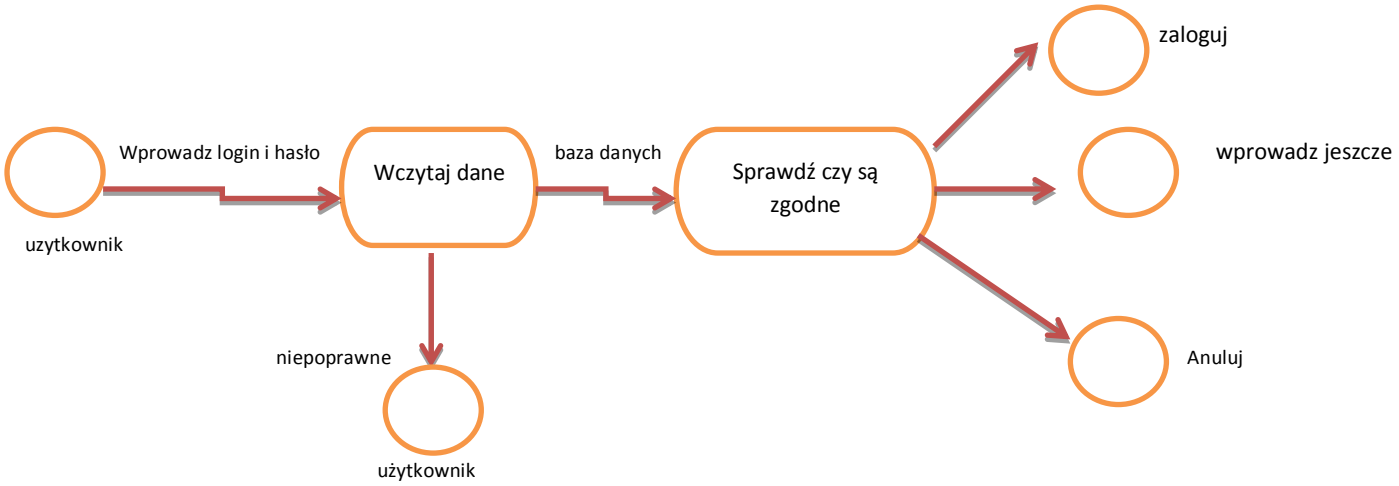
IMPLEMENTACJA FUNKCJONALNOŚCI

1. Logowanie

Sprawdzanie poprawności logowania.

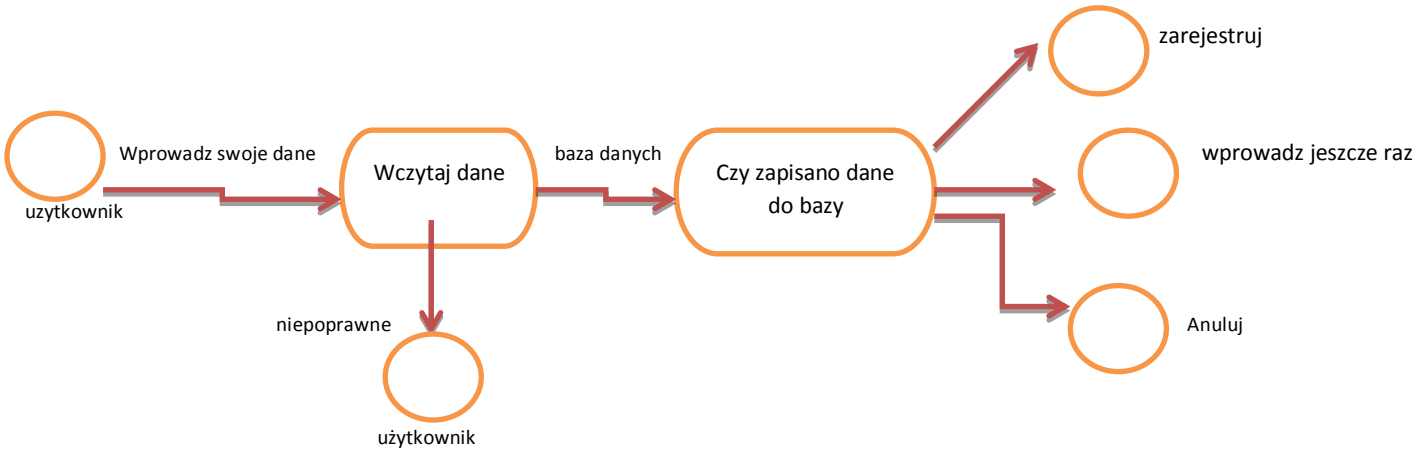
Czy w bazie danych istnieje użytkownik o danym loginie i haśle.




```
public boolean Logowanie(String Login, String Hasło) throws SQLException {
Statement stmt = con.createStatement();
String login = null, hasło = null;

ResultSet rs = stmt.executeQuery("SELECT * from Uzytkownicy");
int i = 0;
while (rs.next()) {
        login = rs.getString(8);

        if (Login.equals(login)) {
                poprawne_Login = true;
                numer = rs.getInt(1);
                uprawnienia = rs.getString(7);
        }
}
rs = stmt.executeQuery("SELECT * from Uzytkownicy");

while (rs.next()) {
        hasło = rs.getString(9);

        if (Hasło.equals(hasło)) {
                poprawne_Hasło = true;
                numer = rs.getInt(1);
                uprawnienia = rs.getString(7);
                // System.err.println(uprawnienia+numer_uzytkownika);
        }
}
boolean zgoda = ((poprawne_Hasło) && (poprawne_Login));

// System.out.println("Zgoda : "+zgoda);
return zgoda;
}
```

2. Rejestracja

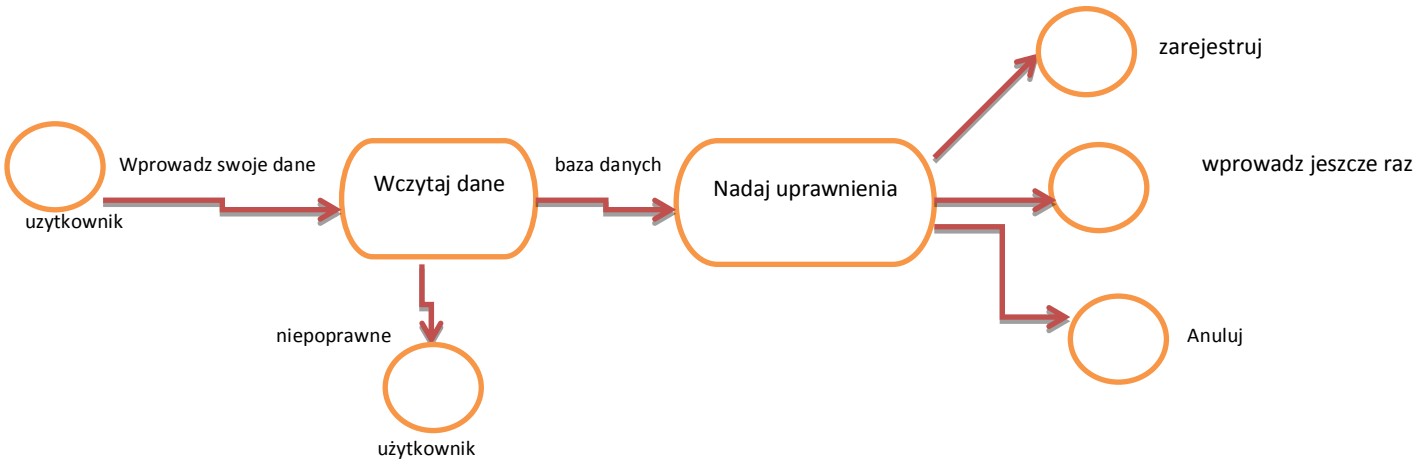
Podanie podstawowych danych do systemu, dzięki którym uzyskuje się możliwość logowania.




```
try {
 JT_Tabela_uzytkowników = log.Rejestracja(imie, nazwisko, adres, email, numer_telfonu, login,haslo);
} catch (SQLException e2) {e2.printStackTrace();}
```

