



ТЕХНИЧЕСКИ УНИВЕРСИТЕТ - ВАРНА
ФАКУЛТЕТ ПО ИЗЧИСЛИТЕЛНА ТЕХНИКА И
АВТОМАТИЗАЦИЯ

катедра
„Компютърни науки и технологии”
дисциплина
„Обектно-ориентирано програмиране
– 2 част”

КУРСОВ ПРОЕКТ

„Хотел”

Изготвили:

Дилян Иванов

Фак. №: 17621639

Валентин Костадинов

Фак. №: 17621780

Специалност: СИТ

Курс: III

Група: I^a

Проверил:

доц. Христо Ненов

III. Хотел

Да се разработи информационна услуга – Хотел. Програмата съхранява и обработва данни за хотелски услуги (резервации и допълнителни услуги). Системата позволява множествен достъп.

Системата поддържа два вида потребители администратор и клиенти (рецепционист, мениджър, собственик) с различни роли за достъп до функционалностите в системата.

Операции за работа с потребители:

- Създаване на собственици на хотел(и) от администратор;
- Създаване на хотел с мениджър от собственик;
- Създаване на рецепционисти от мениджър

Системата поддържа операции за работа с резервации;

- Създаване на клиенти
- Създаване на резервация от рецепционист (Номер на резервация, Тип на резервация, Тип на прекратяване на резервацията, категории стаи, ...);
- Създаване и предоставяне на допълнителни услуги, съобразени със сезона (отчетност на тип услуга и брой ползвания);
- Рейтинговане на клиенти;

Системата поддържа справки по произволен период за:

- Категория клиенти :
 - I. Информаация за клиенти (лични данни);
 - II. Използване на хотел и хотелски услуги;
 - III.Рейтинг на клиенти;
- Рецепционисти (създадени резервации, данните на рецепциониста)
- Създадени резервации (дата, статус, хотел, съдържание на формуляра);
- Стаи (рейтинг на стаите за ползваемост);

Миниджър на хотел достъпва справки само за хотел, за който е отговорен. Собственика достъпва справки за всички притежавани хотели. Рецепциониста има право на справки за заетостта на стаите.

Системата поддържа Известия за събития:

- Изтичаща резервация;
- Известия за рисков клиент (при създаване на нова резервация);

Функционални изисквания:

Потребители

- Програмата съдържа два типа потребители – администратор и клиенти.
- **Администратор** създава профили на собственици на хотели.
- **Собственикът** създава профили на мениджърите.
- **Мениджърът** създава профили на рецепционистите.
- **Рецепционистът** въвежда информация за новите клиенти, създава резервации, може (при нужда) да поставя рейтинг на клиентите (отделно от автоматичния метод за пресмятане), добавя нови стаи (вкл. Типа им, както и възможност за редактиране на съществуващи такива).

Справки

- Всеки един от потребителите има достъп до различни справки.
- **Собственикът** достъпва справки за собствените си хотели: информация за гостите (по уникален идентификационен номер /ГСМ/), рейтинг на стаите, информация за рецепционистите, създадени резервации (търсене по дата).
- **Мениджърът** има право на достъп до информацията за гостите (само за хотелът в който е назначен), рейтинг на стаите, информация за рецепционистите, информация за направените резервации (включващи и рецепциониста, който ги е създал);
- **Рецепционистът** достъпва справки само за хотелът, в който е назначен : информация за стаите, гостите и резервациите;
- **Складовият агент** достъпва справки за складовете, които поддържа, и пази в профила си информация за всички създадени договори.

Система

- Системата позволява множествен достъп.
- Влизането в системата става чрез индивидуално потребителско име и парола.
- Системата позволява създаване на нови потребителски профили.
- Системата позволява добавяне на нови хотели и собственици към тях (последвани от персонал за текущия хотел).
- Системата позволява създаване на нови допълнителни услуги
- Системата позволява създаване на нови категории стаи.
- Системата позволява редактиране на текущи стаи.
- Системата позволява редактиране на гости.
- Системата позволява редактиране на резервации.
- Системата позволява както автоматичното пресмятане на специален **рейтинг** на гостите, така и ръчното му въвеждане и редактиране.
- Системата поддържа известия за изтичащи резервации.

