Основной Класс

package ru.daru\_jo;

import javafx.application.Platform;

import javafx.scene.Parent;

import javafx.scene.Scene;

import javafx.scene.control.Alert;

import javafx.stage.Modality;

import javafx.stage.Stage;

import ru.daru\_jo.config.AbstractJavaFxApplicationSupport;

import ru.daru\_jo.config.ControllersConfiguration;

import ru.daru\_jo.dialogs.Dialogs;

import ru.daru\_jo.entity.FilterUser;

import ru.daru\_jo.entity.Room;

import ru.daru\_jo.entity.User;

import ru.daru\_jo.ui.BrowseRoomController;

import ru.daru\_jo.ui.EditRoomController;

import ru.daru\_jo.ui.EditUserController;

import ru.daru\_jo.ui.MainController;

import java.io.IOException;

public class Application extends AbstractJavaFxApplicationSupport {

private static Stage browseStage;

private static Stage editStage;

private Stage filterStage;

private static Stage browseRoomStage;

private static Stage editRoomStage;

private final String windowTitle = "Студенты";

private ControllersConfiguration.ViewHolder viewBrowse;

public void setFilter(FilterUser filterStrait) {

((MainController) viewBrowse.getController()).setFilter(filterStrait);

}

private ControllersConfiguration.ViewHolder browseRoom;

private ControllersConfiguration.ViewHolder editRoom;

private ControllersConfiguration.ViewHolder viewEdit;

private ControllersConfiguration.ViewHolder viewFilter;

public void closeFilterStage() {

filterStage.close();

}

public void openBrowseStage() {

browseStage.show();

}

public static Stage getBrowseStage() {

return browseStage;

}

@Override

public void start(Stage stagePrimary) {

ControllersConfiguration controllersConfiguration = new ControllersConfiguration();

try {

viewBrowse = controllersConfiguration.getMainView();

viewEdit = controllersConfiguration.getEditView();

browseRoom = controllersConfiguration.getBrowseRoomView();

editRoom = controllersConfiguration.getEditRoomView();

viewFilter = controllersConfiguration.getFilterView();

} catch (IOException e) {

throw new RuntimeException(e);

}

browseStage = stagePrimary;

initBrowseDialog();

openBrowseStage();

}

private void initEditDialog() {

editStage = new Stage();

editStage.initOwner(browseStage);

editStage.initModality(Modality.WINDOW\_MODAL);

editStage.setTitle(windowTitle);

editStage.setScene(new Scene(viewEdit.getView()));

editStage.setResizable(true);

editStage.centerOnScreen();

}

private void initRoomBrowseDialog() {

browseRoomStage = new Stage();

browseRoomStage.initOwner(browseStage);

browseRoomStage.initModality(Modality.WINDOW\_MODAL);

browseRoomStage.setTitle("Комнаты");

browseRoomStage.setScene(new Scene(browseRoom.getView()));

browseRoomStage.setResizable(true);

browseRoomStage.centerOnScreen();

}

private void initRoomEditDialog() {

editRoomStage = new Stage();

editRoomStage.initOwner(browseRoomStage);

editRoomStage.initModality(Modality.WINDOW\_MODAL);

editRoomStage.setTitle("Добавление комнаты");

editRoomStage.setScene(new Scene(editRoom.getView()));

editRoomStage.setResizable(true);

editRoomStage.centerOnScreen();

}

private Stage initDialog(String title, Parent parent) {

Stage stage = new Stage();

stage.initOwner(browseStage);

stage.initModality(Modality.WINDOW\_MODAL);

stage.setTitle(title);

stage.setScene(new Scene(parent));

stage.setResizable(true);

stage.centerOnScreen();

return stage;

}

private void initFilterDialog() {

filterStage = initDialog("Фильтр", viewFilter.getView());

}

private void initBrowseDialog() {

browseStage.setTitle(windowTitle);

browseStage.setScene(new Scene(viewBrowse.getView()));

browseStage.setResizable(true);

browseStage.centerOnScreen();

}

public void editShow() {

if (editStage == null) {

initEditDialog();

}

((EditUserController) viewEdit.getController()).reShow();

editStage.show();

}

public void roomBrowseShow() {

if (browseRoomStage == null) {

initRoomBrowseDialog();

}

((BrowseRoomController) browseRoom.getController()).reShow();

browseRoomStage.show();

}

public void roomEditShow() {

if (editRoomStage == null) {

initRoomEditDialog();

}

((EditRoomController) editRoom.getController()).reShow();

editRoomStage.show();

}

public void filterShow() {

if (filterStage == null) {

initFilterDialog();

}

// ((EditStraitController) viewEdit.getController()).reShow();

filterStage.show();

}

public static void showMes(String text) {

Platform.runLater(() -> Dialogs.showDialog(Alert.AlertType.ERROR, "Ошибка", "Ошибка", text));

}

private static Application INSTANCE;

@Override

public void init() throws Exception {

super.init();

INSTANCE = this;

}

public static Application getInstance() {

return INSTANCE;

}

public static void main(String[] args) {

launchApp(Application.class, args);

}

public void addUser(User user) {

((MainController) viewBrowse.getController()).addUser(user);

}

public void addRoom(Room room) {

((BrowseRoomController) browseRoom.getController()).addRoom(room);

}

}

Класс вспомогательный для запуска javafx

package ru.daru\_jo.config;

import javafx.application.Application;

public abstract class AbstractJavaFxApplicationSupport extends Application {

private static String[] savedArgs;

@Override

public void init() throws Exception {

}

@Override

public void stop() throws Exception {

super.stop();

}

protected static void launchApp(Class<? extends AbstractJavaFxApplicationSupport> appClass, String[] args) {

AbstractJavaFxApplicationSupport.savedArgs = args;

Application.launch(appClass, args);

}

}

Класс настрое JavaFX

package ru.daru\_jo.config;

import javafx.fxml.FXMLLoader;

import javafx.scene.Parent;

import java.io.IOException;

import java.io.InputStream;

public class ControllersConfiguration {

public ViewHolder getMainView() throws IOException {

return loadView("fxml/main.fxml");

}

public ViewHolder getEditView() throws IOException {

return loadView("fxml/edit\_user.fxml");

}

public ViewHolder getFilterView() throws IOException {

return loadView("fxml/filter-view.fxml");