3. Rejestracja admina

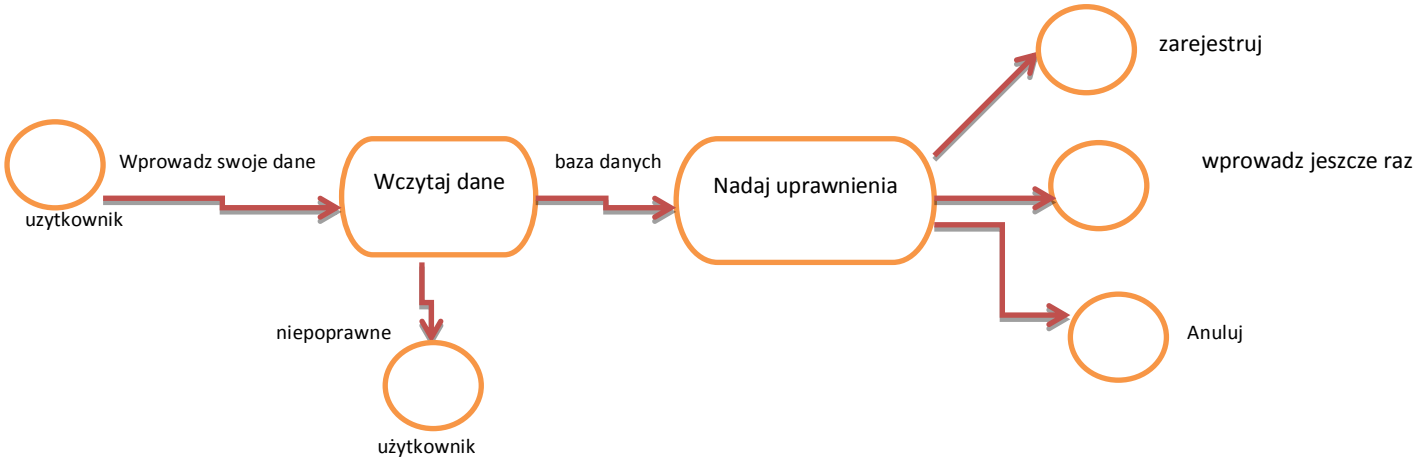
Przełączenie uprawnień na aminowskie.




```
else if (Uprawnienia.equals("Administrator")) {
                        Numer_uztkownika = numer;

                        String[] tablica_dane = algoLog.Dane(Numer_uztkownika);

                        StartDietaTrening(tablica_dane[1], tablica_dane[2], tablica_dane[3], tablica_dane[4], tablica_dane[5],
                                        tablica_dane[6], tablica_dane[7]);
                }
```

4. Rejestracja zwykłego użytkownika

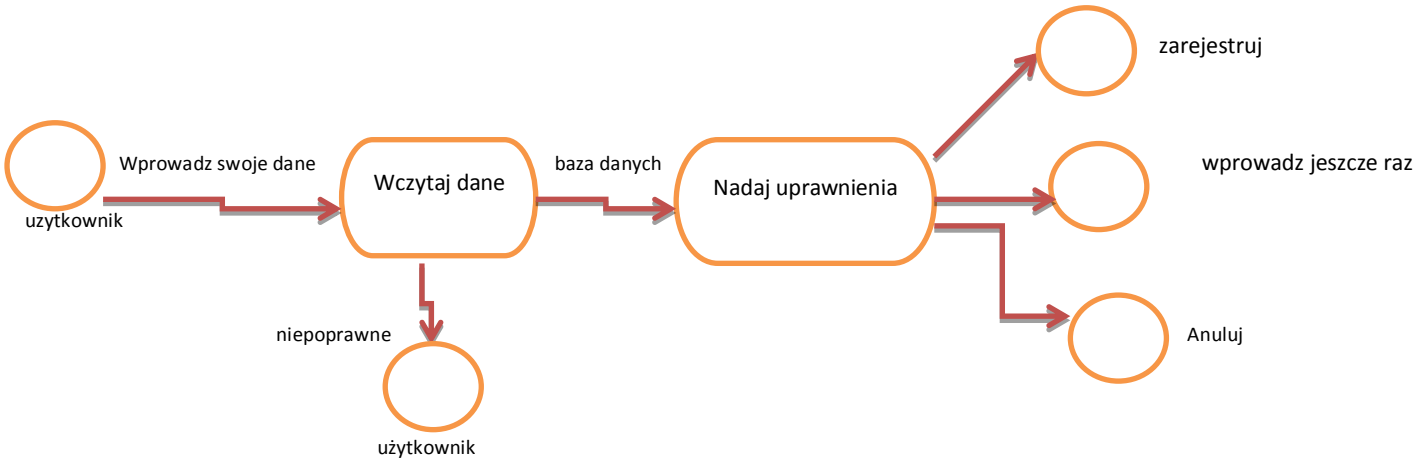
Przełączenie uprawnień na zwykłego użytkownika.




```
if (Uprawnienia.equals("Użytkownik")) {

Numer_uztkownika = numer;

String[] tablica_dane = algoLog.Dane(Numer_uztkownika);
StartDietaTrening(tablica_dane[1], tablica_dane[2], tablica_dane[3], tablica_dane[4], tablica_dane[5],
                  tablica_dane[6], tablica_dane[7]);
btnOdczytajWiadomosc.setVisible(false);
taskPaneGroup_Dodaj_Usun.setVisible(false);
```

5. Rejestracja trenera

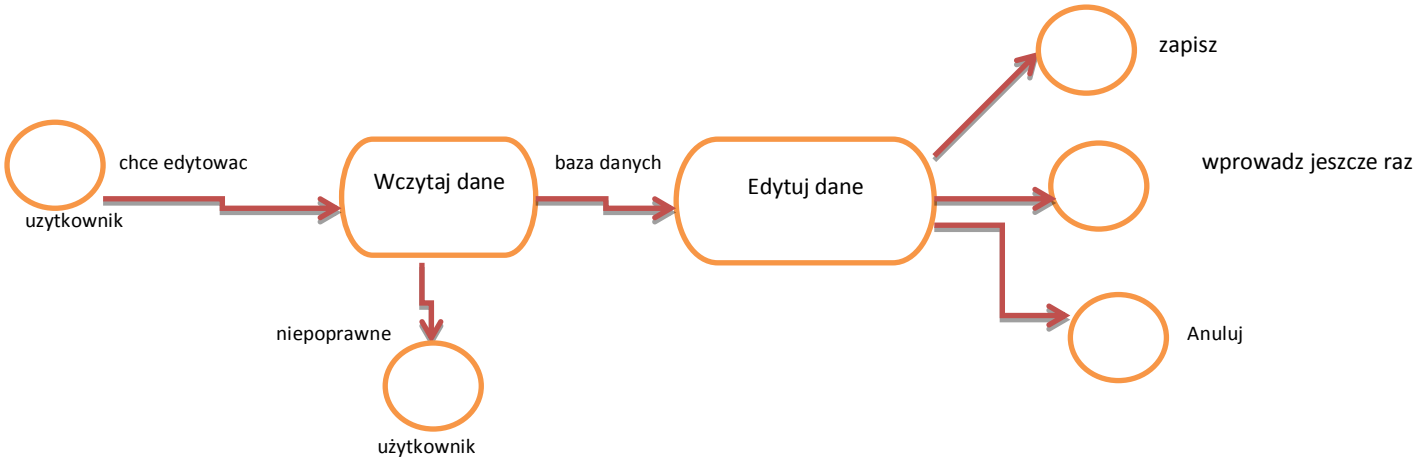
Przełączenie uprawnień na trenera.




```
else if (Uprawnienia.equals("Trener")) {
Numer_uztkownika = numer;

String[] tablica_dane = algoLog.Dane(Numer_uztkownika);
StartDietaTrening(tablica_dane[1], tablica_dane[2], tablica_dane[3], tablica_dane[4], tablica_dane[5],
                  tablica_dane[6], tablica_dane[7]);
btnOdczytajWiadomosc.setVisible(false);
```

