Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Тихоокеанский государственный университет»

Тимошко А.М., Тусикова А.А.

ПРОЕКТИРОВАНИЕ ПРИЛОЖЕНИЯ БАЗЫ ДАННЫХ С ИСПОЛЬЗОВАНИЕМ JAVA

Методические указания к лабораторной работе №4 по дисциплине «Проектирование приложений баз данных» для студентов специальности 231000.62 «Программная инженерия»

Хабаровск
2016

Проектирование приложения базы данных с использованием Java: методические указания к лабораторной работе №4 по дисциплине «Проектирование приложений баз данных» для студентов специальности 231000.62 «Программная инженерия» / сост. Тимошко А.А., Тусикова А.А. – Хабаровск: 2016. – 24 с.

Методические указания к лабораторной работе №4 по дисциплине «Проектирование приложений баз данных» составлены студентами для помощи другим студентам при выполнении данной работы. В них изложен доступным языком материал для практического применения: написание и работа с моделями таблиц базы данных в Java, фильтры, поиск и прочее.

Хабаровск, 2016

В данной работе мы рассмотрим создание приложения для ведения нашей базы данных, используя Java (IDE NetBeans). Реализуем форму авторизации пользователя, создадим меню для навигации по таблицам, настроим вывод таблиц и отображение функциональных кнопок, согласно правам пользователей, фильтры, контекстный поиск и добавим сообщения об ошибках и каскадном удалении записей базы данных.

Шаг 1. Установка NetBeans

1. Загрузите NetBeans 8.1 с JDK 1.8.
2. Выполните команду в Терминале:
   sh jdk-8u91-nb-8_1-linux-x64.sh

Ресурсы для скачивания NetBeans указаны в разделе «Используемые пакеты» в конце данного пособия.

Шаг 2. Создание модели для работы с базой данных

Для того, чтобы в Java работать с базами данных, необходимо создать собственную модель (класс), которая наследуется от AbstractTableModel. Обязательно реализовываются три метода: int getRowCount(), int getColumnCount() и Object getValueAt(int rowIndex, int columnIndex). Напишем нашу модель, т.е. создадим новый класс RelationTableModel:

```
public class RelationTableModel extends AbstractTableModel
{
    // соединение
    private Connection connection;
    // объект для выполнения запросов
    Statement st;
    // первоначальный запрос к БД
    private String query;
    // данные из БД
    private ResultSet result;
    // заголовки колонок для отображения в JTable
    private String[] headers;
    // имя текущей таблицы
    private String title;

    public RelationTableModel()
    {
        super();
    }
```

```
    public RelationTableModel(Connection con, String sql)
    {
        super();
        try
        {   connection = con;
            query = sql;
            st =
connection.createStatement(ResultSet.TYPE_SCROLL_INSENSITI
VE, ResultSet.CONCUR_UPDATABLE);
            result = st.executeQuery(query);
            int colCount =
result.getMetaData().getColumnCount();
            headers = new String[colCount];

            for(int i = 0; i < colCount; i++)
            {
                headers[i] =
result.getMetaData().getColumnName(i + 1);
            }
        }
        catch (SQLException ex)
        {
            System.out.println(ex.getErrorCode()+":
"+ex.getMessage());
        }
    }
    // получить количество строк
    @Override
    public int getRowCount()
    {
        if(result ==  null) return 0;
        else
        {
            try
            {
                int curRow, lastRow;
                curRow = result.getRow();
                result.last();
                lastRow = result.getRow();
                result.absolute(curRow);
                return lastRow;
            }
            catch (SQLException ex)
            {
```

```
                System.out.println(ex.getErrorCode()+":
"+ex.getMessage());
                return 0;
            }
        }
    }
    // получить количество колонок
    @Override
    public int getColumnCount()
    {
        if(result == null) return 0;
        else
            try
            {
                return
result.getMetaData().getColumnCount();
            }
            catch (SQLException ex)
            {
                System.out.println(ex.getErrorCode()+":
"+ex.getMessage());
                return 0;
            }
    }
    // получить значение ячейки
    @Override
    public Object getValueAt(int rowIndex, int
columnIndex)
    {
        int curRow; Object ob;
        rowIndex++;
        columnIndex++;

        try
        {
            curRow = result.getRow();
            result.absolute(rowIndex);
            ob = result.getObject(columnIndex);
            result.absolute(curRow);
            return ob;
        }
        catch (SQLException ex)
        {
            System.out.println(ex.getErrorCode()+":
"+ex.getMessage());
            return null;
```

```
        }
    }
    // установить названия колонок для JTable
    public void setColumnName(int columnIndex, String
columnName)
    {
        headers[columnIndex] = columnName;
    }
    // получить названия колонок для JTable
    @Override
    public String getColumnName(int columnIndex)
    {
        return headers[columnIndex];
    }
    // получить тип данных для колонок
    @Override
    public Class<?> getColumnClass(int columnIndex)
    {
        try
        {
            return result.getMetaData().getClass();
        }
        catch (SQLException ex)
        {
            System.out.println(ex.getErrorCode()+":
"+ex.getMessage());
            return null;
        }
    }
    // можно ли редактировать?
    @Override
    public boolean isCellEditable(int rowIndex, int
columnIndex)
    {
        return true;
    }
    // установить имя текущей таблицы
    public void setTableName(String name)
    {
        title = name;
    }
    // получить имя текущей таблицы
    public String getTableName()
    {
        return title;
    }
```

```
    // выполнить запрос
    public void setQuery(String sql)
    {
        try
        {
            result = st.executeQuery(sql);
            fireTableStructureChanged();
        }
        catch (SQLException ex)
        {
            if(!"Запрос не вернул
результатов.".equals(ex.getMessage()))
                JOptionPane.showMessageDialog(new
JFrame(), ex.getErrorCode()+": "+ex.getMessage());
        }
    }
    // получить текст первоначального запроса
    public String getQuery()
    {
        return query;
    }
}
```

