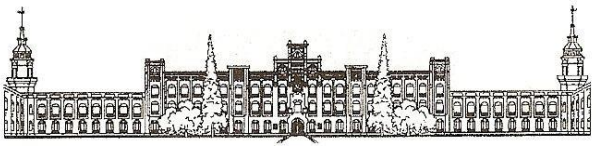


Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут»

Розрахункова графічна робота

з дисципліни «Бази даних»

«Створення додатку бази даних, орієнтованого на взаємодію з СУБД PostgreSQL»

Виконав студент групи: КВ-11

ПІБ: Гультяєв Дмитро Антонович

Telegram: @dimagultiaev

GitHub: Link

Перевірив: _____

Київ 2023

Постановка задачі:

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. 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).

Структура бази даних:

Сутності:

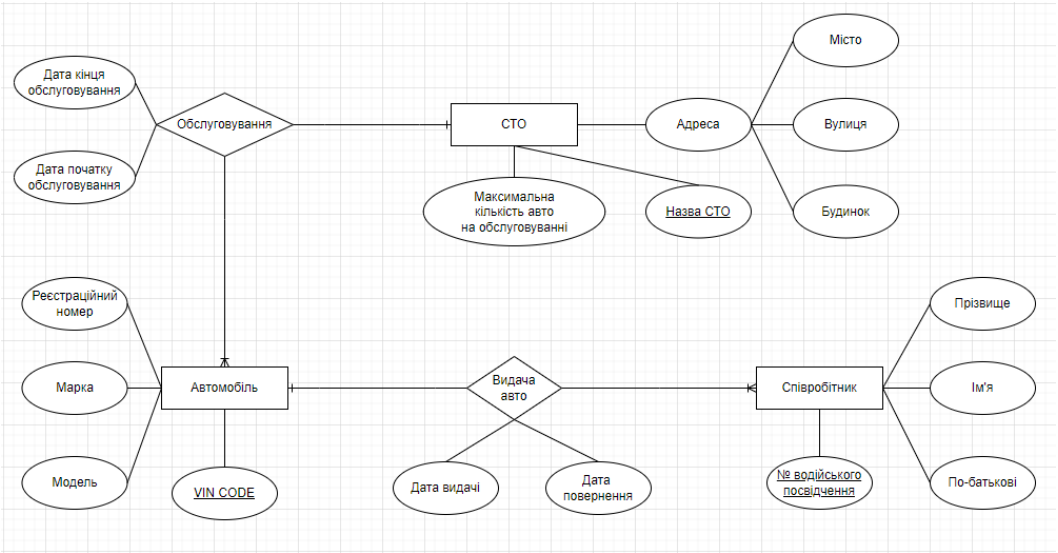
- Автомобіль (Cars) – ця сутність представляє автопарк компанії. У цій сутності ми маємо такі атрибути, як: vin, реєстраційний номер автомобіля, марку та модель.
- Співробітник (Employee) – ця сутність представляє працівників, які мають водійське посвідчення. Ці працівники можуть отримувати автомобілі в розпорядження з автопарку компанії. У цій сутності ми маємо такі атрибути, як: номер водійського посвідчення та ФІО.
- СТО (Service Station) – ця сутність представляє станцію технічного обслуговування, на якій обслуговується автопарк компанії. У цій сутності ми маємо такі атрибути, як: назва автосервісу, адреса та максимальна кількість автомобілей, яку автосервіс може обслуговувати одночасно.

Зв’язки:

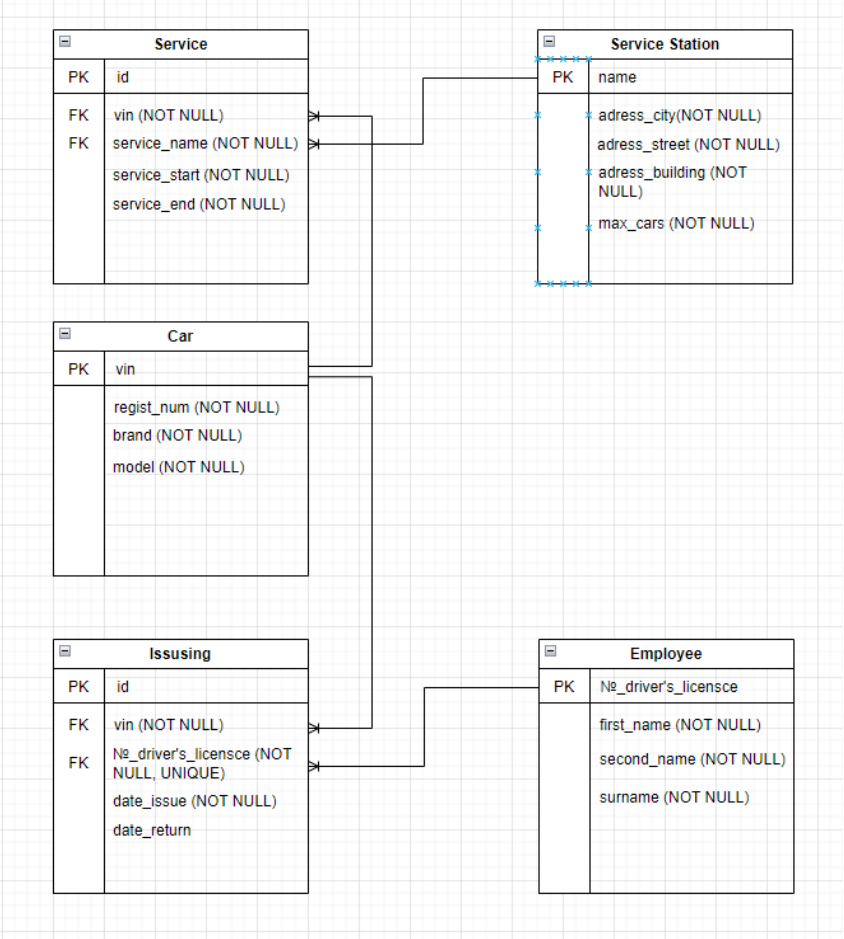
- 1:N – між автомобілем та співробітником (1 автомобіль може з часом переходити до різних співробітників).
- 1:N – між сервісом та автомобілем (1 сервіс обслуговує весь автопарк компанії).

Також я виділив такі зв’язки з атрибутами, як:

- Обслуговування (Service) – цей зв’язок поєднує сутність автомобіль та СТО
- Видача авто (Issuing) - цей зв’язок поєднує сутність автомобіль та співробітник



ER діаграма виконана за нотацією “Пташина лапка”