}

public ViewHolder getBrowseRoomView() throws IOException{

return loadView("fxml/browse-room.fxml");

}

public ViewHolder getEditRoomView() throws IOException{

return loadView("fxml/edit\_room.fxml");

}

protected ViewHolder loadView(String url) throws IOException {

try (InputStream fxmlStream = getClass().getClassLoader().getResourceAsStream(url)) {

FXMLLoader loader = new FXMLLoader();

loader.load(fxmlStream);

return new ViewHolder(loader.getRoot(), loader.getController());

}

}

public class ViewHolder {

private Parent view;

private Object controller;

public ViewHolder(Parent view, Object controller) {

this.view = view;

this.controller = controller;

}

public Parent getView() {

return view;

}

public void setView(Parent view) {

this.view = view;

}

public Object getController() {

return controller;

}

public void setController(Object controller) {

this.controller = controller;

}

}

}

Класс для вывода сообщений

package ru.daru\_jo.dialogs;

import javafx.scene.control.Alert;

import ru.daru\_jo.Application;

public class Dialogs {

public static void showDialog(Alert.AlertType dialogType, String title, String type, String message) {

Alert alert = new Alert(dialogType);

alert.initOwner(Application.getBrowseStage());

alert.setTitle(title);

alert.setHeaderText(type);

alert.setContentText(message);

alert.showAndWait();

}

}

Класс Студентов

package ru.daru\_jo.entity;

import ru.daru\_jo.service.CodeService;

public class User {

private Integer id;

private String fio;

private Integer sex;

private String specialization;

private Integer roomId;

public User(Integer id, String fio, Integer sex, String specialization, Integer roomId) {

this.id = id;

this.fio = fio;

this.sex = sex;

this.specialization = specialization;

this.roomId = roomId;

}

public User(Integer id, String fio, Integer sex, String specialization) {

this.id = id;

this.fio = fio;

this.sex = sex;

this.specialization = specialization;

}

public void setRoomId(Integer roomId) {

this.roomId = roomId;

}

public Integer getId() {

return id;

}

public String getFio() {

return fio;

}

public Integer getSex() {

return sex;

}

public String getSexStr() {

return CodeService.getSex(sex == 1);

}

public String getSpecialization() {

return specialization;

}

public Integer getRoomId() {

return roomId;

}

}

Класс комнат

package ru.daru\_jo.entity;

import ru.daru\_jo.service.CodeService;

public class Room {

private Integer id;

private String name;

private Integer sex;

private String specialization;

private Integer maxStudent;

public Room(Integer id, String name, Integer sex, String specialization,Integer maxStudent) {

this.id = id;

this.name =name;

this.sex = sex;

this.maxStudent = maxStudent;

this.specialization = specialization;

}

public Integer getId() {

return id;

}

public Integer getSex() {

return sex;

}

public String getSpecialization() {

return specialization;

}

public Integer getMaxStudent() {

return maxStudent;

}

public String getName() {

return name;

}

public void setId(Integer id) {

this.id = id;

}

}

Класс комнат для браузера расширеный

package ru.daru\_jo.entity;

import ru.daru\_jo.service.CodeService;

public class RoomMax {

private Integer id;

private String name;

private Integer sex;

private String specialization;

private Integer maxStudent;

private Integer countStudent;

public RoomMax(Integer id, String name, Integer sex, String specialization, Integer countStudent, Integer maxStudent) {

this.id = id;

this.name = name;

this.sex = sex;

this.maxStudent = maxStudent;

this.countStudent = countStudent;

this.specialization = specialization;

}

public Integer getId() {

return id;

}

public Integer getSex() {

return sex;

}

public String getSexStr() {

return CodeService.getSex(sex == 1);

}

public String getSpecialization() {

return specialization;

}

public Integer getMaxStudent() {

return maxStudent;

}

public String getName() {

return name;

}

public Integer getCountStudent() {

return countStudent;

}

}

Класс для фильтрации студентов

package ru.daru\_jo.entity;

public class FilterUser {

private String fio = null;

private Boolean sex = null;

private String specialization =null;

private Integer roomId= null;

public FilterUser(String fio, Boolean sex, String specialization, Integer roomId) {

this.fio = fio;

this.sex = sex;

this.specialization = specialization;

this.roomId = roomId;

}

public FilterUser() {

}

public String getFio() {

return fio;

}

public Boolean getSex() {

return sex;

}

public String getSpecialization() {

return specialization;

}

public Integer getRoomId() {

return roomId;

}

}

Класс для фильтрации комнат

package ru.daru\_jo.entity;

public class FilterRoom {

private Boolean sex;

private String specialization;

public FilterRoom(Boolean sex, String specialization) {

this.sex = sex;

this.specialization = specialization;

}

public Boolean getSex() {

return sex;

}

public String getSpecialization() {

return specialization;

}

}

Класс для генерации ошибки

package ru.daru\_jo.exceptions;

public class UsernameNotFoundException extends RuntimeException{

public UsernameNotFoundException(String message){

super(message);

}

}

Экранные формы

Браузер комнат

package ru.daru\_jo.ui;

import javafx.application.Platform;

import javafx.collections.FXCollections;

import javafx.collections.ObservableList;

import javafx.event.ActionEvent;

import javafx.fxml.FXML;

import javafx.scene.control.TableColumn;

import javafx.scene.control.TableView;

import javafx.scene.control.cell.PropertyValueFactory;

import ru.daru\_jo.Application;

import ru.daru\_jo.entity.FilterUser;

import ru.daru\_jo.entity.User;

import ru.daru\_jo.service.UserService;

import java.util.List;

public class MainController {

private UserService userService;

@FXML

private TableView<User> table;

private ObservableList<User> data;

/\*\*

\* Инициализация контроллера от JavaFX.

\* Метод вызывается после того как FXML загрузчик произвел инъекции полей.

\* <p>

\* Обратите внимание, что имя метода <b>обязательно</b> должно быть "initialize",

\* в противном случае, метод не вызовется.

\* <p>

\* Также на этом этапе еще отсутствуют бины спринга

\* и для инициализации лучше использовать метод,

\* описанный аннотацией @PostConstruct,

\* который вызовется спрингом, после того, как им будут произведены все инъекции.

\* {@link MainController#init()}

\*/

@FXML

public void initialize() {

// Этап инициализации JavaFX

userService = UserService.getInstance();

init();

}

/\*\*

\* На этом этапе уже произведены все возможные инъекции.

