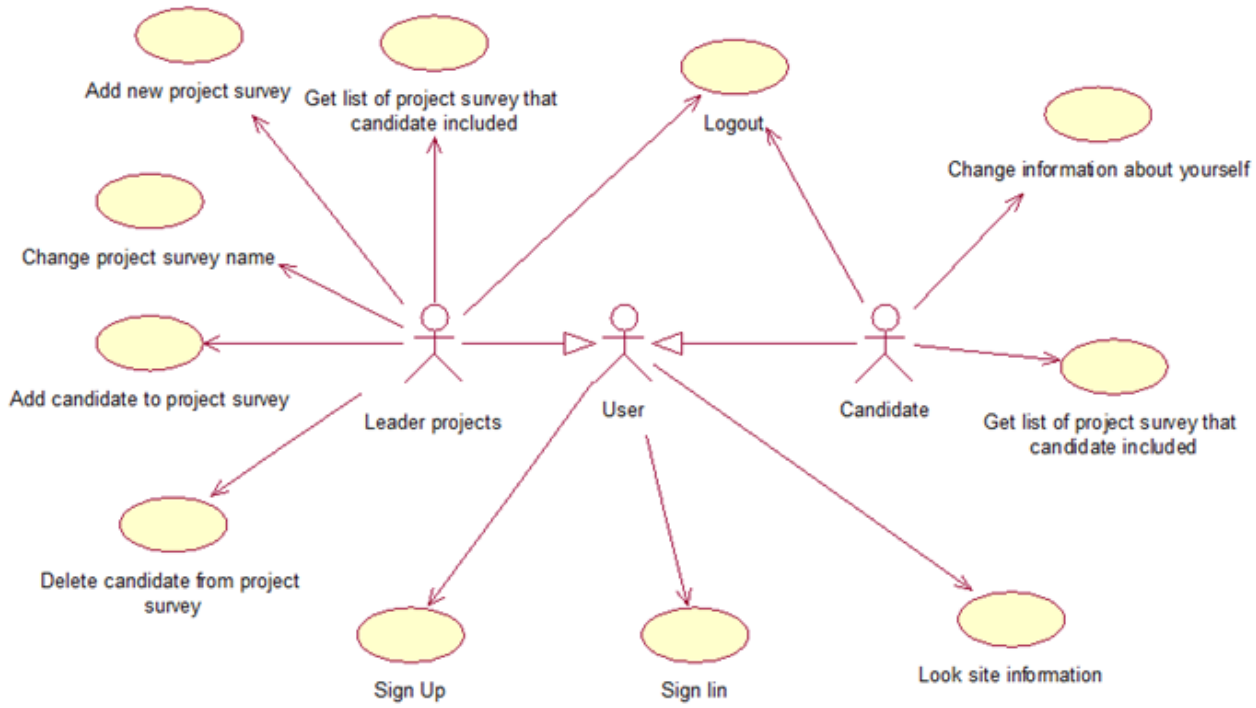
ДОДАТОК А

Графічний матеріал атестаційної роботи

ГЮИК.506160.011

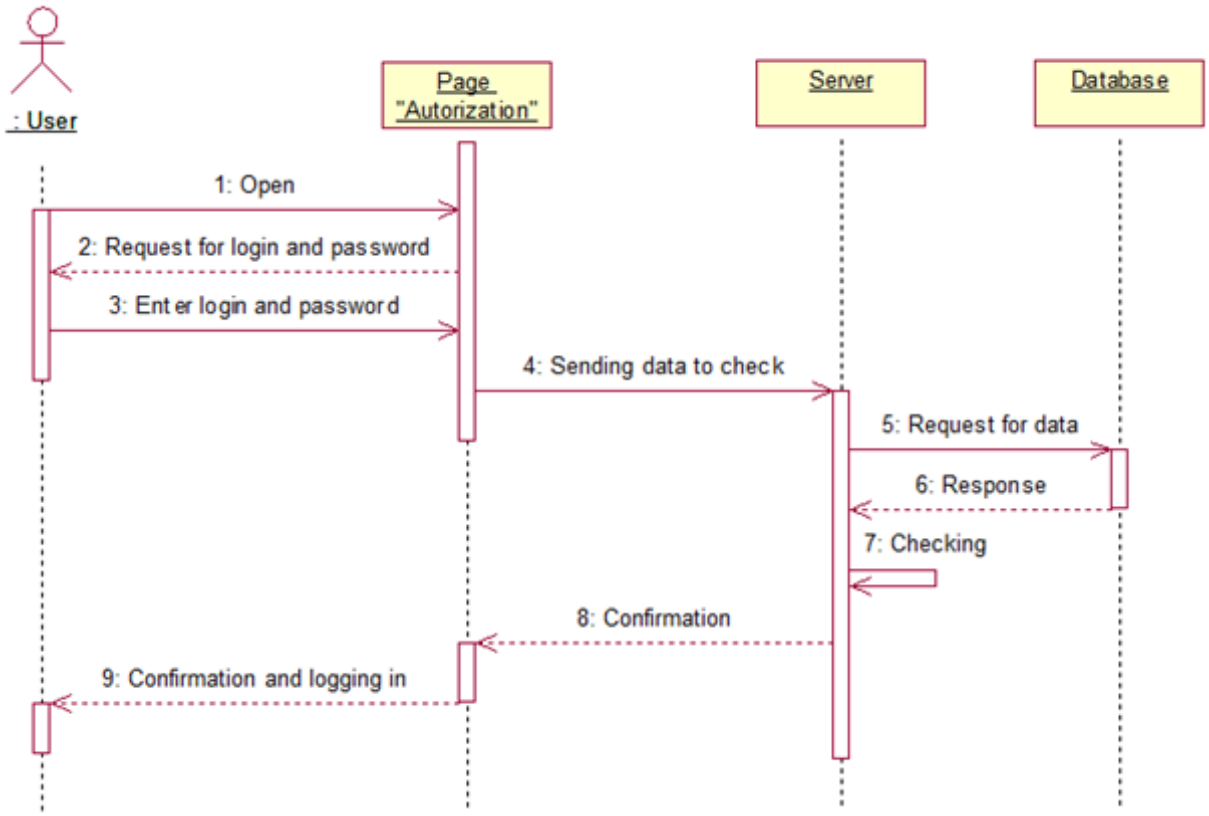
(позначення документу)

