – содержит java классы приложения.
3. Класс Const – содержит константы для более удобного и быстрого редактирования кода.
4. Класс Configs – содержит константы для подключения к базе данных.
5. Класс DatabaseHandler – содержит методы для взаимодействия с базой данных.
6. Класс LoginController – контроллер для окна авторизации.
7. Класс RegistrationController – контроллер для окна регистрации.
8. Класс HomeController – контроллер для окна главной страницы.
9. Класс AddQuote – контроллер для окна добавления цитаты.
10. Класс EditQuote – контроллер для окна редактирования цитаты.
11. Класс User – хранит данные в виде объекта пользователя.
12. Класс Quote – хранит данные в виде объекта цитаты.
13. Main – точка входа в программу.

Диаграмма классов расположена в приложении (1).

2.3. Клиентская часть

Интерфейс приложения разработан при помощи сторонней программы Scene Builder, которая работает в концепции Drag-And-Drop, и сама генерирует файлы формата fxml. Файлы представлены на Рисунок 2.

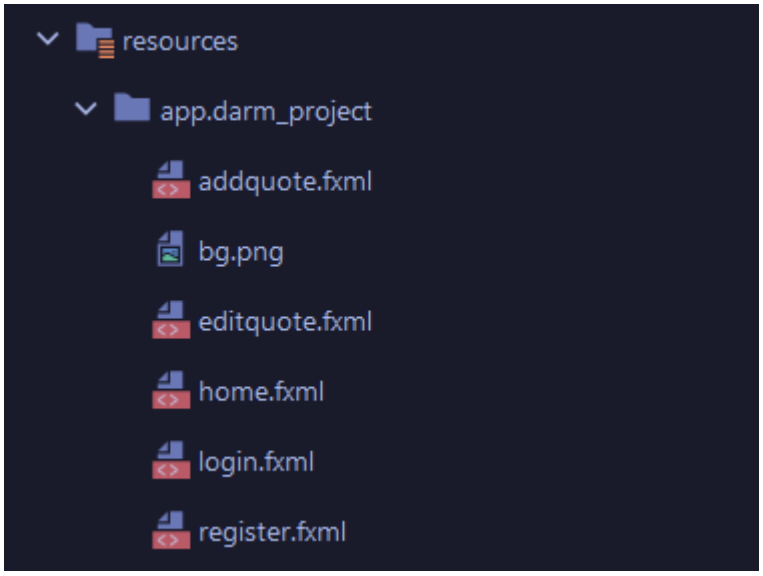

Рисунок 2 — Файлы формата fxml

Для управления объектами из fxml файлов требуется создать контроллер, содержание которого также генерирует само приложения Scene Builder. Файлы контроллеры представлены на Рисунок 3.

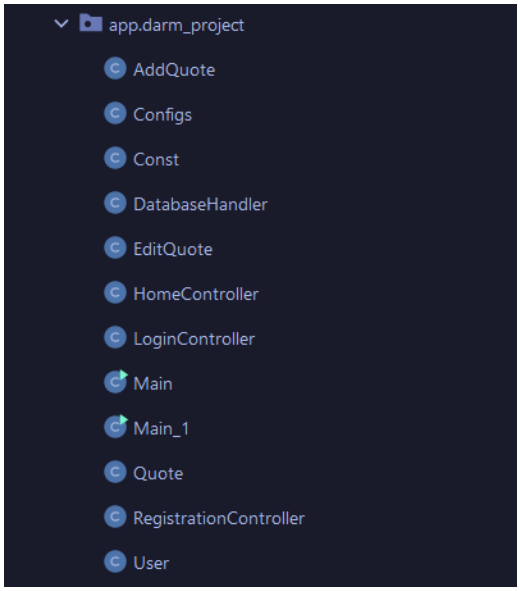

Рисунок 3 — Файлы контроллеры

Чтобы связать контроллер и fxml файл, во втором нужно указать в элементе AnchorPane атрибут controller с указанием ссылки на существующий класс контроллера.

Для того, чтобы иметь возможность передачи актуальных данных авторизованного пользователя в другие классы контроллеры, используется публичный статический объект класса User, в который, при помощи сеттеров, записываются данные необходимого пользователя из базы данных.

Для установки соединения и работы с базой данных реализованы следующие методы:

Метод **getDbConnection()** класса **DatabaseHandler** позволяет установить соединение с базой данных, используя ее данные. Метод возвращает объект интерфейса с результатом подключения.

```
// -- Установить соединение с sql database
public Connection getDbConnection() throws ClassNotFoundException, SQLException {
    String connectionString = "jdbc:mysql://" + host + ":" + port + "/" + dbName;

    Class.forName("com.mysql.cj.jdbc.Driver");
    dbConnection = DriverManager.getConnection(connectionString, user, password);

    return dbConnection;
}
```

Рисунок 4 — Метод getDbConnection()

Метод **signUpUser()** класса **DatabaseHandler** производит запись пользователя в базу данных. Метод ничего не возвращает, лишь отправляет указанный SQL-запрос, который заполняется при помощи данных из текстовых полей интерфейса.

```java
// -- Зарегистрировать пользователя в sql database
public void signUpUser(User user) throws SQLException, ClassNotFoundException {
    String insert = "INSERT INTO " + Const.USER_TABLE + " (" + Const.USERS_LOGIN + "," + Const.USERS_PASSWORD + "," +
            Const.USERS_STUDY_GROUP + "," + Const.USERS_RANK + ")" +  "VALUES(?,?,?,?)";

    PreparedStatement prepState = getDbConnection().prepareStatement(insert);
    prepState.setString( parameterIndex: 1, user.getLogin());
    prepState.setString( parameterIndex: 2, user.getPassword());
    prepState.setString( parameterIndex: 3, user.getStudyGroup());
    prepState.setString( parameterIndex: 4,  x: "user");

    prepState.executeUpdate();
}
```

Рисунок 5 — Метод signUpUser()

