# **ПРИЛОЖЕНИЕА**

**Текст программы**

/\*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Севастопольский государственный университет

Кафедра Информационных систем

Программа «Салон красоты»

Текст программы

РАЗРАБОТАЛА

Студент гр. ИС-б/32-O

Сирота М.Р.

2018

/\*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Утверждено 23.12.2018

Среда программирования Eclipse IDE for Java Developers Version: Photon Release (4.8.0)

Дата последней коррекции: 15.12.2018

Версия 1.0

/\*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

customer.java

**package** ud;

**public** **class** customer **implements** Comparable<customer> {

**protected** **int** id;

**protected** String secondName;

**protected** String name;

**protected** String fatherName;

**protected** String phone;

**protected** **int** visit;

**public** customer(**int** id\_, String secondName\_, String name\_,String fatherName\_, String phone\_, **int** visit\_) {

id=id\_;

secondName=secondName\_;

name=name\_;

fatherName=fatherName\_;

phone=phone\_;

visit=visit\_;

}

**public** **int** getID() {

**return** id;

}

**public** String getSecondName() {

**return** secondName;

}

**public** String getName() {

**return** name;

}

**public** String getFatherName() {

**return** fatherName;

}

**public** String getPhone() {

**return** phone;

}

**public** **int** getVisit() {

**return** visit;

}

**public** **void** setID(**int** id) {

**this**.id = id;

}

**public** **void** setSecondName(String secondName) {

**this**.secondName = secondName;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** **void** setFatherName(String fatherName) {

**this**.fatherName = fatherName;

}

**public** **void** setPhone(String phone) {

**this**.phone = phone;

}

**public** **void** setVisit(**int** visit) {

**this**.visit = visit;

}

@Override

**public** **int** compareTo(customer c) {

**return** **this**.secondName.compareTo(c.secondName);

}

}

CustomerTableModel.java

package ud;

import java.util.Iterator;

import java.util.LinkedList;

import javax.swing.table.AbstractTableModel;

public class CustomerTableModel extends AbstractTableModel {

private static final long serialVersionUID = 7927259757559420606L;

private LinkedList<customer> CustomerT\_obj;

public CustomerTableModel(LinkedList<customer> CustomerT\_obj) {

this.CustomerT\_obj = CustomerT\_obj;

}

public Class<?> getColumnClass(int columnIndex) {

return getValueAt(0, columnIndex).getClass();

}

public int getColumnCount() {

return 6;

}

public String getColumnName(int columnIndex) {

switch (columnIndex) {

case 0:

return "ID";

case 1:

return "Фамилия";

case 2:

return "Имя";

case 3:

return "Отчество";

case 4:

return "Телефон";

case 5:

return "Посещения";

}

return "";

}

public int getRowCount() {

return CustomerT\_obj.size();

}

public Object getValueAt(int rowIndex, int columnIndex) {

customer p = CustomerT\_obj.get(rowIndex);

switch (columnIndex) {

case 0:

return p.getID();

case 1:

return p.getSecondName();

case 2:

return p.getName();

case 3:

return p.getFatherName();

case 4:

return p.getPhone();

case 5:

return p.getVisit();

}

return "";

}

public boolean isCellEditable(int rowIndex, int columnIndex) {

return false;

}

public void setValueAt(Object value, int rowIndex, int columnIndex) {

}

public void addRow(customer nr) { // добавление строки в таблицу

CustomerT\_obj.add(nr);

fireTableDataChanged();// вызываем для обновления таблицы

// выделяем добавленную строку - иначе после обновления таблицы теряет выделение

int index = CustomerT\_obj.indexOf(nr);

tablePane.table.setRowSelectionInterval(index, index);

}

public void updateRow(int index, customer nr) {// обновление строки в таблице

CustomerT\_obj.set(index, nr);

fireTableDataChanged();// вызываем для обновления таблицы

// выделяем измененную строку - иначе после обновления таблицы теряет выделение

tablePane.table.setRowSelectionInterval(index, index);

}

public void deleteRow(int id) { // удаление строки из таблицы

Iterator<customer> isph = CustomerT\_obj.iterator();

boolean flag = false;

while (isph.hasNext()) {

if (isph.next().getID()==id) {

isph.remove();

flag = true;

break;

}

}

if (flag) {

fireTableDataChanged();// вызываем для обновления таблицы

} else

// JOPCT\_objionPane.showMessageDialog(myJframe.this,"Запись "+Model+" ненайдена!")

;

}

public void deleteAll() { // удаление строки из таблицы

int size = CustomerT\_obj.size();

CustomerT\_obj.clear();

fireTableRowsDeleted(0, size);

}

}

insert.java

package ud;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

import javax.swing.JOptionPane;

public class insert {

Statement st;

public insert() {

}

public int insertRow(String sql) {

try (Connection connection = DriverManager.getConnection(

"jdbc:mysql://127.0.0.1:3306/salon?serverTimezone=Europe/Moscow&useSSL=false", "root", "")) {

try {

st = connection.createStatement();

int i = st.executeUpdate(sql, Statement.RETURN\_GENERATED\_KEYS);

if (i > 0) {

JOptionPane.showMessageDialog(null, "Выполнено!");

ResultSet rs = st.getGeneratedKeys();

if (rs.next()) {

return rs.getInt(1);

}

} else {

JOptionPane.showMessageDialog(null, "Невозможно выпонить команду!");

}

return i;

} catch (SQLException e) {

e.printStackTrace();

return -1;

}

} catch (Exception ex) {

JOptionPane.showMessageDialog(null, "Connection failed..." + ex);

System.out.println(ex);

return -1;

}

}

}

Java\_login.java

package ud;

import java.awt.EventQueue;

import javax.swing.JFrame;

import javax.swing.JTextField;

import javax.swing.SwingUtilities;

import javax.swing.JPanel;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JButton;

import java.awt.Font;

import java.awt.event.ActionListener;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

import java.awt.event.ActionEvent;

import java.awt.BorderLayout;

import javax.swing.JPasswordField;

import javax.swing.JRootPane;

public class Java\_login extends JFrame {

/\*\*

\*

\*/

private static final long serialVersionUID = 1L;

private JTextField LoginTextField;

private JLabel lblLogin;

private JLabel lblPassword;

private JPasswordField passwordTextField;

public static boolean isAdmin(String x)

{

Pattern p = Pattern.compile("^admin$");

Matcher m = p.matcher(x);

return m.matches();

}

/\*\*

\* Launch the application.

\*/

public static void main(String[] args) {

EventQueue.invokeLater(new Runnable() {

public void run() {

try {

Java\_login frame = new Java\_login();

frame.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

});

}

public Java\_login() {

setBounds(100, 100, 450, 300);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

getContentPane().setLayout(new BorderLayout());

JPanel panel = new JPanel();

panel.setBounds(0, 0, 434, 261);

getContentPane().add(panel);

panel.setLayout(null);

JPanel panelMenu = new JPanel();

panelMenu.setBounds(0, 0, 434, 261);

panelMenu.setLayout(null);

JButton btnCustomerButton = new JButton("\u041A\u043B\u0438\u0435\u043D\u0442\u044B");

btnCustomerButton.setBounds(150, 10, 120, 30);

btnCustomerButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent arg0) {

EventQueue.invokeLater(new Runnable() {

public void run() {

try {

tablePane frame = new tablePane();

frame.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

});

}

});

panelMenu.add(btnCustomerButton);

JButton btnWorkerButton = new JButton("\u0421\u043E\u0442\u0440\u0443\u0434\u043D\u0438\u043A\u0438");

btnWorkerButton.setBounds(150, 50, 120, 30);

btnWorkerButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent arg0) {

EventQueue.invokeLater(new Runnable() {

public void run() {

try {

tablePanelForWorker frame = new tablePanelForWorker ();

frame.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

});

}

});

panelMenu.add(btnWorkerButton);

JButton btnServiceButton = new JButton("\u0423\u0441\u043B\u0443\u0433\u0438");

btnServiceButton.setBounds(150, 90, 120, 30);

btnServiceButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent arg0) {

EventQueue.invokeLater(new Runnable() {

public void run() {

try {

tablePanelForService frame = new tablePanelForService(2);

frame.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

});

}

});

panelMenu.add(btnServiceButton);

JButton btnRecondButton = new JButton("Записать");

btnRecondButton.setBounds(150, 130, 120, 30);

btnRecondButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent arg0) {

EventQueue.invokeLater(new Runnable() {

public void run() {

try {

visitAdd frame = new visitAdd();

frame.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

});

}

});

panelMenu.add(btnRecondButton);

JButton btnShowVisit = new JButton("Посещения");

btnShowVisit.setBounds(150, 170, 120, 30);

btnShowVisit.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent arg0) {

EventQueue.invokeLater(new Runnable() {

public void run() {

try {

panelRecord frame = new panelRecord();

frame.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

});

}

});

panelMenu.add(btnShowVisit);

JButton btnExitButton = new JButton("Выход");

btnExitButton.setBounds(150, 210, 120, 30);

btnExitButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent arg0) {

getContentPane().remove(panelMenu);

setContentPane(panel);

LoginTextField.setText(null);

passwordTextField.setText(null);

panel.setLayout(null);

update(getGraphics());

/\*dispose();

EventQueue.invokeLater(new Runnable() {

public void run() {

try {

Java\_login frame = new Java\_login();

frame.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

});\*/

}

});

panelMenu.add(btnExitButton);

JButton btnUsers = new JButton("Пользователи");

btnUsers.setBounds(10, 10, 130, 70);

btnUsers.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent arg0) {

EventQueue.invokeLater(new Runnable() {

public void run() {

try {

tablePanelForUser frame = new tablePanelForUser();

frame.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

});

}

});

panelMenu.add(btnUsers);

btnUsers.setVisible(false);

LoginTextField = new JTextField();

LoginTextField.setBounds(187, 57, 86, 20);

panel.add(LoginTextField);

//LoginTextField.setColumns(10);

JButton btnSignUp = new JButton("Login");

btnSignUp.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent arg0) {

try (Connection connection = DriverManager.getConnection(

"jdbc:mysql://127.0.0.1:3306/salon?serverTimezone=Europe/Moscow&useSSL=false", "root", "")) {

// JOptionPane.showMessageDialog(null, "Connection to Store DB succesfull!");

String uName = LoginTextField.getText();

if(isAdmin(uName)){

btnUsers.setVisible(true);

}

@SuppressWarnings("deprecation")

String query = "SELECT \* FROM `user` WHERE `login`='"+LoginTextField.getText()+"' and `password`='"+passwordTextField.getText()+"'";

PreparedStatement statement = connection.prepareStatement(query);

//statement.setString(1, uName);

//statement.setString(2, pass);

ResultSet set = statement.executeQuery();

if (set.next()) {

getContentPane().remove(panel);

// panel.removeAll();

// panel.add(panelMenu);

// getContentPane().add(panelMenu);

setContentPane(panelMenu);

panelMenu.setLayout(new BorderLayout());

update(getGraphics());

} else {

JOptionPane.showMessageDialog(null, "Login uncorrect!");

}

} catch (Exception ex) {

JOptionPane.showMessageDialog(null, "Connection failed..." + ex);

System.out.println(ex);

}

}

});

passwordTextField = new JPasswordField();

passwordTextField.setBounds(187, 88, 86, 20);

panel.add(passwordTextField);

passwordTextField.setColumns(10);

btnSignUp.setBounds(187, 119, 86, 27);

panel.add(btnSignUp);

btnSignUp.setFont(new Font("Tahoma", Font.PLAIN, 15));

lblPassword = new JLabel("Password");

lblPassword.setBounds(74, 87, 61, 19);

panel.add(lblPassword);

lblPassword.setFont(new Font("Tahoma", Font.PLAIN, 15));

lblLogin = new JLabel("Login");

lblLogin.setBounds(74, 56, 33, 19);

panel.add(lblLogin);

lblLogin.setFont(new Font("Tahoma", Font.PLAIN, 15));

JRootPane rootPane = SwingUtilities.getRootPane(btnSignUp);

rootPane.setDefaultButton(btnSignUp);

}

}

record.java

**package** ud;

**public** **class** record {

**protected** String secondNameCustomer;

**protected** String nameCustomer;

**protected** String fatherNameCustomer;

**protected** String phoneCustomer;

**protected** String secondNameWorker;

**protected** String nameWorker;

**protected** String fatherNameWorker;

**protected** String phoneWorker;

**protected** String nameService;

**protected** String dateService;

**protected** **int** hourService;

**protected** **int** statusService;

**public** record (String secondNameCustomer\_, String nameCustomer\_,String fatherNameCustomer\_, String phoneCustomer\_,

String secondNameWorker\_, String nameWorker\_, String fatherNameWorker\_, String phoneWorker\_,

String nameService\_, String dateService\_, **int** hourService\_, **int** statusService\_) {

secondNameCustomer=secondNameCustomer\_;

nameCustomer=nameCustomer\_;

fatherNameCustomer=fatherNameCustomer\_;

phoneCustomer=phoneCustomer\_;

secondNameWorker = secondNameWorker\_;

nameWorker = nameWorker\_;

fatherNameWorker = fatherNameWorker\_;

phoneWorker = phoneWorker\_;

nameService=nameService\_;

dateService=dateService\_;

hourService=hourService\_;

statusService=statusService\_;

}

**public** String getSecondNameCustomer() {

**return** secondNameCustomer;

}

**public** String getNameCustomer() {

**return** nameCustomer;

}

**public** String getFatherNameCustomer() {

**return** fatherNameCustomer;

}

**public** String getPhoneCustomer() {

**return** phoneCustomer;

}

**public** **void** setSecondNameCustomer(String secondNameCustomer) {

**this**.secondNameCustomer = secondNameCustomer;

}

**public** **void** setNameCustomer(String nameCustomer) {

**this**.nameCustomer = nameCustomer;

}

**public** **void** setFatherNameCustomer(String fatherNameCustomer) {

**this**.fatherNameCustomer = fatherNameCustomer;

}

**public** **void** setPhoneCustomer(String phoneCustomer) {

**this**.phoneCustomer = phoneCustomer;

}

**public** String getSecondNameWorker() {

**return** secondNameWorker;

}

**public** String getNameWorker() {

**return** nameWorker;

}

**public** String getFatherNameWorker() {

**return** fatherNameWorker;

}

**public** String getPhoneWorker() {

**return** phoneWorker;

}

**public** **void** setSecondNameWorker(String secondNameWorker) {

**this**.secondNameWorker = secondNameWorker;

}

**public** **void** setName(String nameWorker) {

**this**.nameWorker = nameWorker;

}

**public** **void** setFatherNameWorker(String fatherNameWorker) {

**this**.fatherNameWorker = fatherNameWorker;

}

**public** **void** setPhoneWorker(String phoneWorker) {

**this**.phoneWorker = phoneWorker;

}

**public** String getNameService() {

**return** nameService;

}

**public** **int** getHourService() {

**return** hourService;

}

**public** String getDateService() {

**return** dateService;

}

**public** **int** getStatusService() {

**return** statusService;

}

**public** **void** setNameService(String nameService) {

**this**.nameService = nameService;

}

**public** **void** setHourService(**int** hourService) {

**this**.hourService = hourService;

}

**public** **void** setDateService(String dateService) {

**this**.dateService = dateService;

}

**public** **void** setStatusService(**int** statusService) {

**this**.statusService = statusService;

}

}

RecordTableModel.java

package ud;

import java.util.LinkedList;

import javax.swing.table.AbstractTableModel;

public class RecordTableModel extends AbstractTableModel {

private static final long serialVersionUID = 7927259757559420606L;

private LinkedList<record> RecordT\_obj;

public RecordTableModel(LinkedList<record> RecordT\_obj) {// конструктор модели

this.RecordT\_obj = RecordT\_obj;

