

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені Ігоря Сікорського»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

**Кафедра системного програмування та спеціалізованих комп’ютерних систем**

**Лабораторна робота №2**

з дисципліни **Бази даних і засоби управління**

*на тему: “Створення додатку бази даних, орієнтованого на взаємодію з СУБД PostgreSQL”*

Виконав:

студент ІII курсу

групи КВ-91

Русаков М. В.

Перевірив:

Павловський В. І.

Київ – 2021

Завдання роботи полягає у наступному:

* + - 1. Реалізувати функції перегляду, внесення, редагування та вилучення даних у таблицях бази даних, створених у лабораторній роботі №1, засобами консольного інтерфейсу.
      2. Передбачити автоматичне пакетне генерування «рандомізованих» даних у базі.
      3. Передбачити автоматичне пакетне генерування «рандомізованих» даних у базі. Забезпечити реалізацію пошуку за декількома атрибутами з двох та більше сутностей одночасно: для числових атрибутів – у рамках діапазону, для рядкових – як шаблон функції LIKE оператора SELECT SQL, для логічного типу – значення True/False, для дат – у рамках діапазону дат.
      4. Програмний код виконати згідно шаблону MVC (модель-подання-контролер).

Деталізоване завдання:

Забезпечити можливість уведення/редагування/вилучення даних у таблицях бази даних з можливістю контролю відповідності типів даних атрибутів таблиць (рядків, чисел, дати/часу). Для контролю пропонується два варіанти: контроль при введенні (валідація даних) та перехоплення помилок (try..except) від сервера PostgreSQL при виконанні відповідної команди SQL. Особливу увагу варто звернути на дані таблиць, що мають зв’язок 1:N. При цьому з боку батьківської таблиці необхідно контролювати вилучення рядків за умови наявності даних у підлеглій таблиці. З точки зору підлеглої таблиці варто контролювати наявність відповідного рядка у батьківській таблиці при виконанні внесення нових даних. Унеможливити виведення програмою системних помилок на екрані шляхом їх перехоплення і адекватної обробки. Внесення даних виконується користувачем у консольному вікні програми.

Забезпечити можливість автоматичної генерації великої кількості даних у таблицях за допомогою вбудованих у PostgreSQL функцій роботи з псевдовипадковими числами. Дані мають бути згенерованими не мовою програмування, а відповідним SQL-запитом!

Для реалізації пошуку необхідно підготувати 3 запити, що включають дані з декількох таблиць і фільтрують рядки за 3-4 атрибутами цих таблиць. Забезпечити можливість уведення конкретних значень констант для фільтрації з клавіатури користувачем. Крім того, після виведення даних необхідно вивести час виконання запиту у мілісекундах. Перевірити швидкодію роботи запитів на попередньо згенерованих даних.

Програмний код організувати згідно шаблону Model-View-Controller(MVC). Приклад організації коду згідно шаблону доступний за даним посиланням. При цьому модель, подання та контролер мають бути реалізовані у окремих файлах. Для доступу до бази даних використовувати лише мову SQL (без ORM).

**Логічна модель бази даних**

Нижче (Рис. 1) наведено логічну модель бази даних:

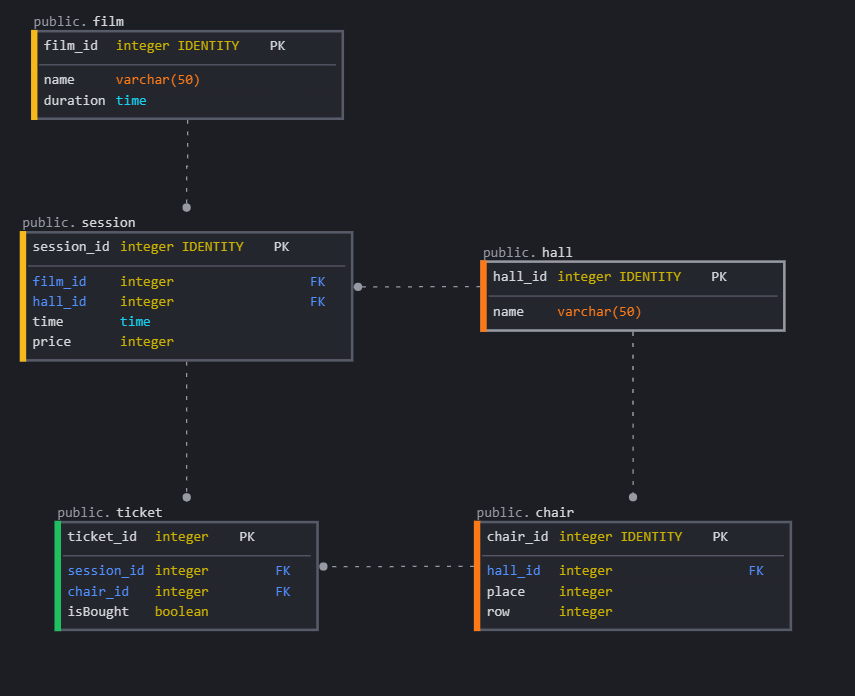


Рис. 1 – Логічна модель бази даних

Зміни у порівнянні з першою лабораторною роботою відсутні.

**Середовище розробки та налаштування підключення до бази даних**

Для виконання лабораторної роботи використовувалась мова програмування Java та середовище розробки IntelliJ IDEA.

Для підключення до серверу бази даних PostgreSQL використано прикладний програмний інтерфейс JDBC.

**Шаблон проектування**

MVC – шаблон проектування, що використовується у програмі.

Model – представляє клас, що описує логіку використовуваних даних. Це файл Model.java.

View – відповідає за ввід/вивід інформації. Це файл View.fxml.  
Controller – Представляє клас, що обробляє ввід користувача, подає відповідну команду до Model.java і очікує результат, який потім виводить користувачу. Це файл Controller.java.

**Структура програми та її зміст**

Нижче (Рис. 2) наведено структуру програми:

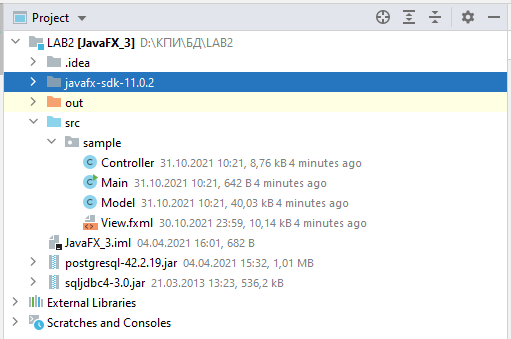


Рис. 2 – Структура програми

1. Main.java – точка входу в програму, запускає початковий інтерфейс.
2. View.fxml – файл, що відповідає за відображення програми у вікні, та приймає ввід/вивід користувача.
3. Controller.java – обробляє ввід користувача, подає відповідну команду до Model.java і очікує результат, який потім виводить користувачу.
4. Model.java – виконує складні операції з БД з конкретними таблицями БД.

**Структура меню програми**

Нижче (Рис. 3) наведено структуру меню програми:

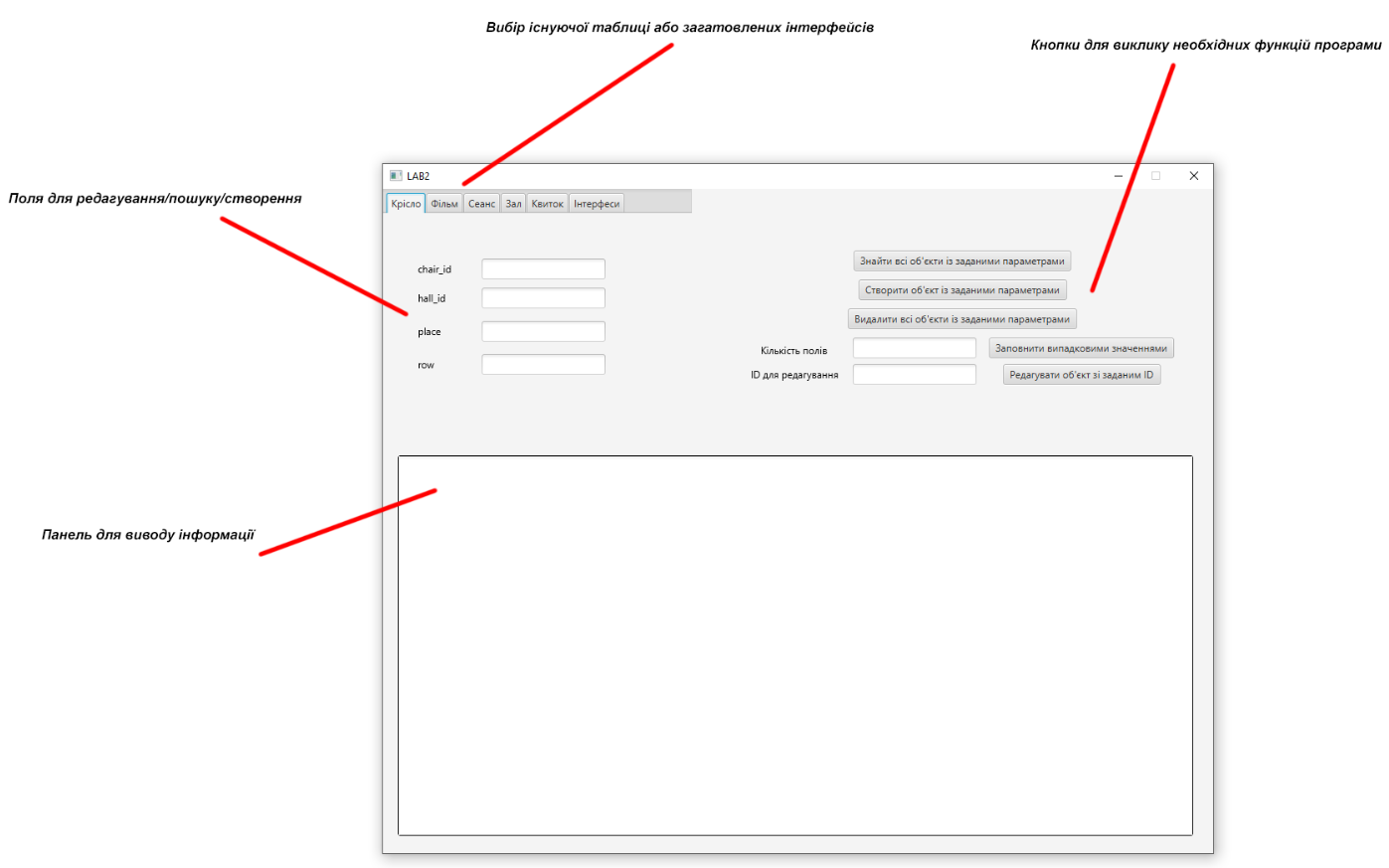


Рис. 3 – Структура меню програми

Нижче (Рис. 4) наведено реалізація інтерфейсів в меню:

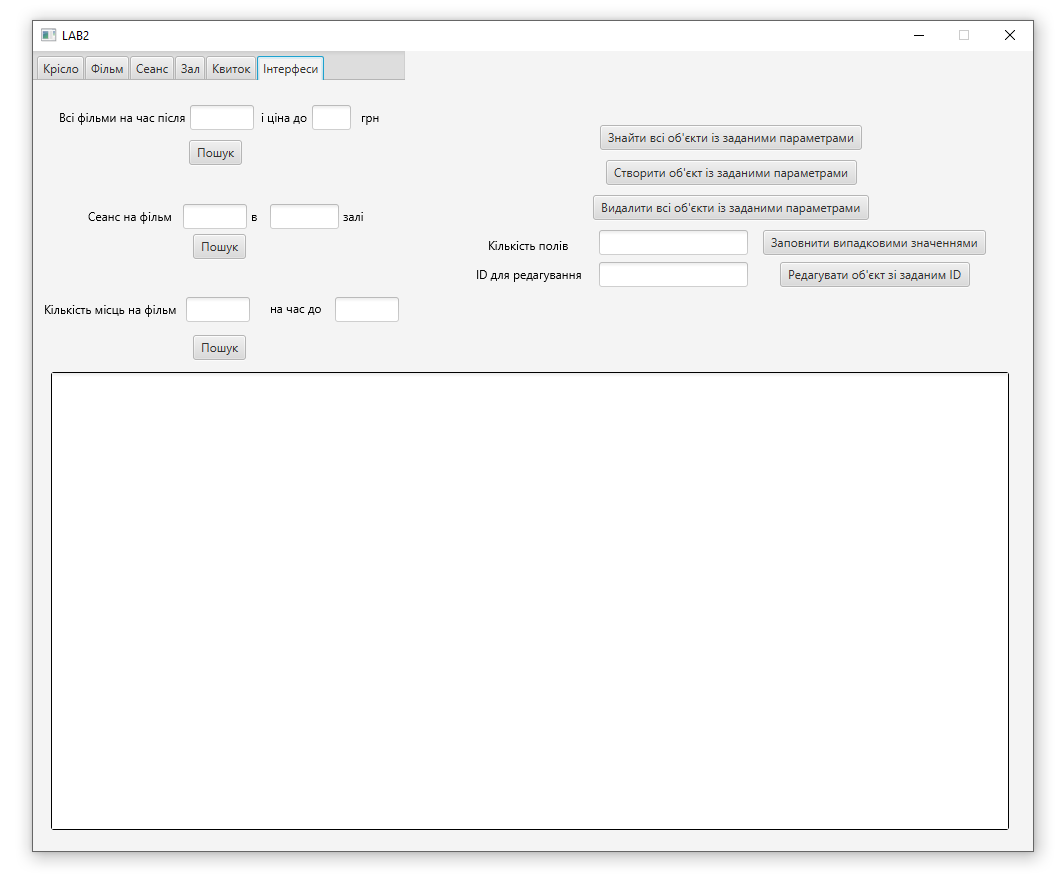


Рис. 3 – Реалізація інтерфейсів в меню

**Лістинг фрагментів програми з запитом пошуку**

Пошук всіх крісел із заданими параметрами, якщо параметри відсутні, то знаходить всі крісла з таблиці. Є контроль введених даних на коректність типу. У разі будь-якої помилки повертає повну інформацію про помилку. Результат, що повертає ця функція, виводиться користувачу.

public static String findAllChair(Connection connection, String chair\_id, String hall\_id, String place, String row) throws SQLException {  
 if (!chair\_id.matches("\\d+|") || !hall\_id.matches("\\d+|") || !place.matches("\\d+|") || !row.matches("\\d+|")) {  
 return "Некоректний ввід даних!";  
 }  
  
 String sql = "SELECT *\** FROM public.chair WHERE";  
 sql += " chair\_id = " + (chair\_id.length() > 0 ? chair\_id : "chair\_id");  
 sql += " and hall\_id = " + (hall\_id.length() > 0 ? hall\_id : "hall\_id");  
 sql += " and place = " + (place.length() > 0 ? place : "place");  
 sql += " and row = " + (row.length() > 0 ? row : "row");  
  