6. Edycja danych użytkownika

Metoda dzięki której edytujemy dane poza logowaniu.

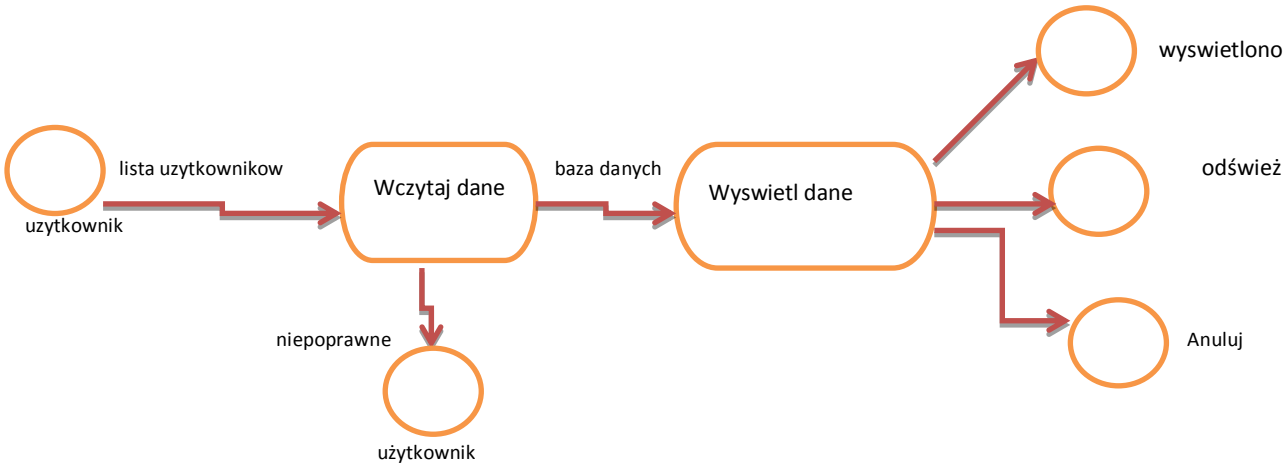



```
public JTable Edytuj(int numer, String imie, String nazwisko, String adres, String email, String nr_tlefonu,
        String login, String haslo, String Uprawnienia) throws SQLException {

// Connection con = DriverManager.getConnection(
// "jdbc:sybase:Tds:localhost:2638", "DBA", "sql");
Statement stmt = con.createStatement();
ResultSet rs = stmt.executeQuery("SELECT count(0) from Uzytkownicy");
JTable result;

int iterator_wierszy = 0;
rs.next();
iterator_wierszy = rs.getInt(1);

rs = stmt.executeQuery("SELECT * from Uzytkownicy");

/**
 * Create the Table
 */

String[] colNames = new String[] { "NR", "imie", "Nazwisko", "Adres", "Telefon", "Email", "Uprawnienia",
                "Login", "Hasło" };

AbstractTableModel model = new DefaultTableModel(colNames, iterator_wierszy) {
        /**
         *
         */
        private static final long serialVersionUID = 1L;

        public boolean isCellEditable(int row, int column) {
                return false;
        };
};
result = new JTable(model);
result.getColumnModel().getColumn(0).setPreferredWidth(80);
result.getColumnModel().getColumn(1).setPreferredWidth(80);
result.setRowHeight(30);
result.setFont(new Font("Arial", 1, 16));
```

```
        sorter = new TableRowSorter<>(result.getModel());

        result.setRowSorter(sorter);

        List<RowSorter.SortKey> sortKeys = new ArrayList<>();

        int columnIndexToSort = 0;
        sortKeys.add(new RowSorter.SortKey(columnIndexToSort, SortOrder.ASCENDING));

        sorter.setSortKeys(sortKeys);

        /**
         * Create the Load Baza.
         */
        String sql = "UPDATE  Uzytkownicy SET Imie='" + imie + "',Nazwisko='" + nazwisko + "',Aderes='" + adres
                                + "',Telefon=" + nr_tlefonu + ",Email='" + email + "',Logi='" + login + "',Haslo='" + haslo
                                + "'WHERE Numer=" + numer;
        stmt = con.createStatement();
        stmt.execute(sql);

        return result;
}
```

7. Lista użytkowników

Pobranie z bazy danych listy użytkowników.

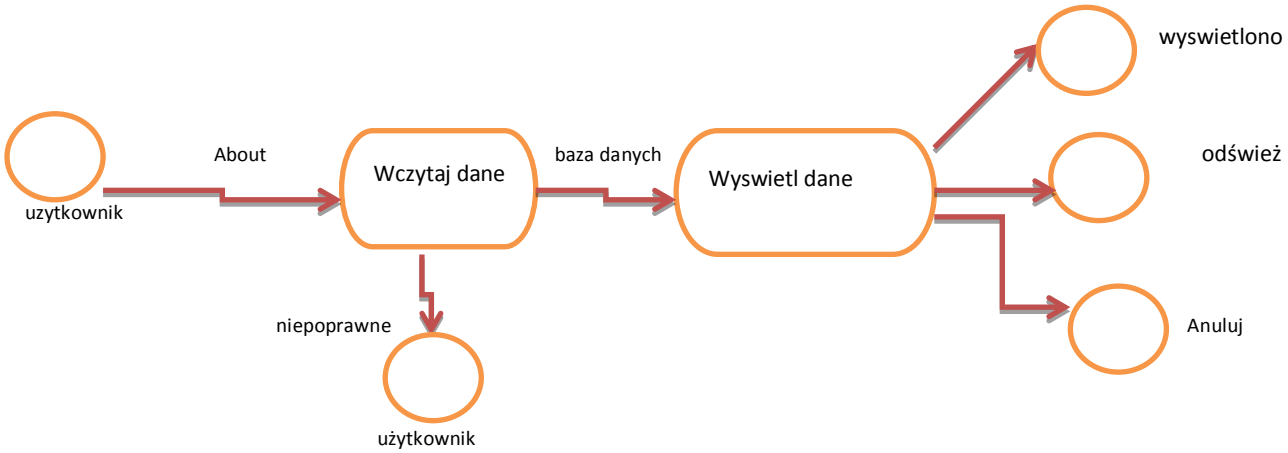



```
        stmt = con.createStatement();
        rs = stmt.executeQuery("SELECT * from Uzytkownicy");

        int it = 0;
        while (rs.next()) {
                uzytkownik.setValueAt(rs.getInt(1), it, 0);
                uzytkownik.setValueAt(rs.getString(2), it, 1);
                uzytkownik.setValueAt(rs.getString(3), it, 2);
                uzytkownik.setValueAt(rs.getString(4), it, 3);
                uzytkownik.setValueAt(rs.getString(5), it, 4);
                uzytkownik.setValueAt(rs.getString(6), it, 5);
                uzytkownik.setValueAt(rs.getString(7), it, 6);
                uzytkownik.setValueAt(rs.getString(8), it, 7);
                uzytkownik.setValueAt(rs.getString(9), it, 8);
```

```
            it++;
        }

        class IntComparator implements Comparator {
            public int compare(Object o1, Object o2) {
                Integer int1 = (Integer) o1;
                Integer int2 = (Integer) o2;
                return int1.compareTo(int2);
            }

            public boolean equals(Object o2) {
                return this.equals(o2);
            }
        }
```

