Министерство просвещения Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

«Российский государственный профессионально-педагогический университет»

Институт инженерно-педагогического образования

Кафедра информационных систем и технологий

РАЗРАБОТКА WINDOWS ПРИЛОЖЕНИЯ ДЛЯ РАБОТЫ С ВНЕШНИМИ ИСТОЧНИКАМИ ДАННЫХ (БД) НА ЯЗЫКЕ ПРОГРАММИРОВАНИЯ JAVA

Курсовая работа

По дисциплине «Технологии программирования»

Студент группы Пим-201

Бирюков Д.В.

Руководитель:

Ассистент кафедры ИС

Шевцов А. С.

Екатеринбург 2023 г.

Оглавление

**[1. Java классы. 3](#_Toc25373)**

[1.1 Класс logins. 3](#_Toc4676)

[1.2 Класс loginsController. 5](#_Toc6793)

[1.3 Класс registrationsController. 8](#_Toc5693)

[1.4 Класс main\_menuController. 12](#_Toc9352)

[1.5 Класс config. 25](#_Toc30883)

[1.6 Класс dataBaseHandler. 25](#_Toc13776)

**[2. Файлы resources. 48](#_Toc28284)**

[2.1 Файл logins.fxml. 48](#_Toc24018)

[2.2 Файл registrations.fxml. 49](#_Toc17903)

[2.3 Файл main\_menu.fxml. 51](#_Toc9387)

[2.4 Файл style.css. 54](#_Toc22259)

[2.5 Файл dark\_style.css. 56](#_Toc22933)

[2.6 Файл logo.png. 60](#_Toc13935)

[2.7 Файл logo\_dark.png. 60](#_Toc7696)

# Java классы.

## 1.1 Класс logins.

package com.example.scheduleproject;

import javafx.application.Application;

import javafx.fxml.FXMLLoader;

import javafx.scene.Scene;

import javafx.stage.Stage;

import javafx.stage.StageStyle;

import java.io.IOException;

import java.util.Timer;

import java.util.TimerTask;

public class logins extends Application {

private double xOffset = 0; // Смещение по оси X

private double yOffset = 0; // Смещение по оси Y

@Override

public void start(Stage stage) throws IOException {

var fxmlLoader = new FXMLLoader(logins.class.getResource("logins.fxml"));

Scene scene = new Scene(fxmlLoader.load());

stage.setTitle("Schedule RSVPU"); // Заголовок окна

stage.setResizable(false); // Запрет изменения размеров окна

stage.initStyle(StageStyle.UNDECORATED); // Удаление оформления окна (без рамок и кнопок закрытия и минимизации)

stage.setScene(scene);

scene.setOnMousePressed(event -> { // Обработчик события нажатия кнопки мыши на сцене

xOffset = event.getSceneX();

yOffset = event.getSceneY();

});

scene.setOnMouseDragged(event -> { // Обработчик события перемещения мыши по сцене

stage.setX(event.getScreenX() - xOffset);

stage.setY(event.getScreenY() - yOffset);

});

stage.show(); // Отображение окна

Timer timer = new Timer();

dataBaseHandler db = new dataBaseHandler();

TimerTask parsingAll = new TimerTask() {

@Override

public void run() {

db.parsingAll();

}

};

TimerTask parsingDictionaries = new TimerTask() {

@Override

public void run() {

db.parsingDictionaries();

}

};

TimerTask parsingOnlySchedule = new TimerTask() {

@Override

public void run() {

db.parsingOnlySchedule();

}

};

// Запускаем задачу через 1 секунд

timer.schedule(parsingAll, 1000);

// Повторяем задачу каждые 30 минут, начиная с момента запуска

timer.schedule(parsingOnlySchedule, 1800000, 1800000);

// Повторяем задачу каждые сутки, начиная с момента запуска

timer.schedule(parsingDictionaries, 86400000, 86400000);

}

public static void main(String[] args) {

launch(); // Запуск приложения

}

}

## 1.2 Класс loginsController.

package com.example.scheduleproject;

import java.io.IOException;

import java.util.List;

import javafx.animation.KeyFrame;

import javafx.animation.Timeline;

import javafx.event.ActionEvent;

import javafx.fxml.FXML;

import javafx.fxml.FXMLLoader;

import javafx.scene.Node;

import javafx.scene.Parent;

import javafx.scene.Scene;

import javafx.scene.control.Button;

import javafx.scene.control.Label;

import javafx.scene.control.PasswordField;

import javafx.scene.control.TextField;

import javafx.scene.input.MouseEvent;

import javafx.stage.Stage;

import javafx.util.Duration;

public class loginsController {

@FXML

public Button signUp; // Кнопка "Зарегистрироваться"

@FXML

public Button registrations; // Кнопка "Регистрация"

@FXML

private Button closeBut; // Кнопка "Закрыть"

@FXML

private Label notification; // Метка для отображения уведомлений

@FXML

private PasswordField password; // Поле ввода пароля

@FXML

private TextField login; // Поле ввода логина

private double xOffset = 0; // Смещение по оси X

private double yOffset = 0; // Смещение по оси Y

dataBaseHandler db = new dataBaseHandler(); // Объект для работы с базой данных

List<String> namePersons = db.retrieveNamePersonData(); // Список имен пользователей

// Метод инициализации контроллера

@FXML

void initialize() {

}

// Обработчик события нажатия кнопки мыши

@FXML

private void onMousePressed(MouseEvent event) {

xOffset = event.getSceneX(); // Запоминаем текущее положение мыши по оси X

yOffset = event.getSceneY(); // Запоминаем текущее положение мыши по оси Y

}

// Обработчик события перемещения мыши

@FXML

private void onMouseDragged(MouseEvent event) {

Stage stage = (Stage) ((Node) event.getSource()).getScene().getWindow();

stage.setX(event.getScreenX() - xOffset); // Задаем новое положение окна по оси X

stage.setY(event.getScreenY() - yOffset); // Задаем новое положение окна по оси Y

}

// Обработчик события нажатия кнопки "Sign Up"

@FXML

public void signUp(ActionEvent event) {

if (login.getText().isEmpty() || password.getText().isEmpty()){

login.clear();

password.clear();

showNotification(notification, "Пустое поле логин или пароль"); // Показываем уведомление о пустых полях логина или пароля

} else if (!db.arrayEntry(login.getText(), namePersons)){

login.clear();

password.clear();

showNotification(notification, "Такого пользователя нет"); // Показываем уведомление о несуществующем пользователе

} else if (!db.accountLogin(login.getText(), dataBaseHandler.encodePassword(password.getText()))){

password.clear();

showNotification(notification, "Пароль неправильный"); // Показываем уведомление о неправильном пароле

} else {

try {

FXMLLoader fxmlLoader = new FXMLLoader(getClass().getResource("main\_menu.fxml"));

Stage stage = (Stage) ((Node) event.getSource()).getScene().getWindow();

stage.setScene(new Scene(fxmlLoader.load()));

stage.setResizable(false);

main\_menuController controller = fxmlLoader.getController();

controller.setPersonName(login.getText());

controller.setPersonGroup(db.getGroup(login.getText()));

controller.setPersonSubgroup(db.getSubgroup(login.getText()));

controller.setPersonTheme(db.getTheme(login.getText()));

// controller.setScene(scene);

db.close();

stage.show();

} catch (IOException e) {

throw new RuntimeException(e);

}

}

}

// Обработчик события нажатия кнопки "Регистрация"

@FXML

public void registrations(ActionEvent event) {

try {

db.close();

FXMLLoader fxmlLoader = new FXMLLoader(getClass().getResource("registrations.fxml"));

Parent root = fxmlLoader.load();

Scene scene = new Scene(root);

Stage stage = (Stage) ((Node) event.getSource()).getScene().getWindow();

stage.setScene(scene);

stage.setResizable(false);

stage.show();

} catch (IOException e) {

throw new RuntimeException(e);

}

}

// Метод для закрытия окна

@FXML

public void closeBut() {

db.close();

Stage stage = (Stage) closeBut.getScene().getWindow();

stage.close();

}

// Метод для отображения уведомления

@FXML

private void showNotification(Label notificationLabel, String message) {

notificationLabel.setText(message);

notificationLabel.setVisible(true);

Timeline timeline = new Timeline(new KeyFrame(Duration.seconds(2), evt -> notificationLabel.setVisible(false)));

timeline.play();

}

}

## 1.3 Класс registrationsController.

package com.example.scheduleproject;

import javafx.animation.KeyFrame;

import javafx.animation.Timeline;

import javafx.event.ActionEvent;

import javafx.fxml.FXML;

import javafx.fxml.FXMLLoader;

import javafx.scene.Node;

import javafx.scene.Parent;

import javafx.scene.Scene;

import javafx.scene.control.Button;

import javafx.scene.control.ComboBox;

import javafx.scene.control.Label;

import javafx.scene.control.PasswordField;

import javafx.scene.control.TextField;

import javafx.scene.input.MouseEvent;

import javafx.stage.Stage;

import javafx.util.Duration;

import java.io.IOException;

import java.util.List;

public class registrationsController{

@FXML

public ComboBox<String> subgroups; // Выпадающий список для выбора подгруппы

@FXML

public Button signUp; // Кнопка "Зарегистрироваться"