Известия

- **Рецепционистът** получава известие при изтичаща резервация (включващо информация за ID на гостът и резервацията, както и номер на стаята);

Проектиране на системата:

База Данни

- Базата данни е реализирана със системата MySQL.
- Таблицата **users** в базата данни съдържа информация за **рецепционистите**. User съдържа информация за техните: потребителско име, парола, име, фамилия, хотел, в който са назначени. Връзката към таблицата се осъществява чрез първичен ключ 'id'
- Таблицата **admin** служи за съхранение на информацията за **администраторът/те**. Таблицата включва полетата : admin_id, admin_username и admin_password.
- Таблицата **extra_services** съхранява информация за името на допълнителната услуга, цената, както и уникален идентификационен номер - es_price.
- Таблицата **guest_table** съдържа информация за гостите на хотела : ID (gID), име (first_name), фамилия (last_name), телефонен номер (GSM), имейл (email), рейтинг (guest_rating).
- Таблицата **manager** е предназначена за съхранение на информацията за мениджърите на хотелите. Включва : ID (manager_id), потребителско име (manager_username), парола (manager_password) , име на хотела (manager_hotel_name).
- Таблицата **owners** съдържа информация за собствениците на хотелите. Включва: ID (owners_id), потребителско име (owners_username), парола (owners_password), име на хотела (hotel_name).
- Таблицата **reservations** съдържа информация за направените резервации: ID на гостите, дата на пристигане, дата на заминаване, рецепционистът направил резервацията, причина за отмяна на резервацията (ако има такава).
- Таблицата **reserv_guest_extra_services** съхранява информация за предлаганите допълнителни услуги. Съдържа ID на госта, ID на избраната услуга, цена, количество от дадената услуга, обща сума.
- Таблицата **rooms** съдържа информация за стаите: номер на стаята, типа ѝ (записан с целочислено число, започващо от 1), телефон на стаята (ако има такъв), статус за заетостта на стаята (заета/свободна).
- Таблицата **roomtype** съдържа информация относно типа на стаята. Съдържа: ID на стаята, описание за типа ѝ, цена.

Бизнес логика

- Програмата представлява разработена информационна система, която съхранява и обработва данни за хотел. Потребителите на програмата са разделени в две категории – администратор и клиенти. Потребителите се

вписват в системата посредством потребителско име и парола. При невалидни такива се извежда грешка и достъпът е отказан.

- Администраторът има право да създава собственици на хотели. Неговият достъп до системата се задава предварително (username / password).
- Собственикът на хотел има право да създава профили на мениджъри, да преглежда информацията за гостите на хотела, рейтинга на стаите, информацията за реципционистите, както и за създадените рецепционисти.
- Миндиджърът има право да създава профили за реципционистите, да преглежда информацията за гостите, рейтинга за стаите, информацията за рецепционистите, както и детайлите за създадените резервации..
- Рецепционистите виждат изтичащите резервации, информация за стите (която могат да редактират), информация за гостите (може да бъде редактирана, включително автоматично – изчисляваният рейтинг), информация за резервациите (включва и евентуална отмяна на резервация, със поле за допълнителна информация).
- Гостите на хотела получават автоматичен рейтинг от системата: при създаването на нов гост автоматично му се „въвежда“ рейтинг 0. При наемане на стая в зависимост от цената и дните на престой се добавя рейтинг : $\text{дни} \times \text{цена} / 100$, При прекратяване на резервация формулата е следната : $\text{дни} \times \text{цена} / 100 \times (-1.5)$.

Графичен интерфейс

Създаденият графичен интерфейс е осъществен с помощта на JavaFX.

Чрез началният екран различните видове потребители могат да се вписват в системата :

USER LOGIN SYSTEM

USERNAME:

PASSWORD:

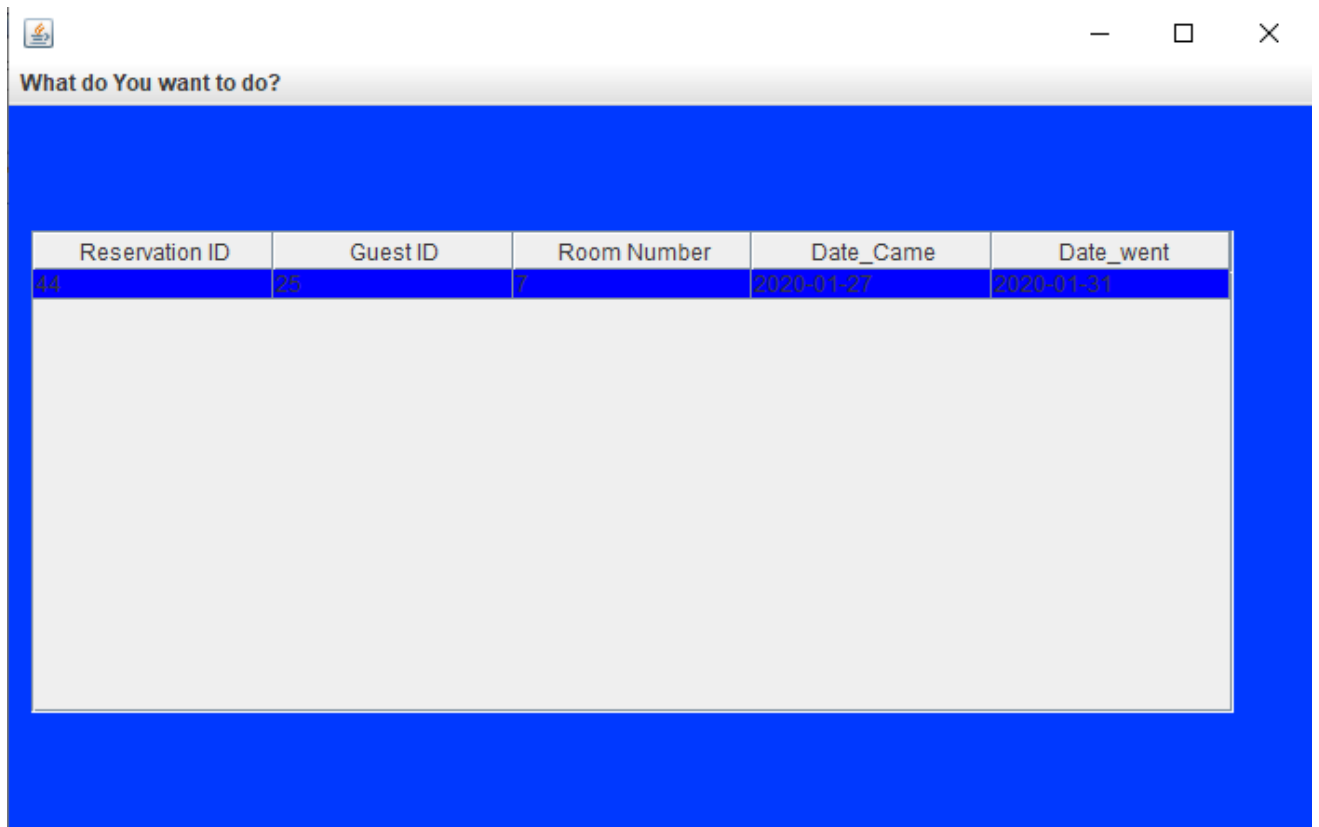
☐ Admin

☐ Owner

☐ Manager

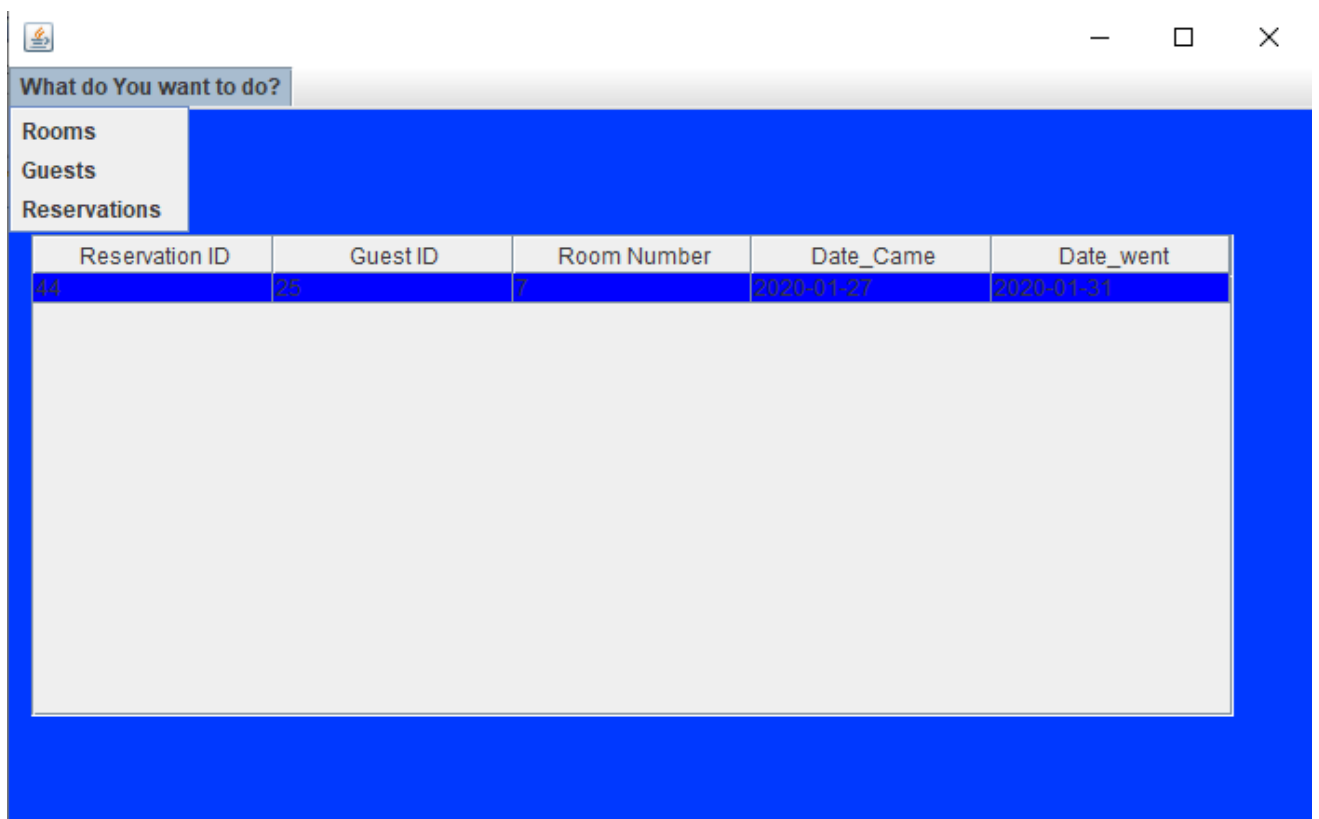
☐ Receptionist

LOGIN




Начален прозорец за рецепционистите.



Началният прозорец съдържа падащо меню, чрез което се достъпват лесно различните функции и опции, преназначени за рецепционистите.



За всеки форм, съдържащ търсене по дата има предоставени две DatePicker полета за въвеждане на начална и крайна дата. Данните се извеждат в таблица.