8. Informacje o programie (About)

Tworzenie głównych komponentów w klasie WindowsAbout.

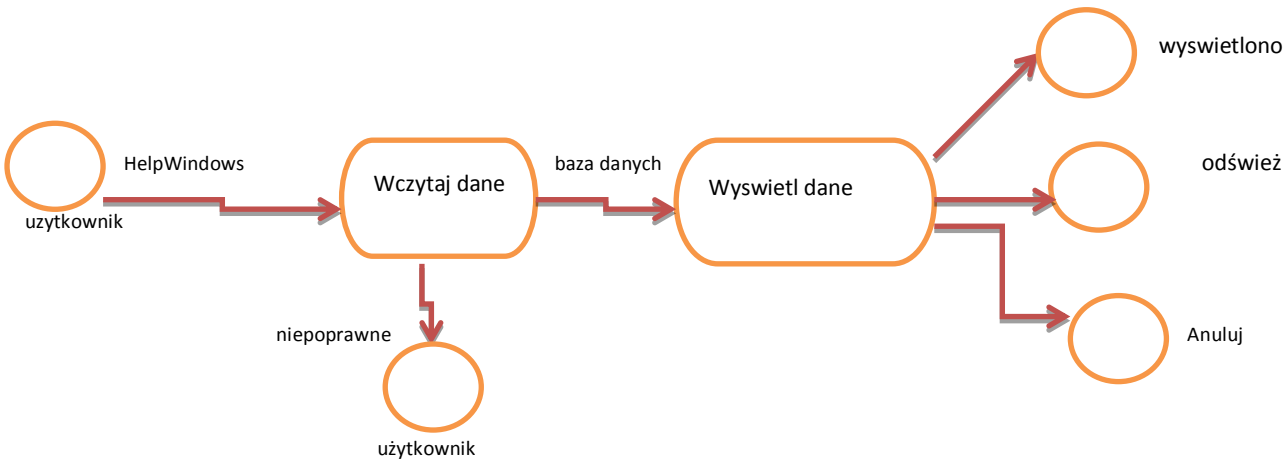



```
            try {
            ImageIcon ikonainformacjeautor = new ImageIcon("fitt.jpg");
            lIcon = new JLabel();
            lIcon.setIcon(new ImageIcon("C:\\Users\\Gregory\\Documents\\JavaEclipse\\Fitnes\\image\\fitt.jpg"));
//                  lIcon = new JLabel(new ImageIcon(
//                          getClass().getResource("author_logo.jpg")));
        }
        catch(Exception e) {
            lIcon = new JLabel();
        }
        jlNazwa_progrmu = new JLabel("Fitt Fitness");
        jlNazwa_progrmu.setFont(font1);
        jlNazwa_progrmu.setHorizontalAlignment(SwingConstants.CENTER);
        jlWersja = new JLabel("wersja 2.2");
        jlWersja.setFont(font1);
        jlWersja.setHorizontalAlignment(SwingConstants.CENTER);
        jlPrawa = new JLabel("Copyright (C) by 2015");
        jlPrawa.setFont(font2);
        jlPrawa.setHorizontalAlignment(SwingConstants.CENTER);
        jlUczelnia = new JLabel("Politechnika Koszalinska - WEiI");
        jlUczelnia.setFont(font3);
        jlUczelnia.setHorizontalAlignment(SwingConstants.CENTER);
```

```
        jlEmail = new JLabel("e-mail: wolosg7@gmail.com");
        jlEmail.setFont(font4);
        lBorder = new JLabel("");
        jBOk = new JButton("Ok");
        jBOk.addActionListener(this);
        line = new EtchedBorder(EtchedBorder.LOWERED);

        lIcon.setBounds(0,-121,559,576);
        jlNazwa_progrmu.setBounds(10,23,210,30);
        jlWersja.setBounds(20,50,210,30);
        jlPrawa.setBounds(10,79,210,20);

        jlUczelnia.setBounds(10,126,186,20);
        lBorder.setBounds(5,185,dialogSize.width-14,2);
        jlEmail.setBounds(10,194,200,20);
        jBOk.setBounds(dialogSize.width-75,192,60,25);
```

9. Pomoc

Tworzenie głównych komponentów w klasie HelpWindows

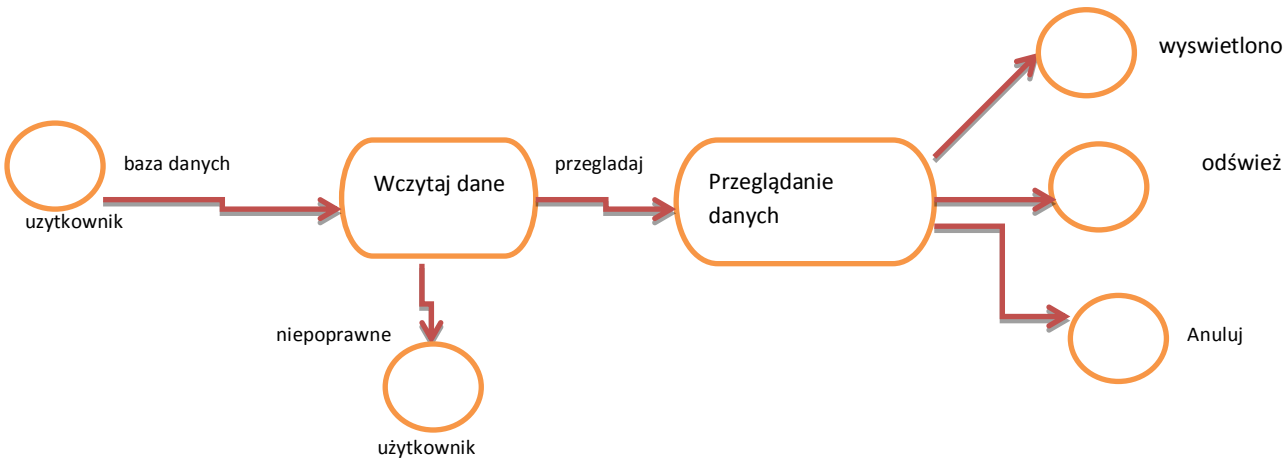



```
        jlNazwa_progrmu = new JLabel("Fitt Fitness");
        jlNazwa_progrmu.setFont(font1);
        jlNazwa_progrmu.setHorizontalAlignment(SwingConstants.CENTER);
        jlWersja = new JLabel("wersja 2.2");
        jlWersja.setFont(font3);
        jlWersja.setHorizontalAlignment(SwingConstants.CENTER);
        jlPrawa = new JLabel("Copyright (C) by 2015");
        jlPrawa.setFont(font2);
        jlPrawa.setHorizontalAlignment(SwingConstants.CENTER);
    jtaTekst = new JTextArea();
    jtaTekst.setEditable(false);
    jtaTekst.setBackground(Color.white);
    jspRolka = new JScrollPane(jtaTekst);
    jspRolka.setViewportView(jtaTekst);
        jlEmail = new JLabel("e-mail: wolosg7@gmail.com");
        jlEmail.setFont(font4);
        lBorder = new JLabel("");
        jbOk = new JButton("Ok");
        jbOk.addActionListener(this);
        line = new EtchedBorder(EtchedBorder.LOWERED);
        lIcon.setBounds(0,-108,618,630);
```

```
jlNazwa_progrmu.setBounds(10,21,210,30);
jlWersja.setBounds(10,46,210,30);
jlPrawa.setBounds(10,74,210,20);
jspRolka.setBounds(135, 140, 450, 330);
lBorder.setBounds(5,475,dialogSize.width-14,2);
jlEmail.setBounds(10,484,200,20);

jbOk.setBounds(dialogSize.width-75,482,60,25);
```

