

Bazy Danych (Projekt)

“Aplikacja dla optymalizacji diety”

Etap 3 - Projekt, implementacja i testy aplikacji bazodanowej

1. Makiety interfejsu graficznego aplikacji bazodanowej

1.1.1 Logowanie

Ekran, który spotka użytkownika i umożliwia logowanie do aplikacji poprzez podanie danych osobowych. Z ekranu **Logowanie**, można trafić na ekran -1.1.2 **Logowanie(error)**-, w przypadku nie poprawnego wpisania jakichkolwiek danych lub w przypadku nieistnienia konta o podanych danych. Oraz na ekran **-1.2.1 Rejestracja-**, w którym można założyć konto osobiste.

Witaj w OptiDieta!

Email lub login

Hasło

Zaloguj się

Nie masz konta? Zarejestruj się

1.1.2 Logowanie(error)

Ekran, wyświetlający komunikat "Nie poprawny Email/login lub hasło :(" w przypadku braku konta o podanych danych lub ich niepoprawności.

Witaj w OptiDieta!

Email lub login

Hasło

Zaloguj się

Nie masz konta? Zarejestruj się

Nie poprawny Email/login lub hasło :(

1.2.1 Rejestracja

Ekran umożliwiający stworzenie własnego konta użytkownika poprzez podanie odpowiednich danych osobowych spełniających wymagania do formatu danych. Po sukcesywnym zarejestrowaniu uruchamia się przekierowanie na ekran -**1.1.1 Logowanie-**.

Załącz konto!

Zarejestrować się

1.2.2 Rejestracja(error)

Ekran, wyświetlający komunikat "Użytkownik o podanych danych już istnieje. Spróbuj ponownie" w przypadku istnienia konta o podanych danych.

Załącz konto!

Zarejestrować się

Użytkownik o podanych danych już istnieje.
Spróbuj ponownie

1.3.1 Ekran główny

Ekran na którym użytkownik może sprawdzić po podaniu danych osobowych wygenerowane: zapotrzebowanie, listę zakupów oraz plan posiłków. Jak i wszystkie poniżej wymienione makiety, zawiera możliwość przekierowania się na inne ekrany poprzez menu tak jak to jest pokazane na -**1.3.3 Ekran główny(Przełączanie między ekranami)**-.



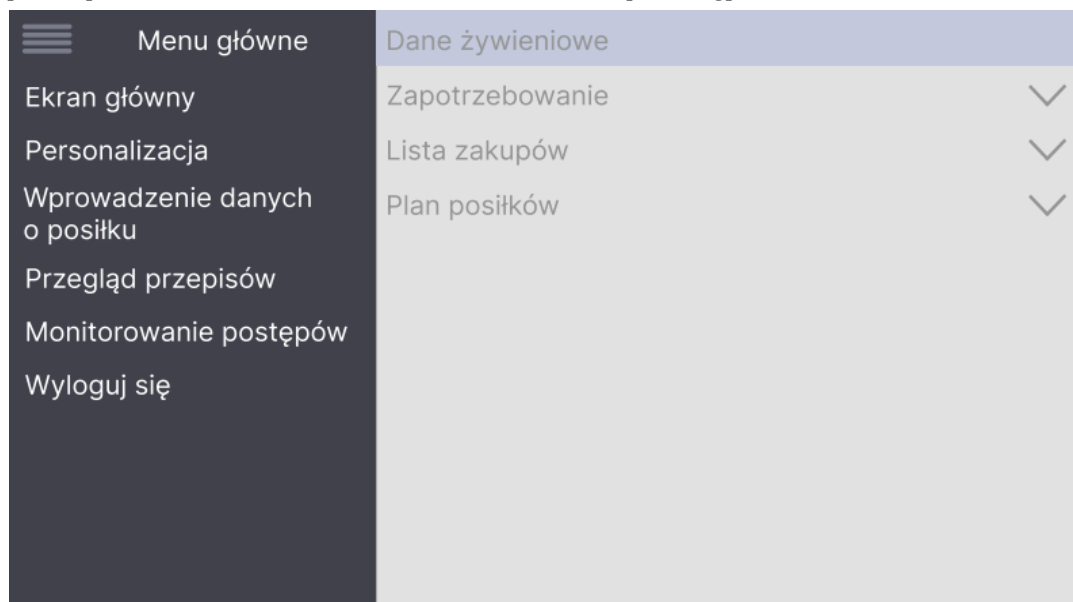
1.3.2 Ekran główny(przegląd danych)

Demonstracja przeglądania danych na głównej stronie.



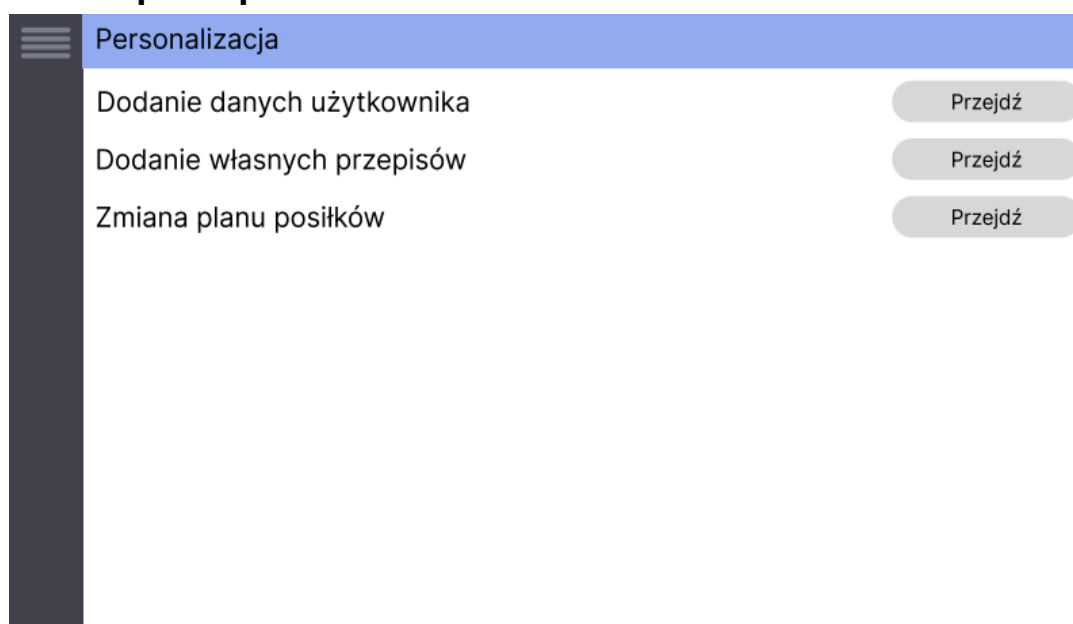
1.3.3 Ekran główny(Przełączanie między ekranami)

Demonstracja menu głównego zawierającego punkty dla przełączenia się między funkcjami aplikacji:-**1.1.1 Logowanie**-,
-**1.3.3 Ekran główny**-, -**1.4.1 Personalizacja**-, -**1.5.1 Wprowadzenie danych o posiłku**-, -**1.6.1 Przegląd przepisów**-, -**1.7.1 Monitorowanie postępów**-.




1.4.1 Personalizacja




Personalizacja umożliwia zmianę danych użytkownika, dostosowanie do siebie planu posiłkowego oraz dodanie własnych przepisów, których nie ma w bazie danych. Wszystko działa poprzez naciśnięcie na odpowiednie przyciski, które przełączą użytkownika na widoki: -**1.4.2 Dodanie danych użytkownika**-, -**1.4.3 Dodanie własnych przepisów**-, -**1.4.4 Zmiana planu posiłków**-.




1.4.2 Dodanie danych użytkownika



Dodanie danych użytkownika

Wiek	<input type="text" value="..."/>	Od 1 do 99
Płeć	<input type="text" value="..."/>	M/K
Wzrost	<input type="text" value="..."/>	Od 100 do 230cm
Aktywność fizyczna	<input type="text" value="..."/>	Wpisać ile godzin
Alergii	<input type="text" value="..."/> 	Proszę wybrać z proponowanych
Typ diety	<input type="text" value="..."/> 	Proszę wybrać z proponowanych
Preferowana ilość posiłków	<input type="text" value="..."/>	Od 1 do 5
Cel diety	<input type="text" value="..."/> 	Proszę wybrać z proponowanych

1.4.3 Dodanie własnych przepisów




Dodanie własnych przepisów

Dodaj swój ulubiony przepis do naszej bazy danych.

Nazwa przepisu:

Składniki:


 

Ilość:


Przepis: 3 pomidory, 2 ogórki, 1 puszka kukurudzy.


Ten przepis zawiera: 2400 kcal, 80 g tłuszczu, 270 g węglowodanów, 70 g białka

1.4.4 Zmiana planu posiłków

 Zmiana planu posiłków

Zmodyfikuj swój plan posiłków, aby dopasować go do swoich potrzeb i preferencji.

Posiłek: 

Składniki: 

Ilość:

Przepis: 10 pomidorów, 25 ogórki, 15 papryczek.

Ten przepis zawiera: 24000 kcal, 810 g tłuszczu, 470 g węglowodanów, 170 g białka

1.5.1 Wprowadzenie danych o posiłku

Na tym ekranie jest pokazane menu na którym użytkownik wprowadza swoje spożyte posiłki poprzez podanie danych o tym posiłku, o jego ilości oraz o czasie kiedy go zjeł.

 Dodaj posiłek

Wprowadź informację o spożytych posiłkach.