@FXML

public Button registrations; // Кнопка "Регистрация"

@FXML

private Label notification; // Метка для отображения уведомлений

@FXML

private ComboBox<String> groups; // Выпадающий список для выбора группы

@FXML

private PasswordField password2; // Поле ввода подтверждения пароля

@FXML

private PasswordField password1; // Поле ввода пароля

@FXML

private TextField login; // Поле ввода логина

@FXML

private Button closeBut; // Кнопка "Закрыть"

private double xOffset = 0; // Смещение по оси X

private double yOffset = 0; // Смещение по оси Y

dataBaseHandler db = new dataBaseHandler(); // Объект для работы с базой данных

List<String> namePersons = db.retrieveNamePersonData(); // Список имен пользователей

// Метод инициализации контроллера

@FXML

void initialize() {

groups.getItems().addAll(db.retrieveNameGroupData()); // Заполняем выпадающий список групп из базы данных при инициализации окна

}

// Обработчик события нажатия кнопки мыши

@FXML

private void onMousePressed(MouseEvent event) {

xOffset = event.getSceneX(); // Запоминаем текущее положение мыши по оси X

yOffset = event.getSceneY(); // Запоминаем текущее положение мыши по оси Y

}

// Обработчик события перемещения мыши

@FXML

private void onMouseDragged(MouseEvent event) {

Stage stage = (Stage) ((Node) event.getSource()).getScene().getWindow();

stage.setX(event.getScreenX() - xOffset); // Задаем новое положение окна по оси X

stage.setY(event.getScreenY() - yOffset); // Задаем новое положение окна по оси Y

}

// Обработчик события нажатия кнопки "Sign Up"

@FXML

public void signUp(ActionEvent event) {

try {

db.close();

FXMLLoader fxmlLoader = new FXMLLoader(getClass().getResource("logins.fxml"));

Parent root = fxmlLoader.load();

Scene scene = new Scene(root);

Stage stage = (Stage) ((Node) event.getSource()).getScene().getWindow();

stage.setScene(scene);

stage.setResizable(false);

stage.show();

} catch (IOException e) {

throw new RuntimeException(e);

}

}

// Обработчик события нажатия кнопки "Регистрация"

@FXML

public void registrations(ActionEvent event) {

if (login.getText().equals("") || password1.getText().equals("") || password2.getText().equals("") || groups.getValue() == null || subgroups.getValue() == null){

showNotification(notification, "Заполните все поля"); // Проверка на заполнение всех полей формы регистрации

} else if (login.getText().length() < 5 || password1.getText().length() < 5){

login.clear();

password1.clear();

password2.clear();

showNotification(notification, "Минимальная длина логина и пароля 5 символов"); // Проверка на минимальную длину логина и пароля

} else if (db.arrayEntry(login.getText(), namePersons)){

login.clear();

showNotification(notification, "Логин занят"); // Проверка на уникальность логина

} else if (!password1.getText().equals(password2.getText())){

password1.clear();

password2.clear();

showNotification(notification, "Пароли не совпадают"); // Проверка на совпадение паролей

} else {

int subgroup;

try {

subgroup = Integer.parseInt(subgroups.getValue());

} catch (NumberFormatException e) {

subgroup = 0;

}

db.insertPersons(subgroup, login.getText(), dataBaseHandler.encodePassword(password1.getText()), groups.getValue()); // Регистрация пользователя в базе данных

try {

db.close();

FXMLLoader fxmlLoader = new FXMLLoader(getClass().getResource("logins.fxml"));

Parent root = fxmlLoader.load();

Scene scene = new Scene(root);

Stage stage = (Stage) ((Node) event.getSource()).getScene().getWindow();

stage.setScene(scene);

stage.setResizable(false);

stage.show();

} catch (IOException e) {

throw new RuntimeException(e);

}

}

}

// Метод для закрытия окна

@FXML

public void closeBut() {

db.close();

Stage stage = (Stage) closeBut.getScene().getWindow();

stage.close();

}

// Метод для отображения уведомления

@FXML

private void showNotification(Label notificationLabel, String message) {

notificationLabel.setText(message);

notificationLabel.setVisible(true);

Timeline timeline = new Timeline(new KeyFrame(Duration.seconds(2), evt -> notificationLabel.setVisible(false)));

timeline.play();

}

}

## 1.4 Класс main\_menuController.

package com.example.scheduleproject;

import javafx.animation.KeyFrame;

import javafx.animation.Timeline;

import javafx.event.ActionEvent;

import javafx.fxml.FXML;

import javafx.fxml.FXMLLoader;

import javafx.scene.Node;

import javafx.scene.Parent;

import javafx.scene.Scene;

import javafx.scene.control.\*;

import javafx.scene.input.MouseEvent;

import javafx.scene.layout.AnchorPane;

import javafx.scene.layout.VBox;

import javafx.stage.Stage;

import javafx.util.Duration;

import java.io.IOException;

import java.time.DayOfWeek;

import java.time.LocalDate;

import java.time.format.DateTimeFormatter;

import java.time.temporal.ChronoUnit;

import java.util.\*;

public class main\_menuController {

@FXML

private Label dateWeek; // Метка для отображения текущей недели

@FXML

private AnchorPane rootPane; // Основная область приложения

@FXML

private Label schedulDay; // Метка для отображения расписания на выбранный день

@FXML

private VBox taskContainer; // Контейнер для задач пользователя

@FXML

private TextField taskTextField; // Поле ввода для добавления новой задачи

@FXML

private AnchorPane menu; // Панель для отображения меню пользователя

@FXML

private AnchorPane blackout; // Затемненная область при открытом меню

@FXML

private Label login; // Метка для отображения имени пользователя

@FXML

private ComboBox<String> group; // Выпадающий список для выбора группы пользователя

@FXML

private ComboBox<String> subgroup; // Выпадающий список для выбора подгруппы пользователя

@FXML

private ToggleButton darkStyle; // Кнопка для включения/выключения темной темы

@FXML

private PasswordField oldPassword; // Поле ввода текущего пароля пользователя

@FXML

private PasswordField newPassword; // Поле ввода нового пароля пользователя

@FXML

private Label notification; // Метка для отображения уведомлений

@FXML

private Button rightBut; // Кнопка для переключения на следующую неделю

@FXML

private Button leftBut; // Кнопка для переключения на предыдущую неделю

@FXML

private Button pn; // Кнопка для выбора понедельника

@FXML

private Button vt; // Кнопка для выбора вторника

@FXML

private Button sr; // Кнопка для выбора среды

@FXML

private Button ct; // Кнопка для выбора четверга

@FXML

private Button pt; // Кнопка для выбора пятницы

@FXML

private Button sb; // Кнопка для выбора субботы

@FXML

private Button vs; // Кнопка для выбора воскресенья

@FXML

private Button closeBut; // Кнопка для закрытия приложения

private int page = 1; // Текущая страница (1 - первая неделя)

private int activeDay; // Активный день недели

private String personName; // Имя пользователя

private String personGroup; // Группа пользователя

private int personSubgroup; // Подгруппа пользователя

private boolean personTheme; // Тема интерфейса пользователя (светлая/темная)

private double xOffset = 0; // Смещение по оси X для перемещения окна

private double yOffset = 0; // Смещение по оси Y для перемещения окна

dataBaseHandler db = new dataBaseHandler(); // Обработчик базы данных

LocalDate today = LocalDate.now(); // Текущая дата

Timer timer = new Timer();

DayOfWeek dayOfWeek = today.getDayOfWeek(); // День недели текущей даты

int numericValue = dayOfWeek.getValue(); // Числовое значение дня недели (1-понедельник, 2-вторник и т.д.)

LocalDate startDate = today.minus(numericValue - 1, ChronoUnit.DAYS); // Дата начала текущей недели

private Scene scene;

@FXML

private void onMousePressed(MouseEvent event) {

xOffset = event.getSceneX();

yOffset = event.getSceneY();

}

@FXML

private void onMouseDragged(MouseEvent event) {

Stage stage = (Stage) ((Node) event.getSource()).getScene().getWindow();

stage.setX(event.getScreenX() - xOffset);

stage.setY(event.getScreenY() - yOffset);

}

@FXML

void initialize() {

// Стартовые действия видимости объектов

leftBut.setStyle("-fx-opacity: 0;");

rightBut.setStyle("-fx-opacity: .50;");

blackout.setVisible(false);

menu.setVisible(false);

// rootPane.setVisible(false);

// Установка активной недели

dateWeek.setText(startDate.format(DateTimeFormatter.ofPattern("dd.MM.yyyy")) + " - " + startDate.plus(6, ChronoUnit.DAYS).format(DateTimeFormatter.ofPattern("dd.MM.yyyy")));

TimerTask start = new TimerTask() {

@Override

public void run() {

scene = rootPane.getScene();

// Установка стартовой темы

if (getPersonTheme()) {

setTheme("dark\_style.css");

darkStyle.fire();

darkStyle.setText("On");

} else {

setTheme("style.css");

darkStyle.setText("Off");

}

// Установка активного дня

setActiveDay();

}

};

timer.schedule(start, 10);

}

@FXML

public void menuBut() {

// Установка логина пользователя

login.setText(getPersonName());

blackout.setVisible(true);

menu.setVisible(true);

