

Министерство образования Республики Беларусь  
Учреждение образования  
Белорусский государственный университет  
информатики и радиоэлектроники  
Кафедра инженерной психологии и эргономики

ОТЧЕТ ПО ПРАКТИКЕ

Студент гр. 910101 Микулич Е.И.

Руководитель практики от  
предприятия

(подпись)

Бунин И.А.

Утверждаю  
(руководитель предприятия,  
зам. руководитель предприятия)

(подпись)

Руководитель практики от  
Кафедры ИПиЭ

(подпись)

Щербина Н.В.

Минск 2022

## Содержание

О предприятии ГУ «БелИСА».....	3
Индивидуальное задание .....	5
Выполнение задания.....	6
Заключение.....	13
Список использованных источников.....	14
Приложение А.....	15

## **О предприятии ГУ «БелИСА»**

Государственное учреждение «Белорусский институт системного анализа и информационного обеспечения научно-технической сферы» (ГУ «БелИСА») Государственного комитета по науке и технологиям Республики Беларусь создано в соответствии с Указом Президента Республики Беларусь № 363 от 13.09.1996 г. в целях выполнения функций системного анализа состояния и тенденций развития научно-технической сферы, проведения научных исследований и разработок для научно-информационного обеспечения деятельности ГКНТ Республики Беларусь, иных органов государственной власти и управления, научных организаций Республики Беларусь, разработки методических основ информационного обеспечения научно-технической сферы.

ГУ «БелИСА» является преемником бывшего БелНИИНТИ Госплана БССР и ГКНТ СССР.

ГУ «БелИСА» является национальным информационным центром, координирующим межгосударственный обмен научно-технической информацией в республике. ГУ «БелИСА» осуществляет государственную регистрацию научно-исследовательских, опытно-конструкторских и опытно-технологических работ (НИР, ОКР и ОТР) в Беларуси, ведет Государственный реестр НИР, ОКР и ОТР, базы данных и фонд отчетной научно-технической документации по зарегистрированным работам (технические задания, итоговые и промежуточные отчеты, пояснительные записки проектов и т.п.).

На ГУ «БелИСА» возложены функции по организации экспертизы материалов заявок для включения в Реестр высокотехнологичных производств и предприятий и ведению этого реестра.

ГУ «БелИСА» осуществляет депонирование рукописей научных работ, издает республиканский сборник непубликуемых работ, бюллетень регистрации НИР, ОКР и ОТР, журнал «Новости науки и технологий» и другие издания, а также обладает уникальными информационными ресурсами в сфере осуществления научно-технической деятельности в Республике Беларусь.

Основные задачи и функции ГУ «БелИСА»:

- Системный анализ состояния и тенденций преобразования научно-технической сферы и оценка перспектив развития конкретных направлений науки и технологий.
- Мониторинг развития научно-технического потенциала республики и выполнения НИР, ОКР в Беларуси.
- Создание и использование банков данных о научном потенциале, инновационной деятельности, научно-технических разработках, новых технологиях.
- Сбор, обобщение, анализ научно-технической информации и обеспечение ею органов государственной власти и управления, научных организаций, иных субъектов хозяйствования, ученых и специалистов для развития научных исследований и разработок, освоения их результатов, обеспечения международных научно-технических связей, создания единого информационного пространства.
- Научно-информационное обеспечение международного научно-технического сотрудничества, расширение сотрудничества с зарубежными научно-информационными центрами и организациями.
- Маркетинг, продвижение на рынок научно-технической продукции.
- Содействие разработке и реализации государственной научно-технической политики в сфере научно-информационной деятельности.
- Разработка научно-методического обеспечения деятельности органов научно-технической информации в Беларуси.
- Освоение и развитие новых информационных и телекоммуникационных технологий доступа к компьютерным сетям и информационным системам.
- Использование научно-информационных компьютерных сетей и банков данных для решения научно-технических проблем в республике.
- Разработка методических основ научно-информационного обеспечения научной, научно-технической и инновационной деятельности.
- Координация выполнения научно-технических программ по основным направлениям своей деятельности.

## Индивидуальное задание

**Задание:** создание функционирующего приложения учёта кадров в архитектуре клиент-сервер с организацией взаимодействия с базой данных на объектно-ориентированном языке Java. Серверное приложение может быть реализовано в виде консольного приложения или GUI-приложения. Клиентское приложение: оконное приложение с использованием стандартных библиотек пользовательского интерфейса JavaFX.

Для успешной реализации системы необходимо, в первую очередь, выделить основные задачи, а также те задачи, которые необходимо выполнить для правильной работы системы.

Основными задачами являются следующие:

- 1 Разработка базы данных для MySQL 5.5+.
- 2 Разработка и использование собственной иерархии классов, расширение базовых классов, предоставляемых JDK.
- 3 Использовать сокрытие данных (инкапсуляция), обработку исключительных ситуаций.
- 4 В разрабатываемом приложении обеспечить добавление, редактирование и удаление записей из базы данных, предоставление пользователю аналитической информации (графики, диаграммы).
- 5 Предусмотреть механизм авторизации пользователей.
- 6 Система должна предоставлять удобный интерфейс для конечного пользователя.

Рассмотрим некоторые функции, реализующие бизнес-логику:

- Авторизация.

