

**Uniwersytet Przyrodniczo-Humanistyczny**

**W Siedlcach**

**Kierunek Informatyka**

**Imię i nazwisko studenta:**

Kacper Trocewicz

**Nr indeksu:**

88639

**Temat Zadania:**

2. System obsługi hotelu Program, który umożliwia gromadzenie i przetwarzanie danych o miejscach oferowanych w hotelu i klientach.

- dane przechowywane w plikach (zapis i odczyt, plik/pliki),
- uwzględnij: zwykłego użytkownika, który może tylko przeglądać ofertę hotelu i dokonywać rezerwacji
- klienta hotelu, który może przeglądać ofertę hotelu, dokonywać rezerwacji, przedłużać pobyt i opłacać pobyt
- pracownika hotelu, który wprowadza dane o pokojach, cennik, potwierdza rezerwacje,
- cennik uwzględnia standard pokoju (piętro, położenie na piętrze), liczbę miejsc w pokoju,
- system umożliwia wyszukiwanie pokoi wg. ceny, piętra, liczby miejsc
- system pozwala sporządzić listę wolnych pokoi, listę rezerwacji
- system umożliwia wysyłanie informacji do klientów o ofertach i promocjach.

dr Anny Kołkowicz

Siedlce, 2022

## **Spis treści**

1	Specyfikacja problemu. Przyjęte założenia i ograniczenia:.....	3
2	Opis klas, algorytmów, zmiennych:.....	4
3	Diagram Klas w UML:.....	5
4	Kod Programu:.....	6
5	Przykładowe dane i wyniki(Zrzuty z opisem):.....	37
6	Instrukcja dla użytkownika(jak uruchomić, jakie dane wprowadzać):.....	41

## **1 Specyfikacja problemu. Przyjęte założenia i ograniczenia:**

### **Ograniczenia:**

- Żeby dodać pokój trzeba wpisać jego nazwę.
- W wyszukiwaniu pokoju najniższa cena za noc jest 50 zł a najwyższa nie ma ograniczeń cenowych.
- Po wybraniu użytkownika trzeba zrestartować program żeby zmienić na pracownika.
- Żeby mieć możliwość dostępu do zarządzania rezerwacjami trzeba mieć najpierw zarezerwowany pokój.
- Jak już osoba zarezerwowała pokój na daną datę inna osoba nie może zarezerwować tego samego pokoju na tę samą datę.
- Po zarezerwowaniu przez użytkownika pokoju stajemy się klientem i musimy zrestartować program żeby pokazała się nam możliwość Zarządzania rezerwacją.
- Pracownik Hotelu może dodać maksymalnie 50 pokoi.

### **Założenia:**

- Program ma za zadanie obsługę hotelu, ma w nim znajdować się klient i pracownik. Pracownik ma za zadanie zarządzać rezerwacjami lub dodawać pokoje, ma również uprawnienia do potwierdzania rezerwacji. Klient ma możliwość sprawdzania ofert wolnych pokoi i rezerwacji danego pokoju, lub przedłużenie jego pobytu.

## 2 Opis klas, algorytmów, zmiennych:

**BookingFrame.java**-Jest to klasa która tworzy nam okno danej rezerwacji w której możemy przedłużyć pobyt zobaczyć czy rezerwacja jest potwierdzona i czy została opłacona

**BookingsFrame.java**-Jest to klasa która tworzy nam okno w której jest tabela która pokazuje dane rezerwacji które wpisał klient.\_

**MainFrame.java**-jest to klasa która tworzy nam okno główne w którym wybieramy klienta lub pracownika.

**NewRoomFrame.java**-jest to klasa która tworzy nam okno w którym mamy możliwość dodania nowego pokoju przez pracownika.

**OfferListFrame.java**- jest to klasa która tworzy okno z tabelą, zawiera ona informacje o dostępności pokoi o danych które klient wpisał w wyszukiwaniu.

**OfferSpecificationFrame.java**- jest to klasa która tworzy nam okno, w którym mamy możliwość wyszukania danego pokoju jaki chcemy zarezerwować.

**Booking.java**-Jest to klasa zarządzająca rezerwacjami.

**Client.java**-Jest to klasa która zarządza klientem, dziedziczy ona login, name, contactnumber, email. Ta Klasa pozwala nam dostać się do takich opcji jak „Wyświetl ofertę”, „Zarządzaj rezerwacjami” ma uprawnienia również takie żeby opłacić rezerwacje jak i zrobić rezerwacje.

**Employee.java**- Jest to klasa która zarządza pracownikiem, dziedziczy ona login, name, contactnumber, email. Ta Klasa pozwala nam dostać się do takich opcji jak „Dodaj pokój”, „Zarządzaj rezerwacjami” ma uprawnienia również takie żeby potwierdzić rezerwacje klienta.

**Hotel.java**-Jest to klasa która posiada w sobie tablice dynamiczną pokoi, służy ona do przeszukiwania list dzięki którym można znaleźć wolny pokój.

**Room.java**-Jest to klasa która posiada metody do segregacji, po wpisanych danych przez klienta hotelu.

**User.java**-Jest to klasa abstrakcyjna w której mamy zmienne login, name, contactnumber.

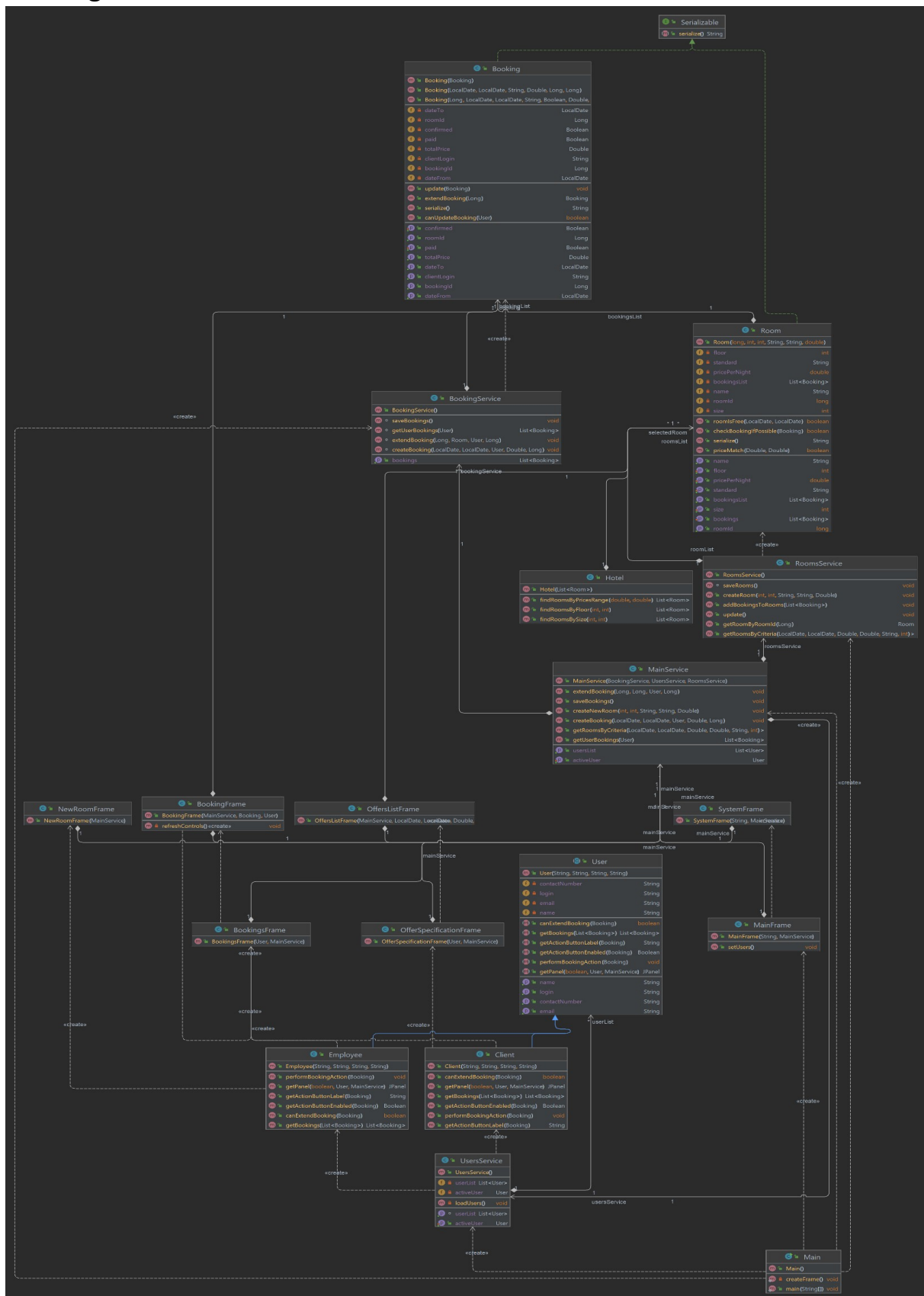
**BookingService.java**-Jest to klasa która odczytuje i zapisuje rezerwacje klientów jak i również przedłużania pobytu.

**MainService.java**-Jest to klasa która posiada tablice dynamiczną, używana jest do zapisywania i tworzenia pokoi i również ich wyszukiwania.