// Установка группы пользователя

setPersonGroup(db.getGroup(getPersonName()));

group.getItems().addAll(db.retrieveNameGroupData());

group.setValue(getPersonGroup());

// Установка подгруппы пользователя

setPersonSubgroup(db.getSubgroup(getPersonName()));

subgroup.setValue(String.valueOf(getPersonSubgroup()));

}

@FXML

public void OkMenuBut() {

// Обновление группы, подгруппы и темы пользователя

int subgroups;

try {

subgroups = Integer.parseInt(subgroup.getValue());

} catch (NumberFormatException e) {

subgroups = 0;

}

setPersonGroup(group.getValue());

setPersonSubgroup(subgroups);

db.updateGroupAndSubGroupAndTheme(getPersonName(), getPersonGroup(), getPersonSubgroup(), getPersonTheme());

// Обновление пароля пользователя

notification.setStyle("-fx-text-fill: #eb5643;");

if (!oldPassword.getText().equals("") || !newPassword.getText().equals("")) {

if (oldPassword.getText().equals(newPassword.getText())) {

newPassword.clear();

showNotification(notification, "Пароли совпадают");

} else if (newPassword.getText().length() < 5) {

newPassword.clear();

showNotification(notification, "Минимальная длина 5");

} else if (!db.accountLogin(getPersonName(), dataBaseHandler.encodePassword(oldPassword.getText()))) {

oldPassword.clear();

showNotification(notification, "Пароль не верен");

} else {

db.updatePassword(getPersonName(), dataBaseHandler.encodePassword(newPassword.getText()));

notification.setStyle("-fx-text-fill: #3f7733;");

notification.setText("Данные изменены");

notification.setVisible(true);

Timeline timeline = new Timeline(new KeyFrame(Duration.seconds(1), evt -> {

oldPassword.clear();

newPassword.clear();

notification.setVisible(false);

blackout.setVisible(false);

menu.setVisible(false);

if (page == 1) {

setActiveDay();

} else {

pn.fire();

}

}));

timeline.play();

}

} else {

notification.setStyle("-fx-text-fill: #3f7733;");

notification.setText("Ok");

notification.setVisible(true);

Timeline timeline = new Timeline(new KeyFrame(Duration.seconds(1), evt -> {

oldPassword.clear();

newPassword.clear();

notification.setVisible(false);

blackout.setVisible(false);

menu.setVisible(false);

if (page == 1) {

setActiveDay();

} else {

pn.fire();

}

}));

timeline.play();

}

}

@FXML

public void closeMenuBut() {

oldPassword.clear();

newPassword.clear();

blackout.setVisible(false);

menu.setVisible(false);

if (page == 1) {

setActiveDay();

} else {

pn.fire();

}

}

@FXML

public void addTaskBut() {

String taskText = taskTextField.getText();

if (!taskText.isEmpty()) {

db.insertTask(getPersonName(), taskText, startDate.plus(activeDay, ChronoUnit.DAYS), false);

CheckBox taskCheckBox = new CheckBox(taskText);

taskCheckBox.setOnAction(event -> handleCheckBoxAction(taskCheckBox));

taskContainer.getChildren().add(taskCheckBox);

taskTextField.clear();

}

}

private void handleCheckBoxAction(CheckBox checkBox) {

db.updateTask(getPersonName(), startDate.plus(activeDay, ChronoUnit.DAYS),checkBox.getText(), checkBox.isSelected());

}

@FXML

public void darkStyle() {

if (darkStyle.isSelected()) {

darkStyle.setText("On");

setTheme("dark\_style.css");

setPersonTheme(true);

} else {

darkStyle.setText("Off");

setTheme("style.css");

setPersonTheme(false);

}

}

@FXML

public void rightBut() {

page = 2;

rightBut.setStyle("-fx-opacity: 0;");

leftBut.setStyle("-fx-opacity: .50;");

dateWeek.setText(startDate.plus(7, ChronoUnit.DAYS).format(DateTimeFormatter.ofPattern("dd.MM.yyyy")) + " - " + startDate.plus(13, ChronoUnit.DAYS).format(DateTimeFormatter.ofPattern("dd.MM.yyyy")));

pn.fire();

}

@FXML

public void leftBut() {

page = 1;

leftBut.setStyle("-fx-opacity: 0;");

rightBut.setStyle("-fx-opacity: .50;");

dateWeek.setText(startDate.format(DateTimeFormatter.ofPattern("dd.MM.yyyy")) + " - " + startDate.plus(6, ChronoUnit.DAYS).format(DateTimeFormatter.ofPattern("dd.MM.yyyy")));

setActiveDay();

}

@FXML

public void pn() {

setDayStyle(1);

}

@FXML

public void vt() {

setDayStyle(2);

}

@FXML

public void sr() {

setDayStyle(3);

}

@FXML

public void ct() {

setDayStyle(4);

}

@FXML

public void pt() {

setDayStyle(5);

}

@FXML

public void sb() {

setDayStyle(6);

}

@FXML

public void vs() {

setDayStyle(7);

}

@FXML

public void closeBut() {

db.close();

Stage stage = (Stage) closeBut.getScene().getWindow();

stage.close();

}

@FXML

public void out(ActionEvent event) {

try {

db.close();

FXMLLoader fxmlLoader = new FXMLLoader(getClass().getResource("logins.fxml"));

Parent root = fxmlLoader.load();

Scene scene = new Scene(root);

Stage stage = (Stage) ((Node) event.getSource()).getScene().getWindow();

stage.setScene(scene);

stage.setResizable(false);

stage.show();

} catch (IOException e) {

throw new RuntimeException(e);

}

db.close();

}

// Метод для установки стиля и выполнения общих операций для дней недели

private void setDayStyle(int day) {

pn.setStyle("-fx-opacity: " + (day == 1 ? "1" : ".32") + ";");

vt.setStyle("-fx-opacity: " + (day == 2 ? "1" : ".32") + ";");

sr.setStyle("-fx-opacity: " + (day == 3 ? "1" : ".32") + ";");

ct.setStyle("-fx-opacity: " + (day == 4 ? "1" : ".32") + ";");

pt.setStyle("-fx-opacity: " + (day == 5 ? "1" : ".32") + ";");

sb.setStyle("-fx-opacity: " + (day == 6 ? "1" : ".32") + ";");

vs.setStyle("-fx-opacity: " + (day == 7 ? "1" : ".32") + ";");

if (page == 1) {

schedulDay.setText(db.getScheduleDay(startDate.plus(day - 1, ChronoUnit.DAYS), getPersonGroup(), getPersonSubgroup()));

activeDay = day - 1;

} else {

schedulDay.setText(db.getScheduleDay(startDate.plus(day + 6, ChronoUnit.DAYS), getPersonGroup(), getPersonSubgroup()));

activeDay = day + 6;

}

taskContainer.getChildren().clear();

for (String e : db.getTasks(getPersonName(), startDate.plus(activeDay, ChronoUnit.DAYS))) {

CheckBox taskCheckBox = new CheckBox(e);

taskCheckBox.setSelected(db.getTaskStatus(getPersonName(), startDate.plus(activeDay, ChronoUnit.DAYS), e));

taskCheckBox.setOnAction(event -> handleCheckBoxAction(taskCheckBox));

taskContainer.getChildren().add(taskCheckBox);

}

}

// Метод для установки активного дня недели

private void setActiveDay() {

switch (numericValue) {

case 1 -> pn.fire();

case 2 -> vt.fire();

case 3 -> sr.fire();

case 4 -> ct.fire();

case 5 -> pt.fire();

case 6 -> sb.fire();

case 7 -> vs.fire();

}

}

// Метод для вывода уведомления на объект label на 2 екунды

private void showNotification(Label notificationLabel, String message) {

notificationLabel.setText(message);

notificationLabel.setVisible(true);

Timeline timeline = new Timeline(new KeyFrame(Duration.seconds(2), evt -> notificationLabel.setVisible(false)));

timeline.play();

}

// Устанавливает тему css

private void setTheme(String theme) {

scene.getStylesheets().clear();

scene.getStylesheets().add(Objects.requireNonNull(getClass().getResource(theme)).toExternalForm());

}

// Возвращает имя персоны

public String getPersonName() {

return personName;

}

// Устанавливает имя персоны

public void setPersonName(String personName) {

this.personName = personName;

}

// Устанавливает группу персоны

public void setPersonGroup(String personGroup) {

this.personGroup = personGroup;

}

// Возвращает группу персоны

public String getPersonGroup() {

return personGroup;

}

// Устанавливает подгруппу персоны

public void setPersonSubgroup(int personSubgroup) {

this.personSubgroup = personSubgroup;

}

// Возвращает подгруппу персоны

public int getPersonSubgroup() {

return personSubgroup;

}

// Возвращает тему персоны

public boolean getPersonTheme() {

return personTheme;

}

// Устанавливает тему персоны

public void setPersonTheme(boolean personTheme) {

this.personTheme = personTheme;

}

}

## 1.5 Класс config.

package com.example.scheduleproject;

// Класс для хранения конфигурации

public class config {

// Переменные для подключения к базе данных

protected String USER = "postgres"; // Имя пользователя

protected String PASSWORD = "3966"; // Пароль

protected String URL = "jdbc:postgresql://localhost/ScheduleDB"; // URL-адрес базы данных

