



**WYŻSZA SZKOŁA  
INFORMATYKI i ZARZĄDZANIA**  
z siedzibą w Rzeszowie

KOLEGIUM INFORMATYKI STOSOWANEJ

**Kierunek: Informatyk**  
**Specjalność: Inżynieria danych**

Korneliusz Koczera  
Nr albumu studenta: w70779

**Planer posiłków**

Prowadzący: mgr inż. Ewa Żesławska

**Praca projektowa programowanie obiektowe C#**

Rzeszów 2025

# Spis treści

<b>Opis założeń projektu i cele projektu</b>	<b>3</b>
Wymagania funkcjonalne . . . . .	3
Wymagania niefunkcjonalne . . . . .	3
<b>Opis struktury projektu</b>	<b>4</b>
Diagram klas . . . . .	4
Opis techniczny . . . . .	5
<b>Harmonogram realizacji projektu</b>	<b>8</b>
<b>Repozytorium i system kontroli wersji</b>	<b>9</b>
<b>Prezentacja warstwy użytkowej projektu</b>	<b>10</b>
<b>Podsumowanie</b>	<b>11</b>
<b>Literatura</b>	<b>12</b>

## Opis założeń projektu i cele projektu

Projekt "Planer Posiłków" to aplikacja desktopowa umożliwiająca zarządzanie planem posiłków. Celem projektu jest stworzenie narzędzia, które pozwoli użytkownikowi na dodawanie, edytowanie, usuwanie i wyświetlanie posiłków w określonych datach, z uwzględnieniem kalorii i rodzaju posiłku. Aplikacja pozwala na łatwe śledzenie spożywanych posiłków oraz planowanie ich na kolejne dni.

Projekt będzie wykorzystywał prosty interfejs graficzny umożliwiający zarządzanie danymi. W aplikacji użytkownik będzie mógł wprowadzać dane posiłków (np. śniadanie, obiad, kolacja), ich kalorie oraz datę i godzinę spożycia. Program zapewni również możliwość edycji posiłków oraz ich usuwania.

Celem jest stworzenie intuicyjnego i funkcjonalnego narzędzia, które umożliwi użytkownikowi pełną kontrolę nad swoim planem posiłków.

## Wymagania funkcjonalne

Dodawanie posiłków:

Użytkownik powinien mieć możliwość dodawania posiłków (Śniadanie, Obiad, Kolacja, Przekąska).

Posiłki muszą zawierać następujące informacje: nazwa, kalorie, data, godzina.

Edycja posiłków:

Użytkownik może edytować szczegóły posiłku, w tym nazwę, kalorie oraz datę/godzinę, przy czym aplikacja musi sprawdzić, czy nie ma już posiłku o tej samej dacie i godzinie.

Usuwanie posiłków:

Użytkownik ma możliwość usunięcia posiłku na podstawie daty i godziny.

Wyświetlanie posiłków:

Posiłki powinny być wyświetlane w interfejsie graficznym (np. ListBox) w porządku od najstarszego do najnowszego posiłku.

Użytkownik może wybrać posiłek z listy, aby edytować lub usunąć go.

Zapis i odczyt danych:

Aplikacja powinna mieć możliwość zapisania posiłków do pliku tekstowego oraz odczytania zapisanych danych po ponownym uruchomieniu aplikacji.

Walidacja danych:

Program powinien zapewniać walidację danych, np. sprawdzanie, czy pole nazwy posiłku nie jest puste, czy kalorie są liczbą, czy data jest poprawna.

Bezpieczeństwo danych:

Aplikacja nie przechowuje danych w bazie danych, ale zapisuje je do pliku, dlatego ważne jest, by zapewnić odpowiednią ochronę pliku przed utratą danych.

## Wymagania niefunkcjonalne

Intuicyjny interfejs użytkownika:

Interfejs aplikacji powinien być prosty i łatwy w obsłudze, aby użytkownicy nie musieli posiadać specjalistycznej wiedzy komputerowej, aby wprowadzać, edytować lub usuwać dane.

