Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Кафедра электронных вычислительных машин

Лабораторная работа №6

«Создание приложения для базы данных»

Больница

Выполнил: Проверил:

студент группы 150502 Игнатович А.О.

Былинский Н.С.

Минск 2024

**1 ИСХОДНОЕ ЗАДАНИЕ**

Создание прикладной программы для работы с базой данных и выполняющей заданные транзакции. Можно использовать любую среду и язык программирования.

**2 ВЫПОЛНЕНИЕ РАБОТЫ**

**2.1 Используемые технологии**

Стек используемых технологий:

Javafx – графическая часть приложения

Java(JDBC) – библиотека для работы с базой данных

PostgreSQL – база данных

Описание:

Для взаимодействия с PostgreSQL из Java используется библиотека Java.sql\*. Она обеспечивает возможность работы с базой данных PostgreSQL, включая выполнение SQL-запросов и получение результатов операций.

**2.2 Разработка программы**

Главное меню, в котором пользователь выбирает, что будет делать с базой данных:

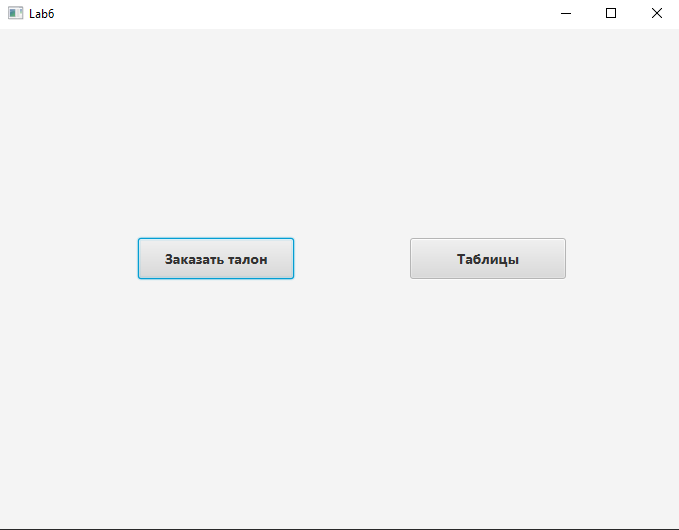


Рисунок 2.1 – Главное меню

При выборе пункта меню “Таблица” происходит переход в окно, редактирования и просмотра таблиц.

При выборе «Заказать талон переходим в окно для заполнения данных при заказе талона.

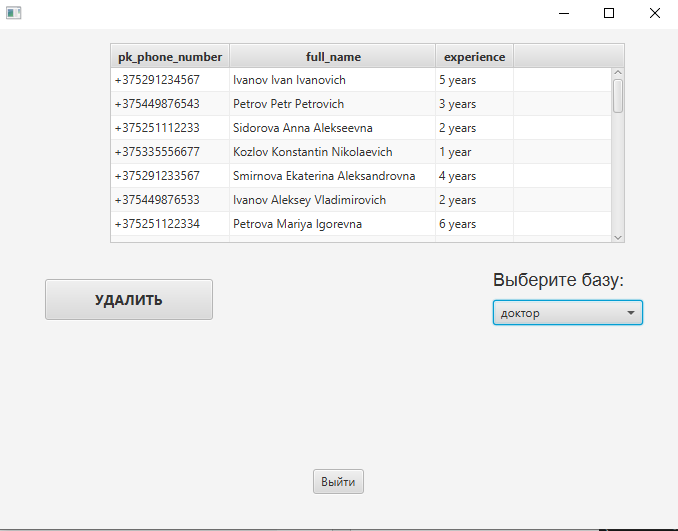


Рисунок 2.2 – Содержимое таблицы “doctor”

Код для первого действия:

public List<ArrayList<String>> getData(){  
 List<ArrayList<String>> data = new ArrayList<>();  
 try (Connection connection = DriverManager.*getConnection*("jdbc:postgresql://localhost:5432/postgres", "postgres", "7969930n");  
 Statement statement = connection.createStatement()) {  
 ResultSet resultSet = statement.executeQuery("select \* from " + dataBase);  
 int columnCount = resultSet.getMetaData().getColumnCount();  
 while (resultSet.next()) {  
 ArrayList<String> row = new ArrayList<>();  
 for (int i = 1; i <= columnCount; i++) {  
 row.add(resultSet.getString(i));  
 }  
 data.add(row);  
 }  
 } catch (SQLException e) {  
 throw new RuntimeException(e);  
 }

При выборе «Удалить» мы обязаны выбрать поле в таблице которое хотим удалить.

