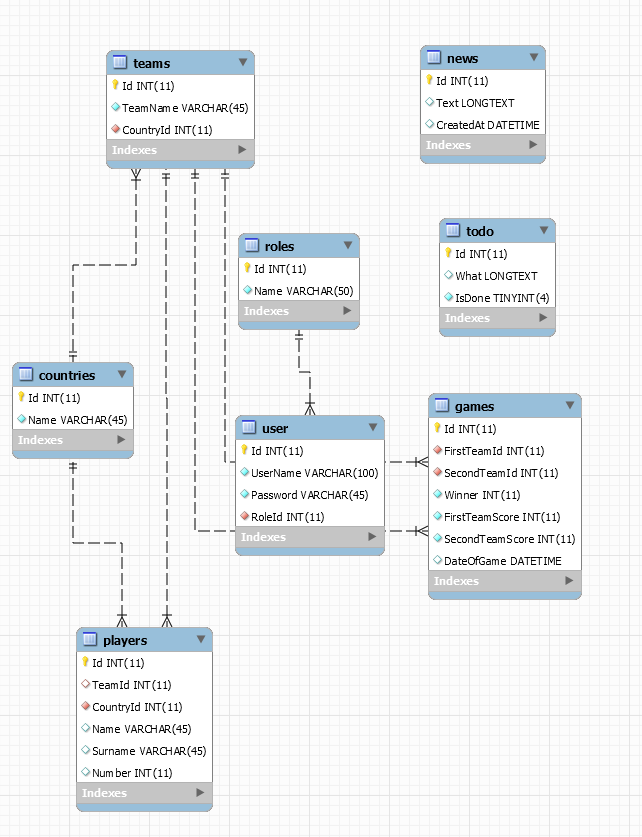
Цели контрольной работы

Целью является разработка АРМ для пользователей национальной баскетбольной лиги.

Лабораторная работа 1: База данных



Выше представлена схема базы данных. В качестве провайдера базы данных выбран MySQL.

Лабораторные работы 2, 3: Разделение на пользователей, авторизация

Для приложения представлены 2 роли пользователей: администратор и обычный пользователь. Администратор имеет права на редактирование базы данных, создание todo списка и добавление новостей. Пользователь может информацию только просматривать. Todo лист пользователю не доступен.

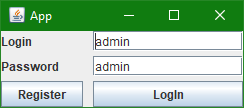


Рисунок 1 - Форма авторизации

Нажав на Register – при успешной регистрации появится сообщение о том, что все хорошо.

После входа как админ видим окно админа:

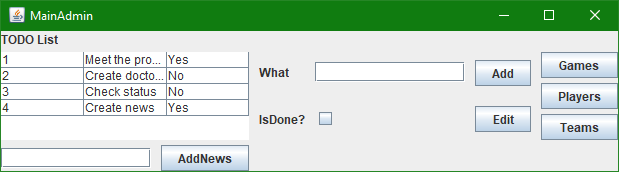


Рисунок 2

Пользователь видит следующее:

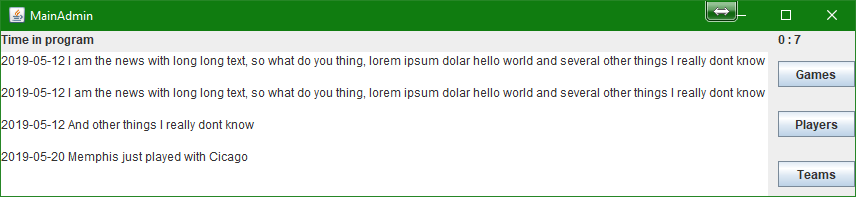


Рисунок 3

Листинг формы авторизации

package basic;  
  
import Model.Repositories.UserRepository;  
import Model.Tables.User;  
import services.UserService;  
import services.WindowService;  
  
import javax.swing.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.awt.event.WindowEvent;  
import java.sql.SQLException;  
import java.util.List;  
  
*/\*\*  
 \* Created by super on 4/28/2019.  
 \*/*public class Login{  
 private JButton login;  
 public JPanel panel;  
 private JTextField loginInput;  
 private JTextField passwordInput;  
 private JButton register;  
  
 private UserRepository userRepository;  
 private WindowService windowService;  
  
 public Login() throws SQLException {  
 userRepository = new UserRepository();  
 windowService = new WindowService();  
  
 login.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 String login = loginInput.getText();  
 String pass = passwordInput.getText();  
  
 try {  
 User user = userRepository.Get(login, pass).get(0);  
  
 if (user != null) {  
 UserService.*currentUser* = user;  
 GoToMainWindow(user);  
 }  
 } catch (SQLException e1) {  
 e1.printStackTrace();  
 }  
  