\*/

@SuppressWarnings("unchecked")

public void init() {

List<User> userList = userService.getUserList();

data = FXCollections.observableArrayList(userList);

TableColumn<User, String> fioColumn = new TableColumn<>("ФИО");

fioColumn.setCellValueFactory(new PropertyValueFactory<>("fio"));

TableColumn<User, String> specColumn = new TableColumn<>("Специальность");

specColumn.setCellValueFactory(new PropertyValueFactory<>("specialization"));

TableColumn<User, String> sexColumn = new TableColumn<>("Пол");

sexColumn.setCellValueFactory(new PropertyValueFactory<>("sexStr"));

TableColumn<User, String> roomColumn = new TableColumn<>("Комната");

roomColumn.setCellValueFactory(new PropertyValueFactory<>("roomId"));

table.getColumns().setAll(fioColumn, specColumn, sexColumn, roomColumn);

// Данные таблицы

table.setItems(data);

}

/\*\*

\* Метод, вызываемый при нажатии на кнопку "Добавить".

\* Привязан к кнопке в FXML файле представления.

\*/

public void addUser(User user) {

data.add(user);

}

@FXML

public void edit() {

Platform.runLater(() ->

Application.getInstance().editShow()

);

}

public void setFilter(FilterUser filterStrait) {

data.clear();

data.setAll(userService.findAll(filterStrait));

}

public void viewFilter(ActionEvent actionEvent) {

Application.getInstance().filterShow();

}

public void viewRooms(ActionEvent actionEvent) {

Platform.runLater(() ->

Application.getInstance().roomBrowseShow()

);

}

}

Экранная форма фильтрации пользователей

package ru.daru\_jo.ui;

import javafx.application.Platform;

import javafx.event.ActionEvent;

import javafx.fxml.FXML;

import javafx.scene.control.Button;

import javafx.scene.control.ChoiceBox;

import javafx.scene.control.TextField;

import javafx.scene.input.KeyEvent;

import ru.daru\_jo.Application;

import ru.daru\_jo.entity.FilterUser;

import ru.daru\_jo.service.CodeService;

public class FilterController {

@FXML

public Button buttonOk;

@FXML

public Button buttonReset;

@FXML

public Button buttonCancel;

@FXML

public ChoiceBox<String> sex;

@FXML

public ChoiceBox<String> specialization;

@FXML

public TextField fio;

public TextField roomIdFx;

@FXML

public void saveFilter(ActionEvent actionEvent) {

Integer room = null;

try {

room = Integer.parseInt((roomIdFx.getText()));

} catch (NumberFormatException exception){

System.out.println(exception.getMessage());

}

Integer finalRoom = room;

Platform.runLater(() ->

Application.getInstance().setFilter(new FilterUser(

fio.getText(),

CodeService.isMan(sex.getValue()),

specialization.getValue(),

finalRoom)

));

Application.getInstance().closeFilterStage();

}

@FXML

public void cancel(ActionEvent actionEvent) {

Application.getInstance().closeFilterStage();

}

public void resetFilter(ActionEvent actionEvent) {

Platform.runLater(() ->

Application.getInstance().setFilter(null)

);

Application.getInstance().closeFilterStage();

}

@FXML

public void initialize() {

CodeService.getSex().forEach(s -> sex.getItems().add(s));

CodeService.getSpecializations().forEach(s -> specialization.getItems().add(s));

}

private String oldRoomId;

public void changeRoomIdFx() {

System.out.println("changeRoomIdFx");

try {

if(roomIdFx.getText() != null && roomIdFx.getText().isEmpty()){

return;

}

Integer.parseInt(roomIdFx.getText());

oldRoomId = roomIdFx.getText();

} catch (NumberFormatException e){

System.out.println(e.getMessage());

roomIdFx.setText(oldRoomId);

roomIdFx.end();

}

}

public void changeRoomIdFx(KeyEvent keyEvent) {

Platform.runLater(this::changeRoomIdFx);

}

}

Экранная форма редактирования Студентов

package ru.daru\_jo.ui;

import javafx.event.ActionEvent;

import javafx.fxml.FXML;

import javafx.scene.control.\*;

import javafx.scene.control.TextField;

import ru.daru\_jo.Application;

import ru.daru\_jo.entity.User;

import ru.daru\_jo.exceptions.UsernameNotFoundException;

import ru.daru\_jo.service.CodeService;

import ru.daru\_jo.service.UserService;

public class EditUserController {

@FXML

public ChoiceBox<String> specialization;

@FXML

public TextField fio;

@FXML

public ChoiceBox<String> sex;

@FXML

public Button generate;

private UserService userService;

@FXML

public void initialize() {

// Этап инициализации JavaFX

userService = UserService.getInstance();

CodeService.getSex().forEach(s -> sex.getItems().add(s));

CodeService.getSpecializations().forEach(s -> specialization.getItems().add(s));

}

public void reShow() {

fio.setText(null);

specialization.setValue(null);

sex.setValue(null);

}

@FXML

public void generate(ActionEvent actionEvent) {

if (checkAll()) {

return;

}

try {

User user = userService.addUser(

fio.getText(),

CodeService.isMan(sex.getValue()),

specialization.getValue()

);

Application.getInstance().addUser(user);

} catch (UsernameNotFoundException ex){

Application.showMes(ex.getMessage());

}

}

public boolean checkAll() {

StringBuilder stringBuilder = new StringBuilder();

if (checkFio()) {

addMes(stringBuilder, "ФИО студента не может состоять только из пробелов и точек, ФИО студента состоять из букв кириллицы пробелов и точек");

}

if (sex.getValue()== null || sex.getValue().isEmpty()) {

addMes(stringBuilder, "Пол должен быть заполнен");

}

if (specialization.getValue()== null || specialization.getValue().isEmpty()) {

addMes(stringBuilder, "Спецальность должен быть заполнена");

}

if (stringBuilder.length() !=0) {

Application.showMes(stringBuilder.toString());

return true;

}

return false;

}

private void addMes(StringBuilder stringBuilder, String text) {

if (stringBuilder.length() !=0) {

stringBuilder.append(", ");

}

stringBuilder.append(text);

}

public boolean checkFio() {

if (fio.getText() == null) {

return true;