ДІАГРАМА ВАРІАНТІВ ВИКОРИСТАННЯ



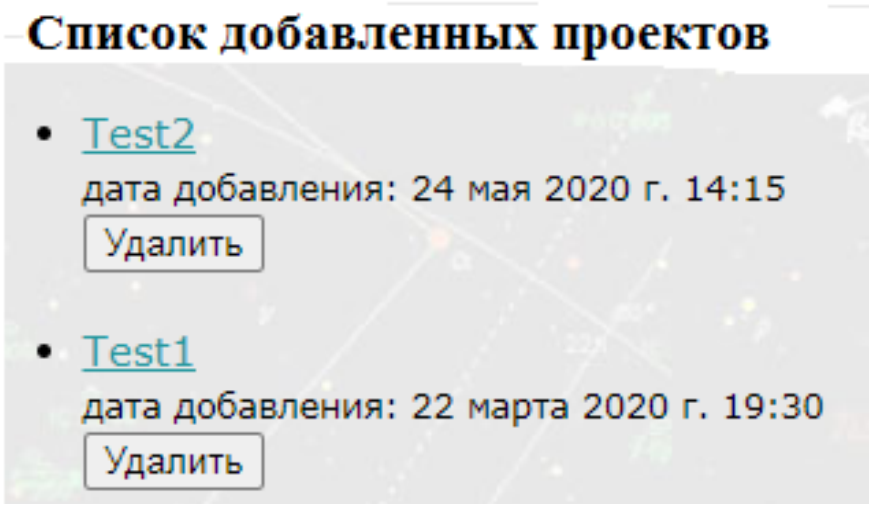
Розроб.	*Нгуєн Т. Н.*			*Метод формування проектних команд з урахуванням міжособистісних відносин співробітників*	
Перевір.	*Імангулова З. А.*				
Н. Контр.	*Імангулова З. А.*				
				СПРм-18-2	*Аркуш 1*
Затверд.	*Гребеннік І.В.*			*СТ*	*Аркушів 1*

ДІАГРАМА ПОСЛІДОВНОСТІ



Розроб.	*Нгуєн Т. Н.*			*Метод формування проектних команд з урахуванням міжособистісних відносин співробітників*	
Перевір.	*Імангулова З. А.*				
Н. Контр.	*Імангулова З. А.*				
				СПРм-18-2	*Аркуш 1*
Затверд.	*Гребеннік І.В.*			*СТ*	*Аркушів 1*

ПОЧАТКОВІ ДАНІ ДЛЯ ВІДБОРУ КАНДИДАТІВ



№ п/п	ФИО	заполнялись выборы?	удаление
1	Тестер1	да	удалить
2	Тестер2	да	удалить
3	Тестер3	да	удалить
4	Тестер4	да	удалить
5	Тестер5	да	удалить
6	Тестер6	да	удалить
7	Тестер7	да	удалить
8	Тестер8	да	удалить
9	Тестер9	**нет**	удалить

