

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

Бітлян А. В.

Перевірив:

Київ – 2021

**Форматування. По всьому змістовному тексту вирівнювання по ширині**

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:

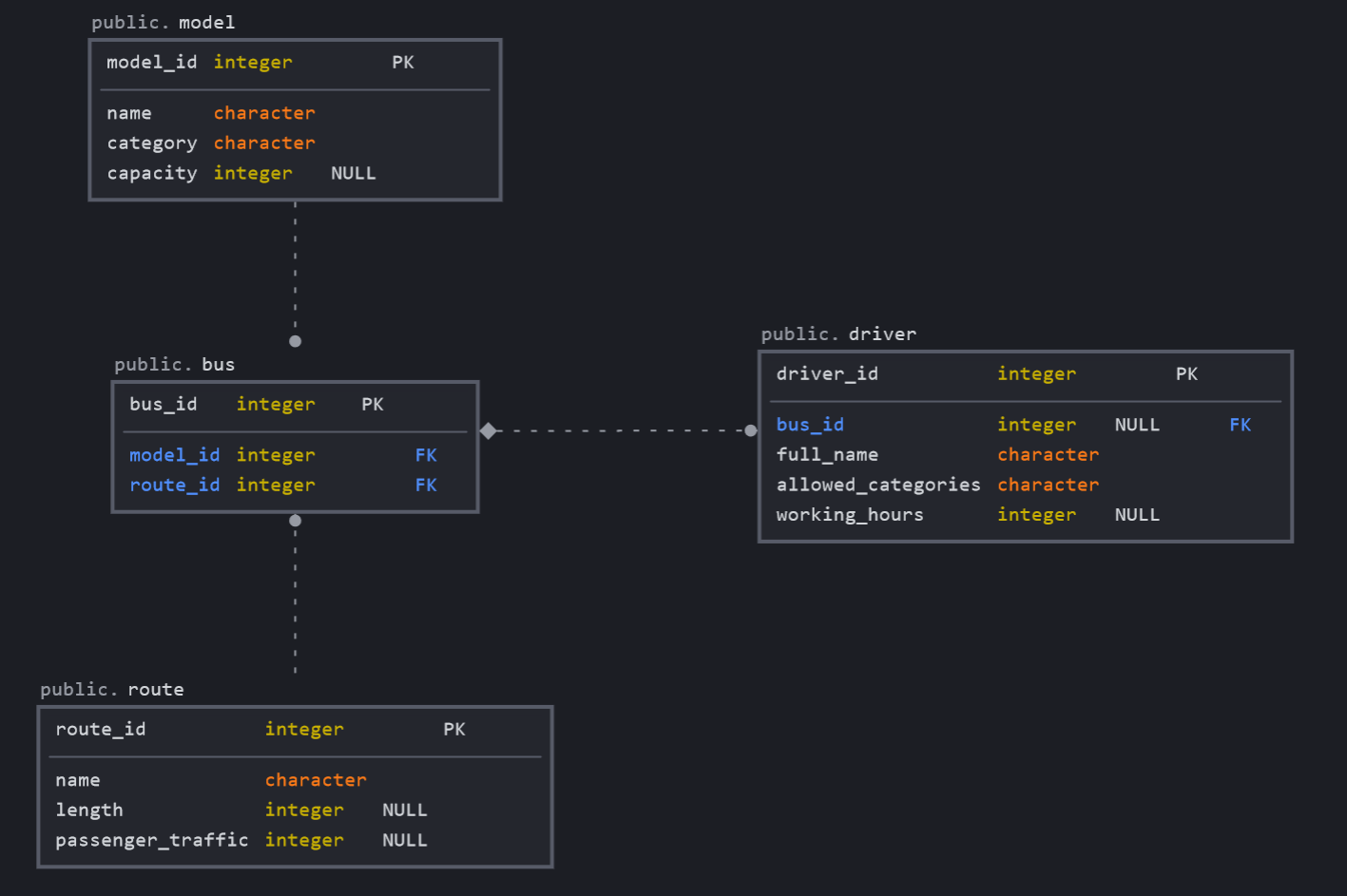


Рис. 1 – Логічна модель бази даних

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:

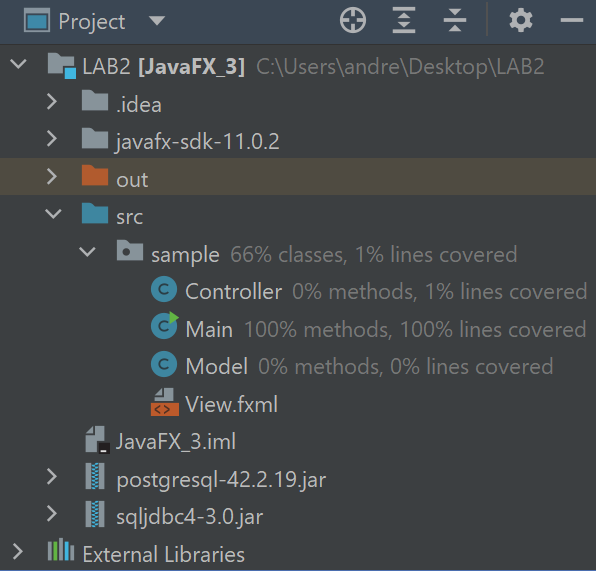


Рис. 2 – Структура програми

1. Main.java – точка входу в програму, запускає початковий інтерфейс.
2. **M**odel.java - виконує операції з БД та безпосередньо з конкретними таблицями.
3. **V**iew.fxml – відповідає за відображення програми у вікні, та приймає ввід/вивід користувача.
4. **C**ontroller.java – обробляє ввід користувача, подає відповідну команду до моделі і очікує результат, який потім виводить користувачу.

**Структура меню програми**

Нижче (Рис. 3) наведено структуру меню програми:

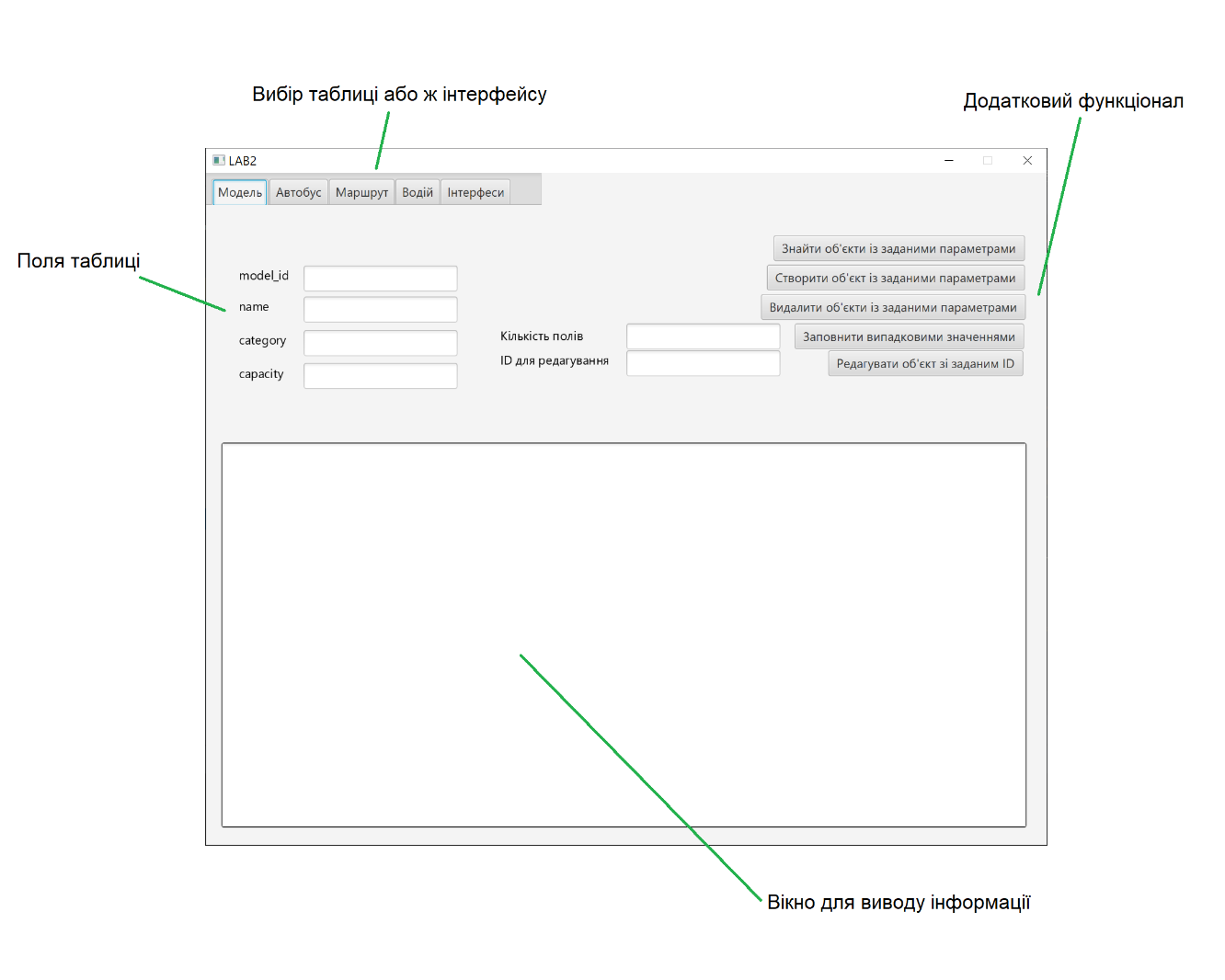
****

Рис. 3 – Структура меню програми

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

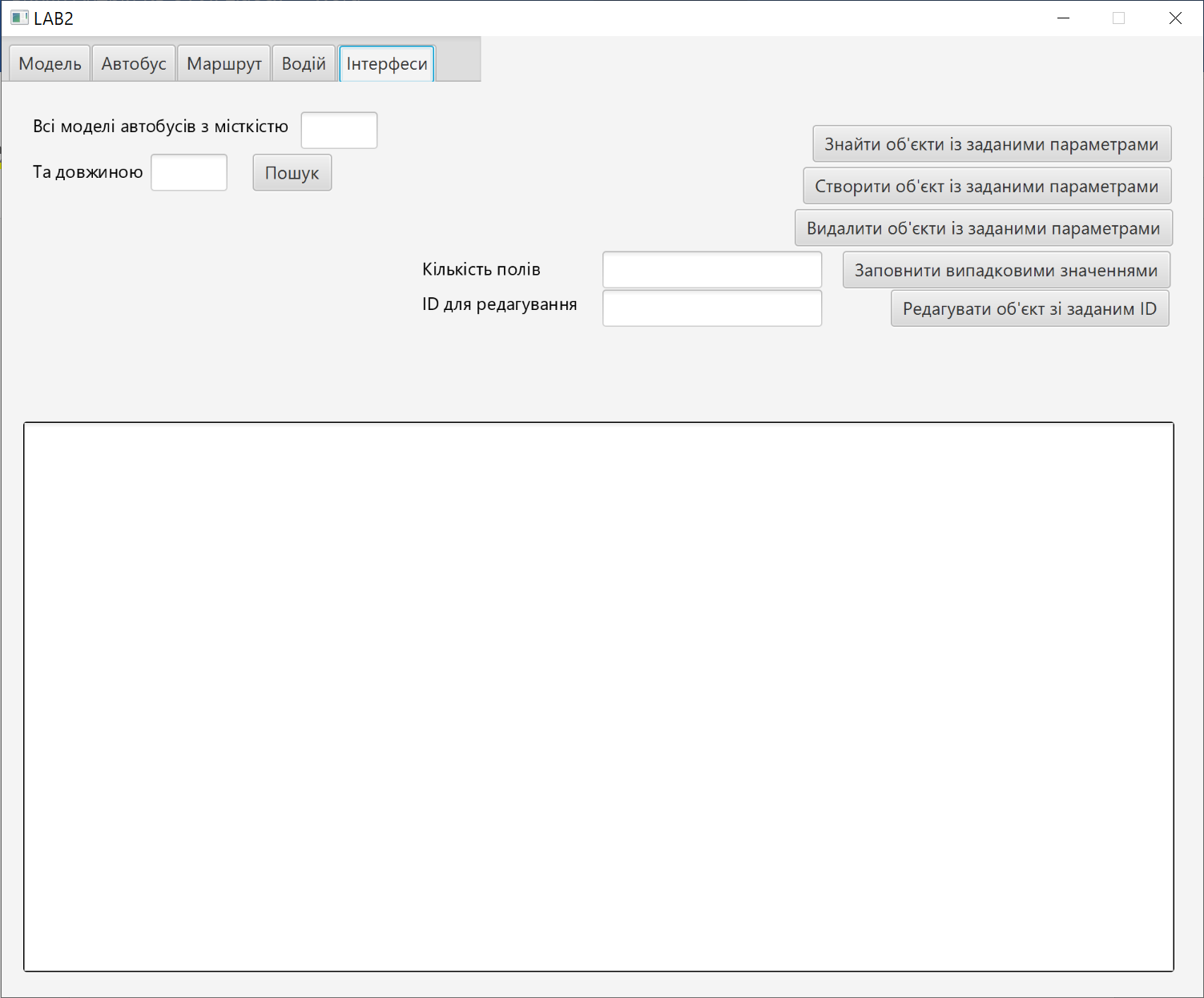
****

Рис. 3 – Реалізація інтерфейсу в меню

**Лістинг фрагментів програми з запитом пошуку**

Пошук всіх моделей із заданими параметрами, якщо параметри відсутні, то знаходить всі моделі 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 findAllmodel(Connection connection, String model\_id, String name, String category, String capacity) throws SQLException {  
  
 if (!model\_id.matches("\\d+|") || !capacity.matches("\\d+|")) {  
 return "Некоректний ввід даних!";  
 }  
  
 String sql = "SELECT \* FROM public.model WHERE";  
 sql += " model\_id = " + (model\_id.length() > 0 ? model\_id : "model\_id");  
 sql += " and name like " + (name.length() > 0 ? ("'%" + name + "%'") : "name");  
 sql += " and category = " + (category.length() > 0 ? category : "category");  
 sql += " and capacity = " + (capacity.length() > 0 ? capacity : "capacity");  
  
 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 += "-> model\_id = " + resultSet.getInt(1);  
 resultStr += " name = " + resultSet.getString(2);  
 resultStr += " category = " + resultSet.getString(3);  
 resultStr += " capacity = " + 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 автобусів із заданими параметрами, якщо параметри відсутні, то знаходить всі автобуси в 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 findAllbus(Connection connection, String bus\_id, String model\_id, String route\_id) throws SQLException {  
  
 if (!bus\_id.matches("\\d+|") || !model\_id.matches("\\d+|") || !route\_id.matches("\\d+|")) {  
 return "Некоректний ввід даних!";  
 }  
  
 String sql = "SELECT \* FROM public.bus WHERE";  
 sql += " bus\_id = " + (bus\_id.length() > 0 ? bus\_id : "bus\_id");  
 sql += " and model\_id = " + (model\_id.length() > 0 ? model\_id : "model\_id");  
 sql += " and route\_id = " + (route\_id.length() > 0 ? route\_id : "route\_id");  
  
 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 += "-> bus\_id = " + resultSet.getInt(1);  
 resultStr += " model\_id = " + resultSet.getInt(2);  
 resultStr += " route\_id = " + resultSet.getInt(3);  
 resultStr += "\n";  
 }  
 } catch (Exception ex) {  
 return ex.getMessage();  
 }  
 return (!resultStr.equals("Результат пошуку по заданим параметрам:\n") ? resultStr : "Нічого не знайдено!");  
 }  
}

Пошук всіх маршрутів із заданими параметрами, якщо параметри відсутні, то знаходить всі маршрути 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 findAllroute(Connection connection, String route\_id, String name, String length, String passenger\_traffic) throws SQLException {  
  
 if (!route\_id.matches("[+]?\\d+|") || !length.matches("[+]?\\d+|") || !passenger\_traffic.matches("[+]?\\d+|")) {  
 return "Некоректний ввід даних!";  
 }  
  
 String sql = "SELECT \* FROM public.route WHERE";  
 sql += " route\_id = " + (route\_id.length() > 0 ? route\_id : "route\_id");  
 sql += " and name like " + (name.length() > 0 ? ("'%" + name + "%'") : "name");  
 sql += " and length = " + (length.length() > 0 ? length : "length");  
 sql += " and passenger\_traffic = " + (passenger\_traffic.length() > 0 ? passenger\_traffic : "passenger\_traffic");  
  