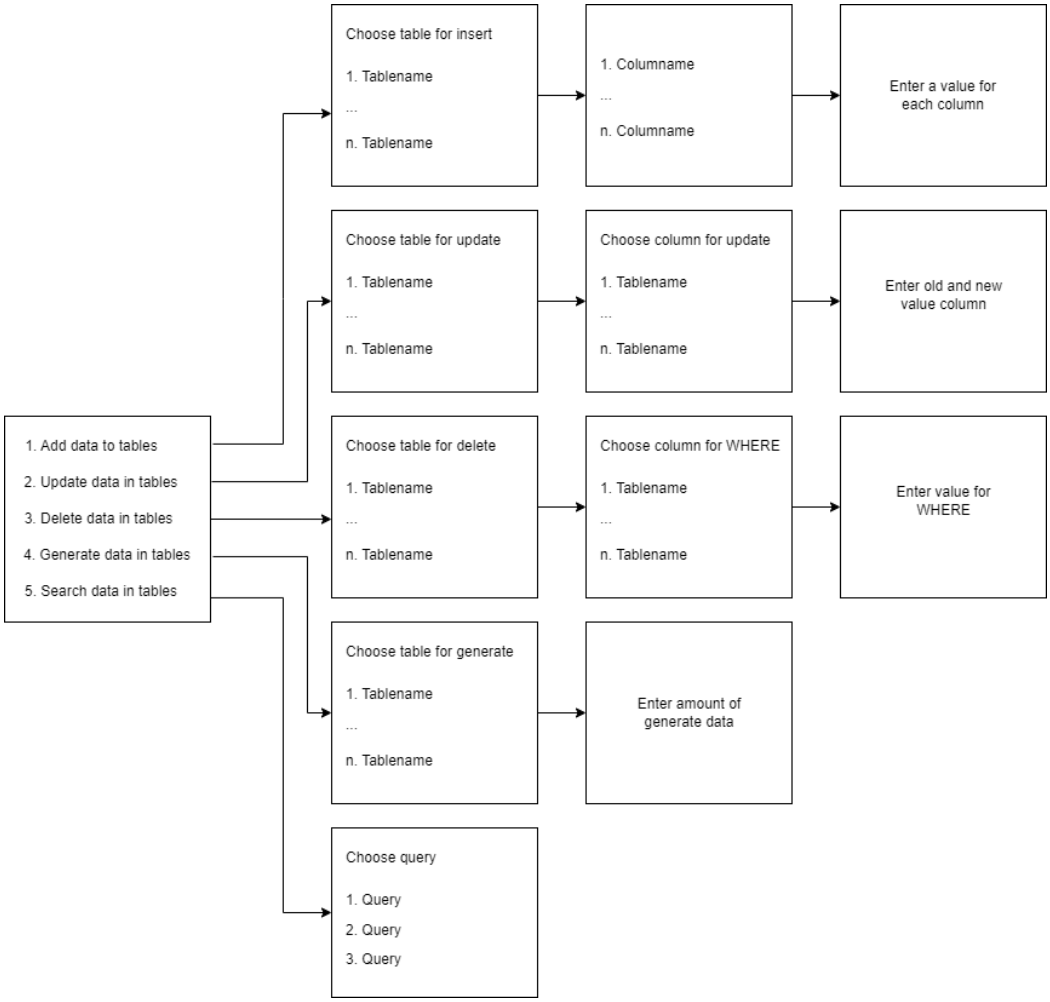


Структура бази даних

Опис меню користувача:

1) Add data to tables. Цей пункт призначений для додавання даних в таблиці. Після вибору цього пункту, користувач вибирає таблицю для вставки даних, а також вводить необхідні данні
2) Update data in tables. Цей пункт призначений для зміни даних в таблиці. Після вибору цього пункту, користувач вибирає таблицю, в якій треба змінити данні, вибирає потрібний аргумент таблиці для зміни, далі вводить старе та нове значення.
3) Delete data in tables. Цей пункт призначений для видалення даних в таблиці. Після вибору цього пункту, користувач вибирає таблицю, в якій треба видалити данні, вибирає потрібний аргумент таблиці для видалення, далі вводить значення цього елементу.

4) Generate data in tables. Цей пункт призначений для генерації даних в таблиці. Після вибору цього пункту, користувач вибирає таблицю, в якій треба згенерувати данні, а далі вводить кількість, яку необхідно згенерувати.
5) Search data in tables. Цей пункт призначений для реалізації пункту 3 завдання (пошук даних). Після вибору цього пункту, користувач вибирає один із 3 запитів, а далі вводить потрібні фільтри для цього запиту.



Графічна схема меню

```
1. Add data to tables
2. Update data in tables
3. Delete data in tables
4. Generate data in tables
5. Search data in tables

Choose your action:
```

Головне меню в консолі

Для створення програми була обрана мова Java. А для посилання запитів до бази даних був вибраний інструмент JDBC

Пункт №1:

Невдале вилучення записів:

```
Choose your action: 3

1. Service Station
2. Service
3. Employee
4. Cars
5. Issuing

Choose table: 4

1. vin
2. regist_num
3. brand
4. model
Choose column with which the data will be deleted: 1

Enter value:
vin - 1GCEK19R5WR144915
ПОМИЛКА: update або delete в таблиці "Cars" порушує обмеження зовнішнього ключа "Service_vin_fkey" таблиці "Service"
  Подробности: На ключ (vin)=(1GCEK19R5WR144915) все ще є посилання в таблиці "Service".
```

Помилка виникає через те, що ми намагаємося видалити автомобіль з таблиці “Cars”, при тому що він на даний момент обслуговується і знаходиться в таблиці Service

```
1. Add data to tables
2. Update data in tables
3. Delete data in tables
4. Generate data in tables
5. Search data in tables

Choose your action: 3

1. Service Station
2. Service
3. Employee
4. Cars
5. Issuing

Choose table: 4

1. vin
2. regist_num
3. brand
4. model
Choose column with which the data will be deleted: 1

Enter value:
vin - 5XYZU3LAXEG277189
ПОМИЛКА: update або delete в таблиці "Cars" порушує обмеження зовнішнього ключа "Issuing_vin_fkey" таблиці "Issuing"
  Подробности: На ключ (vin)=(5XYZU3LAXEG277189) все ще є посилання в таблиці "Issuing".
```

Помилка виникає через те, що ми намагаємося видалити автомобіль з таблиці “Cars”, при тому що він на даний момент знаходиться в користуванні і розміщається в таблиці Issuing.