Пользователь вводит свой логин и пароль. Если логин в БД не найдена или введён не верный пароль, то пользователю будет выведено сообщение об ошибке. Иначе пользователь выполнит вход и в зависимости от роли будет открыто соответствующее окно.

- Просмотр информации.

Клиент посылает сообщения серверу о получении данных о сотрудниках, личных данных или документах, после чего на сервере формируется SQL-запрос и выводится соответствующая информация.

- Просмотр операций.

Формируется SQL-запрос, и выводится информация, о об операциях.

## Выполнение задания

Для того, чтобы задание было выполнено, необходимо в первую очередь создать базу данных, с которой будет взаимодействовать приложение.

Структура базы данных изображена на Рисунке 1.

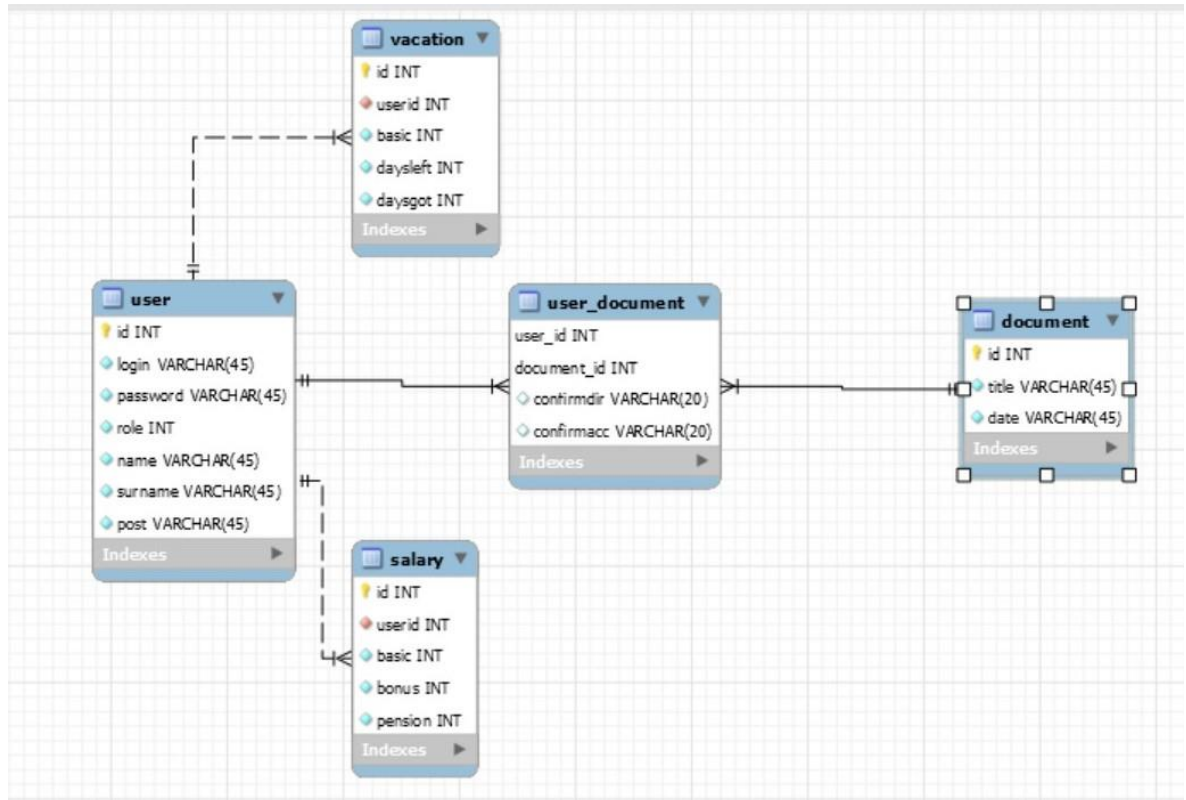


Рисунок 1 – Структура базы данных приложения

После создания базы данных необходимо написать серверную часть приложения, которая будет обрабатывать и выполнять запросы клиентской части приложения. Фрагмент кода серверной части приложения находится в Приложении А.

После написания серверной части приложения, необходимо написать её клиентскую часть. Интерфейс клиентской части изображён на Рисунках 2 - ...

При запуске приложения открывается окно авторизации:

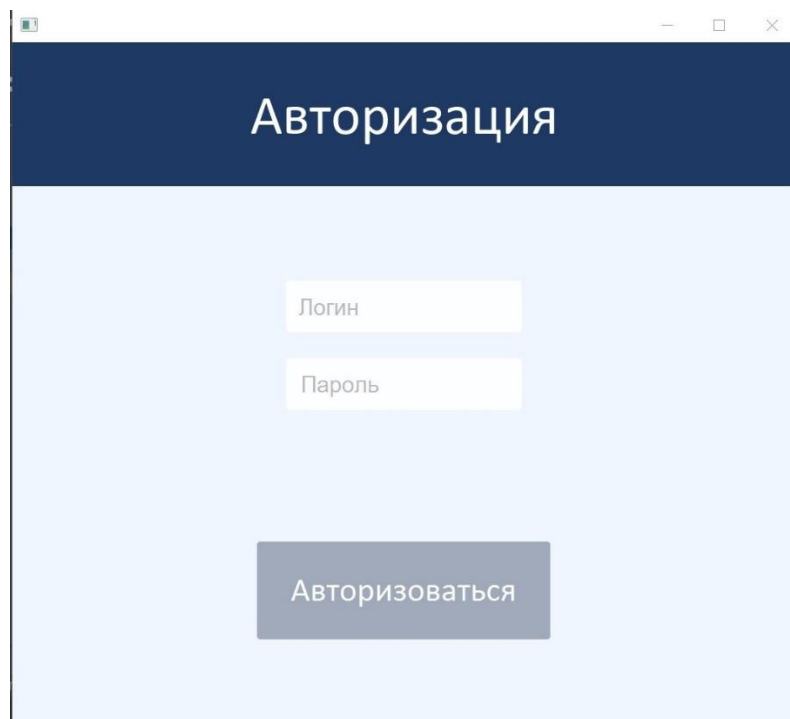


Рисунок 2 – Окно авторизации

В зависимости от роли (Директор, Бухгалтер, Сотрудник) перед пользователем открывается соответствующее меню:

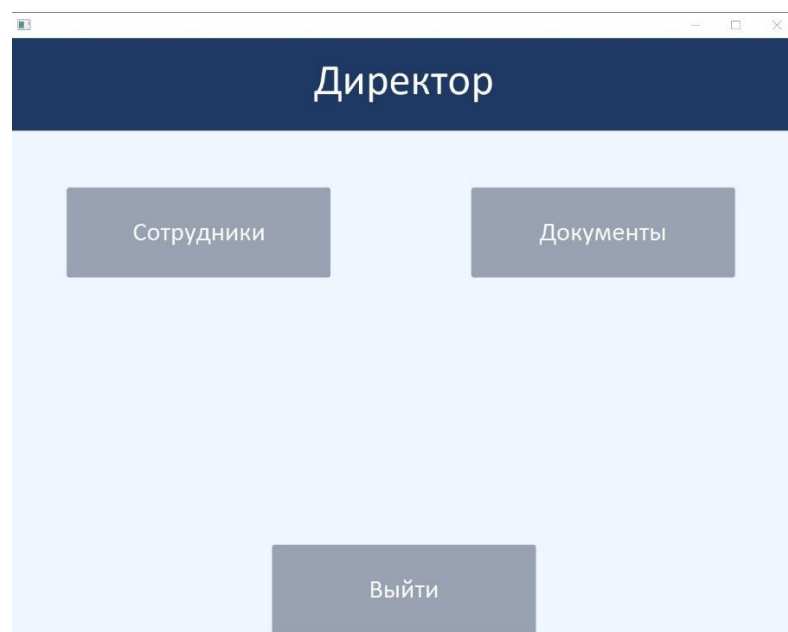


Рисунок 3 – Меню Директора

После авторизации в системе с ролью Директор, пользователь может просмотреть сотрудников, подписать действующие документы, либо вернуться к окну авторизации.



Рисунок 4 – Меню Бухгалтера

После авторизации в системе с ролью Бухгалтер, пользователь может просмотреть сотрудников, подписать действующие документы, добавить нового сотрудника, либо вернуться к окну авторизации.

Назад

Сотрудник

Егор Микулич Программист

Логин: egor Пароль: qwerty

Оклад: 550 Премия: 150 Пенсия: 100

Тема

Отправить

Тема	Дата	Подпись дир.	Подпись бух.
Отпуск 15 дней	2022.07.05	В обработке	В обработке
Приказ 15	2022.07.05	В обработке	В обработке

Отпуск

Изначально: 30

Осталось: 30

Потратил: 0

Запросить отпуск



## Рисунок 5 – Меню Сотрудника

После авторизации в системе с ролью Сотрудник, пользователь может просмотреть личную информацию, запросить отпуск, либо создать документ для подписи.