 //compare with database  
 }  
 });  
  
 register.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 String login = loginInput.getText();  
 String pass = passwordInput.getText();  
  
 try {  
 userRepository.Save(login, pass);  
  
 JOptionPane.*showMessageDialog*(null, "Registered", "InfoBox:", JOptionPane.*INFORMATION\_MESSAGE*);  
 } catch (SQLException e1) {  
 e1.printStackTrace();  
 }  
 }  
 });  
 }  
  
 private void GoToMainWindow(User user) throws SQLException {  
 if (user.getRoleId() == 1) {  
 WindowService.*GoToWindow*(panel, new MainAdmin().panel);  
 }  
 else {  
 WindowService.*GoToWindow*(panel, new MainUser().panel);  
 }  
  
 }  
}

Лабораторная работа 4: Учет времени проведенной в программе.

Функция видна только пользователю, отображается сверху главного окна.



Рисунок 4

Листинг:

WindowService.*SubscribeOnAppTimer*((m, s) -> {  
 timer.setText(m + " : " + s);  
});

public class WindowService {  
 public interface OnAppTimer {  
 void OnTimer(int mins, int secs);  
 }  
  
 private static List<OnAppTimer> *subscribers*;  
 private static long *timeInApp* = 0;  
  
 static {  
 *subscribers* = new ArrayList<OnAppTimer>();  
  
 new Timer(1000, e -> {  
 *timeInApp*++;  
  
 int mins = (int)(*timeInApp* / 60);  
 int secs = (int)(*timeInApp* - mins \* 60);  
  
 for (OnAppTimer sub: *subscribers*) {  
 sub.OnTimer(mins, secs);  
 }  
 }).start();  
 }  
  
 public static void SubscribeOnAppTimer(OnAppTimer listener) {  
 *subscribers*.add(listener);  
 }  
}

Лабораторные работы 6, 7, 8, 9, 10: CRUD базы данных, зажита форм.

Защита форм в приложении представлена в виде сообщений под полями ввода в случае некорректного ввода.

Для примера приведена форма Игр:

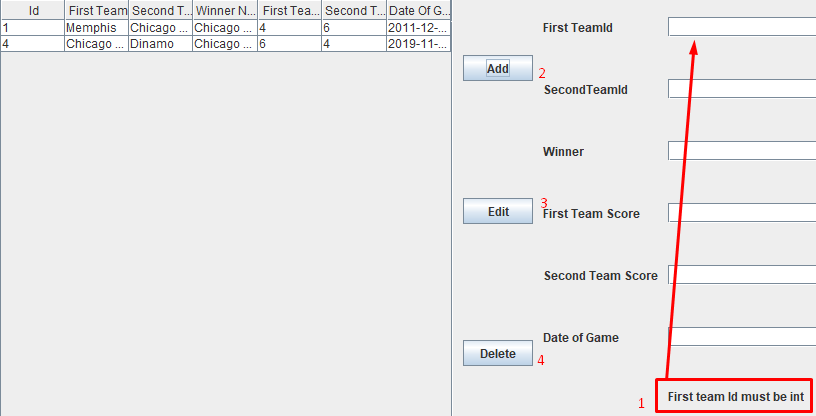


Рисунок 5

1. Сообщение об ошибке;
2. Кнопка добавления;
3. Кнопка изменения;
4. Кнопка удаления;

Нажатие на кнопку добавления/изменения провоцирует соответствующие проверки. В случае некорректного ввода внизу будет отображена ближайшая ошибка. После исправления нужно нажать на кнопку добавления/изменения повторно. В случае успеха запись будет добавлена в БД и на таблицу слева.

Для редактирования и удаления, первоначально нужно выбрать запись в таблице, так, чтобы выделилась синим. После поля справа автоматически заполняться и можно редактировать их.