 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 += "-> route\_id = " + resultSet.getInt(1);  
 resultStr += " name = " + resultSet.getString(2);  
 resultStr += " length = " + resultSet.getInt(3);  
 resultStr += " passenger\_traffic = " + 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 водіїв із заданими параметрами, якщо параметри відсутні, то знаходить всіх водіїв в 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 findAlldriver(Connection connection, String driver\_id, String bus\_id, String full\_name, String allowed\_categories, String working\_hours) throws SQLException {  
  
 if (!driver\_id.matches("\\d+|") || !bus\_id.matches("\\d+|") || !working\_hours.matches("\\d+|")) {  
 return "Некоректний ввід даних!";  
 }  
  
 String sql = "SELECT \* FROM public.driver WHERE";  
 sql += " driver\_id = " + (driver\_id.length() > 0 ? driver\_id : "driver\_id");  
 sql += " and bus\_id = " + (bus\_id.length() > 0 ? bus\_id : "bus\_id");  
 sql += " and full\_name like " + (full\_name.length() > 0 ? ("'%" + full\_name + "%'") : "full\_name");  
 sql += " and allowed\_categories like " + (allowed\_categories.length() > 0 ? ("'%" + allowed\_categories + "%'") : "allowed\_categories");  
 sql += " and working\_hours = " + (working\_hours.length() > 0 ? working\_hours : "working\_hours");  
  
 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 += "-> driver\_id = " + resultSet.getInt(1);  
 resultStr += " bus\_id = " + resultSet.getInt(2);  
 resultStr += " full\_name = " + resultSet.getString(3);  
 resultStr += " allowed\_categories = " + resultSet.getString(4);  
 resultStr += " working\_hours = " + resultSet.getInt(5);  
 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 моделі 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 createModel(Connection connection, String model\_id, String name, String category, String capacity) throws SQLException {  
  
 if (!model\_id.matches("\\d+")|| !capacity.matches("\\d+|") || name.equals("") || category.equals("")) {  
 return "Некоректний ввід даних!";  
 }  
  
 String sql = "INSERT INTO public.model (model\_id, name, category, capacity) VALUES (";  
 sql += model\_id + ", ";  
 sql += "'" + name + "', ";  
 sql += "'" + category + "', ";  
 sql += capacity + ")";  
  
 System.*out*.println("sql: " + sql);  
 try {  
 Statement statement = connection.createStatement();  
 statement.executeUpdate(sql);  
 } catch (Exception ex) {  
 return ex.getMessage();  
 }  
  
 String answer = "Створено:\n";  
 answer += "-> model\_id = " + model\_id;  
 answer += " name = " + name;  
 answer += " category = " + category;  
 answer += " capacity = " + capacity;  
  
 return answer;  
}

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

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

Створення водія 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 createDriver(Connection connection, String driver\_id, String bus\_id, String full\_name, String allowed\_categories, String working\_hours) throws SQLException {  
  
 if (!driver\_id.matches("\\d+|") || !bus\_id.matches("\\d+|") || !working\_hours.matches("\\d+|") || full\_name.equals("") || allowed\_categories.equals("")) {  
 return "Некоректний ввід даних!";  
 }  
  
 String sql = "INSERT INTO public.driver (driver\_id, bus\_id, full\_name, allowed\_categories, working\_hours) VALUES (";  
 sql += driver\_id + ", ";  
 sql += bus\_id + ", ";  
 sql += "'" + full\_name + "', ";  
 sql += "'" + allowed\_categories + "', ";  
 sql += working\_hours + ")";  
  
 System.*out*.println("sql: " + sql);  
 try {  
 Statement statement = connection.createStatement();  
 statement.executeUpdate(sql);  
 } catch (Exception ex) {  
 return ex.getMessage();  
 }  
  
 String answer = "Створено:\n";  
 answer += "-> driver\_id = " + driver\_id;  
 answer += " bus\_id = " + bus\_id;  
 answer += " full\_name = " + full\_name;  
 answer += " allowed\_categories = " + allowed\_categories;  
 answer += " working\_hours = " + working\_hours;  
  
 return answer;  
}

**Лістинг фрагментів програми з запитом вилучення**

Видаляє всі моделі 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 deleteModel(Connection connection, String model\_id, String name, String category, String capacity) throws SQLException {  
  
 if (!model\_id.matches("\\d+|") || !capacity.matches("\\d+|")) {  
 return "Некоректний ввід даних!";  
 }  
  
 String sql = "DELETE from driver where bus\_id in (select bus\_id from bus where model\_id in (select model\_id from model where";  
 sql += " model\_id = " + (model\_id.length() > 0 ? model\_id : "model\_id");  
 sql += " and name like " + (name.length() > 0 ? ("'%" + name + "%'") : "name");  
 sql += " and category = " + (category.length() > 0 ? category : "category");  
 sql += " and capacity = " + (capacity.length() > 0 ? capacity : "capacity") + "));\n";  
  
 sql += "DELETE from bus where model\_id in (select model\_id from model where";  
 sql += " model\_id = " + (model\_id.length() > 0 ? model\_id : "model\_id");  
 sql += " and name like " + (name.length() > 0 ? ("'%" + name + "%'") : "name");  
 sql += " and category = " + (category.length() > 0 ? category : "category");  
 sql += " and capacity = " + (capacity.length() > 0 ? capacity : "capacity") + ");\n";  
  
 sql += "DELETE from model where";  
 sql += " model\_id = " + (model\_id.length() > 0 ? model\_id : "model\_id");  
 sql += " and name like " + (name.length() > 0 ? ("'%" + name + "%'") : "name");  
 sql += " and category = " + (category.length() > 0 ? category : "category");  
 sql += " and capacity = " + (capacity.length() > 0 ? capacity : "capacity") + ";\n";  
  
 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 (model\_id.length() > 0) answer += " model\_id = " + model\_id;  
 if (name.length() > 0) answer += " name = " + name;  
 if (category.length() > 0) answer += " category = " + category;  
 if (capacity.length() > 0) answer += " capacity = " + capacity;  
  
 return answer;  
}

Видаляє всі автобуси 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 deleteBus(Connection connection, String bus\_id, String model\_id, String route\_id) throws SQLException {  
  
 if (!bus\_id.matches("\\d+") || !model\_id.matches("\\d+") || !route\_id.matches("\\d+")) {  
 return "Некоректний ввід даних!";  
 }  
  
 String sql = "DELETE from driver where bus\_id in (select bus\_id from bus where";  
 sql += " bus\_id = " + (bus\_id.length() > 0 ? bus\_id : "bus\_id");  
 sql += " and model\_id = " + (model\_id.length() > 0 ? model\_id : "model\_id");  
 sql += " and route\_id = " + (route\_id.length() > 0 ? route\_id : "route\_id") + ");\n";  
  
 sql += "DELETE from bus where";  
 sql += " bus\_id = " + (bus\_id.length() > 0 ? bus\_id : "bus\_id");  
 sql += " and model\_id = " + (model\_id.length() > 0 ? model\_id : "model\_id");  
 sql += " and route\_id = " + (route\_id.length() > 0 ? route\_id : "route\_id") + ";\n";  
 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 (bus\_id.length() > 0) answer += "-> bus\_id = " + bus\_id;  
 if (model\_id.length() > 0) answer += " model\_id = " + model\_id;  
 if (route\_id.length() > 0) answer += " route\_id = " + route\_id;  
  
 return answer;  
}

Видаляє всі маршрути 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 deleteRoute(Connection connection, String route\_id, String name, String length, String passenger\_traffic) throws SQLException {  
  
 if (!route\_id.matches("[+]?\\d+|") || !length.matches("[+]?\\d+|") || !passenger\_traffic.matches("[+]?\\d+|")) {  
 return "Некоректний ввід даних!";  
 }  
 String sql = "DELETE from driver where bus\_id in (select bus\_id from bus where route\_id in (select route\_id from route where";  
 sql += " route\_id = " + (route\_id.length() > 0 ? route\_id : "route\_id");  
 sql += " and name like " + (name.length() > 0 ? ("'%" + name + "%'") : "name");  
 sql += " and length = " + (length.length() > 0 ? length : "length");  
 sql += " and passenger\_traffic = " + (passenger\_traffic.length() > 0 ? passenger\_traffic : "passenger\_traffic") + "));\n";  
  
 sql += "DELETE from bus where route\_id in (select route\_id from route where";  
 sql += " route\_id = " + (route\_id.length() > 0 ? route\_id : "route\_id");  
 sql += " and name like " + (name.length() > 0 ? ("'%" + name + "%'") : "name");  
 sql += " and length = " + (length.length() > 0 ? length : "length");  
 sql += " and passenger\_traffic = " + (passenger\_traffic.length() > 0 ? passenger\_traffic : "passenger\_traffic") + ");\n";  
  
 sql += "DELETE from route where";  
 sql += " route\_id = " + (route\_id.length() > 0 ? route\_id : "route\_id");  
 sql += " and name like " + (name.length() > 0 ? ("'%" + name + "%'") : "name");  
 sql += " and length = " + (length.length() > 0 ? length : "length");  
 sql += " and passenger\_traffic = " + (passenger\_traffic.length() > 0 ? passenger\_traffic : "passenger\_traffic") + ";\n";  
  