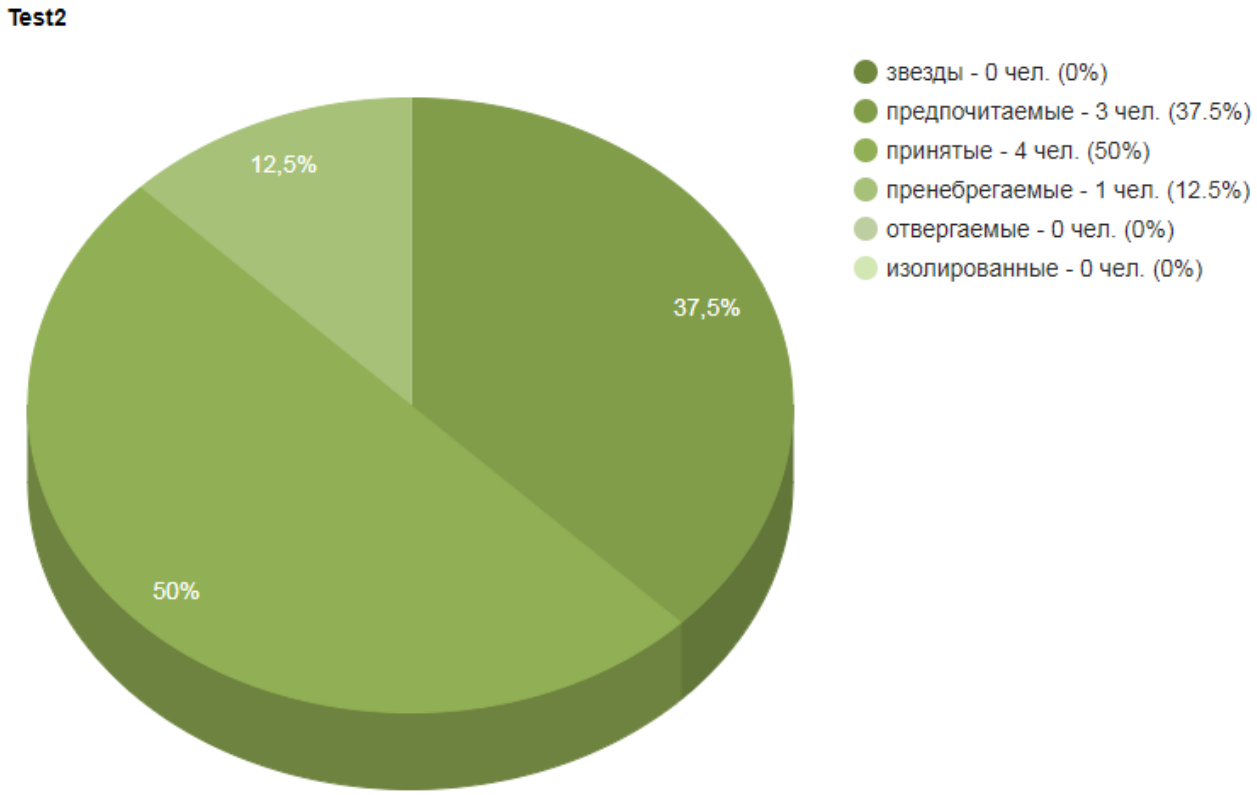
Розроб.	*Нгуєн Т. Н.*			*Метод формування проектних команд з урахуванням міжособистісних відносин співробітників*	
Перевір.	*Імангулова З. А.*				
Н. Контр.	*Імангулова З. А.*				
				СПРм-18-2	*Аркуш 1*
Затверд.	*Гребеннік І.В.*			*СТ*	*Аркушів 1*

СОЦІОМЕТРИЧНА МАТРИЦЯ

№ п/п	Хто вибирає	Кого вибирають серед членів групи								Кількість відданих виборів		
		A_1	B_1	B_2	A_2	C_1	B_3	A_3	C_2	+	-	Всього
1	A_1	X	+	-	0	0	-	+	+	3	2	5
2	B_1	0	X	0	-	0	+	-	+	2	2	4
3	B_2	+	+	X	+	+	0	0	0	4	0	4
4	A_2	0	-	+	X	+	0	-	+	3	2	5
5	C_1	0	0	+	+	X	0	+	-	3	1	4
6	B_3	-	0	+	0	0	X	0	0	1	1	2
7	A_3	+	+	-	+	0	0	X	+	4	1	53
8	C_2	0	+	+	-	0	-	+	X	3	2	5
Кількість отриманих виборів	+	2	4	4	3	2	1	3	4	23		
	-	1	1	2	2	0	2	2	1		11	
Всього		3	5	6	5	2	3	5	5			34

Розроб.	*Нгуєн Т. Н.*			*Метод формування проектних команд з урахуванням міжособистісних відносин співробітників*	
Перевір.	*Імангулова З. А.*				
Н. Контр.	*Імангулова З. А.*				
				СПРм-18-2	*Аркуш 1*
Затверд.	*Гребеннік І.В.*			*СТ*	*Аркушів 1*