10.Baza produktów – przeglądanie

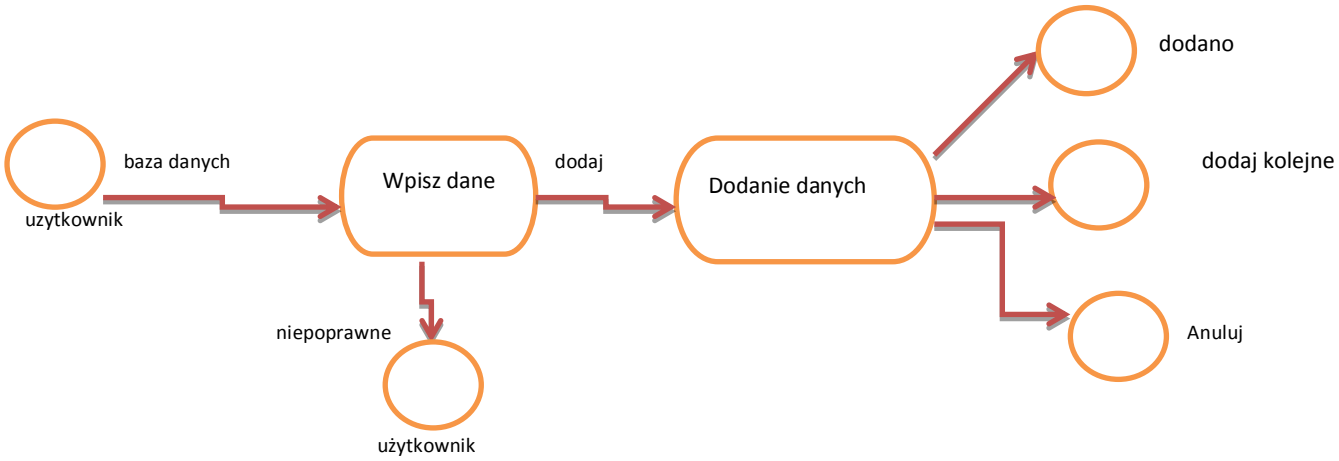
Pobranie z bazy danych i wyświetlenie produktów.




```
stmt = con.createStatement();
rs = stmt.executeQuery("SELECT * from " + nazwa_bazy);

int it = 0;
while (rs.next()) {
        result.setValueAt(rs.getInt(1), it, 0);
        result.setValueAt(rs.getString(2), it, 1);
        result.setValueAt(rs.getInt(3), it, 2);
        result.setValueAt(rs.getInt(4), it, 3);
        result.setValueAt(rs.getInt(5), it, 4);
        result.setValueAt(rs.getInt(6), it, 5);
        it++;
}

class IntComparator implements Comparator {
        public int compare(Object o1, Object o2) {
                Integer int1 = (Integer) o1;
                Integer int2 = (Integer) o2;
                return int1.compareTo(int2);
        }
        public boolean equals(Object o2) {
                return this.equals(o2);
        }
}
sorter.setComparator(0, new IntComparator());
sorter.setComparator(2, new IntComparator());
sorter.setComparator(3, new IntComparator());
sorter.setComparator(4, new IntComparator());
sorter.setComparator(5, new IntComparator())
```

11. Baza produktów – dodanie

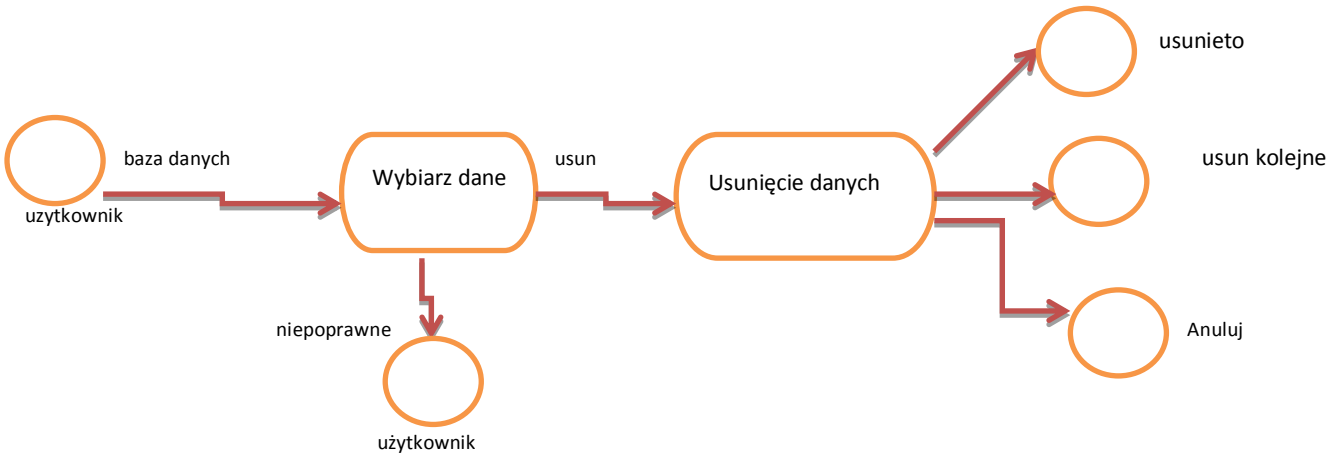
Dodanie produktu do bazy danych.




```
String sql = "INSERT INTO " + nazwa_bazy + "(Numer,Produktu,Kalorie,Białko,Tłuszcze,Węglowodany)Values ("
                    + nowy_numer + ",'" + Nazwa + "'," + Kalorie + "," + Białko + "," + Tłuszcze + "," + Węglowodany + ")";
stmt = con.createStatement();
stmt.execute(sql);
```

