|  |
| --- |
|  |
| МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** |

**Институт кибербезопасности и цифровых технологий**

Дисциплина

«Специальные информационные технологии

в правоохранительной деятельности»

**Отчет**

по практической работе №7-8 на тему:

**«Работа со сторонними базами данными.**

**Построение и оптимизация»**

Выполнил:

Студент 3 курса

Группа ББСО-03-20

Шифр 20Б0410

Мешков Артем Владимирович

Шифр 20Б0409

Мельников Николай Алексеевич

Москва, 2023

**Задание**

На основе практических работ 2-5 создать графический интерфейс для

базы данных со следующим функционалом:

1. Регистрация новых пользователей с выбором ролей (пароли

пользователей должны шифроваться)

2. Отдельное отображение доступной информации для каждой из

ролей в зависимости от привилегий и доступа из лабораторных

работ 2-5

3. Автоматический просмотр задания (в зависимости от функционала

ролей из 2-5 лабораторных работ)

5. Поиск клиентов

6. Отчетность по сотрудникам

7. Отдельная отчетность по заданиям

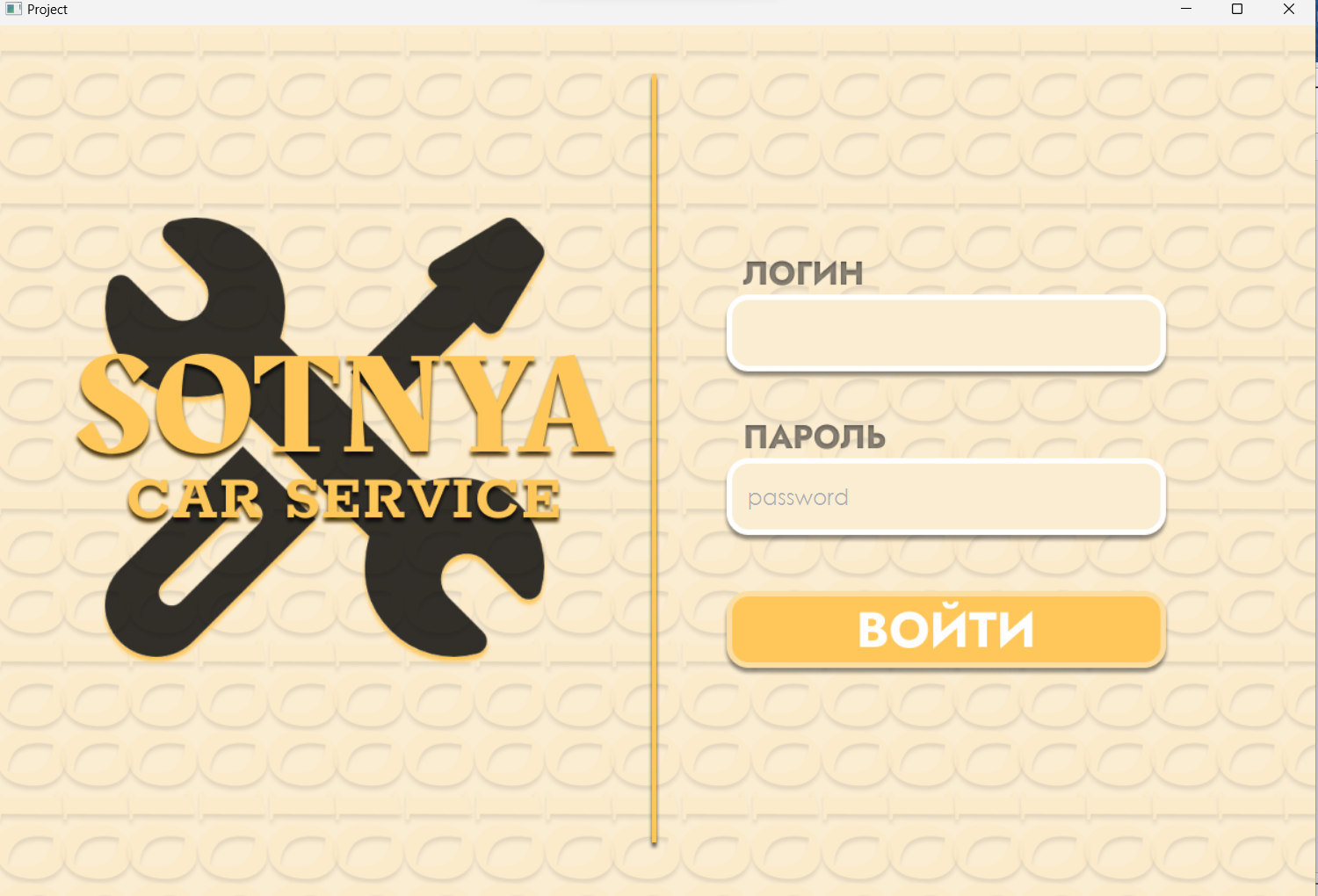
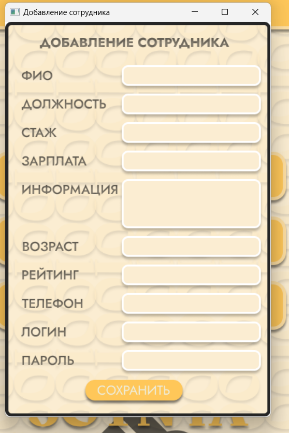