 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 (route\_id.length() > 0) answer += " route\_id = " + route\_id;  
 if (name.length() > 0) answer += " name = " + name;  
  
 return answer;  
}

Видаляє всіх водіїв 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 deleteDriver(Connection connection, String driver\_id, String bus\_id, String full\_name, String allowed\_categories, String working\_hours) throws SQLException {  
  
 if (!driver\_id.matches("\\d+|") || !bus\_id.matches("\\d+|") || !working\_hours.matches("\\d+|")) {  
 return "Некоректний ввід даних!";  
 }  
  
 String sql = "DELETE from driver where";  
 sql += " driver\_id = " + (driver\_id.length() > 0 ? driver\_id : "driver\_id");  
 sql += " and bus\_id = " + (bus\_id.length() > 0 ? bus\_id : "bus\_id");  
 sql += " and full\_name like " + (full\_name.length() > 0 ? ("'%" + full\_name + "%'") : "full\_name");  
 sql += " and allowed\_categories like " + (allowed\_categories.length() > 0 ? ("'%" + allowed\_categories + "%'") : "allowed\_categories");  
 sql += " and working\_hours = " + (working\_hours.length() > 0 ? working\_hours : "working\_hours");  
  
 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 (driver\_id.length() > 0) answer += " driver\_id = " + driver\_id;  
 if (bus\_id.length() > 0) answer += " bus\_id = " + bus\_id;  
 if (full\_name.length() > 0) answer += " full\_name = " + full\_name;  
 if (allowed\_categories.length() > 0) answer += " allowed\_categories = " + allowed\_categories;  
 if (working\_hours.length() > 0) answer += " working\_hours = " + working\_hours;  
  
 return answer;  
}

**Лістинг фрагментів програми з запитом редагування**

Редагує модель 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 changeModel(Connection connection, String model\_id, String name, String category, String capacity, String id) throws SQLException {  
  
 if (!id.matches("\\d+") || !model\_id.matches("\\d+") || !capacity.matches("\\d+|") || name.equals("") || category.equals("")) {  
 return "Некоректний ввід даних!";  
 }  
  
 String sql = "UPDATE public.bus SET";  
 sql += " model\_id = " + (model\_id.length() > 0 ? model\_id : "model\_id") + ";\n";  
  
 sql += "UPDATE public.model SET";  
 sql += " model\_id = " + (model\_id.length() > 0 ? model\_id : "model\_id");  
 sql += ", name = " + (name.length() > 0 ? ("'" + name + "'") : "name");  
 sql += ", category = " + (category.length() > 0 ? category : "category");  
 sql += ", capacity = " + (capacity.length() > 0 ? capacity : "capacity");  
 sql += " WHERE model\_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 автобус 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 changeBus(Connection connection, String bus\_id, String model\_id, String route\_id, String id) throws SQLException {  
  
 if (!id.matches("\\d+") || !bus\_id.matches("\\d+") || !model\_id.matches("\\d+") || !route\_id.matches("\\d+")) {  
 return "Некоректний ввід даних!";  
 }  
  
 String sql = "UPDATE public.bus SET";  
 sql += " bus\_id = " + (bus\_id.length() > 0 ? bus\_id : "bus\_id");  
 sql += ", model\_id = " + (model\_id.length() > 0 ? model\_id : "model\_id");  
 sql += ", route\_id = " + (route\_id.length() > 0 ? route\_id : "route\_id");  
 sql += " WHERE bus\_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 маршрут 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 changeRoute(Connection connection, String route\_id, String name, String length, String passenger\_traffic, String id) throws SQLException {  
  
 if (!id.matches("[+]?\\d+|") || !route\_id.matches("[+]?\\d+|") || !length.matches("[+]?\\d+|") || !passenger\_traffic.matches("[+]?\\d+|")) {  
 return "Некоректний ввід даних!";  
 }  
  
 String sql = "UPDATE public.route SET";  
 sql += " route\_id = " + (route\_id.length() > 0 ? route\_id : "route\_id");  
 sql += ", name = " + (name.length() > 0 ? ("'" + name + "'") : "name");  
 sql += " WHERE route\_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 водія 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 changeDriver(Connection connection, String driver\_id, String bus\_id, String full\_name, String allowed\_categories, String working\_hours, String id) throws SQLException {  
  
 if (!id.matches("\\d+|") || !driver\_id.matches("\\d+|") || !bus\_id.matches("\\d+|") || !working\_hours.matches("\\d+|")) {  
 return "Некоректний ввід даних!";  
 }  
  
 String sql = "UPDATE public.driver SET";  
 sql += " driver\_id = " + (driver\_id.length() > 0 ? driver\_id : "driver\_id");  
 sql += ", bus\_id = " + (bus\_id.length() > 0 ? bus\_id : "bus\_id");  
 sql += ", full\_name = " + (full\_name.length() > 0 ? ("'" + full\_name + "'") : "full\_name");  
 sql += ", allowed\_categories = " + (allowed\_categories.length() > 0 ? ("'" + allowed\_categories + "'") : "allowed\_categories");  
 sql += ", working\_hours = " + (working\_hours.length() > 0 ? working\_hours : "working\_hours");  
 sql += " WHERE driver\_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**

Пошук моделей з місткістю більше введеного, які перевищують задану довжину маршрута. Додатково повертає час виконання запиту в мілісекундах. Є контроль введених даних на коректність типу. Результат, що повертає ця функція, виводиться користувачу

public static String getInterface1(Connection connection, String capacity, String length) {  
  
 if (!capacity.matches("\\d+") || !length.matches("\\d+")) {  
 return "Некоректний ввід даних!";  
 }  
  
 String sql = "Select s.capacity, f.length from public.model s, public.route f, public.bus h where h.model\_id = s.id and h.route\_id = f.id and";  
 sql += " s.capacity < (?)";  
 sql += " and s.length < (?)";  
  
 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*(capacity));  
 statement.setInt(2, *parseInt*(length));  
  
 try {  
 ResultSet resultSet = statement.executeQuery();  
 while (resultSet.next()) {  
 resultStr += "-> Пасажиромісткість '" + resultSet.getInt(1) + ",";  
 resultStr += " Довжина " + resultSet.getInt(2);  
 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.model (model\_id, name, category, capacity) VALUES ((select max(model\_id) from model) + 1, substr(md5(random()::text), 0, 25), substr(md5(random()::text), 0, 25), FLOOR(RANDOM() \* 100))";  
 break;  
 case 2:  
 sql = "INSERT INTO public.bus (bus\_id, model\_id, route\_id) VALUES ((select max(bus\_id) from bus) + 1,(select model\_id from model order by random() limit 1), (select route\_id from route order by random() limit 1))";  
 break;  
 case 3:  
 sql = "INSERT INTO public.route (route\_id, name, length, passenger\_traffic) VALUES ((select max(route\_id) from route) + 1, substr(md5(random()::text), 0, 25), FLOOR(RANDOM() \* 100), FLOOR(RANDOM() \* 100))";  
 break;  
 case 4:  
 sql = "INSERT INTO public.driver (driver\_id, bus\_id, full\_name, allowed\_categories, working\_hours) VALUES ((select max(driver\_id) from driver) + 1, (select bus\_id from bus order by random() limit 1), substr(md5(random()::text), 0, 25), substr(md5(random()::text), 0, 25), FLOOR(RANDOM() \* 100))";  
 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!";  
}