**RoomsService.java**-Jest to klasa która odczytuje i zapisuje wszystkie pokoje i ich dane.

**UsersService.java**-Jest to klasa która odczytuje użytkowników z pliku tekstowego i ustala czy jest pracownikiem czy klientem.

### 3 Diagram Klas w UML:



W E-mailu wysyłam zdjęcie diagramu aby było ono bardziej czytelne.

## 4 Kod Programu:

### BookingFrame.Java:

```
package Project.frames;

import Project.model.Booking;
import Project.model.User;
import Project.service.MainService;

import javax.swing.*.*;
import java.awt.*.*;
import java.time.LocalDate;

public class BookingFrame extends JFrame {
    private MainService mainService;
    private Booking booking;
    private LocalDate tempStayDate;

    private JPanel panel;

    private JButton extendBookingTimeButton;
    private JButton actionButton;
    private JButton finishButton;

    private JLabel labelFrom;
    private JLabel dateFromValueLabel;
    private JLabel labelTo;
    private JLabel labelToValue;
    private JLabel userLabel;
    private JLabel userNameLabel;
    private JLabel priceLabel;
    private JLabel priceValue;
    private JLabel labelPaid;
    private JCheckBox jCheckBoxPaid;
    private JLabel labelConfirmed;
    private JCheckBox jCheckBoxConfirmed;
    private JCheckBox extendStayCheckbox;
    private JSpinner extendStayField;

    public BookingFrame(MainService mainService, Booking booking,
User currentUser) throws HeadlessException {
        this.booking = booking;
        this.mainService = mainService;
        SpinnerNumberModel model = new SpinnerNumberModel(0, 0,
1000, 1);
        extendBookingTimeButton = new JButton("Przedłuż pobyt");
        actionButton = new
JButton(currentUser.getActionButtonLabel(booking));
```

```

actionButton.setEnabled(currentUser.getActionButtonEnabled(booking))
;

    tempStayDate = booking.getDateTo();

    panel = new JPanel();
    labelFrom = new JLabel("Od");
    dateFromValueLabel = new JLabel();
    labelTo = new JLabel("Do");
    labelToValue = new JLabel();
    userLabel = new JLabel("Użytkownik");
    userNameLabel = new JLabel();
    priceLabel = new JLabel("Cena");
    priceValue = new JLabel();
    labelPaid = new JLabel("Opłacono");
    jCheckBoxPaid = new JCheckBox();
    labelConfirmed = new JLabel("Zatwierdzono");
    jCheckBoxConfirmed = new JCheckBox();
    extendStayCheckbox = new JCheckBox();
    extendStayField = new JSpinner(model);
    finishButton = new JButton("Powrót");

    labelFrom.setBounds(20, 0, 100, 25);
    labelTo.setBounds(20, 50, 100, 25);
    userLabel.setBounds(20, 100, 100, 25);
    priceLabel.setBounds(20, 150, 100, 25);
    labelPaid.setBounds(20, 200, 100, 25);
    labelConfirmed.setBounds(20, 250, 100, 25);

    dateFromValueLabel.setBounds(120, 0, 200, 25);
    labelToValue.setBounds(120, 50, 100, 25);
    extendStayCheckbox.setBounds(230, 50, 150, 25);
    extendStayField.setBounds(400, 50, 50, 25);
    userNameLabel.setBounds(120, 100, 200, 25);
    priceValue.setBounds(120, 150, 200, 25);
    jCheckBoxPaid.setBounds(120, 200, 200, 25);
    jCheckBoxConfirmed.setBounds(120, 250, 200, 25);

    extendBookingTimeButton.setBounds(60, 350, 200, 25);
    actionButton.setBounds(270, 350, 200, 25);
    finishButton.setBounds(135, 400, 200, 25);

    panel.add(labelFrom);
    panel.add(dateFromValueLabel);
    panel.add(labelTo);
    panel.add(labelToValue);
    panel.add(userLabel);
    panel.add(userNameLabel);
    panel.add(priceLabel);
    panel.add(priceValue);
    panel.add(labelPaid);
    panel.add(jCheckBoxPaid);

```

```

        panel.add(labelConfirmed);
        panel.add(jCheckBoxConfirmed);
        panel.add(extendBookingTimeButton);
        panel.add(actionButton);
        panel.add(extendStayCheckbox);
        panel.add(extendStayField);
        panel.add(finishButton);

        actionButton.addActionListener(
            e -> {
                currentUser.performBookingAction(booking);
                mainService.saveBookings();
                refreshControls();
            }
        );

        finishButton.addActionListener(
            e -> {
                JFrame newRoomFrame = new
BookingsFrame(currentUser, mainService);
                newRoomFrame.setVisible(true);
                dispose();
            }
        );

        extendStayCheckbox.setVisible(currentUser.canExtendBooking(booking))
;

        extendStayField.setVisible(currentUser.canExtendBooking(booking));
        extendStayField.setEnabled(false);
        extendStayCheckbox.setText("Przedłużyć pobyt?");
        userNameLabel.setText(booking.getClientLogin());

        dateFromValueLabel.setText(booking.getDateFrom().toString());
        jCheckBoxPaid.setEnabled(false);
        jCheckBoxPaid.setSelected(booking.getPaid());
        jCheckBoxConfirmed.setEnabled(false);
        jCheckBoxConfirmed.setSelected(booking.getConfirmed());
        priceValue.setText(booking.getTotalPrice().toString() + "
zł");
        labelToValue.setText(booking.getDateTo().toString());

        extendBookingTimeButton.setVisible(currentUser.canExtendBooking(book
ing));

        extendBookingTimeButton.setEnabled(Integer
.parseInt(extendStayField.getValue().toString()) > 0);

```



```

        extendBookingTimeButton.addActionListener(
            e -> {

mainService.extendBooking(booking.getBookingId(),
booking.getRoomId(), currentUser,
Long.parseLong(extendStayField.getValue().toString()));
                extendBookingTimeButton.setEnabled(false);
                extendStayCheckbox.setSelected(false);
                extendStayField.setValue(0);

labelToValue.setText(booking.getDateTo().toString());
            }
        );

        extendStayCheckbox.addActionListener(
            e ->
extendStayField.setEnabled(extendStayCheckbox.isSelected())
        );
        extendStayField.addChangeListener(
            e -> {
                tempStayDate =
booking.getDateTo().plusDays(Long
.parseLong(extendStayField.getValue().toString()));
                labelToValue.setText(tempStayDate.toString());

extendBookingTimeButton.setEnabled(Integer
.parseInt(extendStayField.getValue().toString()) > 0);
            });

        panel.setLayout(null);

        add(panel);
        setTitle("Rezerwacja nr " + booking.getBookingId());
        setBounds(500, 200, 500, 500);

    }

    private void refreshControls() {
        jCheckBoxConfirmed.setSelected(booking.getConfirmed());
        jCheckBoxPaid.setSelected(booking.getPaid());
    }
}

```

### **BookingsFrame.Java:**

```

package Project.frames;

import Project.model.Booking;
import Project.model.User;
import Project.service.MainService;

```

```

import javax.swing.*;
import javax.swing.table.TableColumnModel;
import java.awt.*;
import java.util.List;

public class BookingsFrame extends JFrame {
    private MainService mainService;

    private JTable clientBookingsTable;
    private JButton manageBookingButton;
    private JScrollPane scrollPane;
    private JPanel bookingsPanel;

    public BookingsFrame(User user, MainService mainService) {
        this.mainService = mainService;
        List<Booking> userBookings =
mainService.getUserBookings(user);
        bookingsPanel = new JPanel();
        scrollPane = new JScrollPane();
        manageBookingButton = new JButton("Zarządzaj rezerwacją");
        manageBookingButton.setEnabled(false);
        manageBookingButton.addActionListener(e -> {
            JFrame bookingFrame = new BookingFrame(mainService,
userBookings.get(clientBookingsTable.getSelectedRow()), user);
            bookingFrame.setVisible(true);
            dispose();
        });
        setBounds(500, 200, 1000, 500);
        String[] columns = {"Od", "Do", "Klient", "Numer Pokoju",
"Cena", "Potwierdzone", "Opłacona"};
        String[][] data = new String[userBookings.size()]
[columns.length];
        for (int i = 0 ; i < userBookings.size(); i++){
            data[i][0] =
userBookings.get(i).getDateFrom().toString();
            data[i][1] = userBookings.get(i).getDateTo().toString();
            data[i][2] = userBookings.get(i).getClientLogin();
            data[i][3] = userBookings.get(i).getRoomId().toString();
            data[i][4] =
userBookings.get(i).getTotalPrice().toString();
            data[i][5] =
userBookings.get(i).getConfirmed().toString();
            data[i][6] = userBookings.get(i).getPaid().toString();
        }
        clientBookingsTable = new JTable(data, columns);

        clientBookingsTable.setAutoResizeMode(JTable
.AUTO_RESIZE_ALL_COLUMNS);
        TableColumnModel model =
clientBookingsTable.getColumnModel();