Rodzaj posiłku: 


Składniki: 

Ilość: 


Czas: 

1.6.1 Przegląd przepisów

Ekran z wyszukiwarką przepisów na którym, można sprawdzić wszystkie dane dania, dodać go do planu lub sprawdzić zamienniki składników.

 Przegląd przepisów

Przeglądaj przepisy dopasowane do Twoich potrzeb i preferencji.


Wprowadź nazwę przepisu: 

Ten przepis zawiera: 2400 kcal, 80 g tłuszczu, 270 g węglowodanów, 70 g białka

Składniki: 2 jajka, 1/2 cebuli, 1/2 papryki, 1/2 pomidora.

dodaj przepis do planu posiłków


lub przeglądaj zamienniki składników w danym przepisie



- Marchew
- Pieczarki
- Por

1.7.1 Monitorowanie postępów

Na tym ekranie użytkownik może sprawdzić swoje osiągnięcia oraz otrzymać nagrodę: -**1.7.2 Monitorowanie postępów (otrzymanie nagrody)-**.

 Monitorowanie postępów

Monitorowanie postępów

Cel: 2000 kcal, 70 g tłuszczu, 250 g węglowodanów, 100 g białka

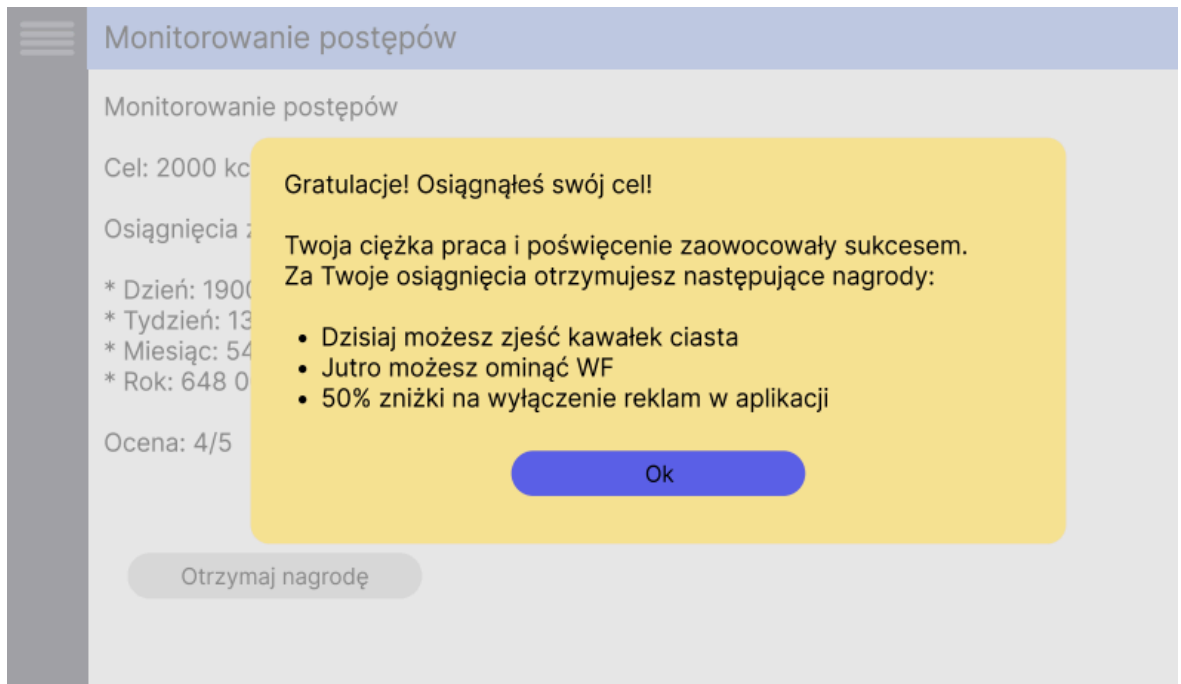
Osiągnięcia za:

- * Dzień: 1900 kcal, 65 g tłuszczu, 245 g węglowodanów, 95 g białka
- * Tydzień: 13 500 kcal, 505 g tłuszczu, 1705 g węglowodanów, 775 g białka
- * Miesiąc: 54 000 kcal, 2020 g tłuszczu, 6820 g węglowodanów, 3050 g białka
- * Rok: 648 000 kcal, 24200 g tłuszczu, 84 300 g węglowodanów, 36 500 g białka

Ocena: 4/5

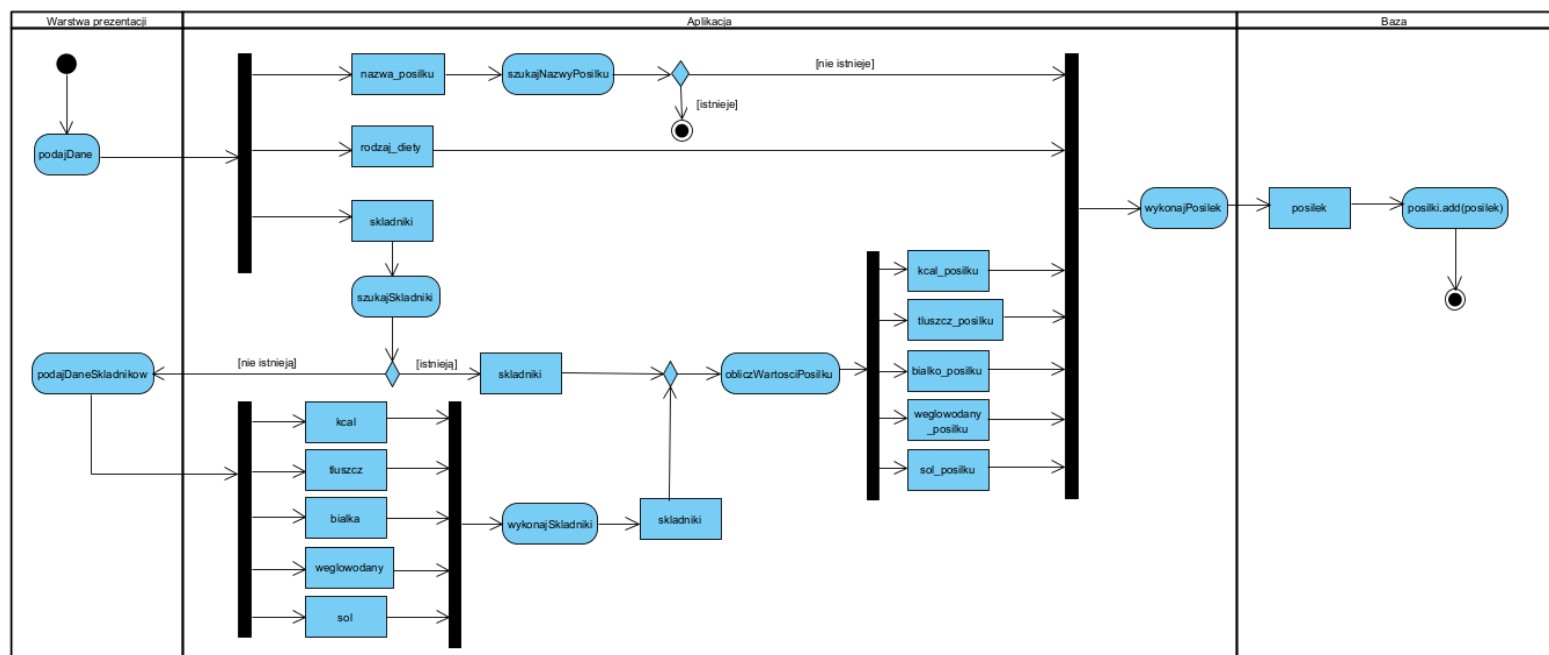
1.7.2 Monitorowanie postępów(otrzymanie nagrody)

Przykład nagrody.