Wydajność:

Aplikacja powinna działać płynnie, bez opóźnień podczas wyświetlania, edytowania czy usuwania posiłków, nawet przy większej liczbie zapisanych posiłków.

Stabilność:

Program nie powinien ulegać awariom ani błędom w przypadku niepoprawnych danych wejściowych, takich jak puste pola czy błędny format daty.

Przenośność:

Aplikacja powinna działać na komputerach z systemem Windows i nie wymagać instalacji dodatkowego oprogramowania lub bibliotek.

Przyjazność dla użytkownika:

Powinna zapewniać jasne komunikaty o błędach i potwierdzenia operacji, np. informowanie o niepoprawnym wprowadzeniu danych lub udanych operacjach (dodanie, edycja, usunięcie).

Rozszerzalność:

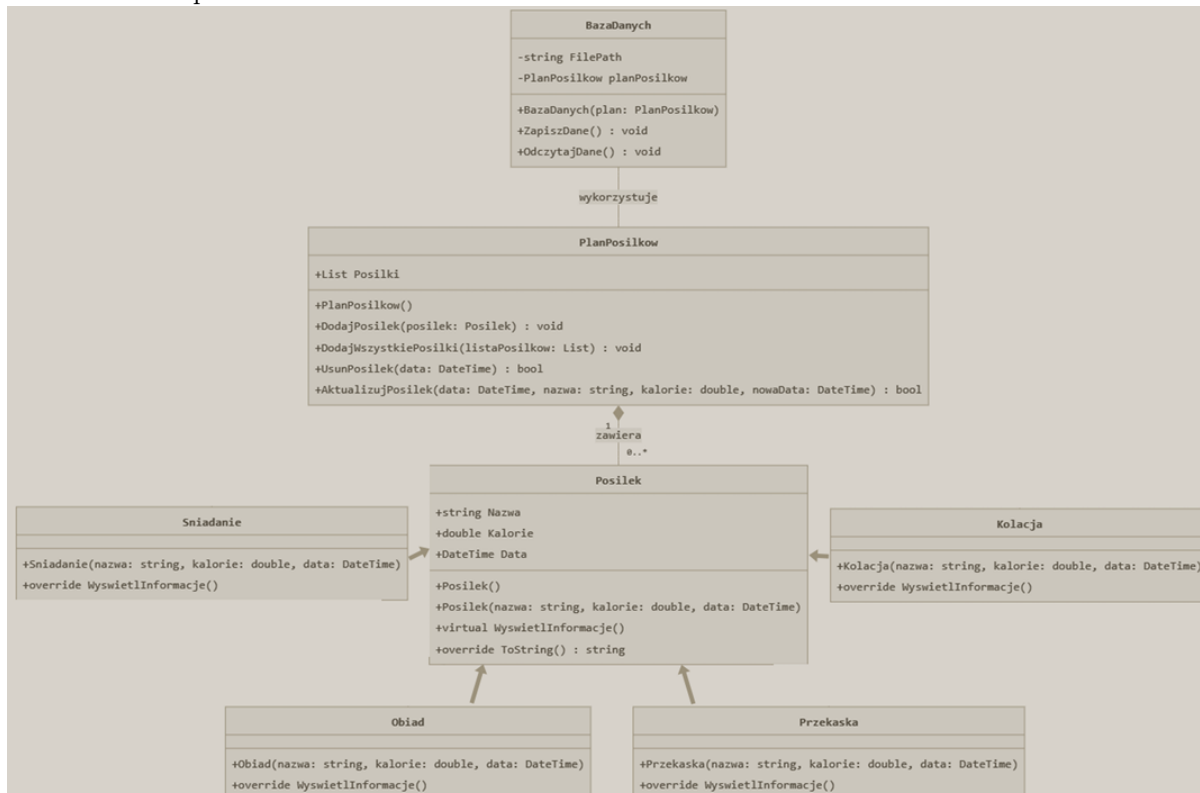
Projekt powinien być zaprojektowany w sposób umożliwiający przyszłą rozbudowę o dodatkowe funkcje, takie jak generowanie raportów o spożytych kaloriach, integracja z bazą danych czy synchronizacja z urządzeniami mobilnymi.