РОЗПОДІЛ УЧАСНИКІВ КОМАНДИ ЗА СТАТУСАМИ

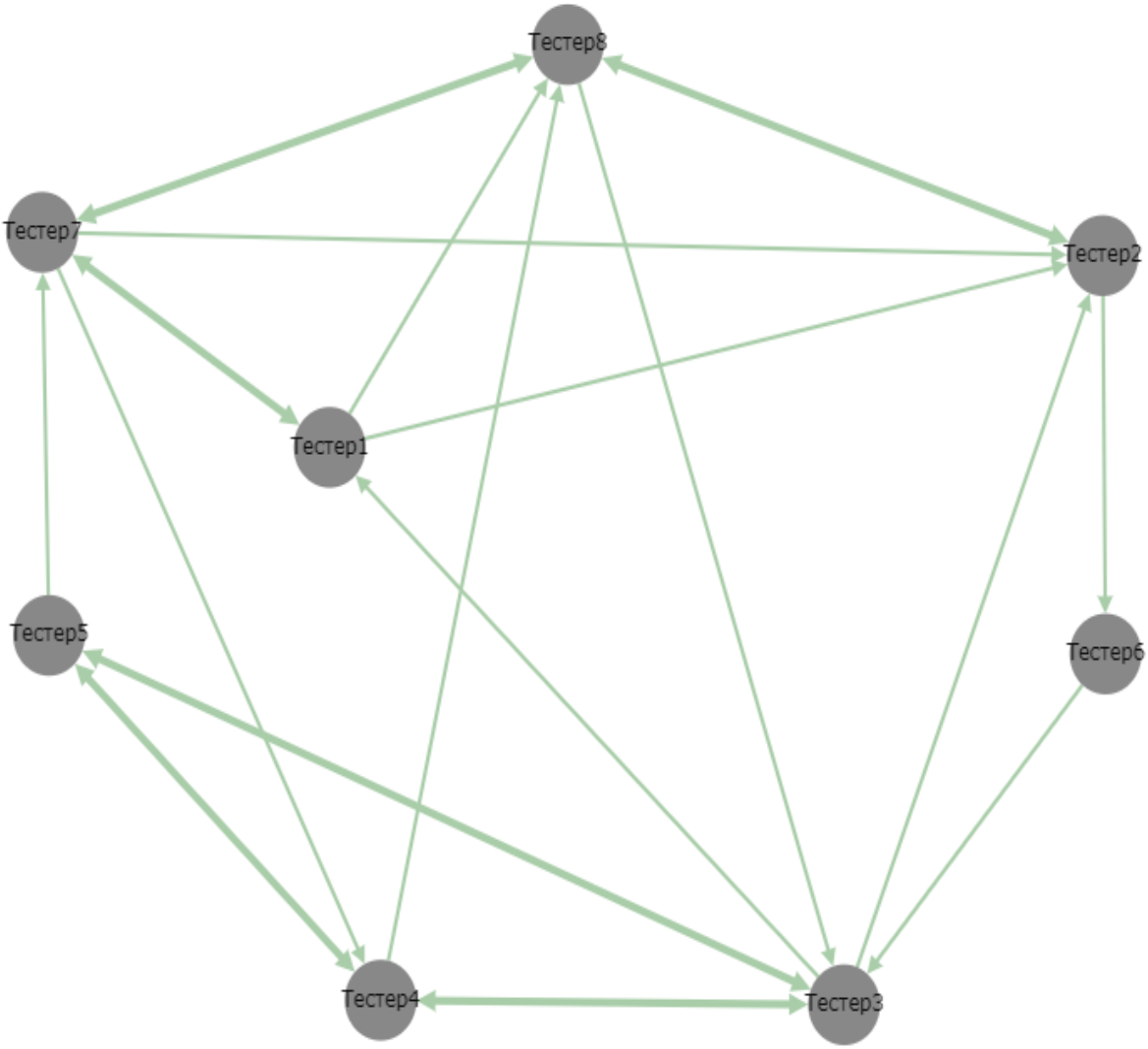


Розроб.	*Нгуєн Т. Н.*			*Метод формування проектних команд з урахуванням міжособистісних відносин співробітників*	
Перевір.	*Імангулова З. А.*				
Н. Контр.	*Імангулова З. А.*				
				СПРм-18-2	*Аркуш 1*
Затверд.	*Гребеннік І.В.*			*СТ*	*Аркушів 1*