```

```

        model.getColumn(0).setWidth(100);

clientBookingsTable.getSelectionModel().addListSelectionListener(event -> {
    if (clientBookingsTable.getSelectedRow() > -1) {
        manageBookingButton.setEnabled(true);
    }
});
scrollPane.setViewportView(clientBookingsTable);
scrollPane.getViewport().setPreferredSize(new
Dimension(1000, 100 * userBookings.size()));
bookingsPanel.add(scrollPane);
bookingsPanel.add(manageBookingButton);
add(bookingsPanel);
}
}

```

### **MainFrame.Java:**

```

package Project.frames;

import Project.service.MainService;

import javax.swing.*.*;
import javax.swing.JPanel;
import java.awt.*.*;

public class MainFrame extends JFrame{

    private MainService mainService;

    private JPanel loginPanel;
    private JLabel selectUserLabel;
    private JButton confirmLogin;
    private JComboBox<String> usersList;

    public void setUsers(){
        mainService getUsersList().forEach(it ->
usersList.addItem(it.getLogin()));
    }

    public MainFrame(String title, MainService mainService) throws
HeadlessException {
        super(title);
        this.mainService = mainService;

        confirmLogin = new JButton("Zatwierdź");
        usersList = new JComboBox<>();
        usersList.setBounds(0, 100, 400, 50);
    }
}

```

```

        confirmLogin.setBounds(420,100,200,50);
        selectUserLabel = new JLabel("Wybierz swojego
użytkownika:");
        selectUserLabel.setBounds(250, 0, 250, 150);

        loginPanel = new JPanel();
        loginPanel.add(selectUserLabel);
        loginPanel.add(usersList);
        loginPanel.add(confirmLogin);

        add(loginPanel);
        setBounds(300, 90, 650, 100);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setResizable(false);
        setUsers();
        confirmLogin.addActionListener(
            e -> {

mainService.setActiveUser(usersList.getItemAt(usersList.getSelectedI
ndex()));

System.out.println(mainService.getActiveUser().getLogin());
                dispose();
                JFrame systemFrame = new JFrame("System",
mainService);

systemFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                systemFrame.setVisible(true);
            }
        );
    }
}

```

### **NewRoomFrame.Java:**

```

package Project.frames;

import Project.service.MainService;

import javax.swing.*.*;

public class NewRoomFrame extends JFrame {

    private MainService mainService;

    private JPanel roomPanel;

    private JLabel guestsLabel;
    private JSpinner spinnerGuests;
    private JLabel floorLabel;

```

```

private JSpinner spinnerFloor;
private JLabel standardLabel;
private JComboBox<String> standardComboBox;
private JLabel nameLabel;
private JTextField nameField;
private JLabel priceLabel;
private JSpinner spinnerPrice;

private JButton saveButton;

public NewRoomFrame(MainService mainService) {
    this.mainService = mainService;

    guestsLabel = new JLabel("Liczba gości:");
    floorLabel = new JLabel("Piętro:");
    standardLabel = new JLabel("Standard:");
    priceLabel = new JLabel("Cena za noc:");
    nameLabel = new JLabel("Nazwa:");
    roomPanel = new JPanel();

    SpinnerNumberModel guestsSpinnerModel = new
SpinnerNumberModel(2, 2, 10, 1);
    SpinnerNumberModel floorSpinnerModel = new
SpinnerNumberModel(0, 0, 50, 1);
    SpinnerNumberModel priceSpinnerModel = new
SpinnerNumberModel(50.0, 50.0, 10000.0, 25.0);
    spinnerGuests = new JSpinner(guestsSpinnerModel);
    spinnerFloor = new JSpinner(floorSpinnerModel);
    spinnerPrice = new JSpinner(priceSpinnerModel);
    nameField = new JTextField();

    standardComboBox = new JComboBox<>();
    standardComboBox.addItem("Budget");
    standardComboBox.addItem("Normal");
    standardComboBox.addItem("Premium");

    guestsLabel.setBounds(10, 10, 100, 25);
    floorLabel.setBounds(10, 50, 100, 25);
    standardLabel.setBounds(10, 90, 100, 25);
    priceLabel.setBounds(10, 130, 100, 25);
    nameLabel.setBounds(10, 170, 100, 25);

    spinnerGuests.setBounds(150, 10, 50, 25);
    spinnerFloor.setBounds(150, 50, 50, 25);
    standardComboBox.setBounds(150, 90, 90, 25);
    spinnerPrice.setBounds(150, 130, 50, 25);
    nameField.setBounds(150, 170, 200, 25);

    saveButton = new JButton("Zapisz");
    saveButton.setBounds(100, 270, 200, 25);

    saveButton.addActionListener(

```

```

        e -> {
            mainService.createNewRoom(

Integer.parseInt(spinnerGuests.getValue().toString()),

Integer.parseInt(spinnerFloor.getValue().toString()),

standardComboBox.getItemAt(standardComboBox.getSelectedIndex()),
                        nameField.getText(),

Double.parseDouble(spinnerPrice.getValue().toString())
                    );
            dispose();
        }
    };

    roomPanel.add(guestsLabel);
    roomPanel.add(floorLabel);
    roomPanel.add(standardLabel);
    roomPanel.add(priceLabel);
    roomPanel.add(nameLabel);

    roomPanel.add(spinnerGuests);
    roomPanel.add(spinnerFloor);
    roomPanel.add(spinnerPrice);
    roomPanel.add(standardComboBox);
    roomPanel.add(nameField);
    roomPanel.add(saveButton);
    roomPanel.setLayout(null);

    add(roomPanel);
    setTitle("Dodaj pokój");
    setBounds(300, 90, 400, 400);
    setResizable(false);
}

}

```

### **OfferListFrame.Java:**

```

package Project.frames;

import Project.model.Room;
import Project.service.MainService;

import javax.swing.*.*;
import javax.swing.table.TableColumnModel;
import java.awt.*.*;
import java.time.LocalDate;
import java.util.List;

```

```

import static java.time.temporal.ChronoUnit.DAYS;

public class OffersListFrame extends JFrame {
    private MainService mainService;

    private JTable clientBookingsTable;
    private JButton createBookingButton;
    private JScrollPane scrollPane;
    private JPanel bookingsPanel;

    private LocalDate dateFrom;
    private LocalDate dateTo;
    private Double priceFrom;
    private Double priceTo;
    private String standard;

    private Room selectedRoom;

    public OffersListFrame(MainService mainService, LocalDate
dateFrom, LocalDate dateTo, Double priceFrom, Double priceTo, String
standard, int guests) {
        this.mainService = mainService;

        System.out.println("Parametry wyszukiwania: od: " + dateFrom
+ " do: " + dateTo + " cenaOd " + priceFrom + " cena do " + priceTo
+ " standard:" + standard + " goście: " + guests);
        List<Room> roomsByCriteria =
mainService.getRoomsByCriteria(dateFrom, dateTo, priceFrom, priceTo,
standard, guests);
        bookingsPanel = new JPanel();
        scrollPane = new JScrollPane();
        createBookingButton = new JButton("Zarezerwuj");
        createBookingButton.setEnabled(false);
        int daysDifference = (int) DAYS.between(dateFrom, dateTo);
        if(daysDifference == 0)
            daysDifference = 1;
        int finalDaysDifference = daysDifference;
        createBookingButton.addActionListener(e -> {
            double pricePerNight = selectedRoom.getPricePerNight();
            mainService.createBooking(dateFrom, dateTo,
mainService.getActiveUser(), pricePerNight * finalDaysDifference,
selectedRoom.getRoomId());
            dispose();
        });
        setBounds(500, 200, 1000, 500);
        String[] columns = {"Nazwa", "Standard", "Piętro",
"Miejsca", "Cena za noc", "Cena za pobyt"};
        String[][] data = new String[roomsByCriteria.size()]
[columns.length];
        for (int i = 0 ; i < roomsByCriteria.size(); i++){

```

```

        data[i][0] = roomsByCriteria.get(i).getName();
        data[i][1] = roomsByCriteria.get(i).getStandard();
        data[i][2] =
String.valueOf(roomsByCriteria.get(i).getFloor());
        data[i][3] =
String.valueOf(roomsByCriteria.get(i).getSize());
        data[i][4] =
String.valueOf(roomsByCriteria.get(i).getPricePerNight());
        data[i][5] =
String.valueOf(roomsByCriteria.get(i).getPricePerNight() *
(daysDifference));
    }
    clientBookingsTable = new JTable(data, columns);

clientBookingsTable.setAutoResizeMode(JTable
.AUTO_RESIZE_ALL_COLUMNS);
    TableColumnModel model =
clientBookingsTable.getColumnModel();
    model.getColumn(0).setWidth(100);

clientBookingsTable.getSelectionModel().addListSelectionListener(event -> {
    if (clientBookingsTable.getSelectedRow() > -1) {
        createBookingButton.setEnabled(true);
        selectedRoom =
roomsByCriteria.get(clientBookingsTable.getSelectedRow());
    }
});
    scrollPane.setViewportView(clientBookingsTable);
    scrollPane.getViewPort().setPreferredSize(new
Dimension(1000,100 * roomsByCriteria.size()));
    bookingsPanel.add(scrollPane);
    bookingsPanel.add(createBookingButton);
    add(bookingsPanel);
}
}