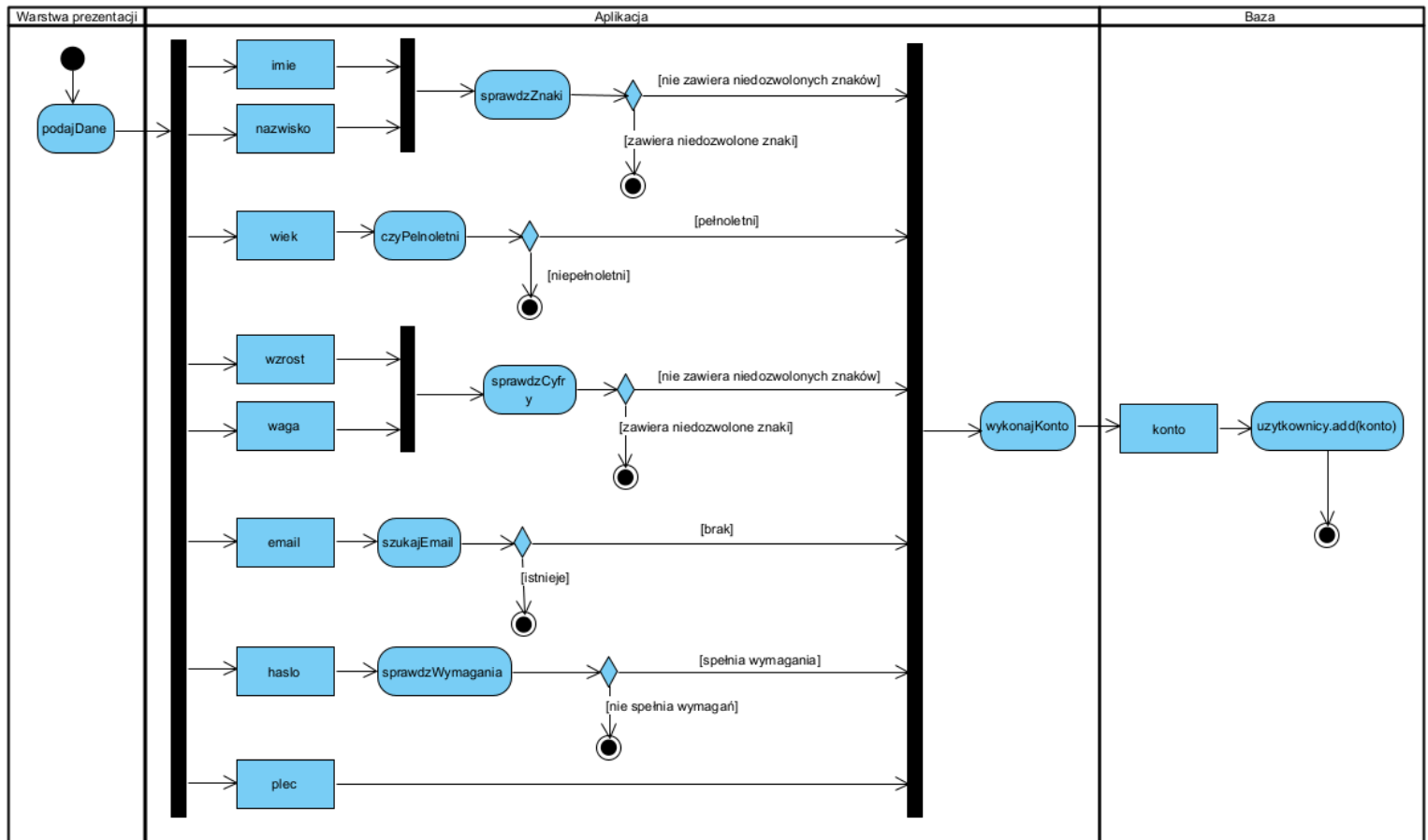


2. Diagramy czynności

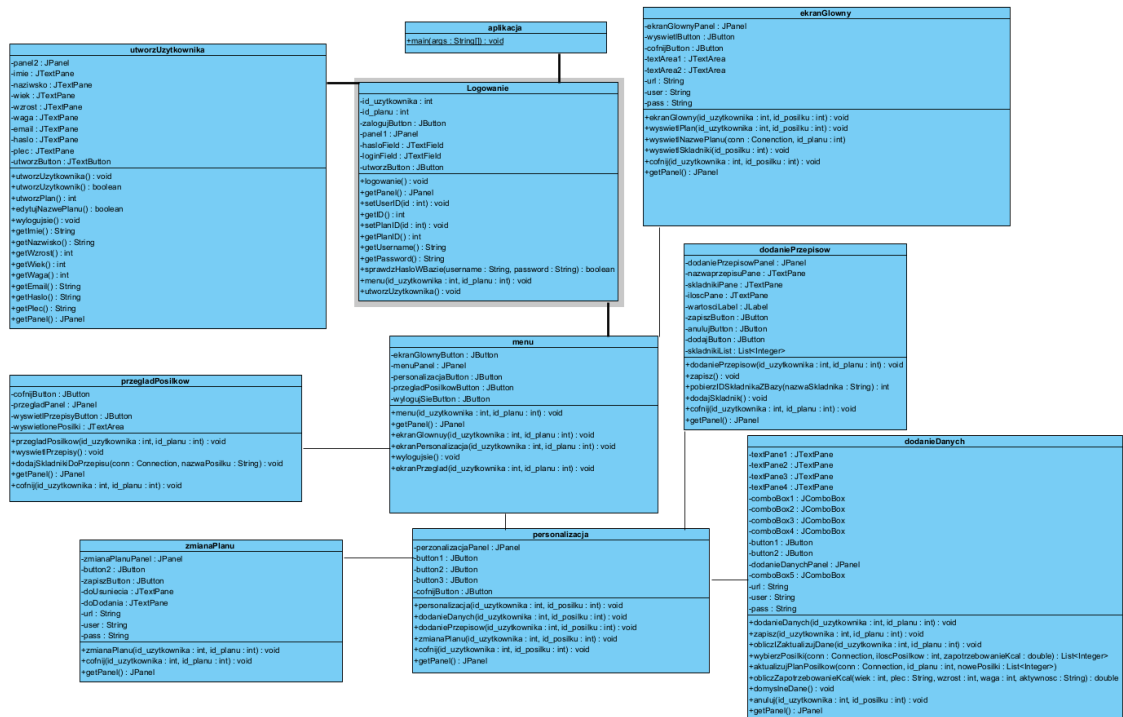
2.1. Diagram czynności dla PU Dodanie posiłku



2.2. Diagram czynności dla PU Rejestracja



3. Diagram klas



4. Aplikacja

Klasa aplikacja

```

public static void main(String[] args) {
    SwingUtilities.invokeLater(new Runnable() {
        @Override
        public void run() {
            JFrame frame = new JFrame( title: "Logowanie");
            frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

            logowanie logowanieUI = new Logowanie();
            frame.setContentPane(logowanieUI.getPanel());

            frame.pack();
            frame.setVisible(true);
        }
    });
}

```

Klasa logowanie

```

public logowanie() {
    zalogujButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            String username = getUsername();
            String password = getPassword();

            if (sprawdzHasloWBazie(username, password)) {
                JOptionPane.showMessageDialog( parentComponent: null, message: "Zalogowano jako: " + username);
                menu(getID(), getPlanID());
                System.out.println(getPlanID());
            }else{
                JOptionPane.showMessageDialog( parentComponent: null, message: "Błędny login lub hasło");
            }
        }
    });

    utworzButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) { utworzUzytkownika(); }
    });
}

```

```

public JPanel getPanel() { return panel1; }

1 usage
private void setUserID(int id) { this.id_uzytkownika = id; }

1 usage
public int getID() { return id_uzytkownika; }

1 usage
private void setPlanID(int id) { this.id_planu = id; }

2 usages
public int getPlanID() { return id_planu; }

1 usage
public String getUsername() { return loginField.getText(); }

1 usage
public String getPassword() { return hasloField.getText(); }

```

```
public boolean sprawdzHasloWBazie(String username, String password) {
    String url = "jdbc:mysql://localhost:3306/sys";
    String user = "root";
    String pass = "Haslo123.";

    try (Connection conn = DriverManager.getConnection(url, user, pass)) {
        String sql = "SELECT id_uzytkownika, id_planu FROM Uzytkownik WHERE email = ? AND haslo = ?";
        try (PreparedStatement statement = conn.prepareStatement(sql)) {
            statement.setString(1, username);
            statement.setString(2, password);
            ResultSet resultSet = statement.executeQuery();
            if (resultSet.next()) {
                int userID = resultSet.getInt("id_uzytkownika");
                setUserID(userID);
                int planID = resultSet.getInt("id_planu");
                setPlanID(planID);
                return true;
            } else {
                return false;
            }
        }
    } catch (SQLException ex) {
        ex.printStackTrace();
        return false;
    }
}
```

```

private void menu(int id_uzytkownika, int id_planu){
    JFrame frame = (JFrame) SwingUtilities.getWindowAncestor(getPanel());
    frame.dispose();

    JFrame utworzFrame = new JFrame( title: "Menu");
    menu Menu = new menu(id_uzytkownika, id_planu);
    utworzFrame.setContentPane(Menu.getPanel());
    utworzFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    utworzFrame.pack();
    utworzFrame.setVisible(true);
}

1 usage
private void utworzUzytkownika() {
    utworzButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            JFrame frame = (JFrame) SwingUtilities.getWindowAncestor(getPanel());
            frame.dispose();
            JFrame utworzFrame = new JFrame( title: "Utwórz Uzytkownika");
            utworzUzytkownika utworzUzyt = new utworzUzytkownika();
            utworzFrame.setContentPane(utworzUzyt.getPanel());
            utworzFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            utworzFrame.pack();
            utworzFrame.setVisible(true);
        }
    });
}

```

Klasa utworzUzytkowniak

```

public utworzUzytkownika(){
    utworzButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            if(utworzUzytkownik()) {
                JOptionPane.showMessageDialog( parentComponent: null, message: "Pomyślnie utworzono użytkownika.");
                wylogujSie();
            } else {
                JOptionPane.showMessageDialog( parentComponent: null, message: "Błędne dane.");
            }
        }
    });
}
}

```

```

public boolean utworzUzytkownik(){
    String url = "jdbc:mysql://localhost:3306/sys";
    String user = "root";
    String pass = "Haslo123.";
    if (getImie().isEmpty() || getNazwisko().isEmpty() || getEmail().isEmpty() || getHaslo().isEmpty()) {return false;
    try {
        getWiek();
        getWzrost();
        getWaga();
    } catch (NumberFormatException e) {
        JOptionPane.showMessageDialog( parentComponent: null, message: "Wprowadź poprawne wartości liczbowe.");
        return false;
    }
    try (Connection conn = DriverManager.getConnection(url, user, pass)) {
        int idPlanu = utworzPlan(conn);
        String sql = "INSERT INTO Uzytkownik (imie, nazwisko, wiek, wzrost, waga, email, haslo, plec, id_planu) VALUES";
        try (PreparedStatement statement = conn.prepareStatement(sql)) {
            statement.setString( parameterIndex: 1, getImie());
            statement.setString( parameterIndex: 2, getNazwisko());
            statement.setInt( parameterIndex: 3, getWiek());
            statement.setInt( parameterIndex: 4, getWzrost());
            statement.setInt( parameterIndex: 5, getWaga());
            statement.setString( parameterIndex: 6, getEmail());
            statement.setString( parameterIndex: 7, getHaslo());
            statement.setString( parameterIndex: 8, getPlec());
            statement.setInt( parameterIndex: 9, idPlanu);

            int rowsAffected = statement.executeUpdate();
            if (rowsAffected > 0) {
                edytujNazwePlanu(conn, idPlanu, getImie(), getNazwisko());
                return true;
            }
        }
    }
    } catch (SQLException ex) {
        ex.printStackTrace();
        return false;
    }
    return false;
}
}