// URL-адреса API

protected String SCHEDULE = "http://forum.rsvpu.ru//contents/api/rasp.php"; // Адрес для получения расписания

protected String GROUPS = "http://forum.rsvpu.ru/contents/api/groups.php"; // Адрес для получения групп

protected String AUDIENCES = "http://forum.rsvpu.ru/contents/api/auds.php"; // Адрес для получения аудиторий

protected String LECTURERS = "http://forum.rsvpu.ru/contents/api/preps.php"; // Адрес для получения преподавателей

}

## 1.6 Класс dataBaseHandler.

package com.example.scheduleproject;

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.net.HttpURLConnection;

import java.nio.charset.StandardCharsets;

import java.security.MessageDigest;

import java.security.NoSuchAlgorithmException;

import java.sql.\*;

import java.time.LocalDate;

import java.time.format.DateTimeFormatter;

import java.util.ArrayList;

import java.util.List;

import java.util.logging.Logger;

import org.json.JSONArray;

import org.json.JSONObject;

import java.net.URL;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

public class dataBaseHandler extends config {

private static final Logger LOGGER = Logger.getLogger(dataBaseHandler.class.getName());

private Connection conn;

// Соеденения с БД

public dataBaseHandler() {

try {

Class.forName("org.postgresql.Driver");

conn = DriverManager.getConnection(URL, USER, PASSWORD);

} catch (ClassNotFoundException | SQLException e) {

LOGGER.warning("[Error] DB connection: " + e.getMessage());

}

}

// Закрытие соеденения с БД

public void close() {

try {

if (conn != null) {

conn.close();

}

} catch (SQLException e) {

LOGGER.warning("[Error] DB disconnection: " + e.getMessage());

}

}

// Метод проверки входа элемента в массив

public boolean arrayEntry(String name, List<String> arr) {

return arr.contains(name);

}

// Хэширование пароля

public static String encodePassword(String password) {

try {

MessageDigest md = MessageDigest.getInstance("MD5");

byte[] mdBytes = md.digest(password.getBytes(StandardCharsets.UTF\_8));

StringBuilder sb = new StringBuilder();

for (byte mdByte : mdBytes) {

sb.append(Integer.toString((mdByte & 0xff) + 0x100, 16).substring(1));

}

return sb.toString();

} catch (NoSuchAlgorithmException e) {

throw new RuntimeException(e);

}

}

// Запросник расписания

public JSONArray jsonSchedule() {

JSONArray schedule = new JSONArray();

try {

// Make API request to retrieve group data

String groupUrl = GROUPS;

String groupResponse = makeApiRequest(groupUrl);

JSONArray groupArray = new JSONArray(groupResponse);

for (int i = 0; i < groupArray.length(); i++) {

JSONObject group = groupArray.getJSONObject(i);

var groupId = group.getInt("oid");

// Make API request to retrieve schedule data for each group

String scheduleUrl = SCHEDULE + "?&v\_gru=" + groupId;

String scheduleResponse = makeApiRequest(scheduleUrl);

if (scheduleResponse.isEmpty()) {

continue;

}

JSONArray scheduleArray = new JSONArray(scheduleResponse);

for (int j = 0; j < scheduleArray.length(); j++) {

JSONObject scheduleObject = scheduleArray.getJSONObject(j);

JSONObject contentObject = scheduleObject.getJSONObject("content");

JSONObject arr = new JSONObject();

// Добавление date, time, lesson, group в schedule

arr.put("date", scheduleObject.getString("date"));

arr.put("time", scheduleObject.getString("time"));

arr.put("lesson", scheduleObject.getInt("lesson"));

arr.put("group", Integer.toString(groupId));

JSONObject content = new JSONObject();

// Добавление disciplina в content

content.put("disciplina", contentObject.getString("disciplina"));

// Добавление type\_disciplina в content

if (!contentObject.isNull("type\_disciplina")) {

content.put("type\_disciplina", contentObject.getString("type\_disciplina"));

} else {

content.put("type\_disciplina", "-"); // Set empty string if null

}

// Добавление aud в content

if (!contentObject.isNull("aud")) {

content.put("aud", contentObject.getString("aud"));

} else {

content.put("aud", "-"); // Set empty string if null

}

// Добавление lecturer в content

content.put("lecturer", contentObject.getString("lecturer"));

// Добавление subgroupname в content

if (!contentObject.isNull("subgroupname")) {

// Определяем регулярное выражение для поиска числа в круглых скобках

Pattern pattern = Pattern.compile("\\((\\d+)\\)");

// Создаем Matcher объект для строки str

Matcher matcher = pattern.matcher(contentObject.getString("subgroupname"));

// Проверяем, есть ли совпадение

if (matcher.find()) {

// Извлекаем найденное число

String number = matcher.group(1);

content.put("subgroupname", number);

}

} else {

content.put("subgroupname", "0");

}

// Добавление content в schedule

arr.put("content", content);

schedule.put(arr);

}

}

} catch (Exception e) {

LOGGER.warning("[Error] Failed to retrieve json schedule:" + e.getMessage());

}

return schedule;

}

// Метод создания API запросов

private String makeApiRequest(String url) throws IOException {

URL apiURL = new URL(url);

HttpURLConnection connection = (HttpURLConnection) apiURL.openConnection();

connection.setRequestMethod("GET");

int responseCode = connection.getResponseCode();

if (responseCode == HttpURLConnection.HTTP\_OK) {

BufferedReader in = new BufferedReader(new InputStreamReader(connection.getInputStream()));

String inputLine;

StringBuilder response = new StringBuilder();

while ((inputLine = in.readLine()) != null) {

response.append(inputLine);

}

in.close();

return response.toString();

} else {

throw new IOException("Failed to make API request. Response code: " + responseCode);

}

}

// Метод получения доступа к аккаунту пользователя

public boolean accountLogin(String login, String password) {

boolean flag = false;

try {

PreparedStatement stmt = conn.prepareStatement("SELECT person\_password FROM public.persons WHERE person\_name=?;");

stmt.setString(1, login);

ResultSet rs = stmt.executeQuery();

if (rs.next()) {

String passwordFromDb = rs.getString("person\_password");

flag = passwordFromDb.equals(password);

}

rs.close();

stmt.close();

} catch (SQLException e) {

LOGGER.warning("[Error] DB query accountLogin: " + e.getMessage());

}

return flag;

}

// Set zone

// Добовление пользователя

public void insertPersons(int subgroup, String personName, String personPassword, String groupName) {

try {

String query = "INSERT INTO persons(subgroup, person\_name, person\_password, name\_group) VALUES(?, ?, ?, ?)";

PreparedStatement pstmt = conn.prepareStatement(query);

pstmt.setInt(1, subgroup);

pstmt.setString(2, personName);

pstmt.setString(3, personPassword);

pstmt.setString(4, groupName);

pstmt.executeUpdate();

pstmt.close();

} catch (SQLException e) {

LOGGER.warning("[Error] User not added: " + e.getMessage());

}

}

// Добавление задачи ежедневника

public void insertTask(String person\_name, String task, LocalDate task\_date, boolean status) {

try {

String query = "INSERT INTO person\_task(person\_name, task, task\_date, task\_status) VALUES(?, ?, ?, ?)";

PreparedStatement stmt = conn.prepareStatement(query);

stmt.setString(1, person\_name);

stmt.setString(2, task);

stmt.setDate(3, Date.valueOf(task\_date));

stmt.setBoolean(4, status);

stmt.executeUpdate();

stmt.close();

} catch (SQLException e) {

LOGGER.warning("[Error] Task not added: " + e.getMessage());

}

}

// Добавление групп в словарь

public void parsingGroups() {

try {

// Make API request to retrieve group data

String groupUrl = GROUPS;

String groupResponse = makeApiRequest(groupUrl);

JSONArray groupArray = new JSONArray(groupResponse);

try (Statement statement = conn.createStatement()) {

for (int i = 0; i < groupArray.length(); i++) {

JSONObject group = groupArray.getJSONObject(i);

int groupId = group.getInt("oid");

String groupName = group.getString("name");

String query = "INSERT INTO groups (id\_group, name\_group) VALUES ('" + groupId + "', '" + groupName + "')";

statement.execute(query);

}

}

} catch (SQLException e) {

LOGGER.warning("[Error] Failed to execute SQL query:" + e.getMessage());

} catch (Exception e) {

LOGGER.warning("[Error] Failed to retrieve json data:" + e.getMessage());

}

}

// Добавление аудиторий в словарь

public void parsingAudience() {

try {

// Make API request to retrieve audience data

String audienceUrl = AUDIENCES;

String audienceResponse = makeApiRequest(audienceUrl);

JSONArray audienceArray = new JSONArray(audienceResponse);

try (Statement statement = conn.createStatement()) {

for (int i = 0; i < audienceArray.length(); i++) {

JSONObject audience = audienceArray.getJSONObject(i);

int audienceId = audience.getInt("oid");

String audienceName = "";

if (!audience.isNull("name")) {

audienceName = audience.getString("name");

}

String query = "INSERT INTO audience (id\_audience, name\_audience) VALUES ('" + audienceId + "', '" + audienceName + "')";

statement.execute(query);