**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, 930, 820));  
 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 model;  
  
 @FXML  
 private TextField model\_model\_id;  
  
 @FXML  
 private TextField model\_name;  
  
 @FXML  
 private TextField model\_category;  
  
 @FXML  
 private TextField model\_capacity;  
  
 @FXML  
 private Tab bus;  
  
 @FXML  
 private TextField bus\_bus\_id;  
  
 @FXML  
 private TextField bus\_model\_id;  
  
 @FXML  
 private TextField bus\_route\_id;  
  
 @FXML  
 private Tab route;  
  
 @FXML  
 private TextField route\_route\_id;  
  
 @FXML  
 private TextField route\_name;  
  
 @FXML  
 private TextField route\_length;  
  
 @FXML  
 private TextField route\_passenger\_traffic;  
  
 @FXML  
 private Tab driver;  
  
 @FXML  
 private TextField driver\_driver\_id;  
  
 @FXML  
 private TextField driver\_bus\_id;  
  
 @FXML  
 private TextField driver\_full\_name;  
  
 @FXML  
 private TextField driver\_allowed\_categories;  
  
 @FXML  
 private TextField driver\_working\_hours;  
  
 @FXML  
 private TextField capacity;  
  
 @FXML  
 private TextField length;  
  
 @FXML  
 private TextField idChange;  
  
 @FXML  
 private TextArea textArea;  
  
 @FXML  
 private TextField randomCount;  
  
 int currentTab() {  
 if (model.isSelected()) return 1;  
 if (bus.isSelected()) return 2;  
 if (route.isSelected()) return 3;  
 if (driver.isSelected()) return 4;  
 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.*findAllmodel*(connection, model\_model\_id.getText(), model\_name.getText(), model\_category.getText(), model\_capacity.getText()));  
 break;  
 case 2:  
 textArea.setText(textArea.getText() + Model.*findAllbus*(connection, bus\_bus\_id.getText(), bus\_model\_id.getText(), bus\_route\_id.getText()));  
 break;  
 case 3:  
 textArea.setText(textArea.getText() + Model.*findAllroute*(connection, route\_route\_id.getText(), route\_name.getText(), route\_length.getText(), route\_passenger\_traffic.getText()));  
 break;  
 case 4:  
 textArea.setText(textArea.getText() + Model.*findAlldriver*(connection, driver\_driver\_id.getText(), driver\_bus\_id.getText(), driver\_full\_name.getText(), driver\_allowed\_categories.getText(), driver\_working\_hours.getText()));  
 break;  
 default:  
 break;  
 }  
 }  
  
 @FXML  
 //Поиск по коду товара  
 public void create(ActionEvent actionEvent) throws SQLException {  
 textArea.clear();  
 connection = Model.*getConnection*();  
 switch (currentTab()) {  
 case 1:  
 textArea.setText(textArea.getText() + Model.*createModel*(connection, model\_model\_id.getText(), model\_name.getText(), model\_category.getText(), model\_capacity.getText()));  
 break;  
 case 2:  
 textArea.setText(textArea.getText() + Model.*createBus*(connection, bus\_bus\_id.getText(), bus\_model\_id.getText(), bus\_route\_id.getText()));  
 break;  
 case 3:  
 textArea.setText(textArea.getText() + Model.*createRoute*(connection, route\_route\_id.getText(), route\_name.getText(), route\_length.getText(), route\_passenger\_traffic.getText()));  
 break;  
 case 4:  
 textArea.setText(textArea.getText() + Model.*createDriver*(connection, driver\_driver\_id.getText(), driver\_bus\_id.getText(), driver\_full\_name.getText(), driver\_allowed\_categories.getText(), driver\_working\_hours.getText()));  
 break;  
 default:  
 break;  
 }  
 }  
  
 @FXML  
 //Поиск по коду товара  
 public void delete(ActionEvent actionEvent) throws SQLException {  
 textArea.clear();  
 connection = Model.*getConnection*();  
 switch (currentTab()) {  
 case 1:  
 textArea.setText(textArea.getText() + Model.*deleteModel*(connection, model\_model\_id.getText(), model\_name.getText(), model\_category.getText(), model\_capacity.getText()));  
 break;  
 case 2:  
 textArea.setText(textArea.getText() + Model.*deleteBus*(connection, bus\_bus\_id.getText(), bus\_model\_id.getText(), bus\_route\_id.getText()));  
 break;  
 case 3:  
 textArea.setText(textArea.getText() + Model.*deleteRoute*(connection, route\_route\_id.getText(), route\_name.getText(), route\_length.getText(), route\_passenger\_traffic.getText()));  
 break;  
 case 4:  
 textArea.setText(textArea.getText() + Model.*deleteDriver*(connection, driver\_driver\_id.getText(), driver\_bus\_id.getText(), driver\_full\_name.getText(), driver\_allowed\_categories.getText(), driver\_working\_hours.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  
 //Поиск по коду товара  
 public void change(ActionEvent actionEvent) throws SQLException {  
 textArea.clear();  
 connection = Model.*getConnection*();  
 switch (currentTab()) {  
 case 1:  
 textArea.setText(textArea.getText() + Model.*changeModel*(connection, model\_model\_id.getText(), model\_name.getText(), model\_category.getText(), model\_capacity.getText(), idChange.getText()));  
 break;  
 case 2:  
 textArea.setText(textArea.getText() + Model.*changeBus*(connection, bus\_bus\_id.getText(), bus\_model\_id.getText(), bus\_route\_id.getText(), idChange.getText()));  
 break;  
 case 3:  
 textArea.setText(textArea.getText() + Model.*changeRoute*(connection, route\_route\_id.getText(), route\_name.getText(), route\_length.getText(), route\_passenger\_traffic.getText(), idChange.getText()));  
 break;  
 case 4:  
 textArea.setText(textArea.getText() + Model.*changeDriver*(connection, driver\_driver\_id.getText(), driver\_bus\_id.getText(), driver\_full\_name.getText(), driver\_allowed\_categories.getText(), driver\_working\_hours.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, capacity.getText(), length.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* = "postgres";  
 static final String *password* = "AZAZ1q1q";  
 static final String *url* = "jdbc:postgresql://localhost:5432/BusDepot";  
  
  
 public static Connection getConnection() throws SQLException {  
 return DriverManager.*getConnection*(*url*, *user*, *password*);  
 }  
  
 public static String findAllmodel(Connection connection, String model\_id, String name, String category, String capacity) throws SQLException {  
  
 if (!model\_id.matches("\\d+|") || !capacity.matches("\\d+|")) {  
 return "Некоректний ввід даних!";  
 }  
  
 String sql = "SELECT \* FROM public.model WHERE";  
 sql += " model\_id = " + (model\_id.length() > 0 ? model\_id : "model\_id");  
 sql += " and name like " + (name.length() > 0 ? ("'%" + name + "%'") : "name");  
 sql += " and category = " + (category.length() > 0 ? category : "category");  
 sql += " and capacity = " + (capacity.length() > 0 ? capacity : "capacity");  
  
 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 += "-> model\_id = " + resultSet.getInt(1);  
 resultStr += " name = " + resultSet.getString(2);  
 resultStr += " category = " + resultSet.getString(3);  
 resultStr += " capacity = " + 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 findAllbus(Connection connection, String bus\_id, String model\_id, String route\_id) throws SQLException {  
  
 if (!bus\_id.matches("\\d+|") || !model\_id.matches("\\d+|") || !route\_id.matches("\\d+|")) {  
 return "Некоректний ввід даних!";  
 }  
  
 String sql = "SELECT \* FROM public.bus WHERE";  
 sql += " bus\_id = " + (bus\_id.length() > 0 ? bus\_id : "bus\_id");  
 sql += " and model\_id = " + (model\_id.length() > 0 ? model\_id : "model\_id");  
 sql += " and route\_id = " + (route\_id.length() > 0 ? route\_id : "route\_id");  
  