}

String fioText = fio.getText().replace(" ", "").replace(".", "");

if (fioText.isEmpty()) {

return true;

}

return fioText.chars().anyMatch(c -> !Character.UnicodeBlock.of(c).equals(Character.UnicodeBlock.CYRILLIC));

}

}

Браузер комнат

package ru.daru\_jo.ui;

import javafx.application.Platform;

import javafx.collections.FXCollections;

import javafx.collections.ObservableList;

import javafx.fxml.FXML;

import javafx.scene.control.TableColumn;

import javafx.scene.control.TableView;

import javafx.scene.control.cell.PropertyValueFactory;

import ru.daru\_jo.Application;

import ru.daru\_jo.entity.Room;

import ru.daru\_jo.entity.RoomMax;

import ru.daru\_jo.service.RoomService;

import java.util.List;

public class BrowseRoomController {

private RoomService roomService;

@FXML

private TableView<RoomMax> table;

private ObservableList<RoomMax> data;

/\*\*

\* Инициализация контроллера от JavaFX.

\* Метод вызывается после того как FXML загрузчик произвел инъекции полей.

\* <p>

\* Обратите внимание, что имя метода <b>обязательно</b> должно быть "initialize",

\* в противном случае, метод не вызовется.

\* <p>

\* Также на этом этапе еще отсутствуют бины спринга

\* и для инициализации лучше использовать метод,

\* описанный аннотацией @PostConstruct,

\* который вызовется спрингом, после того, как им будут произведены все инъекции.

\* {@link BrowseRoomController#init()}

\*/

@FXML

public void initialize() {

// Этап инициализации JavaFX

roomService = RoomService.getInstance();

init();

}

/\*\*

\* На этом этапе уже произведены все возможные инъекции.

\*/

public void init() {

List<RoomMax> rooms = roomService.getRoomMaxList(null, null);

data = FXCollections.observableArrayList(rooms);

TableColumn<RoomMax, String> idColumn = new TableColumn<>("Номер");

idColumn.setCellValueFactory(new PropertyValueFactory<>("id"));

TableColumn<RoomMax, String> nameColumn = new TableColumn<>("Название");

nameColumn.setCellValueFactory(new PropertyValueFactory<>("name"));

TableColumn<RoomMax, String> specColumn = new TableColumn<>("Специальность");

specColumn.setCellValueFactory(new PropertyValueFactory<>("specialization"));

TableColumn<RoomMax, String> sexColumn = new TableColumn<>("Пол");

sexColumn.setCellValueFactory(new PropertyValueFactory<>("sexStr"));

TableColumn<RoomMax, String> countStudentColumn = new TableColumn<>("Заселено студентов");

countStudentColumn.setCellValueFactory(new PropertyValueFactory<>("countStudent"));

TableColumn<RoomMax, String> maxStudentColumn = new TableColumn<>("Заселено студентов");

maxStudentColumn.setCellValueFactory(new PropertyValueFactory<>("maxStudent"));

table.getColumns().setAll(idColumn, nameColumn, specColumn, sexColumn, countStudentColumn, maxStudentColumn);

// Данные таблицы

table.setItems(data);

}

public void reShow() {

data.clear();

init();

}

/\*\*

\* Метод, вызываемый при нажатии на кнопку "Добавить".

\* Привязан к кнопке в FXML файле представления.

\*/

public void addRoom(Room room) {

data.add(roomService.roomToRoomMax(room));

}

@FXML

public void edit() {

Platform.runLater(() ->

Application.getInstance().roomEditShow()

);

}

}

Редактирование комнат

package ru.daru\_jo.ui;

import javafx.application.Platform;

import javafx.event.ActionEvent;

import javafx.fxml.FXML;

import javafx.scene.control.Button;

import javafx.scene.control.ChoiceBox;

import javafx.scene.control.TextField;

import javafx.scene.input.KeyEvent;

import ru.daru\_jo.Application;

import ru.daru\_jo.entity.Room;

import ru.daru\_jo.service.CodeService;

import ru.daru\_jo.service.RoomService;

public class EditRoomController {

@FXML

public ChoiceBox<String> specialization;

@FXML

public TextField name;

@FXML

public ChoiceBox<String> sex;

@FXML

public Button generate;

public TextField maxStudent;

private RoomService roomService;

@FXML

public void initialize() {

// Этап инициализации JavaFX

roomService = RoomService.getInstance();

CodeService.getSex().forEach(s -> sex.getItems().add(s));

CodeService.getSpecializations().forEach(s -> specialization.getItems().add(s));

}

public void reShow() {

name.setText(null);

specialization.setValue(null);

sex.setValue(null);

}

@FXML

public void generate(ActionEvent actionEvent) {

if (checkAll()) {

return;

}

Room room = roomService.addRoom(

name.getText(),

CodeService.isMan(sex.getValue()),

specialization.getValue(),

Integer.parseInt(maxStudent.getText())

);

Application.getInstance().addRoom(room);

}

public boolean checkAll() {

StringBuilder stringBuilder = new StringBuilder();

if (checkName()) {

addMes(stringBuilder, "Название комнаты не может быть пустым");

}

if (sex.getValue()== null || sex.getValue().isEmpty()) {

addMes(stringBuilder, "Пол должен быть заполнен");

}

if (specialization.getValue()== null || specialization.getValue().isEmpty()) {

addMes(stringBuilder, "Спецальность должен быть заполнена");

}

if (maxStudent.getText()== null || maxStudent.getText().isEmpty()) {

addMes(stringBuilder, "Количество мест в комнате должно быть заполнено");

}

if (stringBuilder.length() != 0) {

Application.showMes(stringBuilder.toString());

return true;

}

return false;

}

private void addMes(StringBuilder stringBuilder, String text) {

if (stringBuilder.length() != 0) {

stringBuilder.append(", ");

}

stringBuilder.append(text);

}

public boolean checkName() {

if (name.getText() == null) {

return true;

}

return name.getText().isEmpty();

}

String oldMaxStudent;

public void changeMaxStudentFx() {

System.out.println("changeMaxStudentFx");

try {

if(maxStudent.getText() != null && maxStudent.getText().isEmpty()){

return;

}

Integer.parseInt(maxStudent.getText());

oldMaxStudent = maxStudent.getText();

} catch (NumberFormatException e){

System.out.println(e.getMessage());

maxStudent.setText(oldMaxStudent);

maxStudent.end();