 System.*out*.println("sql: " + sql);  
  
 try (PreparedStatement statement = connection.prepareStatement(sql)) {  
 String resultStr = "Результат пошуку по заданим параметрам:\n";  
 try {  
 ResultSet = statement.executeQuery();  
 while (resultSet.next()) {  
 resultStr += "-> chair\_id = " + resultSet.getInt(1);  
 resultStr += " hall\_id = " + resultSet.getInt(2);  
 resultStr += " place = " + resultSet.getInt(3);  
 resultStr += " row = " + resultSet.getInt(4);  
 resultStr += "\n";  
 }  
 } catch (Exception ex) {  
 return ex.getMessage();  
 }  
 return (!resultStr.equals("Результат пошуку по заданим параметрам:\n") ? resultStr : "Нічого не знайдено!");  
 }  
}

Пошук всіх фільмів із заданими параметрами, якщо параметри відсутні, то знаходить всі фільми з таблиці. Є контроль введених даних на коректність типу. У разі будь-якої помилки повертає повну інформацію про помилку. Результат, що повертає ця функція, виводиться користувачу.

public static String findAllFilm(Connection connection, String film\_id, String name, String duration) throws SQLException {  
  
 if (!duration.matches("")) {  
 SimpleDateFormat sdf = new SimpleDateFormat("hh:mm:ss");  
 try {  
 Date Dateduration = sdf.parse(duration);  
 } catch (Exception ex) {  
 return "Час заданий некроектно";  
 }  
 }  
 if (!film\_id.matches("\\d+|")) {  
 return "Некоректний ввід даних!";  
 }  
  
 String sql = "SELECT *\** FROM public.film WHERE";  
 sql += " film\_id = " + (film\_id.length() > 0 ? film\_id : "film\_id");  
 sql += " and name like " + (name.length() > 0 ? ("'%" + name + "%'") : "name");  
 sql += " and duration = " + (duration.length() > 0 ? ("'" + duration + "'") : "duration");  
  
 System.*out*.println("sql: " + sql);  
  
 try (PreparedStatement statement = connection.prepareStatement(sql)) {  
 String resultStr = "Результат пошуку по заданим параметрам:\n";  
 try {  
 ResultSet resultSet = statement.executeQuery();  
 while (resultSet.next()) {  
 resultStr += "-> film\_id = " + resultSet.getInt(1);  
 resultStr += " name = " + resultSet.getString(2);  
 resultStr += " duration = " + resultSet.getTime(3);  
 resultStr += "\n";  
 }  
 } catch (Exception ex) {  
 return ex.getMessage();  
 }  
 return (!resultStr.equals("Результат пошуку по заданим параметрам:\n") ? resultStr : "Нічого не знайдено!");  
 }  
}

Пошук всіх сеансів із заданими параметрами, якщо параметри відсутні, то знаходить всі сеанси з таблиці. Є контроль введених даних на коректність типу. У разі будь-якої помилки повертає повну інформацію про помилку. Результат, що повертає ця функція, виводиться користувачу.

public static String findAllSession(Connection connection, String session\_id, String film\_id, String hall\_id, String time, String price) throws SQLException {  
  
 if (!time.matches("")) {  
 SimpleDateFormat sdf = new SimpleDateFormat("hh:mm:ss");  
 try {  
 Date Dateduration = sdf.parse(time);  
 } catch (Exception ex) {  
 return "Час заданий некроектно";  
 }  
 }  
 if (!session\_id.matches("\\d+|") || !film\_id.matches("\\d+|") || !hall\_id.matches("\\d+|") || !price.matches("[+]?\\d+|")) {  
 return "Некоректний ввід даних!";  
 }  
  
 String sql = "SELECT *\** FROM public.session WHERE";  
 sql += " session\_id = " + (session\_id.length() > 0 ? session\_id : "session\_id");  
 sql += " and film\_id = " + (film\_id.length() > 0 ? film\_id : "film\_id");  
 sql += " and hall\_id = " + (hall\_id.length() > 0 ? hall\_id : "hall\_id");  
 sql += " and time = " + (time.length() > 0 ? ("'" + time + "'") : "time");  
 sql += " and price = " + (price.length() > 0 ? price : "price");  
  
 System.*out*.println("sql: " + sql);  
  
 try (PreparedStatement statement = connection.prepareStatement(sql)) {  
 String resultStr = "Результат пошуку по заданим параметрам:\n";  
 try {  
 ResultSet resultSet = statement.executeQuery();  
 while (resultSet.next()) {  
 resultStr += "-> session\_id = " + resultSet.getInt(1);  
 resultStr += " film\_id = " + resultSet.getInt(2);  
 resultStr += " hall\_id = " + resultSet.getInt(3);  
 resultStr += " time = " + resultSet.getTime(4);  
 resultStr += " price = " + resultSet.getInt(5);  
 resultStr += "\n";  
 }  
 } catch (Exception ex) {  
 return ex.getMessage();  
 }  
 return (!resultStr.equals("Результат пошуку по заданим параметрам:\n") ? resultStr : "Нічого не знайдено!");  
 }  
}

Пошук всіх залів із заданими параметрами, якщо параметри відсутні, то знаходить всі зали з таблиці. Є контроль введених даних на коректність типу. У разі будь-якої помилки повертає повну інформацію про помилку. Результат, що повертає ця функція, виводиться користувачу.

public static String findAllHall(Connection connection, String hall\_id, String name) throws SQLException {  
 if (!hall\_id.matches("[+]?\\d+|")) {  
 return "Некоректний ввід даних!";  
 }  
  
 String sql = "SELECT *\** FROM public.hall WHERE";  
 sql += " hall\_id = " + (hall\_id.length() > 0 ? hall\_id : "hall\_id");  
 sql += " and name like " + (name.length() > 0 ? ("'%" + name + "%'") : "name");  
  
 System.*out*.println("sql: " + sql);  
  
 try (PreparedStatement statement = connection.prepareStatement(sql)) {  
 String resultStr = "Результат пошуку по заданим параметрам:\n";  
 try {  
 ResultSet resultSet = statement.executeQuery();  
 while (resultSet.next()) {  
 resultStr += "-> hall\_id = " + resultSet.getInt(1);  
 resultStr += " name = " + resultSet.getString(2);  
 resultStr += "\n";  
 }  
 } catch (Exception ex) {  
 return ex.getMessage();  
 }  
 return (!resultStr.equals("Результат пошуку по заданим параметрам:\n") ? resultStr : "Нічого не знайдено!");  
 }  
}

Пошук всіх квитків із заданими параметрами, якщо параметри відсутні, то знаходить всі квитки з таблиці. Є контроль введених даних на коректність типу. У разі будь-якої помилки повертає повну інформацію про помилку. Результат, що повертає ця функція, виводиться користувачу.

public static String findAllTicket(Connection connection, String ticket\_id, String session\_id, String chair\_id, String isbought) throws SQLException {  
  
 if (!ticket\_id.matches("\\d+|") || !session\_id.matches("\\d+|") || !chair\_id.matches("\\d+|") || !isbought.matches("true|false|")) {  
 return "Некоректний ввід даних!";  
 }  
  
 String sql = "SELECT *\** FROM public.ticket WHERE";  
 sql += " ticket\_id = " + (ticket\_id.length() > 0 ? ticket\_id : "ticket\_id");  
 sql += " and session\_id = " + (session\_id.length() > 0 ? session\_id : "session\_id");  
 sql += " and chair\_id = " + (chair\_id.length() > 0 ? chair\_id : "chair\_id");  
 sql += " and isbought = " + (isbought.length() > 0 ? isbought : "isbought");  
  
 System.*out*.println("sql: " + sql);  
  
 try (PreparedStatement statement = connection.prepareStatement(sql)) {  
 String resultStr = "Результат пошуку по заданим параметрам:\n";  
 try {  
 ResultSet resultSet = statement.executeQuery();  
 while (resultSet.next()) {  
 resultStr += "-> ticket\_id = " + resultSet.getInt(1);  
 resultStr += " session\_id = " + resultSet.getInt(2);  
 resultStr += " chair\_id = " + resultSet.getInt(3);  
 resultStr += " isbought = " + resultSet.getBoolean(4);  
 resultStr += "\n";  
 }  
 } catch (Exception ex) {  
 return ex.getMessage();  
 }  
 return (!resultStr.equals("Результат пошуку по заданим параметрам:\n") ? resultStr : "Нічого не знайдено!");  
 }  
}

**Лістинг фрагментів програми з запитом створення**

Створення крісла із заданими параметрами. Якщо не вистачає хоч одного параметра, запит не буде виконуватись. Є контроль введених даних на коректність типу. У разі будь-якої помилки повертає повну інформацію про помилку. Результат, що повертає ця функція, виводиться користувачу.

public static String createChair(Connection connection, String chair\_id, String hall\_id, String place, String row) throws SQLException {  
 if (!chair\_id.matches("\\d+") || !hall\_id.matches("\\d+") || !place.matches("\\d+") || !row.matches("\\d+")) {  
 return "Некоректний ввід даних!";  
 }  
String sql = "INSERT INTO public.chair (chair\_id,hall\_id, place, row) VALUES (";  
 sql += chair\_id + ", ";  
 sql += hall\_id + ", ";  
 sql += place + ", ";  
 sql += row + ")";  
  
 System.*out*.println("sql: " + sql);  
 try {  
 Statement statement = connection.createStatement();  
 statement.executeUpdate(sql);  
 } catch (Exception ex) {  
 return ex.getMessage();  
 }  
  
 String answer = "Створено:\n";  
 answer += "-> chair\_id = " + chair\_id;  
 answer += " hall\_id = " + hall\_id;  
 answer += " place = " + place;  
 answer += " row = " + row;  
  
 return answer;  
}

Створення фільму із заданими параметрами. Якщо не вистачає хоч одного параметра, запит не буде виконуватись. Є контроль введених даних на коректність типу. У разі будь-якої помилки повертає повну інформацію про помилку. Результат, що повертає ця функція, виводиться користувачу.

public static String createFilm(Connection connection, String film\_id, String name, String duration) throws SQLException {  
 SimpleDateFormat sdf = new SimpleDateFormat("hh:mm:ss");  
 try {  
 Date Dateduration = sdf.parse(duration);  
 } catch (Exception ex) {  
 return "Час заданий некроектно";  
 }  
  
 if (!film\_id.matches("\\d+") || name.equals("")) {  
 return "Некоректний ввід даних!";  
 }  
  
 String sql = "INSERT INTO public.film (film\_id, name, duration) VALUES (";  
 sql += film\_id + ", ";  
 sql += "'" + name + "', ";  
 sql += "'" + duration + "')";  
  
 System.*out*.println("sql: " + sql);  
 try {  
 Statement statement = connection.createStatement();  
 statement.executeUpdate(sql);  
 } catch (Exception ex) {  
 return ex.getMessage();  
 }  
  
 String answer = "Створено:\n";  
 answer += "-> film\_id = " + film\_id;  
 answer += " name = " + name;  
 answer += " duration = " + duration;  
  
 return answer;  
}

Створення сеансу із заданими параметрами. Якщо не вистачає хоч одного параметра, запит не буде виконуватись. Є контроль введених даних на коректність типу. У разі будь-якої помилки повертає повну інформацію про помилку. Результат, що повертає ця функція, виводиться користувачу.

public static String createSession(Connection connection, String session\_id, String film\_id, String hall\_id, String time, String price) throws SQLException {  
 SimpleDateFormat sdf = new SimpleDateFormat("hh:mm:ss");  
 try {  
 Date Dateduration = sdf.parse(time);  
 } catch (Exception ex) {  
 return "Час заданий некроектно";  
 }  
  
 if (!session\_id.matches("\\d+") || !film\_id.matches("\\d+") || !hall\_id.matches("\\d+") || !price.matches("\\d+")) {  
 return "Некоректний ввід даних!";  
 }  
  
 String sql = "INSERT INTO public.session (session\_id, film\_id, hall\_id, time, price) VALUES (";  
 sql += session\_id + ", ";  
 sql += film\_id + ", ";  
 sql += hall\_id + ", ";  
 sql += "'" + time + "',";  
 sql += price + ")";  
  
 System.*out*.println("sql: " + sql);  
 try {  
 Statement statement = connection.createStatement();  
 statement.executeUpdate(sql);  
 } catch (Exception ex) {  
 return ex.getMessage();  
 }  
  
 String answer = "Створено:\n";  
 answer += "-> session\_id = " + session\_id;  
 answer += " film\_id = " + film\_id;  
 answer += " hall\_id = " + hall\_id;  
 answer += " time = " + time;  
 answer += " price = " + price;  
  
 return answer;  
}

Створення залу із заданими параметрами. Якщо не вистачає хоч одного параметра, запит не буде виконуватись. Є контроль введених даних на коректність типу. У разі будь-якої помилки повертає повну інформацію про помилку. Результат, що повертає ця функція, виводиться користувачу.

public static String createHall(Connection connection, String hall\_id, String name) throws SQLException {  
  
 if (!hall\_id.matches("\\d+") || name.equals("")) {  
 return "Некоректний ввід даних!";  
 }  
  
 String sql = "INSERT INTO public.hall (hall\_id, name) VALUES (";  
 sql += hall\_id + ", ";  
 sql += "'" + name + "')";  
  
 System.*out*.println("sql: " + sql);  
 try {  
 Statement statement = connection.createStatement();  
 statement.executeUpdate(sql);  
 } catch (Exception ex) {  
 return ex.getMessage();  
 }  
  
 String answer = "Створено:\n";  
 answer += "-> hall\_id = " + hall\_id;  
 answer += " name = " + name;  
  
 return answer;  
}

Створення квитка із заданими параметрами. Якщо не вистачає хоч одного параметра, запит не буде виконуватись. Є контроль введених даних на коректність типу. У разі будь-якої помилки повертає повну інформацію про помилку. Результат, що повертає ця функція, виводиться користувачу.

public static String createTicket(Connection connection, String ticket\_id, String session\_id, String chair\_id, String isbought) throws SQLException {  
 if (!ticket\_id.matches("\\d+") || !session\_id.matches("\\d+") || !chair\_id.matches("\\d+") || !isbought.matches("true|false|")) {  
 return "Некоректний ввід даних!";  
 }  
  
 String sql = "INSERT INTO public.ticket (ticket\_id, session\_id, chair\_id, isbought) VALUES (";  
 sql += ticket\_id + ", ";  
 sql += session\_id + ", ";  
 sql += chair\_id + ", ";  
 sql += isbought + ")";  
  
 System.*out*.println("sql: " + sql);  
 try {  
 Statement statement = connection.createStatement();  
 statement.executeUpdate(sql);  
 } catch (Exception ex) {  
 return ex.getMessage();  
 }  
  
 String answer = "Створено:\n";  
 answer += "-> ticket\_id = " + ticket\_id;  
 answer += " session\_id = " + session\_id;  
 answer += " chair\_id = " + chair\_id;  
 answer += " isbought = " + isbought;  
  
 return answer;  
}

**Лістинг фрагментів програми з запитом вилучення**

Видаляє всі крісла з таблиці із заданими параметрами, якщо параметри відсутні то видаляються всі значення з таблиці. Видаляються не лише об’єкти в самій таблиці, а й всі інші об’єкти, що використовували значення, що повинні бути видаленими. Є контроль введених даних на коректність типу. У разі будь-якої помилки повертає повну інформацію про помилку. Результат, що повертає ця функція, виводиться користувачу.