Метод **getUser()** класса **DatabaseHandler** по указанному логину и паролю находит пользователя в базе данных для его авторизации, а также создает заполняет публичный статический объект класса User user всеми необходимыми данными, чтобы передать их в другие классы контроллеры.

```java
// -- Получить логин и пароль пользователя с sql database
public ResultSet getUser(User user) throws SQLException, ClassNotFoundException {
    ResultSet resSet = null;

    String select = "SELECT * FROM " + Const.USER_TABLE +
            " WHERE " + Const.USERS_LOGIN + "=? AND " + Const.USERS_PASSWORD + "=?";

    PreparedStatement prepState = getDbConnection().prepareStatement(select);
    prepState.setString( parameterIndex: 1, user.getLogin());
    prepState.setString( parameterIndex: 2, user.getPassword());

    resSet = prepState.executeQuery();

    return resSet;
}
```

Рисунок 6 — Метод getUser()

Метод **loginUser()** класса **LoginController** заполняет публичный статический объект класса User user для дальнейшей работы с ним в других классах контроллерах.

```
// -- Авторизация пользователя
private void loginUser(String loginText, String loginPassword) throws SQLException, ClassNotFoundException, IOException {
    DatabaseHandler dbHandler = new DatabaseHandler();
    User user = new User();
    user.setLogin(loginText);
    user.setPassword(loginPassword);
    ResultSet result = dbHandler.getUser(user);
    if (result.next()) {
        do {
            this.user = user;
            user.setId(result.getInt( columnIndex: 1));
            user.setStudyGroup(result.getString( columnIndex: 4));
            user.setRank(result.getString( columnIndex: 5));

            if (user.getRank().equals("admin")) {
                goToScene( fileName: "home",  sceneName: "Главная страница (уровень доступа: Администратор)");
            } else if (user.getRank().equals("user")) {
                goToScene( fileName: "home",  sceneName: "Главная страница (уровень доступа: Обычный пользователь)");
            } else if (user.getRank().equals("headman")) {
                goToScene( fileName: "home",  sceneName: "Главная страница (уровень доступа: Староста)");
            }
        } while(result.next());
    } else {
        setAlertMessage("Пользователь не найден!");
    }
}
```

Рисунок 7 — Метод loginUser()

Метод **singUpNewUser()** класса **RegistrationController** берет данные из полей интерфейса, подставляет их в SQL-запрос и отправляет на сервер, добавляя таким способом нового пользователя в базу данных.

```java
// -- Регистрация пользователя
private void signUpNewUser() {
    DatabaseHandler dbHandler = new DatabaseHandler();

    // -- Получение введенных данных с полей
    String login = loginFieldInLogin.getText();
    String password = passwordField.getText();
    String studyGroup = studyGroupField.getText();
    String repeatPassword = repeatPasswordField.getText();

    // -- Проверка на пустые строки
    if (!login.equals("") && !password.equals("") && !studyGroup.equals("") && !repeatPassword.equals("")) {
        if (password.equals(repeatPassword)) {
            User user = new User(login, password, studyGroup);
            setAlertMessage( text: "Вы успешно зарегистрировались!",  color: "#33ff00");
            try {
                dbHandler.signUpUser(user);
            } catch (SQLException e) {
                e.printStackTrace();
                setAlertMessage( text: "Данный логин уже используется!",  color: "red");
            } catch (ClassNotFoundException e) {
                e.printStackTrace();
                setAlertMessage( text: "Что-то пошло не так!",  color: "red");
            }
        } else {
            setAlertMessage( text: "Пароли не совпадают!",  color: "red");
        }
    } else {
       setAlertMessage( text: "Нельзя оставлять поля пустыми!",  color: "red");
    }
}
```

Рисунок 8 — Метод singUpNewUser()

Метод **loadDate()** класса **HomeController** устанавливает соединение с базой данных, получает из нее данные и загружает в таблицу приложения.

Рисунок 9 — Метод loadDate()

Метод **refreshTable()** класса **HomeController** в зависимости от уровня доступа пользователя отправляет SELECT запрос, а затем создает и заполняет объекты класса Quote данными из таблицы в соответствующие колонки, помещает их в массив, а затем выгружает массив в JavaFX таблицу приложения.

```
// -- Обновить данные в таблице
private void refreshTable() throws SQLException {
    quotesList.clear();

    if (LoginController.user.getRank().equals("guest")) query = "SELECT * FROM " + Const.TEACHER_QUOTES_TABLE;
    if (LoginController.user.getRank().equals("admin")) query = "SELECT * FROM " + Const.TEACHER_QUOTES_TABLE;
    if (LoginController.user.getRank().equals("user")) query =
            "SELECT teacher_quotes.id, teacher_quotes.user_id, teacher_quotes.quote, teacher_quotes.last_name, teacher_quotes.first_name, " +
            "teacher_quotes.second_name, teacher_quotes.lesson, teacher_quotes.publication_date " +
            "FROM users " +
            "JOIN teacher_quotes ON (users.id = teacher_quotes.user_id) " +
            "WHERE (users.study_group ='" + LoginController.user.getStudyGroup() + "')";

    if (LoginController.user.getRank().equals("headman")) query =
            "SELECT teacher_quotes.id, teacher_quotes.user_id, teacher_quotes.quote, teacher_quotes.last_name, teacher_quotes.first_name, " +
                    "teacher_quotes.second_name, teacher_quotes.lesson, teacher_quotes.publication_date " +
                    "FROM users " +
                    "JOIN teacher_quotes ON (users.id = teacher_quotes.user_id) " +
                    "WHERE (users.study_group ='" + LoginController.user.getStudyGroup() + "')";

    preparedStatement = connection.prepareStatement(query);
    resultSet = preparedStatement.executeQuery();

    while (resultSet.next()) {
        quotesList.add(new Quote(
                resultSet.getInt(Const.TEACHERS_ID),
                resultSet.getInt(Const.TEACHERS_USERID),
                resultSet.getString(Const.TEACHERS_QUOTE),
                resultSet.getString(Const.TEACHERS_LAST_NAME),
                resultSet.getString(Const.TEACHERS_FIRST_NAME),
                resultSet.getString(Const.TEACHERS_SECOND_NAME),
                resultSet.getString(Const.TEACHERS_LESSON),
                resultSet.getDate(Const.TEACHERS_DATE)
                )
        );

        teachersQuotesTable.setItems(quotesList);
    }
```

Рисунок 10 — Метод refreshTable()

Также для удобного и быстрого редактирования кода созданы следующие классы:

- Класс Const – хранит статические публичные переменные названий колонок таблиц.
- Класс Configs – хранит публичные статические переменные для подключения к базе данных.