ПЕРСОНАЛЬНІ СОЦІОМЕТРИЧНІ ІНДЕКСИ СПІВРОБІТНИКІВ

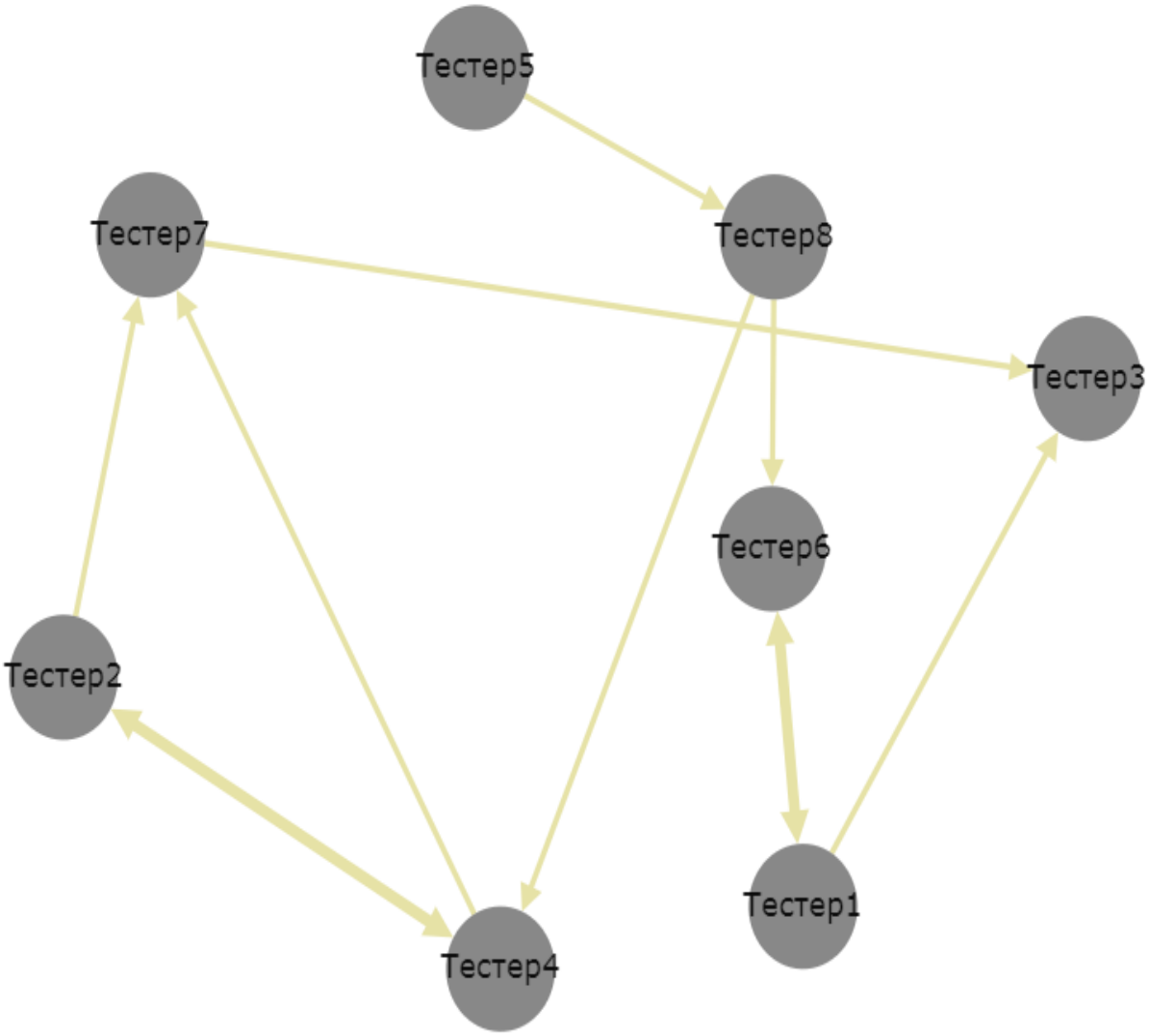
№ п/п	Ф.И.О.	Статус	Персональные социометрические индексы					
			Социометрический статус *			Эмоциональная экспансивность **		
			положи-тельный	отрица-тельный	общий	положи-тельная	отрица-тельная	общая
1	Тестер1	принятый	0.29	0.14	0.14	0.43	0.29	0.71
2	Тестер2	предпочитаемый	0.57	0.14	0.43	0.29	0.29	0.57
3	Тестер3	предпочитаемый	0.57	0.29	0.29	0.57	0	0.57
4	Тестер4	принятый	0.43	0.29	0.14	0.43	0.29	0.71
5	Тестер5	принятый	0.29	0	0.29	0.43	0.14	0.57
6	Тестер6	пренебрегаемый	0.14	0.29	-0.14	0.14	0.14	0.29
7	Тестер7	принятый	0.43	0.29	0.14	0.57	0.14	0.71
8	Тестер8	предпочитаемый	0.57	0.14	0.43	0.43	0.29	0.71