public static String deleteChair(Connection connection, String chair\_id, String hall\_id, String place, String row) throws SQLException {  
 if (!chair\_id.matches("\\d+|") || !hall\_id.matches("\\d+|") || !place.matches("\\d+|") || !row.matches("\\d+|")) {  
 return "Некоректний ввід даних!";  
 }  
  
 String sql = "DELETE from ticket where chair\_id in (select chair\_id from chair where";  
 sql += " chair\_id = " + (chair\_id.length() > 0 ? chair\_id : "chair\_id");  
 sql += " and hall\_id = " + (hall\_id.length() > 0 ? hall\_id : "hall\_id");  
 sql += " and place = " + (place.length() > 0 ? place : "place");  
 sql += " and row = " + (row.length() > 0 ? row : "row") + ");\n";  
  
  
 sql += "DELETE from chair where";  
 sql += " chair\_id = " + (chair\_id.length() > 0 ? chair\_id : "chair\_id");  
 sql += " and hall\_id = " + (hall\_id.length() > 0 ? hall\_id : "hall\_id");  
 sql += " and place = " + (place.length() > 0 ? place : "place");  
 sql += " and row = " + (row.length() > 0 ? row : "row") + ";";  
  
 System.*out*.println("sql: " + sql);  
  
 try {  
 Statement statement = connection.createStatement();  
 statement.executeUpdate(sql);  
 } catch (Exception ex) {  
 return ex.getMessage();  
 }  
  
 String answer = "Видалено все, де:\n->";  
 if (chair\_id.length() > 0) answer += " chair\_id = " + chair\_id;  
 if (hall\_id.length() > 0) answer += " hall\_id = " + hall\_id;  
 if (place.length() > 0) answer += " place = " + place;  
 if (row.length() > 0) answer += " row = " + row;  
  
 return answer;  
}

Видаляє всі фільми з таблиці із заданими параметрами, якщо параметри відсутні то видаляються всі значення з таблиці. Видаляються не лише об’єкти в самій таблиці, а й всі інші об’єкти, що використовували значення, що повинні бути видаленими. Є контроль введених даних на коректність типу. У разі будь-якої помилки повертає повну інформацію про помилку. Результат, що повертає ця функція, виводиться користувачу.

public static String deleteFilm(Connection connection, String film\_id, String name, String duration) throws SQLException {  
 if (!duration.matches("")) {  
 SimpleDateFormat sdf = new SimpleDateFormat("hh:mm:ss");  
 try {  
 Date Dateduration = sdf.parse(duration);  
 } catch (Exception ex) {  
 return "Час заданий некроектно";  
 }  
 }  
 if (!film\_id.matches("\\d+|")) {  
 return "Некоректний ввід даних!";  
 }  
  
 String sql = "DELETE from ticket where session\_id in (select session\_id from session where film\_id in (select film\_id from film where";  
 sql += " film\_id = " + (film\_id.length() > 0 ? film\_id : "film\_id");  
 sql += " and name like " + (name.length() > 0 ? "'" + name + "'" : "name");  
 sql += " and duration = " + (duration.length() > 0 ? "'" + duration + "'" : "duration") + "));\n";  
  
 sql += "DELETE from session where film\_id in (select film\_id from film where";  
 sql += " film\_id = " + (film\_id.length() > 0 ? film\_id : "film\_id");  
 sql += " and name like " + (name.length() > 0 ? "'" + name + "'" : "name");  
 sql += " and duration = " + (duration.length() > 0 ? "'" + duration + "'" : "duration") + ");\n";  
  
 sql += "DELETE from film where";  
 sql += " film\_id = " + (film\_id.length() > 0 ? film\_id : "film\_id");  
 sql += " and name like " + (name.length() > 0 ? "'" + name + "'" : "name");  
 sql += " and duration = " + (duration.length() > 0 ? "'" + duration + "'" : "duration") + ";";  
  
 System.*out*.println("sql: " + sql);  
  
 try {  
 Statement statement = connection.createStatement();  
 statement.executeUpdate(sql);  
 } catch (Exception ex) {  
 return ex.getMessage();  
 }  
  
 String answer = "Видалено все, де:\n->";  
 if (film\_id.length() > 0) answer += " film\_id = " + film\_id;  
 if (name.length() > 0) answer += " name = " + name;  
 if (duration.length() > 0) answer += " duration = " + duration;  
  
 return answer;  
}

Видаляє всі сеанси з таблиці із заданими параметрами, якщо параметри відсутні то видаляються всі значення з таблиці. Видаляються не лише об’єкти в самій таблиці, а й всі інші об’єкти, що використовували значення, що повинні бути видаленими. Є контроль введених даних на коректність типу. У разі будь-якої помилки повертає повну інформацію про помилку. Результат, що повертає ця функція, виводиться користувачу.

public static String deleteSession(Connection connection, String session\_id, String film\_id, String hall\_id, String time, String price) throws SQLException {  
 if (!time.matches("")) {  
 SimpleDateFormat sdf = new SimpleDateFormat("hh:mm:ss");  
 try {  
 Date Dateduration = sdf.parse(time);  
 } catch (Exception ex) {  
 return "Час заданий некроектно";  
 }  
 }  
 if (!session\_id.matches("\\d+|") || !film\_id.matches("\\d+|") || !hall\_id.matches("\\d+|") || !price.matches("\\d+|")) {  
 return "Некоректний ввід даних!";  
 }  
  
 String sql = "DELETE from ticket where session\_id in (select session\_id from session where";  
 sql += " session\_id = " + (session\_id.length() > 0 ? session\_id : "session\_id");  
 sql += " and film\_id = " + (film\_id.length() > 0 ? film\_id : "film\_id");  
 sql += " and hall\_id = " + (hall\_id.length() > 0 ? hall\_id : "hall\_id");  
 sql += " and time = " + (time.length() > 0 ? ("'" + time + "'") : "time");  
 sql += " and price = " + (price.length() > 0 ? price : "price") + ");\n";  
  
 sql += "DELETE from session where";  
 sql += " session\_id = " + (session\_id.length() > 0 ? session\_id : "session\_id");  
 sql += " and film\_id = " + (film\_id.length() > 0 ? film\_id : "film\_id");  
 sql += " and hall\_id = " + (hall\_id.length() > 0 ? hall\_id : "hall\_id");  
 sql += " and time = " + (time.length() > 0 ? ("'" + time + "'") : "time");  
 sql += " and price = " + (price.length() > 0 ? price : "price") + ";";  
  
 System.*out*.println("sql: " + sql);  
  
 try {  
 Statement statement = connection.createStatement();  
 statement.executeUpdate(sql);  
 } catch (Exception ex) {  
 return ex.getMessage();  
 }  
  
 String answer = "Видалено все, де:\n->";  
 if (session\_id.length() > 0) answer += " session\_id = " + session\_id;  
 if (film\_id.length() > 0) answer += " film\_id = " + film\_id;  
 if (hall\_id.length() > 0) answer += " hall\_id = " + hall\_id;  
 if (time.length() > 0) answer += " time = " + time;  
 if (price.length() > 0) answer += " price = " + price;  
  
 return answer;  
}

Видаляє всі зали з таблиці із заданими параметрами, якщо параметри відсутні то видаляються всі значення з таблиці. Видаляються не лише об’єкти в самій таблиці, а й всі інші об’єкти, що використовували значення, що повинні бути видаленими. Є контроль введених даних на коректність типу. У разі будь-якої помилки повертає повну інформацію про помилку. Результат, що повертає ця функція, виводиться користувачу.

public static String deleteHall(Connection connection, String hall\_id, String name) throws SQLException {  
 if (!hall\_id.matches("\\d+|")) {  
 return "Некоректний ввід даних!";  
 }  
  
 String sql = "DELETE from ticket where session\_id in (select session\_id from session where hall\_id in (select hall\_id from hall where";  
 sql += " hall\_id = " + (hall\_id.length() > 0 ? hall\_id : "hall\_id");  
 sql += " and name like " + (name.length() > 0 ? ("'" + name + "'") : "name") + "));\n";  
  
 sql += "DELETE from session where hall\_id in (select hall\_id from hall where";  
 sql += " hall\_id = " + (hall\_id.length() > 0 ? hall\_id : "hall\_id");  
 sql += " and name like " + (name.length() > 0 ? ("'" + name + "'") : "name") + ");\n";  
  
 sql += "DELETE from chair where hall\_id in (select hall\_id from hall where";  
 sql += " hall\_id = " + (hall\_id.length() > 0 ? hall\_id : "hall\_id");  
 sql += " and name like " + (name.length() > 0 ? ("'" + name + "'") : "name") + ");\n";  
  
 sql += "DELETE from hall where";  
 sql += " hall\_id = " + (hall\_id.length() > 0 ? hall\_id : "hall\_id");  
 sql += " and name like " + (name.length() > 0 ? ("'" + name + "'") : "name") + ";";  
  
 System.*out*.println("sql: " + sql);  
  
 try {  
 Statement statement = connection.createStatement();  
 statement.executeUpdate(sql);  
 } catch (Exception ex) {  
 return ex.getMessage();  
 }  
  
 String answer = "Видалено все, де:\n->";  
 if (hall\_id.length() > 0) answer += " hall\_id = " + hall\_id;  
 if (name.length() > 0) answer += " name = " + name;  
  
 return answer;  
}

Видаляє всі квитки з таблиці із заданими параметрами, якщо параметри відсутні то видаляються всі значення з таблиці. Видаляються не лише об’єкти в самій таблиці, а й всі інші об’єкти, що використовували значення, що повинні бути видаленими. Є контроль введених даних на коректність типу. У разі будь-якої помилки повертає повну інформацію про помилку. Результат, що повертає ця функція, виводиться користувачу.

public static String deleteTicket(Connection connection, String ticket\_id, String session\_id, String chair\_id, String isbought) throws SQLException {  
 if (!ticket\_id.matches("\\d+|") || !session\_id.matches("\\d+|") || !chair\_id.matches("\\d+|") || !isbought.matches("true|false|")) {  
 return "Некоректний ввід даних!";  
 }  
  
 String sql = "DELETE from ticket where";  
 sql += " ticket\_id = " + (ticket\_id.length() > 0 ? ticket\_id : "ticket\_id");  
 sql += " and session\_id = " + (session\_id.length() > 0 ? session\_id : "session\_id");  
 sql += " and chair\_id = " + (chair\_id.length() > 0 ? chair\_id : "chair\_id");  
 sql += " and isbought = " + (isbought.length() > 0 ? isbought : "isbought");  
  
 System.*out*.println("sql: " + sql);  
  
 try {  
 Statement statement = connection.createStatement();  
 statement.executeUpdate(sql);  
 } catch (Exception ex) {  
 return ex.getMessage();  
 }  
  
 String answer = "Видалено все, де:\n->";  
 if (ticket\_id.length() > 0) answer += " ticket\_id = " + ticket\_id;  
 if (session\_id.length() > 0) answer += " session\_id = " + session\_id;  
 if (chair\_id.length() > 0) answer += " chair\_id = " + chair\_id;  
 if (isbought.length() > 0) answer += " isbought = " + isbought;  
  
 return answer;  
}

**Лістинг фрагментів програми з запитом редагування**

Редагує крісло із заданим id, замінює значення на задані параметри, якщо всі поля порожні, повертає помилку. Є контроль введених даних на коректність типу. У разі будь-якої помилки повертає повну інформацію про помилку. Результат, що повертає ця функція, виводиться користувачу.

public static String changeChair(Connection connection, String chair\_id, String hall\_id, String place, String row, String id) throws SQLException {  
 if (!id.matches("\\d+")) {  
 return "Введіть id для заміни!";  
 }  
  
 if (!chair\_id.matches("\\d+|") || !hall\_id.matches("\\d+|") || !place.matches("\\d+|") || !row.matches("\\d+|")) {  
 return "Некоректний ввід даних!";  
 }  
  
 if (!(chair\_id.matches("\\d+") || hall\_id.matches("\\d+") || place.matches("\\d+") || row.matches("\\d+"))) {  
 return "Немає чого переписувати!";  
 }  
  
 String sql = "UPDATE public.chair SET";  
 sql += " chair\_id = " + (chair\_id.length() > 0 ? chair\_id : "chair\_id");  
 sql += ", hall\_id = " + (hall\_id.length() > 0 ? hall\_id : "hall\_id");  
 sql += ", place = " + (place.length() > 0 ? place : "place");  
 sql += ", row = " + (row.length() > 0 ? row : "row");  
 sql += "WHERE chair\_id = " + id;  
  
 System.*out*.println("sql: " + sql);  
  
 try {  
 Statement statement = connection.createStatement();  
 statement.executeUpdate(sql);  
 } catch (Exception ex) {  
 return ex.getMessage();  
 }  
  
 return "Оновлено обьект з id = " + id;  
}

Редагує фільм із заданим id, замінює значення на задані параметри, якщо всі поля порожні, повертає помилку. Є контроль введених даних на коректність типу. У разі будь-якої помилки повертає повну інформацію про помилку. Результат, що повертає ця функція, виводиться користувачу.

public static String changeFilm(Connection connection, String film\_id, String name, String duration, String id) throws SQLException {  
 if (!id.matches("\\d+")) {  
 return "Введіть id для заміни!";  
 }  
  
 if (!duration.matches("")) {  
 SimpleDateFormat sdf = new SimpleDateFormat("hh:mm:ss");  
 try {  
 Date Dateduration = sdf.parse(duration);  
 } catch (Exception ex) {  
 return "Час заданий некроектно";  
 }  
 }  
 if (!film\_id.matches("\\d+|")) {  
 return "Некоректний ввід даних!";  
 }  
  
 if (film\_id.equals("") && name.equals("") && duration.equals("")) {  
 return "Немає чого переписувати!";  
 }  
  
 String sql = "UPDATE public.film SET";  
 sql += " film\_id = " + (film\_id.length() > 0 ? film\_id : "film\_id");  
 sql += ", name = " + (name.length() > 0 ? ("'" + name + "'") : "name");  
 sql += ", duration = " + (duration.length() > 0 ? ("'" + duration + "'") : "duration");  
 sql += " WHERE film\_id = " + id;  
  
 System.*out*.println("sql: " + sql);  
  
 try {  
 Statement = connection.createStatement();  
 statement.executeUpdate(sql);  
 } catch (Exception ex) {  
 return ex.getMessage();  
 }  
  
 return "Оновлено обьект з id = " + id;  
}

Редагує сеанс із заданим id, замінює значення на задані параметри, якщо всі поля порожні, повертає помилку. Є контроль введених даних на коректність типу. У разі будь-якої помилки повертає повну інформацію про помилку. Результат, що повертає ця функція, виводиться користувачу.