Шаг 3. Соединение с БД и отображение таблиц

Создадим форму авторизации пользователя. Для обработчика нажатия кнопки «Соединение» напишем код:

```
private void
buttonConnectActionPerformed(java.awt.event.ActionEvent
evt) {
        String host, password;
        int port = (int) jSpinnerPort.getValue();
        host = textHost.getText();
        user = textUser.getText();
        password = jPassword.getText();

        String url;
        url = "jdbc:postgresql://"+host +
":"+port+"/rabota";

        try
        {
            Class.forName("org.postgresql.Driver");
```

```
            connection = DriverManager.getConnection(url,
user, password);
            JOptionPane.showMessageDialog(this,
"Подключение установлено");
            this.setVisible(false);
        }
        catch (ClassNotFoundException ex)
        {
            JOptionPane.showMessageDialog(this, "Драйвер
postgresql не найден");
        }
        catch (SQLException ex)
        {
            JOptionPane.showMessageDialog(this,
ex.getErrorCode()+": "+ex.getMessage());
        }
    }
```

Для обработчика нажатия кнопки «Разъединение» код:

```
private void
buttonDisconnectActionPerformed(java.awt.event.ActionEvent
evt) {
        try
        {
            connection.close();
        }
        catch (SQLException ex)
        {
            System.out.println(ex.getErrorCode()+":
"+ex.getMessage());
        }
    }
```

Чтобы вызвать данную форму при нажатии на пункт меню «Авторизация», нужно написать обработчик в классе главной формы:

```
private void
jItemAutorizationActionPerformed(java.awt.event.ActionEven
t evt) {
        autorizationFrame.setVisible(true);
    }
```