```

```

private int utworzPlan(Connection conn) throws SQLException {
    String insertPlanSQL = "INSERT INTO Plan_posilkow (nazwa_planu) VALUES (?)";
    try (PreparedStatement insertPlanStmt = conn.prepareStatement(insertPlanSQL, Statement.RETURN_GENERATED_KEYS)) {
        insertPlanStmt.setString(1, "Plan uzytkownika:");
        int rowsAffected = insertPlanStmt.executeUpdate();

        if (rowsAffected > 0) {
            try (ResultSet generatedKeys = insertPlanStmt.getGeneratedKeys()) {
                if (generatedKeys.next()) {
                    return generatedKeys.getInt(1);
                }
            }
        }
    }

    return -1;
}

```

1 usage

```

private boolean edytujNazwePlanu(Connection conn, int idPlanu, String imie, String nazwisko) {
    String nowaNazwaPlanu = "Planm uzytkownika "+imie+" "+nazwisko;

    String updatePlanSQL = "UPDATE Plan_posilkow SET nazwa_planu = ? WHERE id_planu = ?";
    try (PreparedStatement updatePlanStmt = conn.prepareStatement(updatePlanSQL)) {
        updatePlanStmt.setString(1, nowaNazwaPlanu);
        updatePlanStmt.setInt(2, idPlanu);

        int rowsAffected = updatePlanStmt.executeUpdate();
        return rowsAffected > 0;
    } catch (SQLException ex) {
        ex.printStackTrace();
        return false;
    }
}

```



```

private void wylogujSie(){
    JFrame frame = (JFrame) SwingUtilities.getWindowAncestor(getPanel());
    frame.dispose();

    JFrame utworzFrame = new JFrame( title: "Logowanie");
    logowanie ekran = new logowanie();
    utworzFrame.setContentPane(ekran.getPanel());
    utworzFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    utworzFrame.pack();
    utworzFrame.setVisible(true);
}

3 usages
public String getImie() { return imie.getText(); }

3 usages
private String getNazwisko() { return nazwisko.getText(); }

2 usages
private int getWiek() { return Integer.parseInt(wiek.getText()); }

2 usages
private int getWzrost() { return Integer.parseInt(wzrost.getText()); }

2 usages
private int getWaga() { return Integer.parseInt(waga.getText()); }

2 usages
private String getEmail() { return email.getText(); }

2 usages
private String getHaslo() { return haslo.getText(); }

1 usage
private String getPlec() { return plec.getText(); }

private int getIdPlanu() { return Integer.parseInt(id_planu.getText()); }

public JPanel getPanel() { return panel2; }

```

Klasa menu

```

public menu(int id_uzytkownika, int id_planu){

    ekranGlownyButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) { ekranGlowny(id_uzytkownika, id_planu); }
    });
    personalizacjaButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) { ekranPersonalizacja(id_uzytkownika, id_planu); }
    });
    wylogujSieButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) { wylogujSie(); }
    });
    przegladPosilkowButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) { ekranPrzeglad(id_uzytkownika, id_planu); }
    });
}

public JPanel getPanel() { return menuPanel; }

1 usage
private void ekranGlowny(int id_uzytkownika, int id_planu){
    JFrame frame = (JFrame) SwingUtilities.getWindowAncestor(getPanel());

    JFrame utworzFrame = new JFrame( title: "EkranGlowny");
    ekranGlowny ekran = new ekranGlowny(id_uzytkownika, id_planu);
    utworzFrame.setContentPane(ekran.getPanel());
    utworzFrame.pack();
    utworzFrame.setVisible(true);
}

```

```

private void ekranPersonalizacja(int id_uzytkownika, int id_planu){
    JFrame frame = (JFrame) SwingUtilities.getWindowAncestor(getPanel());
    frame.dispose();

    JFrame utworzFrame = new JFrame( title: "Personalizacja");
    personalizacja ekran = new personalizacja(id_uzytkownika, id_planu);
    utworzFrame.setContentPane(ekran.getPanel());
    utworzFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    utworzFrame.pack();
    utworzFrame.setVisible(true);
}

1 usage
private void wylogujsie(){
    JFrame frame = (JFrame) SwingUtilities.getWindowAncestor(getPanel());
    frame.dispose();

    JFrame utworzFrame = new JFrame( title: "Logowanie");
    logowanie ekran = new logowanie();
    utworzFrame.setContentPane(ekran.getPanel());
    utworzFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    utworzFrame.pack();
    utworzFrame.setVisible(true);
}

1 usage
private void ekranPrzegląd(int id_uzytkownika, int id_planu){
    JFrame frame = (JFrame) SwingUtilities.getWindowAncestor(getPanel());
    JFrame utworzFrame = new JFrame( title: "Przegląd posiłków");
    przeglądPosiłkow ekran = new przeglądPosiłkow(id_uzytkownika, id_planu);
    utworzFrame.setContentPane(ekran.getPanel());
    utworzFrame.pack();
    utworzFrame.setVisible(true);
}

```

Klasa logowanie dodanieDanych

```

public dodanieDanych(int id_uzytkownika, int id_planu){
    domyslnDane();
    button1.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            zapisz(id_uzytkownika, id_planu);
            obliczIZaktualizujDane(id_uzytkownika, id_planu);
        }
    });
    button2.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) { anuluj(id_uzytkownika, id_planu); }
    });
}

```

```

public void zapisz(int id_uzytkownika, int id_planu){
    String wiek = textPane1.getText();
    String plec = textPane2.getText();
    String wzrost = textPane3.getText();
    String waga = textPane4.getText();
    String alergie = (String) comboBox1.getSelectedItem();
    int iloscPosilkow = Integer.parseInt((String) comboBox3.getSelectedItem());
    String cel = (String) comboBox4.getSelectedItem();

    try (Connection conn = DriverManager.getConnection(url, user, pass)) {

        String updatePreferencjeSQL1 = "UPDATE Uzytkownik SET wiek=?, wzrost=?, waga=?, plec=? WHERE id_uzytkownika=?";
        try (PreparedStatement updateUzytkownikStmt = conn.prepareStatement(updatePreferencjeSQL1)) {
            updateUzytkownikStmt.setInt(1, Integer.parseInt(wiek));
            updateUzytkownikStmt.setInt(2, Integer.parseInt(wzrost));
            updateUzytkownikStmt.setInt(3, Integer.parseInt(waga));
            updateUzytkownikStmt.setString(4, plec);
            updateUzytkownikStmt.setInt(5, id_uzytkownika);
            updateUzytkownikStmt.executeUpdate();
        } catch (SQLException e){
            e.printStackTrace();
            JOptionPane.showMessageDialog( parentComponent: null, message: "Błędne dane.");
        }

        String updatePreferencjeSQL2 = "UPDATE Preferencje SET cel=?, ilosc_posilkow=? WHERE id_preferencji=?";
        try (PreparedStatement updatePreferencjeStmt = conn.prepareStatement(updatePreferencjeSQL2)) {
            updatePreferencjeStmt.setString(1, cel);
            //updatePreferencjeStmt.setString(2, alergie);
            updatePreferencjeStmt.setInt(2, iloscPosilkow);
            updatePreferencjeStmt.setInt(3, id_uzytkownika);
            updatePreferencjeStmt.executeUpdate();
        } catch (SQLException e){
            e.printStackTrace();
            JOptionPane.showMessageDialog( parentComponent: null, message: "Błędne dane.");
        }

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

```

```

public void obliczIZaktualizujDane(int id_uzytkownika, int id_planu) {

    String aktywnosc = (String) comboBox5.getSelectedItemAt();
    int iloscPosilkow = Integer.parseInt((String) comboBox3.getSelectedItemAt());
    try (Connection conn = DriverManager.getConnection(url, user, pass)) {
        String selectUzytkownikSQL = "SELECT wiek, plec, wzrost, waga FROM Uzytkownik WHERE id_uzytkownika=?";
        try (PreparedStatement selectUzytkownikStmt = conn.prepareStatement(selectUzytkownikSQL)) {
            selectUzytkownikStmt.setInt( parameterIndex: 1, id_uzytkownika);

            try (ResultSet uzytkownikResultSet = selectUzytkownikStmt.executeQuery()) {
                if (uzytkownikResultSet.next()) {
                    int wiek = uzytkownikResultSet.getInt( columnLabel: "wiek");
                    String plec = uzytkownikResultSet.getString( columnLabel: "plec");
                    int wzrost = uzytkownikResultSet.getInt( columnLabel: "wzrost");
                    int waga = uzytkownikResultSet.getInt( columnLabel: "waga");
                    double zapotrzebowanieKcal = obliczZapotrzebowanieKcal(wiek, plec, wzrost, waga, aktywnosc);
                    String updatePreferencjeSQL = "UPDATE Preferencje SET cel=?, ilosc_posilkow=? WHERE id_planu=?";
                    try (PreparedStatement updatePreferencjeStmt = conn.prepareStatement(updatePreferencjeSQL)) {
                        updatePreferencjeStmt.setString( parameterIndex: 1, x: "Cel");
                        updatePreferencjeStmt.setInt( parameterIndex: 2, iloscPosilkow);
                        updatePreferencjeStmt.setInt( parameterIndex: 3, id_uzytkownika);

                        updatePreferencjeStmt.executeUpdate();
                    } catch (SQLException e){
                        JOptionPane.showMessageDialog( parentComponent: null, message: "Błędne dane.");
                    }
                    List<Integer> wybranePosilki = wybierzPosilki(conn, iloscPosilkow, zapotrzebowanieKcal);
                    aktualizujPlanPosilkow(conn, id_planu, wybranePosilki);

                    System.out.println("Zapotrzebowanie kaloryczne: " + zapotrzebowanieKcal + " kcal");
                } else {
                    System.out.println("Nie znaleziono użytkownika o ID: " + id_uzytkownika);
                }
            }
        }
    } catch (SQLException e) {
        JOptionPane.showMessageDialog( parentComponent: null, message: "Błędne dane.");
        e.printStackTrace();
    }
}