3. Эксплуатация

При запуске приложения требуется авторизоваться, войти гостем с правами только для просмотра или зарегистрироваться. Окна авторизации и регистрации представлены на Рисунок 11 и Рисунок 12.

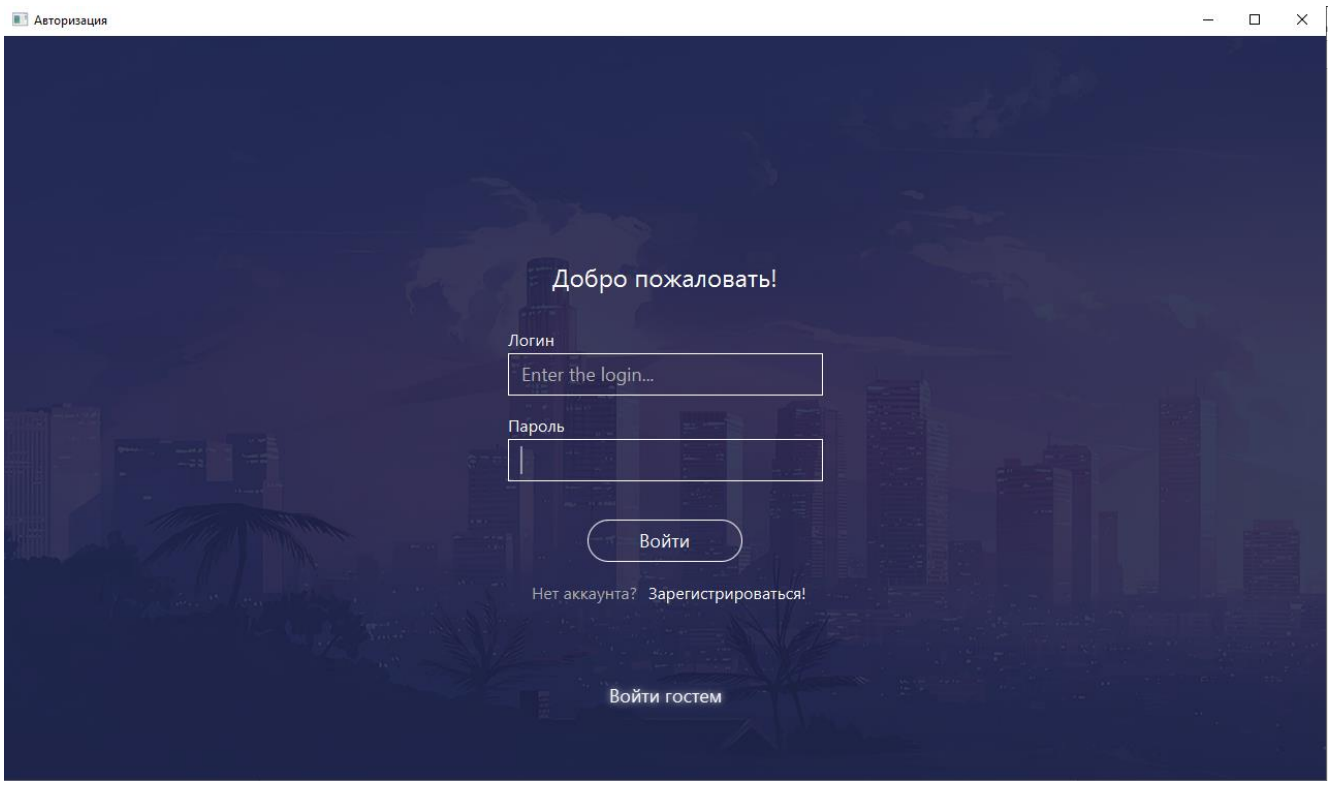

Рисунок 11 — Окно авторизации

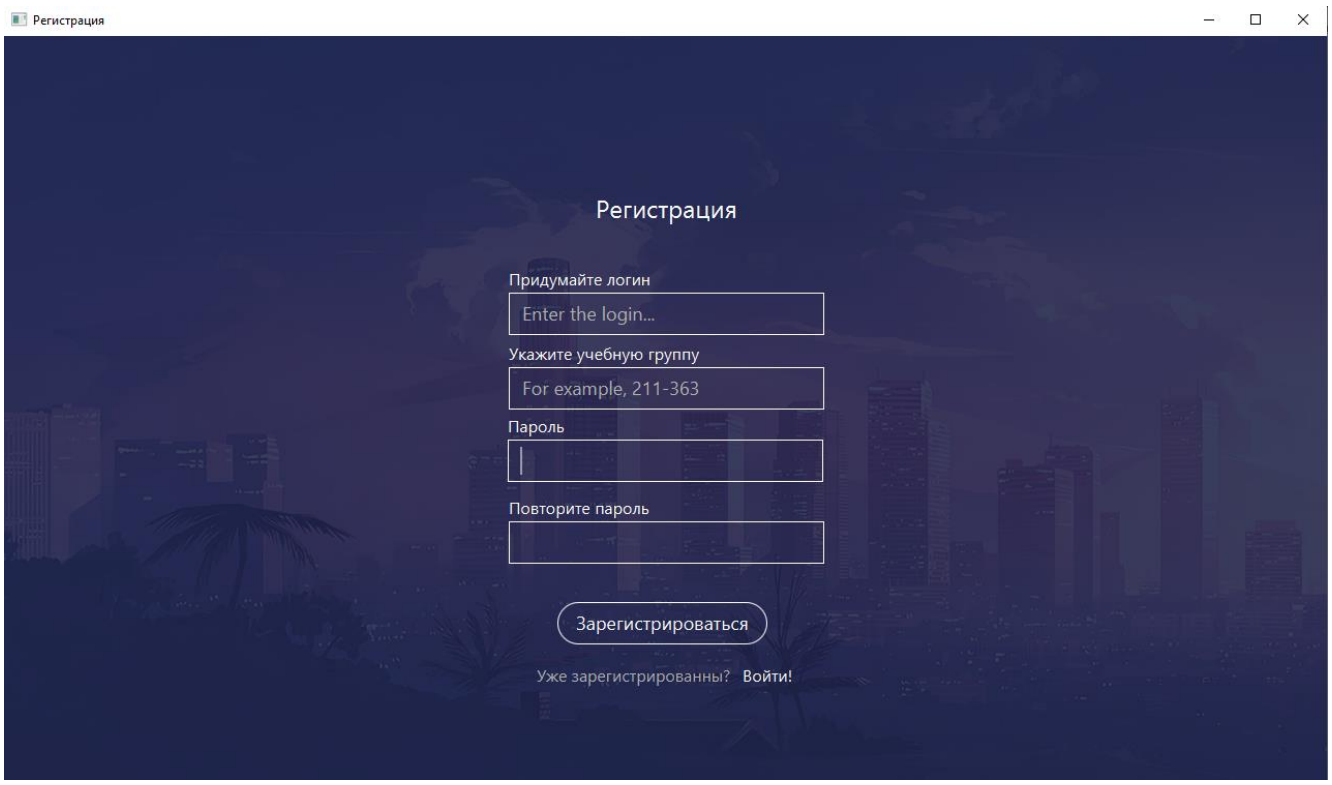

Рисунок 12 — Окно регистрации

В случае не заполнения всех полей, неправильного логина или пароля будет всплывать уведомление об ошибке с ее пояснением красным шрифтом. Об успешном выполнении операции также появляется уведомление с пояснением зеленым шрифтом.