Для отображения таблицы «Работа» в JTable напишем код для обработчика нажатия пункта меню «Основные таблицы – Работа»:

```
private void
jItemRabotaActionPerformed(java.awt.event.ActionEvent evt)
{
    query = "SELECT id_rabota, n_rabota, n_pokazatel,
n_kafedra, fio, year, volume FROM rabota, kafedra,
pokazatel, prepod WHERE (rabota.id_pokazatel =
pokazatel.id_pokazatel) AND (rabota.id_kafedra =
kafedra.id_kafedra) AND (rabota.id_prepod =
prepod.id_prepod)";
    con = autorizationFrame.connection;
    model = new RelationTableModel(con, query);
    model.setTableName("rabota");
    model.setColumnName(1,"Наименование работы");
    model.setColumnName(2,"Наименование показателя");
    model.setColumnName(3,"Наименование кафедры");
    model.setColumnName(4,"Ф.И.О. преподавателя");
    model.setColumnName(5,"Учебный год");
    model.setColumnName(6,"Объем работы");
    jTableView.setModel(model);
    SetHiddenId();
}
```

Код для скрытия ID:

```
private void SetHiddenId()
{
jTableView.getColumnModel().getColumn(0).setMaxWidth(0);
jTableView.getColumnModel().getColumn(0).setMinWidth(0);
jTableView.getColumnModel().getColumn(0).setPreferredWidth
(0);
}
```

Шаг 4. Активность функциональных кнопок

Чтобы активировать или сделать неактивными кнопки «Добавить» и «Удалить» в зависимости от прав пользователя, создадим метод:

```
private void setButtonsEnable(String nameTable, String
nameUser)
    {
        try
        {
            String query_role;
            // до проверки прав кнопки неактивны
            jButtonAdd.setEnabled(false);
            jButtonDelete.setEnabled(false);
```

```
            // запрос проверки прав на добавление
            query_role = "SELECT has_table_privilege('" +
nameUser + "','" + nameTable +"','INSERT')";
            Statement st =
con.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
ResultSet.CONCUR_UPDATABLE);
            ResultSet res = st.executeQuery(query_role);
            res.absolute(1);
            Boolean str = (Boolean) res.getObject(1);
            res.absolute(1);
            // если есть права, сделать активной
            if(str == true) jButtonAdd.setEnabled(true);
            // запрос проверки прав на удаление
            query_role = "SELECT has_table_privilege('" +
nameUser + "','" + nameTable +"','DELETE')";
            res = st.executeQuery(query_role);
            res.absolute(1);
            str = (Boolean) res.getObject(1);
            res.absolute(1);
            // если есть права, сделать активной
            if(str == true)
jButtonDelete.setEnabled(true);
        }
        catch (SQLException ex)
        {

Logger.getLogger(RabotaJFrame.class.getName()).log(Level.S
EVERE, null, ex);
        }
    }
```

Добавим вызов этого метода в методе jItemRabotaActionPerformed:

```
    setButtonsEnable(model.getTableName(),
autorizationFrame.user);
```

Шаг 5. Реализация фильтров

Создадим метод установки модели для JComboBox, чтобы в нем отражались данные:

```
private void SetComboBoxModel(JComboBox combo,
DefaultComboBoxModel modelCombo, String sql)
    {
        try
```

```
        {
            Statement st =
con.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
ResultSet.CONCUR_UPDATABLE);
            ResultSet rs = st.executeQuery(sql);
            // очистить содержимое модели
            modelCombo.removeAllElements();
            // добавить первый элемент «пустая строка»
            modelCombo.addElement("");
            // добавить все элементы из запроса
            while(rs.next())
            {
                modelCombo.addElement(rs.getString(1));
            }
            combo.setModel(modelCombo);
        }
        catch (SQLException ex)
        {

Logger.getLogger(RabotaJFrame.class.getName()).log(Level.S
EVERE, null, ex);
        }
    }
```

При выборе пользователем элемента JComboBox должна применяться фильтрация записей таблицы и скрываться соответствующая колонка (на примере фильтра по кафедре):

```
private void
jBoxKafedraItemStateChanged(java.awt.event.ItemEvent evt)
{
    SetRabota();
    SetHiddenId();
    SetFilterRabota();
}
```

