

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«Национальный исследовательский ядерный университет «МИФИ»

Новоуральский технологический институт –

филиал федерального государственного автономного образовательного учреждения
высшего образования «Национальный исследовательский ядерный университет «МИФИ»

(НТИ НИЯУ МИФИ)

Колледж НТИ

Цикловая методическая комиссия информационных технологий

ОТЧЕТ №9

ПО ПРАКТИЧЕСКОМУ ЗАНЯТИЮ НА ТЕМУ

«ПРОЦЕДУРА АУТЕНТИФИКАЦИИ ПОЛЬЗОВАТЕЛЯ НА ОСНОВЕ ПАРОЛЯ»

ПМ.05 «Разработка программного обеспечения компьютерных сетей»

МДК.05.01 «Защита информации в КС»

Специальность СПО 09.02.03

«Программирование в компьютерных системах»

очная форма обучения

на базе основного общего образования

Выполнил

студент

группы

КПР–47 Д

Егорушкин

И.А.

11.12.2020

—
дата



Hello World

— □ ×

Авторизация

логин:

пароль:

Войти

Регистрация

Hello World

— □ ×

Регистрация

логин

пароль

ФИО

Город

Дата рождения

Телефон

Регистрация

Hello World

Пользователь

id :25b64f45-73c6-4da7-a901-89464647bd48

Login :1

FIO :1

City :егорушкин

Data :2021-01-06

Phone :егорушкин

Старый пароль

Новый пароль

Изменить пароль

открыть файл

подпись

Проверил
преподавател
ь
Горницкая
И.И.

дата

подпись

Новоуральск 2020

Цель работы: Анализ рисков информационной безопасности

Оборудование:

AMD Ryzen 5 3550U

ОЗУ 8 Гб

Программное обеспечение:

Windows 10 Professional 64 бит;

Ход работы:

Работа была поделена на несколько этапов.

2	7	Кириллица (строчные буквы)	При смене пароля: проверка на совпадение пароля с именем пользователя (если используется идентификационный номер, то в системе должны храниться имена каждого пользователя)
---	---	-------------------------------	---

1. Изучение материала
2. Анализ задания
3. Выполнение задания

В ходе изучения материала были выявлены основные задачи шифрации и типы.

В выполнение был выбран язык Java на основе Фреймворка Spring Boot.

Проект должен иметь следующие требования:

Кириллица (строчные буквы) При смене пароля: проверка на совпадение пароля с именем пользователя (если используется идентификационный номер, то в системе должны храниться имена каждого пользователя)

В качестве информационного ресурса использовать любой файл или приложение. 2. Доступ к ресурсу должен быть разрешен только санкционированным пользователям. Для этого в программе должны храниться имена пользователей и их пароли. При попытке доступа пользователя к ресурсу проверяется наличие его идентификатора (имени) в системе и соответствие введенного пароля паролю, который хранится в системе. 3. В системе должна храниться следующая информация о пользователе: ID или имя пользователя, пароль, ФИО, дата рождения, место рождения (город) номер телефона. 4. Пользователь должен иметь возможность поменять пароль (ограничения: см. вариант).

Текст программы

```
package sample;
```

```
import javafx.beans.value.ChangeListener;
import javafx.beans.value.ObservableValue;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.fxml.FXML;
import javafx.geometry.Pos;
import javafx.scene.Node;
import javafx.scene.control.*;
import javafx.scene.layout.HBox;
import javafx.scene.layout.Pane;
import javafx.scene.layout.Priority;
import javafx.scene.layout.VBox;

import java.time.Instant;
import java.time.LocalDate;
import java.time.ZoneId;
import java.time.format.DateTimeFormatter;
import java.util.*;
import java.util.concurrent.atomic.AtomicBoolean;

public class Controller {
    @FXML
    public VBox root;
    @FXML
```

```

public Label labile;
private VBox forma;
private Set<User> userSet;

public Controller() {
    userSet = new TreeSet<>(Comparator.comparing(User::getId));
}

@FXML
public void initialize() {
    difSettings(root);
    this.forma = login();
    this.labile.setText("Авторизация");
    root.getChildren().add(forma);
    HBox.setHgrow(root, Priority.ALWAYS);
    this.root.getChildren().forEach(this::difSettings);
}

private VBox login() {
    this.labile.setText("Авторизация");
    VBox root = new VBox();
    HBox error = new HBox();
    HBox login = new HBox();
    HBox password = new HBox();

    Label errorLabel = new Label();

    Label loginText = new Label("логин:");
    TextField loginField = new TextField();

    Label passwordText = new Label("пароль:");