}

}

} catch (SQLException e) {

LOGGER.warning("[Error] Failed to execute SQL query:" + e.getMessage());

} catch (Exception e) {

LOGGER.warning("[Error] Failed to retrieve json data:" + e.getMessage());

}

}

// Добавление преподавателей в словарь

public void parsingLecturers() {

try {

// Make API request to retrieve audience data

String lecturerUrl = LECTURERS;

String lecturerResponse = makeApiRequest(lecturerUrl);

JSONArray lecturerArray = new JSONArray(lecturerResponse);

try (Statement statement = conn.createStatement()) {

for (int i = 0; i < lecturerArray.length(); i++) {

JSONObject audience = lecturerArray.getJSONObject(i);

int lecturerId = audience.getInt("pid");

String fullName = audience.getString("fio");

String[] names = fullName.split(" ");

StringBuilder abbreviatedName = new StringBuilder();

// Добавляем фамилию

abbreviatedName.append(names[0]).append(" ");

// Добавляем инициалы имени и отчества

for (int n = 1; n < names.length; n++) {

abbreviatedName.append(names[n].charAt(0)).append(".");

}

String lecturerName = String.valueOf(abbreviatedName);

try {

String query = "INSERT INTO lecturers (id\_lecturer, fio) VALUES ('" + lecturerId + "', '" + lecturerName + "')";

statement.execute(query);

} catch (SQLException e) {

String query = "UPDATE lecturers SET fio = '"+lecturerName+"' WHERE id\_lecturer = '"+lecturerId+"';";

statement.executeUpdate(query);

}

}

}

} catch (SQLException e) {

LOGGER.warning("[Error] Failed to execute SQL query:" + e.getMessage());

} catch (Exception e) {

LOGGER.warning("[Error] Failed to lecturers json data:" + e.getMessage());

}

}

// Добавление типов дисцеплин в словарь

public void parsingTypeDisciplines(JSONArray schedule) {

try {

JSONArray seenIds = new JSONArray();

for (int i = 0; i < schedule.length(); i++) {

JSONObject row = schedule.getJSONObject(i);

String typeDiscipline = row.getJSONObject("content").getString("type\_disciplina");

if (!seenIds.toString().contains(typeDiscipline)) {

seenIds.put(typeDiscipline);

}

}

try (Statement statement = conn.createStatement()) {

for (int i = 0; i < seenIds.length(); i++) {

String typeDiscipline = seenIds.getString(i);

String query = "INSERT INTO type\_disciplines (name\_type) VALUES ('" + typeDiscipline + "')";

statement.execute(query);

}

}

} catch (SQLException e) {

LOGGER.warning("[Error] Failed to execute SQL query:" + e.getMessage());

} catch (Exception e) {

LOGGER.warning("[Error] Failed to parse type disciplines:" + e.getMessage());

}

}

// Добавление дисцеплин в словарь

public void parsingDisciplines(JSONArray schedule) {

try {

JSONArray seenIds = new JSONArray();

for (int i = 0; i < schedule.length(); i++) {

JSONObject row = schedule.getJSONObject(i);

String discipline = row.getJSONObject("content").getString("disciplina");

if (!seenIds.toString().contains(discipline)) {

seenIds.put(discipline);

}

}

try (Statement statement = conn.createStatement()) {

for (int i = 0; i < seenIds.length(); i++) {

String discipline = seenIds.getString(i);

String query = "INSERT INTO disciplines (name\_discipline) VALUES ('" + discipline + "')";

statement.execute(query);

}

}

} catch (SQLException e) {

LOGGER.warning("[Error] Failed to execute SQL query:" + e.getMessage());

} catch (Exception e) {

LOGGER.warning("[Error] Failed to parse disciplines:" + e.getMessage());

}

}

// Добавление расписания звонков для всех курсов в словарь

public void parsingCalls(JSONArray schedule) {

try {

JSONArray seenIds = new JSONArray();

for (int i = 0; i < schedule.length(); i++) {

JSONObject row = schedule.getJSONObject(i);

JSONObject arr = new JSONObject();

if (row.has("lesson")) {

Object lessonValue = row.get("lesson");

if (lessonValue instanceof String) {

arr.put("lesson", row.getString("lesson"));

} else if (lessonValue instanceof Integer) {

arr.put("lesson", String.valueOf(row.getInt("lesson")));

}

}

arr.put("lesson\_time", row.getString("time"));

if (!seenIds.toString().contains(arr.toString())) {

seenIds.put(arr);

}

}

try (Statement statement = conn.createStatement()) {

for (int i = 0; i < seenIds.length(); i++) {

JSONObject row = seenIds.getJSONObject(i);

String lesson = row.getString("lesson");

String lessonTime = row.getString("lesson\_time");

String query = "INSERT INTO calls (lesson, lesson\_time) VALUES ('" + lesson + "', '" + lessonTime + "')";

statement.execute(query);

}

}

} catch (SQLException e) {

LOGGER.warning("[Error] Failed to execute SQL query:" + e.getMessage());

} catch (Exception e) {

LOGGER.warning("[Error] Failed to parse calls:" + e.getMessage());

}

}

// Добавление расписания в основную таблицу

public void parsingSchedule(JSONArray schedule) {

try (PreparedStatement preparedStatement = conn.prepareStatement(

"INSERT INTO schedule (date, call\_id, group\_id, subgroup\_name, discipline\_id, type\_discipline\_id, audience\_id, lecturer\_id) " +

"VALUES (?, ?, ?, ?, ?, ?, ?, ?)")) {

DateTimeFormatter dateFormatter = DateTimeFormatter.ofPattern("dd.MM.yyyy");

for (int i = 0; i < schedule.length(); i++) {

JSONObject gru = schedule.getJSONObject(i);

List<String> record = new ArrayList<>();

// Date

String dateString = gru.optString("date");

LocalDate date = LocalDate.parse(dateString, dateFormatter);

// Lesson and Time

int lesson = gru.optInt("lesson");

String time = gru.optString("time");

String callId = getCallIdByLessonAndTime(lesson, time);

record.add(callId);

// Groups

String groupId = gru.optString("group");

record.add(groupId);

// Content

JSONObject content = gru.optJSONObject("content");

if (content == null) {

continue; // Skip this iteration if content is null

}

String subgroupName = content.optString("subgroupname");

record.add(subgroupName);

String discipline = content.optString("disciplina");

String disciplineId = getDisciplineIdByName(discipline);

record.add(disciplineId);

String typeDiscipline = content.optString("type\_disciplina");

String typeDisciplineId = getTypeDisciplineIdByName(typeDiscipline);

record.add(typeDisciplineId);

String audience = content.optString("aud");

String audienceId = getAudienceIdByName(audience);

record.add(audienceId);

String lecturer = content.optString("lecturer");

String lecturerId = getLecturerIdByName(lecturer);

record.add(lecturerId);

// Check for null values

if (record.contains(null)) {

// System.out.println(record);

continue; // Skip this iteration and continue with the next iteration

}

preparedStatement.setDate(1, java.sql.Date.valueOf(date));

for (int j = 0; j < record.size(); j++) {

preparedStatement.setInt(j + 2, Integer.parseInt(record.get(j))); // Corrected the index

}

preparedStatement.executeUpdate();

}

} catch (SQLException e) {

LOGGER.warning("[Error] Failed to execute SQL query: " + e.getMessage());

} catch (Exception e) {

LOGGER.warning("[Error] Failed to parse schedule: " + e.getMessage());

}

}

// Get zone

// Получения списка названий всех групп

public String[] retrieveNameGroupData() {

List<String> nameGroups = new ArrayList<>();

try {

Statement stmt = conn.createStatement();

ResultSet rs = stmt.executeQuery("SELECT name\_group FROM groups");

while (rs.next()) {

String nameGroup = rs.getString("name\_group");

nameGroups.add(nameGroup);

}

rs.close();

stmt.close();

} catch (SQLException e) {

LOGGER.warning("[Error] DB query retrieveNameGroupData: " + e.getMessage());

}

return nameGroups.toArray(new String[0]);

}

// Получения списка пользователей

public List<String> retrieveNamePersonData() {

List<String> namePersons = new ArrayList<>();

try {

Statement stmt = conn.createStatement();

ResultSet rs = stmt.executeQuery("SELECT person\_name FROM public.persons");

while (rs.next()) {

String namePerson = rs.getString("person\_name");

namePersons.add(namePerson);

}

rs.close();

stmt.close();

} catch (SQLException e) {

LOGGER.warning("[Error] DB query retrieveNamePersonData: " + e.getMessage());

}

return namePersons;

}

// Получения расписание на определённый день для определённого пользователя

public String getScheduleDay(LocalDate date, String group, int subgroup) {

StringBuilder schedule = new StringBuilder();

try {

PreparedStatement stmt = conn.prepareStatement(

"SELECT DISTINCT lesson, lesson\_time, name\_discipline, name\_type, name\_audience, fio " +

"FROM schedule, groups, calls, disciplines, type\_disciplines, audience, lecturers " +

"WHERE schedule.call\_id = calls.id\_call and schedule.discipline\_id = disciplines.id\_discipline " +

"and schedule.group\_id = groups.id\_group and schedule.type\_discipline\_id = type\_disciplines.id\_type\_discipline and " +