## Opis struktury projektu

Projekt "Planer Posiłków" jest aplikacją napisaną w języku C, która umożliwia zarządzanie planem posiłków (CRUD) przy wykorzystaniu plików tekstowych do przechowywania danych. Aplikacja umożliwia użytkownikowi dodawanie, edytowanie, usuwanie i przeglądanie posiłków na podstawie daty i godziny. W projekcie zastosowano obiektową organizację danych, a klasy są podzielone na różne typy posiłków oraz mechanizmy do przechowywania i zarządzania tymi danymi.

## Diagram klas

Diagram klas przedstawia strukturę projektu „Planer Posiłków”, ukazując relacje między poszczególnymi klasami oraz ich odpowiedzialności.



## Opis techniczny

Zarządzanie danymi (CRUD)

Program realizuje operacje CRUD (Create, Read, Update, Delete) na danych posiłków. Operacje te odbywają się zarówno w pamięci, jak i na pliku tekstowym. Oto jak każda operacja jest realizowana:

Create (Dodawanie posiłków):

Nowe posiłki dodawane są do listy posiłków w obiekcie [PlanPosilkow](#).

Po dodaniu posiłku do listy, dane są zapisywane do pliku tekstowego za pomocą metody [ZapiszDane\(\)](#) w klasie [BazaDanych](#).

```
public List<Posilek> Posilki { get; private set; }

4 references
public void DodajPosilek(Posilek posilek)
{
    if (posilek == null)
    {
        throw new ArgumentNullException(nameof(posilek), "Posiłek nie może być null.");
    }
    Posilki.Add(posilek);
}

public void ZapiszDane()
{
    using (StreamWriter writer = new StreamWriter(FilePath))
    {
        foreach (var posilek in planPosilkow.Posilki)
        {
            // Zapisujemy datę bez milisekund
            writer.WriteLine($"{posilek.Nazwa};{posilek.GetType().Name};{posilek.Kalorie};{posilek.Data:dd-MM-yyyy HH:mm:ss}");
        }
    }
}
```

Read (Odczyt danych):

Przy starcie aplikacji dane posiłków są wczytywane z pliku tekstowego przez metodę [OdczytajDane\(\)](#) w klasie [BazaDanych](#).

Wczytane dane są przekazywane do obiektu [PlanPosilkow](#), który przechowuje kolekcję posiłków.

```
static class Program
{
    [STAThread]
    0 references
    static void Main()
    {
        Application.EnableVisualStyles();
        Application.SetCompatibleTextRenderingDefault(false);

        PlanPosilkow plan = new PlanPosilkow();
        BazaDanych bazaDanych = new BazaDanych(plan);

        bazaDanych.OdczytajDane();

        Application.Run(new Form1(plan, bazaDanych));
    }
}
```

```

public void OdczytajDane()
{
    if (!File.Exists(filePath))
        return;

    List<Posilek> wczytanePosilki = new List<Posilek>();

    using (StreamReader reader = new StreamReader(filePath))
    {
        string line;
        while ((line = reader.ReadLine()) != null)
        {
            var dane = line.Split(';');
            if (dane.Length == 4 && double.TryParse(dane[2], out double kalorie) && DateTime.TryParseExact(
                dane[3], "dd-MM-yyyy HH:mm:ss", null, System.Globalization.DateTimeStyles.None, out DateTime data))
            {
                // Przycina milisekundy w odczytanej dacie
                data = data.AddMilliseconds(-data.Millisecond);

                switch (dane[1])
                {
                    case "Sniadanie":
                        wczytanePosilki.Add(new Sniadanie(dane[0], kalorie, data));
                        break;
                    case "Obiad":
                        wczytanePosilki.Add(new Obiad(dane[0], kalorie, data));
                        break;
                    case "Przekaska":
                        wczytanePosilki.Add(new Przekaska(dane[0], kalorie, data));
                        break;
                    case "Kolacja":
                        wczytanePosilki.Add(new Kolacja(dane[0], kalorie, data));
                        break;
                }
            }
        }
    }

    planPosilkow.DodajWszystkiePosilki(wczytanePosilki);
}