Метод SetRabota (применить все установленные фильтры):

```
private void SetRabota()
{
        String sql = query;
        if(jBoxKafedra.getSelectedItem() != "")
            sql += " AND (kafedra.n_kafedra = '" +
jBoxKafedra.getSelectedItem() + "')";
        if(jBoxPokazatel.getSelectedItem() != "")
```

```
            sql += " AND (pokazatel.n_pokazatel = '" +
jBoxPokazatel.getSelectedItem() + "')";
        if(jBoxPrepod.getSelectedItem() != "")
            sql += " AND (prepod.fio = '" +
jBoxPrepod.getSelectedItem() + "')";
        if(jBoxYear.getSelectedItem() != "")
            sql += " AND (rabota.year = " +
jBoxYear.getSelectedItem() + ")";
        model.setQuery(sql);
}
```

Метод SetFilterRabota (скрыть нужные колонки):

```
private void SetFilterRabota()
{
if(jBoxKafedra.getSelectedItem() != "")
{
jTableView.getColumnModel().getColumn(3).setMaxWidth(0);
jTableView.getColumnModel().getColumn(3).setMinWidth(0);
jTableView.getColumnModel().getColumn(3).setPreferredWidth
(0);
}
else
{
jTableView.getColumnModel().getColumn(3).setMaxWidth(21474
83647);
jTableView.getColumnModel().getColumn(3).setMinWidth(60);
jTableView.getColumnModel().getColumn(3).setPreferredWidth
(60);
}
// далее аналогично для остальных фильтров данной таблицы
}
```

Шаг 6. Реализация контекстного поиска

Для контекстного поиска реализуем обработчик нажатия клавиш на объект JTextField:

```
private void
jTextSearchKeyReleased(java.awt.event.KeyEvent evt)
{       // получить исходный запрос
        String sql = model.getQuery();
        String str = jTextSearch.getText();
        String str_query = sql;
```

```
        if(null != model.getTableName() &&
!"".equals(str))
            switch (model.getTableName()) {
            case "kafedra":
               str_query = sql + " WHERE n_kafedra LIKE
'%" + str + "%'";
                break;
            case "prepod":
                str_query = sql + " WHERE fio LIKE '%" +
str + "%'";
                break;
            case "ed_izm":
                str_query = sql + " WHERE n_izm || hours
LIKE '%" + str + "%'";
                break;
            case "sfera":
                str_query = sql + " AND (n_izm || n_sfera
LIKE '%" + str + "%')";
                break;
            case "pokazatel":
                str_query = sql + " AND (n_pokazatel ||
n_sfera LIKE '%" + str + "%')";
                break;
            case "rabota":
                str_query = sql + " AND (n_pokazatel ||
n_kafedra || fio || year || volume || n_rabota LIKE '%" +
str + "%')";
                break;
            default:
                break;
        }

        model.setQuery(str_query);
        SetHiddenId();
    }
```

Шаг 7. Добавление и изменение записи

При нажатии кнопки «Добавить» или двойном клике мыши по записи вызывается форма для заполнения данных, причем для каждой таблицы создана «своя» форма. Чтобы оповестить главную форму, что пользователь уже закончил заполнение или редактирование, нужно

добавить слушатель для кнопки «ОК». Но так как все компоненты формы, включая нашу кнопку, объявлены как private, то в классе главной формы слушатель для нее создать не удастся. Поэтому создадим программно вспомогательную кнопку и объявим ее в качестве public, а в конце обработчика нажатия на нашу основную кнопку «ОК» вызовем для вспомогательной кнопки метод doClick(), который «прослушаем» в классе основной формы. Описанная идея работает правильно, но с точки зрения методологии программирования не очень хороша.