Рисунок Окно авторизации

Рисунок 2 Внешний вид администратора

Добавление нового сотрудника возможна только от администратора

Рисунок 3 Добавление нового сотрудника

При нажатии на кнопку сохранить запускается sql скрипт, который добавляет сотрудника в таблицу employee, user и создает роль этому сотруднику в соответствии с его должностью

public static void addEmployee(String name, String position, String exp, String sal, String inf,String age,String score, String num, String login, String passSalt, String pass) throws SQLException, FileNotFoundException, ClassNotFoundException {  
 *statmt*.execute("INSERT INTO public.employee(\n" +  
 "\temployee\_name, employee\_exp, employee\_sal, employee\_inf, employee\_age, employee\_score, employee\_num, employee\_post)\n" +  
 "\tVALUES ('" + name +"', '" + exp + "', '" + sal + "', '" + inf + "', " + age + ", '" + score + "', " + num + ", '" + position +"');");  
 String employee\_id = *getEmployee\_byName*(name);  
 *statmt*.execute("INSERT INTO public.\"user\"(\n" +  
 "\temployee\_id, login, password)\n" +  
 "\tVALUES ('" + employee\_id + "', '" + login + "', '" + passSalt + "');");  
 *statmt*.execute("CREATE ROLE \""+ login +"\""+  
 "\tLOGIN\n" +  
 "\tNOSUPERUSER\n" +  
 "\tNOCREATEDB\n" +  
 "\tNOCREATEROLE\n" +  
 "\tINHERIT\n" +  
 "\tNOREPLICATION\n" +  
 "\tCONNECTION LIMIT -1\n" +  
 "\tPASSWORD '" + pass + "';");  
 String lvl="Staff";  
 if(position.equals("Администратор"))  
 lvl = "Administrator";  
 else if(position.equals("Аналитик"))  
 lvl = "Analyst";  
 else if(position.equals("Управляющий"))  
 lvl = "Local\_manager";  
 else if(position.equals("Менеджер"))  
 lvl = "Manager";  
 *statmt*.execute("GRANT \""+lvl+"\" TO \""+ login+"\" WITH ADMIN OPTION;");  
}

Стоит заметить, что при создании роли пароль передается в чисто виде (т.е. без хэширования и соли), а при занесении в таблицу user при помощи алгоритма sha512 с использованием соли. Защита от sql инъекции осуществляется при помощи jdbc. Для этого была добавлена соответсвующий драйвер, который работает с PostgreSQL.