Вдале вилучення записів:

```
1. Add data to tables
2. Update data in tables
3. Delete data in tables
4. Generate data in tables
5. Search data in tables

Choose your action: 3

1. Service Station
2. Service
3. Employee
4. Cars
5. Issuing

Choose table: 4

1. vin
2. regist_num
3. brand
4. model
Choose column with which the data will be deleted: 1

Enter value:
vin - 5XYZU3LAXEG277189
Delete information is successful!
```

Вигляд таблиці Issuing та Cars після видалення відповідно:

	id [PK] integer	vin character varying (17)	№_drivers_licensce [PK] character varying (9)	date_issue date	date_return date
1	13	XW8B2L2JS4Z2Y1144	1064388	2023-11-02	[null]
2	14	1C4NJPBA8ED597920	713098	2023-11-02	[null]
3	15	YBWAVY5BME8SZ9422	455274	2023-11-02	[null]
4	16	JN1BV7AP6EM693450	1220387	2023-11-10	[null]
5	17	4S2DF58X824686576	163811	2023-11-10	[null]
6	18	1GCEK19R5WR144915	188352	2023-11-10	[null]
7	20	1GCEC19H1SE187203	1088860	2023-11-12	[null]
8	21	KL8CD6S96DC587043	858071	2023-11-15	[null]
9	22	1NXBR32E86Z548589	1297942	2023-11-15	[null]

	vin [PK] character varying (17)	regist_num character varying (8)	brand character varying (20)	model character varying (20)
1	XW8B2L2JS4Z2Y1144	KA5691BT	Volkswagen	Passat
2	YBWAVY5BME8SZ9422	KA5693BT	Volkswagen	Jetta
3	1C4NJPBA8ED597920	KA8976BT	Seat	Ibiza
4	1FMZU72K44UA22337	KA9389BT	Volkswagen	Golf
5	JN1BV7AP6EM693450	KA8903BK	Audi	A6
6	1NXBR32E86Z548589	KA8903AT	Volkswagen	Golf
7	4S2DF58X824686576	KA8768AT	Skoda	Rapid
8	KL8CD6S96DC587043	KA9278AT	Skoda	Skala
9	4T1BE30K55U074717	KA0989AT	Skoda	Kamiq
10	JHMAP1140YT082781	KA8903AT	Skoda	Octavia
11	1GCEC19H1SE187203	KA9367AM	Kia	Rio
12	1FAHP34N59W100133	KA0093AM	Kia	Optima
13	1N4AL3AP6FC116356	KA1287AM	Kia	Optima
14	1GCEK19R5WR144915	KA0361BT	Mazda	3
15	2B3HD46F0VH724530	KA5387AM	Skoda	Octavia

Невдале додавання даних в таблицю:

```
Choose your action: 3

1. Service Station
2. Service
3. Employee
4. Cars
5. Issuing

Choose table: 4

1. vin
2. regist_num
3. brand
4. model
Choose column with which the data will be deleted: 1

Enter value:
vin - 1GCEK19R5WR144915
ПОМИЛКА: update або delete в таблиці "Cars" порушує обмеження зовнішнього ключа "Service_vin_fkey" таблиці "Service"
  Подробности: На ключ (vin)=(1GCEK19R5WR144915) все ще є посилання в таблиці "Service".
```

Ця помилка пов’язана з тим, що ми намагаємося видати автомобіль в користування (додати відповідний запис в таблицю Issuing), при цьому не маючи відповідного автомобіля в автопарку (таблиці “Cars”).

Вдале додавання даних в таблицю:

```
1. Add data to tables
2. Update data in tables
3. Delete data in tables
4. Generate data in tables
5. Search data in tables

Choose your action: 1

1. Service
2. Employee
3. Cars
4. Issuing

Choose table: 4

Enter data for variables:
date_issue - 2023-11-20
vin - 1FAHP34N59W100133
№_drivers_licensce - 332249
Add information is successful!
```

Пункт №2:

Генерація 100 000 працівників для таблиці Employee:

	№_drivers_license [PK] character varying (9)	first_name character varying (20)	last_name character varying (20)	surname character varying (20)
1	1313110	Dmytro	Federenko	Romanovich
2	1193302	Petro	Popov	Viltorovich
3	713248	Petro	Schevchuk	Andriyovich
4	907855	Sergiy	Moroz	Alekseyovich
5	12766	Stepan	Kravchenko	Vladislavovich
6	413821	Anton	Gavrulyk	Olegovich
7	713098	Petro	Vasilenko	Romanovich
8	1220387	Viktor	Kostenko	Fedirovich
9	1064388	Pavlo	Kostyk	Fedirovich
10	342857	Nikita	Pavlenko	Andriyovich
11	163811	Vadim	Savchenko	Kyrilovich
12	302197	Anton	Vlasenko	Vadimovich
13	712975	Petro	Moroz	Denisovich
14	373210	Kyrilo	Tkachuk	Antonovich
15	579050	Aleksey	Tkachuk	Alekseyovich
16	337136	Dmytro	Petrenko	Stepanovich
17	1249865	Aleksandr	Homenko	Ruslanovich
18	1222030	Denis	Federenko	Antonovich
19	1094298	Denis	Vasilenko	Dmytrovich
20	1398947	Nikita	Kostenko	Ivanovich
21	1343677	Yuriy	Vlasenko	Andriyovich
22	1172369	Viktor	Oliynik	Fedirovich
23	86729	Nikita	Oliynik	Stepanovich
24	1019454	Sergiy	Rudenko	Ruslanovich
25	792464	Pavlo	Mazur	Andriyovich
26	455274	Oleg	Tarasenko	Sergiyvich
27	1525781	Denis	Gavrulyk	Denisovich
28	1287078	Nikita	Radchenko	Alekseyovich
29	1121743	Aleksandr	Panchenko	Bogdanovich
30	1037010	Fedir	Tkachuk	Olegovich