```

```

private List<Integer> wybierzPosilki(Connection conn, int iloscPosilkow, double zapotrzebowanieKcal) throws SQLException {
    List<Integer> posilki = new ArrayList<>();

    String selectPosilkiSQL = "SELECT id_posilku FROM Posilki WHERE kcal_posilku <= ? ORDER BY RAND() LIMIT ?";
    try (PreparedStatement selectPosilkiStmt = conn.prepareStatement(selectPosilkiSQL)) {
        selectPosilkiStmt.setDouble( parameterIndex: 1, zapotrzebowanieKcal);
        selectPosilkiStmt.setInt( parameterIndex: 2, iloscPosilkow);

        try (ResultSet rs = selectPosilkiStmt.executeQuery()) {
            while (rs.next()) {
                int idPosilku = rs.getInt( columnLabel: "id_posilku");
                posilki.add(idPosilku);
            }
        }
    }

    return posilki;
}

```

1 usage

```

public void aktualizujPlanPosilkow(Connection conn, int id_planu, List<Integer> nowePosilki) throws SQLException {
    String deletePosilkiSQL = "DELETE FROM Posilki_Plan_posilkow WHERE Plan_posilkowid_planu = ?";
    try (PreparedStatement deletePosilkiStmt = conn.prepareStatement(deletePosilkiSQL)) {
        deletePosilkiStmt.setInt( parameterIndex: 1, id_planu);
        deletePosilkiStmt.executeUpdate();
    }

    String insertPosilkiSQL = "INSERT INTO Posilki_Plan_posilkow (Posilkiid_posilku, Plan_posilkowid_planu) VALUES (?, ?)";
    try (PreparedStatement insertPosilkiStmt = conn.prepareStatement(insertPosilkiSQL)) {
        for (int id_posilku : nowePosilki) {
            insertPosilkiStmt.setInt( parameterIndex: 1, id_posilku);
            insertPosilkiStmt.setInt( parameterIndex: 2, id_planu);
            insertPosilkiStmt.executeUpdate();
        }
    }
}

```

```

public static double obliczZapotrzebowanieKcal(int wiek, String plec, int wzrost, int waga, String aktywnosc) {
    double wspolczynnikKcal = (plec.equalsIgnoreCase("M") ? 1.0 : 0.9);
    double wspolczynnikAktywnosci = 0;

    switch (aktywnosc) {
        case "niska":
            wspolczynnikAktywnosci = 1.2;
            break;
        case "umiarkowana":
            wspolczynnikAktywnosci = 1.55;
            break;
        case "wysoka":
            wspolczynnikAktywnosci = 1.9;
            break;
    }

    return ((10 * waga) + (6.25 * wzrost) - (5 * wiek) + 5) * wspolczynnikAktywnosci * wspolczynnikKcal;
}

1 usage
public void domyslnaDane() {
    try (Connection conn = DriverManager.getConnection(url, user, pass)) {
        int id_uzytkownika = 1;

        String selectDaneSQL = "SELECT wiek, plec, wzrost, waga FROM Uzytkownik WHERE id_uzytkownika=?";
        try (PreparedStatement selectDaneStmt = conn.prepareStatement(selectDaneSQL)) {
            selectDaneStmt.setInt(1, id_uzytkownika);

            try (ResultSet rs = selectDaneStmt.executeQuery()) {
                if (rs.next()) {
                    textPane1.setText(String.valueOf(rs.getInt(1)));
                    textPane2.setText(rs.getString(2));
                    textPane3.setText(String.valueOf(rs.getInt(3)));
                    textPane4.setText(String.valueOf(rs.getInt(4)));
                }
            }
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

```

```

public void anuluj(int id_uzytkownika, int id_planu){
    JFrame frame = (JFrame) SwingUtilities.getWindowAncestor(getPanel());
    frame.dispose();

    JFrame utworzFrame = new JFrame( "Personalizacja");
    personalizacja ekran = new personalizacja(id_uzytkownika, id_planu);
    utworzFrame.setContentPane(ekran.getPanel());
    utworzFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    utworzFrame.pack();
    utworzFrame.setVisible(true);
}

public JPanel getPanel() { return dodanieDanychPanel; }

```

Klasa dodaniePrzepisow

```

public dodaniePrzepisow(int id_uzytkownika, int id_planu){
    zapiszButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) { zapisz(); }
    });
    anulujButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) { cofnij(id_uzytkownika, id_planu); }
    });
    dodajButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) { dodajSkladnik(); }
    });
}
}

```

```

public void zapisz(){
    String url = "jdbc:mysql://localhost:3306/sys";
    String user = "root";
    String pass = "Haslo123.";
    String nazwa_posilku = nazwaPrzepisuPane.getText();
    int sumaKcal = 0;
    float sumaTluszcz = 0;
    float sumaBialka = 0;
    float sumaWeglowodany = 0;
    float sumaSol = 0;

    try (Connection conn = DriverManager.getConnection(url, user, pass)) {
        String[] skladniki = skladnikPane.getText().split( regex: " ");
        String[] ilosci = iloscPane.getText().split( regex: " ");

        if (skladniki.length == ilosci.length) {
            for (int i = 0; i < skladniki.length; i++) {
                String nazwaSkladnika = skladniki[i].trim();
                int iloscSkladnika = Integer.parseInt(ilosci[i].trim());

                String selectSkladnikSQL = "SELECT kcal, tluszcz, bialka, weglowodany, sol FROM Skladniki WHERE nazwa_skladnika = ?";
                try (PreparedStatement selectSkladnikStmt = conn.prepareStatement(selectSkladnikSQL)) {
                    selectSkladnikStmt.setString( parameterIndex: 1, nazwaSkladnika);

                    try (ResultSet skladnikResultSet = selectSkladnikStmt.executeQuery()) {
                        if (skladnikResultSet.next()) {
                            float kcalSkladnika = skladnikResultSet.getFloat( columnLabel: "kcal");
                            float tluszczSkladnika = skladnikResultSet.getFloat( columnLabel: "tluszcz");
                            float bialkaSkladnika = skladnikResultSet.getFloat( columnLabel: "bialka");
                            float weglowodanySkladnika = skladnikResultSet.getFloat( columnLabel: "weglowodany");
                            float solSkladnika = skladnikResultSet.getFloat( columnLabel: "sol");

                            sumaKcal += kcalSkladnika * iloscSkladnika;
                            sumaTluszcz += tluszczSkladnika * iloscSkladnika;
                            sumaBialka += bialkaSkladnika * iloscSkladnika;
                            sumaWeglowodany += weglowodanySkladnika * iloscSkladnika;
                            sumaSol += solSkladnika * iloscSkladnika;
                        } else {

```



```

        return;
    }
}
}
} else {
    return;
}

String updatePrzepisSQL = "INSERT INTO Posilki (nazwa_posilku, kcal_posilku, tluszcz_posilku, bialka_posilku, weglowodany_posilku, sol_posilku) VALUES (?, ?, ?, ?, ?, ?)";
try (PreparedStatement statement = conn.prepareStatement(updatePrzepisSQL, Statement.RETURN_GENERATED_KEYS)) {
    statement.setString(1, nazwa_posilku);
    statement.setInt(2, sumaKcal);
    statement.setFloat(3, sumaTluszcz);
    statement.setFloat(4, sumaBialka);
    statement.setFloat(5, sumaWeglowodany);
    statement.setFloat(6, sumaSol);

    int affectedRows = statement.executeUpdate();

    if (affectedRows > 0) {
        String wyniki = "Kcal: " + sumaKcal + "\n"
            + "Tluszcz: " + sumaTluszcz + "\n"
            + "Białko: " + sumaBialka + "\n"
            + "Weglowodany: " + sumaWeglowodany + "\n"
            + "Sól: " + sumaSol;
        wartosciLabel.setText(wyniki);

        try (ResultSet generatedKeys = statement.getGeneratedKeys()) {
            if (generatedKeys.next()) {
                int idPrzepisu = generatedKeys.getInt(1);

                for (Integer idSkladnika : skladnikilist) {
                    String insertSkladnikPrzepisSQL = "INSERT INTO Posilki_Skladniki (Posilkiid_posilku, Skladnikiid_skladnika) VALUES (?, ?)";
                    try (PreparedStatement skladnikPrzepisStmt = conn.prepareStatement(insertSkladnikPrzepisSQL)) {
                        skladnikPrzepisStmt.setInt(1, idPrzepisu);
                        skladnikPrzepisStmt.setInt(2, idSkladnika);
                        skladnikPrzepisStmt.executeUpdate();
                    }
                }
            }
        }
    }
}