"schedule.audience\_id = audience.id\_audience and schedule.lecturer\_id = lecturers.id\_lecturer and " +

"groups.name\_group = ? and (schedule.subgroup\_name = ? or schedule.subgroup\_name = 0) and schedule.date = ?");

stmt.setString(1, group);

stmt.setInt(2, subgroup);

stmt.setDate(3, Date.valueOf(date));

ResultSet rs = stmt.executeQuery();

while (rs.next()) {

StringBuilder num = new StringBuilder();

String lesson = rs.getString("lesson");

String lessonTime = rs.getString("lesson\_time");

String nameDiscipline = rs.getString("name\_discipline");

String nameType = rs.getString("name\_type");

String nameAudience = rs.getString("name\_audience");

String fioLecturer = rs.getString("fio");

num.append(lesson).append(". ").append(lessonTime).append(" - ").append(nameDiscipline);

if (num.length() < 45){

num.append(" (").append(nameType).append(") ");

} else {

num.append("\n(").append(nameType).append(") ");

schedule.append(num);

num = new StringBuilder();

}

if (num.length() < 45){

num.append(nameAudience).append(" ");

} else {

num.append("\n").append(nameAudience).append(" ");

schedule.append(num);

num = new StringBuilder();

}

if (num.length() < 45){

num.append(fioLecturer).append("\n");

} else {

num.append("\n").append(fioLecturer).append("\n");

schedule.append(num);

}

schedule.append(num);

}

rs.close();

stmt.close();

} catch (SQLException e) {

LOGGER.warning("[Error] DB query getScheduleDay: " + e.getMessage());

}

return schedule.toString();

}

// Получения группы пользователя

public String getGroup(String login) {

String group = "";

try {

PreparedStatement stmt = conn.prepareStatement("SELECT name\_group FROM persons WHERE person\_name=?;");

stmt.setString(1, login);

ResultSet rs = stmt.executeQuery();

if (rs.next()) {

group = rs.getString("name\_group");

}

rs.close();

stmt.close();

} catch (SQLException e) {

LOGGER.warning("[Error] DB query getGroup: " + e.getMessage());

}

return group;

}

// Получения подгруппы пользователя

public int getSubgroup(String login) {

int subgroup = 0;

try {

PreparedStatement stmt = conn.prepareStatement("SELECT subgroup FROM public.persons WHERE person\_name=?;");

stmt.setString(1, login);

ResultSet rs = stmt.executeQuery();

if (rs.next()) {

subgroup = rs.getInt("subgroup");

}

rs.close();

stmt.close();

} catch (SQLException e) {

LOGGER.warning("[Error] DB query getSubgroup: " + e.getMessage());

}

return subgroup;

}

// Получения темы пользователя

public boolean getTheme(String login) {

boolean theme = true;

try {

PreparedStatement stmt = conn.prepareStatement("SELECT theme FROM public.persons WHERE person\_name=?;");

stmt.setString(1, login);

ResultSet rs = stmt.executeQuery();

if (rs.next()) {

theme = rs.getBoolean("theme");

}

rs.close();

stmt.close();

} catch (SQLException e) {

LOGGER.warning("[Error] DB query getTheme: " + e.getMessage());

}

return theme;

}

// Получения списка задач на день

public List<String> getTasks(String person\_name, LocalDate task\_date) {

List<String> tasks = new ArrayList<>();

try {

String query = "SELECT task FROM person\_task WHERE person\_name=? and task\_date=?;";

PreparedStatement stmt = conn.prepareStatement(query);

stmt.setString(1, person\_name);

stmt.setDate(2, Date.valueOf(task\_date));

ResultSet rs = stmt.executeQuery();

while (rs.next()) {

String task = rs.getString("task");

tasks.add(task);

}

rs.close();

stmt.close();

} catch (SQLException e) {

LOGGER.warning("[Error] DB query getTasks: " + e.getMessage());

}

return tasks;

}

// Получения статуса задачи

public Boolean getTaskStatus(String person\_name, LocalDate task\_date, String task) {

try {

String query = "SELECT task\_status FROM person\_task WHERE person\_name=? and task\_date=? and task=?;";

PreparedStatement stmt = conn.prepareStatement(query);

stmt.setString(1, person\_name);

stmt.setDate(2, Date.valueOf(task\_date));

stmt.setString(3, task);

ResultSet rs = stmt.executeQuery();

if (rs.next()) {

return rs.getBoolean(1);

}

rs.close();

stmt.close();

} catch (SQLException e) {

LOGGER.warning("[Error] DB query getTaskStatus: " + e.getMessage());

}

return false;

}

// Получения id расписания звонка для отдельного занятия по разным курсам

private String getCallIdByLessonAndTime(int lesson, String time) throws SQLException {

String query = "SELECT id\_call FROM calls WHERE lesson = ? AND lesson\_time = ?";

PreparedStatement preparedStatement = conn.prepareStatement(query);

preparedStatement.setInt(1, lesson);

preparedStatement.setString(2, time);

ResultSet resultSet = preparedStatement.executeQuery();

if (resultSet.next()) {

return resultSet.getString(1);

}

return null;

}

// Получения id дисцеплины по названию

private String getDisciplineIdByName(String name) throws SQLException {

String query = "SELECT id\_discipline FROM disciplines WHERE name\_discipline = ?";

PreparedStatement preparedStatement = conn.prepareStatement(query);

preparedStatement.setString(1, name);

ResultSet resultSet = preparedStatement.executeQuery();

if (resultSet.next()) {

return resultSet.getString(1);

}

return null;

}

// Получения id типа дисцеплины по названию

private String getTypeDisciplineIdByName(String name) throws SQLException {

String query = "SELECT id\_type\_discipline FROM type\_disciplines WHERE name\_type = ?";

PreparedStatement preparedStatement = conn.prepareStatement(query);

preparedStatement.setString(1, name);

ResultSet resultSet = preparedStatement.executeQuery();

if (resultSet.next()) {

return resultSet.getString(1);

}

return null;

}

// Получения id аудитории по названию

private String getAudienceIdByName(String name) throws SQLException {

String query = "SELECT id\_audience FROM audience WHERE name\_audience = ?";

PreparedStatement preparedStatement = conn.prepareStatement(query);

preparedStatement.setString(1, name);

ResultSet resultSet = preparedStatement.executeQuery();

if (resultSet.next()) {

return resultSet.getString(1);

}

return null;

}

// Получения id преподователя по фио

private String getLecturerIdByName(String name) throws SQLException {

String query = "SELECT id\_lecturer FROM lecturers WHERE fio = ?";

PreparedStatement preparedStatement = conn.prepareStatement(query);

preparedStatement.setString(1, name);

ResultSet resultSet = preparedStatement.executeQuery();

if (resultSet.next()) {

return resultSet.getString(1);

}

return null;

}

// update zone

// Метод обнавления пароля пользователя

public void updatePassword(String login, String newPassword) {

try {

PreparedStatement stmt = conn.prepareStatement("UPDATE persons SET person\_password = ? WHERE person\_name = ?;");

stmt.setString(1, newPassword);

stmt.setString(2, login);

stmt.executeUpdate();

stmt.close();

} catch (SQLException e) {

LOGGER.warning("[Error] DB updatePassword: " + e.getMessage());

}

}

// Метод обнавления пароля пользователя

public void updateTask(String login, LocalDate date, String task, boolean status) {

try {

PreparedStatement stmt = conn.prepareStatement("UPDATE person\_task SET task\_status = ? WHERE person\_name = ? and task\_date = ? and task = ?;");

stmt.setBoolean(1, status);

stmt.setString(2, login);

stmt.setDate(3, Date.valueOf(date));

stmt.setString(4, task);

stmt.executeUpdate();

stmt.close();

} catch (SQLException e) {

LOGGER.warning("[Error] DB updateTask: " + e.getMessage());

}

}

// Метод обнавления группы, подгруппы и темы пользователя

public void updateGroupAndSubGroupAndTheme(String login, String group, int subGroup, boolean theme) {

try {

PreparedStatement stmt = conn.prepareStatement("UPDATE persons SET name\_group = ?, subgroup = ?, theme = ? WHERE person\_name = ?;");

stmt.setString(1, group);

stmt.setInt(2, subGroup);

stmt.setBoolean(3, theme);

stmt.setString(4, login);

stmt.executeUpdate();

stmt.close();

} catch (SQLException e) {

LOGGER.warning("[Error] DB updateGroupAndSubGroupAndTheme: " + e.getMessage());

}

}

// Отчистка всех таблиц словарей

public void clearTables() {

try {

try (Statement statement = conn.createStatement()) {

statement.execute("TRUNCATE groups CASCADE");

statement.execute("TRUNCATE audience CASCADE");

statement.execute("TRUNCATE lecturers CASCADE");

statement.execute("TRUNCATE type\_disciplines CASCADE");

statement.execute("TRUNCATE disciplines CASCADE");

statement.execute("TRUNCATE calls CASCADE");

}

} catch (SQLException e) {

LOGGER.warning("[Error] Failed to clear tables:" + e.getMessage());

}

}

// Отчистка таблицы расписания

public void clearScheduleTable() {

try {

try (Statement statement = conn.createStatement()) {

statement.execute("TRUNCATE schedule");