При нажатии на кнопку «Добавить» вызывается ее обработчик:

```
private void
jButtonAddActionPerformed(java.awt.event.ActionEvent evt)
{
        if("rabota".equals(model.getTableName()))
        {
            int _year;
            if("".equals((String)
jBoxYear.getSelectedItem())) _year = 0;
            else _year = Integer.parseInt((String)
jBoxYear.getSelectedItem());
            // настроить отображение JComboBox для формы
            addRabotaFrame.setOptions(con);
            // установить текущие значения полей формы
            // INSERT - флаг заполнения данных (добавление
новой записи в таблицу)
            addRabotaFrame.setFields(" ",
(String)jBoxPokazatel.getSelectedItem(),
(String)jBoxKafedra.getSelectedItem(),
(String)jBoxPrepod.getSelectedItem(), _year, 0, -1,
"INSERT");
            addRabotaFrame.setVisible(true);
        }
        // далее аналогично для других таблиц
}
```

Слушатель нажатия мыши на JTable (в конструкторе):

```
jTableView.addMouseListener(new MouseAdapter()
        {
            @Override
            public void mouseClicked(MouseEvent e)
            {   // двойной клик мыши
                if(e.getClickCount() == 2)
```

```
                {
                    if
if("rabota".equals(model.getTableName()))
                    {   // получить значения каждой ячейки
                        int row =
jTableView.getSelectedRow();
                        int id_rabota = (int)
model.getValueAt(row, 0);
                        String n_rabota = (String)
model.getValueAt(row, 1);
                        String n_pokazatel = (String)
model.getValueAt(row, 2);
                        String n_kafedra = (String)
model.getValueAt(row, 3);
                        String fio = (String)
model.getValueAt(row, 4);
                        int year = (int)
model.getValueAt(row, 5);
                        float volume = (float)
model.getValueAt(row, 6);
                        // настроить JComboBox
                        addRabotaFrame.setOptions(con);
                        // установить значения полей
                        // UPDATE - флаг заполнения данных
(изменение выбранной записи в таблице)
                        addRabotaFrame.setFields(n_rabota,
n_pokazatel, n_kafedra, fio, year, volume, id_rabota,
"UPDATE");
                        addRabotaFrame.setVisible(true);
                    }
// далее аналогично для других таблиц
                }
            }
        });
```

Далее рассмотрим методы класса формы для добавления и изменения данных в таблице «Работа». Настраиваем JComboBox с помощью метода setOptions:

```
public void setOptions(Connection con)
    {
        modelPokazatel = new DefaultComboBoxModel();
        modelKafedra = new DefaultComboBoxModel();
        modelPrepod = new DefaultComboBoxModel();
```

```
        connect = con;
        try
        {
            Statement st =
connect.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
ResultSet.CONCUR_UPDATABLE);
            ResultSet rs = st.executeQuery("SELECT
n_kafedra FROM kafedra");
            modelKafedra.removeAllElements();
            modelKafedra.addElement("");
            while(rs.next())
            {
                modelKafedra.addElement(rs.getString(1));
            }
            jBoxKafedra.setModel(modelKafedra);
        }
        catch (SQLException ex)
        {

Logger.getLogger(RabotaJFrame.class.getName()).log(Level.S
EVERE, null, ex);
        }
         // далее аналогично для других фильтров
    }
```

Для того, чтобы из таблицы и JComboBox в главной форме приложения передать значения в форму для добавления и изменения записей, существует метод setFields:

```
public void setFields(String rabota, String pokazatel,
String kafedra, String prepod, int year, float volume, int
id_rabota, String flag)
    {
        jTextRabota.setText(rabota);
        jBoxPokazatel.setSelectedItem(pokazatel);
        jBoxKafedra.setSelectedItem(kafedra);
        jBoxPrepod.setSelectedItem(prepod);
        jTextYear.setText(Integer.toString(year));
        jTextVolume.setText(Float.toString(volume));
        this.flag = flag;
        this.id_rabota = id_rabota;
    }
```