Также администратор может редактировать сотрудников

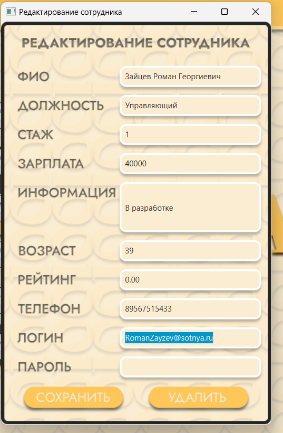


Рисунок Редактирование сотрудника

При редактировании сотрудника, пароль в целях безопасности, не выводится. Если необходимо его заменить, то в соответствующее поле вносится новый пароль, который меняется сразу и в роли этого пользователя и заноситься в хэшированном виде в таблицу user, если пароль не изменяется, то никаких изменений не происходит. При нажатии на кнопку сохранить, происходит update данных, независимо от того, поменялось ли что-то.

Функционал у просто сотрудника это добавить клиента и добавить посещение. А также просмотр записей



 Рисунок 5 Внешний вид персонал

Рисунок 6 Просмотр клиентов

Рисунок 7 Демонстрация работы "Поиск по ФИО"

При вводе текста и нажатии на кнопку «Найти» вызывается функция, которая в двумерном массиве возвращает клиентов, удовлетворяющих поиску

public static String[][] getClient\_byNAME(String name) throws SQLException {  
  
 *data\_statmt* = *data\_connection*.createStatement();  
 String id = DataBase.*getLastClient*();  
 int kol = Integer.*parseInt*(id);  
 String[][] mas = new String[4][kol];  
 *data\_resSet* = *data\_statmt*.executeQuery("SELECT \* FROM client \n" +  
 "WHERE position('" + name + "' in client\_name)>0;");  
 int i = 0;  
 if(*data\_resSet* == null)  
 System.*out*.println(kol);  
 while(*data\_resSet*.next()) {  
 System.*out*.println("i/"+i);  
 mas[0][i] = *data\_resSet*.getString("client\_id");  
 mas[1][i] = *data\_resSet*.getString("client\_name");  
 mas[2][i] = *data\_resSet*.getString("client\_status");  
 mas[3][i] = *data\_resSet*.getString("client\_bonus");  
 i++;  
 }  
 return mas;  
}

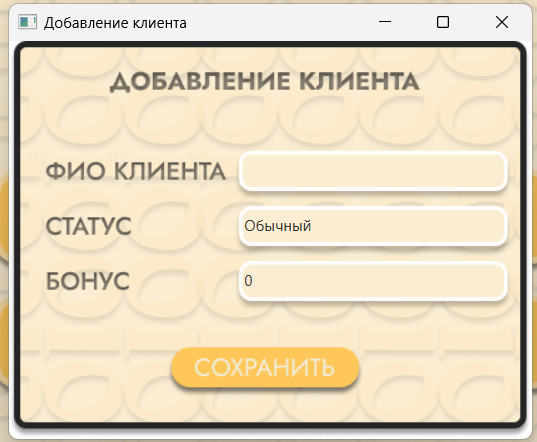
При нажатии на кнопку добавить клиента, появляется окно, в котором можно добавить клиента. При последующем нажатии на кнопку «Сохранить» данный клиент вноситься в базу путем вызова функции addClient

Рисунок 8 Добавление клиента

public static void addClient(String name) throws SQLException {  
 *data\_statmt*.execute("INSERT INTO public.client(\n" +  
 "\tclient\_name)\n" +  
 "\tVALUES ('" + name + "');");  
}

}

Также у персонала есть возможность добавить посещение путем нажатия на кнопку «добавить посещение» напротив конкретного клиента. После нажатия появляется новое окно, в котором при помощи выпадающего списка можно выбрать заведение и услугу.