}

public Class<?> getColumnClass(int columnIndex) {

return getValueAt(0, columnIndex).getClass();

}

public int getColumnCount() {

return 12;

}

public String getColumnName(int columnIndex) {

switch (columnIndex) {

case 0:

return "KЛФамилия";

case 1:

return "KЛИмя";

case 2:

return "КЛОтчество";

case 3:

return "КЛТелефон";

case 4:

return "СФамилия";

case 5:

return "СИмя";

case 6:

return "СОтчество";

case 7:

return "СТелефон";

case 8:

return "Услуга";

case 9:

return "Дата";

case 10:

return "Время";

case 11:

return "Статус";

}

return "";

}

public int getRowCount() {

return RecordT\_obj.size();

}

public Object getValueAt(int rowIndex, int columnIndex) {

record p = RecordT\_obj.get(rowIndex);

switch (columnIndex) {

case 0:

return p.getSecondNameCustomer();

case 1:

return p.getNameCustomer();

case 2:

return p.getFatherNameCustomer();

case 3:

return p.getPhoneCustomer();

case 4:

return p.getSecondNameWorker();

case 5:

return p.getNameWorker();

case 6:

return p.getFatherNameWorker();

case 7:

return p.getPhoneWorker();

case 8:

return p.getNameService();

case 9:

return p.getDateService();

case 10:

return p.getHourService();

case 11:

return p.getStatusService();

}

return "";

}

public void addRow(record nr) { // добавление строки в таблицу

RecordT\_obj.add(nr);

fireTableDataChanged();// вызываем для обновления таблицы

// выделяем добавленную строку - иначе после обновления таблицы теряет выделение

int index = RecordT\_obj.indexOf(nr);

panelRecord.table.setRowSelectionInterval(index, index);

}

public boolean isCellEditable(int rowIndex, int columnIndex) {

return false;

}

public void setValueAt(Object value, int rowIndex, int columnIndex) {

}

public void updateRow(int index, record nr) {

RecordT\_obj.set(index, nr);

fireTableDataChanged();

panelRecord.table.setRowSelectionInterval(index, index);

}

public void deleteAll() { // удаление строки из таблицы

int size = RecordT\_obj.size();

RecordT\_obj.clear();

fireTableRowsDeleted(0, size);

}

}

service.java

**package** ud;

**import** java.util.Comparator;

**public** **class** service **implements** Comparator<service> {

**protected** **int** id;

**protected** String name;

**protected** **int** cost;

**protected** **int** time;

**public** service(**int** id\_, String name\_, **int** cost\_, **int** time\_) {

id=id\_;

name=name\_;

cost=cost\_;

time=time\_;

}

**public** **int** getID() {

**return** id;

}

**public** String getName() {

**return** name;

}

**public** **int** getCost() {

**return** cost;

}

**public** **int** getTime() {

**return** time;

}

**public** **void** setID(**int** id) {

**this**.id = id;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** **void** setCost(**int** cost) {

**this**.cost = cost;

}

**public** **void** setTime(**int** time) {

**this**.time = time;

}

@Override

**public** **int** compare(service c1, service c2) {

**return** c1.getID() - c2.getID();

}

}

ServiceTableModel.java

package ud;

import java.util.Iterator;

import java.util.LinkedList;

import javax.swing.table.AbstractTableModel;

public class ServiceTableModel extends AbstractTableModel {

private static final long serialVersionUID = 7927259757559420606L;

private LinkedList<service> ServiceT\_obj;

public ServiceTableModel(LinkedList<service> ServiceT\_obj) {

this.ServiceT\_obj = ServiceT\_obj;

}

public Class<?> getColumnClass(int columnIndex) {

return getValueAt(0, columnIndex).getClass();

}

public int getColumnCount() {

return 4;

}

public String getColumnName(int columnIndex) {

switch (columnIndex) {

case 0:

return "ID";

case 1:

return "Название";

case 2:

return "Cтоимость";

case 3:

return "Время (минуты)";

}

return "";

}

public void addRow(service nr) { // добавление строки в таблицу

ServiceT\_obj.add(nr);

fireTableDataChanged();// вызываем для обновления таблицы

// выделяем добавленную строку - иначе после обновления таблицы теряет выделение

int index = ServiceT\_obj.indexOf(nr);

tablePanelForService.table.setRowSelectionInterval(index, index);

}

public int getRowCount() {

return ServiceT\_obj.size();

}

public Object getValueAt(int rowIndex, int columnIndex) {

service p = ServiceT\_obj.get(rowIndex);

switch (columnIndex) {

case 0:

return p.getID();

case 1:

return p.getName();

case 2:

return p.getCost();

case 3:

return p.getTime();

}

return "";

}

public boolean isCellEditable(int rowIndex, int columnIndex) {

return false;

}

public void setValueAt(Object value, int rowIndex, int columnIndex) {

}

public void updateRow(int index, service nr) {

ServiceT\_obj.set(index, nr);

fireTableDataChanged();

tablePanelForService.table.setRowSelectionInterval(index, index);

}

public void deleteRow(int id) {

Iterator<service> isph = ServiceT\_obj.iterator();

boolean flag = false;

while (isph.hasNext()) {

if (isph.next().getID()==id) {

isph.remove();

flag = true;

break;

}

}

if (flag) {

fireTableDataChanged();

} else

;

}

public void deleteAll() { // удаление строки из таблицы

int size = ServiceT\_obj.size();

ServiceT\_obj.clear();

fireTableRowsDeleted(0, size);

}

}

tablePane,java

package ud;

import java.awt.EventQueue;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.MouseEvent;

import java.awt.event.MouseListener;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.util.LinkedList;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

import javax.swing.JButton;

import javax.swing.JFrame;

import javax.swing.JPanel;

import javax.swing.JScrollPane;

import javax.swing.JTable;

import javax.swing.border.EmptyBorder;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JTextField;

public class tablePane extends JFrame {

private static final long serialVersionUID = 1L;

private JPanel contentPane;

static JTable table;

private JTextField textFieldName;

private JTextField textFieldSecondName;

private JTextField textFieldFatherName;

private JTextField textFieldPhone;

private JTextField textFieldForSearch;

public static void main(String[] args) {

EventQueue.invokeLater(new Runnable() {

public void run() {

try {

tablePane frame = new tablePane();

frame.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

});

}

/\*\*

\* Create the frame.

\*/

public static boolean isNumeric(String x)

{

Pattern p = Pattern.compile("^\\d{11}$");

Matcher m = p.matcher(x);

return m.matches();

}

public static boolean isWorld(String x)

{

Pattern p = Pattern.compile("^[а-яА-ЯёЁ]+$");

Matcher m = p.matcher(x);

return m.matches();

}

public tablePane() {

setDefaultCloseOperation(JFrame.HIDE\_ON\_CLOSE );

setBounds(100, 100, 600, 350);

contentPane = new JPanel();

contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

setContentPane(contentPane);

getContentPane().setLayout(null);

contentPane.setLayout(null);

JPanel panel = new JPanel();

panel.setBounds(0, 0, 574, 311);

getContentPane().add(panel);

final LinkedList<customer> tbl = new LinkedList<customer>();

final CustomerTableModel CustomerT\_obj = new CustomerTableModel(tbl);

JButton btnAdd = new JButton("\u0414\u043E\u0431\u0430\u0432\u0438\u0442\u044C");

btnAdd.setBounds(470, 209, 90, 23);

btnAdd.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent arg0) {

try (Connection connection = DriverManager.getConnection(

"jdbc:mysql://127.0.0.1:3306/salon?serverTimezone=Europe/Moscow&useSSL=false", "root", "")) {

String sn = textFieldSecondName.getText();

String n = textFieldName.getText();

String fn = textFieldFatherName.getText();

String p = textFieldPhone.getText();

int flagSn = 1;

int flagN = 1;

int flagFn = 1;

int flagP = 1;

if (!isWorld(sn)) {

flagSn = 0;

JOptionPane.showMessageDialog(null, "Введите Фамилию!");

}

if (!isWorld(n)) {

flagN = 0;

JOptionPane.showMessageDialog(null, "Введите Имя!");

}

if (!isWorld(fn)) {

flagFn = 0;

JOptionPane.showMessageDialog(null, "Введите Отчество!");

}

if (!isNumeric(p) && sn.length() != 11) {

flagP = 0;

JOptionPane.showMessageDialog(null, "Введите Телефон!");

}

if (flagSn == 1 && flagN == 1 && flagFn == 1 && flagP == 1) {

String query = "INSERT INTO `salon`.`customer` (`id\_Customer`, `Second\_Name`, `Name`, `Father\_Name`, `Phone`, `Visit`)"

+ " VALUES (NULL, '" + sn + "', '" + n + "', '" + fn + "', '" + p + "', '0');";

insert it = new insert();

int newId = it.insertRow(query);

if (newId > 0) {

CustomerT\_obj.addRow(new customer(newId, sn, n, fn, p, 0));

textFieldSecondName.setText(null);

textFieldName.setText(null);

textFieldFatherName.setText(null);

textFieldPhone.setText(null);

}

}

} catch (Exception ex) {

JOptionPane.showMessageDialog(null, "Неверные данные" + ex);

System.out.println(ex);

}

}

});

panel.setLayout(null);

panel.add(btnAdd);

JScrollPane scrollPane = new JScrollPane();

scrollPane.setBounds(20, 68, 544, 117);

panel.add(scrollPane);

table = new JTable(CustomerT\_obj);

table.getColumnModel().getColumn(1).setPreferredWidth(98);

scrollPane.setViewportView(table);

JLabel lblName = new JLabel("\u0418\u043C\u044F");

lblName.setBounds(210, 185, 70, 14);

panel.add(lblName);

JLabel lblSecondName = new JLabel("\u0424\u0430\u043C\u0438\u043B\u0438\u044F");

lblSecondName.setBounds(110, 185, 70, 14);

panel.add(lblSecondName);

JLabel lblFatherName = new JLabel("\u041E\u0442\u0447\u0435\u0441\u0442\u0432\u043E");

lblFatherName.setBounds(300, 185, 70, 14);

panel.add(lblFatherName);

JLabel lblPhone = new JLabel("\u0422\u0435\u043B\u0435\u0444\u043E\u043D");

lblPhone.setBounds(390, 185, 70, 14);

panel.add(lblPhone);

textFieldName = new JTextField();

textFieldName.setBounds(200, 210, 80, 20);

panel.add(textFieldName);

textFieldName.setColumns(10);

textFieldSecondName = new JTextField();

textFieldSecondName.setBounds(110, 210, 80, 20);

textFieldSecondName.setColumns(10);

panel.add(textFieldSecondName);

textFieldFatherName = new JTextField();

textFieldFatherName.setBounds(290, 210, 80, 20);

textFieldFatherName.setColumns(10);

panel.add(textFieldFatherName);

textFieldPhone = new JTextField();

textFieldPhone.setBounds(380, 210, 80, 20);

textFieldPhone.setColumns(10);

panel.add(textFieldPhone);

JButton btnUpdate = new JButton("\u0418\u0437\u043C\u0435\u043D\u0438\u0442\u044C");

btnUpdate.setBounds(470, 243, 90, 23);

btnUpdate.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

if (table.getSelectedRow() >= 0) {

String sn = textFieldSecondName.getText();

String n = textFieldName.getText();

String fn = textFieldFatherName.getText();

String p = textFieldPhone.getText();

int flagSn = 1;

int flagN = 1;

int flagFn = 1;

int flagP = 1;

if (!isWorld(sn)) {

flagSn = 0;

JOptionPane.showMessageDialog(null, "Введите Фамилию!");

}

if (!isWorld(n)) {

flagN = 0;

JOptionPane.showMessageDialog(null, "Введите Имя!");

}

if (!isWorld(fn)) {

flagFn = 0;

JOptionPane.showMessageDialog(null, "Введите Отчество!");

}

if (!isNumeric(p) && sn.length() != 11) {

flagP = 0;

JOptionPane.showMessageDialog(null, "Введите Телефон!");

}

int row = table.getSelectedRow();

int i = (int) table.getValueAt(row, 0);

if (flagSn == 1 && flagN == 1 && flagFn == 1 && flagN == 1 && flagP == 1) {

String query = "UPDATE `salon`.`customer` SET `Second\_Name` = '" + sn + "', `Name` = '" + n

+ "', `Father\_Name` = '" + fn + "', `Phone` = '" + p + "'"

+ " WHERE `customer`.`id\_Customer` = " + table.getValueAt(row, 0).toString() + ";";

insert it = new insert();

int newId = it.insertRow(query);

if (newId > 0) {

CustomerT\_obj.updateRow(table.getSelectedRow(), new customer(i, sn, n, fn, p, 0));

textFieldSecondName.setText(null);

textFieldName.setText(null);

textFieldFatherName.setText(null);

textFieldPhone.setText(null);

}

}

}

}

});

panel.add(btnUpdate);

JButton btnDelete = new JButton("\u0423\u0434\u0430\u043B\u0438\u0442\u044C");

btnDelete.setBounds(470, 277, 90, 23);

btnDelete.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

int row = table.getSelectedRow();

int i = (int) table.getValueAt(row, 0);

String query = "DELETE FROM `salon`. `customer` WHERE `customer`.`id\_Customer` = " + table.getValueAt(row, 0).toString() + ";";

insert it = new insert();

int newId = it.insertRow(query);

if (newId > 0) {

CustomerT\_obj.deleteRow(i);

}

}

});

panel.add(btnDelete);

JButton btnSearch = new JButton("\u041D\u0430\u0439\u0442\u0438");

btnSearch.setBounds(470, 10, 90, 23);

btnSearch.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

String search= textFieldForSearch.getText();

try (Connection connection = DriverManager.getConnection("jdbc:mysql://127.0.0.1:3306/salon?serverTimezone=Europe/Moscow&useSSL=false", "root",""))

{

PreparedStatement statement = connection.prepareStatement("SELECT \* FROM `customer` "

+ "WHERE `Second\_Name` LIKE '" + search + "' OR `Name` LIKE '" + search + "' OR `"

+ "Father\_Name`LIKE '" + search + "' OR `Phone`LIKE '" + search + "'");

ResultSet res = statement.executeQuery();

//if (!res.next()) {

//JOptionPane.showMessageDialog(null, "Ничего не найдено...");

// }

CustomerT\_obj.deleteAll();

while (res.next()) {

CustomerT\_obj.addRow(new customer(res.getInt(1), res.getString(2), res.getString(3),

res.getString(4), res.getString(5), res.getInt(6)));

}

} catch (Exception ex) {

JOptionPane.showMessageDialog(null, "Connection failed..."+ex);

System.out.println(ex);

}

}

});

panel.add(btnSearch);

textFieldForSearch = new JTextField();

textFieldForSearch.setBounds(20, 11, 440, 20);

panel.add(textFieldForSearch);

textFieldForSearch.setColumns(10);

JButton btnCancel = new JButton("\u041E\u0442\u043C\u0435\u043D\u0430");

btnCancel.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

CustomerT\_obj.deleteAll();

textFieldForSearch.setText(null);

try (Connection connection = DriverManager.getConnection("jdbc:mysql://127.0.0.1:3306/salon?serverTimezone=Europe/Moscow&useSSL=false", "root",""))

{

PreparedStatement statement = connection.prepareStatement("SELECT \* \r\n" +

"FROM `customer`");

ResultSet res = statement.executeQuery();

while(res.next()) {

CustomerT\_obj.addRow(new customer(res.getInt(1), res.getString(2),

res.getString(3),res.getString(4),res.getString(5),res.getInt(6)));