СОЦИОГРАМА ПОЗИТИВНИХ ВИБОРІВ



Розроб.	*Нгуєн Т. Н.*			*Метод формування проектних команд з урахуванням міжособистісних відносин співробітників*	
Перевір.	*Імангулова З. А.*				
Н. Контр.	*Імангулова З. А.*				
				СПРм-18-2	*Аркуш 1*
Затверд.	*Гребеннік І.В.*			*СТ*	*Аркушів 1*

СОЦИОГРАММА НЕГАТИВНИХ ВИБОРІВ



Розроб.	*Нгуєн Т. Н.*			*Метод формування проектних команд з урахуванням міжособистісних відносин співробітників*	
Перевір.	*Імангулова З. А.*				
Н. Контр.	*Імангулова З. А.*				
				СПРм-18-2	*Аркуш 1*
Затверд.	*Гребеннік І.В.*			*СТ*	*Аркушів 1*

РЕЗУЛЬТАТИ ОЦІНЮВАННЯ КАНДИДАТІВ ДО КОМАНДИ ПРОЕКТУ

№ п/п	Ф.И.О.	Личный вклад сотрудника в групповое взаимодействие
1	Тестер1	0,425
2	Тестер2	0,5
3	Тестер3	0,43
4	Тестер4	0,425
5	Тестер5	0,43
6	Тестер6	0,075
7	Тестер7	0,425
8	Тестер8	0,57

№ п/п	Ф.И.О.	Обобщенное значение профессиональных навыков
1	Тестер1	0,38
2	Тестер2	0,34
3	Тестер3	0,58
4	Тестер4	0,8
5	Тестер5	0,42
6	Тестер6	0,4
7	Тестер7	0,78
8	Тестер8	0,72

№ п/п	Ф.И.О.	Обобщенная оценка
1	Тестер1	0,398
2	Тестер2	0,404
3	Тестер3	0,52
4	Тестер4	0,65
5	Тестер5	0,424
6	Тестер6	0,27
7	Тестер7	0,638
8	Тестер8	0,66

ДОДАТОК Б

Текст програми

ГЮИК.506160.011 – 01 12 01

(позначення документу)

ЗАТВЕРДЖЕНО
ГЮИК.506160.011 – 01 12 01 – ЛУ

Метод формування проектних команд з урахуванням міжособистісних відносин співробітників

Текст програми

ГЮИК.506160.011 – 01 12 01 – ЛУ

АРКУШІВ 9

2020 р.

Міністерство освіти і науки України
Харківський Національний Університет Радіоелектроніки

«ЗАТВЕРДЖУЮ»
керівник атестаційної
роботи
доц. Імангулова З.А.

Метод формування проектних команд з урахуванням міжособистісних відносин співробітників

Текст програми

ЛИСТ ЗАТВЕРДЖЕННЯ

ГЮИК.506160.011 – 01 12 01 – ЛУ

РОЗРОБИВ:
ст. гр. СПРм-18-2
Нгуєн Т. Н.