```

### **OfferSpecificationFrame.Java:**

```

package Project.frames;

import Project.model.User;
import Project.service.MainService;

import javax.swing.*;
import java.text.SimpleDateFormat;
import java.time.Instant;
import java.time.ZoneId;
import java.util.Calendar;
import java.util.Date;

public class OfferSpecificationFrame extends JFrame {
    private MainService mainService;

```



```

    private Date dateFrom =
Date.from(Instant.ofEpochMilli(System.currentTimeMillis()));
    private Date dateTo =
Date.from(Instant.ofEpochMilli(System.currentTimeMillis()));
    private JPanel offerSpecificationPanel;
    private JButton searchButton;

    private JSpinner dateFromSpinner;
    private JLabel dateFromLabel;
    private JSpinner dateToSpinner;
    private JLabel dateToLabel;

    private JSpinner priceMinSpinner;
    private JLabel priceMinLabel;
    private JSpinner priceMaxSpinner;
    private JLabel priceMaxLabel;
    private JSpinner guestsSpinner;
    private JLabel guestLabel;

    private JLabel standardLabel;
    private JComboBox<String> standardComboBox;

    private String standard = "Budget";
    private Double priceMin = 50.0;
    private Double priceMax = 75.0;
    private Double guests = 2.0;

    public OfferSpecificationFrame(User user, MainService mainService) {
        this.mainService = mainService;
        offerSpecificationPanel = new JPanel();
        setBounds(500, 200, 240, 350);
        SimpleDateFormat dateFormat = new SimpleDateFormat("dd/MM/yyyy");
        searchButton = new JButton("Wyszukaj");

        dateFromLabel = new JLabel("Data od:");
        dateToLabel = new JLabel("Data do:");
        priceMinLabel = new JLabel("Cena od:");
        priceMaxLabel = new JLabel("Cena do:");
        standardLabel = new JLabel("Standard:");
        guestLabel = new JLabel("Goście:");

        dateFromLabel.setBounds(10, 10, 70, 25);
        dateToLabel.setBounds(10, 50, 70, 25);
        priceMinLabel.setBounds(10, 90, 70, 25);
        priceMaxLabel.setBounds(10, 130, 70, 25);
        standardLabel.setBounds(10, 170, 70, 25);
        guestLabel.setBounds(10, 210, 70, 25);

        SpinnerNumberModel priceMinSpinnerModel = new
SpinnerNumberModel(50.0, 50.0, 10000.0, 25.0);
        SpinnerNumberModel priceMaxSpinnerModel = new
SpinnerNumberModel(75.0, 75.0, 10000.0, 25.0);
        SpinnerNumberModel guestsSpinnerModel = new SpinnerNumberModel(1.0,
1.0, 20.0, 1.0);
        guestsSpinner = new JSpinner(guestsSpinnerModel);
        priceMinSpinner = new JSpinner(priceMinSpinnerModel);

```

```

priceMaxSpinner = new JSpinner(priceMaxSpinnerModel);

standardComboBox = new JComboBox<>();
standardComboBox.addItem("Budget");
standardComboBox.addItem("Normal");
standardComboBox.addItem("Premium");

SpinnerDateModel dateFromSpinnerModel = new
SpinnerDateModel(dateFrom,
    Date.from(Instant.ofEpochSecond(1679330112)),
    Date.from(Instant.ofEpochSecond(1703954112)),
    Calendar.DAY_OF_YEAR);
dateFromSpinner = new JSpinner(dateFromSpinnerModel);
JSpinner.DateEditor dateFromEditor = new
JSpinner.DateEditor(dateFromSpinner, "dd.MM.yyyy");
dateFromEditor.getTextField().setEditable(false);
dateFromSpinner.setEditor(dateFromEditor);

SpinnerDateModel dateToSpinnerModel = new SpinnerDateModel(dateTo,
    Date.from(Instant.ofEpochSecond(1687090800)),
    Date.from(Instant.ofEpochSecond(1718713200)),
    Calendar.DAY_OF_YEAR);
dateToSpinner = new JSpinner(dateToSpinnerModel);
JSpinner.DateEditor dateToEditor = new
JSpinner.DateEditor(dateToSpinner, "dd.MM.yyyy");
dateToEditor.getTextField().setEditable(false);
dateToSpinner.setEditor(dateToEditor);

dateFromSpinner.setBounds(90, 10, 100, 25);
dateToSpinner.setBounds(90, 50, 100, 25);
priceMinSpinner.setBounds(90, 90, 100, 25);
priceMaxSpinner.setBounds(90, 130, 100, 25);
standardComboBox.setBounds(90, 170, 100, 25);
guestsSpinner.setBounds(90, 210, 100, 25);

searchButton.setBounds(60, 250, 100, 25);

priceMinSpinner.addChangeListener(
    e -> priceMin =
Double.valueOf(priceMinSpinner.getValue().toString())
);

priceMaxSpinner.addChangeListener(
    e -> priceMax =
Double.valueOf(priceMaxSpinner.getValue().toString())
);

guestsSpinner.addChangeListener(
    e -> guests = (Double) guestsSpinner.getValue()
);

dateFromSpinner.addChangeListener(
    e -> {
        dateFrom = (Date) dateFromSpinner.getValue();
        if (dateTo.before(dateFrom)) {
            dateToEditor.getModel().setValue(dateFrom);
        }
    }
);

```

```

        }
    };

    dateToSpinner.addChangeListener(
        e -> {
            dateTo = (Date) dateToSpinner.getValue();
            if (dateFrom.after(dateTo)) {
                dateFromEditor.getModel().setValue(dateTo);
            }
        }
    );

    standardComboBox.addActionListener(
        e -> standard =
standardComboBox.getItemAt(standardComboBox.getSelectedIndex())
    );

    offerSpecificationPanel.add(dateFromLabel);
    offerSpecificationPanel.add(dateToLabel);
    offerSpecificationPanel.add(dateFromSpinner);
    offerSpecificationPanel.add(dateToSpinner);
    offerSpecificationPanel.add(priceMaxLabel);
    offerSpecificationPanel.add(priceMinLabel);
    offerSpecificationPanel.add(priceMaxSpinner);
    offerSpecificationPanel.add(priceMinSpinner);
    offerSpecificationPanel.add(standardComboBox);
    offerSpecificationPanel.add(standardLabel);
    offerSpecificationPanel.add(searchButton);
    offerSpecificationPanel.add(guestLabel);
    offerSpecificationPanel.add(guestsSpinner);

    offerSpecificationPanel.setLayout(null);

    searchButton.addActionListener(
        e -> {
            JFrame offerListFrame = new
OffersListFrame(mainService,
dateFrom.toInstant().atZone(ZoneId.systemDefault()).toLocalDate(),
dateTo.toInstant().atZone(ZoneId.systemDefault()).toLocalDate(), priceMin,
priceMax, standard, guests.intValue());
            offerListFrame.setVisible(true);
            dispose();
        }
    );

    add(offerSpecificationPanel);
    setTitle("Specyfikacja");
}
}

```

### **SystemFrame.Java**

```
package Project.frames;

import Project.model.Client;
import Project.model.User;
import Project.service.MainService;

import javax.swing.*.*;
import java.awt.*.*;

public class SystemFrame extends JFrame {

    private MainService mainService;
    private JLabel userLabel;
    private JPanel systemPanel;

    public SystemFrame(String title, MainService mainService) throws
HeadlessException {
        super(title);
        this.mainService = mainService;
        setBounds(300, 90, 650, 100);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setResizable(false);
        systemPanel = new JPanel();
        User user = mainService.getActiveUser();
        userLabel = new JLabel(user.getName());
        systemPanel.add(userLabel);
        add(systemPanel);
        boolean userHasBookings = false;
        if(user instanceof Client)
            userHasBookings = !
mainService.getUserBookings(user).isEmpty();
        JPanel returnspanel = user.getPanel(userHasBookings, user,
mainService);
        JButton returnButton = new JButton("Powrót");
        returnButton.addActionListener(
            e -> {
                JFrame mainFrame = new MainFrame(title,
mainService);
                mainFrame.setVisible(true);
                dispose();
            });
        returnButton.setBounds(10,10,250,25);
        returnspanel.add(returnButton);
        systemPanel.add(returnspanel);
    }
}
```

```
}
```

```
}
```