 — □ ×

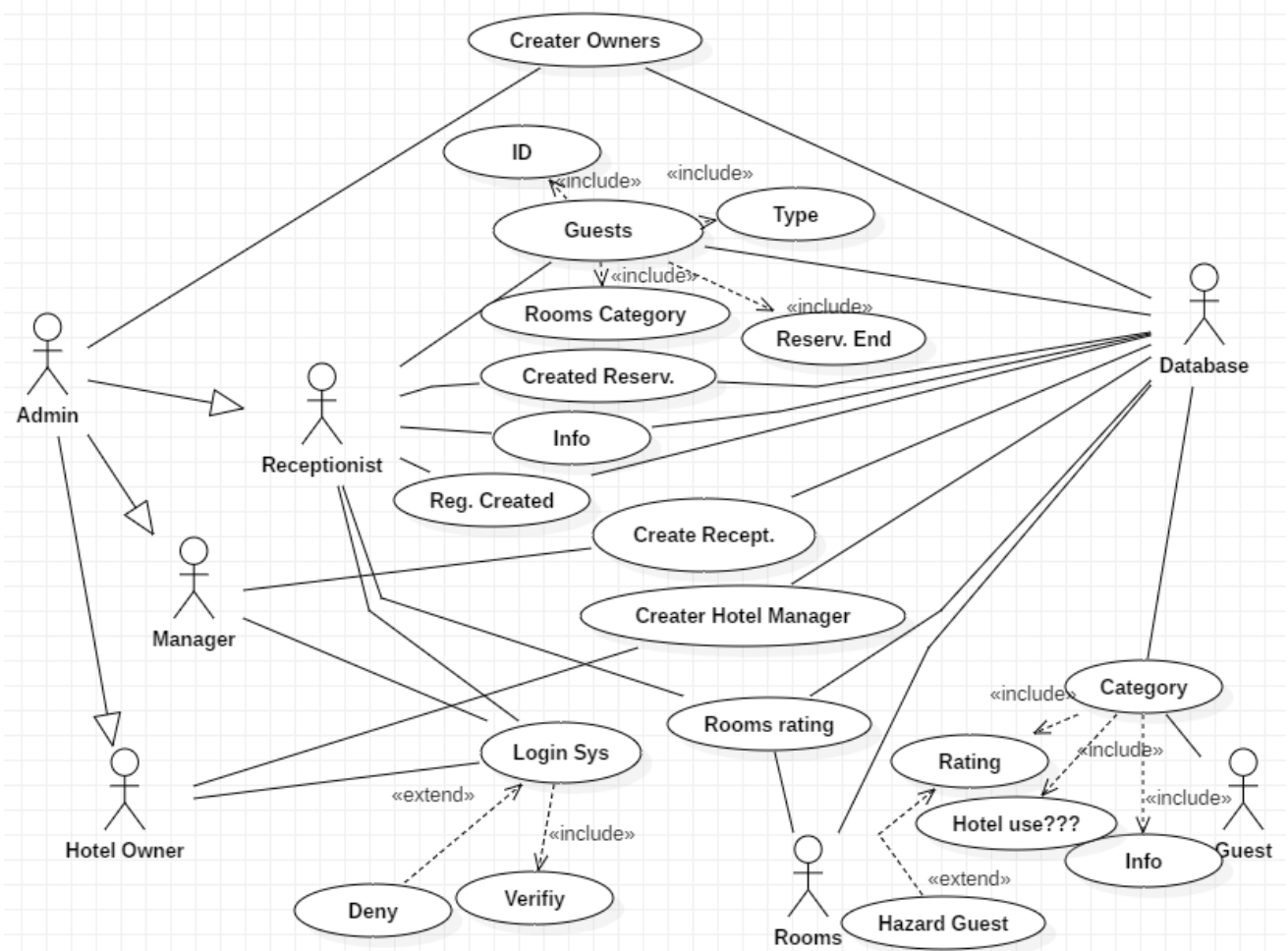
RESERVATION INFO

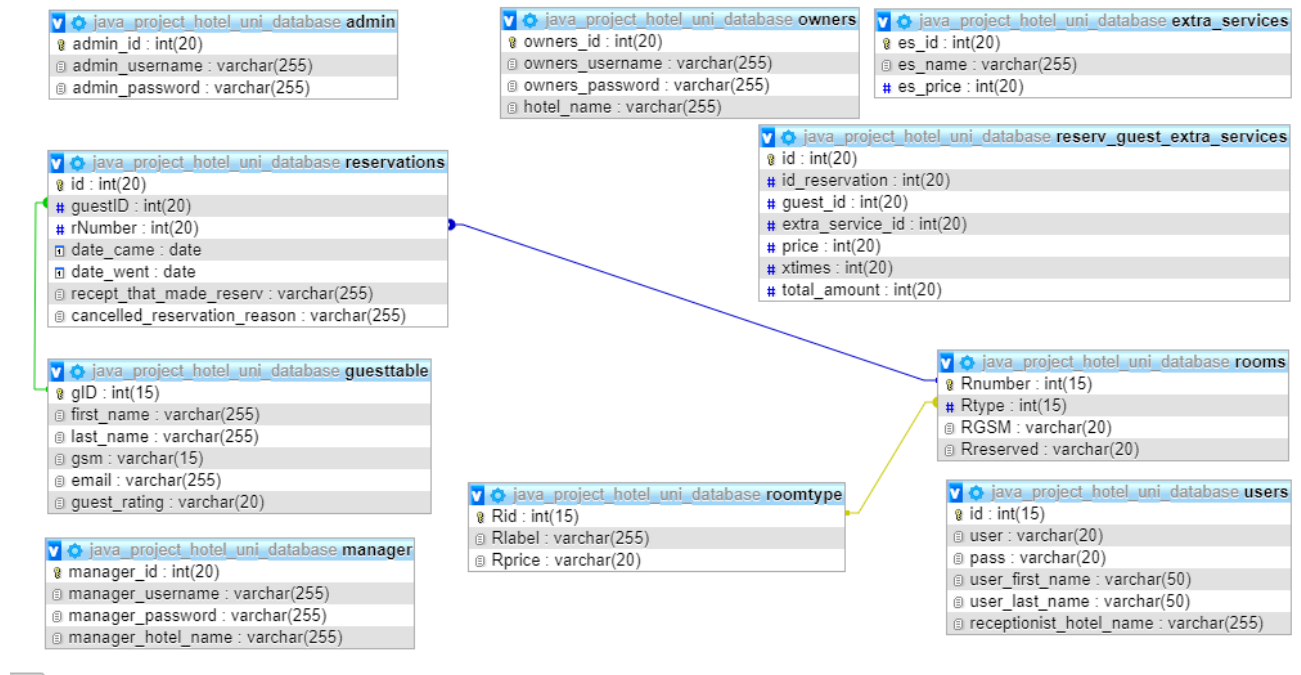
Search By Date of Registration

Reservation ID	Guest ID	Room Number	Date_Came	Date_went	Recept. That Mad...
36	26	5	2020-01-07	2020-01-30	hoteltest
37	31	6	2020-01-07	2020-01-08	hoteltest
38	10	6	2020-01-04	2020-01-06	lorem1
39	17	4	2020-01-01	2020-01-06	lorem1
40	13	2	2020-01-02	2020-01-06	
41	15	3	2020-01-02	2020-01-06	
42	4	6	2020-01-03	2020-01-06	
43	28	5	2020-01-04	2020-01-06	
44	25	7	2020-01-27	2020-01-31	hoteltest

Use Case Diagram:



Релационна схема на базата от данни:



Реализация на системата:

Реализация на слоя за работа с базата данни

```
public Connection devConnect()
{
    Connection con = null;
    MySQLDataSource msds = new MySQLDataSource();
    msds.setServerName("localhost");
    msds.setPortNumber(3306);
    msds.setUser("root");
    msds.setPassword("");
    msds.setDatabaseName("java_project_hotel_uni_database");
    try {
        con = msds.getConnection();
    } catch (SQLException ex) {
        Logger.getLogger(my_SQL_Connect_Class.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

```
return con;  
}
```

Реализация на бизнеслогика и графичен интерфейс

Функция за вход в програмата :

```
private void btn1_loginActionPerformed(java.awt.event.ActionEvent evt) {      //Бутон за вход в  
системата
```

```
    PreparedStatement PreS1 = null;  
    ResultSet ResS1;
```

```
    String uName = textBox1_user.getText(); // запазване на текста от полето във променлива  
    String uPass = String.valueOf(textBox2_pass.getPassword());
```

```
    if(uName.length() <= 4) //проверка за дължината на потребителското име – по-голяма ли е  
от 4
```

```
    {  
        showMessageDialog(null, "Your username should be longer! ", "Error",  
ERROR_MESSAGE);      //извеждане на съобщение при прекалено късо потребителско  
име
```

```
    }else if(uPass.length() <=4 )  
    {  
        showMessageDialog(null, "Your password should be longer! ", "Error",  
ERROR_MESSAGE);  
    }
```

```
    my_SQL_Connect_Class obj1 = new my_SQL_Connect_Class(); // обект от класа за връзка с  
базата данни
```

```
    if(admin_RadioButton.isSelected()) //при селектиран радио бутон 'админ'  
    {  
        String slcQry = "SELECT * FROM `admin` WHERE `admin_username`=? AND  
`admin_password`=?";
```

```
    try {  
        PreS1 = obj1.devConnect().prepareStatement(slcQry);
```

```

PreS1.setString(1, uName);
PreS1.setString(2, uPass);
ResS1 = PreS1.executeQuery();

adminForm ins1 = new adminForm(); //обект от класа за вход във админ форм-а

if(ResS1.next() == true)
{
    ins1.setVisible(true);
    ins1.pack();
    this.dispose();

}else
{
    showMessageDialog(null, "Wrong Data! ", "Error", ERROR_MESSAGE);
}

} catch (SQLException ex) {
    Logger.getLogger(Form_Login.class.getName()).log(Level.SEVERE, null, ex);
}
}

```

Функция за проверка за съществуване на същия GSM номер :

```

public int checkIfThereIsAnotherGSMNumberLikeThisOne(String phone)
{
    String slctQry_1 = String.format("SELECT * FROM `guesttable` WHERE `gsm`='%s'",phone);
    try {
        PreparedStatement PrepaSt_1 = mycon1.devConnect().prepareStatement(slctQry_1);
        ResultSet ResSet_1 = PrepaSt_1.executeQuery();

        if(ResSet_1.next() )
        {
            return 1;
        }
    } catch (SQLException ex) {

```

```
        Logger.getLogger(GuestClass.class.getName()).log(Level.SEVERE, null, ex);
    }
    return 0;
}
```

Функция за добавяне на гост в БД :

```
public boolean AddingGuests(String FN, String LN, String GSM, String Email)
{
    String qry = "INSERT INTO `guestTable`(`first_name`, `last_name`, `gsm`, `email`,
`guest_rating`) VALUES (?, ?, ?, ?, ?)";

    try {
        PreparedStatement PpdSt_1 = mycon1.devConnect().prepareStatement(qry);

        PpdSt_1.setString(1, FN);
        PpdSt_1.setString(2, LN);
        PpdSt_1.setString(3, GSM);
        PpdSt_1.setString(4, Email);
        PpdSt_1.setString(5, "0");

        return (PpdSt_1.executeUpdate() > 0);

    } catch (SQLException ex) {
        Logger.getLogger(GuestClass.class.getName()).log(Level.SEVERE, null, ex);
        return false;
    }
}
```

Функция за редактиране на гост :

```

public boolean editingSelectedGuest(int id, String FN, String LN, String GSM, String Email, String
Rating)
{

    String qry_editingSelectedGuest = "UPDATE `guesttable` SET
`first_name`=?, `last_name`=?, `gsm`=?, `email`=?, `guest_rating`=? WHERE `gID`=?"

    try {
        PreparedStatement PpdSt_1 =
mycon1.devConnect().prepareStatement(qry_editingSelectedGuest);

        PpdSt_1.setString(1, FN);
        PpdSt_1.setString(2, LN);
        PpdSt_1.setString(3, GSM);
        PpdSt_1.setString(4, Email);
        PpdSt_1.setString(5, Rating);
        PpdSt_1.setInt(6, id);

        return (PpdSt_1.executeUpdate() > 0);