12. Baza produktów – usunięcie

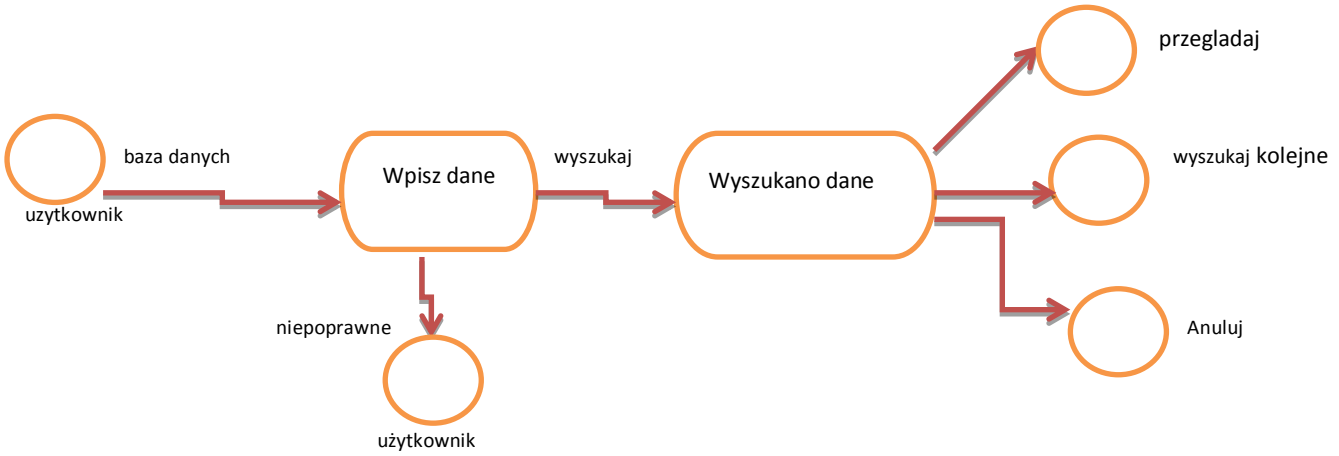
Usunięcie produktu z bazy danych.




```
                    int i = JOptionPane.showConfirmDialog(this,
                    "Czy na pewno chcesz usunąć wiersz " + result.getColumnName(0) + " = " + n, "Potwierdź",
                    JOptionPane.YES_NO_OPTION);

if (i == JOptionPane.YES_OPTION) {
            String sql = "Delete from " + nazwa_bazy + " where Numer=" + n;

            try {
                    stmt.execute(sql);
                    ((DefaultTableModel) model).removeRow(numer);
                    revalidate();
                    repaint();
            } catch (SQLException e) {
                    JOptionPane.showMessageDialog(this, "Wystąpił błąd poczas usuwania.", "Błąd",
                                    JOptionPane.ERROR_MESSAGE);
                    e.printStackTrace(); }}
```

13. Baza produktów – wyszukiwanie

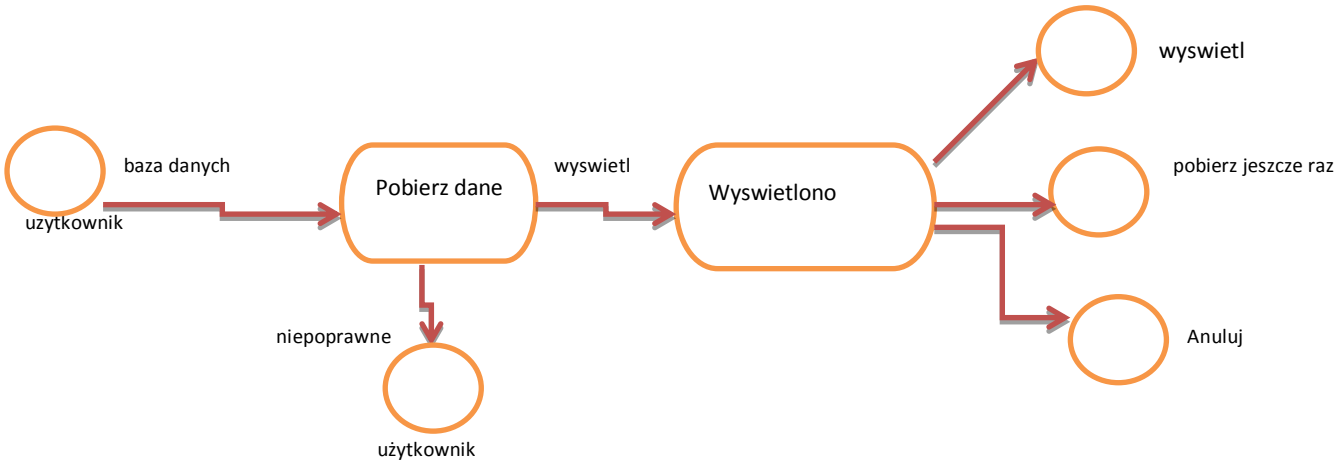
Wyszukiwanie produktów z bazy




```
stmt = con.createStatement();
rs = stmt.executeQuery("SELECT * FROM " + nazwa_bazy + " WHERE Produktu LIKE '%" + nazwa_szukanego + "%'");

int it = 0;
while (rs.next()) {
        result.setValueAt(rs.getInt(1), it, 0);
        result.setValueAt(rs.getString(2), it, 1);
        result.setValueAt(rs.getInt(3), it, 2);
        result.setValueAt(rs.getInt(4), it, 3);
        result.setValueAt(rs.getInt(5), it, 4);
        result.setValueAt(rs.getInt(6), it, 5);
        it++;
}
```

14. Dane startowe użytkownika

Pobranie z bazy danych, danych użytkownika.

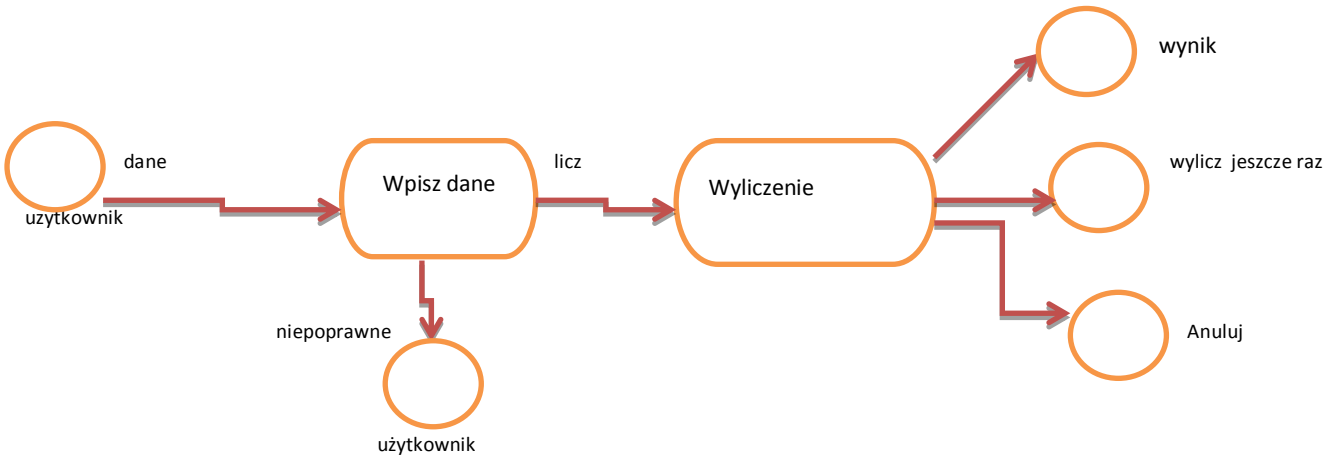



```
Statement stmt = con.createStatement();
String[] dane = new String[8];
ResultSet rs = stmt.executeQuery("SELECT * from Uzytkownicy");
int i = 0;
while (rs.next()) {
        int numer_pomocniczy = rs.getInt(1);

        if (numer == numer_pomocniczy) {

                dane[1] = rs.getString(2);// imie
                dane[2] = rs.getString(3);// nazwisko
```

```
                dane[3] = rs.getString(4);// aderes
                dane[4] = rs.getString(6);// email
                dane[5] = rs.getString(8);// login
                dane[6] = rs.getString(9);// haslo
                dane[7] = rs.getString(5);// tel

            }
        }
```