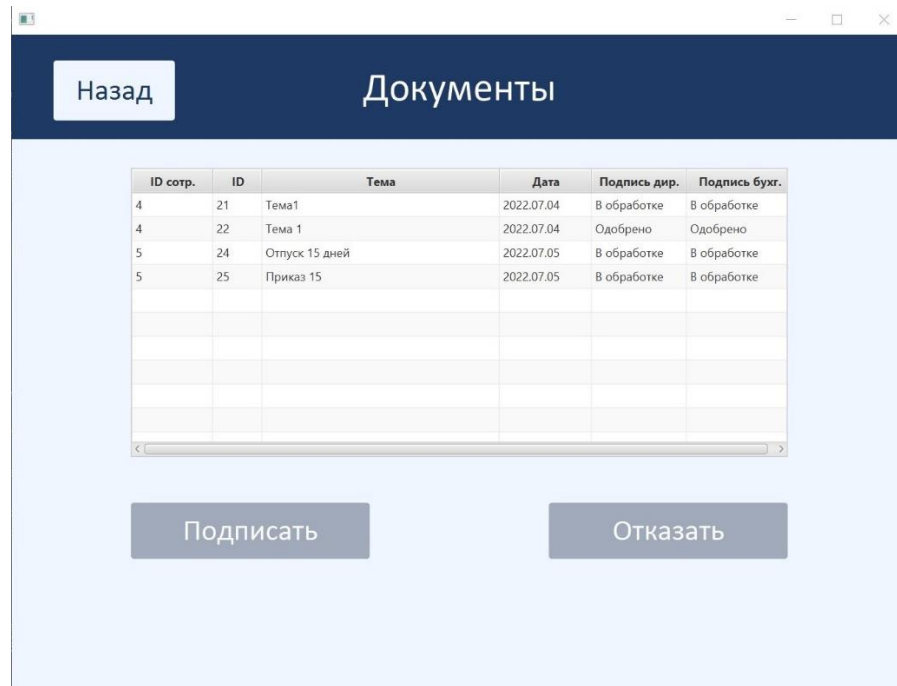


Рисунок 6 – Событие нажатия кнопки «Документы» Директора

Окно «Документы» отображает все действующие документы, которые сотрудники подали на рассмотрение. После того, как в полях «Подпись директора» и «Подпись бухгалтера» будет установлен статус «Одобрено», документ будет являться действительным, в противном случае – недействительным.

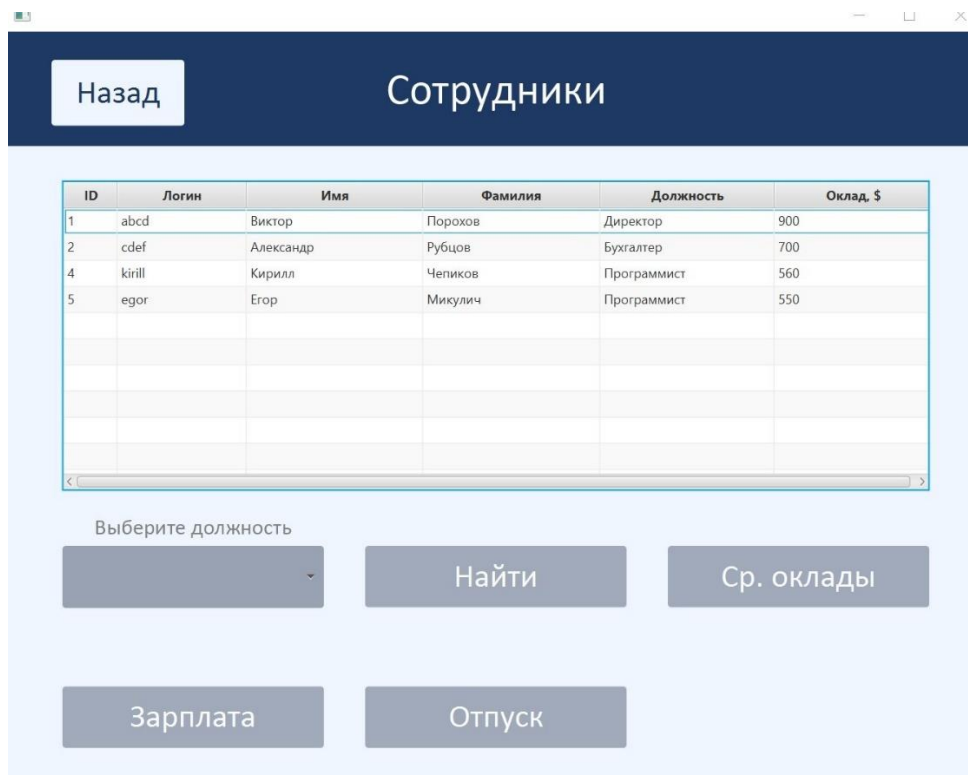


Рисунок 7 – Событие нажатия кнопки «Сотрудники» Директора

В окне «Сотрудники» пользователь с ролью Директор может посмотреть всех сотрудников компании, найти всех сотрудников по определённой должности, с помощью кнопок «Зарплата» и «Отпуск» можно посмотреть соответствующую информацию по выбранному сотруднику, а также графики средних зарплат по должностям (Рисунок 8-9).