    } catch (SQLException ex) {
        Logger.getLogger(GuestClass.class.getName()).log(Level.SEVERE, null, ex);
        return false;
    }
}

```

Функция за изтриване на гост :

```

public boolean delGuest(int idOfGuest)
{

    String qryDELETE = "DELETE FROM `guesttable` WHERE `gID`=?"

    try {

```

```

PreparedStatement PpdSt_1 = mycon1.devConnect().prepareStatement(qryDELETE);

PpdSt_1.setInt(1, idOfGuest);

return (PpdSt_1.executeUpdate() > 0);

} catch (SQLException ex) {
    Logger.getLogger(GuestClass.class.getName()).log(Level.SEVERE, null, ex);
    return false;
}
}

```

Функция за попълване данните (от БД) в таблицата :

```

public void addingItemsIntoTable(JTable myGuestTable)
{
    String slctQry_1 = "SELECT * FROM `guestTable`";
    try {
        PreparedStatement PrepaSt_1 = mycon1.devConnect().prepareStatement(slctQry_1);
        ResultSet ResSet_1 = PrepaSt_1.executeQuery();
        DefaultTableModel DftTM1 = (DefaultTableModel)myGuestTable.getModel();
        Object[] line;
        while(ResSet_1.next() )
        {
            line = new Object[6];
            line[0] = ResSet_1.getInt(1);
            line[1] = ResSet_1.getString(2);
            line[2] = ResSet_1.getString(3);
            line[3] = ResSet_1.getString(4);
            line[4] = ResSet_1.getString(5);
            line[5] = ResSet_1.getString(6);

            DftTM1.addRow(line);
        }
    } catch (SQLException ex) {
        Logger.getLogger(GuestClass.class.getName()).log(Level.SEVERE, null, ex);
    }
}

```

```
}
```

Функция за извеждане на текущия рейтинг на гост на клиента от БД :

```
public int get_guest_rating_from_DB(int gID)
{
    String slctQry_1 = String.format("SELECT `guest_rating` FROM `guesttable` WHERE
`gID`=%d",gID);
    try {
        PreparedStatement PrepaSt_1 = mycon1.devConnect().prepareStatement(slctQry_1);
        ResultSet ResSet_1 = PrepaSt_1.executeQuery();

        if(ResSet_1.next() )
        {
            return Integer.valueOf(ResSet_1.getString(1));
        }
    } catch (SQLException ex) {
        Logger.getLogger(GuestClass.class.getName()).log(Level.SEVERE, null, ex);
    }
    return 0;
}
```

Функция за добавяне на рейтинг към гост в БД :

```
public boolean add_guest_Rating_in_DB(int gID, int rating)
{
    String qry_editingSelectedGuest = "UPDATE `guesttable` SET `guest_rating`=? WHERE
`gID`=?";

    try {
        PreparedStatement PpdSt_1 =
mycon1.devConnect().prepareStatement(qry_editingSelectedGuest);
```

```

        PpdSt_1.setString(1, String.valueOf(rating));
        PpdSt_1.setInt(2, glID);

        return (PpdSt_1.executeUpdate() > 0);

    } catch (SQLException ex) {
        Logger.getLogger(GuestClass.class.getName()).log(Level.SEVERE, null, ex);
        return false;
    }
}

```

// Функция всеки 1ви ред във таблицата да е син и всеки 2ри да е сив :

```

jTable1.setDefaultRenderer(Object.class, new DefaultTableCellRenderer()
{
    @Override
    public Component getTableCellRendererComponent(JTable table, Object value, boolean
isSelected, boolean hasFocus, int row, int column)
    {
        final Component c = super.getTableCellRendererComponent(table, value, isSelected,
hasFocus, row, column);
        c.setBackground(row % 2 == 0 ? Color.BLUE : Color.LIGHT_GRAY);
        return c;
    }
});

```

// Функция за извеждане в таблицата клиентите чиито резервации изтичат в текущия ден

```

private void adding_ENDING_ReservationsIntoTable(JTable myGuestTable)
{

    String todays_Date = java.time.LocalDate.now().toString(); //Формат : yyyy-MM-dd

    // -----

```


// Изтриване на информацията в таблицата :

```
DefaultTableModel model = (DefaultTableModel) myGuestTable.getModel();
```

```
model.setRowCount(0);
```

```
// -----
```

```
String slctQry_1 = String.format("SELECT * FROM `reservations` WHERE `date_went`='%s' AND `cancelled_reservation_reason`='%s'",today's_Date,"");
```

/* по този начин НЕ се визуализират отменените резервации (при всички без отменените резервации полето cancelled_reservation_reason в DB е празен стринг)*/