Вся основная работа происходит в обработчике нажатия кнопки «ОК»:

```
private void
jButtonOKActionPerformed(java.awt.event.ActionEvent evt)
{
        rabota = jTextRabota.getText();
        pokazatel = (String)
jBoxPokazatel.getSelectedItem();
        kafedra = (String) jBoxKafedra.getSelectedItem();
        prepod = (String) jBoxPrepod.getSelectedItem();
        year = jTextYear.getText();
        volume = jTextVolume.getText();
        int id_pokazatel = -1;
        int id_kafedra = -1;
        int id_prepod = -1;
        // проверка на корректность заполнения данных
        if("".equals(pokazatel) || "".equals(kafedra) ||
"".equals(prepod) || "".equals(year) || "".equals(volume))
        {
            JOptionPane.showMessageDialog(new JFrame(),
"Введено пустое значение!");
        }
        else if(Float.parseFloat(volume) <= 0 ||
Float.parseFloat(volume) > 1000)
        {
            JOptionPane.showMessageDialog(new JFrame(),
"Объем работы должен быть больше 0 и не более 1000!");
        }
        else if(Integer.parseInt(year) < 1980 ||
Integer.parseInt(year) > 2016)
        {
            JOptionPane.showMessageDialog(new JFrame(),
"Учебный год должен быть больше 1979 и не более 2016!");
        }
        else
        {   // получить id выбранного по имени показателя
            try
            {
                Statement st;
                st =
connect.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
ResultSet.CONCUR_UPDATABLE);
```

```
                ResultSet res = st.executeQuery("SELECT
id_pokazatel FROM pokazatel WHERE n_pokazatel = '" +
pokazatel + "'");
                if(res != null)
                {
                    res.absolute(1);
                    id_pokazatel = (int) res.getObject(1);
                    res.absolute(1);
                }
            }
            catch (SQLException ex)
            {

            }
            // аналогично для других JComboBox
            // запрос для добавления новой записи
            if("INSERT".equals(flag))
                sql = "INSERT INTO rabota(n_rabota,
id_pokazatel, id_kafedra, id_prepod, year, volume) VALUES
('" + rabota + "','" + id_pokazatel + "','" + id_kafedra +
"','" + id_prepod + "','" + year + "','" + volume + "')";
            // запрос для изменения выбранной записи
            else if("UPDATE".equals(flag))
            {   sql = "UPDATE rabota SET n_rabota = '" +
rabota + "' WHERE id_rabota = " + id_rabota + "; ";
                sql += "UPDATE rabota SET id_pokazatel = "
+ id_pokazatel + " WHERE id_rabota = " + id_rabota + "; ";
                sql += "UPDATE rabota SET id_kafedra = " +
id_kafedra + " WHERE id_rabota = " + id_rabota + "; ";
                sql += "UPDATE rabota SET id_prepod = " +
id_prepod + " WHERE id_rabota = " + id_rabota + "; ";
                sql += "UPDATE rabota SET year = " + year
+ " WHERE id_rabota = " + id_rabota + "; ";
                sql += "UPDATE rabota SET volume = " +
volume + " WHERE id_rabota = " + id_rabota + "; ";
            }
            // выполнить запрос
            try
            {
                Statement st =
connect.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
ResultSet.CONCUR_UPDATABLE);
```

```
                st.executeQuery(sql);
            }
            catch (SQLException ex)
            {
                if(!"Запрос не вернул
результатов.".equals(ex.getMessage()))
                    JOptionPane.showMessageDialog(new
JFrame(), ex.getErrorCode()+": "+ex.getMessage());
            }
            // вызвать метод нашей вспомогательной кнопки,
чтобы прослушать ее в классе главной формы, т.е. послать
сигнал, что пользователь ввел все данные, запрос выполнен
            OK.doClick();
            // скрыть форму
            this.setVisible(false);
        }
    }
```