```

Update (Edycja posiłków):

W przypadku edycji, użytkownik może wybrać posiłek z listy, a następnie zmodyfikować jego właściwości (np. nazwę, kalorie, datę).

Zmienione dane są aktualizowane w kolekcji **Posilki** w obiekcie **PlanPosilkow** i zapisane w pliku tekstowym.

```

private void button4_Click(object sender, EventArgs e)
{
    if (edytowanyPosilek != null)
    {
        Double kalorie;
        if (!Double.TryParse(textBox2.Text, out kalorie))
        {
            MessageBox.Show("Wprowadź prawidłową liczbę kalorii!");
            return;
        }

        // Przycina sekund i milisekund w danych (ignoruje sekundy i milisekundy)
        DateTime data = edytowanyPosilek.Data
            .AddMilliseconds(-edytowanyPosilek.Data.Millisecond);

        planPosilkow.AktualizujPosilek(data, textBox1.Text, kalorie, dateTimePicker1.Value);
        AktualizujListe();
        bazaDanych.ZapiszDane();

        WylaczTrybEdycji();
    }
}

public bool AktualizujPosilek(DateTime data, string nazwa, double kalorie, DateTime nowaData)
{
    var posilekDoAktualizacji = Posilki.FirstOrDefault(p => p.Data.Equals(data));
    if (posilekDoAktualizacji != null)
    {
        posilekDoAktualizacji.Nazwa = nazwa;
        posilekDoAktualizacji.Kalorie = kalorie;
        posilekDoAktualizacji.Data = nowaData;
        return true;
    }
    return false;
}

```

Delete (Usuwanie posiłków):

Posiłek można usunąć na podstawie daty, a jego dane są następnie usuwane z kolekcji w **PlanPosilkow** oraz z pliku tekstowego.

Usuwanie odbywa się na podstawie daty. Data jest unikalna dla każdego posiłku.

```

public bool UsunPosilek(DateTime data)
{
    var posilekDoUsuniecia = Posilki.FirstOrDefault(p => p.Data.Equals(data));

    if (posilekDoUsuniecia != null)
    {
        Posilki.Remove(posilekDoUsuniecia);
        return true;
    }

    return false;
}

```

Struktura pliku tekstowego

Dane są przechowywane w pliku tekstowym (**db.txt**), gdzie każdy posiłek jest zapisywany w jednym wierszu. Format danych jest następujący:

**NazwaPosilku;TypPosilku;Kalorie;Data**

Przykład wiersza w pliku:

**Jajecznicza;Sniadanie;200;2025-02-18 07:30:00**

Podczas odczytu danych, program wczytuje każdy wiersz, rozdziela go na poszczególne elementy (nazwa, typ, kalorie, data), a następnie na ich podstawie tworzy obiekty odpowiednich klas posiłków. Po zapisaniu nowych posiłków, dane są zapisywane z powrotem do pliku.

Interfejs użytkownika

Program posiada interfejs graficzny (GUI), zbudowany przy użyciu Windows Forms, w którym użytkownik może:

Dodawać nowe posiłki przez formularz z polami tekstowymi i przyciskami,

Edytować istniejące posiłki,

Usunąć posiłki,

Przeglądać listę posiłków w **ListBox**, wyświetlaną w porządku od najstarszego do najnowszego.