 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 += "-> bus\_id = " + resultSet.getInt(1);  
 resultStr += " model\_id = " + resultSet.getInt(2);  
 resultStr += " route\_id = " + resultSet.getInt(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 findAllroute(Connection connection, String route\_id, String name, String length, String passenger\_traffic) throws SQLException {  
  
 if (!route\_id.matches("[+]?\\d+|") || !length.matches("[+]?\\d+|") || !passenger\_traffic.matches("[+]?\\d+|")) {  
 return "Некоректний ввід даних!";  
 }  
  
 String sql = "SELECT \* FROM public.route WHERE";  
 sql += " route\_id = " + (route\_id.length() > 0 ? route\_id : "route\_id");  
 sql += " and name like " + (name.length() > 0 ? ("'%" + name + "%'") : "name");  
 sql += " and length = " + (length.length() > 0 ? length : "length");  
 sql += " and passenger\_traffic = " + (passenger\_traffic.length() > 0 ? passenger\_traffic : "passenger\_traffic");  
  
 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 += "-> route\_id = " + resultSet.getInt(1);  
 resultStr += " name = " + resultSet.getString(2);  
 resultStr += " length = " + resultSet.getInt(3);  
 resultStr += " passenger\_traffic = " + 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 findAlldriver(Connection connection, String driver\_id, String bus\_id, String full\_name, String allowed\_categories, String working\_hours) throws SQLException {  
  
 if (!driver\_id.matches("\\d+|") || !bus\_id.matches("\\d+|") || !working\_hours.matches("\\d+|")) {  
 return "Некоректний ввід даних!";  
 }  
  
 String sql = "SELECT \* FROM public.driver WHERE";  
 sql += " driver\_id = " + (driver\_id.length() > 0 ? driver\_id : "driver\_id");  
 sql += " and bus\_id = " + (bus\_id.length() > 0 ? bus\_id : "bus\_id");  
 sql += " and full\_name like " + (full\_name.length() > 0 ? ("'%" + full\_name + "%'") : "full\_name");  
 sql += " and allowed\_categories like " + (allowed\_categories.length() > 0 ? ("'%" + allowed\_categories + "%'") : "allowed\_categories");  
 sql += " and working\_hours = " + (working\_hours.length() > 0 ? working\_hours : "working\_hours");  
  
 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 += "-> driver\_id = " + resultSet.getInt(1);  
 resultStr += " bus\_id = " + resultSet.getInt(2);  
 resultStr += " full\_name = " + resultSet.getString(3);  
 resultStr += " allowed\_categories = " + resultSet.getString(4);  
 resultStr += " working\_hours = " + 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 createModel(Connection connection, String model\_id, String name, String category, String capacity) throws SQLException {  
  
 if (!model\_id.matches("\\d+")|| !capacity.matches("\\d+|") || name.equals("") || category.equals("")) {  
 return "Некоректний ввід даних!";  
 }  
  
 String sql = "INSERT INTO public.model (model\_id, name, category, capacity) VALUES (";  
 sql += model\_id + ", ";  
 sql += "'" + name + "', ";  
 sql += "'" + category + "', ";  
 sql += capacity + ")";  
  
 System.*out*.println("sql: " + sql);  
 try {  
 Statement statement = connection.createStatement();  
 statement.executeUpdate(sql);  
 } catch (Exception ex) {  
 return ex.getMessage();  
 }  
  
 String answer = "Створено:\n";  
 answer += "-> model\_id = " + model\_id;  
 answer += " name = " + name;  
 answer += " category = " + category;  
 answer += " capacity = " + capacity;  
  
 return answer;  
 }  
  
 public static String createBus(Connection connection, String bus\_id, String model\_id, String route\_id) throws SQLException {  
  
 if (!bus\_id.matches("\\d+") || !model\_id.matches("\\d+") || !route\_id.matches("\\d+")) {  
 return "Некоректний ввід даних!";  
 }  
  
 String sql = "INSERT INTO public.bus (bus\_id, model\_id, route\_id) VALUES (";  
 sql += bus\_id + ", ";  
 sql += model\_id + ", ";  
 sql += route\_id + ")";  
  
 System.*out*.println("sql: " + sql);  
 try {  
 Statement statement = connection.createStatement();  
 statement.executeUpdate(sql);  
 } catch (Exception ex) {  
 return ex.getMessage();  
 }  
  
 String answer = "Створено:\n";  
 answer += "-> bus\_id = " + bus\_id;  
 answer += " model\_id = " + model\_id;  
 answer += " route\_id = " + route\_id;  
  
 return answer;  
 }  
  
 public static String createRoute(Connection connection, String route\_id, String name, String length, String passenger\_traffic) throws SQLException {  
  
 if (!route\_id.matches("[+]?\\d+|") || !length.matches("[+]?\\d+|") || !passenger\_traffic.matches("[+]?\\d+|") || name.equals("")) {  
 return "Некоректний ввід даних!";  
 }  
  
 String sql = "INSERT INTO public.route (route\_id, name, length, passenger\_traffic) VALUES (";  
 sql += route\_id + ", ";  
 sql += "'" + name + "', ";  
 sql += length + ", ";  
 sql += passenger\_traffic + ")";  
  
 System.*out*.println("sql: " + sql);  
 try {  
 Statement statement = connection.createStatement();  
 statement.executeUpdate(sql);  
 } catch (Exception ex) {  
 return ex.getMessage();  
 }  
  
 String answer = "Створено:\n";  
 answer += "-> route\_id = " + route\_id;  
 answer += " name = " + name;  
  
 return answer;  
 }  
  
 public static String createDriver(Connection connection, String driver\_id, String bus\_id, String full\_name, String allowed\_categories, String working\_hours) throws SQLException {  
  
 if (!driver\_id.matches("\\d+|") || !bus\_id.matches("\\d+|") || !working\_hours.matches("\\d+|") || full\_name.equals("") || allowed\_categories.equals("")) {  
 return "Некоректний ввід даних!";  
 }  
  
 String sql = "INSERT INTO public.driver (driver\_id, bus\_id, full\_name, allowed\_categories, working\_hours) VALUES (";  
 sql += driver\_id + ", ";  
 sql += bus\_id + ", ";  
 sql += "'" + full\_name + "', ";  
 sql += "'" + allowed\_categories + "', ";  
 sql += working\_hours + ")";  
  
 System.*out*.println("sql: " + sql);  
 try {  
 Statement statement = connection.createStatement();  
 statement.executeUpdate(sql);  
 } catch (Exception ex) {  
 return ex.getMessage();  
 }  
  
 String answer = "Створено:\n";  
 answer += "-> driver\_id = " + driver\_id;  
 answer += " bus\_id = " + bus\_id;  
 answer += " full\_name = " + full\_name;  
 answer += " allowed\_categories = " + allowed\_categories;  
 answer += " working\_hours = " + working\_hours;  
  
 return answer;  
 }  
  
 public static String deleteModel(Connection connection, String model\_id, String name, String category, String capacity) throws SQLException {  
  
 if (!model\_id.matches("\\d+|") || !capacity.matches("\\d+|")) {  
 return "Некоректний ввід даних!";  
 }  
  
 String sql = "DELETE from driver where bus\_id in (select bus\_id from bus where model\_id in (select model\_id from model where";  
 sql += " model\_id = " + (model\_id.length() > 0 ? model\_id : "model\_id");  
 sql += " and name like " + (name.length() > 0 ? ("'%" + name + "%'") : "name");  
 sql += " and category = " + (category.length() > 0 ? category : "category");  
 sql += " and capacity = " + (capacity.length() > 0 ? capacity : "capacity") + "));\n";  
  
 sql += "DELETE from bus where model\_id in (select model\_id from model where";  
 sql += " model\_id = " + (model\_id.length() > 0 ? model\_id : "model\_id");  
 sql += " and name like " + (name.length() > 0 ? ("'%" + name + "%'") : "name");  
 sql += " and category = " + (category.length() > 0 ? category : "category");  
 sql += " and capacity = " + (capacity.length() > 0 ? capacity : "capacity") + ");\n";  
  
 sql += "DELETE from model where";  
 sql += " model\_id = " + (model\_id.length() > 0 ? model\_id : "model\_id");  
 sql += " and name like " + (name.length() > 0 ? ("'%" + name + "%'") : "name");  
 sql += " and category = " + (category.length() > 0 ? category : "category");  
 sql += " and capacity = " + (capacity.length() > 0 ? capacity : "capacity") + ";\n";  
  