При входе в систему под статусом «Гость» система заблокирует все кнопки управления, оставив лишь функцию «Обновить» для просмотра всех записей. Демонстрация главной страницы для гостя представлена на Рисунок 13.

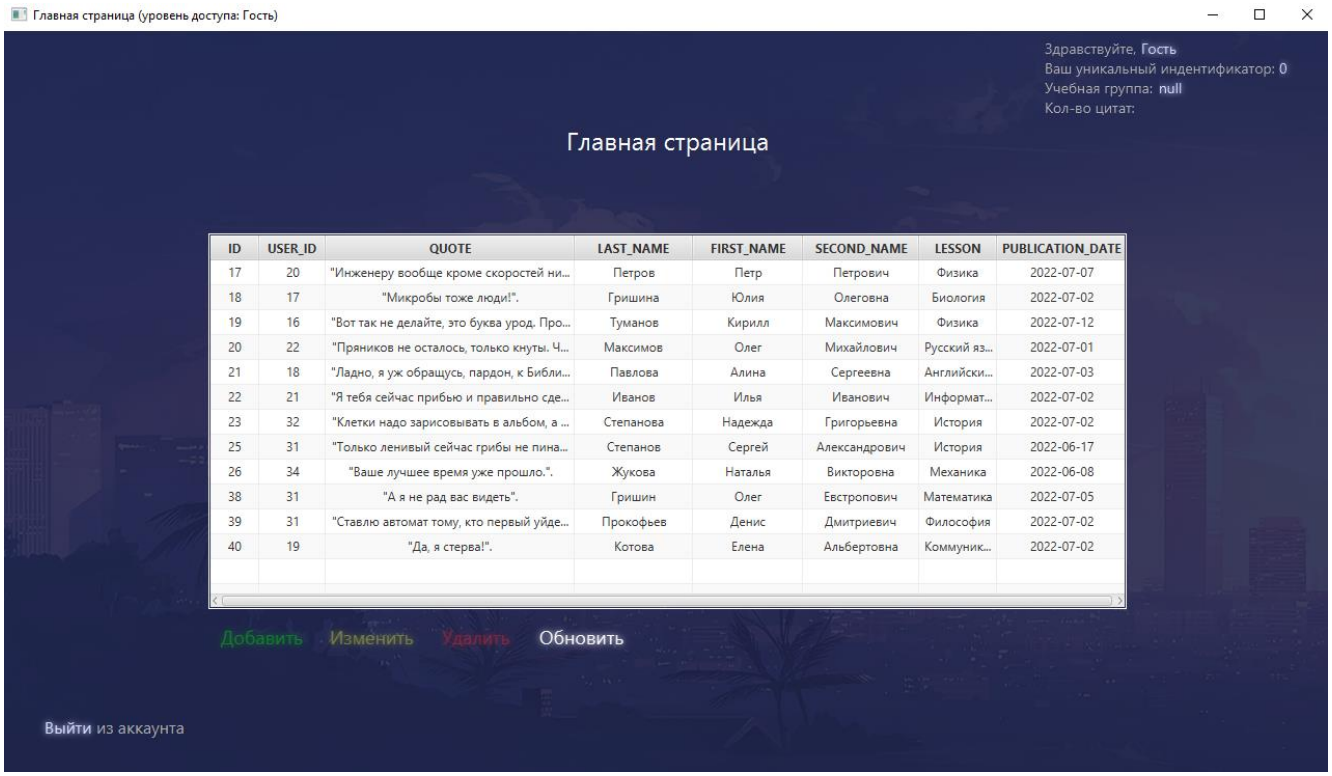

Рисунок 13 — Режим гостя

При авторизации с аккаунта с уровнем доступа рядового пользователя у владельца видны записи только его одногруппников, причем редактировать или удалить он может лишь свои записи. У каждого пользователя имеется уголок информации в правом верхнем углу, в котором отображается его логин, уникальный идентификатор, группа и количество добавленных им цитат. Демонстрация главной страницы для рядового пользователя представлена на Рисунок 14.