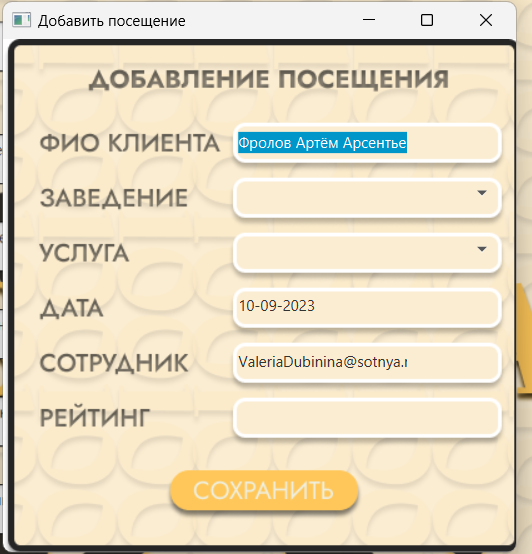


Рисунок 10 Просмотр посещений

Рисунок 9 Добавление посещения

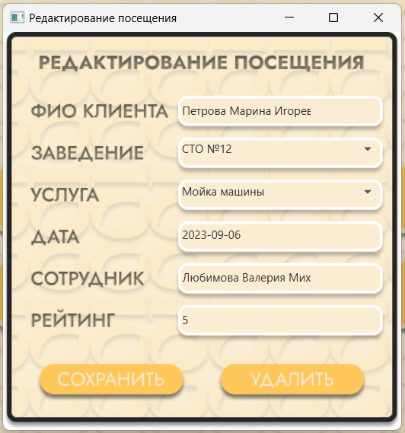
Менеджер по условию имеет такой же функционал как у персонала, но с возможностью редактирования и удаления

Рисунок 11 Редактирование посещения

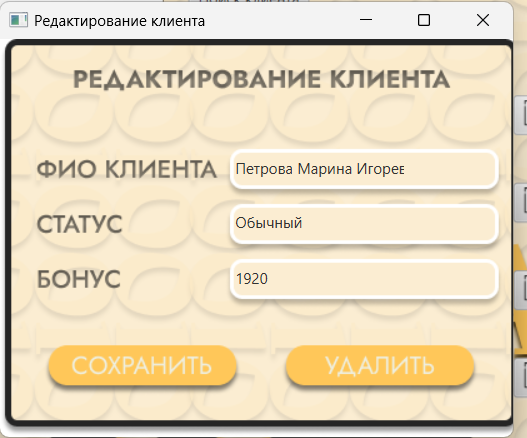
При нажатии на иконку редактирования открывается соответствующее окно, в котором уже появляется кнопка «удалить»

Рисунок 12 Редактирование клиента

При нажатии на кнопки «удалить» вызывается функция delClient для удаления клиента. Удаление происходит каскадно, т.е. сначало удаляется из таблицы visit, а потом уде из таблицы client.

public static void delClient(String t1) throws SQLException {  
 *statmt*.execute("DELETE FROM visit "+  
 " where client\_id = " + t1+";");  
 *statmt*.execute("DELETE FROM client "+  
 " where client\_id = " + t1+";");  
}

При нажатии удалить в окне редактирования посещения вызывается функция delVisit.

public static void delVisit(String t1) throws SQLException {  
 *statmt*.execute("DELETE FROM visit "+  
 " where visit\_id = " + t1+";");  
}

Аналитик по условию может просматривать все страницы

Рисунок 13 Главный экран у аналитика



А также смотреть отчеты, которые расположены в соответствующем месте. Отчеты при помощи скрипта сохраняются на жестком диске.



Рисунок 14 Информация о своих сотрудниках

Последним остался управляющий, в его полномочиях редактировать информацию о заведениях и сотрудниках.