### **Booking.Java:**

```
package Project.model;
```

```
import java.time.LocalDate;
```

```
public class Booking implements Serializable {  
    static Long bookings = 0L;
```

```
    private Long bookingId;  
    private Long roomId;  
    private LocalDate dateFrom;  
    private LocalDate dateTo;  
    private String clientLogin;  
    private Boolean confirmed;  
    private Double totalPrice;  
    private Boolean paid;
```

```
    public Booking(LocalDate dateFrom, LocalDate dateTo, String  
clientLogin, Double totalPrice, Long roomId, Long bookingId) {  
        this.dateFrom = dateFrom;  
        this.dateTo = dateTo;  
        this.clientLogin = clientLogin;  
        this.totalPrice = totalPrice;  
        this.roomId = roomId;  
        this.confirmed = false;  
        this.paid = false;  
        this.bookingId = bookingId;  
    }
```

```
    public Booking(Booking other){  
        this.dateFrom = other.dateFrom;  
        this.dateTo = other.dateTo;  
        this.clientLogin = other.clientLogin;  
        this.confirmed = other.confirmed;  
        this.totalPrice = other.totalPrice;  
        this.paid = other.paid;  
        this.bookingId = other.bookingId;  
        this.roomId = other.roomId;  
    }
```

```
    public Booking(Long bookingId, LocalDate dateFrom, LocalDate  
dateTo, String clientLogin, Boolean confirmed, Double totalPrice,  
Boolean paid, Long roomId) {  
        this.dateFrom = dateFrom;  
        this.dateTo = dateTo;  
        this.clientLogin = clientLogin;  
        this.confirmed = confirmed;
```

```

        this.totalPrice = totalPrice;
        this.paid = paid;
        this.bookingId = bookingId;
        this.roomId = roomId;
        if(bookingId >= bookings)
            bookings = bookingId + 1;
    }

    public Long getRoomId() {
        return roomId;
    }

    public boolean canUpdateBooking(User updatingUser){

        return updatingUser.getLogin().equals(getClientLogin());
    }

    public LocalDate getDateFrom() {
        return dateFrom;
    }

    public LocalDate getDateTo() {
        return dateTo;
    }

    public String getClientLogin() {
        return clientLogin;
    }

    public Boolean getConfirmed() {
        return confirmed;
    }

    public Double getTotalPrice() {
        return totalPrice;
    }

    public Boolean getPaid() {
        return paid;
    }

    public Long getBookingId() {
        return bookingId;
    }

    public Booking extendBooking(Long days){
        Booking tmpBooking = new Booking(this);
        tmpBooking.dateTo = dateTo.plusDays(days);
        return tmpBooking;
    }

    public void setPaid(Boolean paid) {

```

```

        this.paid = paid;
    }

    public void setConfirmed(Boolean confirmed) {
        this.confirmed = confirmed;
    }

    public void update(Booking booking) {
        this.dateTo = booking.getDateTo();
        this.confirmed = false;
        this.paid = false;
    }

    @Override
    public String serialize() {
        return "" + bookingId + "," + clientLogin + "," + roomId +
        "," + dateFrom.getDayOfMonth() + "," + dateFrom.getMonthValue() + "," +
        dateFrom.getYear()
            + "," + dateTo.getDayOfMonth() + "," +
        dateTo.getMonthValue() + "," + dateTo.getYear() + "," + totalPrice +
        "," + confirmed + "," + paid;
    }
}

```

#### **Client.Java:**

```

package Project.model;

import Project.Main;
import Project.frames.BookingsFrame;
import Project.frames.MainFrame;
import Project.frames.OfferSpecificationFrame;
import Project.service.MainService;

import javax.swing.*;
import java.util.List;
import java.util.stream.Collectors;

public class Client extends User {

    public Client(String login, String name, String contactNumber,
String email) {
        super(login, name, contactNumber, email);
    }

    @Override
    public JPanel getPanel(boolean hasReservations, User activeUser,
MainService mainService) {
        JPanel panel = new JPanel();

        JButton offerButton = new JButton("Wyświetl ofertę");
        offerButton.addActionListener(

```

```

        e -> {
            JFrame offerFrame = new
OfferSpecificationFrame(activeUser, mainService);
            offerFrame.setVisible(true);
        }
    );

    panel.add(offerButton);
    if(hasReservations) {
        JButton bookingsManagement = new JButton("Zarządzaj
rezerwacjami");
        bookingsManagement.addActionListener(
            e -> {
                JFrame clientFrame = new
BookingsFrame(activeUser, mainService);
                clientFrame.setVisible(true);
            }
        );
        panel.add(bookingsManagement);
    }
    return panel;
}

@Override
public boolean canExtendBooking(Booking booking) {
    return booking.getClientLogin().equals(getLogin());
}

@Override
public String getActionButtonLabel(Booking booking) {
    if(booking.getClientLogin().equals(getLogin()))
        return "Opłacić";
    else return "Zarezerwuj";
}

@Override
public Boolean getActionButtonEnabled(Booking booking) {
    if(booking.getClientLogin().equals(getLogin()))
        return !booking.getPaid();
    else return true;
}

@Override
public List<Booking> getBookings(List<Booking> bookings) {
    return bookings.stream().filter(it ->
it.getClientLogin().equals(getLogin())).collect(Collectors
.toList());
}

@Override
public void performBookingAction(Booking booking) {
    booking.setPaid(true);
}

```



```
    }  
  
}
```

### **Employee.Java:**

```
package Project.model;  
  
import Project.frames.BookingsFrame;  
import Project.frames.NewRoomFrame;  
import Project.service.MainService;  
  
import javax.swing.*.*;  
import java.util.List;  
  
public class Employee extends User {  
  
    public Employee(String login, String name, String contactNumber,  
String email) {  
        super(login, name, contactNumber, email);  
    }  
  
    @Override  
    public JPanel getPanel(boolean hasReservations, User activeUser,  
MainService mainService) {  
        JPanel panel = new JPanel();  
        JButton addRoomButton = new JButton("Dodaj pokój");  
        panel.add(addRoomButton);  
        addRoomButton.addActionListener(  
            e -> {  
                JFrame newRoomFrame = new  
NewRoomFrame(mainService);  
                newRoomFrame.setVisible(true);  
            }  
        );  
        JButton manageBookingsButton = new JButton("Zarządzaj  
rezerwacjami");  
        panel.add(manageBookingsButton);  
        manageBookingsButton.addActionListener(  
            e -> {  
                JFrame newRoomFrame = new  
BookingsFrame(activeUser, mainService);  
                newRoomFrame.setVisible(true);  
            }  
        );  
        panel.add(manageBookingsButton);  
        return panel;  
    }  
  
    @Override  
    public boolean canExtendBooking(Booking booking) {  
        return false;  
    }  
}
```

```

@Override
public String getActionButtonLabel(Booking booking) {
    return "Potwierdź";
}

@Override
public Boolean getActionButtonEnabled(Booking booking) {
    return !booking.getConfirmed();
}

@Override
public List<Booking> getBookings(List<Booking> bookings) {
    return bookings;
}

@Override
public void performBookingAction(Booking booking) {
    booking.setConfirmed(true);
}
}

```

### **Hotel.Java:**

```

package Project.model;

import java.util.List;
import java.util.stream.Collectors;

public class Hotel {

    private List <Room> roomsList;

    public Hotel(List<Room> roomsList) {
        this.roomsList = roomsList;
    }

    public List<Room> findRoomsByPricesRange(double minPrice, double
maxPrice){
        return roomsList.stream().filter(
            it -> it.getPricePerNight() >= minPrice &&
it.getPricePerNight() <= maxPrice
        ).collect(Collectors.toList());
    }

    public List<Room> findRoomsByFloor(int minFloor, int maxFloor){
        return roomsList.stream().filter(
            it -> it.getFloor() >= minFloor && it.getFloor() <=
maxFloor
        ).collect(Collectors.toList());
    }

    public List<Room> findRoomsBySize(int minSize, int maxSize){

```

```

        return roomsList.stream().filter(
            it -> it.getSize() >= minSize && it.getSize() <=
maxSize
        ).collect(Collectors.toList());
    }

}

```

### **Room.Java:**

```

package Project.model;

import java.time.LocalDate;
import java.util.ArrayList;
import java.util.List;

public class Room implements Serializable {
    private long roomId;
    private int size;
    private int floor;
    private String standard;
    private String name;
    private double pricePerNight;
    private List<Booking> bookingsList = new ArrayList<>();

    public long getRoomId() {
        return roomId;
    }

    public String getName() {
        return name;
    }

    public List<Booking> getBookingsList() {
        return bookingsList;
    }

    public Room(long roomId, int size, int floor, String standard,
String name, double pricePerNight) {
        this.size = size;
        this.floor = floor;
        this.standard = standard;
        this.name = name;
        this.pricePerNight = pricePerNight;
        this.roomId = roomId;
    }

    public boolean checkBookingIfPossible(Booking booking){
        return bookingsList.stream().noneMatch(
            book ->
(booking.getDateFrom().isAfter(book.getDateFrom())) &&
booking.getDateFrom().isBefore(book.getDateTo()) ||

```

```

booking.getDateTo().isAfter(book.getDateFrom()) &&
booking.getDateTo().isBefore(book.getDateTo()) && !
booking.getBookingId().equals(book.getBookingId()));
    }

    public boolean roomIsFree(LocalDate dateFrom, LocalDate dateTo){
        return bookingsList.stream().noneMatch(book ->
dateFrom.isAfter(book.getDateFrom()) &&
dateFrom.isBefore(book.getDateTo()) ||
            dateTo.isAfter(book.getDateFrom()) &&
dateTo.isBefore(book.getDateTo()) ||
dateFrom.isEqual(book.getDateFrom()) ||
dateTo.isEqual(book.getDateTo()));
    }

    public boolean priceMatch(Double priceMin, Double priceMax){
        System.out.println("Cena pokoju " + pricePerNight);
        return pricePerNight >= priceMin && pricePerNight <=
priceMax;
    }

    public int getSize() {
        return size;
    }

    public void setSize(int size) {
        this.size = size;
    }

    public int getFloor() {
        return floor;
    }

    public void setFloor(int floor) {
        this.floor = floor;
    }

    public String getStandard() {
        return standard;
    }

    public void setStandard(String standard) {
        this.standard = standard;
    }

    public double getPricePerNight() {
        return pricePerNight;
    }

    public void setPricePerNight(int pricePerNight) {