public static String changeSession(Connection connection, String session\_id, String film\_id, String hall\_id, String time, String price, String id) throws SQLException {  
 if (!id.matches("\\d+")) {  
 return "Введіть id для заміни!";  
 }  
  
 if (!time.matches("")) {  
 SimpleDateFormat sdf = new SimpleDateFormat("hh:mm:ss");  
 try {  
 Date Dateduration = sdf.parse(time);  
 } catch (Exception ex) {  
 return "Час заданий некроектно";  
 }  
 }  
 if (!session\_id.matches("\\d+|") || !film\_id.matches("\\d+|") || !hall\_id.matches("\\d+|") || !price.matches("[+]?\\d+|")) {  
 return "Некоректний ввід даних!";  
 }  
  
 if (session\_id.equals("") && film\_id.equals("") && hall\_id.equals("") && time.equals("") && price.equals("")) {  
 return "Немає чого переписувати!";  
 }  
  
 String sql = "UPDATE public.session SET";  
 sql += " session\_id = " + (session\_id.length() > 0 ? session\_id : "session\_id");  
 sql += ", film\_id = " + (film\_id.length() > 0 ? film\_id : "film\_id");  
 sql += ", hall\_id = " + (hall\_id.length() > 0 ? hall\_id : "hall\_id");  
 sql += ", time = " + (time.length() > 0 ? ("'" + time + "'") : "time");  
 sql += ", price = " + (price.length() > 0 ? price : "price");  
 sql += " WHERE session\_id = " + id;  
  
 System.*out*.println("sql: " + sql);  
  
 try {  
 Statement statement = connection.createStatement();  
 statement.executeUpdate(sql);  
 } catch (Exception ex) {  
 return ex.getMessage();  
 }  
  
 return "Оновлено обьект з id = " + id;  
}

Редагує зал із заданим id, замінює значення на задані параметри, якщо всі поля порожні, повертає помилку. Є контроль введених даних на коректність типу. У разі будь-якої помилки повертає повну інформацію про помилку. Результат, що повертає ця функція, виводиться користувачу.

public static String changeHall(Connection connection, String hall\_id, String name, String id) throws SQLException {  
 if (!id.matches("\\d+")) {  
 return "Введіть id для заміни!";  
 }  
  
 if (!hall\_id.matches("[+]?\\d+|")) {  
 return "Некоректний ввід даних!";  
 }  
  
 if (hall\_id.equals("") && name.equals("")) {  
 return "Немає чого переписувати!";  
 }  
  
 String sql = "UPDATE public.hall SET";  
 sql += " hall\_id = " + (hall\_id.length() > 0 ? hall\_id : "hall\_id");  
 sql += ", name = " + (name.length() > 0 ? ("'" + name + "'") : "name");  
 sql += " WHERE hall\_id = " + id;  
  
 System.*out*.println("sql: " + sql);  
  
 try {  
 Statement statement = connection.createStatement();  
 statement.executeUpdate(sql);  
 } catch (Exception ex) {  
 return ex.getMessage();  
 }  
  
 return "Оновлено обьект з id = " + id;  
}

Редагує квиток із заданим id, замінює значення на задані параметри, якщо всі поля порожні, повертає помилку. Є контроль введених даних на коректність типу. У разі будь-якої помилки повертає повну інформацію про помилку. Результат, що повертає ця функція, виводиться користувачу.

public static String changeTicket(Connection connection, String ticket\_id, String session\_id, String chair\_id, String isbought, String id) throws SQLException {  
 if (!id.matches("\\d+")) {  
 return "Введіть id для заміни!";  
 }  
  
 if (!ticket\_id.matches("\\d+|") || !session\_id.matches("\\d+|") || !chair\_id.matches("\\d+|") || !isbought.matches("true|false|")) {  
 return "Некоректний ввід даних!";  
 }  
  
 if (ticket\_id.equals("") && session\_id.equals("") && chair\_id.equals("") && isbought.equals("")) {  
 return "Немає чого переписувати!";  
 }  
  
 String sql = "UPDATE public.ticket SET";  
 sql += " ticket\_id = " + (ticket\_id.length() > 0 ? ticket\_id : "ticket\_id");  
 sql += ", session\_id = " + (session\_id.length() > 0 ? session\_id : "session\_id");  
 sql += ", chair\_id = " + (chair\_id.length() > 0 ? chair\_id : "chair\_id");  
 sql += ", isbought = " + (isbought.length() > 0 ? isbought : "isbought");  
 sql += " WHERE ticket\_id = " + id;  
  
 System.*out*.println("sql: " + sql);  
  
 try {  
 Statement statement = connection.createStatement();  
 statement.executeUpdate(sql);  
 } catch (Exception ex) {  
 return ex.getMessage();  
 }  
  
 return "Оновлено обьект з id = " + id;  
}

**Лістинг фрагментів програми з підготовленим запитом пошуку**

*Запит 1*

Пошук фільмів на час після заданого, які коштують менше ніж задана вартість. Додатково повертає час виконання запиту в мілісекундах. Є контроль введених даних на коректність типу. У разі будь-якої помилки повертає повну інформацію про помилку. Результат, що повертає ця функція, виводиться користувачу.

public static String getInterface1(Connection connection, String time, String price) {  
 if (!price.matches("\\d+")) {  
 return "Введіть ціну!";  
 }  
  
 SimpleDateFormat sdf = new SimpleDateFormat("hh:mm:ss");  
 try {  
 Date Dateduration = sdf.parse(time);  
 } catch (Exception ex) {  
 return "Час заданий некроектно";  
 }  
 String sql = "Select f.name, s.time, s.price from public.film f, public.session s where f.film\_id = s.film\_id and";  
 sql += " s.price < (?)";  
 sql += " and s.time > (?)";  
  
 System.*out*.println("sql: " + sql);  
 String resultStr = "Результат пошуку по заданим параметрам:\n";  
  
 long m = System.*currentTimeMillis*();  
  
 try (PreparedStatement statement = connection.prepareStatement(sql)) {  
 statement.setInt(1, *parseInt*(price));  
 statement.setTime(2, Time.*valueOf*(time));  
  
 try {  
 ResultSet resultSet = statement.executeQuery();  
 while (resultSet.next()) {  
 resultStr += "-> Фільм '" + resultSet.getString(1) + "'";  
 resultStr += " на (" + resultSet.getTime(2) + "),";  
 resultStr += " ціна - " + resultSet.getInt(3);  
 resultStr += "\n";  
 }  
 } catch (Exception ex) {  
 return ex.getMessage();  
 }  
 } catch (SQLException exception) {  
 return exception.getMessage();  
 }  
  
 m = System.*currentTimeMillis*() - m;  
  
 return (!resultStr.equals("Результат пошуку по заданим параметрам:\n") ? resultStr + "\nВитрачено " + m + " мілісекунд" : "Нічого не знайдено!");   
}

*Запит 2*

Пошук сеансів на заданий фільм в заданому залі . Додатково повертає час виконання запиту в мілісекундах. Є контроль введених даних на коректність типу. У разі будь-якої помилки повертає повну інформацію про помилку. Результат, що повертає ця функція, виводиться користувачу.

public static String getInterface2(Connection connection, String film\_name, String hall\_name) {  
  
 if (film\_name.equals("") || hall\_name.equals("")) {  
 return "Не всі поля заповнення!";  
 }  
  
 String sql = "Select f.name, s.time, h.name from public.film f, public.session s, public.hall h where f.film\_id = s.film\_id and s.hall\_id = h.hall\_id and";  
 sql += " f.name like ?";  
 sql += " and h.name like ?";  
  
 System.*out*.println("sql: " + sql);  
 String resultStr = "Результат пошуку по заданим параметрам:\n";  
  
 long m = System.*currentTimeMillis*();  
  
 try (PreparedStatement statement = connection.prepareStatement(sql)) {  
 statement.setString(1, film\_name);  
 statement.setString(2, hall\_name);  
  
 try {  
 ResultSet resultSet = statement.executeQuery();  
 while (resultSet.next()) {  
 resultStr += "-> Фільм '" + resultSet.getString(1) + "'";  
 resultStr += " на (" + resultSet.getTime(2) + "),";  
 resultStr += " в залі '" + resultSet.getString(3) + "'";  
 resultStr += "\n";  
 }  
 } catch (Exception ex) {  
 return ex.getMessage();  
 }  
 } catch (SQLException exception) {  
 return exception.getMessage();  
 }  
  
 m = System.*currentTimeMillis*() - m;  
  
 return (!resultStr.equals("Результат пошуку по заданим параметрам:\n") ? resultStr + "\nВитрачено " + m + " мілісекунд" : "Нічого не знайдено!");  
}

*Запит 3*

Пошук кількості місць на заданий фільм на час після заданого. Додатково повертає час виконання запиту в мілісекундах. Є контроль введених даних на коректність типу. У разі будь-якої помилки повертає повну інформацію про помилку. Результат, що повертає ця функція, виводиться користувачу.

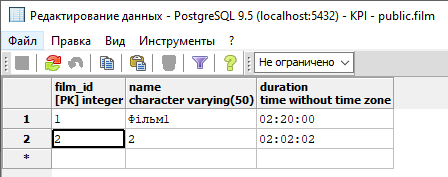
public static String getInterface3(Connection connection, String film\_name, String time) {  
  
 if (film\_name.equals("") || time.equals("")) {  
 return "Не всі поля заповнення!";  
 }  
  
 SimpleDateFormat sdf = new SimpleDateFormat("hh:mm:ss");  
 try {  
 Date Dateduration = sdf.parse(time);  
 } catch (Exception ex) {  
 return "Час заданий некроектно";  
 }  
  
 *//SELECT COUNT(column\_name) FROM table\_name WHERE condition;* String sql = "Select *count*(t.chair\_id), f.name, s.time from public.ticket t, public.session s, public.film f where f.film\_id = s.film\_id and s.session\_id = t.session\_id and";  
 sql += " f.name like ?";  
 sql += " and s.time < ?";  
 sql += " GROUP BY f.name, s.time";  
  
 System.*out*.println("sql: " + sql);  
 String resultStr = "Результат пошуку по заданим параметрам:\n";  
  
 long m = System.*currentTimeMillis*();  
  
 try (PreparedStatement statement = connection.prepareStatement(sql)) {  
 statement.setString(1, film\_name);  
 statement.setTime(2, Time.*valueOf*(time));  
  
 try {  
 ResultSet resultSet = statement.executeQuery();  
 while (resultSet.next()) {  
 resultStr += "-> " + resultSet.getInt(1) + " місць на";  
 resultStr += " фільм '" + resultSet.getString(2) + "'";  
 resultStr += " час (" + resultSet.getTime(3) + ")";  
 resultStr += "\n";  
 }  
 } catch (Exception ex) {  
 return ex.getMessage();  
 }  
 } catch (SQLException exception) {  
 return exception.getMessage();  
 }  
  
 m = System.*currentTimeMillis*() - m;  
  
 return (!resultStr.equals("Результат пошуку по заданим параметрам:\n") ? resultStr + "\nВитрачено " + m + " мілісекунд" : "Нічого не знайдено!");  
}

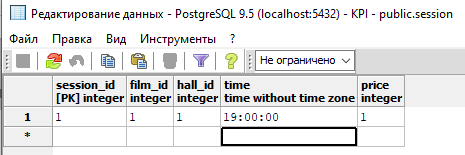
**Лістинги фрагментів програм генерування випадкових даних в таблицях БД**

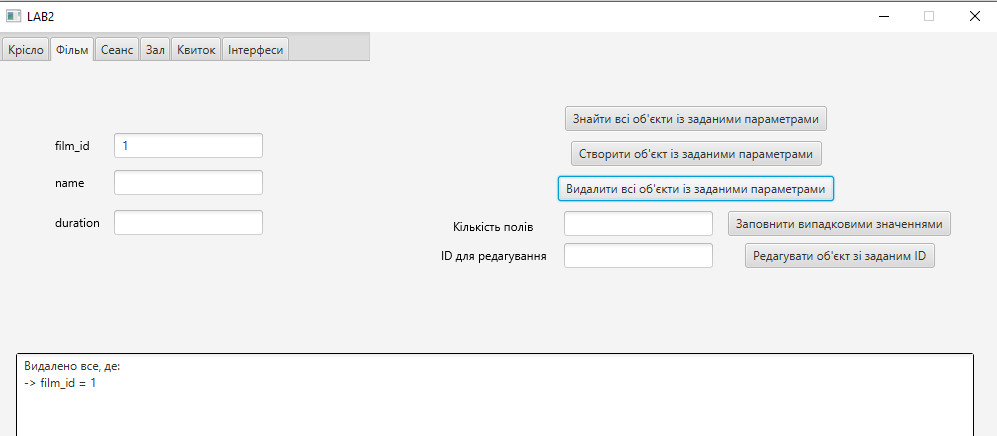
Генерує випадкові дані в залежності від вибраної таблиці. Для кожного з типів даних використовується свій метод генерації. Є контроль зовнішніх та внутрішніх ключів, що робить їх генерацію обмеженою в рамках можливих.

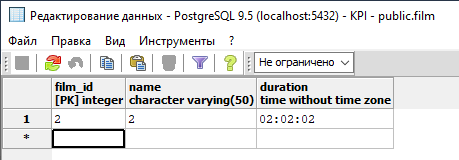
public static String genRandom(Connection connection, String count, int currentTab) throws SQLException {  
 if (count.equals("")) count = "20";  
 if (!count.matches("\\d+")) {  
 return "Некоректний ввід даних!";  
 }  
 String sql = "";  
  
 switch (currentTab) {  
 case 1:  
 sql = "INSERT INTO public.chair (chair\_id, hall\_id, place, row) VALUES ((select *max*(chair\_id) from chair) + 1, (select hall\_id from hall order by *random*() limit 1), *FLOOR*(*RANDOM*() \* 100), *FLOOR*(*RANDOM*() \* 100))";  
 break;  
 case 2:  
 sql = "INSERT INTO public.film (film\_id, name, duration) VALUES ((select *max*(film\_id) from film) + 1, *substr*(*md5*(*random*()::text), 0, 25), *to\_timestamp*(*random*()\*2147483647)::time)";  
 break;  
 case 3:  
 sql = "INSERT INTO public.session (session\_id, film\_id, hall\_id, time, price) VALUES ((select *max*(session\_id) from session) + 1,(select film\_id from film order by *random*() limit 1), (select hall\_id from hall order by *random*() limit 1), *to\_timestamp*(*random*()\*2147483647)::time, *FLOOR*(*RANDOM*() \* 100))";  
 break;  
 case 4:  
 sql = "INSERT INTO public.hall (hall\_id, name) VALUES ((select *max*(hall\_id) from hall) + 1, *substr*(*md5*(*random*()::text), 0, 25))";  
 break;  
 case 5:  
 sql = "INSERT INTO public.ticket (ticket\_id, session\_id, chair\_id, isbought) VALUES ((select *max*(ticket\_id) from ticket) + 1, (select session\_id from session order by *random*() limit 1), (select chair\_id from chair order by *random*() limit 1), *random*() > 0.5)";  
 break;  
 default:  
 break;  
 }  
  
 for (int i = 0; i < *parseInt*(count); i++) {  
 System.*out*.println(i);  
  
 try {  
 Statement statement = connection.createStatement();  
 statement.executeUpdate(sql);  
 } catch (Exception ex) {  
 System.*out*.println(ex.getMessage());  
 }  
 }  
  
 return "Об'єкти створені!";  
}

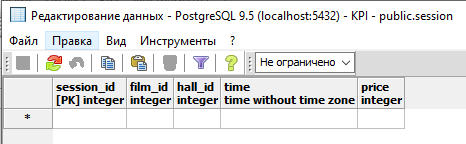
**Скріншоти результатів виконання операції вилучення запису батьківської таблиці та виведення вмісту дочірньої таблиці після цього вилучення**