 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 (model\_id.length() > 0) answer += " model\_id = " + model\_id;  
 if (name.length() > 0) answer += " name = " + name;  
 if (category.length() > 0) answer += " category = " + category;  
 if (capacity.length() > 0) answer += " capacity = " + capacity;  
  
 return answer;  
 }  
  
 public static String deleteBus(Connection connection, String bus\_id, String model\_id, String route\_id) throws SQLException {  
  
 if (!bus\_id.matches("\\d+") || !model\_id.matches("\\d+") || !route\_id.matches("\\d+")) {  
 return "Некоректний ввід даних!";  
 }  
  
 String sql = "DELETE from driver where bus\_id in (select bus\_id from bus where";  
 sql += " bus\_id = " + (bus\_id.length() > 0 ? bus\_id : "bus\_id");  
 sql += " and model\_id = " + (model\_id.length() > 0 ? model\_id : "model\_id");  
 sql += " and route\_id = " + (route\_id.length() > 0 ? route\_id : "route\_id") + ");\n";  
  
 sql += "DELETE from bus where";  
 sql += " bus\_id = " + (bus\_id.length() > 0 ? bus\_id : "bus\_id");  
 sql += " and model\_id = " + (model\_id.length() > 0 ? model\_id : "model\_id");  
 sql += " and route\_id = " + (route\_id.length() > 0 ? route\_id : "route\_id") + ";\n";  
 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 (bus\_id.length() > 0) answer += "-> bus\_id = " + bus\_id;  
 if (model\_id.length() > 0) answer += " model\_id = " + model\_id;  
 if (route\_id.length() > 0) answer += " route\_id = " + route\_id;  
  
 return answer;  
 }  
  
 public static String deleteRoute(Connection connection, String route\_id, String name, String length, String passenger\_traffic) throws SQLException {  
  
 if (!route\_id.matches("[+]?\\d+|") || !length.matches("[+]?\\d+|") || !passenger\_traffic.matches("[+]?\\d+|")) {  
 return "Некоректний ввід даних!";  
 }  
 String sql = "DELETE from driver where bus\_id in (select bus\_id from bus where route\_id in (select route\_id from route where";  
 sql += " route\_id = " + (route\_id.length() > 0 ? route\_id : "route\_id");  
 sql += " and name like " + (name.length() > 0 ? ("'%" + name + "%'") : "name");  
 sql += " and length = " + (length.length() > 0 ? length : "length");  
 sql += " and passenger\_traffic = " + (passenger\_traffic.length() > 0 ? passenger\_traffic : "passenger\_traffic") + "));\n";  
  
 sql += "DELETE from bus where route\_id in (select route\_id from route where";  
 sql += " route\_id = " + (route\_id.length() > 0 ? route\_id : "route\_id");  
 sql += " and name like " + (name.length() > 0 ? ("'%" + name + "%'") : "name");  
 sql += " and length = " + (length.length() > 0 ? length : "length");  
 sql += " and passenger\_traffic = " + (passenger\_traffic.length() > 0 ? passenger\_traffic : "passenger\_traffic") + ");\n";  
  
 sql += "DELETE from route where";  
 sql += " route\_id = " + (route\_id.length() > 0 ? route\_id : "route\_id");  
 sql += " and name like " + (name.length() > 0 ? ("'%" + name + "%'") : "name");  
 sql += " and length = " + (length.length() > 0 ? length : "length");  
 sql += " and passenger\_traffic = " + (passenger\_traffic.length() > 0 ? passenger\_traffic : "passenger\_traffic") + ";\n";  
  
 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 (route\_id.length() > 0) answer += " route\_id = " + route\_id;  
 if (name.length() > 0) answer += " name = " + name;  
  
 return answer;  
 }  
  
 public static String deleteDriver(Connection connection, String driver\_id, String bus\_id, String full\_name, String allowed\_categories, String working\_hours) throws SQLException {  
  
 if (!driver\_id.matches("\\d+|") || !bus\_id.matches("\\d+|") || !working\_hours.matches("\\d+|")) {  
 return "Некоректний ввід даних!";  
 }  
  
 String sql = "DELETE from driver where";  
 sql += " driver\_id = " + (driver\_id.length() > 0 ? driver\_id : "driver\_id");  
 sql += " and bus\_id = " + (bus\_id.length() > 0 ? bus\_id : "bus\_id");  
 sql += " and full\_name like " + (full\_name.length() > 0 ? ("'%" + full\_name + "%'") : "full\_name");  
 sql += " and allowed\_categories like " + (allowed\_categories.length() > 0 ? ("'%" + allowed\_categories + "%'") : "allowed\_categories");  
 sql += " and working\_hours = " + (working\_hours.length() > 0 ? working\_hours : "working\_hours");  
  
 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 (driver\_id.length() > 0) answer += " driver\_id = " + driver\_id;  
 if (bus\_id.length() > 0) answer += " bus\_id = " + bus\_id;  
 if (full\_name.length() > 0) answer += " full\_name = " + full\_name;  
 if (allowed\_categories.length() > 0) answer += " allowed\_categories = " + allowed\_categories;  
 if (working\_hours.length() > 0) answer += " working\_hours = " + working\_hours;  
  
 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.model (model\_id, name, category, capacity) VALUES ((select max(model\_id) from model) + 1, substr(md5(random()::text), 0, 25), substr(md5(random()::text), 0, 25), FLOOR(RANDOM() \* 100))";  
 break;  
 case 2:  
 sql = "INSERT INTO public.bus (bus\_id, model\_id, route\_id) VALUES ((select max(bus\_id) from bus) + 1,(select model\_id from model order by random() limit 1), (select route\_id from route order by random() limit 1))";  
 break;  
 case 3:  
 sql = "INSERT INTO public.route (route\_id, name, length, passenger\_traffic) VALUES ((select max(route\_id) from route) + 1, substr(md5(random()::text), 0, 25), FLOOR(RANDOM() \* 100), FLOOR(RANDOM() \* 100))";  
 break;  
 case 4:  
 sql = "INSERT INTO public.driver (driver\_id, bus\_id, full\_name, allowed\_categories, working\_hours) VALUES ((select max(driver\_id) from driver) + 1, (select bus\_id from bus order by random() limit 1), substr(md5(random()::text), 0, 25), substr(md5(random()::text), 0, 25), FLOOR(RANDOM() \* 100))";  
 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 changeModel(Connection connection, String model\_id, String name, String category, String capacity, String id) throws SQLException {  
  
 if (!id.matches("\\d+") || !model\_id.matches("\\d+") || !capacity.matches("\\d+|") || name.equals("") || category.equals("")) {  
 return "Некоректний ввід даних!";  
 }  
  
 String sql = "UPDATE public.bus SET";  
 sql += " model\_id = " + (model\_id.length() > 0 ? model\_id : "model\_id") + ";\n";  
  
 sql += "UPDATE public.model SET";  
 sql += " model\_id = " + (model\_id.length() > 0 ? model\_id : "model\_id");  
 sql += ", name = " + (name.length() > 0 ? ("'" + name + "'") : "name");  
 sql += ", category = " + (category.length() > 0 ? category : "category");  
 sql += ", capacity = " + (capacity.length() > 0 ? capacity : "capacity");  
 sql += " WHERE model\_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 changeBus(Connection connection, String bus\_id, String model\_id, String route\_id, String id) throws SQLException {  
  
 if (!id.matches("\\d+") || !bus\_id.matches("\\d+") || !model\_id.matches("\\d+") || !route\_id.matches("\\d+")) {  
 return "Некоректний ввід даних!";  
 }  
  
 String sql = "UPDATE public.bus SET";  
 sql += " bus\_id = " + (bus\_id.length() > 0 ? bus\_id : "bus\_id");  
 sql += ", model\_id = " + (model\_id.length() > 0 ? model\_id : "model\_id");  
 sql += ", route\_id = " + (route\_id.length() > 0 ? route\_id : "route\_id");  
 sql += " WHERE bus\_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 changeRoute(Connection connection, String route\_id, String name, String length, String passenger\_traffic, String id) throws SQLException {  
  
 if (!id.matches("[+]?\\d+|") || !route\_id.matches("[+]?\\d+|") || !length.matches("[+]?\\d+|") || !passenger\_traffic.matches("[+]?\\d+|")) {  
 return "Некоректний ввід даних!";  
 }  
  