```

```

        this.pricePerNight = pricePerNight;
    }

    @Override
    public String serialize() {
        return "" + roomId + "," + size + "," + floor + "," +
standard + "," + name + "," + pricePerNight;
    }

    public void setBookings(List<Booking> roomBookings) {
        this.bookingsList = roomBookings;
    }
}

```

### **Serializable.Java:**

```

package Project.model;

public interface Serializable {
    String serialize();
}

```

### **User.Java:**

```

package Project.model;

import Project.service.MainService;

import javax.swing.*.*;
import java.util.List;

public abstract class User {
    private String login;
    private String name;
    private String contactNumber;
    private String email;

    public User(String login, String name, String contactNumber,
String email) {
        this.login = login;
        this.name = name;
        this.contactNumber = contactNumber;
        this.email = email;
    }

    public String getLogin() {
        return login;
    }

    public String getName() {
        return name;
    }
}

```

```

    public String getContactNumber() {
        return contactNumber;
    }

    public String getEmail() {
        return email;
    }

    public abstract JPanel getPanel(boolean hasReservations, User
activeUser, MainService mainService);
    public abstract boolean canExtendBooking(Booking booking);
    public abstract String getActionButtonLabel(Booking booking);
    public abstract Boolean getActionButtonEnabled(Booking booking);
    public abstract List<Booking> getBookings(List<Booking>
bookings);
    public abstract void performBookingAction(Booking booking);
}

```

### **Bookingservice.Java:**

```

package Project.service;

import Project.model.Booking;
import Project.model.Room;
import Project.model.User;

import java.io.*;
import java.time.LocalDate;
import java.util.ArrayList;
import java.util.List;
import java.util.NoSuchElementException;
import java.util.stream.Collectors;

public class BookingService {
    private List<Booking> bookingList;

    public BookingService() {
        bookingList = new ArrayList<>();
        BufferedReader reader;
        try {
            reader = new BufferedReader(new FileReader(
                "bookings"));
            String line = reader.readLine();
            while (line != null) {
                String[] data = line.split(",");
                LocalDate dateFrom =
LocalDate.of(Integer.parseInt(data[5]), Integer.parseInt(data[4]),
Integer.parseInt(data[3]));
                LocalDate dateTo =
LocalDate.of(Integer.parseInt(data[8]), Integer.parseInt(data[7]),
Integer.parseInt(data[6]));
                bookingList.add(new Booking(Long.parseLong(data[0]),
dateFrom, dateTo, data[1], Boolean.parseBoolean(data[10]),

```

```

Double.parseDouble(data[9]), Boolean.parseBoolean(data[11]),
Long.parseLong(data[2])));
        line = reader.readLine();
    }
    reader.close();
} catch (IOException e) {
    e.printStackTrace();
}

}

List<Booking> getUserBookings(User user) {
    return user.getBookings(bookingList);
}

void extendBooking(Long bookingId, Room room, User user, Long
daysExtended){
    Booking booking = bookingList.stream().filter(book ->
book.getBookingId().equals(bookingId)).findFirst().get();
    Booking bookingTmp = booking.extendBooking(daysExtended);
    if(room.checkBookingIfPossible(bookingTmp) &&
        booking.canUpdateBooking(user)) {
        booking.update(bookingTmp);
    } else throw new RuntimeException("Nie można przedłużyć
pobytu!");
    saveBookings();
}

void createBooking(LocalDate dateFrom, LocalDate dateTo, User
user, Double totalPrice, Long roomId){

    Long bookingId=
bookingList.stream().map(Booking::getBookingId).max(Long::compareTo)
.orElse(0L) + 1;;
    bookingList.add(new Booking(dateFrom, dateTo,
user.getLogin(), totalPrice, roomId, bookingId));
    saveBookings();
}

public List<Booking> getBookings() {
    return bookingList;
}

void saveBookings() {
    String bookings =
bookingList.stream().map(Booking::serialize).collect(Collectors
.joining("\n"));
    try {
        BufferedWriter writer = new BufferedWriter(new
FileWriter("bookings"));
        writer.write(bookings);
        writer.close();
    }
}

```

```

        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}

```

### **MainService.Java:**

```

package Project.service;

import Project.model.Booking;
import Project.model.Room;
import Project.model.User;

import javax.swing.*;
import java.time.LocalDate;
import java.util.List;

public class MainService {

    private BookingService bookingService;
    private UsersService usersService;
    private RoomsService roomsService;

    public MainService(BookingService bookingService, UsersService
usersService, RoomsService roomsService) {
        this.bookingService = bookingService;
        this.usersService = usersService;
        this.roomsService = roomsService;

roomsService.addBookingsToRooms(bookingService.getBookings());

    }

    public List<User> getUsersList(){
        return usersService.getUserList();
    }

    public void createBooking(LocalDate dateFrom, LocalDate dateTo,
User user, Double totalPrice, Long roomId){
        bookingService.createBooking(dateFrom, dateTo, user,
totalPrice, roomId);
    }

    public void extendBooking(Long bookingId, Long roomId, User
user, Long daysExtended){
        Room room = roomsService.getRoomByRoomId(roomId);
        bookingService.extendBooking(bookingId, room, user,
daysExtended);
    }
}

```



```

        public List<Room> getRoomsByCriteria(LocalDate dateFrom,
        LocalDate dateTo, Double priceMin, Double priceMax, String standard,
        int guests){
            return roomsService.getRoomsByCriteria(dateFrom, dateTo,
        priceMin, priceMax, standard, guests);
        }

        public List<Booking> getUserBookings(User user) {
            return bookingService.getUserBookings(user);
        }

        public void setActiveUser(String userLogin) {
            usersService.setActiveUser(userLogin);
        }

        public void createNewRoom(int guests, int floor, String
        standard, String name, Double price){
            if(roomsService.getRoomSize()>49){
                JOptionPane.showMessageDialog(null,"Za duzo pokoi!");
            }else
            if(name.equals("")){
                JOptionPane.showMessageDialog(null,"Podaj Nazwe!");
            }else {
                roomsService.createRoom(guests, floor, standard, name,
        price);
            }
        }

        public void saveBookings(){
            bookingService.saveBookings();
        }

        public User getActiveUser(){
            return usersService.getActiveUser();
        }
    }

```

### **RoomsService.Java:**

```

package Project.service;

import Project.model.Booking;
import Project.model.Room;

import java.io.*;
import java.time.LocalDate;
import java.util.ArrayList;
import java.util.List;
import java.util.stream.Collectors;

public class RoomsService {

```

```

private List<Room> roomList;

public int getRoomSize(){
    return roomList.size();
}
public RoomsService() {
    roomList = new ArrayList<>();
    BufferedReader reader;
    try {
        reader = new BufferedReader(new FileReader(
            "rooms"));
        String line = reader.readLine();
        while (line != null) {
            String[] data = line.split("\t");
            roomList.add(new Room(Integer.parseInt(data[0]),
Integer.parseInt(data[1]), Integer.parseInt(data[2]), data[3],
data[4], Double.parseDouble(data[5])));
            line = reader.readLine();
        }
        reader.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
}

public Room getRoomByRoomId(Long roomId) {
    return roomList.stream().filter(room -> room.getRoomId() ==
roomId).findFirst().get();
}

public void createRoom(int guests, int floor, String standard,
String name, Double price) {
    long nextRoomId = roomList.stream().map(r ->
r.getRoomId()).max(Long::compare).orElse(0L) + 1;
    roomList.add(new Room(nextRoomId, guests, floor, standard,
name, price));
    try {
        saveRooms();
    } catch (IOException e) {
        e.printStackTrace();
    }
}

public List<Room> getRoomsByCriteria(LocalDate dateFrom,
LocalDate dateTo, Double priceMin, Double priceMax, String standard,
int guests) {
    return roomList.stream().filter(room ->
        room.priceMatch(priceMin, priceMax) &&
room.getStandard().equals(standard) && room.getSize() >= guests &&
room.isFree(dateFrom, dateTo)
    ).collect(Collectors.toList());
}

```

```

        public void update() {
            String serialized =
roomList.stream().map(Room::serialize).reduce((acc, x) -> acc + "\n"
+ x).get();
            System.out.println(serialized);
        }

        public void addBookingsToRooms(List<Booking> bookings) {
            roomList.forEach(room -> {
                List<Booking> roomBookings =
bookings.stream().filter(booking -> booking.getRoomId() ==
room.getRoomId()).collect(Collectors.toList());
                room.setBookings(roomBookings);
            });
        }

        void saveRooms() throws IOException {
            String rooms =
roomList.stream().map(Room::serialize).collect(Collectors.joining("\n"));
            BufferedWriter writer = new BufferedWriter(new
FileWriter("rooms"));
            writer.write(rooms);
            writer.close();
        }
    }
}