При нажатии на кнопку редактирования открывается новое окно, в котором управляющий может поправить информацию о соответствующем сотруднике

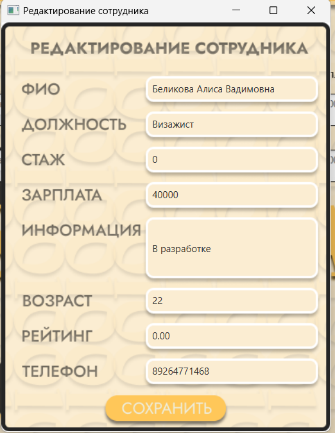


Рисунок 15 Редактирование сотрудника

При нажатии на кнопку данные обновляются вне зависимости от того, изменились они или нет. Данные обновляются путем вызова функции UpdateEmployee

public static void UpdateEmployee(String id, String t1, String t2, String t3, String t4,  
 String t5, String t6, String t7, String t8) throws SQLException {  
 *data\_statmt*.execute("UPDATE employee " +  
 "set employee\_name = '" + t1 + "'," +  
 " employee\_post = '"+ t2 + "'," +  
 " employee\_exp = " + t3 + "," +  
 " employee\_inf = '" + t5 + "'," +  
 " employee\_age = " + t6 + "," +  
 " employee\_num = " + t8 +  
 " where employee\_id = " + id+";");  
}



Рисунок 16 Просмотр своего заведения

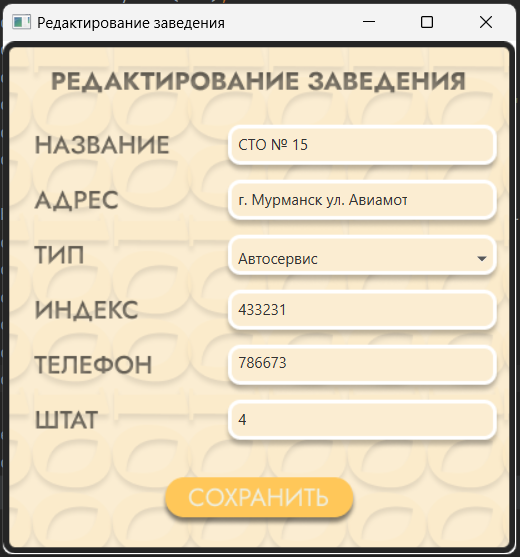


Рисунок 20 Редактирование своего заведения

При нажатии на кнопку «сохранить» вызывается метод UpdateStructure.

public static void UpdateStructure(String id, String t1, String t2, String t3, String t4, String t5, String t6) throws SQLException {  
 //System.out.println(t1+"|"+t2+"|"+ t3+"|" + t4);  
 *statmt*.execute("UPDATE structure " +  
 "set structure\_name = '" + t1 + "'," +  
 " address = '"+ t2 + "'," +  
 " specialty\_name = '" + t3 + "'," +  
 " post\_index = '" + t4 + "'," +  
 " tel\_number = '" + t5 + "'," +  
 " num\_employees = " + t6 +  
 " where structure\_id = " + id+";");  
}

**Список источников**

"PostgreSQL Documentation" // PostgreSQL [Электронный ресурс] URL: <https://www.postgresql.org/docs/14/index.html>

"PostgreSQL Tutorial" // PostgreSQL Tutorial [Электронный ресурс] URL:<https://www.postgresqltutorial.com/>

"JavaFX Documentation" // OpenJFX [Электронный ресурс] URL:<https://openjfx.io/>

"JavaFX Tutorial" // Tutorialspoint [Электронный ресурс] URL:<https://www.tutorialspoint.com/javafx/index.htm>

"JavaMail API Documentation" // JavaEE [Электронный ресурс] URL:<https://javaee.github.io/javamail/>

"PostgreSQL JDBC Driver Documentation" // PostgreSQL JDBC Driver [Электронный ресурс] URL:<https://jdbc.postgresql.org/documentation/>