```

```

        }
    }
    } else {
        wartoscilabel.setText("Nie udało się zaktualizować przepisu.");
    }
}
} catch (SQLException e) {
    JOptionPane.showMessageDialog( parentComponent: null, message: "Błąd przy zapisie. ");
    e.printStackTrace();
}
}
}

```

1 usage

```

private int pobierzIdSkładnikaZBazy(String nazwaSkładnika) {
    String url = "jdbc:mysql://localhost:3306/sys";
    String user = "root";
    String pass = "Hasło123.";

    try (Connection conn = DriverManager.getConnection(url, user, pass)) {
        String selectSkładnikSQL = "SELECT id_składnika FROM Składniki WHERE nazwa_składnika = ?";
        try (PreparedStatement selectSkładnikStmt = conn.prepareStatement(selectSkładnikSQL)) {
            selectSkładnikStmt.setString( parameterIndex: 1, nazwaSkładnika);

            try (ResultSet składnikResultSet = selectSkładnikStmt.executeQuery()) {
                if (składnikResultSet.next()) {
                    return składnikResultSet.getInt( columnLabel: "id_składnika");
                }
            }
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }

    return -1;
}
}

```

```

public void dodajSkladnik() {
    String nazwaSkladnika = skladnikPane.getText();

    int idSkladnika = pobierzIdSkladnikaZBazy(nazwaSkladnika);

    if (idSkladnika != -1) {
        skladnikiList.add(idSkladnika);
    } else {
        JOptionPane.showMessageDialog( parentComponent: null, message: "Nie znaleziono skladnika o nazwie: " + nazwaSkladnika);
    }
}
}

1 usage
private void cofnij(int id_uzytkownika, int id_planu){
    JFrame frame = (JFrame) SwingUtilities.getWindowAncestor(getPanel());
    frame.dispose();
    JFrame utworzFrame = new JFrame( title: "Personalizacja");
    personalizacja ekran = new personalizacja(id_uzytkownika, id_planu);
    utworzFrame.setContentPane(ekran.getPanel());
    utworzFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    utworzFrame.pack();
    utworzFrame.setVisible(true);
}

public JPanel getPanel() { return dodaniePrzepisowPanel; }

```

Klasa ekranGlowny

```

ekranGlowny(int id_uzytkownika, int id_planu){
    wyswietlButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) { wyswietlPlan(id_uzytkownika, id_planu); }
    });
    cofnijButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) { cofnij(id_uzytkownika, id_planu); }
    });
}

1 usage
public void wyswietlPlan(int id_uzytkownika, int id_planu) {

    try (Connection conn = DriverManager.getConnection(url, user, pass)) {
        String selectIdPlanuSQL = "SELECT id_planu FROM Uzytkownik WHERE id_uzytkownika = ?";
        try (PreparedStatement selectIdPlanuStmt = conn.prepareStatement(selectIdPlanuSQL)) {
            selectIdPlanuStmt.setInt( parameterIndex: 1, id_uzytkownika);

            try (ResultSet resultSet = selectIdPlanuStmt.executeQuery()) {
                if (resultSet.next()) {
                    wyswietlNazwePlanu(conn, id_planu);
                } else {
                    System.out.println("Brak id_planu dla uzytkownika o id_uzytkownika: " + id_uzytkownika);
                }
            }
        }
    }
} catch (SQLException e) {
    e.printStackTrace();
}
}

```

```

public void wyswietlNazwePlanu(Connection conn, int id_planu) throws SQLException {
    String selectNazwaPlanuSQL = "SELECT nazwa_planu FROM Plan_posilkow WHERE id_planu = ?";
    try (PreparedStatement selectNazwaPlanuStmt = conn.prepareStatement(selectNazwaPlanuSQL)) {
        selectNazwaPlanuStmt.setInt( parameterIndex: 1, id_planu);
    }
    try (ResultSet resultSet = selectNazwaPlanuStmt.executeQuery()) {
        if (resultSet.next()) {
            String nazwaPlanu = resultSet.getString( columnLabel: "nazwa_planu");
            textArea1.setText(nazwaPlanu);
            String posilkiSQL = "SELECT Posilki.id_posilku, Posilki.nazwa_posilku, Posilki.kcal_posilku, Posilki.t
                                \"JOIN Posilki_Plan_posilkow ON Posilki.id_posilku = Posilki_Plan_posilkow.Posilkiid_posilku \"
                                \"WHERE Posilki_Plan_posilkow.Plan_posilkowid_planu = ?\"";
            try (PreparedStatement posilkiStmt = conn.prepareStatement(posilkiSQL)) {
                posilkiStmt.setInt( parameterIndex: 1, id_planu);
                try (ResultSet posilkiResultSet = posilkiStmt.executeQuery()) {
                    StringBuilder posilkiStringBuilder = new StringBuilder();
                    while (posilkiResultSet.next()) {
                        String nazwaPosilku = posilkiResultSet.getString( columnLabel: "nazwa_posilku");
                        int kcalPosilku = posilkiResultSet.getInt( columnLabel: "kcal_posilku");
                        float tluszczPosilku = posilkiResultSet.getFloat( columnLabel: "tluszcz_posilku");
                        float bialkaPosilku = posilkiResultSet.getFloat( columnLabel: "bialka_posilku");
                        float wegglePosilku = posilkiResultSet.getFloat( columnLabel: "weglowodany_posilku");
                        float solPosilku = posilkiResultSet.getFloat( columnLabel: "sol_posilku");
                        posilkiStringBuilder.append(" - Posilek: ").append(nazwaPosilku)
                            .append(", kcal: ").append(kcalPosilku)
                            .append(", tluszcz: ").append(tluszczPosilku)
                            .append(", bialka: ").append(bialkaPosilku)
                            .append(", weglowodany: ").append(wegglePosilku)
                            .append(", sol: ").append(solPosilku)
                            .append("\n");
                        wyswietlSkladniki(posilkiResultSet.getInt( columnLabel: "id_posilku"));
                        posilkiStringBuilder.append("\n");
                    }
                    textArea1.setText(posilkiStringBuilder.toString());
                }
            }
        } else {
            textArea1.setText("Brak planu o id_planu: " + id_planu);
        }
    }
}

```

```

public void wyswietlSkladniki(int id_posilku) {

    try (Connection conn = DriverManager.getConnection(url, user, pass)) {
        String skladnikiSQL = "SELECT Skladniki.nazwa_skladnika, Skladniki.kcal, Skladniki.tluszcz, Skladniki.bialka, Skladniki.weglowodany, Skladniki.sol
        JOIN Posilki_Skladniki ON Skladniki.id_skladnika = Posilki_Skladniki.Skladnikiid_skladnika
        WHERE Posilki_Skladniki.Posilkiid_posilku = ?";
        try (PreparedStatement skladnikiStmt = conn.prepareStatement(skladnikiSQL)) {
            skladnikiStmt.setInt(1, id_posilku);

            try (ResultSet skladnikiResultSet = skladnikiStmt.executeQuery()) {
                StringBuilder skladnikiStringBuilder = new StringBuilder();
                while (skladnikiResultSet.next()) {
                    String nazwaSkladnika = skladnikiResultSet.getString(1);
                    int kcalSkladnika = skladnikiResultSet.getInt(2);
                    float tluszczSkladnika = skladnikiResultSet.getFloat(3);
                    float bialkaSkladnika = skladnikiResultSet.getFloat(4);
                    float weggleSkladnika = skladnikiResultSet.getFloat(5);
                    float solSkladnika = skladnikiResultSet.getFloat(6);
                    skladnikiStringBuilder.append("- Skladnik: ").append(nazwaSkladnika)
                        .append(", kcal: ").append(kcalSkladnika)
                        .append(", tluszcz: ").append(tluszczSkladnika)
                        .append(", bialka: ").append(bialkaSkladnika)
                        .append(", weglowodany: ").append(weggleSkladnika)
                        .append(", sol: ").append(solSkladnika)
                        .append("\n\n");
                }
                textArea2.append(skladnikiStringBuilder.toString());
            }
            System.out.println(skladnikiStringBuilder.toString());
        }
    }
} catch (SQLException e) {
    e.printStackTrace();
}
}

```

```

usage
public void cofnij(int id_uzytkownika, int id_planu){
    JFrame frame = (JFrame) SwingUtilities.getWindowAncestor(getPanel());
    frame.dispose();

}

public JPanel getPanel() { return ekranGlownyPanel; }

```

Klasa personalizacja

```

public personalizacja(int id_uzytkownika, int id_planu){
    cofnijButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) { cofnij(id_uzytkownika, id_planu); }
    });
    button1.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) { dodanieDanych(id_uzytkownika, id_planu); }
    });
    button2.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) { dodaniePrzepisow(id_uzytkownika, id_planu); }
    });
    button3.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) { zmianaPlanu(id_uzytkownika, id_planu); }
    });
}

1 usage
private void dodanieDanych(int id_uzytkownika, int id_planu){
    JFrame frame = (JFrame) SwingUtilities.getWindowAncestor(getPanel());
    frame.dispose();

    JFrame utworzFrame = new JFrame( title: "Dodanie danych uzytkownika");
    dodanieDanych ekran = new dodanieDanych(id_uzytkownika, id_planu);
    utworzFrame.setContentPane(ekran.getPanel());
    utworzFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    utworzFrame.pack();
    utworzFrame.setVisible(true);
}