Total rows: 1000 of 100000 | Query complete 00:00:00.246

SQL запит для генерації працівників:

```
insert into "Employee" (№_drivers_license, first_name, last_name, surname)
select
row_number() OVER () as №_drivers_license,
first_name,
last_name,
surname
from generate_series(1, 100) as №_drivers_license
cross join
(select
unnest(array['Ivan','Dmytro','Aleksandr','Aleksey','Vadim','Viktor','Kyrilo',
'Denis','Nikita','Oleg','Pavlo','Petro','Roman','Ruslan','Sergiy','Stepan','F
edir','Anton','Andriy','Yuriy']) as first_name) as f
cross join
(select
unnest(array['Melnyk','Schevchenko','Kovalenko','Bondarenko','Tkaschenko','Kr
avchenko','Kovalchuk','Schevchuk','Oliynik','Tkachuk','Savchenko','Bondar','M
archenko','Rudenko','Moroz','Petrenko','Pavlenko','Vasilenko','Levchenko','Ka
rpenko','Gavrulyk','Popov','Panchenko','Mazur','Homenko','Goncharenko','Koste
nko','Kostyk','Kozak','Federenko','Kovtun','Bilous','Nesterenko','Tarasenko',
```

```
'Vovk','Zhuk','Vlasenko','Radchenko','Voloshyn','Velyshko']) as last_name) as
l
cross join
(select
unnest(array['Alexandrovich','Sergiyvich','Petrovich','Bogdanovich','Denisovi
ch','Olegovich','Ruslanovich','Romanovich','Stepanovich','Fedirovich','Antono
vich','Andriyovich','Viltorovich','Ivanovich','Vadimovich','Kyrilovich','Alek
seyovich','Dmytrovich','Maksymovich','Vladislavovich']) as surname) as s
order by random()
LIMIT 100000;
```

Пункт №3:

Перший запит:

```
1. Add data to tables
2. Update data in tables
3. Delete data in tables
4. Generate data in tables
5. Search data in tables

Choose your action: 5

1. First query
2. Second query
3. Third query

Choose query: 1

Enter value for query: 2023-11-12
Kia          | Karpenko     | Dmytro       | 2023-11-12

Query time: 12 mils
```

Другий запит:

```
1. Add data to tables
2. Update data in tables
3. Delete data in tables
4. Generate data in tables
5. Search data in tables

Choose your action: 5

1. First query
2. Second query
3. Third query

Choose query: 2

Enter value for query: Skoda
Skoda        | Kovtun       | Roman        | 2023-11-15
Skoda        | Savchenko    | Vadim        | 2023-11-10

Query time: 14 mils
```

Третій запит:

```
1. Add data to tables
2. Update data in tables
3. Delete data in tables
4. Generate data in tables
5. Search data in tables

Choose your action: 5

1. First query
2. Second query
3. Third query

Choose query: 3

Enter value for query: Skoda
Skoda        | Octavia      | 2023-11-05   | 2023-11-06
Skoda        | Skala        | 2023-10-09   | 2023-10-10
Skoda        | Skala        | 2023-10-20   | 2023-10-21

Query time: 13 mils
```

Перший запит:

```
SELECT "Cars".brand, "Employee".last_name, "Employee".first_name,
"Issuing".date_issue from "Issuing"
JOIN "Cars" ON "Issuing".vin = "Cars".vin
JOIN "Employee" ON "Issuing".№_drivers_licensce =
"Employee".№_drivers_license
WHERE "Issuing".date_issue = '2023-11-12'
GROUP BY "Cars".brand, "Employee".first_name, "Employee".last_name,
"Issuing".date_issue
```