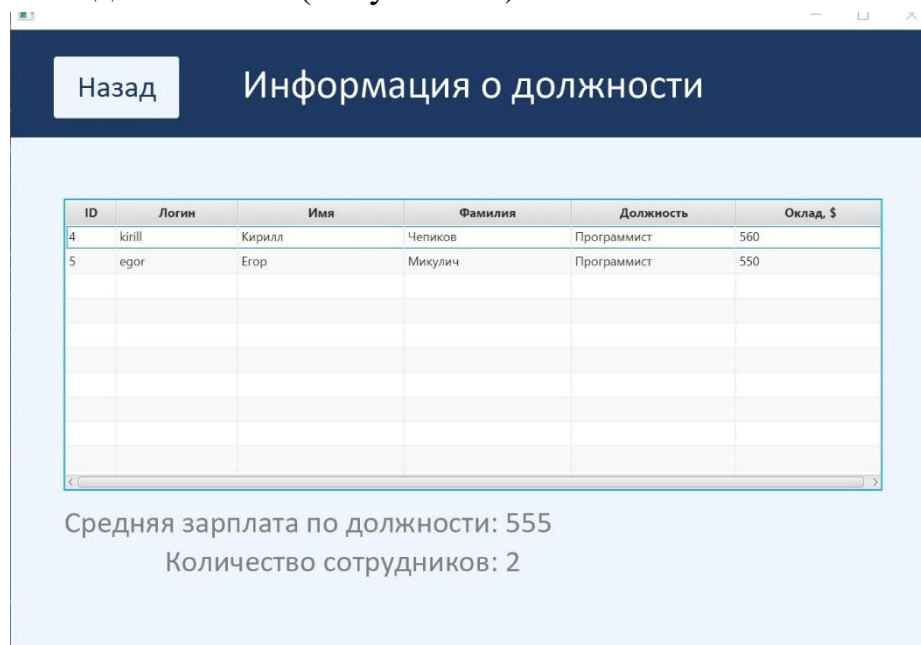


Рисунок 8 – Событие нажатия кнопки «Найти» Директора

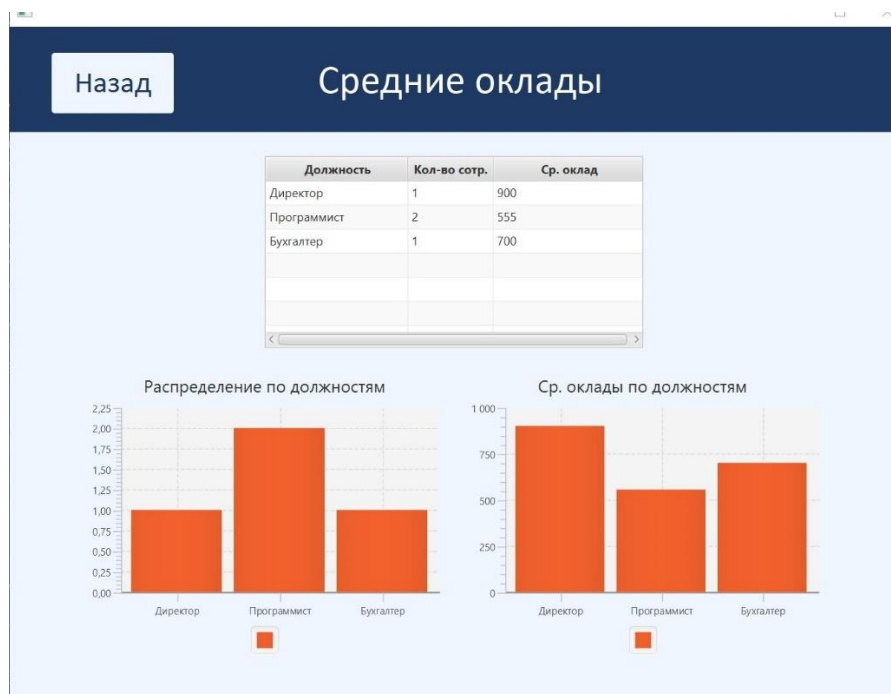


Рисунок 9 – Событие нажатия кнопки «Ср. оклады» Директора

ID	Имя	Фамилия	Должность	Оклад, \$	Премия, \$	Пенсия, \$
1	Виктор	Порохов	Директор	900	500	100
2	Александр	Рубцов	Бухгалтер	700	300	100
4	Кирилл	Чепиков	Программист	560	200	90
5	Егор	Микучич	Программист	550	150	100

Удалить

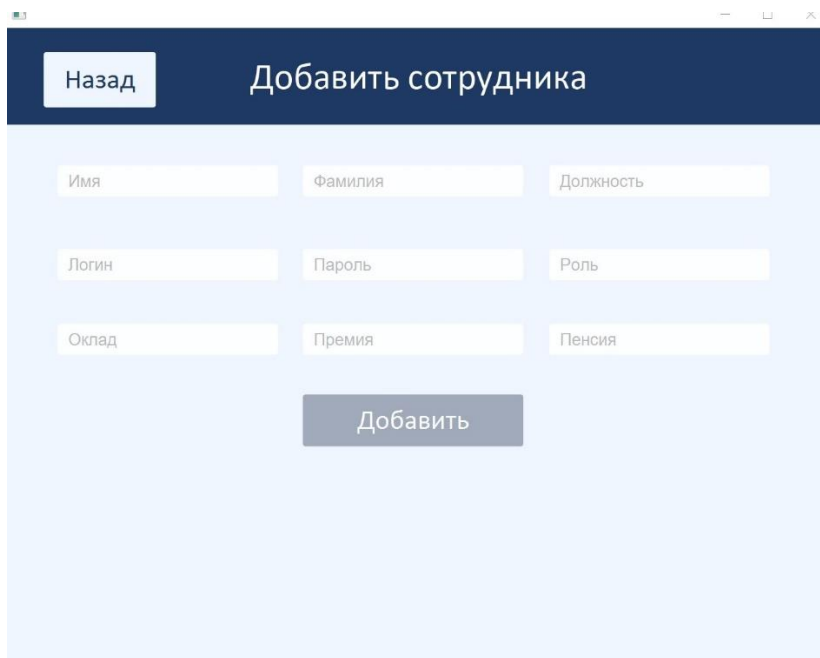
Должность

Оклад Премия Пенсия

Изменить

Рисунок 10 – Событие нажатия кнопки «Сотрудники» Бухгалтера

В окне «Сотрудники» пользователь с ролью Бухгалтер может посмотреть сотрудников, удалить сотрудника, также изменить информацию (Рисунок 11).