}

}

public void changeMaxStudentFx(KeyEvent keyEvent) {

Platform.runLater(this::changeMaxStudentFx);

}

}

Класс настроек выпадаущих списков

package ru.daru\_jo.service;

import java.util.HashSet;

import java.util.Set;

public class CodeService {

private static Set<String> specializations;

private static Set<String> sex;

public static Set<String> getSpecializations() {

if (specializations == null){

specializations = new HashSet<>();

specializations.add("Разработка ПО");

specializations.add("Тестирование ПО");

}

return specializations;

}

public static Set<String> getSex() {

if (sex == null){

sex = new HashSet<>();

sex.add("Мужской");

sex.add("Женский");

}

return sex;

}

public static String getSex(boolean isMan) {

if (isMan) {

return "Мужской";

} else{

return "Женский";

}

}

public static Boolean isMan(String sex) {

if (sex == null){

return null;

}

return sex.equalsIgnoreCase("Мужской");

}

}

Класс Сервиса студентов

package ru.daru\_jo.service;

import ru.daru\_jo.entity.FilterUser;

import ru.daru\_jo.entity.User;

import ru.daru\_jo.exceptions.UsernameNotFoundException;

import ru.daru\_jo.repository.UserRepository;

import java.util.\*;

public class UserService {

private static UserService instance;

public static UserService getInstance(){

if(instance == null){

instance = new UserService();

}

return instance;

}

UserRoomService userRoomService;

public UserService() {

instance = this;

this.userRepository = new UserRepository();

this.userRoomService = UserRoomService.getInstance();

}

private final UserRepository userRepository;

public List<User> findAll(FilterUser filterStrait) {

return userRepository.findAll(filterStrait);

}

public void checkNull(String filed, String text) {

if (filed == null || filed.isEmpty()) {

throw new UsernameNotFoundException("Не заполнено поле " + text);

}

}

public User saveUser(User user) {

checkNull(user.getFio(), "ФИО");

if (user.getId() != null) {

if (userRepository.findByFioAndIdIsNot(user.getFio(), user.getId()).isPresent()) {

throw new UsernameNotFoundException("Уже есть пользователь с таким ФИО");

}

User saveUser = userRepository.findById(user.getId()).orElseThrow(() -> new UsernameNotFoundException("Студент с id " + user.getId() + " не найден"));

} else {

if (userRepository.findByFioIgnoreCase(user.getFio()).isPresent()) {

throw new UsernameNotFoundException("Уже есть студент с таким ФИО");

}

}

return userRepository.save(user);

}

public List<User> getUserList() {

return findAll(new FilterUser());

}

public User addUser(String fio, Boolean sex, String specialization) {

User user = new User(null,fio,sex? 1:0,specialization);

userRoomService.addRoomIdForUser(user);

if (user.getRoomId() == null){

throw new UsernameNotFoundException("Не удалось найти свободную комнату для пола и специальности " );

}

return saveUser(user);

}

public Integer countUser(int roomId) {

return userRepository.countUser(roomId);

}

}

Класс сервиса комнат

package ru.daru\_jo.service;

import ru.daru\_jo.entity.FilterRoom;

import ru.daru\_jo.entity.Room;

import ru.daru\_jo.entity.RoomMax;

import ru.daru\_jo.exceptions.UsernameNotFoundException;

import ru.daru\_jo.repository.RoomRepository;

import java.util.List;

public class RoomService {

private static RoomService instance;

public static RoomService getInstance() {

if (instance == null) {

instance = new RoomService();

}

return instance;

}

private final RoomRepository roomRepository;

private final UserRoomService userRoomService;

public RoomService() {

instance = this;

this.roomRepository = new RoomRepository();

this.userRoomService = UserRoomService.getInstance();

}

public List<Room> findAll(FilterRoom filterRoom) {

return roomRepository.findAll(filterRoom);

}

public void checkNull(String filed, String text) {

if (filed == null || filed.isEmpty()) {

throw new UsernameNotFoundException("Не заполнено поле " + text);

}

}

public Room saveRoom(Room room) {

return roomRepository.save(room);

}

public List<Room> getRoomList(Boolean sex, String specialization) {

return findAll(new FilterRoom(sex, specialization));

}

public Room addRoom(String name, Boolean sex, String specialization, Integer maxStudent) {

Room room = new Room(null, name, sex ? 1 : 0, specialization, maxStudent);

return saveRoom(room);

}

public RoomMax roomToRoomMax(Room room) {

return userRoomService.roomToRoomMax (room);

}

public List<RoomMax> getRoomMaxList(Boolean sex, String specialization) {

return userRoomService.getRoomMaxList(sex,specialization);

}

}

Отдельный класс для добавления студентов в комнаты

package ru.daru\_jo.service;

import ru.daru\_jo.entity.Room;

import ru.daru\_jo.entity.RoomMax;

import ru.daru\_jo.entity.User;

import java.util.List;

import java.util.stream.Collectors;

public class UserRoomService {

private static UserRoomService instance;

public static UserRoomService getInstance() {

if (instance == null) {

instance = new UserRoomService();

}

return instance;

}

RoomService roomService;

UserService userService;

public UserRoomService() {

instance = this;

roomService = RoomService.getInstance();

userService = UserService.getInstance();

}

public User addRoomIdForUser(User user) {

List<RoomMax> rooms = getRoomMaxList(user.getSex() == 1, user.getSpecialization());

for (RoomMax room : rooms) {

if (room.getMaxStudent() > room.getCountStudent()) {

user.setRoomId(room.getId());

return user;

}

}

return user;

}

public List<RoomMax> getRoomMaxList(Boolean sex, String specialization) {

return roomService.getRoomList(sex, specialization).stream().map(this::roomToRoomMax).collect(Collectors.toList());

}

public RoomMax roomToRoomMax(Room room) {

return new RoomMax(room.getId(), room.getName(), room.getSex(), room.getSpecialization(), userService.countUser(room.getId()), room.getMaxStudent());