```

```
TextField passwordField = new TextField();
```

```
Button loginButton = new Button("Войти");
```

```
Button regButton = new Button("Регистрация");
```

```
loginButton.setAlignment(Pos.CENTER);
```

```
error.getChildren().add(errorLabel);
```

```
login.getChildren().addAll(loginText, loginField);
```

```
password.getChildren().addAll(passwordText, passwordField);
```

```
root.getChildren().addAll(error, login, password, loginButton, regButton);
```

```
root.setAlignment(Pos.CENTER);
```

```
defSettings(loginButton);
```

```
defSettings(regButton);
```

```
loginButton.setMaxWidth(Double.MAX_VALUE);
```

```
regButton.setMaxWidth(Double.MAX_VALUE);
```

```
regButton.setOnMouseClicked(event -> {
```

```
    this.root.getChildren().remove(this.forma);
```

```
    this.forma = reg();
```

```
    this.root.getChildren().add(this.forma);
```

```
});
```

```
loginButton.setOnMouseClicked(event -> {
```

```
    userSet.stream().filter(user -> user.login.equals(loginField.getText()) &
```

```
        user.password.equals(passwordField.getText()))).forEach(user -> {
```

```
        this.root.getChildren().remove(this.forma);
```

```
        this.forma = user(user);
```

```
        this.root.getChildren().add(this.forma);
```

```
    });
```

```
});
```

```
setNumberFilter(passwordField);
```

```

        error.getChildren().forEach(this::difSettings);
        login.getChildren().forEach(this::difSettings);
        password.getChildren().forEach(this::difSettings);
        return root;
    }

```

```

private VBox user(User user) {
    VBox vBoxRoot = new VBox();
    Label error = new Label();
    this.labile.setText("ПОЛЬЗОВАТЕЛЬ");
    ObservableList<String> langs = FXCollections.observableArrayList();
    ListView<String> langsListView = new ListView<String>(langs);
    langs.add("id :" + user.getId());
    langs.add("Login :" + user.getLogin());
    langs.add("FIO :" + user.getFIO());
    langs.add("City :" + user.getCity());
    langs.add("Data :" + user.getData());
    langs.add("Phone :" + user.getPhone());
    langs.add("Password :" + user.getPassword());

```

```

    HBox oldPassword = IText("Старый пароль");
    HBox newPassword = IText("Новый пароль");
    TextField oldPasswordTF = (TextField) oldPassword.getChildren().get(1);
    TextField newPasswordTF = (TextField) newPassword.getChildren().get(1);
    setNumberFilter(oldPasswordTF);
    setNumberFilter(newPasswordTF);

```

```

    Button button = new Button("Изменить пароль");
    button.setOnMouseClicked(event -> {
        if (oldPasswordTF.getText().equals(user.getPassword())) {
            if (newPasswordTF.getText().length() < 7) {
                user.setPassword(newPasswordTF.getText());
                this.root.getChildren().remove(this.forma);
            }
        }
    });

```



```

        this.forma = login();
        this.root.getChildren().add(this.forma);
        error.setText("");
    } else {
        error.setText("Пароль должен быть больше или равен 8 символам");
    }
} else {
    error.setText("Неверный пароль");
}
});

```

```

vBoxRoot.getChildren().addAll(langsListView, oldPassword, newPassword, button);
this.root.getChildren().forEach(this::difSettings);

```

```

return vBoxRoot;
}

```

```

protected void setNumberFilter(TextField textBox) {
    textBox.textProperty().addListener(new ChangeListener<String>() {
        @Override
        public void changed(ObservableValue<? extends String> ov,
                            String oldValue, String newValue) {
            System.out.println(newValue);
            if (!newValue.matches("^[A-Яa-яЁё\\s]+$")) {
                try {
                    textBox.setText(oldValue);
                } catch (Exception e) {
                    textBox.setText("");
                }
            }
        }
    });
}

```

```

        }
    );
}

```

```

private VBox reg() {
    this.labile.setText("Регистрация");
    VBox vBoxRoot = new VBox();
    Label error = new Label("");
    HBox login = IText("логин");
    HBox password = IText("пароль");
    HBox FIO = IText("ФИО");
    HBox cite = IText("Город");
    HBox data = IText("Дата рождения");
    HBox phone = IText("Телефон");
    Button reg = new Button("Регистрация");
    vBoxRoot.getChildren()
        .addAll(new VBox(new Label()), new HBox(error), login, password, FIO, cite, data,
phone, reg);
    reg.setOnMouseClicked(event -> {
        User user = new User();
        user.setId(UUID.randomUUID().toString());
        vBoxRoot.getChildren().forEach(node -> {
            if (node instanceof HBox) {

                HBox hBox = (HBox) node;
                Label label = (Label) hBox.getChildren().get(0);
                try {

                    if (hBox.getChildren().get(1) instanceof DatePicker) {
                        DatePicker datePicker = (DatePicker) hBox.getChildren().get(1);
                        DateTimeFormatter dateFormatter =
                            DateTimeFormatter.ofPattern("dd-MM-yyyy");
                        LocalDate localDate = datePicker.getValue();