```

### **UserService.Java:**

```

package Project.service;

import Project.model.Client;
import Project.model.Employee;
import Project.model.User;

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

public class UsersService {

    private List<User> userList;
    private User activeUser;

    public UsersService() {
        loadUsers();
    }

    List<User> getUserList() {
        return userList;
    }
}

```

```

    }

    public User getActiveUser() {
        return activeUser;
    }

    void setActiveUser(String userLogin) {
        this.activeUser = userList.stream().filter(it ->
it.getLogin().equals(userLogin)).findFirst().get();
    }

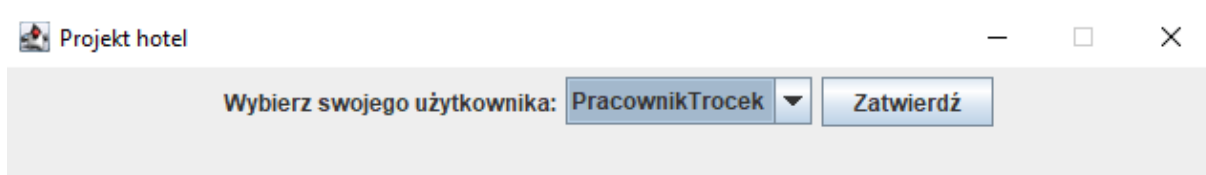
    private void loadUsers(){
        userList = new ArrayList<>();
        BufferedReader reader;
        try {
            reader = new BufferedReader(new FileReader(
                "users"));
            String line = reader.readLine();
            while (line != null) {
                String[] data = line.split(",");
                if(data[0].equals("employee"))
                    userList.add(new Employee(data[1], data[2],
data[3], data[4]));
                else
                    userList.add(new Client(data[1], data[2],
data[3], data[4]));
                line = reader.readLine();
            }
            reader.close();
        } catch (IOException e) {
            e.printStackTrace();
        }

    }

}

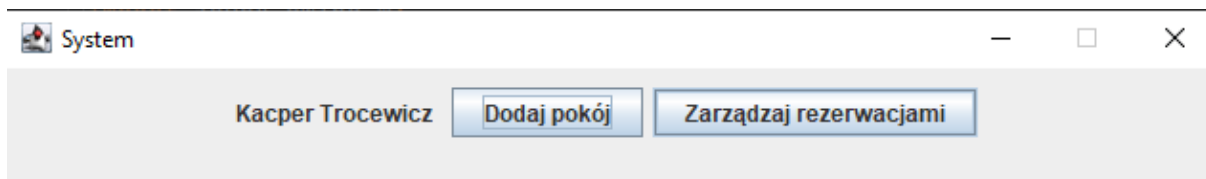
```

## 5 Przykładowe dane i wyniki(Zrzuty z opisem):



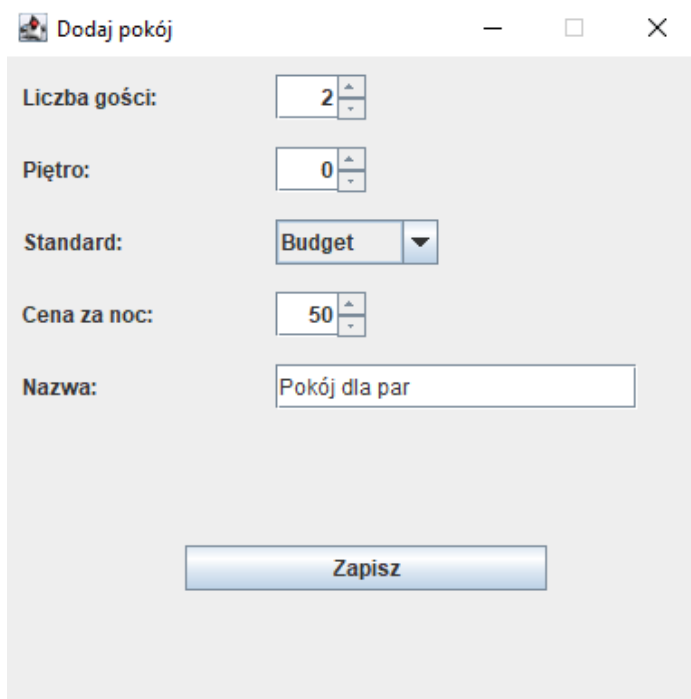
Rysunek 1.Menu Główne.

Na tym screen'ie widoczna jest możliwość wybrania użytkownika lub pracownika.



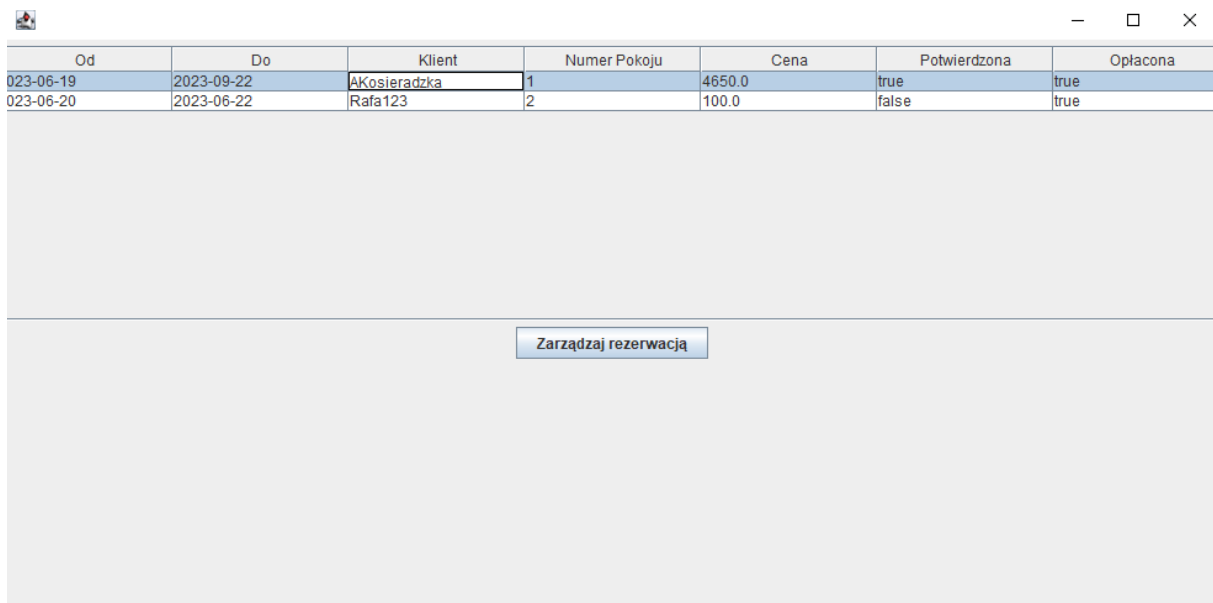
Rysunek 2.Pracownik Hotelu.

Po wybraniu pracownika wyświetlają się jego uprawnienia – jak widoczne jest na screen'ie istnieje możliwość dodania pokoju i zarządzania rezerwacjami.



Rysunek 3.Dodawanie Pokoi w hotelu przez pracownika.

Po wybraniu opcji „Dodaj pokój” wyświetli się okno oraz opcja wyboru rodzaju pokoju, który chcemy dodać do oferty.

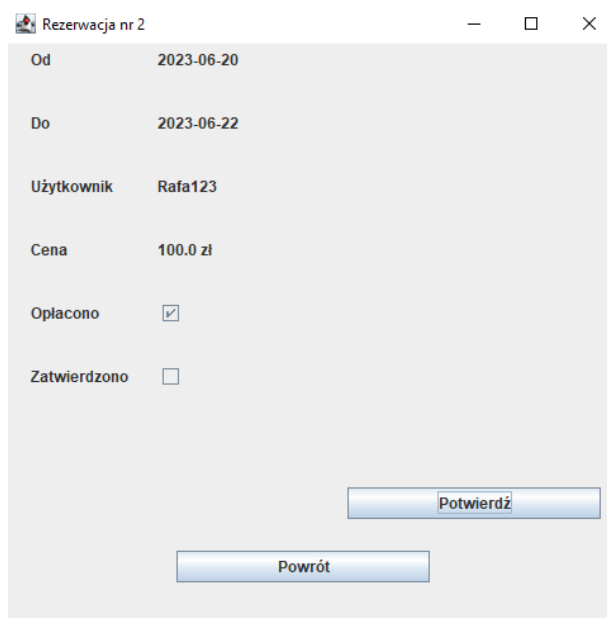


Od	Do	Klient	Numer Pokoju	Cena	Potwierdzona	Opłacona
023-06-19	2023-09-22	AKosieradzka	1	4650.0	true	true
023-06-20	2023-06-22	Rafa123	2	100.0	false	true

Zarządzaj rezerwacją

**Rysunek 4.**Spis Rezerwacji przez klientów u pracownika.

Po wybraniu opcji „Zarządzaj rezerwacjami” istnieje możliwość wyboru jednej, konkretnej rezerwacji ze wszystkich aktualnych rezerwacji.