### Proces uruchamiania aplikacji

Przy uruchomieniu aplikacji w metodzie **Main()** następuje:

Utworzenie obiektu **PlanPosilkow**, który zarządza kolekcją posiłków,

Utworzenie obiektu **BazaDanych**, który jest odpowiedzialny za zapis i odczyt danych z pliku,

Wczytanie danych z pliku za pomocą **OdczytajDane()** w klasie **BazaDanych**,

Uruchomienie głównego okna aplikacji (**Form1**), który przyjmuje jako argumenty obiekty **PlanPosilkow** i **BazaDanych**.

```

static class Program
{
    [STAThread]
    0 references
    static void Main()
    {
        Application.EnableVisualStyles();
        Application.SetCompatibleTextRenderingDefault(false);

        PlanPosilkow plan = new PlanPosilkow();
        BazaDanych bazaDanych = new BazaDanych(plan);

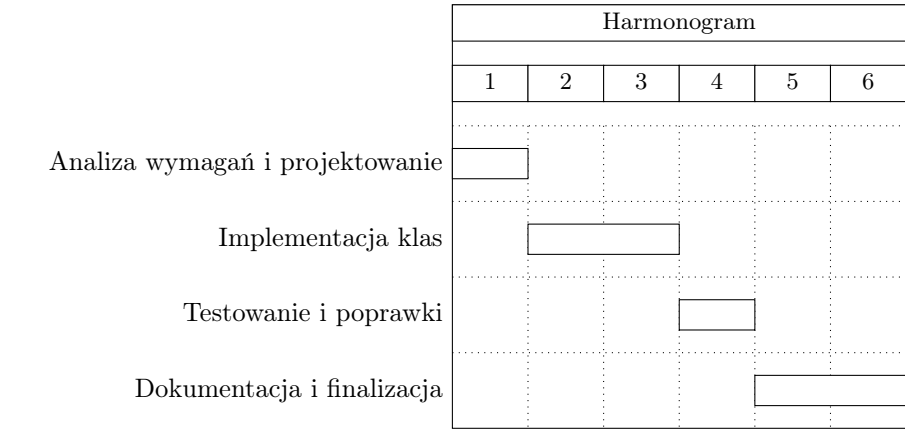
        bazaDanych.OdczytajDane();

        Application.Run(new Form1(plan, bazaDanych));
    }
}

```

# Harmonogram realizacji projektu

W poniższym harmonogramie przedstawiono etapy realizacji projektu wraz z ich orientacyjnymi terminami. Każdy etap został zaplanowany w taki sposób, aby zapewnić płynny przebieg prac oraz odpowiednią ilość czasu na testowanie i finalizację.



Każdy etap realizacji projektu ma kluczowe znaczenie dla jego końcowego sukcesu. Analiza wymagań pozwala na zrozumienie celów, implementacja zapewnia działającą funkcjonalność, testowanie eliminuje błędy, a dokumentacja i finalizacja pozwalają na uporządkowanie efektów pracy.



## Repozytorium i system kontroli wersji

W projekcie "Planer Pożytków" zastosowano system kontroli wersji Git, który umożliwia zarządzanie historią zmian w projekcie, współpracę z innymi programistami oraz śledzenie rozwoju aplikacji. Repozytorium jest hostowane na platformie GitHub, dzięki czemu dostęp do kodu źródłowego jest łatwy i zorganizowany.