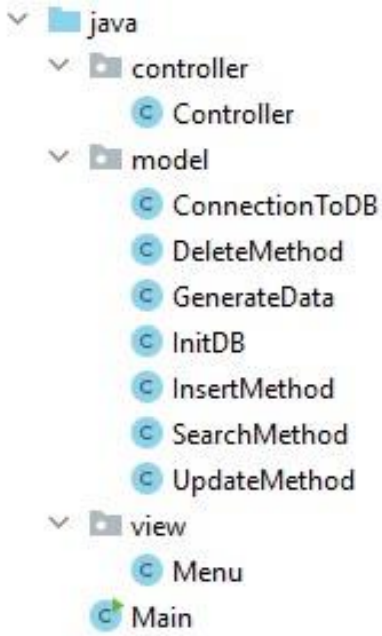
Другий запит:

```
SELECT "Cars".brand, "Employee".last_name, "Employee".first_name,
"Issuing".date_issue from "Issuing"
JOIN "Cars" ON "Issuing".vin = "Cars".vin
JOIN "Employee" ON "Issuing".№_drivers_licensce =
"Employee".№_drivers_license
WHERE "Cars".brand = 'Skoda'
GROUP BY "Cars".brand, "Employee".first_name, "Employee".last_name,
"Issuing".date_issue
```

Третій запит:

```
SELECT "Cars".brand, "Cars".model, "Service".service_start_date,
"Service".service_end_date FROM "Service"
JOIN "Cars" ON "Cars".vin = "Service".vin
WHERE "Cars".brand = 'Skoda'
GROUP BY "Cars".brand, "Cars".model, "Service".service_start_date,
"Service".service_end_date
```

Пункт №4:



Файлова структура програми

- ConnectionToDB – в цьому файлі створюється підключення до бази даних. Для цього використовується адреса БД, ім’я користувача та пароль.

```
public class ConnectionToDB {
    9 usages
    public static Connection connectionToDB(){
        Connection connection = null;

        try(FileInputStream file = new FileInputStream( name: "src/main/resources/config.properties")) {
            Properties properties = new Properties();
            properties.load(file);
            connection = DriverManager.getConnection(properties.getProperty("URL"), properties.getProperty("USERNAME"), properties.getProperty("PASSWORD"));

        } catch (IOException e) {
            throw new RuntimeException(e);
        } catch (SQLException e) {
            System.out.println("Connection Failed : " + e.getMessage());
        }

        return connection;
    }
}
```

- DeleteMethod – в цьому файлі прописаний метод для видалення даних з таблиці. Також, якщо ми видаляємо данні з таблиці Issuing чи Service, то перед цим перевіряється чи не використовується автомобіль або чи не стоїть

автомобіль на даний момент на обслуговуванні.

```
public static void delete(String tableName, String column, String value) throws SQLException {
    try (Connection connection = ConnectionToDB.connectionToDB();
         Statement statement = connection.createStatement()) {

        if(tableName.equals("Issuing")){
            ResultSet resultSet = statement.executeQuery( sql: "SELECT vin, date_return FROM \"" + tableName +"\" WHERE " + column + " = '" + value + "'");
            String date_return = null;

            while(resultSet.next()){
                if(tableName.equals("Issuing")){
                    date_return = resultSet.getString( columnLabel: "date_return");
                } else {
                    date_return = resultSet.getString( columnLabel: "service_end_date");
                }
            }

            if(date_return == null){
                if(tableName.equals("Issuing")){
                    throw new SQLException("This car is currently in use! Try to delete another car!");
                } else {
                    throw new SQLException("This car is currently in service! Try to delete another car!");
                }
            }

        }

        String query = "DELETE FROM \"" + tableName + "\" WHERE " + column + " = " + "'" + value + "'";
        statement.executeUpdate(query);
    }
}
```

- GenerateMethod - в цьому файлі прописаний метод для генерування даних в таблиці.

```
public class GenerateData {
    1 usage
    public static void generateForEmployee(int amount) throws SQLException {
        try(Connection connection = ConnectionToDB.connectionToDB();
            Statement statement = connection.createStatement()) {

            String query = "insert into \"Employee\" (№_drivers_license, first_name, last_name, surname)\n" +
                    "select \n" +
                    "row_number() OVER () as №_drivers_license, \n" +
                    "first_name, \n" +
                    "last_name, \n" +
                    "surname\n" +
                    "from generate_series(1, 100) as №_drivers_license\n" +
                    "cross join\n" +
                    "(select unnest(array['Ivan','Dmytro','Aleksandr','Aleksey','Vadim','Viktor','Kyrilo','Denis','Nikita','Oleg','Pavlo','Petro','Roman','Ruslan','Sergiy','
                    "cross join\n" +
                    "(select unnest(array['Melnyk','Schevchenko','Kovalenko','Bondarenko','Tkaschenko','Kravchenko','Kovalchuk','Schevchuk','Oliynik','Tkachuk','Savchenko','
                    "cross join\n" +
                    "(select unnest(array['Alexandrovich','Sergiyvich','Petrovich','Bogdanovich','Denisovich','Olegovich','Ruslanovich','Romanovich','Stepanovich','Fedirovich
                    "order by random()\n" +
                    "LIMIT " + amount;
            statement.executeUpdate(query);
        }
    }
}
```