2020 р.

```
package socialBuilding;


import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.annotation.WebServlet;

@WebServlet(name = " socialBuildingInit", urlPatterns = {"/socialBuildingInit"})
public class socialBuildingInit extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        HttpSession session = request.getSession(false);
        request.getRequestDispatcher("/index.jsp").forward(request, response);
    }
}

package socialBuilding.additionalEntity;

public enum UserRole {
    ADMIN, CLIENT;
    public static UserRole getRole(int id) {
        int roleId;
        if(id==1) {
            roleId = 0;
        }
        else {
            roleId=1;
        }
        return UserRole.values()[roleId];
    }

    public String getName() {
        return name().toLowerCase();
    }
}

package socialBuilding.SQL;

import socialBuilding.Interface.IUserDao;

import java.sql.*;

import static java.sql.DriverManager.*;

public class UserSQL implements IUserDao {
    Connection connection;
    Statement statement;
    PreparedStatement preparedStatement;
    ResultSet rs;
```

```
    private static final String ADD_USER = "insert into user(firstName, lastName, patromic,
address, phone, login, password) Values (?, ?, ?, ?, ?, ?, ?)";
    Driver driver;
    private static final String FIND_ACCOUNT = "SELECT id_user FROM user WHERE login=?
AND password=?";

    public UserSQL() {
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            driver = new com.mysql.cj.jdbc.Driver();
            DriverManager.registerDriver(driver);
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        } catch (SQLException e) {
            e.printStackTrace();
        }
        try {
            connection = getConnection("jdbc:mysql://localhost:3306/
socialBuilding?autoReconnect=true&useSSL=false&useLegacyDatetimeCode=false&serverTime
zone=UTC", "root", "root");
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }


    @Override
    public boolean regUser(String firstName, String lastName, String patromic, String login, String
password, String confirmPassword, String address, String phone) {
        int updatedRows = 0;
        if (!firstName.equals("") && !lastName.equals("") && !patromic.equals("")
                && !address.equals("") && !phone.equals("")
                && password.equals(confirmPassword) && !login.equals("")
                && !confirmPassword.equals("")) {
            try {
                System.out.println("prepar");
                preparedStatement = connection.prepareStatement(ADD_USER);
                preparedStatement.setString(1, firstName);
                preparedStatement.setString(2, lastName);
                preparedStatement.setString(3, patromic);
                preparedStatement.setString(4, address);
                preparedStatement.setString(5, phone);
                preparedStatement.setString(6, login);
                preparedStatement.setString(7, password);

                System.out.println("set");
                updatedRows = preparedStatement.executeUpdate();
            } catch (SQLException e) {
            } finally {
                try {
                    preparedStatement.close();
                } catch (SQLException ex) {
```

```
            }
            try {
                connection.close();
            } catch (SQLException ex) {
            }
        }
    }
    if (updatedRows == 0) {
        return false;
    } else {
        return true;
    }
}

@Override
public int loginAccount(String login, String password) {
    int id = 0;
    if (!login.equals("") && !password.equals("")) {
        try {
            preparedStatement = connection.prepareStatement(FIND_ACCOUNT);
            preparedStatement.setString(1, login);
            preparedStatement.setString(2, password);
            rs = preparedStatement.executeQuery();
            while (rs.next()) {
                id = rs.getInt(1);
                System.out.println(rs.getInt(1)+ " "+rs.getInt(2));
            }
        } catch (SQLException ex) {
        } finally {
            try {
                preparedStatement.close();
            } catch (SQLException ex) {
            }
            try {
                connection.close();
            } catch (SQLException ex) {
            }
        }
    }
    return id;
}
}

package socialBuilding.SQL;

import socialBuilding.Interface.IClientDao;
import socialBuilding.Inventory;
import socialBuilding.Order;

import java.sql.*;
import java.util.ArrayList;
```

```
import static java.sql.DriverManager.getConnection;

public class ClientSQL implements IClientDao {
    Connection connection;
    Statement statement;
    PreparedStatement preparedStatement;
    ResultSet resultSet;
    Driver driver;
    private static final String GET_INVENTORYS = "select*from inventorys";
    private static final String GET_ORDERS = "Select * from orders where id_user= ?";
    private static final String MAKE_ORDER = "INSERT INTO orders( id_inventory, id_user,
name, price, descript, src_photo, status) \n" +
            "select id_inventory, ?, name, price, descript, src_photo, 'Not processed' from inventorys
where id_inventory=?";

    public ClientSQL() {
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            driver = new com.mysql.cj.jdbc.Driver();
            DriverManager.registerDriver(driver);
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        } catch (SQLException e) {
            e.printStackTrace();
        }
        try {
            connection = getConnection("jdbc:mysql://localhost:3306/
socialBuilding?autoReconnect=true&useSSL=false&useLegacyDatetimeCode=false&serverTime
zone=UTC", "root", "root");
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    @Override
    public ArrayList<Inventory> getInventory() {
        ArrayList<Inventory> inventorys = new ArrayList<>();
        try {
            statement = connection.createStatement();
            resultSet = statement.executeQuery(GET_INVENTORYS);
            while (resultSet.next()) {
                Inventory inventory = new Inventory();
                inventory.setId(resultSet.getInt(1));
                inventory.setType(resultSet.getString(2));
                inventory.setPrice(resultSet.getInt(3));
                inventory.setDescription(resultSet.getString(4));
                inventory.setUrl(resultSet.getString(5));
                inventorys.add(inventory);
            }
        } catch (SQLException ex) {
        } finally {
            try {
```

```
            resultSet.close();
        } catch (SQLException ex) {
        }
        try {
            statement.close();
        } catch (SQLException ex) {
        }
        try {
            connection.close();
        } catch (SQLException ex) {
        }
    }
    return inventorys;
}

@Override
public ArrayList<Order> getOrders(int idClient) {
    ArrayList<Order> orders = new ArrayList<>();
    try {
        preparedStatement = connection.prepareStatement(GET_ORDERS);
        preparedStatement.setInt(1, idClient);
        resultSet = preparedStatement.executeQuery();
        AdminSQL.queryGetOrders(orders, resultSet);
    } catch (SQLException ex) {
    }
    finally {
        try {
            resultSet.close();
        } catch (SQLException ex) {
        }
        try {
            preparedStatement.close();
        } catch (SQLException ex) {
        }
        try {
            connection.close();
        } catch (SQLException ex) {
        }
    }
    return orders;
}

@Override
public boolean makeOrder(String idInventory, int idClient) {
    int updateRows = 0;
    try {
        preparedStatement = connection.prepareStatement(MAKE_ORDER);
        preparedStatement.setInt(1, idClient);
        preparedStatement.setString(2, idInventory);

        updateRows = preparedStatement.executeUpdate();
    } catch (SQLException e) {
```

```
            e.printStackTrace();
        } finally {

            try {
                preparedStatement.close();
            } catch (SQLException ex) {
            }
            try {
                connection.close();
            } catch (SQLException ex) {
            }
        }
        if (updateRows == 0) {
            return false;
        } else {
            return true;
        }
    }
}

package socialBuilding.SQL;

import socialBuilding.Interface.IAdminDao;
import socialBuilding.Order;

import java.sql.*;
import java.util.ArrayList;

import static java.sql.DriverManager.getConnection;

public class AdminSQL implements IAdminDao {
    Connection connection;
    Statement statement;
    PreparedStatement preparedStatement;
    ResultSet resultSet;
    Driver driver;
    private static final String CHANGE_STATUS = "update orders set status = ? where
id_order=?";
    private static final String GET_ORDERS = "Select * from orders";

    public AdminSQL() {
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            driver = new com.mysql.cj.jdbc.Driver();
            DriverManager.registerDriver(driver);
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        } catch (SQLException e) {
            e.printStackTrace();
        }
        try {
```

```
            connection = getConnection("jdbc:mysql://localhost:3306/
socialBuilding?autoReconnect=true&useSSL=false&useLegacyDatetimeCode=false&serverTime
zone=UTC", "root", "root");
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    @Override
    public ArrayList<Order> getOrders() {
        ArrayList<Order> orders = new ArrayList<>();
        try {
            statement = connection.createStatement();
            resultSet = statement.executeQuery(GET_ORDERS);
            queryGetOrders(orders, resultSet);
        } catch (SQLException e) {
        } finally {
            try {
                resultSet.close();
            } catch (SQLException ex) {
            }
            try {
                statement.close();
            } catch (SQLException ex) {
            }
            try {
                connection.close();
            } catch (SQLException ex) {
            }
        }
        return orders;
    }

    @Override
    public boolean changeStatus(String status, String idOrder) {
        int updateRows = 0;
        try {
            preparedStatement = connection.prepareStatement(CHANGE_STATUS);
            preparedStatement.setString(1, status);
            preparedStatement.setString(2, idOrder);
            updateRows = preparedStatement.executeUpdate();
        } catch (SQLException e) {
        } finally {
            try {
                preparedStatement.close();
            } catch (SQLException ex) {
            }
            try {
                connection.close();
            } catch (SQLException ex) {
            }
        }
```

```
        if (updateRows == 0) {
            return false;
        } else {
            return true;
        }
    }

    public static void queryGetOrders(ArrayList<Order> orders, ResultSet resultSet) throws
SQLException {
        while (resultSet.next()) {
            Order order = new Order();
            order.setIdOrder(resultSet.getInt(1));
            order.setIdInventory(resultSet.getInt(2));
            order.setIdUser(resultSet.getInt(3));
            order.setInventoryType(resultSet.getString(4));
            order.setPrice(resultSet.getInt(5));
            order.setDescript(resultSet.getString(6));
            order.setUrl(resultSet.getString(7));
            order.setStatus(resultSet.getString(8));
            orders.add(order);
        }
    }
}
```