Вид вышеописанной формы приведен на рис. 1. Вернемся к коду класса главной формы и напишем в конструкторе слушатель для вспомогательной кнопки:

```
addRabotaFrame.OK.addActionListener(new ActionListener()
{
    @Override
    public void actionPerformed(ActionEvent e)
    {   // выполнить исходный запрос
        model.setQuery(query);
        SetRabota();
        SetHiddenId();
        SetFilterRabota();
        // обновить модель для фильтра по году, так как
при добавлении или изменении записей может появиться новый
год или исчезнуть старый
        String year = "SELECT DISTINCT year FROM rabota
ORDER BY year";
        SetComboBoxModel(jBoxYear, modelBoxYear, year);
    }
});
```

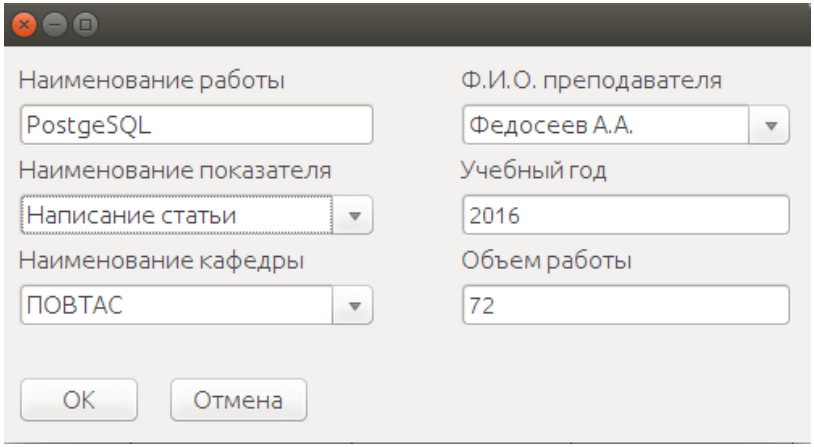

Рис. 1. Форма для добавления или изменения записи в таблице «Работа»

Шаг 8. Удаление записи

Реализуем удаление выбранной записи из таблицы при нажатии на кнопку «Удалить». Не забываем, что нужно уведомить пользователя при наличии ссылочных данных и предоставить ему выбор: удалять или нет.

```
private void
jButtonDeleteActionPerformed(java.awt.event.ActionEvent
evt)
{       // получить номер выбранной строки
        int row = jTableView.getSelectedRow();
        // получить ID строки
        int value = (int) model.getValueAt(row, 0);
       // определить название колонки с ID записи таблицы
        String fild = "";
        String tableName = model.getTableName();
        if(null != tableName)
            switch (tableName) {
            case "kafedra":
                fild = "id_kafedra";
                break;
            case "prepod":
                fild = "id_prepod";
                break;
            case "ed_izm":
                fild = "id_izm";
                break;
```

```
            case "sfera":
                fild = "id_sfera";
                break;
            case "pokazatel":
                fild = "id_pokazatel";
                break;
            case "rabota":
                fild = "id_rabota";
                break;
            default:
                break;
        }
        // текст запроса для проверки ссылок
        String query_check = "";
        if(null != tableName)
            switch (tableName) {
            case "kafedra":
            case "prepod":
            case "pokazatel":
                query_check = "SELECT id_rabota FROM
rabota WHERE " + fild + " = " + value;
                break;
            case "sfera":
                query_check = "SELECT id_pokazatel FROM
pokazatel WHERE " + fild + " = " + value;
                break;
            case "ed_izm":
                query_check = "SELECT id_sfera FROM sfera
WHERE " + fild + " = " + value;
                break;
            default:
                break;
        }
        // выполнить запрос и получить результат
        int id = -1;
        try
        {
            Statement st;
            st =
con.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
ResultSet.CONCUR_UPDATABLE);
            ResultSet res = st.executeQuery(query_check);
```

```
            // если найдена хотя бы одна ссылка, записать
первую в id
            if(res != null)
            {
                res.absolute(1);
                id = (int) res.getObject(1);
                res.absolute(1);
            }
        }
        catch (SQLException ex)
        {

        }
        // если есть ссылка, сообщить пользователю
        if(id != -1)
            answer = JOptionPane.showConfirmDialog(this,
"Внимание! Найдены зависимые записи в дочерних таблицах.
Выполнить каскадное удаление?", "Каскадное удаление",
JOptionPane.YES_NO_OPTION);
        // при согласии пользователя или отсутствии ссылок
выполнить запрос на удаление
        if(answer == YES_OPTION || id == -1)
        {
            String str_query = "DELETE FROM " + tableName
+ " WHERE " + fild + " = " + value;
            model.setQuery(str_query);
            // вернуться к первоначальному запросу
            model.setQuery(query);
        }
        SetHiddenId();
        SetFilter();
    }
```