- InitDB - в цьому файлі прописані два методи: для отримання множини таблиць, для отримання множини колонок певної таблиці.

```
public class InitDB {
    4 usages
    public static List<String> getTables(){
        List<String> tableNames = new ArrayList<>();

        try(Connection connection = ConnectionToDB.connectionToDB();
            Statement statement = connection.createStatement();
            ResultSet tableNamesSet = statement.executeQuery( sql: "SELECT table_name FROM information_schema.columns WHERE table_schema='public' GROUP BY table_name")) {

            while(tableNamesSet.next()){
                tableNames.add(tableNamesSet.getString( columnLabel: "table_name"));
            }

        } catch (SQLException e) {
            throw new RuntimeException(e);
        }

        return tableNames;
    }

    3 usages
    public static List<String> getTableColumns(String tableName){
        List<String> tableColumns = new ArrayList<>();

        try(Connection connection = ConnectionToDB.connectionToDB();
            Statement statement = connection.createStatement();
            ResultSet columnsSet = statement.executeQuery( sql: "SELECT column_name FROM information_schema.columns WHERE table_name='" + tableName + "'")) {

            while(columnsSet.next()){
                tableColumns.add(columnsSet.getString( columnLabel: "column_name"));
            }

        } catch (SQLException e) {
            throw new RuntimeException(e);
        }

        return tableColumns;
    }
}
```

- InsertMethod – в цьому файлі прописана функція для вставки значень в таблицю, а також для генерації самого запиту Insert

```
private static String createInsertQuery(String tableName, Map<String, String> data){
    StringBuilder insertQuery = new StringBuilder("INSERT INTO \"" + tableName + "\" (");
    Set<String> dataKeySet = data.keySet();
    Iterator<String> iterator = dataKeySet.iterator();

    for(int i = 0; i < dataKeySet.size(); i++){
        if(i != dataKeySet.size() - 1){
            insertQuery.append(iterator.next()).append(",");
        } else {
            insertQuery.append(iterator.next()).append(")");
        }
    }

    insertQuery.append(" VALUES (");

    for(int i = 0; i < dataKeySet.size(); i++){
        if(i != dataKeySet.size() - 1){
            insertQuery.append("?").append(",");
        } else {
            insertQuery.append("?)");
        }
    }

    return String.valueOf(insertQuery);
}

1 usage
public static void insert(String tableName, Map<String, String> data) throws SQLException {
    try(Connection connection = ConnectionToDB.connectionToDB(); PreparedStatement preparedStatement = connection.prepareStatement(createInsertQuery(tableName, data))){
        Set<String> dataKeySet = data.keySet();
        Iterator<String> iterator = dataKeySet.iterator();

        if(tableName.equals("Issuing")){
            Statement statement = connection.createStatement();
            ResultSet resultSet = statement.executeQuery( sql: "SELECT vin, date_return FROM \"Issuing\"");
            Map<String, String> map = new HashMap<>();

            while(resultSet.next()){
                map.put(resultSet.getString( columnLabel: "vin"), resultSet.getString( columnLabel: "date_return"));
            }

            if(map.containsKey(data.get("vin")) && map.get(data.get("vin")) == null){
                throw new SQLException("This car is currently in use! Try another car!");
            }

            statement.close();
            resultSet.close();
        }

        for(int i = 0; i < data.size(); i++){
            String column = iterator.next();
            if(column.equals("date_issue") || column.equals("date_return") || column.equals("service_start_date") || column.equals("service_end_date")){
                preparedStatement.setDate( parameterIndex: i+1, Date.valueOf(data.get(column)));
            } else {
                preparedStatement.setString( parameterIndex: i+1, data.get(column));
            }
        }

        preparedStatement.executeUpdate();
    }
```