**Rezerwacja nr 2**

Od: 2023-06-20

Do: 2023-06-22

Użytkownik: Rafa123

Cena: 100.0 zł

Opłacono: ☒

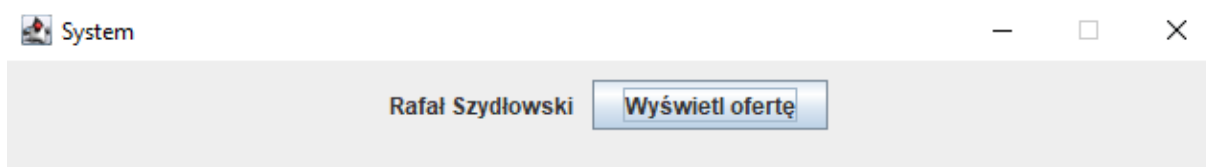
Zatwierdzono: ☐

Potwierdź

Powrót

**Rysunek 5.**Zarządzanie rezerwacją przez Pracownika.

Po naciśnięciu opcji „Zarządzaj rezerwacją” wyświetli się opcja potwierdzenia rezerwacji dokonanej rezerwacji.



Rysunek 6.Menu Użytkownika.

Po wybraniu użytkownika wyświetli się okienko „Wyświetl ofertę”.

A screenshot of a software window titled "Specyfik...". The window contains several input fields and a button. The fields are: "Data od:" with the value "20.06.2023", "Data do:" with the value "22.06.2023", "Cena od:" with the value "50", "Cena do:" with the value "75", "Standard:" with a dropdown menu showing "Budget", and "Goście:" with the value "1". Each of these fields has a small up/down arrow icon to its right. At the bottom of the form is a button labeled "Wyszukaj". The window has standard Windows-style window controls in the top left corner.

Rysunek 7.Parametry wolnych pokoi do wyszukania.

Po wybraniu opcji „Wyświetl ofertę” pojawi się okienko z możliwością wybrania daty rezerwacji, od jakiej ceny do jakiej możliwy do wyboru jest pakiet i ile gości z nami będzie.

Nazwa	Standard	Piętro	Miejsca	Cena za noc	Cena za pobyt
Pokoje dla par	Budget	0	2	50.0	100.0

Zarezerwuj

Rysunek 8..Wszystkie wolne pokoje według parametrów.

Po naciśnięciu wyszukaj wyświetli się tabelka ze wszystkimi możliwymi pokojami i możliwością zarezerwowania go.

System

Rafał Szydłowski

Wyświetl ofertę Zarządzaj rezerwacjami

Rysunek 9.Menu Klienta.

Dopiero po zarezerwowaniu pokoju wyświetla się opcja „Zarządzaj rezerwacjami”.

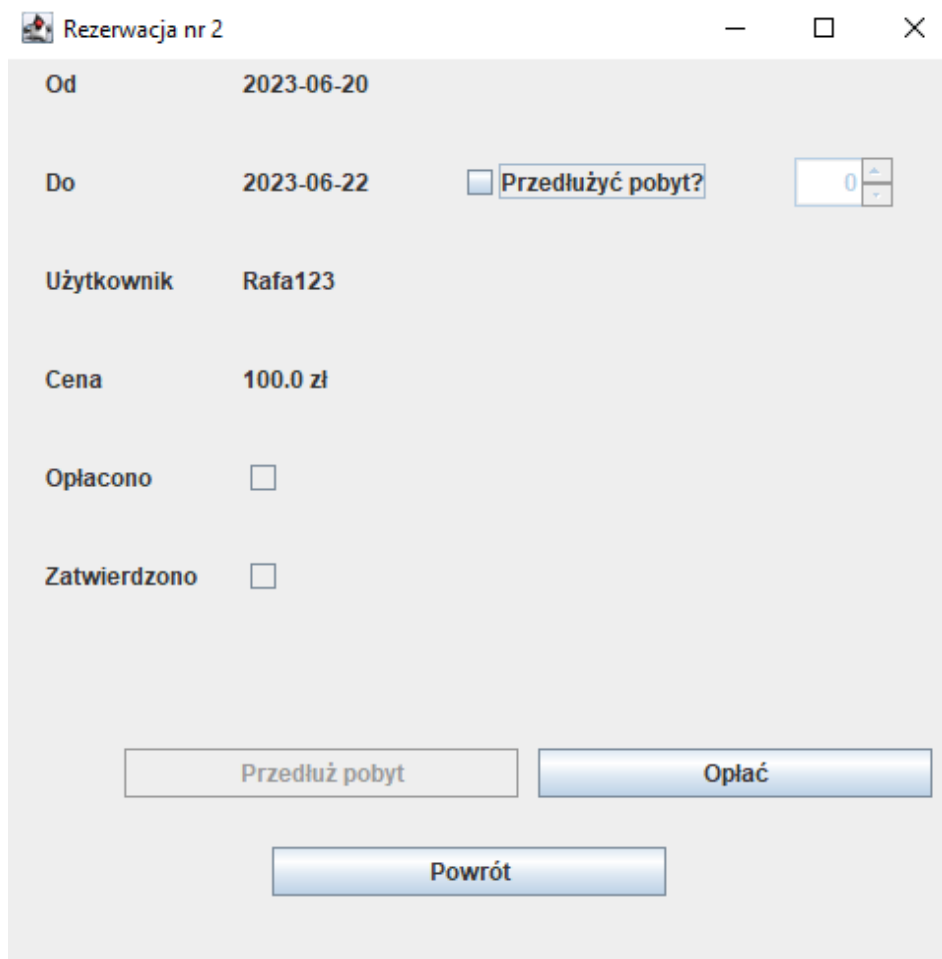
Od	Do	Klient	Numer Pokoju	Cena	Potwierdzona	Opłacona
2023-06-20	2023-06-22	Rafa123	2	100.0	false	false

Zarządzaj rezerwacją

Rysunek 10..Spis rezerwacji danego klienta.

Po kliknięciu Zarządzaj rezerwacjami pokaże się tabelka z pokojami, które klient zarezerwował czy je opłacił czy pracownik potwierdził cena i data.





Rezerwacja nr 2

Od 2023-06-20

Do 2023-06-22 ☐ Przedłużyć pobyt? 0

Użytkownik Rafa123

Cena 100.0 zł

Opłacono ☐

Zatwierdzono ☐

Przedłuż pobyt Opłać

Powrót

Rysunek 11. Zarządzanie rezerwacją przez klienta.

Po kliknięciu Zarządzaj rezerwacją wyświetlają się wszystkie dane, które klient wybrał i ma możliwość opłacenia rezerwacji, jak i przedłużenia pobytu.

## 6 Instrukcja dla użytkownika(jak uruchomić, jakie dane wprowadzać):

Po skompilowaniu programu wyświetli się okienko, w którym istnieje możliwość wyboru użytkownika lub pracownika. Po wybraniu jednej z dwóch opcji zależne od wyboru pokazują się okienko, np. u Pracownika pokażą się dwie opcje: Dodaj Pokój, Zarządzaj Rezerwacjami. Po wybraniu opcji Dodaj pokój pokazuje się kolejne okienko w którym są informacje o pokoju, tj. Liczba gości, Piętro, Standard, Cena za noc, Nazwa. Jednak gdy chcemy wybrać opcję Zarządzaj rezerwacjami pokazują się nam tabelka ze wszystkimi rezerwacjami jakie klienci dokonali podana jest Data Od, Do, Nazwisko Klienta, Cena, Potwierdzona, Opłacona, po naciśnięciu jednej rezerwacji możemy nią zarządzać i jako pracownik mamy możliwość potwierdzenia rezerwacji i przycisk powrotu, który pozwala powrócić do tabelki z klientami. Po Wybraniu użytkownika pokazuje się okienko, w którym są do wyboru opcje Wyświetl ofertę lub Zarządzaj rezerwacjami. Po Wybraniu Wyświetl ofertę pokaże się okienko, w którym należy wybrać Data od, Data do, Cena od, Cena do, Standard, Goście, po dostosowaniu opcji klikamy przycisk wyszukaj, który przeszuka wszystkie pokoje i pokaże te dostępne na wybrane dane. Mamy możliwość wybrania pokoi, które są dostępne i następnie

musimy nacisnąć przycisk Zarezerwuj, który zamknie okno i zarezerwuje dany pokój. Po wybraniu drugiej opcji Zarządzaj rezerwacjami ukaże się tabela, wskazująca które pokoje zarezerwowaliśmy i wszystkie dane które wybraliśmy. Po wybraniu pokoju możemy przedłużyć pobyt lub opłacić rezerwację.

Rysunek 1.Menu Główne.....	37
Rysunek 2.Pracownik Hotelu.....	37
Rysunek 3.Dodawanie Pokoi w hotelu przez pracownika.....	37
Rysunek 4.Spis Rezerwacji przez klientów u pracownika.....	38
Rysunek 5.Zarządzanie rezerwacją przez Pracownika.....	38
Rysunek 6.Menu Użytkownika.....	39
Rysunek 7.Parametry wolnych pokoi do wyszukania.....	39
Rysunek 8..Wszystkie wolne pokoje według parametrów.....	39
Rysunek 9.Menu Klienta.....	40
Rysunek 10..Spis rezerwacji danego klienta.....	40
Rysunek 11.Zarządzanie rezerwacją przez klienta.....	40