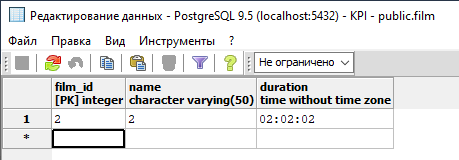


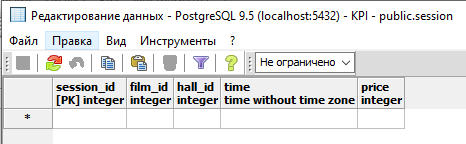


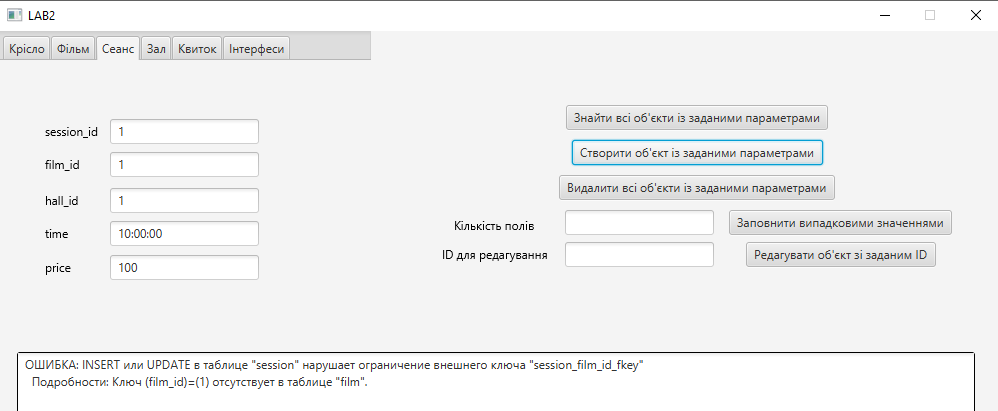


В моїй реалізації будь-яке видалення даних можливе, воно відбувається каскадно. Лістинги реалізації наводились раніше.

**Скріншоти результатів виконання операції вставки запису в дочірню таблицю та виведення повідомлення про її неможливість, якщо в батьківські таблиці нема відповідного запису**







Лістинги реалізації наводились раніше.

**Посилання для навігації по тексту програми**

[Main.java](#Main)

[Controller.java](#Controller)

[Model.java](#Model)

[View.fxml](#View)

**Текст програми**

***Main.java***

package sample;  
  
import javafx.application.Application;  
import javafx.fxml.FXMLLoader;  
import javafx.scene.Parent;  
import javafx.scene.Scene;  
import javafx.stage.Stage;  
  
public class Main extends Application  
{  
 @Override  
 public void start(Stage primaryStage) throws Exception  
 {  
 Parent root = FXMLLoader.*load*(getClass().getResource("View.fxml"));  
 primaryStage.setTitle("LAB2");  
 primaryStage.setScene(new Scene(root, 1000, 800));  
 primaryStage.setResizable(false);  
 primaryStage.show();  
 }  
  
 public static void main(String[] args) {  
 *launch*(args);  
 }  
}

***Controller.java***

package sample;  
  
import java.sql.Connection;  
import java.sql.SQLException;  
  
import javafx.event.ActionEvent;  
import javafx.fxml.FXML;  
import javafx.scene.control.\*;  
  
public class Controller {  
  
 Connection connection;  
  
 @FXML  
 private Tab chair;  
  
 @FXML  
 private TextField chair\_chair\_id;  
  
 @FXML  
 private TextField chair\_hall\_id;  
  
 @FXML  
 private TextField chair\_place;  
  
 @FXML  
 private TextField chair\_row;  
  
 @FXML  
 private Tab film;  
  
 @FXML  
 private TextField film\_film\_id;  
  
 @FXML  
 private TextField film\_name;  
  
 @FXML  
 private TextField film\_duration;  
  
 @FXML  
 private Tab session;  
  
 @FXML  
 private TextField session\_session\_id;  
  
 @FXML  
 private TextField session\_film\_id;  
  
 @FXML  
 private TextField session\_hall\_id;  
  
 @FXML  
 private TextField session\_time;  
  
 @FXML  
 private TextField session\_price;  
  
 @FXML  
 private Tab hall;  
  
 @FXML  
 private TextField hall\_hall\_id;  
  
 @FXML  
 private TextField hall\_name;  
  
 @FXML  
 private Tab ticket;  
  
 @FXML  
 private TextField ticket\_ticket\_id;  
  
 @FXML  
 private TextField ticket\_chair\_id;  
  
 @FXML  
 private TextField ticket\_session\_id;  
  
 @FXML  
 private TextField ticket\_isbought;  
  
 @FXML  
 private TextField time1;  
  
 @FXML  
 private TextField price1;  
  
 @FXML  
 private TextField film\_name2;  
  
 @FXML  
 private TextField hall\_name2;  
  
 @FXML  
 private TextField film\_name3;  
  
 @FXML  
 private TextField time3;  
  
 @FXML  
 private TextField idChange;  
  
 @FXML  
 private TextArea textArea;  
  
 @FXML  
 private TextField randomCount;  
  
 int currentTab() {  
 if (chair.isSelected()) return 1;  
 if (film.isSelected()) return 2;  
 if (session.isSelected()) return 3;  
 if (hall.isSelected()) return 4;  
 if (ticket.isSelected()) return 5;  
 return 0;  
 }  
  
 @FXML  
 *//Поиск по коду товара* public void findALL(ActionEvent actionEvent) throws SQLException {  
 textArea.clear();  
 connection = Model.*getConnection*();  
 switch (currentTab()) {  
 case 1:  
 textArea.setText(textArea.getText() + Model.*findAllChair*(connection, chair\_chair\_id.getText(), chair\_hall\_id.getText(), chair\_place.getText(), chair\_row.getText()));  
 break;  
 case 2:  
 textArea.setText(textArea.getText() + Model.*findAllFilm*(connection, film\_film\_id.getText(), film\_name.getText(), film\_duration.getText()));  
 break;  
 case 3:  
 textArea.setText(textArea.getText() + Model.*findAllSession*(connection, session\_session\_id.getText(), session\_film\_id.getText(), session\_hall\_id.getText(), session\_time.getText(), session\_price.getText()));  
 break;  
 case 4:  
 textArea.setText(textArea.getText() + Model.*findAllHall*(connection, hall\_hall\_id.getText(), hall\_name.getText()));  
 break;  
 case 5:  
 textArea.setText(textArea.getText() + Model.*findAllTicket*(connection, ticket\_ticket\_id.getText(), ticket\_session\_id.getText(), ticket\_chair\_id.getText(), ticket\_isbought.getText()));  
 break;  
 default:  
 break;  
  
 }  
 }  
  
 @FXML  
 public void create(ActionEvent actionEvent) throws SQLException {  
 textArea.clear();  
 connection = Model.*getConnection*();  
 switch (currentTab()) {  
 case 1:  
 textArea.setText(textArea.getText() + Model.*createChair*(connection, chair\_chair\_id.getText(), chair\_hall\_id.getText(), chair\_place.getText(), chair\_row.getText()));  
 break;  
 case 2:  
 textArea.setText(textArea.getText() + Model.*createFilm*(connection, film\_film\_id.getText(), film\_name.getText(), film\_duration.getText()));  
 break;  
 case 3:  
 textArea.setText(textArea.getText() + Model.*createSession*(connection, session\_session\_id.getText(), session\_film\_id.getText(), session\_hall\_id.getText(), session\_time.getText(), session\_price.getText()));  
 break;  
 case 4:  
 textArea.setText(textArea.getText() + Model.*createHall*(connection, hall\_hall\_id.getText(), hall\_name.getText()));  
 break;  
 case 5:  
 textArea.setText(textArea.getText() + Model.*createTicket*(connection, ticket\_ticket\_id.getText(), ticket\_session\_id.getText(), ticket\_chair\_id.getText(), ticket\_isbought.getText()));  
 break;  
 default:  
 break;  
  
 }  
 }  
  
 @FXML  
 public void delete(ActionEvent actionEvent) throws SQLException {  
 textArea.clear();  
 connection = Model.*getConnection*();  
 switch (currentTab()) {  
 case 1:  
 textArea.setText(textArea.getText() + Model.*deleteChair*(connection, chair\_chair\_id.getText(), chair\_hall\_id.getText(), chair\_place.getText(), chair\_row.getText()));  
 break;  
 case 2:  
 textArea.setText(textArea.getText() + Model.*deleteFilm*(connection, film\_film\_id.getText(), film\_name.getText(), film\_duration.getText()));  
 break;  
 case 3:  
 textArea.setText(textArea.getText() + Model.*deleteSession*(connection, session\_session\_id.getText(), session\_film\_id.getText(), session\_hall\_id.getText(), session\_time.getText(), session\_price.getText()));  
 break;  
 case 4:  
 textArea.setText(textArea.getText() + Model.*deleteHall*(connection, hall\_hall\_id.getText(), hall\_name.getText()));  
 break;  
 case 5:  
 textArea.setText(textArea.getText() + Model.*deleteTicket*(connection, ticket\_ticket\_id.getText(), ticket\_session\_id.getText(), ticket\_chair\_id.getText(), ticket\_isbought.getText()));  
 break;  
 default:  
 break;  
 }  
 }  
  
 public void genRandom(ActionEvent actionEvent) throws SQLException {  
 textArea.clear();  
 connection = Model.*getConnection*();  
 textArea.setText(textArea.getText() + Model.*genRandom*(connection, randomCount.getText(), currentTab()));  
 }  
  
 @FXML  
 public void change(ActionEvent actionEvent) throws SQLException {  
 textArea.clear();  
 connection = Model.*getConnection*();  
 switch (currentTab()) {  
 case 1:  
 textArea.setText(textArea.getText() + Model.*changeChair*(connection, chair\_chair\_id.getText(), chair\_hall\_id.getText(), chair\_place.getText(), chair\_row.getText(), idChange.getText()));  
 break;  
 case 2:  
 textArea.setText(textArea.getText() + Model.*changeFilm*(connection, film\_film\_id.getText(), film\_name.getText(), film\_duration.getText(), idChange.getText()));  
 break;  
 case 3:  
 textArea.setText(textArea.getText() + Model.*changeSession*(connection, session\_session\_id.getText(), session\_film\_id.getText(), session\_hall\_id.getText(), session\_time.getText(), session\_price.getText(), idChange.getText()));  
 break;  
 case 4:  
 textArea.setText(textArea.getText() + Model.*changeHall*(connection, hall\_hall\_id.getText(), hall\_name.getText(), idChange.getText()));  
 break;  
 case 5:  
 textArea.setText(textArea.getText() + Model.*changeTicket*(connection, ticket\_ticket\_id.getText(), ticket\_session\_id.getText(), ticket\_chair\_id.getText(), ticket\_isbought.getText(), idChange.getText()));  
 break;  
 default:  
 break;  
 }  
 }  
  
 @FXML  
 public void interface1(ActionEvent actionEvent) throws SQLException {  
 textArea.clear();  
 connection = Model.*getConnection*();  
 textArea.setText(textArea.getText() + Model.*getInterface1*(connection, time1.getText(), price1.getText()));  
 }  
  
 @FXML  
 void interface2(ActionEvent event) throws SQLException {  
 textArea.clear();  
 connection = Model.*getConnection*();  
 textArea.setText(textArea.getText() + Model.*getInterface2*(connection, film\_name2.getText(), hall\_name2.getText()));  
 }  
  
 @FXML  
 void interface3(ActionEvent event) throws SQLException {  
 textArea.clear();  
 connection = Model.*getConnection*();  
 textArea.setText(textArea.getText() + Model.*getInterface3*(connection, film\_name3.getText(), time3.getText()));  
 }  
}

***Model.java***

package sample;  
  
import java.sql.\*;  
import java.text.SimpleDateFormat;  
import java.util.Date;  
  
import static java.lang.Integer.\*;  
  
class Model {  
 static final String *user* = "sysdba";  
 static final String *password* = "masterkey";  
 static final String *url* = "jdbc:postgresql://localhost:5432/KPI";  
  
 *//Функция для подключения к БД* public static Connection getConnection() throws SQLException {  
 return DriverManager.*getConnection*(*url*, *user*, *password*);  
 }  
  
 public static String findAllChair(Connection connection, String chair\_id, String hall\_id, String place, String row) throws SQLException {  
 if (!chair\_id.matches("\\d+|") || !hall\_id.matches("\\d+|") || !place.matches("\\d+|") || !row.matches("\\d+|")) {  
 return "Некоректний ввід даних!";  
 }  
  
 String sql = "SELECT *\** FROM public.chair WHERE";  
 sql += " chair\_id = " + (chair\_id.length() > 0 ? chair\_id : "chair\_id");  
 sql += " and hall\_id = " + (hall\_id.length() > 0 ? hall\_id : "hall\_id");  
 sql += " and place = " + (place.length() > 0 ? place : "place");  
 sql += " and row = " + (row.length() > 0 ? row : "row");  
  
 System.*out*.println("sql: " + sql);  
  
 try (PreparedStatement statement = connection.prepareStatement(sql)) {  
 String resultStr = "Результат пошуку по заданим параметрам:\n";  
 try {  
 ResultSet resultSet = statement.executeQuery();  
 while (resultSet.next()) {  
 resultStr += "-> chair\_id = " + resultSet.getInt(1);  
 resultStr += " hall\_id = " + resultSet.getInt(2);  
 resultStr += " place = " + resultSet.getInt(3);  
 resultStr += " row = " + resultSet.getInt(4);  
 resultStr += "\n";  
 }  
 } catch (Exception ex) {  
 return ex.getMessage();  
 }  
 return (!resultStr.equals("Результат пошуку по заданим параметрам:\n") ? resultStr : "Нічого не знайдено!");  
 }  
 }  
  
 public static String findAllFilm(Connection connection, String film\_id, String name, String duration) throws SQLException {  
  
 if (!duration.matches("")) {  
 SimpleDateFormat sdf = new SimpleDateFormat("hh:mm:ss");  
 try {  
 Date Dateduration = sdf.parse(duration);  
 } catch (Exception ex) {  
 return "Час заданий некроектно";  
 }  
 }  
 if (!film\_id.matches("\\d+|")) {  
 return "Некоректний ввід даних!";  
 }  
  
 String sql = "SELECT *\** FROM public.film WHERE";  
 sql += " film\_id = " + (film\_id.length() > 0 ? film\_id : "film\_id");  
 sql += " and name like " + (name.length() > 0 ? ("'%" + name + "%'") : "name");  
 sql += " and duration = " + (duration.length() > 0 ? ("'" + duration + "'") : "duration");  
  