The screenshot shows a web application window titled 'Добавить сотрудника' (Add Employee). At the top left is a 'Назад' (Back) button. The form contains nine input fields arranged in a 3x3 grid: 'Имя' (Name), 'Фамилия' (Surname), 'Должность' (Position), 'Логин' (Login), 'Пароль' (Password), 'Роль' (Role), 'Оклад' (Salary), 'Премия' (Bonus), and 'Пенсия' (Pension). Below the grid is a 'Добавить' (Add) button.

Рисунок 11 – Событие нажатия кнопки «Добавить сотрудника»  
Бухгалтера

Событие нажатия кнопки «Документы» Бухгалтера аналогично событию нажатия кнопки «Документы» Директора.

## **Заключение**

В процессе прохождения производственной практики были выполнены все поставленные задачи, а именно:

- Ознакомление с организационной структурой и производственными процессами предприятия;
- Закрепление теоретических знаний в производственно-технологических условиях;
- Закрепление умений и навыков по проектированию (решению) производственно-технологических задач и их программной реализаций.
- Изучение информационных технологий предприятия, используемых и разрабатываемых на предприятии;
- Приобретение навыков эксплуатации ИТ-систем и их компонентов;
- Выполнение индивидуального задания.

С помощью средств разработки было создано функционирующее приложения учёта кадров в архитектуре клиент-сервер с организацией взаимодействия с базой данных на объектно-ориентированном языке Java. Все цели индивидуального задания были выполнены.

## Список использованных источников

- [1] [Электронный ресурс]. – Электронные данные. - Режим доступа: <https://en.wikipedia.org/wiki/Java>
- [2] Комличенко, В.Н. Компьютерные сети: лаб. практикум, Минск: БГУИР, 2012.
- [3] [Электронный ресурс]. – Электронные данные. - Режим доступа: <https://www.mysql.com/>;
- [4] [Электронный ресурс]. – Электронные данные. - Режим доступа: <https://metanit.com/java/javafx/>;
- [5] [Электронный ресурс]. – Электронные данные. - Режим доступа: <http://qt-doc.ru> - “Модель "клиент-сервер””;
- [6] [Электронный ресурс]. – Электронные данные. - Режим доступа: <http://www.ru.wikipedia.org>;
- [7] Живицкая, Е.Н., Комаровский А.О., Швед О.И. Системный анализ и проектирование информационных систем. Лабораторный практикум: учеб-метод. Пособие – Минск: БГУИР, 2011.
- [8] Документация по архитектуре клиент-сервер [Электронный ресурс] – Режим доступа: <http://www.4stud.info/networking/>
- [9] Макконнелл, С. Совершенный код. –СПб. : Питер, 2005. –896 с

## Приложение А

### Листинг программного кода

#### Файл *Server.java*

```
package sample.classes;

import sample.classes.db.*;

import java.io.*;
import java.net.ServerSocket;
import java.net.Socket;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.HashSet;
import java.util.Objects;
import java.util.Set;

public class Server extends Thread {
    private Handler handler = new Handler();
    private Socket clientSocket = null;
    public Server(Socket socket) {
        this.clientSocket = socket;
    }

    public static void main(String[] args) throws IOException {
        ServerSocket serverSocket = new ServerSocket(8000);

        while(true) {
            Socket clientSocket = serverSocket.accept();
            System.out.println("Клиент подключился!");
            Server server = new Server(clientSocket);
            server.start();
        }

        public void run() {
            try{
                while (true){
                    ObjectOutputStream writerObj = new
ObjectOutputStream(clientSocket.getOutputStream());
                    ObjectInputStream readerObj = new
ObjectInputStream(clientSocket.getInputStream());

                    while (!clientSocket.isClosed()){
                        String action = (String) readerObj.readObject();
                        String request = "";
                        String choices = "";
                        String post = "";
                        String id = "";
                        String[] arr;

                        switch (action){

                            case "CHECK_VALID" -> {
                                String data =
```

```

String.valueOf(readerObj.readObject());
String[] splited = data.split("\\s+");

if (handler.AuthCheck(splited[0], splited[1]))
{
    System.out.println("Логин и пароль верный.");

writerObj.writeObject(handler.returnUser(splited[0], splited[1]));
}
else {
    System.out.println("Логин и пароль неверный,
либо не существует.");
    writerObj.writeObject("INVALID");
}
}

case "SHOW_UDOC" -> {
    System.out.println("Показать доки");
    ArrayList<Document> documents =
handler.show("document");
    ArrayList<UserDocument> userDocuments =
handler.show("user_document");
    ArrayList<Unique> result = new ArrayList<>();

    for(UserDocument ud:userDocuments){
        Unique unique = new
Unique("", "", "", "", "", "", "");

unique.setText1(String.valueOf(ud.getUser_id()));

unique.setText2(String.valueOf(ud.getDocument_id()));
        unique.setText5(ud.getConfirmDir());
        unique.setText6(ud.getConfirmAcc());
        for (Document doc:documents){
            if(ud.getDocument_id() == doc.getId()){
                unique.setText3(doc.getTitle());
                unique.setText4(doc.getDate());
            }
        }
        result.add(unique);
    }
    writerObj.writeObject(result);
}

case "SHOW_USERS" -> {
    ArrayList<Unique> result = new ArrayList<>();
    ArrayList<User> users = handler.show("user");
    ArrayList<Salary> salaris = handler.show("salary");
    for(User u: users){
        Unique unique = new
Unique("", "", "", "", "", "", "");

unique.setText1(String.valueOf(u.getId()));
unique.setText2(u.getName());
unique.setText3(u.getSurname());
unique.setText4(u.getPost());
        for(Salary sal:salaris){
            if(u.getId() == sal.getUserId()){