}

} catch (Exception ex) {

JOptionPane.showMessageDialog(null, "Connection failed..."+ex);

System.out.println(ex);

}

}

});

btnCancel.setBounds(470, 34, 89, 23);

panel.add(btnCancel);

table.addMouseListener(new MouseListener() {

public void mouseClicked(MouseEvent e) {

int row = table.getSelectedRow();

textFieldSecondName.setText(table.getValueAt(row, 1).toString());

textFieldName.setText(table.getValueAt(row, 2).toString());

textFieldFatherName.setText(table.getValueAt(row, 3).toString());

textFieldPhone.setText(table.getValueAt(row, 4).toString());

}

@Override

public void mouseEntered(MouseEvent e) {

// TODO Auto-generated method stub

}

@Override

public void mouseExited(MouseEvent e) {

// TODO Auto-generated method stub

}

@Override

public void mousePressed(MouseEvent e) {

// TODO Auto-generated method stub

}

@Override

public void mouseReleased(MouseEvent e) {

// TODO Auto-generated method stub

}

});

try (Connection connection = DriverManager.getConnection("jdbc:mysql://127.0.0.1:3306/salon?serverTimezone=Europe/Moscow&useSSL=false", "root",""))

{

PreparedStatement statement = connection.prepareStatement("SELECT \* \r\n" +

"FROM `customer`");

ResultSet res = statement.executeQuery();

while(res.next()) {

CustomerT\_obj.addRow(new customer(res.getInt(1), res.getString(2),

res.getString(3),res.getString(4),res.getString(5),res.getInt(6)));

}

} catch (Exception ex) {

JOptionPane.showMessageDialog(null, "Connection failed..."+ex);

System.out.println(ex);

}

table.getColumnModel().getColumn(0).setMinWidth(0);

table.getColumnModel().getColumn(0).setMaxWidth(0);

table.getColumnModel().getColumn(0).setWidth(0);

}

}

tablePanelForService.java

package ud;

import java.awt.EventQueue;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.MouseEvent;

import java.awt.event.MouseListener;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.util.LinkedList;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

import javax.swing.JButton;

import javax.swing.JFrame;

import javax.swing.JPanel;

import javax.swing.JScrollPane;

import javax.swing.JTable;

import javax.swing.border.EmptyBorder;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JTextField;

public class tablePanelForService extends JFrame {

private static final long serialVersionUID = 1L;

private JPanel contentPane;

static JTable table;

private JTextField textFieldTime;

private JTextField textFieldCost;

private JTextField textFieldName;

private JTextField textFieldForSearch;

public String idServiceForWorker;

public static void main(String[] args) {

EventQueue.invokeLater(new Runnable() {

public void run() {

try {

tablePanelForService frame = new tablePanelForService(1);

frame.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

});

}

public static boolean isNumeric(String x)

{

Pattern p = Pattern.compile("^\\d+(?:\\.\\d+)?$");

Matcher m = p.matcher(x);

return m.matches();

}

public static boolean isWorld(String x)

{

Pattern p = Pattern.compile("^[а-яА-ЯёЁ]+$");

Matcher m = p.matcher(x);

return m.matches();

}

public String getIDForWorker(String x)

{

return idServiceForWorker;

}

/\*\*

\* Create the frame.

\*/

public tablePanelForService(int choose) {

if (choose==1)

loadTwo();

else load();

}

public void load() {

setDefaultCloseOperation(JFrame.HIDE\_ON\_CLOSE );

setBounds(100, 100, 600, 350);

contentPane = new JPanel();

contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

setContentPane(contentPane);

getContentPane().setLayout(null);

contentPane.setLayout(null);

JPanel panel = new JPanel();

panel.setBounds(0, 0, 584, 311);

getContentPane().add(panel);

final LinkedList<service> tbl = new LinkedList<service>();

final ServiceTableModel ServiceT\_obj = new ServiceTableModel(tbl);

JButton btnAdd = new JButton("\u0414\u043E\u0431\u0430\u0432\u0438\u0442\u044C");

btnAdd.setBounds(485, 209, 89, 23);

btnAdd.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent arg0) {

try (Connection connection = DriverManager.getConnection("jdbc:mysql://127.0.0.1:3306/salon?serverTimezone=Europe/Moscow&useSSL=false", "root",""))

{

String n=textFieldName.getText();

String c=textFieldCost.getText();

String t=textFieldTime.getText();

int flagN = 1;

int flagC = 1;

int flagT = 1;

if (!isWorld(n)) {

flagN = 0;

JOptionPane.showMessageDialog(null, "Введите Название!");

}

if (!isNumeric(c)) {

flagC = 0;

JOptionPane.showMessageDialog(null, "Введите Стоимость!");

}

if (!isNumeric(t)) {

flagT = 0;

JOptionPane.showMessageDialog(null, "Введите Длительность!");

}

if (flagN == 1 && flagC == 1 && flagT == 1) {

String query="INSERT INTO `salon`.`sevice` (`id\_Service`, `Name`, `Cost`, `Time`)"

+" VALUES (NULL, '"+n+"', '"+c+"', '"+t+"');";

insert it=new insert();

int newId = it.insertRow(query);

if(newId > 0 ) {

ServiceT\_obj.addRow(new service(newId,n,Integer.parseInt(textFieldCost.getText()),

Integer.parseInt(textFieldTime.getText())));

}

}

} catch (Exception ex) {

JOptionPane.showMessageDialog(null, "Connection failed..."+ex);

System.out.println(ex);

}

}

});

panel.setLayout(null);

panel.add(btnAdd);

JScrollPane scrollPane = new JScrollPane();

scrollPane.setBounds(20, 72, 554, 120);

panel.add(scrollPane);

table = new JTable(ServiceT\_obj);

table.getColumnModel().getColumn(1).setPreferredWidth(98);

scrollPane.setViewportView(table);

JLabel lblTime = new JLabel("\u0412\u0440\u0435\u043C\u044F");

lblTime.setBounds(340, 203, 50, 14);

panel.add(lblTime);

JLabel lblCost = new JLabel("\u0421\u0442\u043E\u0438\u043C\u043E\u0441\u0442\u044C");

lblCost.setBounds(250, 203, 70, 14);

panel.add(lblCost);

textFieldTime = new JTextField();

textFieldTime.setBounds(340, 220, 80, 20);

panel.add(textFieldTime);

textFieldTime.setColumns(10);

textFieldCost = new JTextField();

textFieldCost.setBounds(250, 220, 80, 20);

textFieldCost.setColumns(10);

panel.add(textFieldCost);

JLabel lblName = new JLabel("\u041D\u0430\u0437\u0432\u0430\u043D\u0438\u0435");

lblName.setBounds(160, 203, 70, 14);

panel.add(lblName);

textFieldName = new JTextField();

textFieldName.setBounds(160, 220, 80, 20);

panel.add(textFieldName);

textFieldName.setColumns(10);

table.addMouseListener(new MouseListener() {

public void mouseClicked(MouseEvent e) {

int row = table.getSelectedRow();

textFieldName.setText(table.getValueAt(row, 1).toString());

textFieldCost.setText(table.getValueAt(row, 2).toString());

textFieldTime.setText(table.getValueAt(row, 3).toString());

idServiceForWorker=table.getValueAt(row, 0).toString();

// сделать кнопку обновления недоступной(пока нет изменений)

//btnUpd.setEnabled(false);

}

@Override

public void mouseEntered(MouseEvent e) {

// TODO Auto-generated method stub

}

@Override

public void mouseExited(MouseEvent e) {

// TODO Auto-generated method stub

}

@Override

public void mousePressed(MouseEvent e) {

// TODO Auto-generated method stub

}

@Override

public void mouseReleased(MouseEvent e) {

// TODO Auto-generated method stub

}

});

JButton btnUpdate = new JButton("\u0418\u0437\u043C\u0435\u043D\u0438\u0442\u044C");

btnUpdate.setBounds(485, 243, 89, 23);

btnUpdate.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

if (table.getSelectedRow() >= 0) {

String n=textFieldName.getText();

String c=textFieldCost.getText();

String t=textFieldTime.getText();

int flagN = 1;

int flagC = 1;

int flagT = 1;

if (!isWorld(n)) {

flagN = 0;

JOptionPane.showMessageDialog(null, "Введите Название!");

}

if (!isNumeric(c)) {

flagC = 0;

JOptionPane.showMessageDialog(null, "Введите Стоимость!");

}

if (!isNumeric(t)) {

flagT = 0;

JOptionPane.showMessageDialog(null, "Введите Длительность!");

}

int row = table.getSelectedRow();

int i = (int) table.getValueAt(row, 0);

if (flagN == 1 && flagC == 1 && flagT == 1) {

String query = "UPDATE `salon`.`sevice` SET `Name` = '" + n + "', `Cost` = '" + c

+ "', `Time` = '" + t + "'"

+ " WHERE `sevice`.`id\_Service` = " + table.getValueAt(row, 0).toString() + ";";

insert it = new insert();

int newId = it.insertRow(query);

if (newId > 0) {

ServiceT\_obj.updateRow(table.getSelectedRow(), new service(i, n,Integer.parseInt(textFieldCost.getText()),

Integer.parseInt(textFieldTime.getText())));

textFieldName.setText(null);

textFieldCost.setText(null);

textFieldTime.setText(null);

}

}

}

}

});

panel.add(btnUpdate);

JButton btnDelete = new JButton("\u0423\u0434\u0430\u043B\u0438\u0442\u044C");

btnDelete.setBounds(485, 277, 89, 23);

btnDelete.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

int row = table.getSelectedRow();

int i = (int) table.getValueAt(row, 0);

String query = "DELETE FROM `salon`. `sevice` WHERE `sevice`.`id\_Service` = " + table.getValueAt(row, 0).toString() + ";";

insert it = new insert();

int newId = it.insertRow(query);

if (newId > 0) {

ServiceT\_obj.deleteRow(i);

}

}

});

panel.add(btnDelete);

JButton btnSearch = new JButton("\u041D\u0430\u0439\u0442\u0438");

btnSearch.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

String search= textFieldForSearch.getText();

try (Connection connection = DriverManager.getConnection("jdbc:mysql://127.0.0.1:3306/salon?serverTimezone=Europe/Moscow&useSSL=false", "root",""))

{

PreparedStatement statement = connection.prepareStatement("SELECT \* FROM `sevice` "

+ "WHERE `Name` LIKE '" + search + "' OR `Cost` LIKE '" + search + "' OR `"

+ "Time`LIKE '" + search + "'");

ResultSet res = statement.executeQuery();

//if (!res.next()) {

//JOptionPane.showMessageDialog(null, "Ничего не найдено...");

// }

ServiceT\_obj.deleteAll();

while (res.next()) {

ServiceT\_obj.addRow(new service(res.getInt(1), res.getString(2),res.getInt(3),res.getInt(4)));

}

} catch (Exception ex) {

JOptionPane.showMessageDialog(null, "Connection failed..."+ex);

System.out.println(ex);

}

}

});

btnSearch.setBounds(485, 12, 89, 23);

panel.add(btnSearch);

JButton btnCancel = new JButton("\u041E\u0442\u043C\u0435\u043D\u0430");

btnCancel.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

ServiceT\_obj.deleteAll();

textFieldForSearch.setText(null);

try (Connection connection = DriverManager.getConnection("jdbc:mysql://127.0.0.1:3306/salon?serverTimezone=Europe/Moscow&useSSL=false", "root",""))

{

PreparedStatement statement = connection.prepareStatement("SELECT \* \r\n" +

"FROM `sevice`");

ResultSet res = statement.executeQuery();

while(res.next()) {

ServiceT\_obj.addRow(new service(res.getInt(1), res.getString(2),res.getInt(3),res.getInt(4)));

}

} catch (Exception ex) {

JOptionPane.showMessageDialog(null, "Connection failed..."+ex);

System.out.println(ex);

}

}

});

btnCancel.setBounds(485, 38, 89, 23);

panel.add(btnCancel);

textFieldForSearch = new JTextField();

textFieldForSearch.setBounds(20, 12, 455, 20);

panel.add(textFieldForSearch);

textFieldForSearch.setColumns(10);

try (Connection connection = DriverManager.getConnection("jdbc:mysql://127.0.0.1:3306/salon?serverTimezone=Europe/Moscow&useSSL=false", "root",""))

{

PreparedStatement statement = connection.prepareStatement("SELECT \* \r\n" +

"FROM `sevice`");

ResultSet rest = statement.executeQuery();

while(rest.next()) {

ServiceT\_obj.addRow(new service(rest.getInt("id\_Service"),rest.getString("Name"), rest.getInt("Cost"),

rest.getInt("Time")));

}

} catch (Exception ex) {

JOptionPane.showMessageDialog(null, "Connection failed..."+ex);

System.out.println(ex);

}

table.getColumnModel().getColumn(0).setMinWidth(0);

table.getColumnModel().getColumn(0).setMaxWidth(0);

table.getColumnModel().getColumn(0).setWidth(0);

}

public String idConn;

public void setIDConn(String idConn) {

this.idConn = idConn;

}

public void loadTwo() {

setDefaultCloseOperation(JFrame.HIDE\_ON\_CLOSE );

setBounds(100, 100, 600, 350);

contentPane = new JPanel();

contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

setContentPane(contentPane);

getContentPane().setLayout(null);

contentPane.setLayout(null);

JPanel panel = new JPanel();

panel.setBounds(0, 0, 584, 311);

getContentPane().add(panel);

final LinkedList<service> tbl = new LinkedList<service>();

final ServiceTableModel ServiceT\_obj = new ServiceTableModel(tbl);

JButton btnAdd = new JButton("\u0414\u043E\u0431\u0430\u0432\u0438\u0442\u044C");

btnAdd.setBounds(485, 209, 89, 23);

btnAdd.setEnabled(false);

btnAdd.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent arg0) {

try (Connection connection = DriverManager.getConnection("jdbc:mysql://127.0.0.1:3306/salon?serverTimezone=Europe/Moscow&useSSL=false", "root",""))

{

int row = table.getSelectedRow();

String iy = table.getValueAt(row, 0).toString();

String query="INSERT INTO `salon`.`service\_worker` (`id\_Service`, `id\_Worker`) "

+" VALUES ('"+iy+"', '"+idConn+"');";

insert it=new insert();

it.insertRow(query);

} catch (Exception ex) {

JOptionPane.showMessageDialog(null, "Connection failed..."+ex);

System.out.println(ex);

}

dispose();

}

});

panel.setLayout(null);

panel.add(btnAdd);

JScrollPane scrollPane = new JScrollPane();

scrollPane.setBounds(20, 72, 554, 120);

panel.add(scrollPane);

table = new JTable(ServiceT\_obj);

table.getColumnModel().getColumn(1).setPreferredWidth(98);

scrollPane.setViewportView(table);

table.addMouseListener(new MouseListener() {

public void mouseClicked(MouseEvent e) {

btnAdd.setEnabled(true);

int row = table.getSelectedRow();

table.getValueAt(row, 0).toString();

}

@Override

public void mouseEntered(MouseEvent e) {

// TODO Auto-generated method stub

}

@Override

public void mouseExited(MouseEvent e) {

// TODO Auto-generated method stub

}

@Override

public void mousePressed(MouseEvent e) {

// TODO Auto-generated method stub

}

@Override

public void mouseReleased(MouseEvent e) {

// TODO Auto-generated method stub

}

});

try (Connection connection = DriverManager.getConnection("jdbc:mysql://127.0.0.1:3306/salon?serverTimezone=Europe/Moscow&useSSL=false", "root",""))

{

PreparedStatement statement = connection.prepareStatement("SELECT \* \r\n" +

"FROM `sevice`");

ResultSet rest = statement.executeQuery();

while(rest.next()) {

ServiceT\_obj.addRow(new service(rest.getInt("id\_Service"),rest.getString("Name"), rest.getInt("Cost"),

rest.getInt("Time")));

}

} catch (Exception ex) {

JOptionPane.showMessageDialog(null, "Connection failed..."+ex);

System.out.println(ex);

}

table.getColumnModel().getColumn(0).setMinWidth(0);

table.getColumnModel().getColumn(0).setMaxWidth(0);

table.getColumnModel().getColumn(0).setWidth(0);

}

}

tablePanelForUser.java

package ud;

import java.awt.EventQueue;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.MouseEvent;

import java.awt.event.MouseListener;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.util.LinkedList;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

import javax.swing.JButton;

import javax.swing.JFrame;

import javax.swing.JPanel;

import javax.swing.JScrollPane;

import javax.swing.JTable;

import javax.swing.border.EmptyBorder;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JTextField;

public class tablePanelForUser extends JFrame {

private static final long serialVersionUID = 1L;

private JPanel contentPane;

static JTable table;

private JTextField textFieldPassword;

private JTextField textFieldLogin;

public static void main(String[] args) {

EventQueue.invokeLater(new Runnable() {

public void run() {

try {

tablePanelForUser frame = new tablePanelForUser();

frame.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

});

}

/\*\*

\* Create the frame.