- SearchMethod – в цьому файлі прописані 3 функції-запиту для пошуку інформації в базі даних.

```
public static void firstSearchMethod(String data) throws SQLException{
    try(Connection connection = ConnectionToDB.connectionToDB(); Statement statement = connection.createStatement()){

        long startTime = System.currentTimeMillis();

        ResultSet resultSet = statement.executeQuery( sql: "SELECT \"Cars\".brand, \"Employee\".last_name, \"Employee\".first_name, \"Issuing\".date_issue from \"Issuing\"\
                "JOIN \"Cars\" ON \"Issuing\".vin = \"Cars\".vin\n" +
                "JOIN \"Employee\" ON \"Issuing\".№_drivers_licensce = \"Employee\".№_drivers_license\n" +
                "WHERE \"Issuing\".date_issue = '" + data + "'\n" +
                "GROUP BY \"Cars\".brand, \"Employee\".first_name, \"Employee\".last_name, \"Issuing\".date_issue");

        long endTime = System.currentTimeMillis();

        while(resultSet.next()){
            String brand = resultSet.getString( columnLabel: "brand");
            String last_name = resultSet.getString( columnLabel: "last_name");
            String first_name = resultSet.getString( columnLabel: "first_name");
            String date_issue = resultSet.getString( columnLabel: "date_issue");

            System.out.printf("%-12s | %-12s | %-12s | %-12s \n", brand, last_name, first_name, date_issue);
        }
        System.out.println("\nQuery time: " + (endTime -startTime) + " mils");
    }
}
```

```
public static void secondSearchMethod(String data) throws SQLException{
    try(Connection connection = ConnectionToDB.connectionToDB(); Statement statement = connection.createStatement()){

        long startTime = System.currentTimeMillis();

        ResultSet resultSet = statement.executeQuery( sql: "SELECT \"Cars\".brand, \"Employee\".last_name, \"Employee\".first_name, \"Issuing\".date_issue from \"Issuing\"\n" +
                "JOIN \"Cars\" ON \"Issuing\".vin = \"Cars\".vin\n" +
                "JOIN \"Employee\" ON \"Issuing\".№_drivers_licensce = \"Employee\".№_drivers_license\n" +
                "WHERE \"Cars\".brand = '" + data + "'\n" +
                "GROUP BY \"Cars\".brand, \"Employee\".first_name, \"Employee\".last_name, \"Issuing\".date_issue");

        long endTime = System.currentTimeMillis();

        while(resultSet.next()){
            String brand = resultSet.getString( columnLabel: "brand");
            String last_name = resultSet.getString( columnLabel: "last_name");
            String first_name = resultSet.getString( columnLabel: "first_name");
            String date_issue = resultSet.getString( columnLabel: "date_issue");

            System.out.printf("%-12s | %-12s | %-12s | %-12s \n", brand, last_name, first_name, date_issue);
        }
        System.out.println("\nQuery time: " + (endTime -startTime) + " mils");
    }
}

1 usage
public static void thirdSearchMethod(String data) throws SQLException{
    try(Connection connection = ConnectionToDB.connectionToDB(); Statement statement = connection.createStatement()){

        long startTime = System.currentTimeMillis();

        ResultSet resultSet = statement.executeQuery( sql: "SELECT \"Cars\".brand, \"Cars\".model, \"Service\".service_start_date, \"Service\".service_end_date FROM \"Service\"\n" +
                "JOIN \"Cars\" ON \"Cars\".vin = \"Service\".vin\n" +
                "WHERE \"Cars\".brand = '" + data + "'\n" +
                "GROUP BY \"Cars\".brand, \"Cars\".model, \"Service\".service_start_date, \"Service\".service_end_date");

        long endTime = System.currentTimeMillis();

        while(resultSet.next()){
            String brand = resultSet.getString( columnLabel: "brand");
            String model = resultSet.getString( columnLabel: "model");
            String service_start_date = resultSet.getString( columnLabel: "service_start_date");
            String service_end_date = resultSet.getString( columnLabel: "service_end_date");

            System.out.printf("%-12s | %-12s | %-12s | %-12s \n", brand, model, service_start_date, service_end_date);
        }
        System.out.println("\nQuery time: " + (endTime -startTime) + " mils");
    }
}
}
```

- UpdateMethod – в цьому файлі прописана функція для оновлення даних в таблиці.

```
public class UpdateMethod {
    1 usage
    public static void update(String tableName, String column, String newValue, String columnForWhere, String whereValue) throws SQLException {
        try(Connection connection = ConnectionToDB.connectionToDB();
            Statement statement = connection.createStatement()){
            String query = "UPDATE \"" + tableName + "\" SET " + column + " = '" + newValue + "' WHERE " + columnForWhere + " = '" + whereValue +"'";
            statement.executeUpdate(query);
        }
    }
}
```