```



```

unique.setText5(String.valueOf(sal.getBasic()));

unique.setText6(String.valueOf(sal.getBonus()));

unique.setText7(String.valueOf(sal.getPension()));
    }
    }
    result.add(unique);
}
writerObj.writeObject(result);
}

case "SHOW_QUERY" -> {

    ArrayList<User> list = handler.set ch();
    writerObj.writeObject(list);
}

case "SHOW_AVG_SALARIES" -> {
    ArrayList<Unique> result = new ArrayList<>();
    ArrayList<Unique> posts_salaries = new
ArrayList<>();

    Set<String> posts_tmp = new HashSet<>();
    ArrayList<sample.classes.db.User> users =

handler.show("user");

    ArrayList<Salary> salaris = handler.show("salary");

    for(User u: users){
        posts tmp.add(u.getPost());
    }
    ArrayList<String> posts = new
ArrayList<>(posts_tmp);

    for(User u: users){
        Unique unique = new Unique("", "");
        unique.setText1(u.getPost());
        for(Salary sal:salaris){
            if(u.getId() == sal.getUserId()){

unique.setText2(String.valueOf(sal.getBasic()));
                }
            }
            posts_salaries.add(unique);
        }

        for (String s : posts) {
            Unique unique = new Unique("", "", "");
            int wrkrs = 0;
            int sumsal = 0;
            for (Unique posts salary : posts_salaries) {
                System.out.println(s);
                System.out.println(posts_salary.getText1());
                if (Objects.equals(s,
posts_salary.getText1())) {
                    wrkrs += 1;
                    sumsal +=
Integer.parseInt(posts_salary.getText2());
                }
            }
        }
    }
}

```

```

        System.out.println(s);
        System.out.println(wrkr);
        System.out.println(sumsal);
        int avgsal = sumsal / wrkr;
        unique.setText1(s);
        unique.setText2(String.valueOf(wrkr));
        unique.setText3(String.valueOf(avgsal));

        result.add(unique);

    }
    writerObj.writeObject(result);
}

case "CONFIRM" -> {
    Unique unique = (Unique) readerObj.readObject();
    String role =
String.valueOf(readerObj.readObject());
    if (Objects.equals(role, "DIRECTOR")){
handler.docSign(Integer.parseInt(unique.getText1()),
Integer.parseInt(unique.getText2()),1,"Одобрено");
    }
    else{
handler.docSign(Integer.parseInt(unique.getText1()),
Integer.parseInt(unique.getText2()),2,"Одобрено");
    }
}

case "DECLINE" -> {
    Unique unique = (Unique) readerObj.readObject();
    String role =
String.valueOf(readerObj.readObject());
    if (Objects.equals(role, "DIRECTOR")){
handler.docSign(Integer.parseInt(unique.getText1()),
Integer.parseInt(unique.getText2()),1,"Отклонено");
    }
    else{
handler.docSign(Integer.parseInt(unique.getText1()),
Integer.parseInt(unique.getText2()),2,"Отклонено");
    }
}

case "ADD_USER" -> {

    User user = (User) readerObj.readObject();
    Salary salary = (Salary) readerObj.readObject();
    handler.addUser(user);
    User buf = handler.returnUser(user.getLogin(),
user.getPassword());
    handler.addSalary(salary, buf.getId());
}

```

```

        handler.addVacation(buf.getId());
    }

    case "DELETE_USER" -> {
        Unique unique = (Unique) readerObj.readObject();
        int userId = Integer.parseInt(unique.getText1());
        handler.deleteUser(userId);
        handler.deleteSalary(userId);
        handler.deleteVacation(userId);
        handler.deleteUserDocument(userId);
    }

    case "UPDATE_POST" -> {
        String postt = (String) readerObj.readObject();
        String userId = (String) readerObj.readObject();

        handler.updatePost(Integer.parseInt(userId), postt);
    }

    case "UPDATE_SALARY" -> {

        String salary = (String) readerObj.readObject();
        String userId = (String) readerObj.readObject();
        handler.updateSalary(Integer.parseInt(userId),
Integer.parseInt(salary));
    }

    case "UPDATE_BONUS" -> {

        String bonus = (String) readerObj.readObject();
        String userId = (String) readerObj.readObject();
        handler.updateBonus(Integer.parseInt(userId),
Integer.parseInt(bonus));
    }

    case "UPDATE_PENSION" -> {

        String pension = (String) readerObj.readObject();
        String userId = (String) readerObj.readObject();
        handler.updatePension(Integer.parseInt(userId),
Integer.parseInt(pension));
    }

    case "UPDATE_VAC" -> {
        int userId = (int) readerObj.readObject();
        handler.updateVacation(userId);
    }

    case "SHOW_USERS_DIR" -> {