\*/

public static boolean isWorld(String x)

{

Pattern p = Pattern.compile("^[A-Za-z0-9]+$");

Matcher m = p.matcher(x);

return m.matches();

}

public tablePanelForUser() {

setDefaultCloseOperation(JFrame.HIDE\_ON\_CLOSE );

setBounds(100, 100, 436, 309);

contentPane = new JPanel();

contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

setContentPane(contentPane);

getContentPane().setLayout(null);

contentPane.setLayout(null);

JPanel panel = new JPanel();

panel.setBounds(0, 0, 420, 270);

getContentPane().add(panel);

final LinkedList<user> tbl = new LinkedList<user>();

final userTableModel UserT\_obj = new userTableModel(tbl);

JButton btnAdd = new JButton("\u0414\u043E\u0431\u0430\u0432\u0438\u0442\u044C");

btnAdd.setBounds(210, 151, 90, 23);

btnAdd.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent arg0) {

try (Connection connection = DriverManager.getConnection(

"jdbc:mysql://127.0.0.1:3306/salon?serverTimezone=Europe/Moscow&useSSL=false", "root", "")) {

String sn = textFieldLogin.getText();

String n = textFieldPassword.getText();

int flagSn = 1;

if (!isWorld(sn)) {

flagSn = 0;

JOptionPane.showMessageDialog(null, "Введите логин!");

}

if (flagSn == 1) {

String query = "INSERT INTO `salon`.`user` (`login`, `password`)"

+ " VALUES ('" + sn + "', '" + n + "');";

insert it = new insert();

int newId = it.insertRow(query);

if (newId > 0) {

UserT\_obj.addRow(new user(sn, n));

textFieldLogin.setText(null);

textFieldPassword.setText(null);

}

}

} catch (Exception ex) {

JOptionPane.showMessageDialog(null, "Неверные данные" + ex);

System.out.println(ex);

}

}

});

panel.setLayout(null);

panel.add(btnAdd);

JScrollPane scrollPane = new JScrollPane();

scrollPane.setBounds(20, 23, 280, 117);

panel.add(scrollPane);

table = new JTable(UserT\_obj);

table.getColumnModel().getColumn(1).setPreferredWidth(98);

scrollPane.setViewportView(table);

JLabel lblPassword = new JLabel("\u041F\u0430\u0440\u043E\u043B\u044C");

lblPassword.setBounds(120, 151, 70, 14);

panel.add(lblPassword);

JLabel lblLogin = new JLabel("\u041B\u043E\u0433\u0438\u043D");

lblLogin.setBounds(20, 151, 70, 14);

panel.add(lblLogin);

textFieldPassword = new JTextField();

textFieldPassword.setBounds(120, 176, 80, 20);

panel.add(textFieldPassword);

textFieldPassword.setColumns(10);

textFieldLogin = new JTextField();

textFieldLogin.setBounds(20, 176, 80, 20);

textFieldLogin.setColumns(10);

panel.add(textFieldLogin);

JButton btnUpdate = new JButton("\u0418\u0437\u043C\u0435\u043D\u0438\u0442\u044C");

btnUpdate.setBounds(210, 185, 90, 23);

btnUpdate.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

if (table.getSelectedRow() >= 0) {

String sn = textFieldLogin.getText();

String n = textFieldPassword.getText();

int flagSn = 1;

if (!isWorld(sn)) {

flagSn = 0;

JOptionPane.showMessageDialog(null, "Введите Фамилию!");

}

int row = table.getSelectedRow();

if (flagSn == 1) {

String query = "UPDATE `salon`.`user` SET `login` = '" + sn + "', `password` = '" + n

+ "'"

+ " WHERE `user`.`login` = '" + table.getValueAt(row, 0).toString() + "';";

insert it = new insert();

int newId = it.insertRow(query);

if (newId > 0) {

UserT\_obj.updateRow(table.getSelectedRow(), new user(sn, n));

textFieldLogin.setText(null);

textFieldPassword.setText(null);

}

}

}

}

});

panel.add(btnUpdate);

JButton btnDelete = new JButton("\u0423\u0434\u0430\u043B\u0438\u0442\u044C");

btnDelete.setBounds(210, 219, 90, 23);

btnDelete.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

int row = table.getSelectedRow();

String i = table.getValueAt(row, 0).toString();

String query = "DELETE FROM `salon`. `user` WHERE `user`.`login` = '" + table.getValueAt(row, 0).toString() + "';";

insert it = new insert();

int newId = it.insertRow(query);

if (newId > 0) {

UserT\_obj.deleteRow(i);

}

textFieldLogin.setText(null);

textFieldPassword.setText(null);

}

});

panel.add(btnDelete);

table.addMouseListener(new MouseListener() {

public void mouseClicked(MouseEvent e) {

int row = table.getSelectedRow();

textFieldLogin.setText(table.getValueAt(row, 0).toString());

textFieldPassword.setText(table.getValueAt(row, 1).toString());

}

@Override

public void mouseEntered(MouseEvent e) {

// TODO Auto-generated method stub

}

@Override

public void mouseExited(MouseEvent e) {

// TODO Auto-generated method stub

}

@Override

public void mousePressed(MouseEvent e) {

// TODO Auto-generated method stub

}

@Override

public void mouseReleased(MouseEvent e) {

// TODO Auto-generated method stub

}

});

try (Connection connection = DriverManager.getConnection("jdbc:mysql://127.0.0.1:3306/salon?serverTimezone=Europe/Moscow&useSSL=false", "root",""))

{

PreparedStatement statement = connection.prepareStatement("SELECT \* \r\n" +

"FROM `user`");

ResultSet res = statement.executeQuery();

while(res.next()) {

UserT\_obj.addRow(new user(res.getString(1), res.getString(2)));

}

} catch (Exception ex) {

JOptionPane.showMessageDialog(null, "Connection failed..."+ex);

System.out.println(ex);

}

}

}

package ud;

import java.awt.EventQueue;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.MouseEvent;

import java.awt.event.MouseListener;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.util.LinkedList;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

import javax.swing.JButton;

import javax.swing.JFrame;

import javax.swing.JPanel;

import javax.swing.JScrollPane;

import javax.swing.JTable;

import javax.swing.border.EmptyBorder;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JTextField;

public class tablePanelForWorker extends JFrame {

private static final long serialVersionUID = 1L;

private JPanel contentPane;

static JTable table;

private JTextField textFieldName;

private JTextField textFieldSecondName;

private JTextField textFieldFatherName;

private JTextField textFieldPhone;

private JTextField textFieldProssent;

private JTextField textFieldDate;

private JTextField textFieldForSearch;

public static void main(String[] args) {

EventQueue.invokeLater(new Runnable() {

public void run() {

try {

tablePanelForWorker frame = new tablePanelForWorker();

frame.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

});

}

public static boolean isNumeric(String x) {

Pattern p = Pattern.compile("^\\d{11}$");

Matcher m = p.matcher(x);

return m.matches();

}

public static boolean isWorld(String x) {

Pattern p = Pattern.compile("^[а-яА-ЯёЁ]+$");

Matcher m = p.matcher(x);

return m.matches();

}

public static boolean isDate(String x) {

Pattern p = Pattern.compile("^20\\d{2}-\\d{1,2}-\\d{1,2}$");

Matcher m = p.matcher(x);

return m.matches();

}

/\*\*

\* Create the frame.

\*/

public tablePanelForWorker() {

setDefaultCloseOperation(JFrame.HIDE\_ON\_CLOSE);

setBounds(100, 100, 600, 350);

contentPane = new JPanel();

contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

setContentPane(contentPane);

getContentPane().setLayout(null);

contentPane.setLayout(null);

JPanel panel = new JPanel();

panel.setBounds(0, 0, 584, 311);

getContentPane().add(panel);

final LinkedList<worker> tbl = new LinkedList<worker>();

final WorkerTableModel WorkerT\_obj = new WorkerTableModel(tbl);

JButton btnAdd = new JButton("\u0414\u043E\u0431\u0430\u0432\u0438\u0442\u044C");

btnAdd.setBounds(482, 209, 91, 23);

btnAdd.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent arg0) {

try (Connection connection = DriverManager.getConnection(

"jdbc:mysql://127.0.0.1:3306/salon?serverTimezone=Europe/Moscow&useSSL=false", "root", "")) {

String sn = textFieldSecondName.getText();

String n = textFieldName.getText();

String fn = textFieldFatherName.getText();

String p = textFieldPhone.getText();

String pr = textFieldProssent.getText();

String d = textFieldDate.getText();

int flagSn = 1;

int flagN = 1;

int flagFn = 1;

int flagP = 1;

int flagPr = 1;

int flagD = 1;

if (!isWorld(sn)) {

flagSn = 0;

JOptionPane.showMessageDialog(null, "Введите Фамилию!");

}

if (!isWorld(n)) {

flagN = 0;

JOptionPane.showMessageDialog(null, "Введите Имя!");

}

if (!isWorld(fn)) {

flagFn = 0;

JOptionPane.showMessageDialog(null, "Введите Отчество!");

}

if (!isNumeric(p)) {

flagP = 0;

JOptionPane.showMessageDialog(null, "Введите Телефон!");

}

if (!isDate(d)) {

flagD = 0;

JOptionPane.showMessageDialog(null, "Введите Телефон!");

}

if (flagSn == 1 && flagN == 1 && flagFn == 1 && flagP == 1 && flagPr == 1 && flagD == 1) {

String query = "INSERT INTO `salon`.`worker` (`id\_Worker`, `Second\_Name`, `Name`, `Father\_Name`, `Phone`, `Prossent`, `Start\_Day`) "

+ " VALUES (NULL, '" + sn + "', '" + n + "', '" + fn + "', '" + p + "', '" + pr + "', '"

+ d + "');";

insert it = new insert();

int newId = it.insertRow(query);

if (newId > 0) {

WorkerT\_obj.addRow(

new worker(newId, sn, n, fn, p, Float.parseFloat(textFieldProssent.getText()), d));

}

int row = table.getSelectedRow();

String idWorkerForInsertToTableWorkerService = table.getValueAt(row, 0).toString();

try {

tablePanelForService frame = new tablePanelForService(1);

frame.setVisible(true);

frame.setIDConn(idWorkerForInsertToTableWorkerService);

} catch (Exception e1) {

e1.printStackTrace();

}

}

} catch (Exception ex) {

JOptionPane.showMessageDialog(null, "Connection failed..." + ex);

System.out.println(ex);

}

}

});

panel.setLayout(null);

panel.add(btnAdd);

JScrollPane scrollPane = new JScrollPane();

scrollPane.setBounds(20, 67, 554, 110);

panel.add(scrollPane);

table = new JTable(WorkerT\_obj);

table.getColumnModel().getColumn(1).setPreferredWidth(98);

scrollPane.setViewportView(table);

JLabel lblName = new JLabel("\u0418\u043C\u044F");

lblName.setBounds(110, 185, 36, 14);

panel.add(lblName);

JLabel lblSecondName = new JLabel("\u0424\u0430\u043C\u0438\u043B\u0438\u044F");

lblSecondName.setBounds(20, 185, 70, 14);

panel.add(lblSecondName);

JLabel lblFatherName = new JLabel("\u041E\u0442\u0447\u0435\u0441\u0442\u0432\u043E");

lblFatherName.setBounds(200, 185, 70, 14);

panel.add(lblFatherName);

JLabel lblPhone = new JLabel("\u0422\u0435\u043B\u0435\u0444\u043E\u043D");

lblPhone.setBounds(20, 235, 70, 14);

panel.add(lblPhone);

textFieldName = new JTextField();

textFieldName.setBounds(110, 200, 70, 20);

panel.add(textFieldName);

textFieldName.setColumns(10);

textFieldSecondName = new JTextField();

textFieldSecondName.setBounds(20, 200, 70, 20);

textFieldSecondName.setColumns(10);

panel.add(textFieldSecondName);

textFieldFatherName = new JTextField();

textFieldFatherName.setBounds(200, 200, 70, 20);

textFieldFatherName.setColumns(10);

panel.add(textFieldFatherName);

textFieldPhone = new JTextField();

textFieldPhone.setBounds(20, 250, 70, 20);

textFieldPhone.setColumns(10);

panel.add(textFieldPhone);

JLabel lblProssent = new JLabel("\u041F\u0440\u043E\u0446\u0435\u043D\u0442");

lblProssent.setBounds(110, 235, 70, 14);

panel.add(lblProssent);

textFieldProssent = new JTextField();

textFieldProssent.setBounds(110, 250, 70, 20);

panel.add(textFieldProssent);

textFieldProssent.setColumns(10);

JLabel label = new JLabel("\u0414\u0430\u0442\u0430 \u041F\u0440\u0438\u043D\u044F\u0442\u0438\u044F");

label.setBounds(200, 235, 91, 14);

panel.add(label);

textFieldDate = new JTextField();

textFieldDate.setBounds(200, 250, 70, 20);

panel.add(textFieldDate);

textFieldDate.setColumns(10);

JButton btnUpdate = new JButton("\u0418\u0437\u043C\u0435\u043D\u0438\u0442\u044C");

btnUpdate.setBounds(482, 243, 89, 23);

btnUpdate.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

if (table.getSelectedRow() >= 0) {

String sn = textFieldSecondName.getText();

String n = textFieldName.getText();

String fn = textFieldFatherName.getText();

String p = textFieldPhone.getText();

String pr = textFieldProssent.getText();

String d = textFieldDate.getText();

int flagSn = 1;

int flagN = 1;

int flagFn = 1;

int flagP = 1;

int flagPr = 1;

int flagD = 1;

if (!isWorld(sn)) {

flagSn = 0;

JOptionPane.showMessageDialog(null, "Введите Фамилию!");

}

if (!isWorld(n)) {

flagN = 0;

JOptionPane.showMessageDialog(null, "Введите Имя!");

}

if (!isWorld(fn)) {

flagFn = 0;

JOptionPane.showMessageDialog(null, "Введите Отчество!");

}

if (!isNumeric(p)) {

flagP = 0;

JOptionPane.showMessageDialog(null, "Введите Телефон!");

}

int row = table.getSelectedRow();

int i = (int) table.getValueAt(row, 0);

if (flagSn == 1 && flagN == 1 && flagFn == 1 && flagP == 1 && flagPr == 1 && flagD == 1) {