 System.*out*.println("sql: " + sql);  
  
 try (PreparedStatement statement = connection.prepareStatement(sql)) {  
 String resultStr = "Результат пошуку по заданим параметрам:\n";  
 try {  
 ResultSet resultSet = statement.executeQuery();  
 while (resultSet.next()) {  
 resultStr += "-> film\_id = " + resultSet.getInt(1);  
 resultStr += " name = " + resultSet.getString(2);  
 resultStr += " duration = " + resultSet.getTime(3);  
 resultStr += "\n";  
 }  
 } catch (Exception ex) {  
 return ex.getMessage();  
 }  
 return (!resultStr.equals("Результат пошуку по заданим параметрам:\n") ? resultStr : "Нічого не знайдено!");  
 }  
 }  
  
 public static String findAllSession(Connection connection, String session\_id, String film\_id, String hall\_id, String time, String price) throws SQLException {  
  
 if (!time.matches("")) {  
 SimpleDateFormat sdf = new SimpleDateFormat("hh:mm:ss");  
 try {  
 Date Dateduration = sdf.parse(time);  
 } catch (Exception ex) {  
 return "Час заданий некроектно";  
 }  
 }  
 if (!session\_id.matches("\\d+|") || !film\_id.matches("\\d+|") || !hall\_id.matches("\\d+|") || !price.matches("[+]?\\d+|")) {  
 return "Некоректний ввід даних!";  
 }  
  
 String sql = "SELECT *\** FROM public.session WHERE";  
 sql += " session\_id = " + (session\_id.length() > 0 ? session\_id : "session\_id");  
 sql += " and film\_id = " + (film\_id.length() > 0 ? film\_id : "film\_id");  
 sql += " and hall\_id = " + (hall\_id.length() > 0 ? hall\_id : "hall\_id");  
 sql += " and time = " + (time.length() > 0 ? ("'" + time + "'") : "time");  
 sql += " and price = " + (price.length() > 0 ? price : "price");  
  
 System.*out*.println("sql: " + sql);  
  
 try (PreparedStatement statement = connection.prepareStatement(sql)) {  
 String resultStr = "Результат пошуку по заданим параметрам:\n";  
 try {  
 ResultSet resultSet = statement.executeQuery();  
 while (resultSet.next()) {  
 resultStr += "-> session\_id = " + resultSet.getInt(1);  
 resultStr += " film\_id = " + resultSet.getInt(2);  
 resultStr += " hall\_id = " + resultSet.getInt(3);  
 resultStr += " time = " + resultSet.getTime(4);  
 resultStr += " price = " + resultSet.getInt(5);  
 resultStr += "\n";  
 }  
 } catch (Exception ex) {  
 return ex.getMessage();  
 }  
 return (!resultStr.equals("Результат пошуку по заданим параметрам:\n") ? resultStr : "Нічого не знайдено!");  
 }  
 }  
  
 public static String findAllHall(Connection connection, String hall\_id, String name) throws SQLException {  
 if (!hall\_id.matches("[+]?\\d+|")) {  
 return "Некоректний ввід даних!";  
 }  
  
 String sql = "SELECT *\** FROM public.hall WHERE";  
 sql += " hall\_id = " + (hall\_id.length() > 0 ? hall\_id : "hall\_id");  
 sql += " and name like " + (name.length() > 0 ? ("'%" + name + "%'") : "name");  
  
 System.*out*.println("sql: " + sql);  
  
 try (PreparedStatement statement = connection.prepareStatement(sql)) {  
 String resultStr = "Результат пошуку по заданим параметрам:\n";  
 try {  
 ResultSet resultSet = statement.executeQuery();  
 while (resultSet.next()) {  
 resultStr += "-> hall\_id = " + resultSet.getInt(1);  
 resultStr += " name = " + resultSet.getString(2);  
 resultStr += "\n";  
 }  
 } catch (Exception ex) {  
 return ex.getMessage();  
 }  
 return (!resultStr.equals("Результат пошуку по заданим параметрам:\n") ? resultStr : "Нічого не знайдено!");  
 }  
 }  
  
 public static String findAllTicket(Connection connection, String ticket\_id, String session\_id, String chair\_id, String isbought) throws SQLException {  
  
 if (!ticket\_id.matches("\\d+|") || !session\_id.matches("\\d+|") || !chair\_id.matches("\\d+|") || !isbought.matches("true|false|")) {  
 return "Некоректний ввід даних!";  
 }  
  
 String sql = "SELECT *\** FROM public.ticket WHERE";  
 sql += " ticket\_id = " + (ticket\_id.length() > 0 ? ticket\_id : "ticket\_id");  
 sql += " and session\_id = " + (session\_id.length() > 0 ? session\_id : "session\_id");  
 sql += " and chair\_id = " + (chair\_id.length() > 0 ? chair\_id : "chair\_id");  
 sql += " and isbought = " + (isbought.length() > 0 ? isbought : "isbought");  
  
 System.*out*.println("sql: " + sql);  
  
 try (PreparedStatement statement = connection.prepareStatement(sql)) {  
 String resultStr = "Результат пошуку по заданим параметрам:\n";  
 try {  
 ResultSet resultSet = statement.executeQuery();  
 while (resultSet.next()) {  
 resultStr += "-> ticket\_id = " + resultSet.getInt(1);  
 resultStr += " session\_id = " + resultSet.getInt(2);  
 resultStr += " chair\_id = " + resultSet.getInt(3);  
 resultStr += " isbought = " + resultSet.getBoolean(4);  
 resultStr += "\n";  
 }  
 } catch (Exception ex) {  
 return ex.getMessage();  
 }  
 return (!resultStr.equals("Результат пошуку по заданим параметрам:\n") ? resultStr : "Нічого не знайдено!");  
 }  
 }  
  
 public static String createChair(Connection connection, String chair\_id, String hall\_id, String place, String row) throws SQLException {  
 if (!chair\_id.matches("\\d+") || !hall\_id.matches("\\d+") || !place.matches("\\d+") || !row.matches("\\d+")) {  
 return "Некоректний ввід даних!";  
 }  
  
 String sql = "INSERT INTO public.chair (chair\_id,hall\_id, place, row) VALUES (";  
 sql += chair\_id + ", ";  
 sql += hall\_id + ", ";  
 sql += place + ", ";  
 sql += row + ")";  
  
 System.*out*.println("sql: " + sql);  
 try {  
 Statement statement = connection.createStatement();  
 statement.executeUpdate(sql);  
 } catch (Exception ex) {  
 return ex.getMessage();  
 }  
  
 String answer = "Створено:\n";  
 answer += "-> chair\_id = " + chair\_id;  
 answer += " hall\_id = " + hall\_id;  
 answer += " place = " + place;  
 answer += " row = " + row;  
  
 return answer;  
 }  
  
 public static String createFilm(Connection connection, String film\_id, String name, String duration) throws SQLException {  
 SimpleDateFormat sdf = new SimpleDateFormat("hh:mm:ss");  
 try {  
 Date Dateduration = sdf.parse(duration);  
 } catch (Exception ex) {  
 return "Час заданий некроектно";  
 }  
  
 if (!film\_id.matches("\\d+") || name.equals("")) {  
 return "Некоректний ввід даних!";  
 }  
  
 String sql = "INSERT INTO public.film (film\_id, name, duration) VALUES (";  
 sql += film\_id + ", ";  
 sql += "'" + name + "', ";  
 sql += "'" + duration + "')";  
  
 System.*out*.println("sql: " + sql);  
 try {  
 Statement statement = connection.createStatement();  
 statement.executeUpdate(sql);  
 } catch (Exception ex) {  
 return ex.getMessage();  
 }  
  
 String answer = "Створено:\n";  
 answer += "-> film\_id = " + film\_id;  
 answer += " name = " + name;  
 answer += " duration = " + duration;  
  
 return answer;  
 }  
  
 public static String createSession(Connection connection, String session\_id, String film\_id, String hall\_id, String time, String price) throws SQLException {  
 SimpleDateFormat sdf = new SimpleDateFormat("hh:mm:ss");  
 try {  
 Date Dateduration = sdf.parse(time);  
 } catch (Exception ex) {  
 return "Час заданий некроектно";  
 }  
  
 if (!session\_id.matches("\\d+") || !film\_id.matches("\\d+") || !hall\_id.matches("\\d+") || !price.matches("\\d+")) {  
 return "Некоректний ввід даних!";  
 }  
  
 String sql = "INSERT INTO public.session (session\_id, film\_id, hall\_id, time, price) VALUES (";  
 sql += session\_id + ", ";  
 sql += film\_id + ", ";  
 sql += hall\_id + ", ";  
 sql += "'" + time + "',";  
 sql += price + ")";  
  
 System.*out*.println("sql: " + sql);  
 try {  
 Statement statement = connection.createStatement();  
 statement.executeUpdate(sql);  
 } catch (Exception ex) {  
 return ex.getMessage();  
 }  
  
 String answer = "Створено:\n";  
 answer += "-> session\_id = " + session\_id;  
 answer += " film\_id = " + film\_id;  
 answer += " hall\_id = " + hall\_id;  
 answer += " time = " + time;  
 answer += " price = " + price;  
  
 return answer;  
 }  
  
 public static String createHall(Connection connection, String hall\_id, String name) throws SQLException {  
  
 if (!hall\_id.matches("\\d+") || name.equals("")) {  
 return "Некоректний ввід даних!";  
 }  
  
 String sql = "INSERT INTO public.hall (hall\_id, name) VALUES (";  
 sql += hall\_id + ", ";  
 sql += "'" + name + "')";  
  
 System.*out*.println("sql: " + sql);  
 try {  
 Statement statement = connection.createStatement();  
 statement.executeUpdate(sql);  
 } catch (Exception ex) {  
 return ex.getMessage();  
 }  
  
 String answer = "Створено:\n";  
 answer += "-> hall\_id = " + hall\_id;  
 answer += " name = " + name;  
  
 return answer;  
 }  
  
 public static String createTicket(Connection connection, String ticket\_id, String session\_id, String chair\_id, String isbought) throws SQLException {  
 if (!ticket\_id.matches("\\d+") || !session\_id.matches("\\d+") || !chair\_id.matches("\\d+") || !isbought.matches("true|false|")) {  
 return "Некоректний ввід даних!";  
 }  
  
 String sql = "INSERT INTO public.ticket (ticket\_id, session\_id, chair\_id, isbought) VALUES (";  
 sql += ticket\_id + ", ";  
 sql += session\_id + ", ";  
 sql += chair\_id + ", ";  
 sql += isbought + ")";  
  
 System.*out*.println("sql: " + sql);  
 try {  
 Statement statement = connection.createStatement();  
 statement.executeUpdate(sql);  
 } catch (Exception ex) {  
 return ex.getMessage();  
 }  
  
 String answer = "Створено:\n";  
 answer += "-> ticket\_id = " + ticket\_id;  
 answer += " session\_id = " + session\_id;  
 answer += " chair\_id = " + chair\_id;  
 answer += " isbought = " + isbought;  
  
 return answer;  
 }  
  
 public static String deleteChair(Connection connection, String chair\_id, String hall\_id, String place, String row) throws SQLException {  
 if (!chair\_id.matches("\\d+|") || !hall\_id.matches("\\d+|") || !place.matches("\\d+|") || !row.matches("\\d+|")) {  
 return "Некоректний ввід даних!";  
 }  
  
 String sql = "DELETE from ticket where chair\_id in (select chair\_id from chair where";  
 sql += " chair\_id = " + (chair\_id.length() > 0 ? chair\_id : "chair\_id");  
 sql += " and hall\_id = " + (hall\_id.length() > 0 ? hall\_id : "hall\_id");  
 sql += " and place = " + (place.length() > 0 ? place : "place");  
 sql += " and row = " + (row.length() > 0 ? row : "row") + ");\n";  
  
  
 sql += "DELETE from chair where";  
 sql += " chair\_id = " + (chair\_id.length() > 0 ? chair\_id : "chair\_id");  
 sql += " and hall\_id = " + (hall\_id.length() > 0 ? hall\_id : "hall\_id");  
 sql += " and place = " + (place.length() > 0 ? place : "place");  
 sql += " and row = " + (row.length() > 0 ? row : "row") + ";";  
  
 System.*out*.println("sql: " + sql);  
  
 try {  
 Statement statement = connection.createStatement();  
 statement.executeUpdate(sql);  
 } catch (Exception ex) {  
 return ex.getMessage();  
 }  
  
 String answer = "Видалено все, де:\n->";  
 if (chair\_id.length() > 0) answer += " chair\_id = " + chair\_id;  
 if (hall\_id.length() > 0) answer += " hall\_id = " + hall\_id;  
 if (place.length() > 0) answer += " place = " + place;  
 if (row.length() > 0) answer += " row = " + row;  
  
 return answer;  
 }  
  
 public static String deleteFilm(Connection connection, String film\_id, String name, String duration) throws SQLException {  
 if (!duration.matches("")) {  
 SimpleDateFormat sdf = new SimpleDateFormat("hh:mm:ss");  
 try {  
 Date Dateduration = sdf.parse(duration);  
 } catch (Exception ex) {  
 return "Час заданий некроектно";  
 }  
 }  
 if (!film\_id.matches("\\d+|")) {  
 return "Некоректний ввід даних!";  
 }  
  
 String sql = "DELETE from ticket where session\_id in (select session\_id from session where film\_id in (select film\_id from film where";  
 sql += " film\_id = " + (film\_id.length() > 0 ? film\_id : "film\_id");  
 sql += " and name like " + (name.length() > 0 ? "'" + name + "'" : "name");  
 sql += " and duration = " + (duration.length() > 0 ? "'" + duration + "'" : "duration") + "));\n";  
  
 sql += "DELETE from session where film\_id in (select film\_id from film where";  
 sql += " film\_id = " + (film\_id.length() > 0 ? film\_id : "film\_id");  
 sql += " and name like " + (name.length() > 0 ? "'" + name + "'" : "name");  
 sql += " and duration = " + (duration.length() > 0 ? "'" + duration + "'" : "duration") + ");\n";  
  
 sql += "DELETE from film where";  
 sql += " film\_id = " + (film\_id.length() > 0 ? film\_id : "film\_id");  
 sql += " and name like " + (name.length() > 0 ? "'" + name + "'" : "name");  
 sql += " and duration = " + (duration.length() > 0 ? "'" + duration + "'" : "duration") + ";";  
  
 System.*out*.println("sql: " + sql);  
  
 try {  
 Statement statement = connection.createStatement();  
 statement.executeUpdate(sql);  
 } catch (Exception ex) {  
 return ex.getMessage();  
 }  
  
 String answer = "Видалено все, де:\n->";  
 if (film\_id.length() > 0) answer += " film\_id = " + film\_id;  
 if (name.length() > 0) answer += " name = " + name;  
 if (duration.length() > 0) answer += " duration = " + duration;  
  
 return answer;  
 }  
  
 public static String deleteSession(Connection connection, String session\_id, String film\_id, String hall\_id, String time, String price) throws SQLException {  
 if (!time.matches("")) {  
 SimpleDateFormat sdf = new SimpleDateFormat("hh:mm:ss");  
 try {  
 Date Dateduration = sdf.parse(time);  
 } catch (Exception ex) {  
 return "Час заданий некроектно";  
 }  
 }  
 if (!session\_id.matches("\\d+|") || !film\_id.matches("\\d+|") || !hall\_id.matches("\\d+|") || !price.matches("\\d+|")) {  
 return "Некоректний ввід даних!";  
 }  
  