}

} catch (SQLException e) {

LOGGER.warning("[Error] Failed to clear schedule table:" + e.getMessage());

}

}

// Pars data zone

// Обновление таблиц словарей

public void parsingDictionaries(){

clearTables();

JSONArray schedule = jsonSchedule();

parsingGroups();

parsingAudience();

parsingLecturers();

parsingDisciplines(schedule);

parsingTypeDisciplines(schedule);

parsingCalls(schedule);

}

// Обновление таблицы расписания

public void parsingOnlySchedule(){

clearScheduleTable();

JSONArray schedule = jsonSchedule();

parsingSchedule(schedule);

}

// Обнавление всех таблиц связанных с расисанием

public void parsingAll(){

clearTables();

clearScheduleTable();

JSONArray schedule = jsonSchedule();

parsingGroups();

parsingAudience();

parsingLecturers();

parsingDisciplines(schedule);

parsingTypeDisciplines(schedule);

parsingCalls(schedule);

parsingSchedule(schedule);

}

}

# Файлы resources.

## Файл logins.fxml.

<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.image.\*?>

<?import javafx.scene.text.\*?>

<?import javafx.scene.control.\*?>

<?import javafx.scene.layout.\*?>

<AnchorPane onMousePressed="#onMousePressed" onMouseDragged="#onMouseDragged" maxHeight="-Infinity"

maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="300.0" prefWidth="450.0"

styleClass="fon" stylesheets="@style.css" xmlns="http://javafx.com/javafx/8"

xmlns:fx="http://javafx.com/fxml/1" fx:controller="com.example.scheduleproject.loginsController">

<Text fill="#464646" layoutX="203.0" layoutY="76.0" strokeType="OUTSIDE" strokeWidth="0.0" styleClass="h1"

text="Вход" textAlignment="CENTER" textOrigin="CENTER"/>

<Text layoutX="163.0" layoutY="289.0" strokeType="OUTSIDE" strokeWidth="0.0"

text="Schedule RSVPU, Biryukov © 2023.">

<font>

<Font size="8.0"/>

</font>

</Text>

<Button onAction="#signUp" fx:id="signUp" layoutX="190.0" layoutY="180.0" mnemonicParsing="false" prefHeight="25.0" prefWidth="71.0"

text="Войти" textAlignment="CENTER"/>

<Button onAction="#registrations" fx:id="registrations" layoutX="321.0" layoutY="32.0" mnemonicParsing="false" prefHeight="27.0"

prefWidth="94.0" text="Регистрация"/>

<TextField fx:id="login" layoutX="143.0" layoutY="100.0" prefHeight="29.0" prefWidth="165.0" promptText="Логин"

styleClass="input"/>

<PasswordField fx:id="password" layoutX="143.0" layoutY="139.0" prefHeight="29.0" prefWidth="165.0"

promptText="Пароль" styleClass="input"/>

<Label fx:id="notification" alignment="CENTER" layoutX="104.0" layoutY="226.0" prefHeight="27.0" prefWidth="243.0"

styleClass="info" textAlignment="CENTER" textFill="#eb5643"/>

<ImageView fitHeight="49.0" fitWidth="43.0" layoutX="31.0" layoutY="21.0" pickOnBounds="true" preserveRatio="true"

styleClass="iconImage"/>

<Button onAction="#closeBut" fx:id="closeBut" layoutX="416.0" mnemonicParsing="false" styleClass="closeBut" text="❌"/>

</AnchorPane>

## Файл registrations.fxml.

<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.image.\*?>

<?import javafx.collections.\*?>

<?import javafx.scene.text.\*?>

<?import javafx.scene.control.\*?>

<?import java.lang.\*?>

<?import javafx.scene.layout.\*?>

<AnchorPane onMousePressed="#onMousePressed" onMouseDragged="#onMouseDragged" maxHeight="-Infinity" maxWidth="-Infinity"

minHeight="-Infinity" minWidth="-Infinity" prefHeight="300.0" prefWidth="450.0" styleClass="fon"

stylesheets="@style.css" xmlns="http://javafx.com/javafx/8" xmlns:fx="http://javafx.com/fxml/1"

fx:controller="com.example.scheduleproject.registrationsController">

<Text fill="#464646" layoutX="166.0" layoutY="66.0" strokeType="OUTSIDE" strokeWidth="0.0" styleClass="h1"

text="Регистрация" textAlignment="CENTER" textOrigin="CENTER"/>

<Button onAction="#signUp" fx:id="signUp" layoutX="361.0" layoutY="32.0" mnemonicParsing="false" prefHeight="27.0" prefWidth="54.0"

text="Вход"/>

<Label fx:id="notification" alignment="CENTER" layoutX="102.0" layoutY="249.0" prefHeight="27.0" prefWidth="243.0"

styleClass="info" textAlignment="CENTER" textFill="#eb5643"/>

<ComboBox fx:id="groups" layoutX="233.0" layoutY="97.0" prefHeight="31.0" prefWidth="165.0" promptText="Группа"

styleClass="input"/>

<PasswordField fx:id="password2" layoutX="48.0" layoutY="174.0" prefHeight="29.0" prefWidth="165.0"

promptText="Повторите пароль" styleClass="input"/>

<PasswordField fx:id="password1" layoutX="48.0" layoutY="136.0" prefHeight="29.0" prefWidth="165.0"

promptText="Пароль" styleClass="input"/>

<TextField fx:id="login" layoutX="48.0" layoutY="97.0" prefHeight="29.0" prefWidth="165.0" promptText="Логин"

styleClass="input"/>

<Button onAction="#registrations" fx:id="registrations" layoutX="156.0" layoutY="215.0" mnemonicParsing="false" prefHeight="27.0"

prefWidth="134.0" text="Зарегистрироваться"/>

<Text layoutX="163.0" layoutY="289.0" strokeType="OUTSIDE" strokeWidth="0.0"

text="Schedule RSVPU, Biryukov © 2023.">

<font>

<Font size="8.0"/>

</font>

</Text>

<ComboBox fx:id="subgroups" layoutX="233.0" layoutY="136.0" prefHeight="31.0" prefWidth="165.0"

promptText="Подгруппа" styleClass="input">

<items>

<FXCollections fx:factory="observableArrayList">

<String fx:value="Нет"/>

<String fx:value="1"/>

<String fx:value="2"/>

<String fx:value="3"/>

</FXCollections>

</items>

</ComboBox>

<Button onAction="#closeBut" fx:id="closeBut" layoutX="416.0" mnemonicParsing="false" styleClass="closeBut" text="❌"/>

<ImageView fitHeight="49.0" fitWidth="43.0" layoutX="30.0" layoutY="22.0" opacity="0.12" pickOnBounds="true"

preserveRatio="true">

<Image url="@logo.png"/>

</ImageView>

</AnchorPane>

## Файл main\_menu.fxml.

<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.collections.\*?>

<?import javafx.geometry.\*?>

<?import javafx.scene.control.\*?>

<?import java.lang.\*?>

<?import javafx.scene.layout.\*?>

<AnchorPane fx:id="rootPane" maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity"

onMouseDragged="#onMouseDragged" onMousePressed="#onMousePressed" prefHeight="400.0" prefWidth="535.0"

styleClass="fon" xmlns="http://javafx.com/javafx/8" xmlns:fx="http://javafx.com/fxml/1"

fx:controller="com.example.scheduleproject.main\_menuController">

<Button fx:id="pn" layoutX="36.0" layoutY="49.0" maxHeight="50.0" maxWidth="50.0" mnemonicParsing="false"

onAction="#pn" prefHeight="40.0" prefWidth="40.0" text="ПН"/>

<Button fx:id="vt" layoutX="36.0" layoutY="98.0" mnemonicParsing="false" onAction="#vt" prefHeight="40.0"

prefWidth="40.0" text="ВТ"/>

<Button fx:id="sr" layoutX="36.0" layoutY="147.0" mnemonicParsing="false" onAction="#sr" prefHeight="40.0"

prefWidth="40.0" text="СР"/>

<Button fx:id="ct" layoutX="36.0" layoutY="196.0" mnemonicParsing="false" onAction="#ct" prefHeight="40.0"

prefWidth="40.0" text="ЧТ"/>

<Button fx:id="pt" layoutX="36.0" layoutY="245.0" mnemonicParsing="false" onAction="#pt" prefHeight="40.0"

prefWidth="40.0" text="ПТ"/>

<Button fx:id="sb" layoutX="36.0" layoutY="294.0" mnemonicParsing="false" onAction="#sb" prefHeight="40.0"

prefWidth="40.0" text="СБ"/>

<Button fx:id="vs" layoutX="36.0" layoutY="343.0" mnemonicParsing="false" onAction="#vs" prefHeight="40.0"

prefWidth="40.0" text="ВС"/>

<Button fx:id="closeBut" layoutX="500.0" mnemonicParsing="false" onAction="#closeBut" styleClass="closeBut"

text="❌"/>

<Button fx:id="menuBut" layoutX="30.0" layoutY="-3.0" mnemonicParsing="false" onAction="#menuBut"

styleClass="menBut" text="≡">

<opaqueInsets>

<Insets right="2.0"/>

</opaqueInsets>

</Button>

<Label fx:id="schedulDay" layoutX="93.0" layoutY="74.0" prefHeight="162.0" prefWidth="403.0" styleClass="text1"/>