String query = "UPDATE `salon`.`worker` SET `Second\_Name` = '" + sn + "', `Name` = '" + n

+ "', `Father\_Name` = '" + fn + "', `Phone` = '" + p + "'," + "`Prossent` = '" + pr

+ "',`Start\_Day` = '" + d + "'" + " WHERE `worker`.`id\_Worker` = "

+ table.getValueAt(row, 0).toString() + ";";

insert it = new insert();

int newId = it.insertRow(query);

if (newId > 0) {

WorkerT\_obj.updateRow(table.getSelectedRow(),

new worker(i, sn, n, fn, p, Float.parseFloat(textFieldProssent.getText()), d));

textFieldSecondName.setText(null);

textFieldName.setText(null);

textFieldFatherName.setText(null);

textFieldPhone.setText(null);

textFieldProssent.setText(null);

textFieldDate.setText(null);

}

}

}

}

});

panel.add(btnUpdate);

JButton btnDelete = new JButton("\u0423\u0434\u0430\u043B\u0438\u0442\u044C");

btnDelete.setBounds(482, 277, 89, 23);

btnDelete.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

int row = table.getSelectedRow();

int i = (int) table.getValueAt(row, 0);

String query = "DELETE FROM `salon`. `worker` WHERE `worker`.`id\_Worker` = "

+ table.getValueAt(row, 0).toString() + ";";

insert it = new insert();

int newId = it.insertRow(query);

if (newId > 0) {

WorkerT\_obj.deleteRow(i);

}

}

});

panel.add(btnDelete);

textFieldForSearch = new JTextField();

textFieldForSearch.setBounds(20, 11, 452, 20);

panel.add(textFieldForSearch);

textFieldForSearch.setColumns(10);

JButton btnSearch = new JButton("\u041D\u0430\u0439\u0442\u0438");

btnSearch.setBounds(482, 10, 89, 23);

btnSearch.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

String search = textFieldForSearch.getText();

try (Connection connection = DriverManager.getConnection(

"jdbc:mysql://127.0.0.1:3306/salon?serverTimezone=Europe/Moscow&useSSL=false", "root", "")) {

PreparedStatement statement = connection

.prepareStatement("SELECT \* FROM `worker` " + "WHERE `Second\_Name` LIKE '" + search

+ "' OR `Name` LIKE '" + search + "' OR `" + "Father\_Name` LIKE '" + search

+ "' OR `Phone` LIKE '" + search + "'" + "OR `Prossent` LIKE '" + search + "'");

ResultSet res = statement.executeQuery();

// if (!res.next()) {

// JOptionPane.showMessageDialog(null, "Ничего не найдено...");

// }

WorkerT\_obj.deleteAll();

while (res.next()) {

WorkerT\_obj.addRow(new worker(res.getInt(1), res.getString(2), res.getString(3),

res.getString(4), res.getString(5), res.getFloat(6), res.getString(7)));

}

} catch (Exception ex) {

JOptionPane.showMessageDialog(null, "Connection failed..." + ex);

System.out.println(ex);

}

}

});

panel.add(btnSearch);

JButton btnCancel = new JButton("\u041E\u0442\u043C\u0435\u043D\u0430");

btnCancel.setBounds(482, 34, 89, 23);

btnCancel.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

WorkerT\_obj.deleteAll();

textFieldForSearch.setText(null);

try (Connection connection = DriverManager.getConnection(

"jdbc:mysql://127.0.0.1:3306/salon?serverTimezone=Europe/Moscow&useSSL=false", "root", "")) {

PreparedStatement statement = connection.prepareStatement("SELECT \* \r\n" + "FROM `worker`");

ResultSet res = statement.executeQuery();

while (res.next()) {

WorkerT\_obj.addRow(new worker(res.getInt(1), res.getString(2), res.getString(3),

res.getString(4), res.getString(5), res.getFloat(6), res.getString(7)));

}

} catch (Exception ex) {

JOptionPane.showMessageDialog(null, "Connection failed..." + ex);

System.out.println(ex);

}

}

});

panel.add(btnCancel);

JButton btnSelect = new JButton("\u0412\u044B\u0431\u0440\u0430\u0442\u044C");

btnSelect.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

int row = table.getSelectedRow();

String idWorkerForInsertToTableWorkerService = table.getValueAt(row, 0).toString();

try {

tablePanelForService frame = new tablePanelForService(1);

frame.setVisible(true);

frame.setIDConn(idWorkerForInsertToTableWorkerService);

} catch (Exception e1) {

e1.printStackTrace();

}

}

});

btnSelect.setBounds(356, 249, 89, 23);

panel.add(btnSelect);

table.addMouseListener(new MouseListener() {

public void mouseClicked(MouseEvent e) {

int row = table.getSelectedRow();

textFieldSecondName.setText(table.getValueAt(row, 1).toString());

textFieldName.setText(table.getValueAt(row, 2).toString());

textFieldFatherName.setText(table.getValueAt(row, 3).toString());

textFieldPhone.setText(table.getValueAt(row, 4).toString());

textFieldProssent.setText(table.getValueAt(row, 5).toString());

textFieldDate.setText(table.getValueAt(row, 6).toString());

}

@Override

public void mouseEntered(MouseEvent e) {

// TODO Auto-generated method stub

}

@Override

public void mouseExited(MouseEvent e) {

// TODO Auto-generated method stub

}

@Override

public void mousePressed(MouseEvent e) {

// TODO Auto-generated method stub

}

@Override

public void mouseReleased(MouseEvent e) {

// TODO Auto-generated method stub

}

});

try (Connection connection = DriverManager.getConnection(

"jdbc:mysql://127.0.0.1:3306/salon?serverTimezone=Europe/Moscow&useSSL=false", "root", "")) {

PreparedStatement statement = connection.prepareStatement("SELECT \* \r\n" + "FROM `worker`");

ResultSet res = statement.executeQuery();

while (res.next()) {

WorkerT\_obj.addRow(new worker(res.getInt(1), res.getString(2), res.getString(3), res.getString(4),

res.getString(5), res.getFloat(6), res.getString(7)));

}

} catch (Exception ex) {

JOptionPane.showMessageDialog(null, "Connection failed..." + ex);

System.out.println(ex);

}

table.getColumnModel().getColumn(0).setMinWidth(0);

table.getColumnModel().getColumn(0).setMaxWidth(0);

table.getColumnModel().getColumn(0).setWidth(0);

}

}

tablePanelForWorker.java

**package** ud;

**public** **class** user {

**protected** String login;

**protected** String password;

**public** user (String login\_, String password\_) {

login=login\_;

password=password\_;

}

**public** String getLogin() {

**return** login;

}

**public** String getPassword() {

**return** password;

}

**public** **void** setLogin(String login) {

**this**.login = login;

}

**public** **void** setPassword(String password) {

**this**.password = password;

}

}

user.java

**package** ud;

**public** **class** user {

**protected** String login;

**protected** String password;

**public** user (String login\_, String password\_) {

login=login\_;

password=password\_;

}

**public** String getLogin() {

**return** login;

}

**public** String getPassword() {

**return** password;

}

**public** **void** setLogin(String login) {

**this**.login = login;

}

**public** **void** setPassword(String password) {

**this**.password = password;

}

}

userTableModel.java

package ud;

import java.util.Iterator;

import java.util.LinkedList;

import javax.swing.table.AbstractTableModel;

public class userTableModel extends AbstractTableModel{

private static final long serialVersionUID = 7927259757559420606L;

private LinkedList<user> UserT\_obj;

public userTableModel(LinkedList<user> UserT\_obj) {

this.UserT\_obj = UserT\_obj;

}

public Class<?> getColumnClass(int columnIndex) {

return getValueAt(0, columnIndex).getClass();

}

public int getColumnCount() {

return 2;

}

public String getColumnName(int columnIndex) {

switch (columnIndex) {

case 0:

return "Логин";

case 1:

return "Пароль";

}

return "";

}

public void addRow(user nr) { // добавление строки в таблицу

UserT\_obj.add(nr);

fireTableDataChanged();// вызываем для обновления таблицы

// выделяем добавленную строку - иначе после обновления таблицы теряет выделение

int index = UserT\_obj.indexOf(nr);

tablePanelForUser.table.setRowSelectionInterval(index, index);

}

public int getRowCount() {

return UserT\_obj.size();

}

public Object getValueAt(int rowIndex, int columnIndex) {

user p = UserT\_obj.get(rowIndex);

switch (columnIndex) {

case 0:

return p.getLogin();

case 1:

return p.getPassword();

}

return "";

}

public boolean isCellEditable(int rowIndex, int columnIndex) {

return false;

}

public void setValueAt(Object value, int rowIndex, int columnIndex) {

}

public void updateRow(int index, user nr) {

UserT\_obj.set(index, nr);

fireTableDataChanged();

tablePanelForUser.table.setRowSelectionInterval(index, index);

}

public void deleteRow(String login) {

Iterator<user> isph = UserT\_obj.iterator();

boolean flag = false;

while (isph.hasNext()) {

if (isph.next().getLogin().equals(login)) {

isph.remove();

flag = true;

break;

}

}

if (flag) {

fireTableDataChanged();

} else

;

}

public void deleteAll() { // удаление строки из таблицы

int size = UserT\_obj.size();

UserT\_obj.clear();

fireTableRowsDeleted(0, size);

}

}

visitAdd.java

package ud;

import java.awt.EventQueue;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

import javax.swing.JFrame;

import javax.swing.JPanel;

import javax.swing.border.EmptyBorder;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JTextField;

import javax.swing.JComboBox;

import javax.swing.JButton;

import javax.swing.JTextPane;

public class visitAdd extends JFrame {

/\*\*

\*

\*/

private static final long serialVersionUID = 1L;

private JPanel contentPane;

private JTextField textFieldSecondNameCustomer;

private JTextField textFieldNameCustomer;

private JTextField textFieldDay;

public String idCustomerSelected;

public String idServiceSelected;

public String idWorkerSelected;

public String daySelected;

public String serviceIdSelected;

public String workerIdSelected;

public String timeSelected ;

public String serviceNameWorkerNameSelected;

/\*\*

\* Launch the application.

\*/

public static void main(String[] args) {

EventQueue.invokeLater(new Runnable() {

public void run() {

try {

visitAdd frame = new visitAdd();

frame.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

});

}

public static boolean isWorld(String x)

{

Pattern p = Pattern.compile("^[а-яА-ЯёЁ]+$");

Matcher m = p.matcher(x);

return m.matches();

}

public static boolean isDate(String x)

{

Pattern p = Pattern.compile("^20\\d{2}-\\d{2}-\\d{2}$");

Matcher m = p.matcher(x);

return m.matches();

}

/\*\*

\* Create the frame.

\*/

public void checkFio() {

if(textFieldNameCustomer.getText()!=null && textFieldSecondNameCustomer.getText()!=null)

{

String sn=textFieldSecondNameCustomer.getText();

String n=textFieldNameCustomer.getText();

int flagSn = 1;

int flagN = 1;

if (!isWorld(sn)) {

flagSn = 0;

JOptionPane.showMessageDialog(null, "Введите Фамилию!");

}

if (!isWorld(n)) {

flagN = 0;

JOptionPane.showMessageDialog(null, "Введите Имя!");

}

if (flagSn == 1 && flagN == 1) {

try (Connection connection = DriverManager.getConnection("jdbc:mysql://127.0.0.1:3306/salon?serverTimezone=Europe/Moscow&useSSL=false", "root",""))

{

PreparedStatement statement = connection.prepareStatement("SELECT `id\_Customer` \r\n" +

"FROM `customer`"

+ "where `Second\_Name`='"+sn+"'AND `Name`='"+n+"'");

ResultSet rest = statement.executeQuery();

while(rest.next()) {

idCustomerSelected=rest.getString(1);

}

} catch (Exception ex) {

JOptionPane.showMessageDialog(null, "Connection failed..."+ex);

System.out.println(ex);

}

}

}

}

public int checkDateTime() {

String d=textFieldDay.getText();

String flagForCheckExistenceTimeAndDate=null;

int flag=0;

int flagD=1;if (!isDate(d)) {

flagD = 0;

JOptionPane.showMessageDialog(null, "Введите Телефон!");

}

if (flagD == 1) {

try (Connection connection = DriverManager.getConnection("jdbc:mysql://127.0.0.1:3306/salon?serverTimezone=Europe/Moscow&useSSL=false", "root",""))

{

PreparedStatement statement = connection.prepareStatement("SELECT \* FROM `record` "

+ "WHERE `id\_Service` = '"+idServiceSelected+"' AND `id\_Worker` = '"+idWorkerSelected+"' AND `Date` = '"+textFieldDay.getText()+"' AND `Hour` = '"+timeSelected+"'");

ResultSet rest = statement.executeQuery();

while(rest.next()) {

flagForCheckExistenceTimeAndDate=rest.getString(1);

flag=1;

}

} catch (Exception ex) {

JOptionPane.showMessageDialog(null, "Connection failed..."+ex+flagForCheckExistenceTimeAndDate);

System.out.println(ex);

}

}

return flag;

}

public visitAdd() {

setDefaultCloseOperation(JFrame.HIDE\_ON\_CLOSE);

setBounds(100, 100, 582, 222);

contentPane = new JPanel();

contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

setContentPane(contentPane);

contentPane.setLayout(null);

JLabel lblDay = new JLabel("\u0414\u0430\u0442\u0430");

lblDay.setBounds(321, 11, 31, 14);

contentPane.add(lblDay);

JLabel lblTime = new JLabel("\u0412\u0440\u0435\u043C\u044F");

lblTime.setBounds(250, 42, 46, 14);

contentPane.add(lblTime);

JLabel lblSecondNameCustomer = new JLabel("\u0424\u0430\u043C\u0438\u043B\u0438\u044F");

lblSecondNameCustomer.setBounds(10, 11, 46, 14);

contentPane.add(lblSecondNameCustomer);

JLabel lblNameCustomer = new JLabel("\u0418\u043C\u044F");

lblNameCustomer.setBounds(171, 11, 31, 14);

contentPane.add(lblNameCustomer);

textFieldSecondNameCustomer = new JTextField();

textFieldSecondNameCustomer.setBounds(65, 8, 86, 20);

contentPane.add(textFieldSecondNameCustomer);

textFieldSecondNameCustomer.setColumns(10);

textFieldNameCustomer = new JTextField();

textFieldNameCustomer.setBounds(210, 8, 86, 20);

contentPane.add(textFieldNameCustomer);

textFieldNameCustomer.setColumns(10);

textFieldDay = new JTextField();

textFieldDay.setBounds(362, 8, 86, 20);

contentPane.add(textFieldDay);

textFieldDay.setColumns(10);

JComboBox<String> comboBoxServiceWorker = new JComboBox<String>();

comboBoxServiceWorker.setBounds(10, 67, 230, 20);

contentPane.add(comboBoxServiceWorker);

try (Connection connection = DriverManager.getConnection("jdbc:mysql://127.0.0.1:3306/salon?serverTimezone=Europe/Moscow&useSSL=false", "root",""))

{

PreparedStatement statement = connection.prepareStatement("SELECT `sevice`.`Name`, `worker`.`Second\_Name`, `worker`.`Name` FROM `sevice` \r\n" +

"inner join `service\_worker` \r\n" +

"on (`sevice`.`id\_Service`=`service\_worker`.`id\_Service`) \r\n" +

"inner join `worker` \r\n" +

"on (`worker`.`id\_Worker`=`service\_worker`.`id\_worker`)\r\n" +

"\r\n" +

"");