### **Struktura repozytorium:**

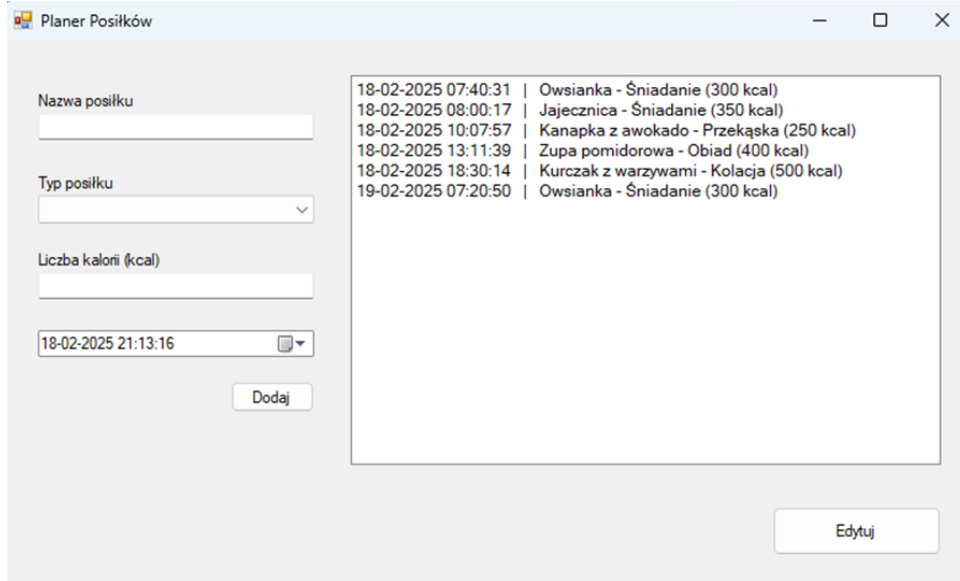
Główna gałąź (branch) main zawiera stabilną wersję aplikacji.

Dla nowych funkcjonalności tworzone są osobne gałęzie, które po zakończeniu pracy są scalane (merge) do gałęzi main.

Każda zmiana w kodzie jest dokumentowana poprzez commit i wiadomość, co pozwala na łatwe śledzenie postępów i ewentualne przywrócenie wcześniejszych wersji kodu.

## Prezentacja warstwy użytkowej projektu

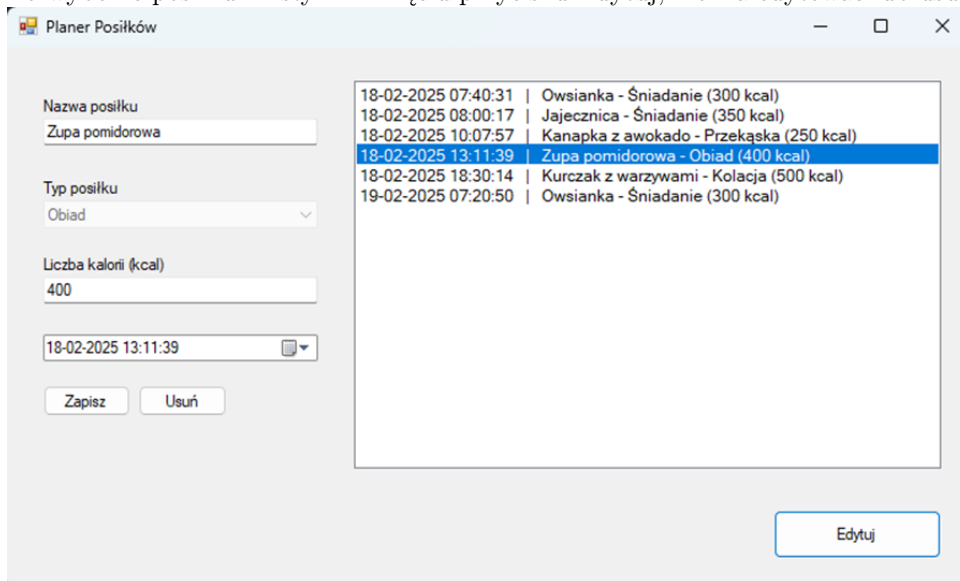
Aplikacja posiada interfejs użytkownika, który umożliwia dodawanie posiłków, edytowanie, usuwanie oraz wyświetlanie posiłków w kolejności od najbliższego. Poniżej znajduje się zrzut ekranu interfejsu.