```
try {
```

```
    PreparedStatement PrepaSt_1 =
```

```
mysqlconn_reservation_obj1.devConnect().prepareStatement(slctQry_1);
```

```
    ResultSet ResSet_1 = PrepaSt_1.executeQuery();
```

```
    DefaultTableModel DftTM1 = (DefaultTableModel)myGuestTable.getModel();
```

```
    Object[] line;
```

```
    while(ResSet_1.next() )
```

```
    {
```

```
        line = new Object[5];
```

```
        line[0] = ResSet_1.getInt(1);
```

```
        line[1] = ResSet_1.getInt(2);
```

```
        line[2] = ResSet_1.getInt(3);
```

```
        line[3] = ResSet_1.getString(4);
```

```
        line[4] = ResSet_1.getString(5);
```

```
        DftTM1.addRow(line);
```

```
    }
```

```
} catch (SQLException ex) {
```

```
    Logger.getLogger(GuestClass.class.getName()).log(Level.SEVERE, null, ex);
```

```
}
```

```
}
```

Функция за извличане на номера на стаята от резервацията :

```

public int getRoomNumberFromReservation(int reservationID)
{
    String slctQry_1 = "SELECT `rNumber` FROM `reservations` WHERE `id` = ?";
    try {
        PreparedStatement PrepaSt_1 =
mysqlconn_reservation_obj1.devConnect().prepareStatement(slctQry_1);

        PrepaSt_1.setInt(1, reservationID);

        ResultSet ResSet_1 = PrepaSt_1.executeQuery();

        if(ResSet_1.next())
        {
            return ResSet_1.getInt(1);
        }else
        {
            return 0;
        }

    } catch (SQLException ex) {
        Logger.getLogger(GuestClass.class.getName()).log(Level.SEVERE, null, ex);
        return 0;
    }
}

```

Функция за добавяне на стаите във комбобокс :

```

public void addingTypeOfRoomsIntoComboBox(JComboBox myComboBox)
{
    String slctQry_1 = "SELECT * FROM `roomtype`";
    try {
        PreparedStatement PrepaSt_1 = mycon1.devConnect().prepareStatement(slctQry_1);
        ResultSet ResSet_1 = PrepaSt_1.executeQuery();

        while(ResSet_1.next() )
        {

```

```

        myComboBox.addItem(ResSet_1.getInt(1));
    }
} catch (SQLException ex) {
    Logger.getLogger(GuestClass.class.getName()).log(Level.SEVERE, null, ex);
}
}

```

Функция за промяна статуса на стаята на резервирана :

```

public boolean setingRoomToReserved(int number, String isFree)
{

    String qry_editingSelectedGuest = "UPDATE `rooms` SET `Reserved`=? WHERE
`Rnumber`=?";

    try {
        PreparedStatement PpdSt_1 =
mycon1.devConnect().prepareStatement(qry_editingSelectedGuest);

        PpdSt_1.setString(1, isFree);
        PpdSt_1.setInt(2, number);

        return (PpdSt_1.executeUpdate() > 0);

    } catch (SQLException ex) {
        Logger.getLogger(GuestClass.class.getName()).log(Level.SEVERE, null, ex);
        return false;
    }
}

```

// Функция за създаване на собственик на хотел :

```

private void create_hotel_owners_btn1ActionPerformed(java.awt.event.ActionEvent evt)
{

    String _UserName = textBox1_user_name.getText();
    String _UserPassword = textBox2_password.getText();
    String _HotelName = textBox3_hotel_name.getText();

    if(textBox1_user_name.getText().equals("") || textBox2_password.getText().equals("") )
    {
        showMessageDialog(null, "Please fill all of the text boxes! ", "Error", ERROR_MESSAGE);
    }else
    {
        if(AddingOwner(_UserName, _UserPassword, _HotelName))
        {
            SuccessOrNot_label1.setText("Successfully Added");
        }else
        {
            SuccessOrNot_label1.setText("Something Went Wrong");
        }
    }
}

```

// Функция, която позволява (в случая) числата само от 0-9, backspace и delete

```

Robot rob;

public void restrictTextFieldsInput(java.awt.event.KeyEvent evt)
{
    try {
        rob= new Robot();
        char c = evt.getKeyChar();
        if (!(c >= '0') && (c <= '9') ||
            (c == KeyEvent.VK_BACK_SPACE) ||
            (c == KeyEvent.VK_DELETE))) {
            getToolkit().beep();
            rob.keyPress(KeyEvent.VK_BACK_SPACE);
            rob.keyRelease(KeyEvent.VK_BACK_SPACE);
            evt.consume();
        }
    }
}

```

```
    }} catch (AWTException ex) {  
        Logger.getLogger(extra_Services_Form.class.getName()).log(Level.SEVERE, null, ex);  
    }  
}
```