```

```

private void dodaniePrzepisow(int id_uzytkownika, int id_planu){
    JFrame frame = (JFrame) SwingUtilities.getWindowAncestor(getPanel());
    frame.dispose();

    JFrame utworzFrame = new JFrame( title: "Dodanie własnych przepisow");
    dodaniePrzepisow ekran = new dodaniePrzepisow(id_uzytkownika, id_planu);
    utworzFrame.setContentPane(ekran.getPanel());
    utworzFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    utworzFrame.pack();
    utworzFrame.setVisible(true);
}

1 usage
private void zmianaPlanu(int id_uzytkownika, int id_planu){
    JFrame frame = (JFrame) SwingUtilities.getWindowAncestor(getPanel());
    frame.dispose();

    JFrame utworzFrame = new JFrame( title: "Zmiana planu posilkow");
    zmianaPlanu ekran = new zmianaPlanu(id_uzytkownika, id_planu);
    utworzFrame.setContentPane(ekran.getPanel());
    utworzFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    utworzFrame.pack();
    utworzFrame.setVisible(true);
}

1 usage
private void cofnij(int id_uzytkownika, int id_planu){
    JFrame frame = (JFrame) SwingUtilities.getWindowAncestor(getPanel());
    frame.dispose();

    JFrame utworzFrame = new JFrame( title: "Menu");
    menu ekran = new menu(id_uzytkownika, id_planu);
    utworzFrame.setContentPane(ekran.getPanel());
    utworzFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    utworzFrame.pack();
    utworzFrame.setVisible(true);
}

public JPanel getPanel() { return personalizacjaPanel; }

```

Klasa przegladPosilkow


```

1 usage
public przegladPosilkow(int id_uzytkownika, int id_planu){
    cofnijButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) { cofnij(id_uzytkownika, id_planu); }
    });
    wyswietlPrzepisyButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) { wyswietlPrzepisy(); }
    });
}

```

```

private void wyswietlPrzepisy() {
    try {
        String url = "jdbc:mysql://localhost:3306/sys";
        String user = "root";
        String pass = "Haslo123.";

        Connection conn = DriverManager.getConnection(url, user, pass);

        String selectPrzepisySQL = "SELECT nazwa_posilku, kcal_posilku, tluszcz_posilku, bialka_posilku, weglowodan";

        try (PreparedStatement selectPrzepisyStmt = conn.prepareStatement(selectPrzepisySQL)) {
            ResultSet rs = selectPrzepisyStmt.executeQuery();

            StringBuilder posilkiStringBuilder = new StringBuilder();

            while (rs.next()) {
                String nazwaPosilku = rs.getString( columnLabel: "nazwa_posilku");
                int kcalPosilku = rs.getInt( columnLabel: "kcal_posilku");
                float tluszczPosilku = rs.getFloat( columnLabel: "tluszcz_posilku");
                float bialkaPosilku = rs.getFloat( columnLabel: "bialka_posilku");
                float wegglePosilku = rs.getFloat( columnLabel: "weglowodany_posilku");
                float solPosilku = rs.getFloat( columnLabel: "sol_posilku");

                posilkiStringBuilder.setLength(0);
                posilkiStringBuilder.append("Nazwa: ").append(nazwaPosilku).append(", Kcal: ").append(kcalPosilku)
                    .append(", kcal: ").append(kcalPosilku)
                    .append(", tluszcz: ").append(tluszczPosilku)
                    .append(", bialka: ").append(bialkaPosilku)
                    .append(", weglowodany: ").append(wegglePosilku)
                    .append(", sol: ").append(solPosilku)
                    .append("\n");
                wyswietlonePosilki.append(String.valueOf(posilkiStringBuilder));

                dodajSkladnikiDoPrzepisu(conn, nazwaPosilku);
                wyswietlonePosilki.append("\n");
            }

            conn.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}

```

```

private void dodajSkladnikiDoPrzepisu(Connection conn, String nazwaPosilku) {
    try {
        String selectSkladnikiSQL = "SELECT nazwa_skladnika, kcal, tluszcz, bialka, weglowodany, sol FROM Skladniki " +
            "JOIN Posilki_Skladniki ON Skladniki.id_skladnika = Posilki_Skladniki.Skladnikiid_skladnika " +
            "JOIN Posilki ON Posilki_Skladniki.Posilkiid_posilku = Posilki.id_posilku " +
            "WHERE Posilki.nazwa_posilku = ?";

        try (PreparedStatement selectSkladnikiStmt = conn.prepareStatement(selectSkladnikiSQL)) {
            selectSkladnikiStmt.setString( parameterIndex: 1, nazwaPosilku);
            ResultSet skladnikiRs = selectSkladnikiStmt.executeQuery();

            StringBuilder skladnikiStringBuilder = new StringBuilder();

            while (skladnikiRs.next()) {
                String nazwaSkladnika = skladnikiRs.getString( columnLabel: "nazwa_skladnika");
                int kcalSkladnika = skladnikiRs.getInt( columnLabel: "kcal");
                float tluszczSkladnika = skladnikiRs.getFloat( columnLabel: "tluszcz");
                float bialkaSkladnika = skladnikiRs.getFloat( columnLabel: "bialka");
                float wegleSkladnika = skladnikiRs.getFloat( columnLabel: "weglowodany");
                float solSkladnika = skladnikiRs.getFloat( columnLabel: "sol");
                skladnikiStringBuilder.append("\t- Skladnik: ").append(nazwaSkladnika).append(", Kcal: ").append(kcalSkladnika)
                    .append(", tluszcz: ").append(tluszczSkladnika)
                    .append(", bialka: ").append(bialkaSkladnika)
                    .append(", weglowodany: ").append(wegleSkladnika)
                    .append(", sol: ").append(solSkladnika)
                    .append("\n");
            }
            wyswietlonePosilki.append(skladnikiStringBuilder.toString());
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

```

```

public JPanel getPanel() { return przegladPanel; }

1 usage
private void cofnij(int id_uzytkownika, int id_planu){
    JFrame frame = (JFrame) SwingUtilities.getWindowAncestor(getPanel());
    frame.dispose();
}

```

Klasa utworzUzytkownika

```

public utworzUzytkownika(){
    utworzButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            if(utworzUzytkownik()) {
                JOptionPane.showMessageDialog( parentComponent: null, message: "Pomyślnie utworzono użytkownika.");
                wylogujSie();
            } else {
                JOptionPane.showMessageDialog( parentComponent: null, message: "Błędne dane.");
            }
        }
    });
}

```

```

public boolean utworzUzytkownik(){
    String url = "jdbc:mysql://localhost:3306/sys";
    String user = "root";
    String pass = "Haslo123.";
    if (getImie().isEmpty() || getNazwisko().isEmpty() || getEmail().isEmpty() || getHaslo().isEmpty()) {return false;}
    try {
        getWiek();
        getWzrost();
        getWaga();
    } catch (NumberFormatException e) {
        JOptionPane.showMessageDialog( parentComponent: null, message: "Wprowadź poprawne wartości liczbowe.");
        return false;
    }
    try (Connection conn = DriverManager.getConnection(url, user, pass)) {
        int idPlanu = utworzPlan(conn);
        String sql = "INSERT INTO Uzytkownik (imie, nazwisko, wiek, wzrost, waga, email, haslo, plec, id_planu) VALUES ("
        try (PreparedStatement statement = conn.prepareStatement(sql)) {
            statement.setString( parameterIndex: 1, getImie());
            statement.setString( parameterIndex: 2, getNazwisko());
            statement.setInt( parameterIndex: 3, getWiek());
            statement.setInt( parameterIndex: 4, getWzrost());
            statement.setInt( parameterIndex: 5, getWaga());
            statement.setString( parameterIndex: 6, getEmail());
            statement.setString( parameterIndex: 7, getHaslo());
            statement.setString( parameterIndex: 8, getPlec());
            statement.setInt( parameterIndex: 9, idPlanu);

            int rowsAffected = statement.executeUpdate();
            if (rowsAffected > 0) {
                edytujNazwePlanu(conn, idPlanu, getImie(), getNazwisko());
                return true;
            }
        }
    } catch (SQLException ex) {
        ex.printStackTrace();
        return false;
    }
    return false;
}

```

```

private int utworzPlan(Connection conn) throws SQLException {
    String insertPlanSQL = "INSERT INTO Plan_posilkow (nazwa_planu) VALUES (?)";
    try (PreparedStatement insertPlanStmt = conn.prepareStatement(insertPlanSQL, Statement.RETURN_GENERATED_KEYS)) {
        insertPlanStmt.setString(1, "Plan uzytkownika");
        int rowsAffected = insertPlanStmt.executeUpdate();

        if (rowsAffected > 0) {
            try (ResultSet generatedKeys = insertPlanStmt.getGeneratedKeys()) {
                if (generatedKeys.next()) {
                    return generatedKeys.getInt(1);
                }
            }
        }
    }

    return -1;
}

```

1 usage

```

private boolean edytujNazwePlanu(Connection conn, int idPlanu, String imie, String nazwisko) {
    String nowaNazwaPlanu = "Planm uzytkownika "+imie+" "+nazwisko;

    String updatePlanSQL = "UPDATE Plan_posilkow SET nazwa_planu = ? WHERE id_planu = ?";
    try (PreparedStatement updatePlanStmt = conn.prepareStatement(updatePlanSQL)) {
        updatePlanStmt.setString(1, nowaNazwaPlanu);
        updatePlanStmt.setInt(2, idPlanu);

        int rowsAffected = updatePlanStmt.executeUpdate();
        return rowsAffected > 0;
    } catch (SQLException ex) {
        ex.printStackTrace();
        return false;
    }
}