```

```

        Instant instant =
Instant.from(localDate.atStartOfDay(ZoneId.systemDefault()));
        Date date = Date.from(instant);
        System.out.println(localDate);
        user.setData(localDate + "");
    } else {
        TextField textField = (TextField) hBox.getChildren().get(1);
        switch (label.getText()) {
            case "логин":
                if (textField.getText().equals("")) {
                    error.setText(label.getText() + " пусг");
                } else {
                    error.setText("");
                }
                user.setLogin(textField.getText());
                break;
            case "пароль":
                if (textField.getText().length() < 7) {
                    error.setText(label.getText() + "Пароль должен быть больше или
равен 8 символам");
                }
                user.setPassword(textField.getText());
                break;
            case "ФИО":
                if (textField.getText().equals("")) {
                    error.setText(label.getText() + " пусг");
                }
                user.setFIO(textField.getText());
                break;
            case "Город":
                if (textField.getText().equals("")) {
                    error.setText(label.getText() + " пусг");
                }
                user.setCity(textField.getText());
                break;

```

```

        case "Телефон":
            if (textField.getText().equals("")) {
                error.setText(label.getText() + " пуст");
            }
            user.setPhone(textField.getText());
            break;
        }
    }
} catch (Exception e) {
}
}

));
AtomicBoolean dubl = new AtomicBoolean(false);
userSet.stream().filter(user1 -> user1.login.equals(user.login)).forEach(user1 -> {
    error.setText("Логин занят");
});
if (error.getText().equals("")) {
    this.userSet.add(user);
    this.root.getChildren().remove(this.forma);
    this.forma = login();
    this.root.getChildren().add(this.forma);
}
});
;
setNumberFilter((TextField) password.getChildren().get(1));
vBoxRoot.getChildren().forEach(this::difSettings);

return vBoxRoot;
}

private HBox lText(String s) {
    HBox lText = new HBox();
    Label label = new Label(s);
    if (s.equals("Дата рождения")) {

```

```

        DatePicker passwordField = new DatePicker();
        lText.getChildren().addAll(label, passwordField);
        difSettings(lText);
        difSettings(label);
        difSettings(passwordField);

    } else {
        TextField passwordField = new TextField();
        lText.getChildren().addAll(label, passwordField);
        difSettings(lText);
        difSettings(label);
        difSettings(passwordField);
    }

    return lText;
}

```

```

void difSettings(Node node) {
    HBox.setHgrow(node, Priority.ALWAYS);
    node.maxWidth(Double.MAX_VALUE);
    node.maxHeight(Double.MAX_VALUE);
    if (node instanceof VBox) {
        VBox vBox = (VBox) node;
        vBox.getChildren().forEach(this::difSettings);
    }
    if (node instanceof HBox) {
        HBox hBox = (HBox) node;
        hBox.getChildren().forEach(this::difSettings);
    }
    if (node instanceof Button) {
        Button button = (Button) node;
        button.setAlignment(Pos.CENTER);
    }
}

```

}

}

}

Пример работы программы

The image displays three screenshots of a desktop application titled 'Hello World'.

Top Left Screenshot: Пользователь (User)
This screen shows a list of user details in a table-like format. The first row is highlighted with a blue border. Below the table are input fields for 'Старый пароль' (Old password) and 'Новый пароль' (New password), followed by buttons 'Изменить пароль' (Change password) and 'открыть файл' (Open file).

Пользователь
id :25b64f45-73c6-4da7-a901-89464647bd48
Login :1
FIO :1
City :егорушкин
Data :2021-01-06
Phone :егорушкин

Старый пароль
Новый пароль
Изменить пароль
открыть файл

Top Right Screenshot: Регистрация (Registration)
This screen contains a registration form with input fields for 'логин' (login), 'пароль' (password), 'ФИО' (full name), 'Город' (city), 'Дата рождения' (date of birth), and 'Телефон' (phone). A 'Регистрация' button is at the bottom.

Регистрация

логин
пароль
ФИО
Город
Дата рождения
Телефон
Регистрация

Bottom Screenshot: Авторизация (Authorization)
This screen shows an authorization form with input fields for 'логин:' (login) and 'пароль:' (password). Below the fields are buttons for 'Войти' (Login) and 'Регистрация' (Registration).

Авторизация

логин:
пароль:
Войти
Регистрация

Вывод: В ходе работы были изучены методы авторизация по форме логин пароль.