 String sql = "DELETE from ticket where session\_id in (select session\_id from session where";  
 sql += " session\_id = " + (session\_id.length() > 0 ? session\_id : "session\_id");  
 sql += " and film\_id = " + (film\_id.length() > 0 ? film\_id : "film\_id");  
 sql += " and hall\_id = " + (hall\_id.length() > 0 ? hall\_id : "hall\_id");  
 sql += " and time = " + (time.length() > 0 ? ("'" + time + "'") : "time");  
 sql += " and price = " + (price.length() > 0 ? price : "price") + ");\n";  
  
 sql += "DELETE from session where";  
 sql += " session\_id = " + (session\_id.length() > 0 ? session\_id : "session\_id");  
 sql += " and film\_id = " + (film\_id.length() > 0 ? film\_id : "film\_id");  
 sql += " and hall\_id = " + (hall\_id.length() > 0 ? hall\_id : "hall\_id");  
 sql += " and time = " + (time.length() > 0 ? ("'" + time + "'") : "time");  
 sql += " and price = " + (price.length() > 0 ? price : "price") + ";";  
  
 System.*out*.println("sql: " + sql);  
  
 try {  
 Statement statement = connection.createStatement();  
 statement.executeUpdate(sql);  
 } catch (Exception ex) {  
 return ex.getMessage();  
 }  
  
 String answer = "Видалено все, де:\n->";  
 if (session\_id.length() > 0) answer += " session\_id = " + session\_id;  
 if (film\_id.length() > 0) answer += " film\_id = " + film\_id;  
 if (hall\_id.length() > 0) answer += " hall\_id = " + hall\_id;  
 if (time.length() > 0) answer += " time = " + time;  
 if (price.length() > 0) answer += " price = " + price;  
  
 return answer;  
 }  
  
 public static String deleteHall(Connection connection, String hall\_id, String name) throws SQLException {  
 if (!hall\_id.matches("\\d+|")) {  
 return "Некоректний ввід даних!";  
 }  
  
 String sql = "DELETE from ticket where session\_id in (select session\_id from session where hall\_id in (select hall\_id from hall where";  
 sql += " hall\_id = " + (hall\_id.length() > 0 ? hall\_id : "hall\_id");  
 sql += " and name like " + (name.length() > 0 ? ("'" + name + "'") : "name") + "));\n";  
  
 sql += "DELETE from session where hall\_id in (select hall\_id from hall where";  
 sql += " hall\_id = " + (hall\_id.length() > 0 ? hall\_id : "hall\_id");  
 sql += " and name like " + (name.length() > 0 ? ("'" + name + "'") : "name") + ");\n";  
  
 sql += "DELETE from chair where hall\_id in (select hall\_id from hall where";  
 sql += " hall\_id = " + (hall\_id.length() > 0 ? hall\_id : "hall\_id");  
 sql += " and name like " + (name.length() > 0 ? ("'" + name + "'") : "name") + ");\n";  
  
 sql += "DELETE from hall where";  
 sql += " hall\_id = " + (hall\_id.length() > 0 ? hall\_id : "hall\_id");  
 sql += " and name like " + (name.length() > 0 ? ("'" + name + "'") : "name") + ";";  
  
 System.*out*.println("sql: " + sql);  
  
 try {  
 Statement statement = connection.createStatement();  
 statement.executeUpdate(sql);  
 } catch (Exception ex) {  
 return ex.getMessage();  
 }  
  
 String answer = "Видалено все, де:\n->";  
 if (hall\_id.length() > 0) answer += " hall\_id = " + hall\_id;  
 if (name.length() > 0) answer += " name = " + name;  
  
 return answer;  
 }  
  
 public static String deleteTicket(Connection connection, String ticket\_id, String session\_id, String chair\_id, String isbought) throws SQLException {  
 if (!ticket\_id.matches("\\d+|") || !session\_id.matches("\\d+|") || !chair\_id.matches("\\d+|") || !isbought.matches("true|false|")) {  
 return "Некоректний ввід даних!";  
 }  
  
 String sql = "DELETE from ticket where";  
 sql += " ticket\_id = " + (ticket\_id.length() > 0 ? ticket\_id : "ticket\_id");  
 sql += " and session\_id = " + (session\_id.length() > 0 ? session\_id : "session\_id");  
 sql += " and chair\_id = " + (chair\_id.length() > 0 ? chair\_id : "chair\_id");  
 sql += " and isbought = " + (isbought.length() > 0 ? isbought : "isbought");  
  
 System.*out*.println("sql: " + sql);  
  
 try {  
 Statement statement = connection.createStatement();  
 statement.executeUpdate(sql);  
 } catch (Exception ex) {  
 return ex.getMessage();  
 }  
  
 String answer = "Видалено все, де:\n->";  
 if (ticket\_id.length() > 0) answer += " ticket\_id = " + ticket\_id;  
 if (session\_id.length() > 0) answer += " session\_id = " + session\_id;  
 if (chair\_id.length() > 0) answer += " chair\_id = " + chair\_id;  
 if (isbought.length() > 0) answer += " isbought = " + isbought;  
  
 return answer;  
 }  
  
 public static String genRandom(Connection connection, String count, int currentTab) throws SQLException {  
 if (count.equals("")) count = "20";  
 if (!count.matches("\\d+")) {  
 return "Некоректний ввід даних!";  
 }  
 String sql = "";  
  
 switch (currentTab) {  
 case 1:  
 sql = "INSERT INTO public.chair (chair\_id, hall\_id, place, row) VALUES ((select *max*(chair\_id) from chair) + 1, (select hall\_id from hall order by *random*() limit 1), *FLOOR*(*RANDOM*() \* 100), *FLOOR*(*RANDOM*() \* 100))";  
 break;  
 case 2:  
 sql = "INSERT INTO public.film (film\_id, name, duration) VALUES ((select *max*(film\_id) from film) + 1, *substr*(*md5*(*random*()::text), 0, 25), *to\_timestamp*(*random*()\*2147483647)::time)";  
 break;  
 case 3:  
 sql = "INSERT INTO public.session (session\_id, film\_id, hall\_id, time, price) VALUES ((select *max*(session\_id) from session) + 1,(select film\_id from film order by *random*() limit 1), (select hall\_id from hall order by *random*() limit 1), *to\_timestamp*(*random*()\*2147483647)::time, *FLOOR*(*RANDOM*() \* 100))";  
 break;  
 case 4:  
 sql = "INSERT INTO public.hall (hall\_id, name) VALUES ((select *max*(hall\_id) from hall) + 1, *substr*(*md5*(*random*()::text), 0, 25))";  
 break;  
 case 5:  
 sql = "INSERT INTO public.ticket (ticket\_id, session\_id, chair\_id, isbought) VALUES ((select *max*(ticket\_id) from ticket) + 1, (select session\_id from session order by *random*() limit 1), (select chair\_id from chair order by *random*() limit 1), *random*() > 0.5)";  
 break;  
 default:  
 break;  
 }  
  
 for (int i = 0; i < *parseInt*(count); i++) {  
 System.*out*.println(i);  
  
 try {  
 Statement statement = connection.createStatement();  
 statement.executeUpdate(sql);  
 } catch (Exception ex) {  
 System.*out*.println(ex.getMessage());  
 }  
 }  
  
 return "Об'єкти створені!";  
 }  
  
 public static String changeChair(Connection connection, String chair\_id, String hall\_id, String place, String row, String id) throws SQLException {  
 if (!id.matches("\\d+")) {  
 return "Введіть id для заміни!";  
 }  
  
 if (!chair\_id.matches("\\d+|") || !hall\_id.matches("\\d+|") || !place.matches("\\d+|") || !row.matches("\\d+|")) {  
 return "Некоректний ввід даних!";  
 }  
  
 if (!(chair\_id.matches("\\d+") || hall\_id.matches("\\d+") || place.matches("\\d+") || row.matches("\\d+"))) {  
 return "Немає чого переписувати!";  
 }  
  
 String sql = "UPDATE public.chair SET";  
 sql += " chair\_id = " + (chair\_id.length() > 0 ? chair\_id : "chair\_id");  
 sql += ", hall\_id = " + (hall\_id.length() > 0 ? hall\_id : "hall\_id");  
 sql += ", place = " + (place.length() > 0 ? place : "place");  
 sql += ", row = " + (row.length() > 0 ? row : "row");  
 sql += "WHERE chair\_id = " + id;  
  
 System.*out*.println("sql: " + sql);  
  
 try {  
 Statement statement = connection.createStatement();  
 statement.executeUpdate(sql);  
 } catch (Exception ex) {  
 return ex.getMessage();  
 }  
  
 return "Оновлено обьект з id = " + id;  
 }  
  
 public static String changeFilm(Connection connection, String film\_id, String name, String duration, String id) throws SQLException {  
 if (!id.matches("\\d+")) {  
 return "Введіть id для заміни!";  
 }  
  
 if (!duration.matches("")) {  
 SimpleDateFormat sdf = new SimpleDateFormat("hh:mm:ss");  
 try {  
 Date Dateduration = sdf.parse(duration);  
 } catch (Exception ex) {  
 return "Час заданий некроектно";  
 }  
 }  
 if (!film\_id.matches("\\d+|")) {  
 return "Некоректний ввід даних!";  
 }  
  
 if (film\_id.equals("") && name.equals("") && duration.equals("")) {  
 return "Немає чого переписувати!";  
 }  
  
 String sql = "UPDATE public.film SET";  
 sql += " film\_id = " + (film\_id.length() > 0 ? film\_id : "film\_id");  
 sql += ", name = " + (name.length() > 0 ? ("'" + name + "'") : "name");  
 sql += ", duration = " + (duration.length() > 0 ? ("'" + duration + "'") : "duration");  
 sql += " WHERE film\_id = " + id;  
  
 System.*out*.println("sql: " + sql);  
  
 try {  
 Statement statement = connection.createStatement();  
 statement.executeUpdate(sql);  
 } catch (Exception ex) {  
 return ex.getMessage();  
 }  
  
 return "Оновлено обьект з id = " + id;  
 }  
  
 public static String changeSession(Connection connection, String session\_id, String film\_id, String hall\_id, String time, String price, String id) throws SQLException {  
 if (!id.matches("\\d+")) {  
 return "Введіть id для заміни!";  
 }  
  
 if (!time.matches("")) {  
 SimpleDateFormat sdf = new SimpleDateFormat("hh:mm:ss");  
 try {  
 Date Dateduration = sdf.parse(time);  
 } catch (Exception ex) {  
 return "Час заданий некроектно";  
 }  
 }  
 if (!session\_id.matches("\\d+|") || !film\_id.matches("\\d+|") || !hall\_id.matches("\\d+|") || !price.matches("[+]?\\d+|")) {  
 return "Некоректний ввід даних!";  
 }  
  
 if (session\_id.equals("") && film\_id.equals("") && hall\_id.equals("") && time.equals("") && price.equals("")) {  
 return "Немає чого переписувати!";  
 }  
  
 String sql = "UPDATE public.session SET";  
 sql += " session\_id = " + (session\_id.length() > 0 ? session\_id : "session\_id");  
 sql += ", film\_id = " + (film\_id.length() > 0 ? film\_id : "film\_id");  
 sql += ", hall\_id = " + (hall\_id.length() > 0 ? hall\_id : "hall\_id");  
 sql += ", time = " + (time.length() > 0 ? ("'" + time + "'") : "time");  
 sql += ", price = " + (price.length() > 0 ? price : "price");  
 sql += " WHERE session\_id = " + id;  
  
 System.*out*.println("sql: " + sql);  
  
 try {  
 Statement statement = connection.createStatement();  
 statement.executeUpdate(sql);  
 } catch (Exception ex) {  
 return ex.getMessage();  
 }  
  
 return "Оновлено обьект з id = " + id;  
 }  
  
 public static String changeHall(Connection connection, String hall\_id, String name, String id) throws SQLException {  
 if (!id.matches("\\d+")) {  
 return "Введіть id для заміни!";  
 }  
  
 if (!hall\_id.matches("[+]?\\d+|")) {  
 return "Некоректний ввід даних!";  
 }  
  
 if (hall\_id.equals("") && name.equals("")) {  
 return "Немає чого переписувати!";  
 }  
  
 String sql = "UPDATE public.hall SET";  
 sql += " hall\_id = " + (hall\_id.length() > 0 ? hall\_id : "hall\_id");  
 sql += ", name = " + (name.length() > 0 ? ("'" + name + "'") : "name");  
 sql += " WHERE hall\_id = " + id;  
  
 System.*out*.println("sql: " + sql);  
  
 try {  
 Statement statement = connection.createStatement();  
 statement.executeUpdate(sql);  
 } catch (Exception ex) {  
 return ex.getMessage();  
 }  
  
 return "Оновлено обьект з id = " + id;  
 }  
  
 public static String changeTicket(Connection connection, String ticket\_id, String session\_id, String chair\_id, String isbought, String id) throws SQLException {  
 if (!id.matches("\\d+")) {  
 return "Введіть id для заміни!";  
 }  
  
 if (!ticket\_id.matches("\\d+|") || !session\_id.matches("\\d+|") || !chair\_id.matches("\\d+|") || !isbought.matches("true|false|")) {  
 return "Некоректний ввід даних!";  
 }  
  
 if (ticket\_id.equals("") && session\_id.equals("") && chair\_id.equals("") && isbought.equals("")) {  
 return "Немає чого переписувати!";  
 }  
  
 String sql = "UPDATE public.ticket SET";  
 sql += " ticket\_id = " + (ticket\_id.length() > 0 ? ticket\_id : "ticket\_id");  
 sql += ", session\_id = " + (session\_id.length() > 0 ? session\_id : "session\_id");  
 sql += ", chair\_id = " + (chair\_id.length() > 0 ? chair\_id : "chair\_id");  
 sql += ", isbought = " + (isbought.length() > 0 ? isbought : "isbought");  
 sql += " WHERE ticket\_id = " + id;  
  
 System.*out*.println("sql: " + sql);  
  
 try {  
 Statement statement = connection.createStatement();  
 statement.executeUpdate(sql);  
 } catch (Exception ex) {  
 return ex.getMessage();  
 }  
  
 return "Оновлено обьект з id = " + id;  
 }  
  
 public static String getInterface1(Connection connection, String time, String price) {  
 if (!price.matches("\\d+")) {  
 return "Введіть ціну!";  
 }  
  
 SimpleDateFormat sdf = new SimpleDateFormat("hh:mm:ss");  
 try {  
 Date Dateduration = sdf.parse(time);  
 } catch (Exception ex) {  
 return "Час заданий некроектно";  
 }  
 String sql = "Select f.name, s.time, s.price from public.film f, public.session s where f.film\_id = s.film\_id and";  
 sql += " s.price < (?)";  
 sql += " and s.time > (?)";  
  