ДОДАТОК В

Сертифікат учасника конференції

№	Позначення	Найменування	Додаткові відомості
		Текстові документи	
1.	ГЮИК.506160.011 ПЗ	Пояснювальна записка	60 стор.
3.	ГЮИК. 506160.011 -01 12 01	Текст програми	8 стор.
		Графічні документи	
5.		Діаграма варіантів використання	1 аркуш
6.		Діаграма послідовності	1 аркуш
7.		Початкові дані для відбору кандидатів	1 аркуш
8.		Соціометрична матриця	1 аркуш
9.		Розподіл учасників команди за статусами	1 аркуш
10.		Персональні соціометричні індекси співробітників	1 аркуш
11.		Социограма позитивних виборів	1 аркуш
12.		Социограмма негативних виборів	1 аркуш
13.		Результати оцінювання кандидатів до команди проекту	1 аркуш

					ГЮИК. 506160.011 Д3			
Змін.	Арк.	№ докум.	Підп.	Дата				
Розроб.		*Нгуєн Т. Н.*			Метод формування проектних команд з урахуванням міжособистісних відносин співробітників		Аркуш	Аркушів
Перевір.		*Імангулова З. А.*					*1*	*1*
Н. Контр.		*Імангулова З. А.*				*ХНУРЕ Кафедра СТ*		
Затв.		*Гребеннік І.В.*						