Остальные формы: Игроки и Команды

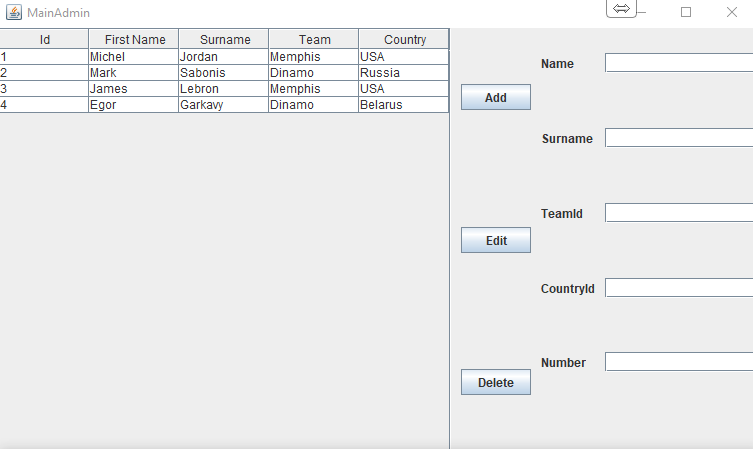


Рисунок 6

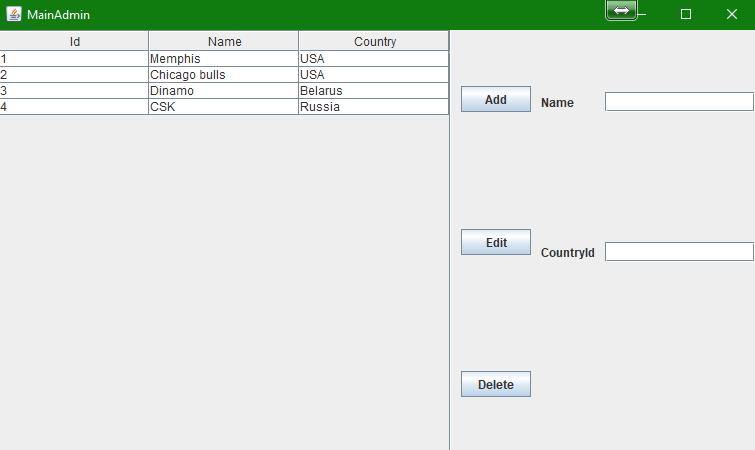


Рисунок 7

Листинг формы игр

package basic;  
  
import Model.Repositories.CountryRepository;  
import Model.Repositories.GamesRepository;  
import Model.Repositories.PlayersRepository;  
import Model.Repositories.TeamRepository;  
import Model.Tables.Country;  
import Model.Tables.Game;  
import Model.Tables.Player;  
import Model.Tables.Team;  
import services.UserService;  
  
import javax.swing.\*;  
import javax.swing.table.DefaultTableModel;  
import java.awt.event.MouseAdapter;  
import java.awt.event.MouseEvent;  
import java.sql.Date;  
import java.sql.SQLException;  
import java.text.DateFormat;  
import java.text.ParseException;  
import java.text.SimpleDateFormat;  
import java.util.List;  
  
*/\*\*  
 \* Created by super on 5/19/2019.  
 \*/*public class Games {  
 public JPanel panel;  
 private JTable table;  
 private JButton addBtn;  
 private JButton editBtn;  
 private JButton deleteBtn;  
 private JTextField firstIdField;  
 private JTextField secondIdField;  
 private JTextField winnerField;  
 private JTextField fScoreField;  
 private JTextField sScoreField;  
 private JLabel errorLabel;  
 private JPanel editPanel;  
 private JTextField dateFIeld;  
  
 private List<Game> games;  
 private PlayersRepository playersRepository;  
 private CountryRepository countryRepository;  
 private TeamRepository teamRepository;  
 private GamesRepository gamesRepository;  
  
 private DefaultTableModel model;  
  
 public Games() throws SQLException {  
 playersRepository = new PlayersRepository();  
 countryRepository = new CountryRepository();  
 teamRepository = new TeamRepository();  
 gamesRepository = new GamesRepository();  
  
 if (UserService.*currentUser*.getRoleId() == 2) {  
 editPanel.setVisible(false);  
 }  
  
 Object[] columns = {"Id", "First Team", "Second Team", "Winner Name", "First Team Score", "Second Team Score", "Date Of Game"};  
 model = new DefaultTableModel();  
 model.setColumnIdentifiers(columns);  
 table.setModel(model);  
  
 FillTable();  
  
 addBtn.addActionListener(e -> {  
 try {  
 Add();  
 } catch (SQLException e1) {  
 e1.printStackTrace();  
 } catch (ParseException e1) {  
 e1.printStackTrace();  
 }  
 });  
 deleteBtn.addActionListener(e -> {  
 try {  
 Delete();  
 } catch (SQLException e1) {  
 e1.printStackTrace();  
 }  
 });  
 table.addMouseListener(new MouseAdapter() {  
 @Override  
 public void mouseClicked(MouseEvent e) {  
 super.mouseClicked(e);  
  