 System.*out*.println("sql: " + sql);  
 String resultStr = "Результат пошуку по заданим параметрам:\n";  
  
 long m = System.*currentTimeMillis*();  
  
 try (PreparedStatement statement = connection.prepareStatement(sql)) {  
 statement.setInt(1, *parseInt*(price));  
 statement.setTime(2, Time.*valueOf*(time));  
  
 try {  
 ResultSet resultSet = statement.executeQuery();  
 while (resultSet.next()) {  
 resultStr += "-> Фільм '" + resultSet.getString(1) + "'";  
 resultStr += " на (" + resultSet.getTime(2) + "),";  
 resultStr += " ціна - " + resultSet.getInt(3);  
 resultStr += "\n";  
 }  
 } catch (Exception ex) {  
 return ex.getMessage();  
 }  
 } catch (SQLException exception) {  
 return exception.getMessage();  
 }  
  
 m = System.*currentTimeMillis*() - m;  
  
 return (!resultStr.equals("Результат пошуку по заданим параметрам:\n") ? resultStr + "\nВитрачено " + m + " мілісекунд" : "Нічого не знайдено!");  
 }  
  
 public static String getInterface2(Connection connection, String film\_name, String hall\_name) {  
  
 if (film\_name.equals("") || hall\_name.equals("")) {  
 return "Не всі поля заповнення!";  
 }  
  
 String sql = "Select f.name, s.time, h.name from public.film f, public.session s, public.hall h where f.film\_id = s.film\_id and s.hall\_id = h.hall\_id and";  
 sql += " f.name like ?";  
 sql += " and h.name like ?";  
  
 System.*out*.println("sql: " + sql);  
 String resultStr = "Результат пошуку по заданим параметрам:\n";  
  
 long m = System.*currentTimeMillis*();  
  
 try (PreparedStatement statement = connection.prepareStatement(sql)) {  
 statement.setString(1, film\_name);  
 statement.setString(2, hall\_name);  
  
 try {  
 ResultSet resultSet = statement.executeQuery();  
 while (resultSet.next()) {  
 resultStr += "-> Фільм '" + resultSet.getString(1) + "'";  
 resultStr += " на (" + resultSet.getTime(2) + "),";  
 resultStr += " в залі '" + resultSet.getString(3) + "'";  
 resultStr += "\n";  
 }  
 } catch (Exception ex) {  
 return ex.getMessage();  
 }  
 } catch (SQLException exception) {  
 return exception.getMessage();  
 }  
  
 m = System.*currentTimeMillis*() - m;  
  
 return (!resultStr.equals("Результат пошуку по заданим параметрам:\n") ? resultStr + "\nВитрачено " + m + " мілісекунд" : "Нічого не знайдено!");  
 }  
  
 public static String getInterface3(Connection connection, String film\_name, String time) {  
  
 if (film\_name.equals("") || time.equals("")) {  
 return "Не всі поля заповнення!";  
 }  
  
 SimpleDateFormat sdf = new SimpleDateFormat("hh:mm:ss");  
 try {  
 Date Dateduration = sdf.parse(time);  
 } catch (Exception ex) {  
 return "Час заданий некроектно";  
 }  
  
 *//SELECT COUNT(column\_name) FROM table\_name WHERE condition;* String sql = "Select *count*(t.chair\_id), f.name, s.time from public.ticket t, public.session s, public.film f where f.film\_id = s.film\_id and s.session\_id = t.session\_id and";  
 sql += " f.name like ?";  
 sql += " and s.time < ?";  
 sql += " GROUP BY f.name, s.time";  
  
 System.*out*.println("sql: " + sql);  
 String resultStr = "Результат пошуку по заданим параметрам:\n";  
  
 long m = System.*currentTimeMillis*();  
  
 try (PreparedStatement statement = connection.prepareStatement(sql)) {  
 statement.setString(1, film\_name);  
 statement.setTime(2, Time.*valueOf*(time));  
  
 try {  
 ResultSet resultSet = statement.executeQuery();  
 while (resultSet.next()) {  
 resultStr += "-> " + resultSet.getInt(1) + " місць на";  
 resultStr += " фільм '" + resultSet.getString(2) + "'";  
 resultStr += " час (" + resultSet.getTime(3) + ")";  
 resultStr += "\n";  
 }  
 } catch (Exception ex) {  
 return ex.getMessage();  
 }  
 } catch (SQLException exception) {  
 return exception.getMessage();  
 }  
  
 m = System.*currentTimeMillis*() - m;  
  
 return (!resultStr.equals("Результат пошуку по заданим параметрам:\n") ? resultStr + "\nВитрачено " + m + " мілісекунд" : "Нічого не знайдено!");  
 }  
}

***View.fxml***

*<?*xml version="1.0" encoding="UTF-8"*?>  
  
<?*import javafx.scene.control.Button*?>  
<?*import javafx.scene.control.Tab*?>  
<?*import javafx.scene.control.TabPane*?>  
<?*import javafx.scene.control.TextArea*?>  
<?*import javafx.scene.control.TextField*?>  
<?*import javafx.scene.layout.AnchorPane*?>  
<?*import javafx.scene.layout.BorderPane*?>  
<?*import javafx.scene.text.Text*?>*<AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="800.0" prefWidth="1000.0" xmlns="http://javafx.com/javafx/8.0.171" xmlns:fx="http://javafx.com/fxml/1" fx:controller="sample.Controller">  
 <children>  
 <TabPane layoutX="-1.0" prefHeight="310.0" prefWidth="373.0" tabClosingPolicy="UNAVAILABLE">  
 <tabs>  
 <Tab fx:id="chair" text="Крісло">  
 <content>  
 <AnchorPane minHeight="0.0" minWidth="0.0" prefHeight="257.0" prefWidth="339.0">  
 <children>  
 <TextField fx:id="chair\_chair\_id" layoutX="120.0" layoutY="55.0" />  
 <Text layoutX="43.0" layoutY="72.0" strokeType="OUTSIDE" strokeWidth="0.0" text="chair\_id" />  
 <TextField fx:id="chair\_hall\_id" layoutX="120.0" layoutY="90.0" />  
 <Text layoutX="43.0" layoutY="107.0" strokeType="OUTSIDE" strokeWidth="0.0" text="hall\_id" />  
 <TextField fx:id="chair\_place" layoutX="120.0" layoutY="130.0" />  
 <Text layoutX="43.0" layoutY="147.0" strokeType="OUTSIDE" strokeWidth="0.0" text="place" />  
 <TextField fx:id="chair\_row" layoutX="120.0" layoutY="170.0" />  
 <Text layoutX="43.0" layoutY="187.0" strokeType="OUTSIDE" strokeWidth="0.0" text="row" />  
 </children>  
 </AnchorPane>  
 </content>  
 </Tab>  
 <Tab fx:id="film" text="Фільм">  
 <content>  
 <AnchorPane minHeight="0.0" minWidth="0.0" prefHeight="719.0" prefWidth="1080.0">  
 <children>  
 <TextField fx:id="film\_film\_id" layoutX="117.0" layoutY="72.0" />  
 <Text layoutX="58.0" layoutY="89.0" strokeType="OUTSIDE" strokeWidth="0.0" text="film\_id" />  
 <TextField fx:id="film\_name" layoutX="117.0" layoutY="109.0" />  
 <Text layoutX="58.0" layoutY="126.0" strokeType="OUTSIDE" strokeWidth="0.0" text="name" />  
 <TextField fx:id="film\_duration" layoutX="117.0" layoutY="149.0" />  
 <Text layoutX="58.0" layoutY="166.0" strokeType="OUTSIDE" strokeWidth="0.0" text="duration" />  
 </children>  
 </AnchorPane>  
 </content>  
 </Tab>  
 <Tab fx:id="session" text="Сеанс">  
 <content>  
 <AnchorPane minHeight="0.0" minWidth="0.0" prefHeight="180.0" prefWidth="200.0">  
 <children>  
 <TextField fx:id="session\_session\_id" layoutX="112.0" layoutY="59.0" />  
 <Text layoutX="47.0" layoutY="76.0" strokeType="OUTSIDE" strokeWidth="0.0" text="session\_id" />  
 <TextField fx:id="session\_film\_id" layoutX="112.0" layoutY="92.0" />  
 <Text layoutX="47.0" layoutY="109.0" strokeType="OUTSIDE" strokeWidth="0.0" text="film\_id" />  
 <TextField fx:id="session\_hall\_id" layoutX="112.0" layoutY="128.0" />  
 <Text layoutX="47.0" layoutY="145.0" strokeType="OUTSIDE" strokeWidth="0.0" text="hall\_id" />  
 <TextField fx:id="session\_time" layoutX="112.0" layoutY="161.0" />  
 <Text layoutX="47.0" layoutY="178.0" strokeType="OUTSIDE" strokeWidth="0.0" text="time" />  
 <TextField fx:id="session\_price" layoutX="112.0" layoutY="195.0" />  
 <Text layoutX="47.0" layoutY="212.0" strokeType="OUTSIDE" strokeWidth="0.0" text="price" />  
 </children>  
 </AnchorPane>  
 </content>  
 </Tab>  
 <Tab fx:id="hall" text="Зал">  
 <content>  
 <AnchorPane minHeight="0.0" minWidth="0.0" prefHeight="180.0" prefWidth="200.0">  
 <children>  
 <Text layoutX="48.0" layoutY="75.0" strokeType="OUTSIDE" strokeWidth="0.0" text="hall\_id" />  
 <TextField fx:id="hall\_hall\_id" layoutX="106.0" layoutY="58.0" />  
 <Text layoutX="48.0" layoutY="110.0" strokeType="OUTSIDE" strokeWidth="0.0" text="name" />  
 <TextField fx:id="hall\_name" layoutX="106.0" layoutY="93.0" />  
 </children>  
 </AnchorPane>  
 </content>  
 </Tab>  
 <Tab fx:id="ticket" text="Квиток">  
 <content>  
 <AnchorPane minHeight="0.0" minWidth="0.0" prefHeight="180.0" prefWidth="200.0">  
 <children>  
 <Text layoutX="49.0" layoutY="73.0" strokeType="OUTSIDE" strokeWidth="0.0" text="ticket\_id" />  
 <TextField fx:id="ticket\_ticket\_id" layoutX="107.0" layoutY="56.0" />  
 <Text layoutX="49.0" layoutY="106.0" strokeType="OUTSIDE" strokeWidth="0.0" text="chair\_id" />  
 <TextField fx:id="ticket\_chair\_id" layoutX="107.0" layoutY="89.0" />  
 <Text layoutX="49.0" layoutY="141.0" strokeType="OUTSIDE" strokeWidth="0.0" text="session\_id" />  
 <TextField fx:id="ticket\_session\_id" layoutX="107.0" layoutY="124.0" />  
 <Text layoutX="49.0" layoutY="176.0" strokeType="OUTSIDE" strokeWidth="0.0" text="isbought" />  
 <TextField fx:id="ticket\_isbought" layoutX="107.0" layoutY="159.0" />  
 </children>  
 </AnchorPane>  
 </content>  
 </Tab>  
 <Tab text="Інтерфеси">  
 <content>  
 <AnchorPane minHeight="0.0" minWidth="0.0" prefHeight="281.0" prefWidth="373.0">  
 <children>  
 <Text layoutX="27.0" layoutY="42.0" strokeType="OUTSIDE" strokeWidth="0.0" text="Всі фільми на час після" />  
 <Text layoutX="229.0" layoutY="42.0" strokeType="OUTSIDE" strokeWidth="0.0" text="і ціна до" />  
 <Text layoutX="329.0" layoutY="42.0" strokeType="OUTSIDE" strokeWidth="0.0" text="грн" />  
 <TextField fx:id="time1" layoutX="158.0" layoutY="25.0" prefHeight="25.0" prefWidth="64.0" />  
 <TextField fx:id="price1" layoutX="280.0" layoutY="25.0" prefHeight="25.0" prefWidth="39.0" />  
 <Text layoutX="56.0" layoutY="141.0" strokeType="OUTSIDE" strokeWidth="0.0" text="Сеанс на фільм" />  
 <TextField fx:id="film\_name2" layoutX="151.0" layoutY="124.0" prefHeight="25.0" prefWidth="64.0" />  
 <Text layoutX="219.0" layoutY="141.0" strokeType="OUTSIDE" strokeWidth="0.0" text="в" />  
 <TextField fx:id="hall\_name2" layoutX="238.0" layoutY="124.0" prefHeight="25.0" prefWidth="69.0" />  
 <Text layoutX="311.0" layoutY="141.0" strokeType="OUTSIDE" strokeWidth="0.0" text="залі" />  
 <Text layoutX="12.0" layoutY="234.0" strokeType="OUTSIDE" strokeWidth="0.0" text="Кількість місць на фільм" />  
 <TextField fx:id="film\_name3" layoutX="154.0" layoutY="217.0" prefHeight="25.0" prefWidth="64.0" />  
 <Text layoutX="238.0" layoutY="233.0" strokeType="OUTSIDE" strokeWidth="0.0" text="на час до" />  
 <TextField fx:id="time3" layoutX="303.0" layoutY="217.0" prefHeight="25.0" prefWidth="64.0" />  
 <Button layoutX="157.0" layoutY="60.0" mnemonicParsing="false" onAction="#interface1" text="Пошук" />  
 <Button layoutX="161.0" layoutY="154.0" mnemonicParsing="false" onAction="#interface2" text="Пошук" />  
 <Button layoutX="161.0" layoutY="255.0" mnemonicParsing="false" onAction="#interface3" text="Пошук" />  
 </children></AnchorPane>  
 </content>  
 </Tab>  
 </tabs>  
 </TabPane>  
 <Button layoutX="560.0" layoutY="144.0" mnemonicParsing="false" onAction="#delete" text="Видалити всі об'єкти із заданими параметрами" />  
 <Button layoutX="567.0" layoutY="74.0" mnemonicParsing="false" onAction="#findALL" text="Знайти всі об'єкти із заданими параметрами" />  
 <Button layoutX="573.0" layoutY="109.0" mnemonicParsing="false" onAction="#create" text="Створити об'єкт із заданими параметрами" />  
 <Button layoutX="730.0" layoutY="179.0" mnemonicParsing="false" onAction="#genRandom" text="Заповнити випадковими значеннями" />  
 <TextField fx:id="idChange" layoutX="566.0" layoutY="211.0" />  
 <Text layoutX="443.0" layoutY="228.0" strokeType="OUTSIDE" strokeWidth="0.0" text="ID для редагування" />  
 <Button layoutX="747.0" layoutY="211.0" mnemonicParsing="false" onAction="#change" text="Редагувати об'єкт зі заданим ID" />  
 <BorderPane layoutX="18.0" layoutY="321.0" prefHeight="459.0" prefWidth="958.0">  
 <top>  
 <TextArea fx:id="textArea" cache="true" focusTraversable="false" prefHeight="458.0" prefWidth="958.0" style="-fx-background-color: black;" wrapText="true" BorderPane.alignment="CENTER" />  
 </top>  
 </BorderPane>  
 <TextField fx:id="randomCount" layoutX="566.0" layoutY="179.0" />  
 <Text layoutX="455.0" layoutY="199.0" strokeType="OUTSIDE" strokeWidth="0.0" text="Кількість полів" />  
 </children>  
</AnchorPane>