}

}

Класс для получения подключений к бд

package ru.daru\_jo.repository;  
  
import java.sql.\*;  
import java.util.HashMap;  
import java.util.Map;  
  
public class ConnectionCenter {  
 private static ConnectionCenter *Instance*;  
 private final String URL = "jdbc:sqlite:db\\storage.db";  
 private final Map<Connection, Boolean> connectDBs = new HashMap<>();  
  
 private ConnectionCenter() {  
 try {  
 Class.*forName*("org.sqlite.JDBC");  
 } catch (ClassNotFoundException e) {  
 throw new RuntimeException(e);  
 }  
 }  
  
 public Connection getConnection() {  
 for (Map.Entry<Connection, Boolean> entry : connectDBs.entrySet()) {  
 if (!entry.getValue()) {  
 entry.setValue(true);  
 return entry.getKey();  
 }  
 }  
 try {  
 Connection connection = DriverManager.*getConnection*(URL);  
 connectDBs.put(connection, true);  
 return connection;  
 } catch (SQLException e) {  
 throw new RuntimeException(e);  
 }  
  
 }  
  
 public void disableConnect(Connection connection) {  
 connectDBs.put(connection, false);  
 }  
  
   
 public static ConnectionCenter getInstance() {  
 if (*Instance* == null) {  
 *Instance* = new ConnectionCenter();  
 }  
 return *Instance*;  
 }  
  
 public static void close() {  
 if (*Instance* == null) {  
 return;  
 }  
 try {  
 for (Map.Entry<Connection, Boolean> entry : *Instance*.connectDBs.entrySet()) {  
  
 entry.getKey().close();  
  
 }  
 } catch (SQLException e) {  
 throw new RuntimeException(e);  
 }  
 }  
  
  
  
}

Работа с БД студенты

package ru.daru\_jo.repository;

import ru.daru\_jo.entity.FilterUser;

import ru.daru\_jo.entity.User;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.util.ArrayList;

import java.util.List;

import java.util.Optional;

public class UserRepository {

public Optional<User> findByFioIgnoreCase(String fio) {

Connection connection = ConnectionCenter.getInstance().getConnection();

User user = null;

try (PreparedStatement prepInsert = connection.prepareStatement("SELECT id,fio, sex, specialization, room\_id FROM usersStorage WHERE fio = ? ")) {

prepInsert.setString(1, fio);

ResultSet resultSet = prepInsert.executeQuery();

while (resultSet.next()) {

user = new User(resultSet.getInt("id"),

resultSet.getString("fio"),

resultSet.getInt("sex"),

resultSet.getString("specialization"),

resultSet.getInt("room\_id")

);

// return Optional.ofNullable(user);

}

} catch (SQLException e) {

return null;

} finally {

ConnectionCenter.getInstance().disableConnect(connection);

}

return Optional.ofNullable(user);

}

public int findLast() {

Connection connection = ConnectionCenter.getInstance().getConnection();

int id = 0;

try (PreparedStatement prepInsert = connection.prepareStatement("SELECT id FROM usersStorage ORDER BY id DESC LIMIT 1;")) {

ResultSet resultSet = prepInsert.executeQuery();

while (resultSet.next()) {

id = resultSet.getInt("id");

}

return id;

} catch (SQLException e) {

return 0;

} finally {

ConnectionCenter.getInstance().disableConnect(connection);

}

}

public Optional<User> findByFioAndIdIsNot(String fio, Integer id) {

Connection connection = ConnectionCenter.getInstance().getConnection();

User user = null;

try (PreparedStatement prepInsert = connection.prepareStatement("SELECT id,fio, sex, specialization, room\_id FROM usersStorage WHERE fio = ? and id != ? ")) {

prepInsert.setString(1, fio);

prepInsert.setInt(2, id);

ResultSet resultSet = prepInsert.executeQuery();

while (resultSet.next()) {

user = new User(resultSet.getInt("id"),

resultSet.getString("fio"),

resultSet.getInt("sex"),

resultSet.getString("specialization"),

resultSet.getInt("room\_id")

);

// return Optional.ofNullable(user);

}

} catch (SQLException e) {

return Optional.empty();

} finally {

ConnectionCenter.getInstance().disableConnect(connection);

}

return Optional.ofNullable(user);

}

public Optional<User> findById(Integer id) {

Connection connection = ConnectionCenter.getInstance().getConnection();

User user = null;

try (PreparedStatement prepInsert = connection.prepareStatement("SELECT id,fio, sex, specialization, room\_id FROM usersStorage WHERE id = ? COLLATE NOCASE")) {

prepInsert.setInt(1, id);

ResultSet resultSet = prepInsert.executeQuery();

while (resultSet.next()) {

user = new User(resultSet.getInt("id"),

resultSet.getString("fio"),

resultSet.getInt("sex"),

resultSet.getString("specialization"),

resultSet.getInt("room\_id")

);

}

} catch (SQLException e) {

return Optional.empty();

} finally {

ConnectionCenter.getInstance().disableConnect(connection);

}

return Optional.ofNullable(user);

}

public User save(User user) {

try {

if (user.getId() == null) {

addUser(user);

} else {

updateUser(user);

}

return user;

} catch (

SQLException e) {

throw new RuntimeException(e);

}

}

public void addUser(User user) throws SQLException {

int id = findLast() + 1;

Connection connection = ConnectionCenter.getInstance().getConnection();

try (PreparedStatement prepInsert = connection.prepareStatement("INSERT INTO usersStorage (fio, sex, specialization, room\_id, id) VALUES (?,?,?,?,?)")) {

prepInsert.setString(1, user.getFio());

prepInsert.setInt(2, user.getSex());

prepInsert.setString(3, user.getSpecialization());

prepInsert.setInt(4, user.getRoomId());

prepInsert.setInt(5, id);

prepInsert.execute();

} finally {

ConnectionCenter.getInstance().disableConnect(connection);

}

}

public void updateUser(User user) throws SQLException {

Connection connection = ConnectionCenter.getInstance().getConnection();

try (PreparedStatement prepInsert = connection.prepareStatement("UPDATE usersStorage SET fio = ?, sex = ?, specialization = ?,room\_id = ? where usersStorage.id = ? ")) {

prepInsert.setString(1, user.getFio());

prepInsert.setInt(2, user.getSex());

prepInsert.setString(3, user.getSpecialization());

prepInsert.setInt(4, user.getRoomId());

prepInsert.setInt(5, user.getId());

prepInsert.execute();

} finally {

ConnectionCenter.getInstance().disableConnect(connection);

}

}

public List<User> findAll(FilterUser filterUser) {

List<User> users = new ArrayList<>();