 FillEditField();  
 }  
 });  
 editBtn.addActionListener(e -> {  
 try {  
 Edit();  
 } catch (SQLException e1) {  
 e1.printStackTrace();  
 }  
 });  
 }  
  
 private void Add() throws SQLException, ParseException {  
 String fId = firstIdField.getText();  
 String sId = secondIdField.getText();  
 String winnerId = winnerField.getText();  
 String fScore = fScoreField.getText();  
 String sScore = sScoreField.getText();  
 String date = dateFIeld.getText();  
  
 String error = ValidateGame(fId, sId, winnerId, fScore, sScore, date);  
  
 if (error.length() > 0) {  
 errorLabel.setText(error);  
 return;  
 }  
  
 Game game = new Game(Integer.*parseInt*(fId), Integer.*parseInt*(sId), Integer.*parseInt*(winnerId), new SimpleDateFormat("dd/MM/yyyy").parse(date), Integer.*parseInt*(fScore), Integer.*parseInt*(sScore));  
  
 gamesRepository.Save(game);  
  
 FillTable();  
 }  
  
 private void Delete() throws SQLException {  
 int i = table.getSelectedRow();  
  
 if (i >= 0) {  
 Game game = GetGameFromTable();  
  
 gamesRepository.Delete(game.getId());  
 }  
  
 FillTable();  
 }  
  
 private String ValidateGame(String firstId, String secondId, String winner, String fScore, String sScore, String date) throws SQLException {  
 String error = "";  
  
 if (firstId == null || secondId == null || winner == null || fScore == null || sScore == null || date == null) {  
 error = "All fields are required";  
  
 return error;  
 }  
  
 if (!TryParseInt(firstId)) {  
 error = "First team Id must be int";  
 return error;  
 }  
  
 if (!TryParseInt(secondId)) {  
 error = "Second team Id must be int";  
 return error;  
 }  
  
 if (!TryParseInt(winner)) {  
 error = "winner must be int";  
 return error;  
 }  
  
 if (TryParseInt(winner)) {  
 int winnerId = Integer.*parseInt*(winner);  
  
 if (winnerId != 1 && winnerId != 2) {  
 error = "winner must be in range of 1 or 2";  
 }  
 }  
  
 if (!TryParseInt(fScore)) {  
 error = "first Score must be int";  
 return error;  
 }  
  
 if (!TryParseInt(sScore)) {  
 error = "second Score be int";  
 return error;  
 }  
  
 if (!TryParseDate(date)) {  
 error = "date must be date";  
  
 return error;  
 }  
  
 List<Team> teams = teamRepository.Get();  
  
 int fId = Integer.*parseInt*(firstId);  
 int sId = Integer.*parseInt*(secondId);  
  
 if (!teams.stream().anyMatch(x -> x.getId() == fId)) {  
 error = "Choose correct first team Id";  
 return error;  
 }  
 if (!teams.stream().anyMatch(x -> x.getId() == sId)) {  
 error = "Choose correct second team Id";  
 return error;  
 }  
  
 return error;  
 }  
  
 private void Edit() throws SQLException {  
 Game game = GetGameFromTable();  
  
 FillGameFromEditFields(game);  
  
 gamesRepository.Update(game);  
  
 FillTable();  
 }  
  
 private void FillGameFromEditFields(Game game) {  
 game.setFirstTeamId(Integer.*parseInt*(firstIdField.getText()));  
 game.setSecondTeamId(Integer.*parseInt*(secondIdField.getText()));  
 game.setWinnerId(Integer.*parseInt*(winnerField.getText()));  
 game.setFirstTeamScore(Integer.*parseInt*(fScoreField.getText()));  
 game.setSecondTeamScore(Integer.*parseInt*(sScoreField.getText()));  
 try {  
 game.setDateOfGame(new SimpleDateFormat("dd/MM/yyyy").parse(dateFIeld.getText()));  
 } catch (ParseException e) {  
 e.printStackTrace();  
 }  
 }  
  
 private Game GetGameFromTable() {  
 int i = table.getSelectedRow();  
  
 int gameId = Integer.*parseInt*(model.getValueAt(i, 0).toString());  
  
 Game game = games.stream().filter(x -> x.getId() == gameId).findFirst().get();  
  
 return game;  
 }  
  
 private void FillEditField() {  
 Game game = GetGameFromTable();  
  