```

```

        ArrayList<Unique> result = new ArrayList<>();
        ArrayList<sample.classes.db.User> users =

handler.show("user");

        ArrayList<Salary> salaris = handler.show("salary");
        for (User u: users) {
            Unique unique = new Unique("", "", "", "", "", "");
            unique.setText1(String.valueOf(u.getId()));
            unique.setText2(u.getLogin());
            unique.setText3(u.getName());
            unique.setText4(u.getSurname());
            unique.setText5(u.getPost());
            for (Salary sal:salaris) {
                if (u.getId() == sal.getUserId()) {

unique.setText6(String.valueOf(sal.getBasic()));

                }
            }
            result.add(unique);
        }
        writerObj.writeObject(result);
    }

    case "SHOW_SEL_SALARY" -> {
        id = (String) readerObj.readObject();
        ArrayList<Unique> result = new ArrayList<>();
        ArrayList<sample.classes.db.User> users =

handler.show("user");

        ArrayList<Salary> salaris = handler.show("salary");
        for (User u: users) {
            if (u.getId() == Integer.parseInt(id)) {
                Unique unique = new Unique("", "", "");
                for (Salary sal:salaris) {
                    if (u.getId() == sal.getUserId()) {

unique.setText1(String.valueOf(sal.getBasic()));

unique.setText2(String.valueOf(sal.getBonus()));

unique.setText3(String.valueOf(sal.getPension()));

                    }
                }
                result.add(unique);
            }
        }
        writerObj.writeObject(result);
    }

    case "SHOW_SEL_VACATION" -> {
        id = (String) readerObj.readObject();
        ArrayList<Unique> result = new ArrayList<>();
        ArrayList<sample.classes.db.User> users =

handler.show("user");

        ArrayList<Vacation> vacations =

handler.show("vacation");

        for (User u: users) {
            if (u.getId() == Integer.parseInt(id)) {
                Unique unique = new Unique("", "", "");
                for (Vacation vac:vacations) {
                    if (u.getId() == vac.getUserId()) {

```

```

unique.setText1(String.valueOf(vac.getBasic()));
unique.setText2(String.valueOf(vac.getDaysLeft()));
unique.setText3(String.valueOf(vac.getDaysGot()));
    }
    }
    result.add(unique);
    }
    }
    writerObj.writeObject(result);
}

case "SET_WORKER" -> {
    id = (String) readerObj.readObject();
    ArrayList<Unique> result = new ArrayList<>();
    ArrayList<sample.classes.db.User> users =
handler.show("user");

    ArrayList<Salary> salaris = handler.show("salary");
    ArrayList<Vacation> vacations =
handler.show("vacation");

    for(User u: users){
        if(u.getId() == Integer.parseInt(id)){
            Unique unique = new
Unique("", "", "", "", "", "", "", "", "", "", "", "");
            unique.setText1(u.getName());
            unique.setText2(u.getSurname());
            unique.setText3(u.getPost());
            unique.setText4(u.getLogin());
            unique.setText5(u.getPassword());

unique.setText6(String.valueOf(u.getRole()));
            for(Salary sal:salaris){
                if(u.getId() == sal.getUserId()){

unique.setText7(String.valueOf(sal.getBasic()));
unique.setText8(String.valueOf(sal.getBonus()));
unique.setText9(String.valueOf(sal.getPension()));
                }
            }
            for(Vacation vac:vacations){
                if(u.getId() == vac.getUserId()){

unique.setText10(String.valueOf(vac.getDaysLeft()));
unique.setText11(String.valueOf(vac.getDaysGot()));
                }
            }
            result.add(unique);
        }
    }
    writerObj.writeObject(result);
}

case "ADD_DOC" ->{
    Document doc = (Document) readerObj.readObject();

```

```

        System.out.println(doc.toString());
        handler.addDoc(doc);
    }

    case "SET_WORKER_DOCS" -> {
        id = (String) readerObj.readObject();
        System.out.println("Показать доки");
        ArrayList<Document> documents =
handler.show("document");
        ArrayList<UserDocument> userDocuments =
handler.show("user document");
        ArrayList<Unique> result = new ArrayList<>();

        for (UserDocument ud:userDocuments) {
            Unique unique = new Unique("", "", "", "");
            unique.setText3(ud.getConfirmDir());
            unique.setText4(ud.getConfirmAcc());
            for (Document doc:documents) {
                if (ud.getDocument_id() == doc.getId()) {
                    unique.setText1(doc.getTitle());
                    unique.setText2(doc.getDate());
                }
            }
            if (Objects.equals(ud.getUser_id(),
Integer.parseInt(id))) {
                result.add(unique);
            }
        }
        writerObj.writeObject(result);
    }

    case "SET_CHOICES" -> {
        ArrayList list = handler.set_ch();
        writerObj.writeObject(list);
    }

    case "SHOW_POSTS" -> {
        request = (String) readerObj.readObject();
        ArrayList list = handler.showPost(request);
        writerObj.writeObject(list);
    }
}

readerObj.close();
writerObj.close();
clientSocket.close();
}
} catch (IOException | ClassNotFoundException | SQLException e) {
    e.printStackTrace();
}
}
}

```