StringBuilder command;

boolean first = true;

command = new StringBuilder();

if (filterUser.getFio() != null && !filterUser.getFio().isEmpty()) {

command.append(" fio = ? ");

first = false;

}

if (filterUser.getRoomId() != null) {

command.append(first ? "" : "AND ").append("room\_id = ? ");

first = false;

}

if (filterUser.getSex() != null) {

command.append(first ? "" : "AND ").append("sex = ? ");

first = false;

}

if (filterUser.getSpecialization() != null) {

command.append(first ? "" : "AND ").append("specialization = ? ");

first = false;

}

if (first) {

command.insert(0, "SELECT id,fio, sex, specialization, room\_id FROM usersStorage ");

} else {

command.insert(0, "SELECT id,fio, sex, specialization, room\_id FROM usersStorage WHERE ");

}

// command.append("where usersStorage.id = ? ");

// command.append(" COLLATE NOCASE ");

Connection connection = ConnectionCenter.getInstance().getConnection();

try (PreparedStatement prepSelect = connection.prepareStatement(command.toString())) {

int count = 1;

if (filterUser.getFio() != null && !filterUser.getFio().isEmpty()) {

prepSelect.setString(count++, filterUser.getFio());

}

if (filterUser.getRoomId() != null) {

prepSelect.setInt(count++, filterUser.getRoomId());

}

if (filterUser.getSex() != null) {

prepSelect.setInt(count++, filterUser.getSex() ? 1 : 0);

}

if (filterUser.getSpecialization() != null) {

prepSelect.setString(count, filterUser.getSpecialization());

}

ResultSet resultSet = prepSelect.executeQuery();

while (resultSet.next()) {

User user = new User(resultSet.getInt("id"),

resultSet.getString("fio"),

resultSet.getInt("sex"),

resultSet.getString("specialization"),

resultSet.getInt("room\_id")

);

users.add(user);

}

} catch (SQLException e) {

throw new RuntimeException(e);

} finally {

ConnectionCenter.getInstance().disableConnect(connection);

}

return users;

}

public Integer countUser(int roomId) {

return findAll (new FilterUser(null,null,null,roomId)).size();

}

}

Работа с БД комнаты

package ru.daru\_jo.repository;

import ru.daru\_jo.entity.FilterRoom;

import ru.daru\_jo.entity.Room;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.util.ArrayList;

import java.util.List;

public class RoomRepository {

public List<Room> findAll(FilterRoom filterRoom) {

List<Room> rooms = new ArrayList<>();

StringBuilder command;

boolean first = true;

command = new StringBuilder();

if (filterRoom.getSex() != null) {

command.append("sex = ? ");

first = false;

}

if (filterRoom.getSpecialization() != null) {

command.append(first ? "" : "AND ").append("specialization = ? ");

first = false;

}

if (first) {

command.insert(0, "SELECT id,name, sex, specialization, max\_student FROM roomsStorage ");

} else {

command.insert(0, "SELECT id,name, sex, specialization, max\_student FROM roomsStorage WHERE ");

}

// command.append("where roomsStorage.id = ? ");

Connection connection = ConnectionCenter.getInstance().getConnection();

try (PreparedStatement prepSelect = connection.prepareStatement(command.toString())) {

int count = 1;

if (filterRoom.getSex() != null) {

prepSelect.setInt(count++, filterRoom.getSex() ? 1 : 0);

}

if (filterRoom.getSpecialization() != null) {

prepSelect.setString(count, filterRoom.getSpecialization());

}

ResultSet resultSet = prepSelect.executeQuery();

while (resultSet.next()) {

Room room = new Room(resultSet.getInt("id"),

resultSet.getString("name"),

resultSet.getInt("sex"),

resultSet.getString("specialization"),

resultSet.getInt("max\_student")

);

rooms.add(room);

}

} catch (SQLException e) {

throw new RuntimeException(e);

} finally {

ConnectionCenter.getInstance().disableConnect(connection);

}

return rooms;

}

public Room save(Room room) {

try {

if (room.getId() == null) {

room = addRoom(room);

}

return room;

} catch (

SQLException e) {

throw new RuntimeException(e);

}

}

public Room addRoom(Room room) throws SQLException {

int id = findLast() + 1;

Connection connection = ConnectionCenter.getInstance().getConnection();

try (PreparedStatement prepInsert = connection.prepareStatement("INSERT INTO roomsStorage (sex, specialization, max\_student, name, id) VALUES (?,?,?,?,?)")) {

prepInsert.setInt(1, room.getSex());

prepInsert.setString(2, room.getSpecialization());

prepInsert.setInt(3, room.getMaxStudent());

prepInsert.setString(4, room.getName());

prepInsert.setInt(5, id);

prepInsert.execute();

room.setId(id);

return room;

} finally {

ConnectionCenter.getInstance().disableConnect(connection);

}

}

public int findLast() {

Connection connection = ConnectionCenter.getInstance().getConnection();

int id = 0;

try (PreparedStatement prepInsert = connection.prepareStatement("SELECT id FROM roomsStorage ORDER BY id DESC LIMIT 1;")) {

ResultSet resultSet = prepInsert.executeQuery();

while (resultSet.next()) {

id = resultSet.getInt("id");

}

return id;

} catch (SQLException e) {

return 0;

} finally {

ConnectionCenter.getInstance().disableConnect(connection);

}

}

}

Ресурсы

Экранная форма браузера Студентов

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

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

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

<AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="450.0"

prefWidth="800.0" xmlns="http://javafx.com/javafx/16" xmlns:fx="http://javafx.com/fxml/1"

fx:controller="ru.daru\_jo.ui.MainController">

<children>

<TableView fx:id="table" editable="true" prefHeight="200.0" prefWidth="405.0" tableMenuButtonVisible="true"

AnchorPane.bottomAnchor="50.0" AnchorPane.leftAnchor="0.0" AnchorPane.rightAnchor="0.0"

AnchorPane.topAnchor="0.0">

<columnResizePolicy>

<TableView fx:constant="CONSTRAINED\_RESIZE\_POLICY"/>

</columnResizePolicy>

</TableView>

<HBox alignment="CENTER" layoutX="21.0" layoutY="207.0" prefHeight="50.0" prefWidth="300.0"

AnchorPane.bottomAnchor="0.0" AnchorPane.leftAnchor="10.0" AnchorPane.rightAnchor="10.0">