15. Trening – zapotrzebowanie kaloryczne

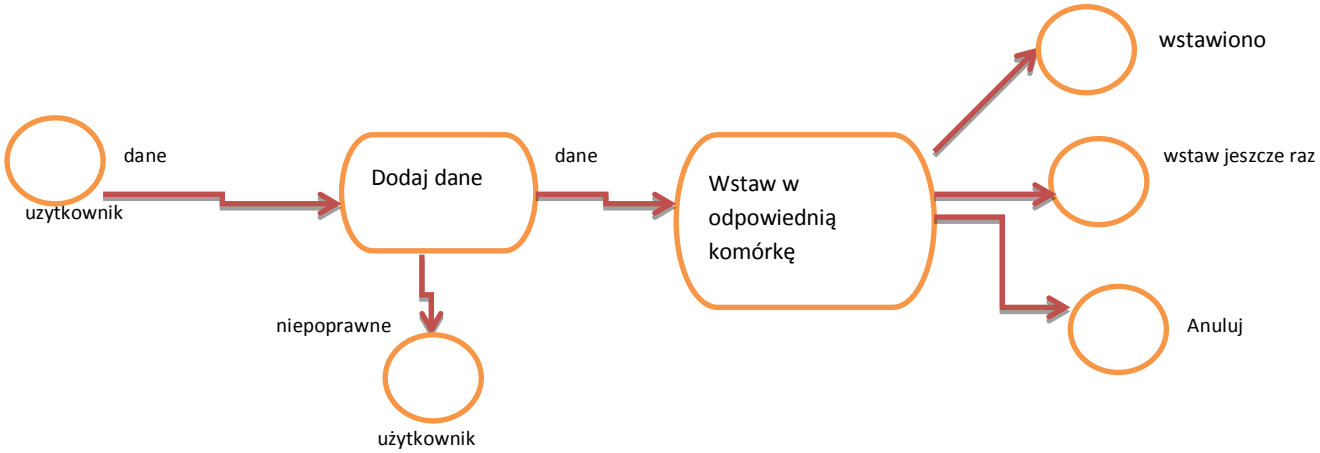
Zmiera wartosci i wylicza zapotrzebowanie kaloryczne.




```
                for (int k = w; k <= wk; k++) {
                kolumna = dzien;
                wiersz = k;
                int ile_czasu = Integer.parseInt(spinner_ile.getValue().toString());
                table_trenig.setValueAt(nazwa, wiersz - 1, kolumna);

                spalanie = (spalanie * (ile_czasu / 60));

                tab[kolumna][wiersz] = spalanie;

                int liczbaSuma = 0;
                for (int i = 0; i < tab.length; i++) {
                        for (int j = 0; j < tab[i].length; j++) {
                                liczbaSuma += tab[i][j];
                                jtfSpalanie_trenig.setText("" + liczbaSuma);
                        }
                }
            }
        }
```

16. Trening – dodanie

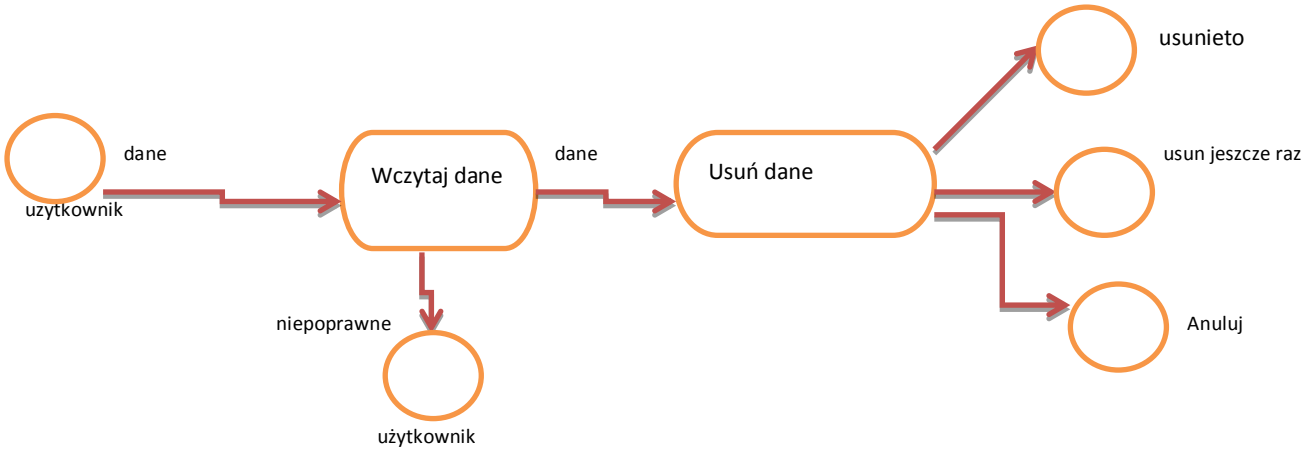
Dodanie do tabeli oraz wstawienie w odpowiednią komórkę.




```
kolumna = Integer.parseInt(spinner_kolumny.getValue().toString());
                    wiersz = Integer.parseInt(spinner_wiersz.getValue().toString());
                    int ile_czasu = Integer.parseInt(spinner_ile.getValue().toString());
                    table_trenig.setValueAt(nazwa, wiersz - 1, kolumna);

                    spalanie = (spalanie * (ile_czasu / 60));

                    tab[kolumna][wiersz] = spalanie;

                    int liczbaSuma = 0;
                    for (int i = 0; i < tab.length; i++) {
                            for (int j = 0; j < tab[i].length; j++) {
                                    liczbaSuma += tab[i][j];
                                    jtfSpalanie_trenig.setText("" + liczbaSuma);
                            }
                    }

            }
```