<Label layoutX="96.0" layoutY="237.0" styleClass="text1" text="Задачи:"/>

<Button fx:id="rightBut" layoutX="506.0" layoutY="55.0" mnemonicParsing="false" onAction="#rightBut" opacity="0.58"

prefHeight="322.0" prefWidth="25.0" text="&gt;"/>

<Button fx:id="leftBut" layoutX="5.0" layoutY="55.0" mnemonicParsing="false" onAction="#leftBut" opacity="0.58"

prefHeight="322.0" prefWidth="25.0" text="&lt;"/>

<Label fx:id="dateWeek" layoutX="93.0" layoutY="31.0" styleClass="text1"/>

<ScrollPane layoutX="96.0" layoutY="261.0" prefHeight="93.0" prefWidth="393.0" styleClass="scroll">

<content>

<VBox fx:id="taskContainer" prefHeight="93.0" prefWidth="391.0" styleClass="scroll"/>

</content>

</ScrollPane>

<Button fx:id="addTaskBut" layoutX="414.0" layoutY="364.0" mnemonicParsing="false" onAction="#addTaskBut"

prefHeight="25.0" prefWidth="75.0" text="Добавить "/>

<TextField fx:id="taskTextField" layoutX="96.0" layoutY="364.0" prefHeight="25.0" prefWidth="311.0"

promptText="Ваша задача" styleClass="input1"/>

<AnchorPane fx:id="blackout" layoutY="1.0" prefHeight="400.0" prefWidth="535.0" styleClass="blackout"/>

<AnchorPane fx:id="menu" layoutX="159.0" layoutY="11.0" prefHeight="377.0" prefWidth="200.0" styleClass="fonMenu">

<Label fx:id="login" layoutX="25.0" layoutY="10.0" styleClass="h1" text="Login">

<opaqueInsets>

<Insets/>

</opaqueInsets>

</Label>

<Label layoutX="25.0" layoutY="57.0" styleClass="text1" text="Группа"/>

<Label layoutX="25.0" layoutY="97.0" styleClass="text1" text="Подгруппа"/>

<Label layoutX="25.0" layoutY="137.0" styleClass="text1" text="Тёмная тема"/>

<ToggleButton fx:id="darkStyle" layoutX="120.0" layoutY="134.0" mnemonicParsing="false"

onAction="#darkStyle" prefHeight="25.0" prefWidth="53.0" styleClass="input1" text="Off"/>

<Button fx:id="closeMenuBut" layoutX="165.0" mnemonicParsing="false" onAction="#closeMenuBut"

styleClass="closeBut" text="❌"/>

<Button fx:id="OkMenuBut" layoutX="74.0" layoutY="338.0" mnemonicParsing="false" onAction="#OkMenuBut"

prefHeight="25.0" prefWidth="52.0" text="✓"/>

<PasswordField fx:id="newPassword" layoutX="21.0" layoutY="269.0" prefHeight="31.0" prefWidth="159.0"

promptText="Новый пароль" styleClass="input"/>

<PasswordField fx:id="oldPassword" layoutX="21.0" layoutY="230.0" prefHeight="31.0" prefWidth="159.0"

promptText="Старый пароль" styleClass="input"/>

<Label layoutX="51.0" layoutY="202.0" styleClass="text1" text="Смена пароля"/>

<ComboBox fx:id="group" layoutX="76.0" layoutY="54.0" prefHeight="25.0" prefWidth="98.0"

styleClass="input1"/>

<ComboBox fx:id="subgroup" layoutX="104.0" layoutY="94.0" prefHeight="25.0" prefWidth="69.0"

styleClass="input1">

<items>

<FXCollections fx:factory="observableArrayList">

<String fx:value="Нет"/>

<String fx:value="1"/>

<String fx:value="2"/>

<String fx:value="3"/>

</FXCollections>

</items>

</ComboBox>

<Label fx:id="notification" contentDisplay="TOP" layoutX="21.0" layoutY="310.0" prefHeight="17.0"

prefWidth="159.0" styleClass="info" textFill="WHITE"/>

<Button layoutX="25.0" layoutY="170.0" mnemonicParsing="false" onAction="#out" prefHeight="25.0"

prefWidth="148.0" text="Выйти"/>

</AnchorPane>

</AnchorPane>

## Файл style.css.

.fon{

-fx-background-color: linear-gradient(to top, #eeaeca, #94bbe9);

-fx-font-family: System;

}

.iconImage{

-fx-image: url('logo.png');

-fx-opacity: .12;

}

.h1{

-fx-font-size: 20px;

}

.button{

-fx-background-color: #fadcf0;

-fx-font-size: 12px;

-fx-opacity: 0.80;

-fx-background-radius: 10;

}

.closeBut{

-fx-background-color: none;

-fx-font-size: 15px;

}

.input{

-fx-background-color: #fadcf0;

-fx-background-radius: 7;

-fx-font-size: 15px;

}

.info{

-fx-font-size: 14px;

-fx-text-fill: #eb5643;

-fx-alignment: center;

}

.menBut{

-fx-background-color: none;

-fx-font-size: 25px;

}

.blackout{

-fx-background-color: #363636;

-fx-opacity: .42;

}

.schedul{

-fx-alignment: top-left;

}

.fonMenu{

-fx-background-color: linear-gradient(to top, #eeaeca, #94bbe9);

-fx-background-radius: 5;

-fx-opacity: 1;

}

.text1{

-fx-font-size: 15px;

}

.input1{

-fx-background-color: #fadcf0;

-fx-background-radius: 7;

-fx-font-size: 12px;

}

ScrollPane {

-fx-fit-to-width: true;

-fx-opacity: 0.80;

}

ScrollPane .scroll-bar {

-fx-background-color: #fadcf0;

}

.scroll{

-fx-background-color: linear-gradient(to top, #e4afce, #cfb2d5);

}

.check-box .box {

-fx-background-radius: 20;

-fx-background-color: #fadcf0;

}

.check-box:selected .box {

-fx-background-color: rgb(190, 255, 181);

}

.check-box .text {

-fx-font-size: 12px;

}

.check-box:selected .text {

-fx-strikethrough: true;

}

#rootPane .fon,

#rootPane .h1,

#rootPane .button,

#rootPane .closeBut,

#rootPane .input,

#rootPane .input.prompt-text,

#rootPane .menBut,

#rootPane .text1,

#rootPane .input1,

#rootPane .schedul,

#rootPane .check-box .text {

-fx-text-fill: #363636;

}

## Файл dark\_style.css.

.fon{

-fx-background-color: linear-gradient(to top, #8b8b8b, #486f49);

-fx-font-family: System;

}

.iconImage{

-fx-image: url('logo\_dark.png');

-fx-opacity: .12;

}

.h1{

-fx-font-size: 20px;

}

.button{

-fx-background-color: #376b407e;

-fx-font-size: 12px;

-fx-opacity: 0.80;

-fx-background-radius: 10;

}

.closeBut{

-fx-background-color: none;

-fx-font-size: 15px;

}

.input{

-fx-background-color: #55775e;

-fx-background-radius: 7;

-fx-font-size: 15px;

}

.info{

-fx-font-size: 14px;

-fx-text-fill: #eb5643;

-fx-alignment: center;

}

.menBut{

-fx-background-color: none;

-fx-font-size: 25px;

}

.blackout{

-fx-background-color: #363636;

-fx-opacity: .42;

}

.schedul{

-fx-alignment: top-left;

}

.fonMenu{

-fx-background-color: linear-gradient(to top, #8b8b8b, #486f49);

-fx-background-radius: 5;

}

.text1{

-fx-font-size: 15px;

-fx-textFill: #adabab;

}

.input1{

-fx-background-color: #718a77;

-fx-background-radius: 7;

-fx-textFill: #adabab;

-fx-font-size: 12px;

}

ScrollPane {

-fx-fit-to-width: true;

-fx-opacity: 0.80;

}

ScrollPane .scroll-bar {

-fx-background-color: #90ad97;

-fx-background-radius: 7;

}

.scroll{

-fx-background-color: linear-gradient(to top, #848884, #748174);

}

.check-box .box {

-fx-background-radius: 20;

-fx-background-color: #90ad97;

}

.check-box:selected .box {

-fx-background-color: rgb(190, 255, 181);

}

.check-box .text {

/\* -fx-fill: #dddddd; \*/

-fx-font-size: 12px;

}

.check-box:selected .text {

-fx-strikethrough: true;

}

#rootPane .fon,

#rootPane .h1,

#rootPane .button,

#rootPane .closeBut,

#rootPane .input,

#rootPane .input.prompt-text,

#rootPane .menBut,

#rootPane .text1,

#rootPane .input1,

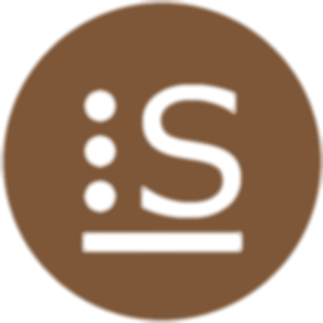
#rootPane .schedul,

#rootPane .check-box .text {

-fx-text-fill: #ffffff;

}

## Файл logo.png.



## Файл logo\_dark.png.