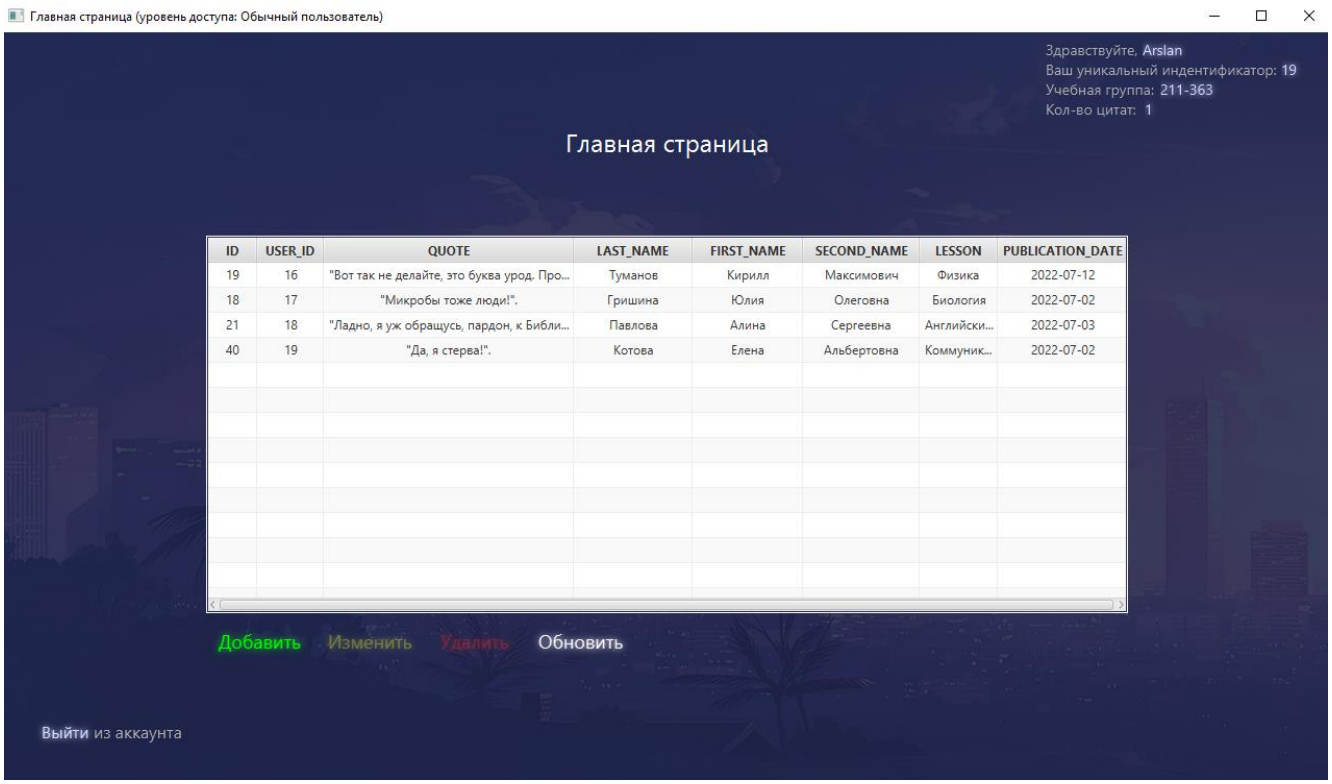

Рисунок 14 — Режим рядового пользователя

При нажатии на кнопку «Добавить» открывается дополнительное окно с формами для заполнения данных, которые подставятся в SQL-запрос, после чего он отправится на сервер и при нажатии кнопки «Обновить» данные в таблицы синхронизируются с базой данных. Демонстрация добавления представлена на Рисунок 15.