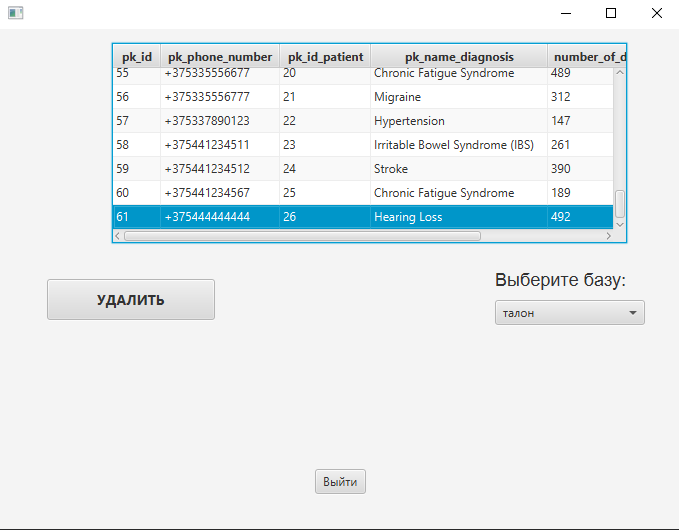


Рисунок 2.4 – Удаление строки “ticket”

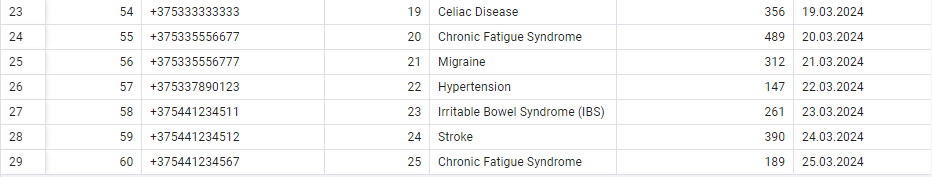


Рисунок 2.5 – Таблица после удаления строки

Код для удаления:

public void DeleteData(String id){  
 try (Connection connection = DriverManager.*getConnection*("jdbc:postgresql://localhost:5432/postgres", "postgres", "7969930n");  
 Statement statement = connection.createStatement()) {  
 System.*out*.println(getColumNames().get(0));  
 statement.executeUpdate("delete from "+dataBase+ " where " +getColumNames().get(0)+" ='"+id+"'");  
 } catch (SQLException e) {  
 throw new RuntimeException(e);  
 }  
}

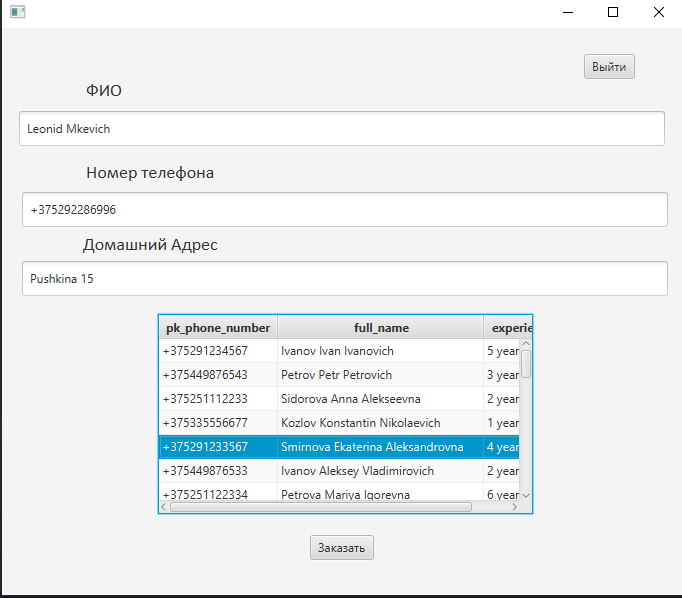
 При выборе пункта “Заказать талон” вводим данные и выбираем врача.

Рисунок 2.6 – Добавление строки в “ticket” и “patient”

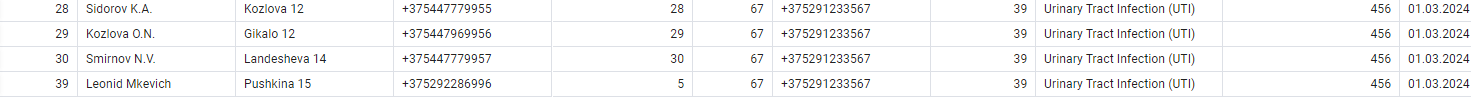


Рисунок 2.7 – Таблицы “ticket” и “patient” после добавления

Код данной функции:

public void addDataBase(String name, String phoneNumber, String address, String doctor\_id){  
 try (Connection connection = DriverManager.*getConnection*("jdbc:postgresql://localhost:5432/postgres", "postgres","7969930n");  
 Statement statement = connection.createStatement()) {  
 System.*out*.println("connect");  
 statement.executeUpdate("insert into patient values (DEFAULT,'"+name+"','"+address+"','"+phoneNumber+"',5)");  
 statement.executeUpdate("insert into ticket values (DEFAULT, '"+doctor\_id+"',( SELECT pk\_id FROM patient ORDER BY pk\_id DESC LIMIT 1), 'Urinary Tract Infection (UTI)', 456, '01.03.2024')");  
 } catch (SQLException e) {  
 throw new RuntimeException(e);  
 }  
}