<children>

<Button minWidth="-Infinity" mnemonicParsing="false" onAction="#edit" text="Добавить"/>

<Button minWidth="-Infinity" mnemonicParsing="false" onAction="#viewRooms" text="Комнаты"/>

<Button minWidth="-Infinity" mnemonicParsing="false" onAction="#viewFilter" text="Фильтр"/>

</children>

</HBox>

</children>

</AnchorPane>

Экранная форма фильтрации студентов

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

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

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

<AnchorPane prefHeight="400.0" prefWidth="600.0" xmlns="http://javafx.com/javafx/16" xmlns:fx="http://javafx.com/fxml/1" fx:controller="ru.daru\_jo.ui.FilterController">

<children>

<Label layoutX="20.0" layoutY="24.0" text="ФИО заказчика" />

<TextField fx:id="fio" layoutX="118.0" layoutY="20.0" promptText="ФИО" />

<Label layoutX="20.0" layoutY="62.0" text="Пол" />

<ChoiceBox fx:id="sex" layoutX="118.0" layoutY="58.0" prefHeight="25.0" prefWidth="149.0" />

<Label layoutX="21.0" layoutY="103.0" text="Специальность" />

<ChoiceBox fx:id="specialization" layoutX="118.0" layoutY="99.0" prefHeight="25.0" prefWidth="149.0" />

<Label layoutX="19.0" layoutY="149.0" text="Комната" />

<TextField fx:id="roomIdFx" layoutX="118.0" layoutY="145.0" onKeyReleased="#changeRoomIdFx" promptText="Номер комнаты" />

<Button fx:id="buttonOk" layoutX="104.0" layoutY="278.0" mnemonicParsing="false" onAction="#saveFilter" text="Применить" />

<Button fx:id="buttonCancel" layoutX="183.0" layoutY="278.0" mnemonicParsing="false" onAction="#cancel" text="Отмена" />

<Button fx:id="buttonReset" layoutX="241.0" layoutY="278.0" mnemonicParsing="false" onAction="#resetFilter" text="Сбросить" />

</children>

</AnchorPane>

Экранная форма редатирования студентов

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

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

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

<AnchorPane prefHeight="400.0" prefWidth="600.0" xmlns="http://javafx.com/javafx/16" xmlns:fx="http://javafx.com/fxml/1" fx:controller="ru.daru\_jo.ui.EditUserController">

<children>

<Label layoutX="10.0" layoutY="20.0" text="ФИО студента" />

<TextField fx:id="fio" layoutX="100.0" layoutY="20.0" promptText="ФИО" />

<Label layoutX="20.0" layoutY="74.0" text="Пол" />

<ChoiceBox fx:id="sex" layoutX="104.0" layoutY="70.0" prefHeight="25.0" prefWidth="149.0" />

<Label layoutX="20.0" layoutY="111.0" text="Специальность" />

<ChoiceBox fx:id="specialization" layoutX="104.0" layoutY="111.0" prefHeight="25.0" prefWidth="149.0" />

<Button fx:id="generate" layoutX="66.0" layoutY="218.0" mnemonicParsing="false" onAction="#generate" text="Создать" />

</children>

</AnchorPane>

Экранная форма браузер комнат

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

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

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

<AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="450.0"

prefWidth="800.0" xmlns="http://javafx.com/javafx/16" xmlns:fx="http://javafx.com/fxml/1"

fx:controller="ru.daru\_jo.ui.BrowseRoomController">

<children>

<TableView fx:id="table" editable="true" prefHeight="200.0" prefWidth="405.0" tableMenuButtonVisible="true"

AnchorPane.bottomAnchor="50.0" AnchorPane.leftAnchor="0.0" AnchorPane.rightAnchor="0.0"

AnchorPane.topAnchor="0.0">

<columnResizePolicy>

<TableView fx:constant="CONSTRAINED\_RESIZE\_POLICY"/>

</columnResizePolicy>

</TableView>

<HBox alignment="CENTER" layoutX="21.0" layoutY="207.0" prefHeight="50.0" prefWidth="300.0"

AnchorPane.bottomAnchor="0.0" AnchorPane.leftAnchor="10.0" AnchorPane.rightAnchor="10.0">

<children>

<Button minWidth="-Infinity" mnemonicParsing="false" onAction="#edit" text="Добавить"/>

</children>

</HBox>

</children>

</AnchorPane>

Экранная форма редактировани комнат

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

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

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

<AnchorPane prefHeight="400.0" prefWidth="600.0" xmlns="http://javafx.com/javafx/16" xmlns:fx="http://javafx.com/fxml/1" fx:controller="ru.daru\_jo.ui.EditRoomController">

<children>

<Label layoutX="19.0" layoutY="29.0" text="Комната" />

<TextField fx:id="name" layoutX="145.0" layoutY="25.0" />

<Label layoutX="20.0" layoutY="74.0" text="Пол" />

<ChoiceBox fx:id="sex" layoutX="145.0" layoutY="70.0" prefHeight="25.0" prefWidth="149.0" />

<Label layoutX="20.0" layoutY="111.0" text="Специальность" />

<ChoiceBox fx:id="specialization" layoutX="145.0" layoutY="107.0" prefHeight="25.0" prefWidth="149.0" />

<Label layoutX="14.0" layoutY="150.0" text="Колличество учащихся" />

<TextField fx:id="maxStudent" layoutX="145.0" layoutY="146.0" onKeyReleased="#changeMaxStudentFx" />

<Button fx:id="generate" layoutX="66.0" layoutY="218.0" mnemonicParsing="false" onAction="#generate" text="Создать" />

</children>

</AnchorPane>

Скриты создания таблиц

Студентов

CREATE TABLE usersStorage (

id INTEGER,

fio TEXT,

sex INTEGER,

specialization TEXT,

room\_id INTEGER

);

CREATE UNIQUE INDEX usersStorage\_id\_IDX ON usersStorage (id);

CREATE INDEX usersStorage\_room\_id\_IDX ON usersStorage (room\_id);

Комнат

CREATE TABLE roomsStorage (

id INTEGER,

name TEXT,

sex INTEGER,

specialization TEXT,

max\_student INTEGER

);

CREATE INDEX roomsStorage\_id\_IDX ON roomsStorage (id);

CREATE INDEX roomsStorage\_specialization\_IDX ON roomsStorage (specialization,sex);