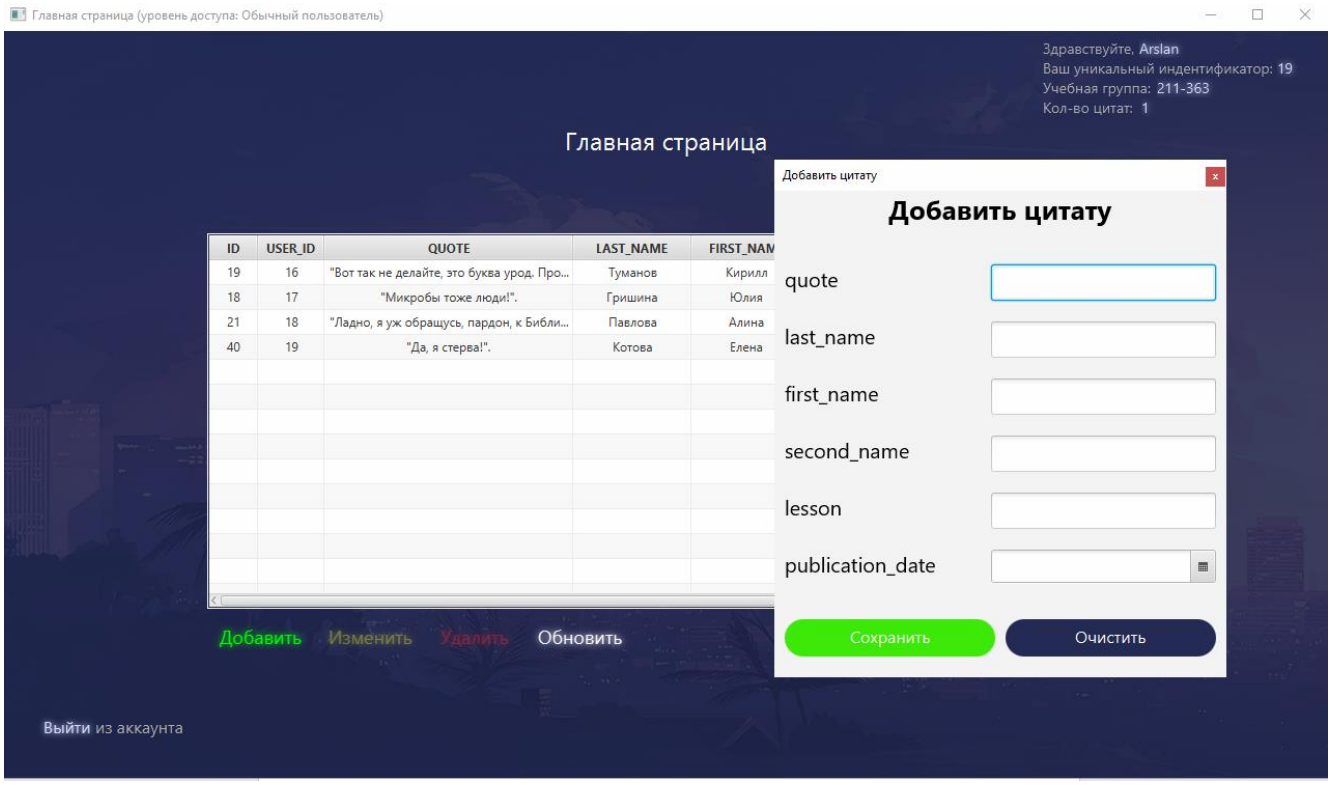

Рисунок 15 — Добавление цитаты

Чтобы изменить уже существующую цитату, необходимо выделить ее в таблице, затем нажать на кнопку «Изменить» и внести нужные данные. После чего отправится SQL-запрос на UPDATE таблицы, а атрибуты id и user_id автоматически определятся системой. В случае какой-либо ошибки или успехе появится уведомление в соответствующим цветом шрифта. Демонстрация работы редактирования представлена на Рисунок 16.

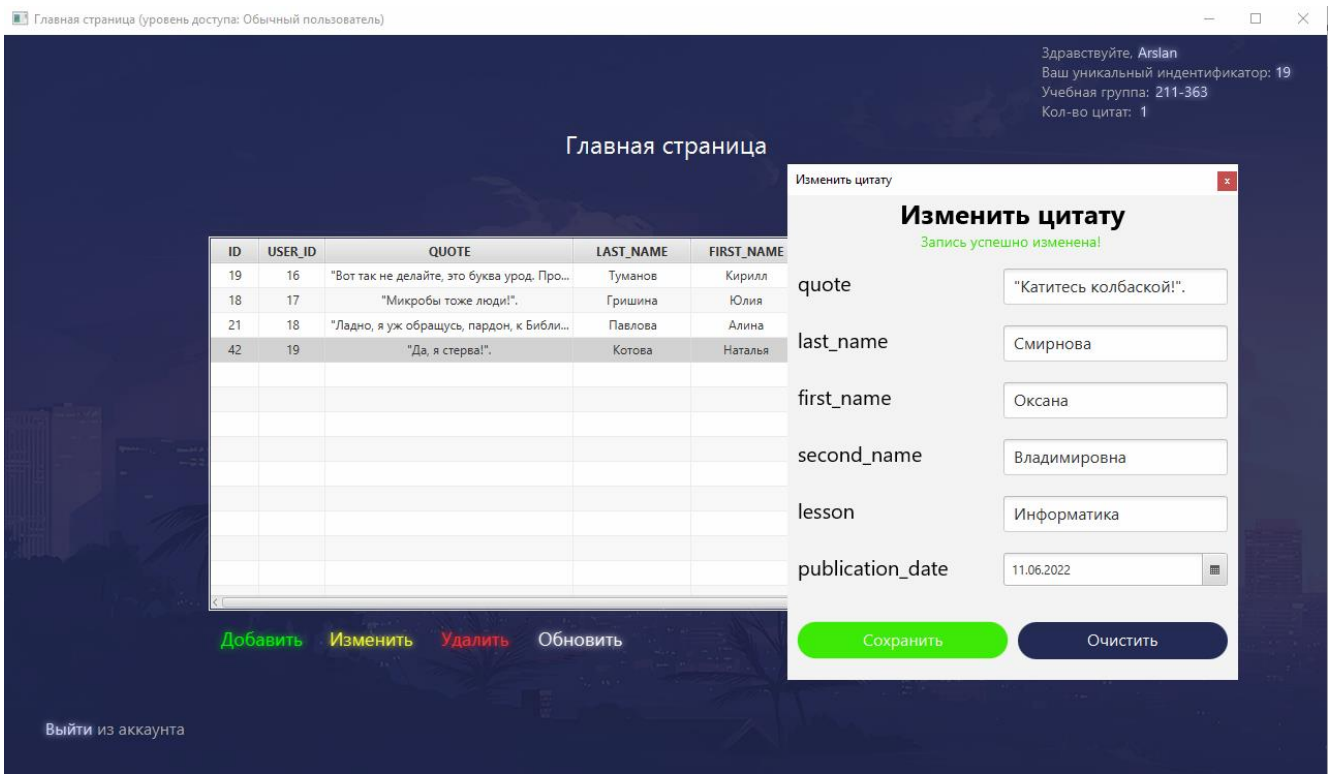

Рисунок 16 — Изменить цитату

Пользователи с уровнем доступа «Староста» также видят записи только своих одногруппников, но могут редактировать не только свои. Главная страница старосты представлена на Рисунок 17.