The screenshot shows the 'Planer Posiłków' application window. On the left is a form to add a new meal with fields for 'Nazwa posiłku' (Meal name), 'Typ posiłku' (Meal type), 'Liczba kalorii (kcal)' (Number of calories), and a date-time picker. A 'Dodaj' (Add) button is at the bottom of the form. On the right is a list of existing meals, each showing a date-time, a meal name, and its calorie count. An 'Edytuj' (Edit) button is at the bottom right of the window.

Time	Meal Name	Calories (kcal)
18-02-2025 07:40:31	Owsianka - Śniadanie	300
18-02-2025 08:00:17	Jajecznica - Śniadanie	350
18-02-2025 10:07:57	Kanapka z awokado - Przekąska	250
18-02-2025 13:11:39	Zupa pomidorowa - Obiad	400
18-02-2025 18:30:14	Kurczak z warzywami - Kolacja	500
19-02-2025 07:20:50	Owsianka - Śniadanie	300

Po wyborze posiłku z listy i kliknięciu przycisku Edytuj, można edytować lub usunąć dany posiłek.



The screenshot shows the 'Planer Posiłków' application window with the 'Edytuj' (Edit) form active. The form fields are pre-filled with the details of the selected meal: 'Zupa pomidorowa' (Soup), 'Obiad' (Dinner), and 400 kcal. The date-time picker shows '18-02-2025 13:11:39'. There are 'Zapisz' (Save) and 'Usuń' (Delete) buttons at the bottom of the form. The list of meals on the right is the same as in the previous screenshot, but the entry '18-02-2025 13:11:39 | Zupa pomidorowa - Obiad (400 kcal)' is highlighted in blue. An 'Edytuj' (Edit) button is at the bottom right of the window.

Time	Meal Name	Calories (kcal)
18-02-2025 07:40:31	Owsianka - Śniadanie	300
18-02-2025 08:00:17	Jajecznica - Śniadanie	350
18-02-2025 10:07:57	Kanapka z awokado - Przekąska	250
18-02-2025 13:11:39	Zupa pomidorowa - Obiad	400
18-02-2025 18:30:14	Kurczak z warzywami - Kolacja	500
19-02-2025 07:20:50	Owsianka - Śniadanie	300

# Podsumowanie

## Zrealizowane prace:

Stworzenie aplikacji do zarządzania planem posiłków, z możliwością dodawania, edytowania, usuwania i przeglądania posiłków.

Implementacja operacji CRUD przy użyciu pliku tekstowego jako bazy danych.

Stworzenie obiektowego modelu posiłków (hierarchia klas **Posilek**, **Sniadanie**, **Obiad**, **Kolacja**, **Przekaska**).

Integracja interfejsu użytkownika (GUI) z mechanizmami zarządzania danymi, przy użyciu Windows Forms.

Implementacja systemu zapisu i odczytu danych z pliku tekstowego, umożliwiającego trwałe przechowywanie danych.

## Planowane dalsze prace rozwojowe:

Rozszerzenie funkcjonalności aplikacji o możliwość generowania raportów z posiłków (np. dzienny bilans kalorii).

Dodanie opcji eksportu planu posiłków do pliku PDF lub CSV.

Udoskonalenie interfejsu użytkownika poprzez dodanie powiadomień lub przypomnień o zaplanowanych posiłkach.

Rozbudowa aplikacji o opcje takich jak: dodawanie kategorii posiłków (np. dieta wegetariańska, bezglutenowa) oraz dostosowanie aplikacji do różnych jednostek kalorycznych (np. kilokalorie, joule).

## Literatura

- Dokumentacja języka C# – Microsoft. Dostępne online: <https://learn.microsoft.com/en-us/dotnet/csharp/>
- Windows Forms Documentation – Microsoft. Dostępne online: <https://learn.microsoft.com/en-us/dotnet/desktop/winforms/?view=net-9.0>