ResultSet rest = statement.executeQuery();

while(rest.next()) {

String concat= rest.getString(1) +" "+rest.getString(2)+" "+ rest.getString(3);

comboBoxServiceWorker.addItem(concat);

}

} catch (Exception ex) {

JOptionPane.showMessageDialog(null, "Connection failed..."+ex);

System.out.println(ex);

}

comboBoxServiceWorker.setSelectedItem(null);

JLabel lblServiceWorker = new JLabel("\u0423\u0441\u043B\u0443\u0433\u0430");

lblServiceWorker.setBounds(10, 42, 46, 14);

contentPane.add(lblServiceWorker);

JComboBox<String> comboBoxTime = new JComboBox<String>();

comboBoxTime.setBounds(250, 67, 63, 20);

contentPane.add(comboBoxTime);

comboBoxTime.addItem("8");

comboBoxTime.addItem("10");

comboBoxTime.addItem("12");

comboBoxTime.addItem("14");

comboBoxTime.addItem("16");

comboBoxTime.addItem("18");

comboBoxTime.setSelectedItem(null);

comboBoxTime.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

@SuppressWarnings("unchecked")

JComboBox<String> comboBoxTime = (JComboBox<String>)e.getSource();

timeSelected = (String)comboBoxTime.getSelectedItem();

JOptionPane.showMessageDialog(null, timeSelected);

}

});

comboBoxServiceWorker.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

@SuppressWarnings("unchecked")

JComboBox<String> comboBoxService = (JComboBox<String>)e.getSource();

serviceNameWorkerNameSelected = (String)comboBoxService.getSelectedItem();

JOptionPane.showMessageDialog(null, serviceNameWorkerNameSelected);

String str []=serviceNameWorkerNameSelected.split(" ");

String s=str[0];

String sn=str[1];

String n=str[2];

try (Connection connection = DriverManager.getConnection("jdbc:mysql://127.0.0.1:3306/salon?serverTimezone=Europe/Moscow&useSSL=false", "root",""))

{

PreparedStatement statement = connection.prepareStatement("SELECT `service\_worker`.`id\_Service`, `service\_worker`.`id\_Worker` FROM `service\_worker` \r\n" +

"inner join `sevice` \r\n" +

"on (`sevice`.`id\_Service`=`service\_worker`.`id\_Service` and `sevice`.`Name`='"+s+"' ) \r\n" +

"inner join `worker` \r\n" +

"on (`worker`.`id\_Worker`=`service\_worker`.`id\_worker` and `worker`.`Name`='"+n+"' and `worker`.`Second\_Name`='"+sn+"' )\r\n" +

"");

ResultSet rest = statement.executeQuery();

while(rest.next()) {

idServiceSelected=rest.getString(1);

idWorkerSelected=rest.getString(2);

}

} catch (Exception ex) {

JOptionPane.showMessageDialog(null, "Connection failed..."+ex);

System.out.println(ex);

}

}

});

JButton btnWriteRecord = new JButton("\u0417\u0430\u043F\u0438\u0441\u0430\u0442\u044C");

btnWriteRecord.setBounds(467, 149, 89, 23);

btnWriteRecord.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

try (Connection connection = DriverManager.getConnection("jdbc:mysql://127.0.0.1:3306/salon?serverTimezone=Europe/Moscow&useSSL=false", "root",""))

{

String d=textFieldDay.getText();

int flagD=1;

int flagC=1;

int flagS=1;

int flagW=1;

String flagForCheckExistenceRecord=null;

if (!isDate(d)) {

flagD = 0;

JOptionPane.showMessageDialog(null, "Введите Телефон!");

}

checkFio();

if(idCustomerSelected==null)

{

flagC=0;

JOptionPane.showMessageDialog(null, "Такого клиента не существует.");

}

if(idServiceSelected==null)

{

flagS=0;

JOptionPane.showMessageDialog(null, "Выберете услугу!");

}

if(idWorkerSelected==null)

{

flagW=0;

JOptionPane.showMessageDialog(null, "Выберете услFFFугу!");

}

int t=checkDateTime();

if (flagD==1 && flagC==1 && flagS==1 &&flagW==1 && t==0) {

PreparedStatement statement = connection.prepareStatement("SELECT `Status` FROM `record` where "

+ "`id\_Customer`='"+idCustomerSelected+"' "

+ "and `id\_Service`='"+idServiceSelected+"' and `id\_Worker`= '"+idWorkerSelected+"'and `Date` = '"+d+"'and `Hour`='"+timeSelected+"'");

ResultSet rest = statement.executeQuery();

while(rest.next()) {

flagForCheckExistenceRecord=rest.getString(1);

}

if(flagForCheckExistenceRecord==null) {

insert it = new insert();

it.insertRow("INSERT INTO `salon`.`record` \r\n" +

"(`id\_Customer` , `id\_Service` , `id\_Worker` , `Date` , `Hour` , `Status`)\r\n" +

"VALUES (\r\n" +

"'"+idCustomerSelected+"', '"+idServiceSelected+"', '"+idWorkerSelected+"', '"+d+"', '"+timeSelected+"', '0')");

String str []=serviceNameWorkerNameSelected.split(" ");

String serviceTemp=str[0];

String workerSNameTemp=str[1];

String workerNameTemp=str[2];

JTextPane textPane = new JTextPane();

textPane.setBounds(10, 108, 447, 80);

contentPane.add(textPane);

textPane.setText("Клиент:"+textFieldSecondNameCustomer.getText()+" "+textFieldNameCustomer.getText()

+" \nУслуга: "+serviceTemp+" \n"+"Сотрудник: "+workerSNameTemp+" "

+workerNameTemp+"\n Дата:"+textFieldDay.getText()

+"\nВремя:"+timeSelected);

}

else if(flagForCheckExistenceRecord!=null) {

JOptionPane.showMessageDialog(null, "Это время занято! Выберете другое.");

}

}

else {

JOptionPane.showMessageDialog(null, "Это время занято! Выберете другое345434r3t.");

}

} catch (Exception ex) {

JOptionPane.showMessageDialog(null, "Connection failed..."+ex);

System.out.println(ex);

}

dispose();

}

});

contentPane.add(btnWriteRecord);

}

}

worker java

**package** ud;

**public** **class** worker **implements** Comparable<worker> {

**protected** **int** id;

**protected** String secondName;

**protected** String name;

**protected** String fatherName;

**protected** String phone;

**protected** **float** prossent;

**protected** String date;

**public** worker(**int** id\_, String secondName\_, String name\_, String fatherName\_, String phone\_, **float** prossent\_,

String date\_) {

id = id\_;

secondName = secondName\_;

name = name\_;

fatherName = fatherName\_;

phone = phone\_;

prossent = prossent\_;

date=date\_;

}

**public** **int** getID() {

**return** id;

}

**public** String getSecondName() {

**return** secondName;

}

**public** String getName() {

**return** name;

}

**public** String getFatherName() {

**return** fatherName;

}

**public** String getPhone() {

**return** phone;

}

**public** **float** getProssent() {

**return** prossent;

}

**public** String getDate() {

**return** date;

}

**public** **void** setID(**int** id) {

**this**.id = id;

}

**public** **void** setSecondName(String secondName) {

**this**.secondName = secondName;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** **void** setFatherName(String fatherName) {

**this**.fatherName = fatherName;

}

**public** **void** setPhone(String phone) {

**this**.phone = phone;

}

**public** **void** setProssent(**float** prossent) {

**this**.prossent = prossent;

}

**public** **void** setDate(String date) {

**this**.date = date;

}

@Override

**public** **int** compareTo(worker c) {

**return** **this**.secondName.compareTo(c.secondName);

}

@Override

**public** String toString() {

**return** secondName +" "+name;

}

}

WorkerTableModel.java

package ud;

import java.util.Iterator;

import java.util.LinkedList;

import javax.swing.table.AbstractTableModel;

public class WorkerTableModel extends AbstractTableModel {

private static final long serialVersionUID = 7927259757559420606L;

private LinkedList<worker> WorkerT\_obj;

public WorkerTableModel(LinkedList<worker> WorkerT\_obj) {

this.WorkerT\_obj = WorkerT\_obj;

}

public Class<?> getColumnClass(int columnIndex) {

return getValueAt(0, columnIndex).getClass();

}

public int getColumnCount() {

return 7;

}

public String getColumnName(int columnIndex) {

switch (columnIndex) {

case 0:

return "ID";

case 1:

return "Фамилия";

case 2:

return "Имя";

case 3:

return "Отчество";

case 4:

return "Телефон";

case 5:

return "Процент";

case 6:

return "ДтПрнт";

}

return "";

}

public void addRow(worker nr) { // добавление строки в таблицу

WorkerT\_obj.add(nr);

fireTableDataChanged();// вызываем для обновления таблицы

// выделяем добавленную строку - иначе после обновления таблицы теряет выделение

int index = WorkerT\_obj.indexOf(nr);

tablePanelForWorker.table.setRowSelectionInterval(index, index);

}

public int getRowCount() {

return WorkerT\_obj.size();

}

public Object getValueAt(int rowIndex, int columnIndex) {

worker p = WorkerT\_obj.get(rowIndex);

switch (columnIndex) {

case 0:

return p.getID();

case 1:

return p.getSecondName();

case 2:

return p.getName();

case 3:

return p.getFatherName();

case 4:

return p.getPhone();

case 5:

return p.getProssent();

case 6:

return p.getDate();

}

return "";

}

public boolean isCellEditable(int rowIndex, int columnIndex) {

return false;

}

public void setValueAt(Object value, int rowIndex, int columnIndex) {

}

public void updateRow(int index, worker nr) {

WorkerT\_obj.set(index, nr);

fireTableDataChanged();

tablePanelForWorker.table.setRowSelectionInterval(index, index);

}

public void deleteRow(int id) {

Iterator<worker> isph = WorkerT\_obj.iterator();

boolean flag = false;

while (isph.hasNext()) {

if (isph.next().getID()==id) {

isph.remove();

flag = true;

break;

}

}

if (flag) {

fireTableDataChanged();

} else

;

}

public void deleteAll() { // удаление строки из таблицы

int size = WorkerT\_obj.size();

WorkerT\_obj.clear();

fireTableRowsDeleted(0, size);

}

}

**Приложение Б**

**Запросы к базе данных**

Примеры запросов к базе данных:

* запрос на вывод фамилии и имени сотрудников;

SELECT `second\_name` , `name`

FROM `worker`

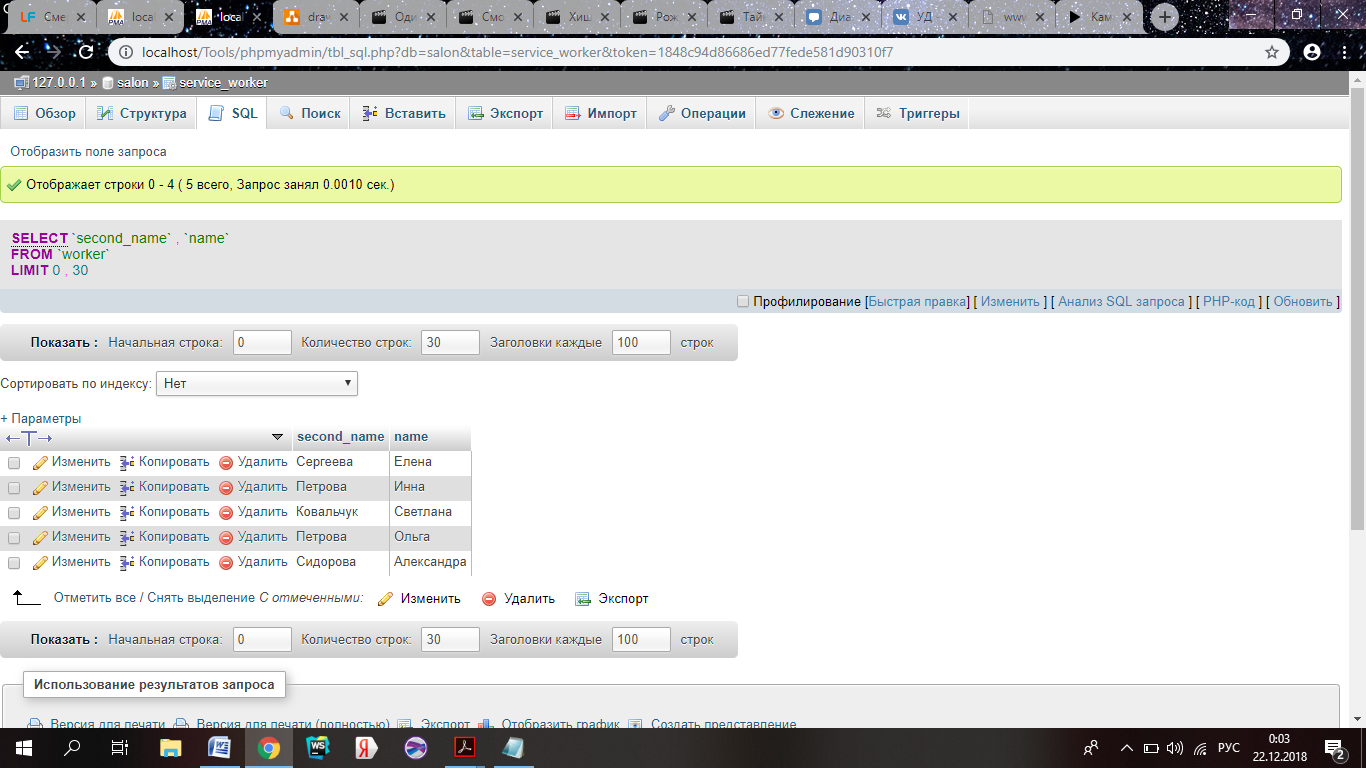


Рисунок Б.1 –Запрос на вывод фамилии и имени сотрудников

* запрос на выборку всех сотрудников, принятых на работу с 2018 года;

SELECT \* FROM `worker`

WHERE `Start\_Day` > '2018-01-01'

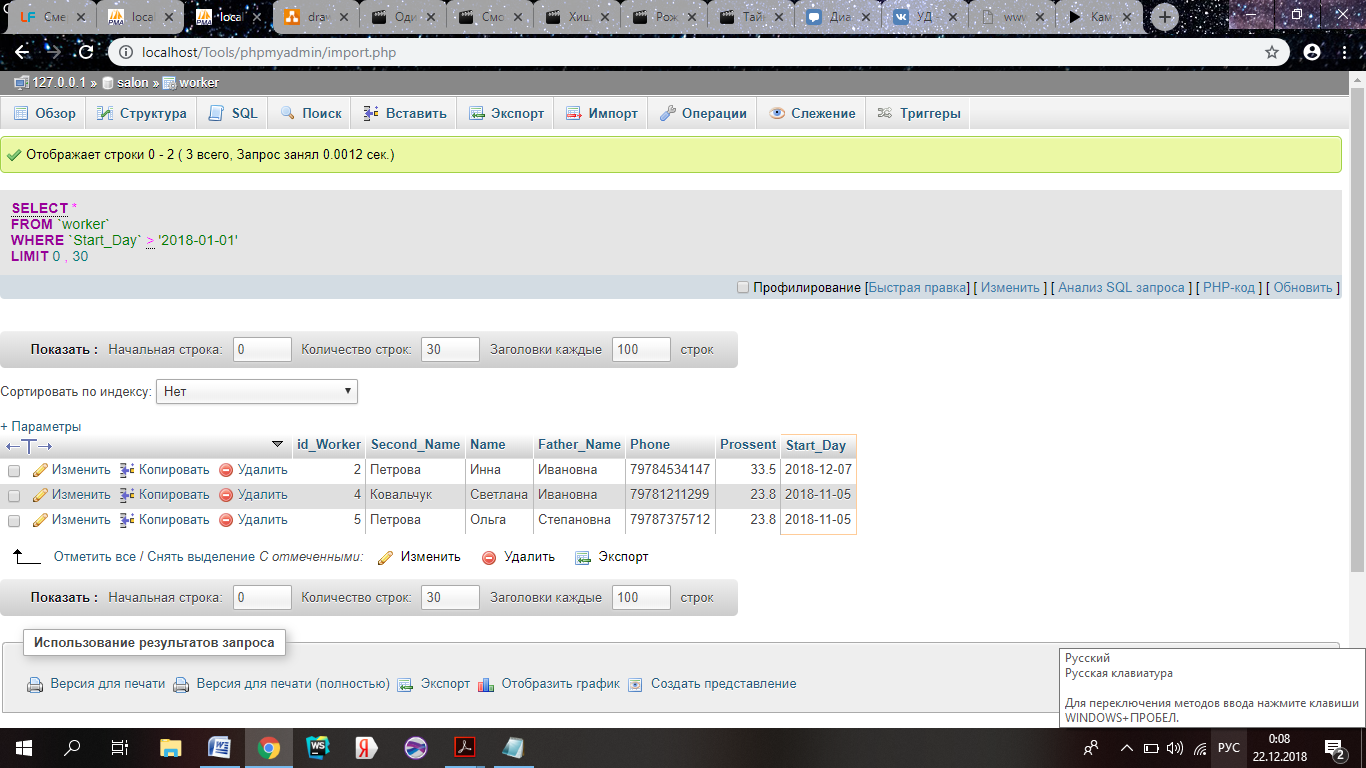


Рисунок Б.2 – Запрос на выборку всех сотрудников, принятых на работу после 2018-01-01