17. Trening – usunięcie

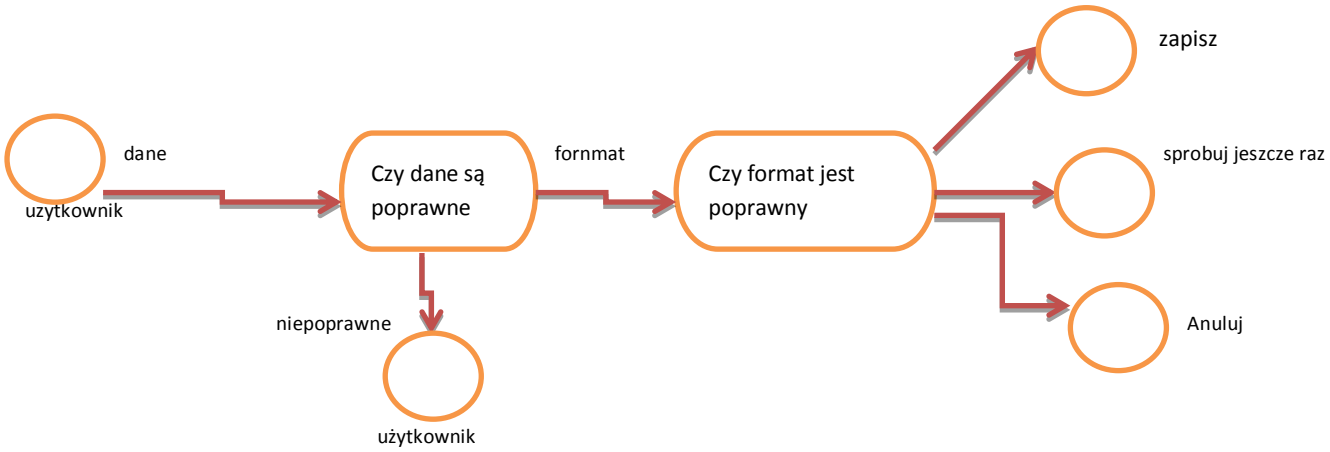
Usunięcie treningu.




```
kolumna = Integer.parseInt(spinner_kolumny.getValue().toString());
wiersz = Integer.parseInt(spinner_wiersz.getValue().toString());
table_trenig.setValueAt("", wiersz - 1, kolumna);

tab[kolumna][wiersz] = 0;

int liczbaSuma = 0;
for (int i = 0; i < tab.length; i++) {
        for (int j = 0; j < tab[i].length; j++) {
                liczbaSuma += tab[i][j];
                jtfSpalanie_trenig.setText("" + liczbaSuma);
        }
}

}
```

18. Trening - zapis do PDF

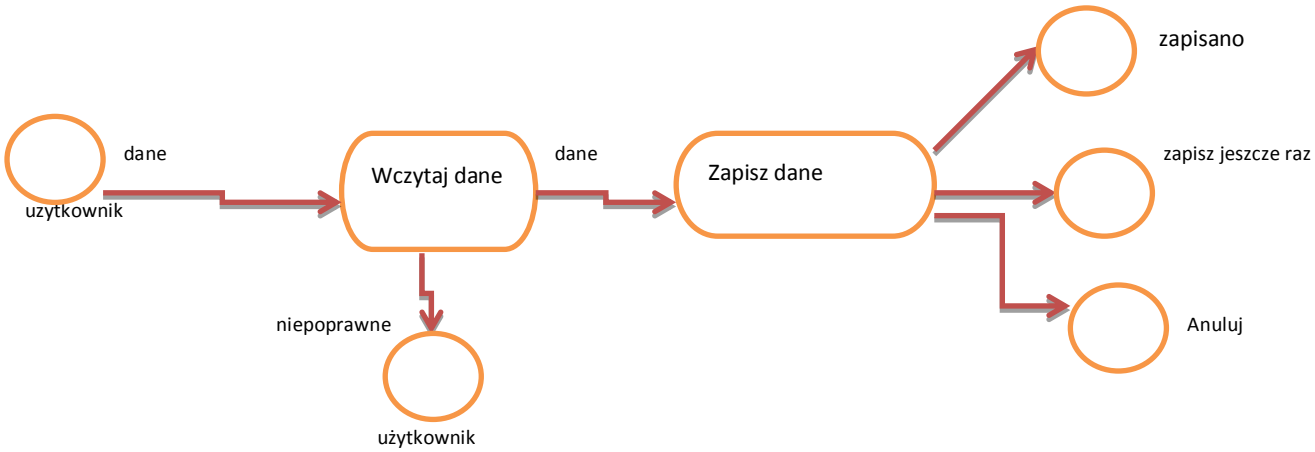
Zapis do PDF.




```
Document document = new Document();
        boolean shapes = true;

        try {

                PdfWriter writer;
                if (shapes)
                        writer = PdfWriter.getInstance(document, new FileOutputStream(userLogin + ".pdf"));
                else
                        writer = PdfWriter.getInstance(document, new FileOutputStream(userLogin + ".pdf"));
                document.open();
                PdfContentByte cb = writer.getDirectContent();
                PdfTemplate tp = cb.createTemplate(1500, 500);
                Graphics2D g2;
                if (shapes)
                        g2 = tp.createGraphicsShapes(1500, 500);
                else
                        g2 = tp.createGraphics(1500, 500);
                table_trenig.print(g2);
                g2.dispose();
                cb.addTemplate(tp, 30, 300);

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

        document.close();
```

19. Trening - zapis do bazy

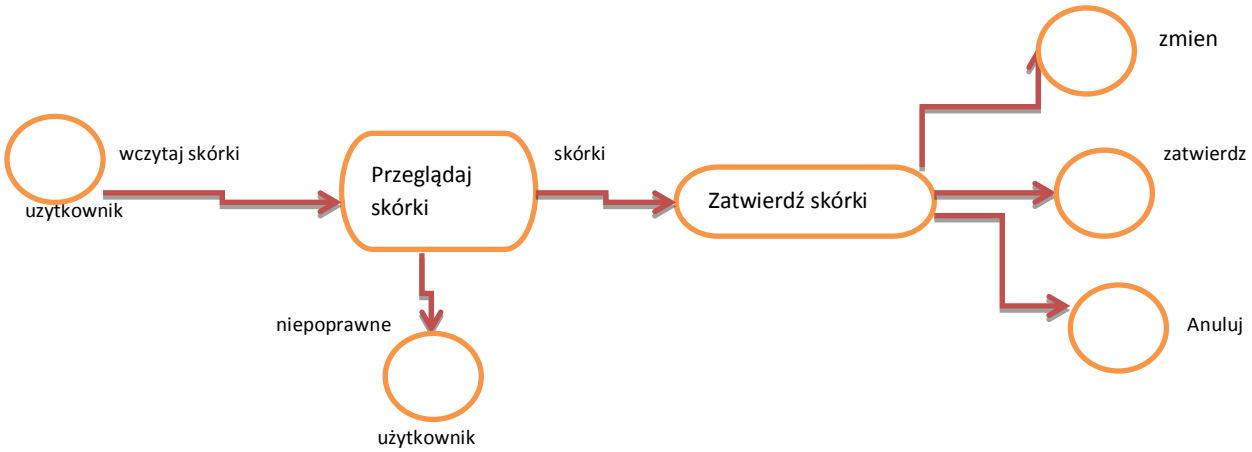
Dane z treningu zapisują się do bazy




```
zapiszTabele(userLogin + ".dieta.fit", table);
        zapiszTabele(userLogin + ".trening.fit", table_trenig);
zapiszTabele(String nazwaPliku, JTable tab) {

        try {

                FileWriter fw = new FileWriter(new File(nazwaPliku));
                BufferedWriter bw = new BufferedWriter(fw);
                String komorka;

                for (int i = 0; i < tab.getRowCount(); i++) {
                        for (int j = 0; j < tab.getColumnCount(); j++) {
                                komorka = Objects.toString(tab.getModel().getValueAt(i, j), "");
                                bw.write(komorka);
                                bw.write("\t");
                        }
                        bw.write("\n");
                }
                bw.close();
                fw.close();

        } catch (Exception e) {
                e.printStackTrace();
        }
```

20. Opcje skórka

Możliwość wyboru skórki i zmiany na tą która nam się najbardziej podoba.




```
if (źródło == rdbtnmntmMetal) {

                try {

                        UIManager.setLookAndFeel("javax.swing.plaf.metal.MetalLookAndFeel");

                } catch (ClassNotFoundException | InstantiationException | IllegalAccessException
                                | UnsupportedLookAndFeelException e) {

                        e.printStackTrace();
                }
                SwingUtilities.updateComponentTreeUI(this);
                this.repaint();
```