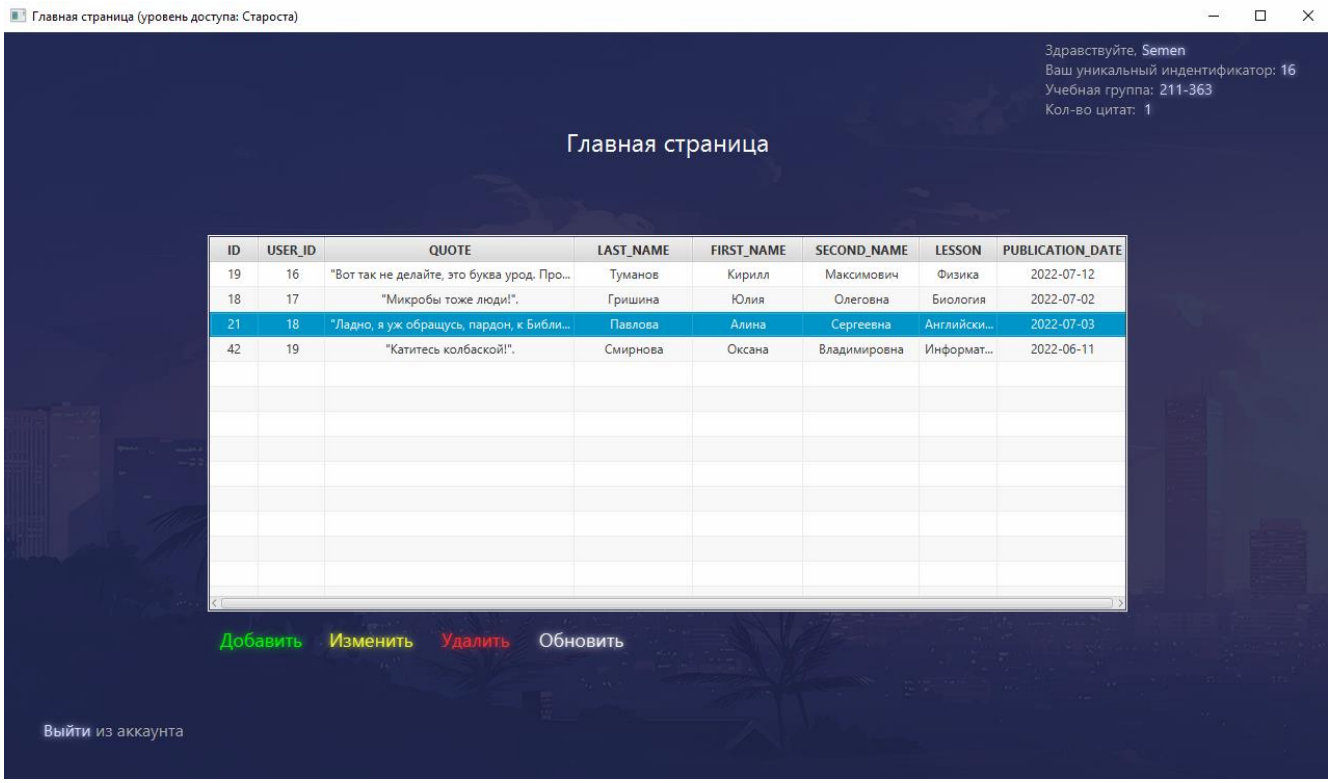

Рисунок 17 — Уровень доступа старосты

Суперпользователь, он же администратор, способен видеть и редактировать абсолютно все записи в таблице. Главная страница администратора представлена на Рисунок 18.

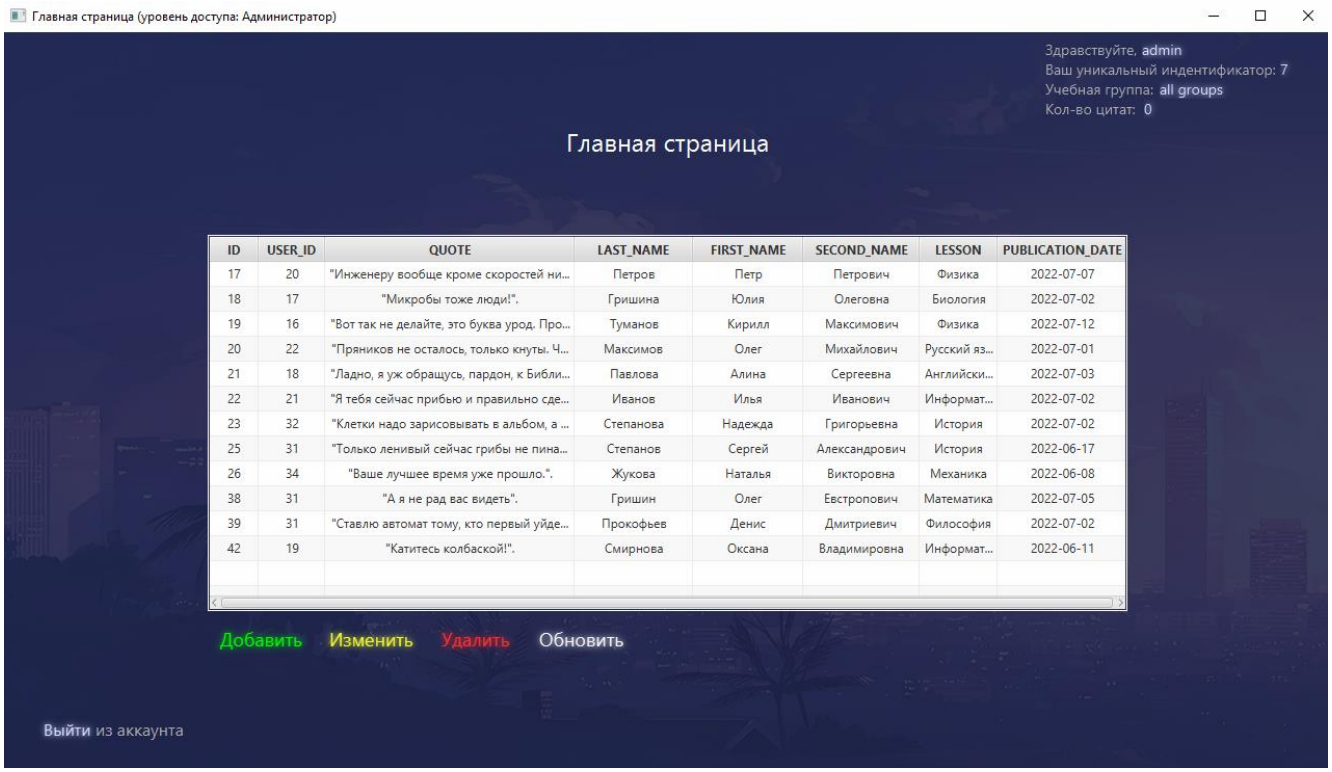

Рисунок 18 — Уровень доступа администратора

Заключение

Итогом разработки стало клиент-серверное приложение, написанное на языке Java, способное осуществлять соединение с базой данных MySQL, а также вносить в нее изменения на основе ограничительной кластеризационной модели.