* запрос на выборку услуг стоимостью больше 1000 рублей и длительностью больше 90 минут;

SELECT \* FROM `sevice` WHERE `Cost` >1000 AND `Time` >90

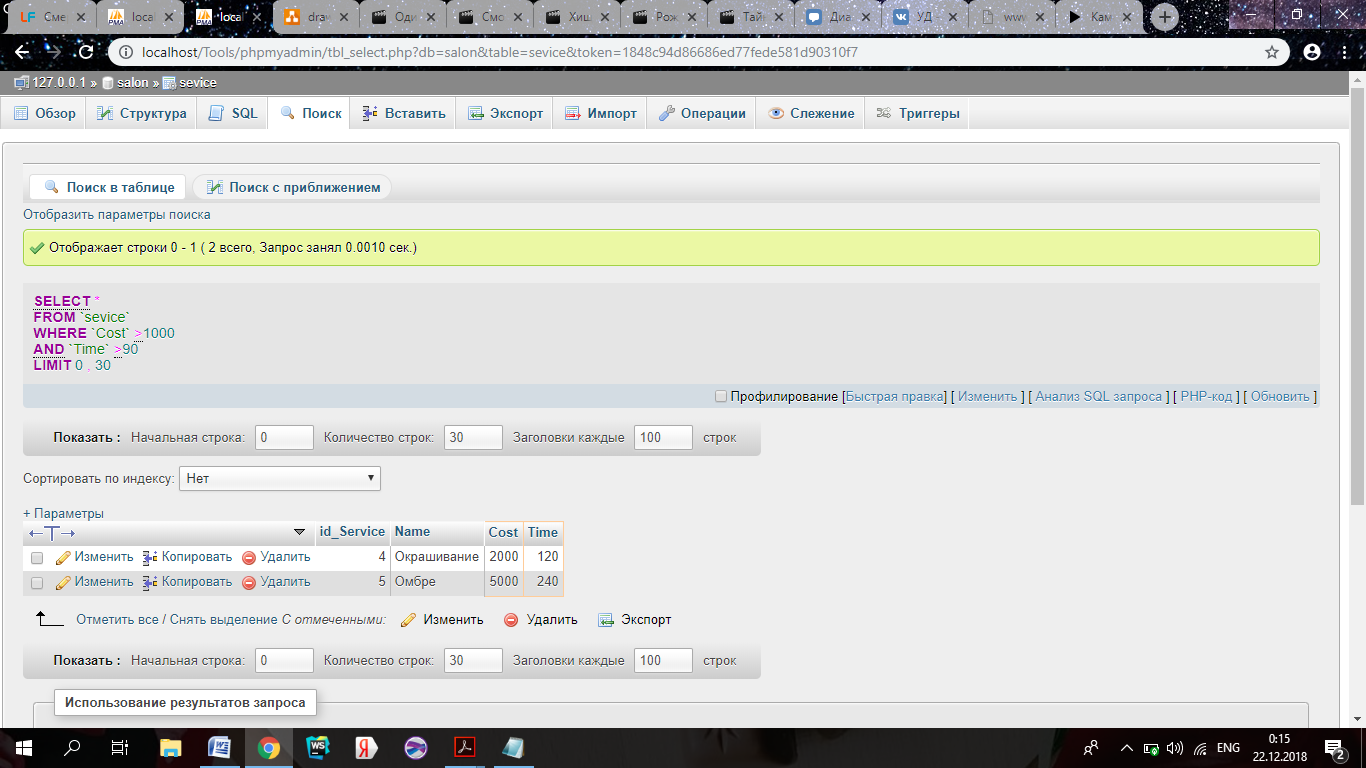


Рисунок Б.3 – Запрос на выборку услуг стоимостью больше 1000 рублей и длительностью больше 90 минут

* запрос на выборку сотрудников, осуществляющих услугу Маникюр;

SELECT `worker`.`Second\_Name`, `worker`.`Name` FROM `worker`

join `service\_worker` on

(`worker`.`id\_worker`=`service\_worker`.`id\_worker`)

join `sevice` on

(`service\_worker`.`id\_service`=`sevice`.`id\_service` and `sevice`.`Name`='Маникюр')

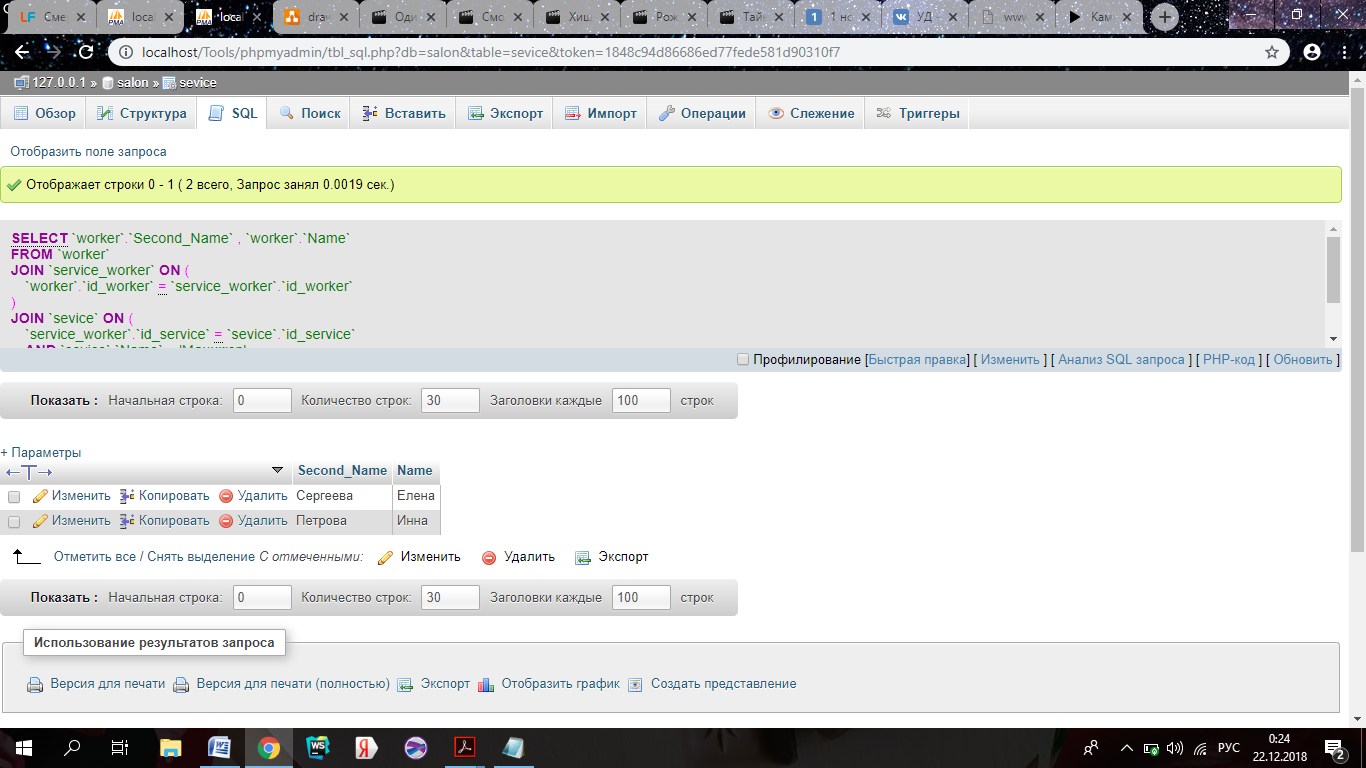


Рисунок Б.4 – запрос на выборку сотрудников, осуществляющих услугу Маникюр

* запрос на выборку запланированных посещений на 24 декабря 2018 года;

SELECT `customer`.`Second\_Name`, `customer`.`Name`, `customer`.`Father\_Name`, `customer`.`Phone`, `worker`.`Second\_Name`, `worker`.`Name`,`worker`.`Father\_Name`,`worker`.`Phone`, `sevice`.`Name`,`record`.`date`,`record`.`hour`

FROM `customer`

inner join `record`

on(`customer`.`id\_Customer`=`record`.`id\_Customer`and`record`.`date`='2018-12-24')

inner join `service\_worker`

on(`record`.`id\_Worker`=`service\_worker`.`id\_worker`and`record`.`id\_Service` = `service\_worker`.`id\_Service`)

inner join `worker`

on (`worker`.`id\_Worker`=`service\_worker`.`id\_worker`)

inner join `sevice`

on(`sevice`.`id\_Service`=`service\_worker`.`id\_Service`)

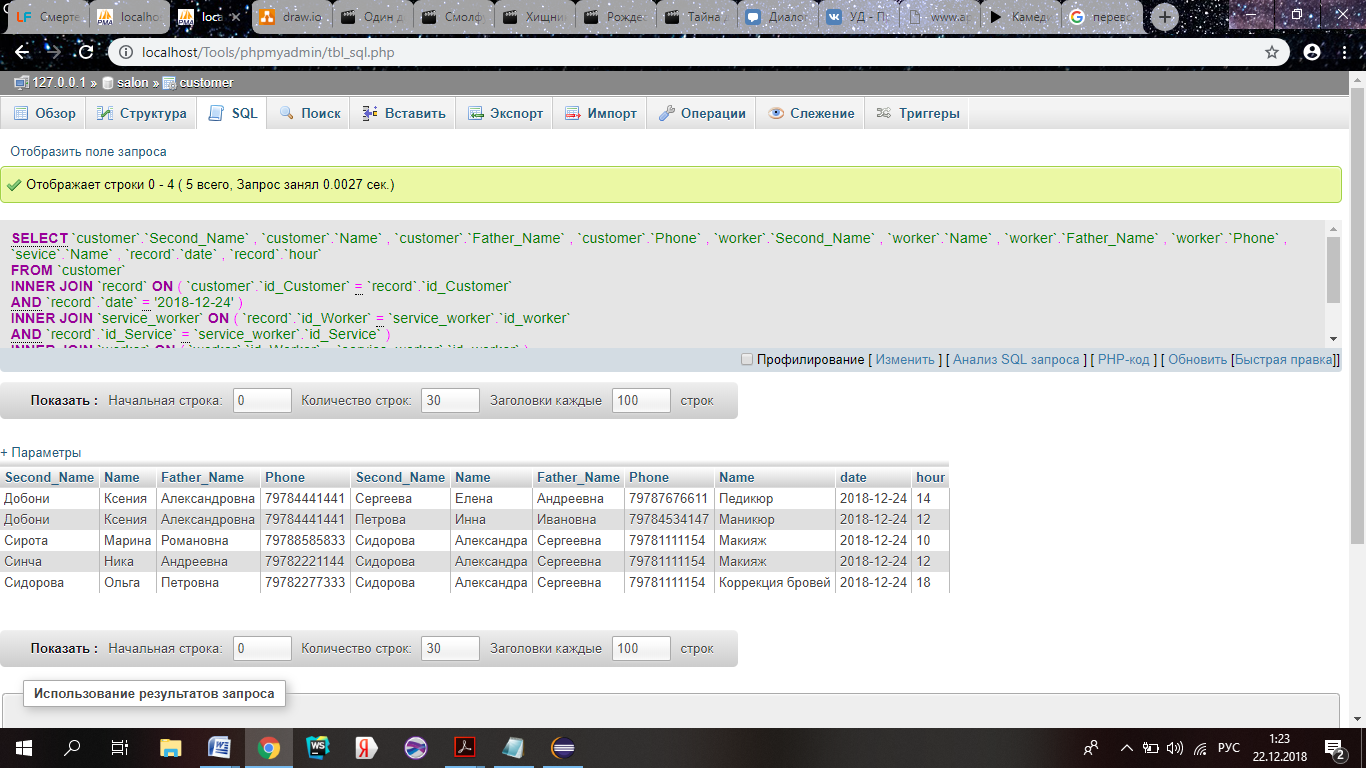


Рисунок Б.5 – Запрос на выборку запланированных посещений на 24 декабря 2018 года

* запрос на выборку запланированных посещений на 24 декабря 2018 года у сотрудника Сидоровой;

SELECT `customer`.`Second\_Name`, `customer`.`Name`, `customer`.`Father\_Name`, `customer`.`Phone`,

`worker`.`Second\_Name`, `worker`.`Name`, `worker`.`Father\_Name`, `worker`.`Phone`,

`sevice`.`Name`,

`record`.`date`,`record`.`hour`

FROM `customer`

inner join `record`

on(`customer`.`id\_Customer`=`record`.`id\_Customer`and`record`.`date`='2018-12-24')

inner join `service\_worker`

on(`record`.`id\_Worker`=`service\_worker`.`id\_worker`and record`.`id\_Service` = `service\_worker`.`id\_Service`)

inner join `worker`

on(`worker`.`id\_Worker`=`service\_worker`.`id\_worker`and `worker`.`second\_name` = 'Сидорова')

inner join `sevice`

on(`sevice`.`id\_Service`=`service\_worker`.`id\_Service`)