 FillEditField(game);  
 }  
  
 private void FillEditField(Game game) {  
 firstIdField.setText(Integer.*toString*(game.getFirstTeamId()));  
 secondIdField.setText(Integer.*toString*(game.getSecondTeamId()));  
 sScoreField.setText(Integer.*toString*(game.getSecondTeamScore()));  
 fScoreField.setText(Integer.*toString*(game.getFirstTeamScore()));  
 winnerField.setText(Integer.*toString*(game.getWinnerId()));  
 dateFIeld.setText(game.getDateOfGame().toString());  
 }  
  
 private boolean TryParseInt(String val) {  
 try {  
 Integer.*parseInt*(val);  
 return true;  
 } catch (NumberFormatException e) {  
 return false;  
 }  
 }  
  
 private boolean TryParseDate(String val) {  
 try {  
 new SimpleDateFormat("dd/MM/yyyy").parse(val);  
 return true;  
 } catch (NumberFormatException e) {  
 return false;  
 } catch (ParseException e) {  
 e.printStackTrace();  
 return false;  
 }  
 }  
  
 public void FillTable() throws SQLException {  
 model.setRowCount(0);  
  
 games = gamesRepository.Get();  
 FillTable(games);  
 }  
  
 public void FillTable(List<Game> games) {  
 for(Game game : games) {  
 Object[] row = new Object[7];  
  
 row[0] = game.getId();  
 row[1] = game.getFirstTeamName();  
 row[2] = game.getSecondTeamName();  
 row[3] = game.getWinnerName();  
 row[4] = game.getFirstTeamScore();  
 row[5] = game.getSecondTeamScore();  
 row[6] = game.getDateOfGame();  
  
 model.addRow(row);  
 }  
 }  
}

Лабораторная работа 11 Новости

Администратор может добавлять новости (Рисунок 2, слева внизу). Пользователь видит эти новости у себя на форме (Рисунок 3)

Листинг добавления новостей:

addNewsButton.addActionListener(e -> {  
 try {  
 newsRepository.Save(newsField.getText());  
 JOptionPane.*showMessageDialog*(null, "News added", "InfoBox:", JOptionPane.*INFORMATION\_MESSAGE*);  
  
 } catch (SQLException e1) {  
 e1.printStackTrace();  
 JOptionPane.*showMessageDialog*(null, "Something gone wrong", "InfoBox:", JOptionPane.*INFORMATION\_MESSAGE*);  
 }  
});

Лабораторная работа 12 Список дел

Todo лист создан только для админа. Todo может быть как выполнением и невыполненным (IsDone).

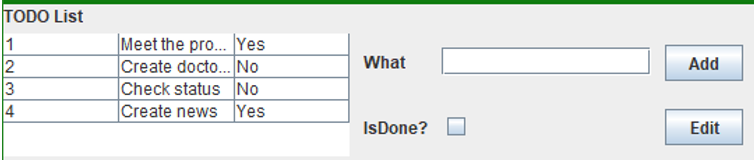


Рисунок 8

Листинг todo листа

private void Add() throws SQLException {  
 String what = whatField.getText();  
 boolean isDone = isDoneCheckBox.isSelected();  
  
 ToDo todo = new ToDo(what, isDone);  
  
 toDoRepository.Save(todo);  
  
 FillTable();  
}  
  
private void Edit() throws SQLException {  
 ToDo todo = GetToDoFromTable();  
  
 FillToDoFromEditFields(todo);  
  
 toDoRepository.Update(todo);  
  
 FillTable();  
}  
  
private void FillToDoFromEditFields(ToDo todo) {  
 todo.setDone(isDoneCheckBox.isSelected());  
 todo.setWhat(whatField.getText());  
}  
  
private ToDo GetToDoFromTable() {  
 int i = table.getSelectedRow();  
  
 int todoId = Integer.*parseInt*(model.getValueAt(i, 0).toString());  
  
 ToDo toDo = todos.stream().filter(x -> x.getId() == todoId).findFirst().get();  
  
 return toDo;  
}  
  
private void FillEditField() {  
 ToDo todo = GetToDoFromTable();  
  
 FillEditField(todo);  
}  
  
private void FillEditField(ToDo todo) {  
 whatField.setText(todo.getWhat());  
 isDoneCheckBox.setSelected(todo.isDone());  
}  
  
public void FillTable() throws SQLException {  
 model.setRowCount(0);  
  
 todos = toDoRepository.Get();  
 FillTable(todos);  
}  
  
public void FillTable(List<ToDo> todos) {  
 for(ToDo todo : todos) {  
 Object[] row = new Object[3];  
  
 row[0] = todo.getId();  
 row[1] = todo.getWhat();  
 row[2] = todo.isDone() ? "Yes" : "No";  
  
 model.addRow(row);  
 }  
}