Разработка приложения сопровождалась ведением удаленного репозитория посредством системы контроля версии Git, которая доступна по следующей ссылке на GitHub: https://github.com/scharkoff/db_access_restriction_models

Список литературы и интернет-ресурсы

1. MySQL. MySQL Connector/J 8.0 Developer Guide. [Электронный ресурс]. URL: https://dev.mysql.com/doc/connector-j/8.0/en/ (дата обращения: 7.07.2022).
2. JavaFX. Getting Started with JavaFX. [Электронный ресурс]. URL: https://openjfx.io/openjfx-docs/ (дата обращения: 07.07.2022).
3. Habr. Учебник по JavaFX: FXML и SceneBuilder. [Электронный ресурс]. URL: https://habr.com/ru/post/474982/ (дата обращения: 08.07.2022).
4. Interface PreparedStatement [Электронный ресурс]. URL: https://docs.oracle.com/en/java/javase/15/docs/api/java.sql/java/sql/ PreparedStatement.html (дата обращения: 08.07.2022).
5. Compile and build applications with IntelliJ IDEA [Электронный ресурс]. URL: https://www.jetbrains.com/help/idea/compiling-applications.html (дата обращения: 08.07.2022).

Приложения

1. Диаграмма классов

Рисунок 19 — Диаграмма классов приложения

2. Физическая модель базы данных

CREATE TABLE `teacher_quotes` (

`id` int(11) NOT NULL,

`user_id` int(11) NOT NULL,

`quote` text NOT NULL,

`last_name` text NOT NULL,

`first_name` text NOT NULL,

`second_name` text NOT NULL,

`lesson` text NOT NULL,

```
  `publication_date` date NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;


INSERT INTO `teacher_quotes` (`id`, `user_id`, `quote`, `last_name`, `first_name`, `second_name`, `lesson`, `publication_date`) VALUES
(16, 19, '\"Для того, чтобы что-то сделать, надо что-то сделать\".', 'Иванов', 'Иван', 'Иванович', 'Математика', '2022-07-01'),
(17, 20, '\"Инженеру вообще кроме скоростей ничего не нужно\".', 'Петров', 'Петр', 'Петрович', 'Физика', '2022-07-07'),
(18, 17, '\"Микробы тоже люди!\".', 'Гришина', 'Юлия', 'Олеговна', 'Биология', '2022-07-02'),
(19, 16, '\"Вот так не делайте, это буква урод. Просто буква-урод\".', 'Туманов', 'Кирилл', 'Максимович', 'Физика', '2022-07-12'),
(20, 22, '\"Пряников не осталось, только кнуты. Чего вы боитесь больше всего, признавайтесь. Буду давить на эти точки\".', 'Максимов', 'Олег', 'Михайлович', 'Русский язык', '2022-07-01'),
(21, 18, '\"Ладно, я уж обращусь, пардон, к Библии...\".', 'Павлова', 'Алина', 'Сергеевна', 'Английский язык', '2022-07-03'),
(22, 21, '\"Я тебя сейчас прибью и правильно сделаю\".', 'Иванов', 'Илья', 'Иванович', 'Информатика', '2022-07-02'),
(23, 32, '\"Клетки надо зарисовывать в альбом, а не в тетрадь. И на листы, а не в скетчбуки!\".', 'Степанова', 'Надежда', 'Григорьевна', 'История', '2022-07-02'),
(25, 31, '\"Только ленивый сейчас грибы не пинает\".', 'Степанов', 'Сергей', 'Александрович', 'История', '2022-06-17'),
```

(26, 34, '\"Ваше лучшее время уже прошло.\".', 'Жукова', 'Наталья', 'Викторовна', 'Механика', '2022-06-08');

```
CREATE TABLE `users` (
  `id` int(11) NOT NULL,
  `login` varchar(255) NOT NULL,
  `password` int(11) NOT NULL,
  `study_group` text NOT NULL,
  `rank` text
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

INSERT INTO `users` (`id`, `login`, `password`, `study_group`, `rank`) VALUES
(7, 'admin', 123, 'all groups', 'admin'),
(15, 'Eugene', 123, '211-363', 'user'),
(16, 'Semen', 123, '211-363', 'headman'),
(17, 'Liza', 123, '211-363', 'user'),
(18, 'Petya', 123, '211-363', 'user'),
(19, 'Arslan', 123, '211-363', 'user'),
(20, 'Daniel', 123, '211-362', 'user'),
(21, 'Oleg', 123, '211-362', 'user'),
(22, 'Polina', 123, '211-362', 'user'),
(25, 'Ivan', 123, '211-362', 'headman'),
```

```
(26, 'Andrey', 123, '211-363', 'user'),
(30, 'Nikita', 123, '211-362', 'user'),
(31, 'Gleb', 123, '211-362', 'user'),
(32, 'Ira', 123, '211-362', 'user'),
(33, 'Stepa', 123, '211-362', 'user'),
(34, 'Katya', 123, '211-362', 'user'),
(35, 'Vika', 123, '211-363', 'user');

ALTER TABLE `teacher_quotes`
  ADD PRIMARY KEY (`id`),
  ADD KEY `id` (`user_id`);

ALTER TABLE `users`
  ADD PRIMARY KEY (`id`),
  ADD UNIQUE KEY `login` (`login`);

ALTER TABLE `teacher_quotes`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=37;

ALTER TABLE `users`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=41;
```

```
ALTER TABLE `teacher_quotes`
  ADD CONSTRAINT `teacher_quotes_ibfk_1` FOREIGN KEY (`user_id`) REFERENCES `users` (`id`);

COMMIT;
```