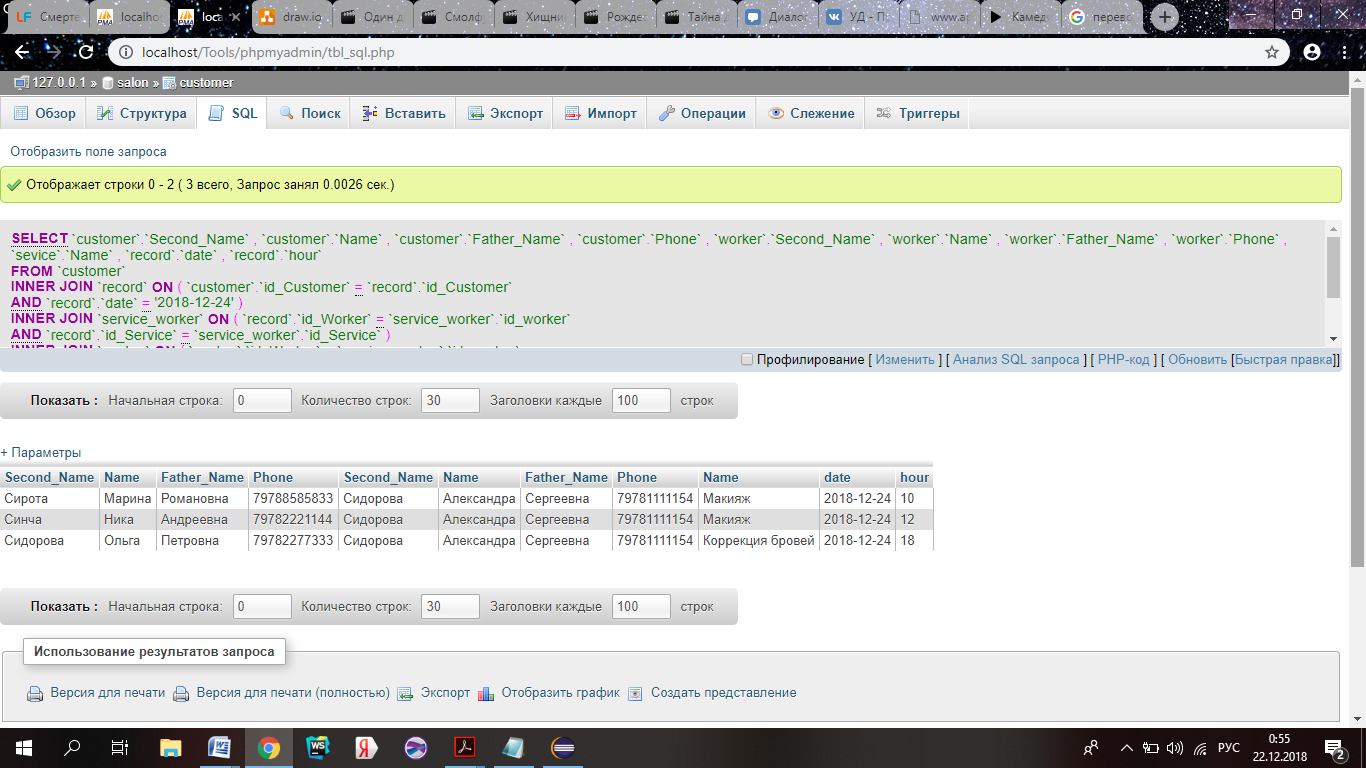


Рисунок Б.6 – Запрос на выборку запланированных посещений на 24 декабря 2018 года у сотрудника Сидоровой

* запрос на выборку запланированных посещений на 24 декабря 2018 года у сотрудника Сидоровой на услугу Макияж

SELECT `customer`.`Second\_Name`, `customer`.`Name`, `customer`.`Father\_Name`, `customer`.`Phone`, `worker`.`Second\_Name`, `worker`.`Name`, `worker`.`Father\_Name`, `worker`.`Phone`, `sevice`.`Name`,`record`.`date`,`record`.`hour`

FROM `customer` inner join `record`on (`customer`.`id\_Customer`=`record`.`id\_Customer` and `record`.`date`='2018-12-24')

inner join `service\_worker`

on(`record`.`id\_Worker`=`service\_worker`.`id\_worker` and`record`.`id\_Service`=`service\_worker`.`id\_Service`)

inner join `worker`

on(`worker`.`id\_Worker`=`service\_worker`.`id\_worker`

and `worker`.`second\_name`='Сидорова')

inner join `sevice`

on(`sevice`.`id\_Service`=`service\_worker`.`id\_Service`

and `sevice`.`name`='макияж')

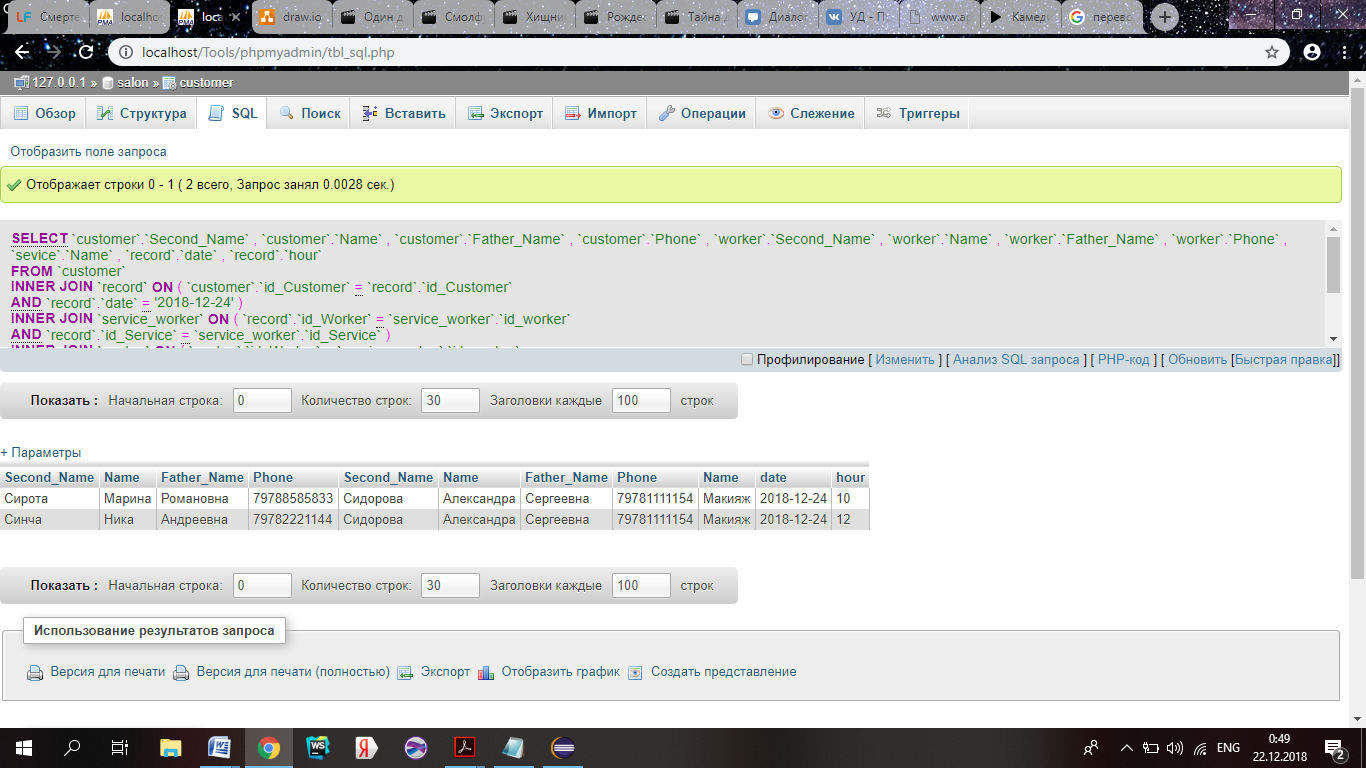


Рисунок Б.7 – Запрос на выборку запланированных посещений на 24 декабря 2018 года у сотрудника Сидоровой на услугу Макияж

* запрос на выбор всех данных обо всех посещениях;

SELECT `customer`.`Second\_Name`, `customer`.`Name`, `customer`.`Father\_Name`, `customer`.`Phone`, `worker`.`Second\_Name`, `worker`.`Name`, `worker`.`Father\_Name`, `worker`.`Phone`, `sevice`.`Name`,`record`.`date`,`record`.`hour`

FROM `customer`

inner join `record`on (`customer`.`id\_Customer`=`record`.`id\_Customer`)

inner join `service\_worker`

on(`record`.`id\_Worker`=`service\_worker`.`id\_worker` and`record`.`id\_Service`=`service\_worker`.`id\_Service`)

inner join `worker`

on(`worker`.`id\_Worker`=`service\_worker`.`id\_worker`)

inner join `sevice`

on(`sevice`.`id\_Service`=`service\_worker`.`id\_Service`)

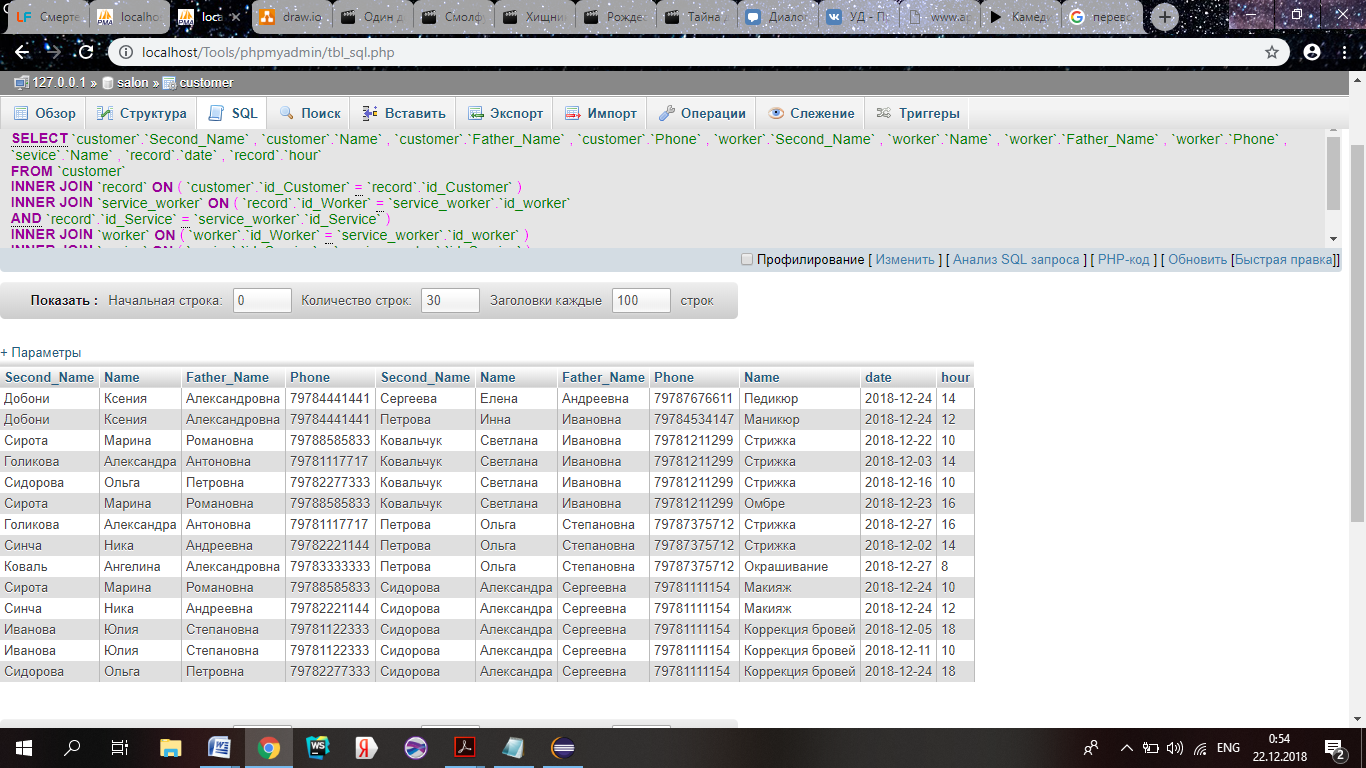


Рисунок Б.8 – Запрос на выбор всех данных обо всех посещениях

* запрос на выбор информации о посещениях услуги Стрижка клиентом Сирота;

SELECT `customer`.`Second\_Name`, `customer`.`Name`, `customer`.`Father\_Name`, `customer`.`Phone`, `sevice`.`Name`,`record`.`date`,`record`.`hour`

FROM `customer` inner join `record`on (`customer`.`id\_Customer`=`record`.`id\_Customer` and `customer`.`Second\_name`='Сирота')

inner join `service\_worker`on (`record`.`id\_Worker`=`service\_worker`.`id\_worker` and`record`.`id\_Service`=`service\_worker`.`id\_Service`)

inner join `worker` on (`worker`.`id\_Worker`=`service\_worker`.`id\_worker`)

inner join `sevice` on (`sevice`.`id\_Service`=`service\_worker`.`id\_Service` and `sevice`.`name`='Стрижка')

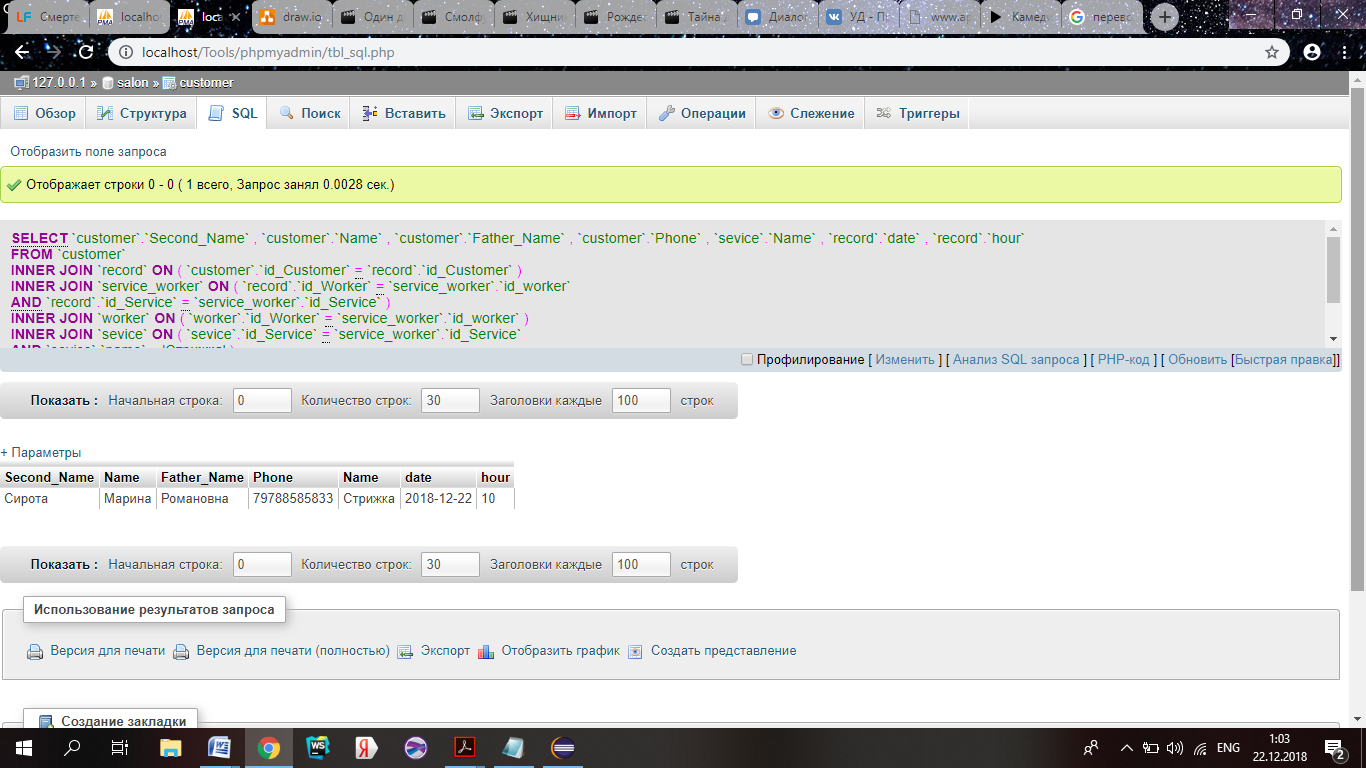


Рисунок Б.8 – Запрос на выбор информации о посещениях услуги Стрижка клиентом Сирота