```

```

private void wylogujSie(){
    JFrame frame = (JFrame) SwingUtilities.getWindowAncestor(getPanel());
    frame.dispose();

    JFrame utworzFrame = new JFrame( title: "Logowanie");
    logowanie ekran = new logowanie();
    utworzFrame.setContentPane(ekran.getPanel());
    utworzFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    utworzFrame.pack();
    utworzFrame.setVisible(true);
}

3 usages
public String getImie() { return imie.getText(); }

3 usages
private String getNazwisko() { return nazwisko.getText(); }

2 usages
private int getWiek() { return Integer.parseInt(wiek.getText()); }

2 usages
private int getWzrost() { return Integer.parseInt(wzrost.getText()); }

2 usages
private int getWaga() { return Integer.parseInt(waga.getText()); }

2 usages
private String getEmail() { return email.getText(); }

2 usages
private String getHaslo() { return haslo.getText(); }

1 usage
private String getPlec() { return plec.getText(); }

private int getIdPlanu() { return Integer.parseInt(id_planu.getText()); }

public JPanel getPanel() { return panel2; }

```

Klasa zmianaPlanu

```

1 usage
public zmianaPlanu(int id_uzytkownika, int id_planu){
    zapiszButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) { zapisz(id_uzytkownika, id_planu); }
    });
    button2.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) { cofnij(id_uzytkownika, id_planu); }
    });
}

```

```

private void zapisz(int id_uzytkownika, int id_planu) {
    String posilekDoUsuniecia = doUsuniecia.getText().trim();
    String posilekDoDodania = doDodania.getText().trim();

    try (Connection conn = DriverManager.getConnection(url, user, pass)) {
        conn.setAutoCommit(false);

        try {
            String usunPosilekSQL = "DELETE FROM Posilki_Plan_posilkow WHERE Posilkiid_posilku IN " +
                "(SELECT id_posilku FROM Posilki WHERE nazwa_posilku = ?)";
            try (PreparedStatement usunPosilekStmt = conn.prepareStatement(usunPosilekSQL)) {
                usunPosilekStmt.setString(1, posilekDoUsuniecia);
                usunPosilekStmt.executeUpdate();
            }
            String dodajPosilekSQL = "INSERT INTO Posilki_Plan_posilkow (Posilkiid_posilku, Plan_posilkowid_planu) " +
                "VALUES ((SELECT id_posilku FROM Posilki WHERE nazwa_posilku = ?), ?)";
            try (PreparedStatement dodajPosilekStmt = conn.prepareStatement(dodajPosilekSQL)) {
                dodajPosilekStmt.setString(1, posilekDoDodania);
                dodajPosilekStmt.setInt(2, id_uzytkownika);
                dodajPosilekStmt.executeUpdate();
            }

            conn.commit();
        } catch (SQLException ex) {
            JOptionPane.showMessageDialog(parentComponent, null, "Błędnie wpisane posiłki.");
            conn.rollback();
            ex.printStackTrace();
        }
    } catch (SQLException ex) {
        ex.printStackTrace();
    }
}

```

```

1 usage
private void cofnij(int id_uzytkownika, int id_planu){
    JFrame frame = (JFrame) SwingUtilities.getWindowAncestor(getPanel());
    frame.dispose();
    JFrame utworzFrame = new JFrame(title: "Personalizacja");
    personalizacja_ekran = new personalizacja(id_uzytkownika, id_planu);
    utworzFrame.setContentPane(ekran.getPanel());
    utworzFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    utworzFrame.pack();
    utworzFrame.setVisible(true);
}

public JPanel getPanel() { return zmianaPlanuPanel; }

```

5. Testy aplikacji

Test poprawnego logowania

```
@Test
public void testLogowania() {
    logowanie loginScreen = new logowanie();

    loginScreen.loginField.setText("test");
    loginScreen.hasloField.setText("test");

    assertTrue(loginScreen.sprawdzHasloWBazie( loginScreen.loginField.getText(), loginScreen.hasloField.getText()));
}
```

✓ Tests passed: 1 of 1 test – 638 ms

Test niepoprawnego logowania

```
@Test
public void testZleLogowania() {
    logowanie loginScreen = new logowanie();

    loginScreen.loginField.setText("testzly");
    loginScreen.hasloField.setText("testzly");

    assertFalse(loginScreen.sprawdzHasloWBazie( loginScreen.loginField.getText(), loginScreen.hasloField.getText()));
}
```

✓ Tests passed: 1 of 1 test – 654 ms

Test poprawnego dodania składnika

```
@Test
public void testDodajSkladnik() {
    dodaniePrzepisow dodaniePrzepisow = new dodaniePrzepisow( id_uzytkownika: 1, id_planu: 1);

    dodaniePrzepisow.skladnikPane.setText("Pomidor");
    dodaniePrzepisow.iloscPane.setText("2");

    dodaniePrzepisow.dodajSkladnik();

    assertTrue(dodaniePrzepisow.skladnikiList.contains(1));
}
```

✓ Tests passed: 1 of 1 test – 523 ms

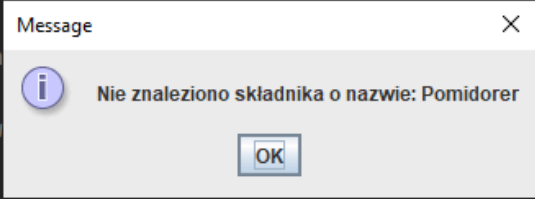
Test niepoprawnego poprawnego dodania składnika

```
@Test
public void testDodajZlySkladnik() {
    dodaniePrzepisow dodaniePrzepisow = new dodaniePrzepisow( id_uzytkownika: 1, id_planu: 1);

    dodaniePrzepisow.skladnikPane.setText("Pomidorer");
    dodaniePrzepisow.iloscPane.setText("2");

    dodaniePrzepisow.dodajSkladnik();

    assertFalse(dodaniePrzepisow.wartosciLabel.getText().contains("Kcal:"));
}
```

A message dialog box with a title bar "Message" and a close button "X". It contains an information icon, the text "Nie znaleziono skladnika o nazwie: Pomidorer", and an "OK" button.

Test poprawnego dodania przepisu do bazy danych

```
@Test
public void testZapiszPrzepis() {
    dodaniePrzepisow dodaniePrzepisow = new dodaniePrzepisow( id_uzytkownika: 1, id_planu: 1);

    dodaniePrzepisow.nazwaPrzepisuPane.setText("Jajeczniczka");
    dodaniePrzepisow.skladnikPane.setText("Pomidor");
    dodaniePrzepisow.iloscPane.setText("2");

    dodaniePrzepisow.zapisz();

    assertTrue(dodaniePrzepisow.wartosciLabel.getText().contains("Kcal:"));
}
```

Tests passed: 1 of 1 test – 604 ms

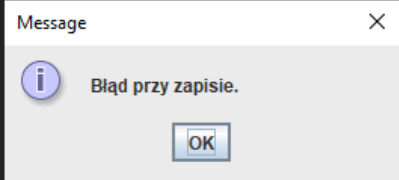
Test niepoprawnego dodania przepisu do bazy danych

```
@Test
public void testZapiszZlyPrzepis() {
    dodaniePrzepisow dodaniePrzepisow = new dodaniePrzepisow( id_uzytkownika: 1, id_planu: 1);

    dodaniePrzepisow.nazwaPrzepisuPane.setText("Jajecznicza");
    dodaniePrzepisow.skladnikPane.setText("Pomidor");
    dodaniePrzepisow.iloscPane.setText("2");

    dodaniePrzepisow.zapisz();

    assertFalse(dodaniePrzepisow.wartosciLabel.getText().contains("Kcal:"));
}
```

A message dialog box with a title bar "Message" and a close button "X". It contains an information icon, the text "Błąd przy zapisie.", and an "OK" button.

Test poprawnego zapisu danych osobowych


```

@Test
public void testZapiszDane() {
    int id_uzytkownika = 1;
    int id_planu = 1;

    dodanieDanych dodanieDane = new dodanieDanych(id_uzytkownika, id_planu);

    dodanieDane.textPane1.setText("31");
    dodanieDane.textPane2.setText("M");
    dodanieDane.textPane3.setText("180");
    dodanieDane.textPane4.setText("75");
    dodanieDane.comboBox1.setSelectedItem("Brak alergii");
    dodanieDane.comboBox3.setSelectedItem("3");
    dodanieDane.comboBox4.setSelectedItem("Redukcja");

    dodanieDane.zapisz(id_uzytkownika, id_planu);
    assertDoesNotThrow(() -> dodanieDane.zapisz(id_uzytkownika, id_planu));
}

```

✓ Tests passed: 1 of 1 test – 760 ms

Test niepoprawnego zapisu danych osobowych

```

@Test
public void testZapiszZleDane() {
    int id_uzytkownika = 1;
    int id_planu = 1;

    dodanieDanych dodanieDane = new dodanieDanych(id_uzytkownika, id_planu);

    dodanieDane.textPane1.setText("13");
    dodanieDane.textPane2.setText("M");
    dodanieDane.textPane3.setText("180");
    dodanieDane.textPane4.setText("75");
    dodanieDane.comboBox1.setSelectedItem("Brak alergii");
    dodanieDane.comboBox3.setSelectedItem("3");
    dodanieDane.comboBox4.setSelectedItem("Redukcja");

    dodanieDane.zapisz(id_uzytkownika, id_planu);
    assertDoesNotThrow(() -> dodanieDane.zapisz(id_uzytkownika, id_planu));
}

```