Реализация метода setFilter:

```
    private void SetFilter()
    {
        String tableName = model.getTableName();
        if(null != tableName)
            switch (tableName) {
            case "pokazatel":
            {
                SetPokazatel();
                SetHiddenId();
```

```
                SetFilterSfera();
                break;
            }
            case "sfera":
            {
                SetSfera();
                SetHiddenId();
                SetFilterIzm();
                break;
            }
            case "rabota":
            {
                SetRabota();
                SetHiddenId();
                SetFilterRabota();
                break;
            }
            default:
                break;
        }
    }
```

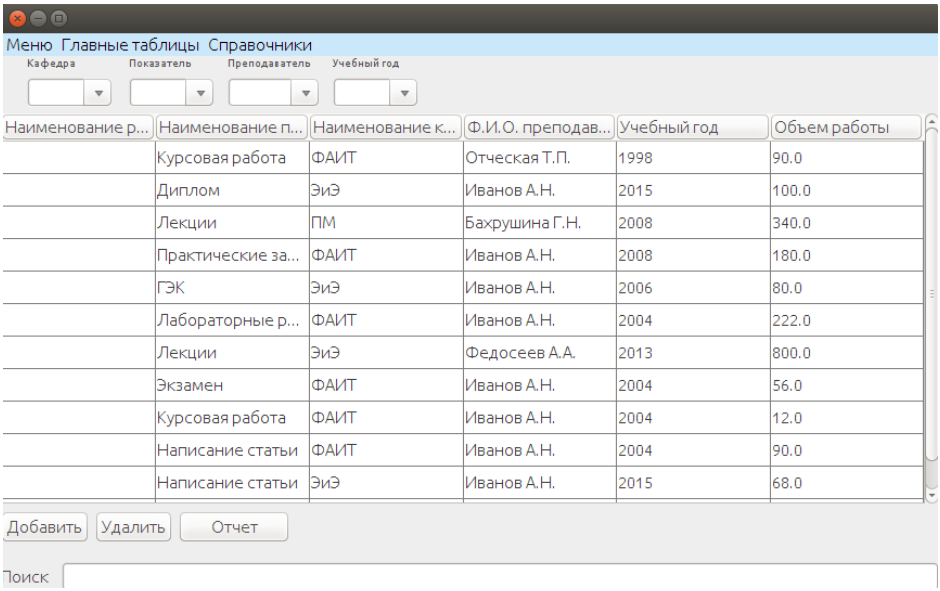

Рис. 2. Вид главной формы приложения с открытой таблицей «Работа»

Используемые источники

1. https://javaswing.wordpress.com/2010/05/05/jtabletablemodel/
2. https://docs.oracle.com/javase/tutorial/jdbc/basics/jdbcswing.html
3. https://docs.oracle.com/javase/tutorial/uiswing/components/table.html
4. http://stackoverflow.com/questions/27250401/creating-a-jtable-from-abstracttablemodel-in-netbeans

Используемые пакеты

1. NetBeans 8.1 на официальном сайте: https://netbeans.org/downloads/
2. NetBeans 8.1 + JDK 1.8 на сайте Oracle: http://www.oracle.com/technetwork/articles/javase/jdk-netbeans-jsp-142931.html