 String sql = "UPDATE public.route SET";  
 sql += " route\_id = " + (route\_id.length() > 0 ? route\_id : "route\_id");  
 sql += ", name = " + (name.length() > 0 ? ("'" + name + "'") : "name");  
 sql += " WHERE route\_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 changeDriver(Connection connection, String driver\_id, String bus\_id, String full\_name, String allowed\_categories, String working\_hours, String id) throws SQLException {  
  
 if (!id.matches("\\d+|") || !driver\_id.matches("\\d+|") || !bus\_id.matches("\\d+|") || !working\_hours.matches("\\d+|")) {  
 return "Некоректний ввід даних!";  
 }  
  
 String sql = "UPDATE public.driver SET";  
 sql += " driver\_id = " + (driver\_id.length() > 0 ? driver\_id : "driver\_id");  
 sql += ", bus\_id = " + (bus\_id.length() > 0 ? bus\_id : "bus\_id");  
 sql += ", full\_name = " + (full\_name.length() > 0 ? ("'" + full\_name + "'") : "full\_name");  
 sql += ", allowed\_categories = " + (allowed\_categories.length() > 0 ? ("'" + allowed\_categories + "'") : "allowed\_categories");  
 sql += ", working\_hours = " + (working\_hours.length() > 0 ? working\_hours : "working\_hours");  
 sql += " WHERE driver\_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 capacity, String length) {  
  
 if (!capacity.matches("\\d+") || !length.matches("\\d+")) {  
 return "Некоректний ввід даних!";  
 }  
  
 String sql = "Select s.capacity, f.length from public.model s, public.route f, public.bus h where h.model\_id = s.id and h.route\_id = f.id and";  
 sql += " s.capacity < (?)";  
 sql += " and s.length < (?)";  
  
 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*(capacity));  
 statement.setInt(2, *parseInt*(length));  
  
 try {  
 ResultSet resultSet = statement.executeQuery();  
 while (resultSet.next()) {  
 resultStr += "-> Пасажиромісткість '" + resultSet.getInt(1) + ",";  
 resultStr += " Довжина " + resultSet.getInt(2);  
 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="400.0" tabClosingPolicy="UNAVAILABLE">  
 <tabs>  
 <Tab fx:id="model" text="Модель">  
 <content>  
 <AnchorPane minHeight="0.0" minWidth="0.0" prefHeight="719.0" prefWidth="1080.0">  
 <children>  
 <TextField fx:id="model\_model\_id" layoutX="117.0" layoutY="72.0" />  
 <Text layoutX="40.0" layoutY="89.0" strokeType="OUTSIDE" strokeWidth="0.0" text="model\_id" />  
 <TextField fx:id="model\_name" layoutX="117.0" layoutY="109.0" />  
 <Text layoutX="40.0" layoutY="126.0" strokeType="OUTSIDE" strokeWidth="0.0" text="name" />  
 <TextField fx:id="model\_category" layoutX="117.0" layoutY="149.0" />  
 <Text layoutX="40.0" layoutY="166.0" strokeType="OUTSIDE" strokeWidth="0.0" text="category" />  
 <TextField fx:id="model\_capacity" layoutX="117.0" layoutY="188.0" />  
 <Text layoutX="40.0" layoutY="206.0" strokeType="OUTSIDE" strokeWidth="0.0" text="capacity" />  
 </children>  
 </AnchorPane>  
 </content>  
 </Tab>  
 <Tab fx:id="bus" text="Автобус">  
 <content>  
 <AnchorPane minHeight="0.0" minWidth="0.0" prefHeight="180.0" prefWidth="200.0">  
 <children>  
 <TextField fx:id="bus\_bus\_id" layoutX="112.0" layoutY="59.0" />  
 <Text layoutX="47.0" layoutY="76.0" strokeType="OUTSIDE" strokeWidth="0.0" text="bus\_id" />  
 <TextField fx:id="bus\_model\_id" layoutX="112.0" layoutY="92.0" />  
 <Text layoutX="47.0" layoutY="109.0" strokeType="OUTSIDE" strokeWidth="0.0" text="model\_id" />  
 <TextField fx:id="bus\_route\_id" layoutX="112.0" layoutY="128.0" />  
 <Text layoutX="47.0" layoutY="145.0" strokeType="OUTSIDE" strokeWidth="0.0" text="route\_id" />  
 </children>  
 </AnchorPane>  
 </content>  
 </Tab>  
 <Tab fx:id="route" text="Маршрут">  
 <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="route\_id" />  
 <TextField fx:id="route\_route\_id" layoutX="106.0" layoutY="58.0" />  
 <Text layoutX="48.0" layoutY="110.0" strokeType="OUTSIDE" strokeWidth="0.0" text="name" />  
 <TextField fx:id="route\_name" layoutX="106.0" layoutY="93.0" />  
 <Text layoutX="48.0" layoutY="145.0" strokeType="OUTSIDE" strokeWidth="0.0" text="length" />  
 <TextField fx:id="route\_length" layoutX="106.0" layoutY="128.0" />  
 <Text layoutX="48.0" layoutY="190.0" strokeType="OUTSIDE" strokeWidth="0.0" text="passenger\_traffic" />  
 <TextField fx:id="route\_passenger\_traffic" layoutX="235.0" layoutY="163.0" />  
 </children>  
 </AnchorPane>  
 </content>  
 </Tab>  
 <Tab fx:id="driver" text="Водій">  
 <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="driver\_id" />  
 <TextField fx:id="driver\_driver\_id" layoutX="107.0" layoutY="56.0" />  
 <Text layoutX="49.0" layoutY="106.0" strokeType="OUTSIDE" strokeWidth="0.0" text="bus\_id" />  
 <TextField fx:id="driver\_bus\_id" layoutX="107.0" layoutY="89.0" />  
 <Text layoutX="49.0" layoutY="141.0" strokeType="OUTSIDE" strokeWidth="0.0" text="full\_name" />  
 <TextField fx:id="driver\_full\_name" layoutX="107.0" layoutY="124.0" />  
 <Text layoutX="49.0" layoutY="176.0" strokeType="OUTSIDE" strokeWidth="0.0" text="allowed\_categories" />  
 <TextField fx:id="driver\_allowed\_categories" layoutX="107.0" layoutY="159.0" />  
 <Text layoutX="49.0" layoutY="211.0" strokeType="OUTSIDE" strokeWidth="0.0" text="working\_hours" />  
 <TextField fx:id="driver\_working\_hours" layoutX="107.0" layoutY="194.0" />  
 </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 моделі автобусів з місткістю" />  
 <TextField fx:id="capacity" layoutX="250.0" layoutY="25.0" prefHeight="25.0" prefWidth="64.0" />  
 <TextField fx:id="length" layoutX="125.0" layoutY="60" prefHeight="25.0" prefWidth="64.0" />  
 <Text layoutX="27.0" layoutY="80" strokeType="OUTSIDE" strokeWidth="0.0" text="Та довжиною" />  
 <Button layoutX="210.0" layoutY="60.0" mnemonicParsing="false" onAction="#interface1" text="Пошук" />  
 </children></AnchorPane>  
 </content>  
 </Tab>  
 </tabs>  
 </TabPane>  
 <Button layoutX="660.0" layoutY="144.0" mnemonicParsing="false" onAction="#delete" text="Видалити об'єкти із заданими параметрами" />  
 <Button layoutX="675.0" layoutY="74.0" mnemonicParsing="false" onAction="#findALL" text="Знайти об'єкти із заданими параметрами" />  
 <Button layoutX="667.0" layoutY="109.0" mnemonicParsing="false" onAction="#create" text="Створити об'єкт із заданими параметрами" />  
 <Button layoutX="700.0" layoutY="179.0" mnemonicParsing="false" onAction="#genRandom" text="Заповнити випадковими значеннями" />  
 <TextField fx:id="idChange" layoutX="500.0" layoutY="211.0" />  
 <Text layoutX="350.0" layoutY="228.0" strokeType="OUTSIDE" strokeWidth="0.0" text="ID для редагування" />  
 <Button layoutX="740.0" 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="500.0" layoutY="179.0" />  
 <Text layoutX="350.0" layoutY="199.0" strokeType="OUTSIDE" strokeWidth="0.0" text="Кількість полів" />  
 </children>  
</AnchorPane>