

POLITECHNIKA ŚLĄSKA W GLIWICACH WYDZIAŁ ELEKTRYCZNY

Instytut Elektrotechniki i Informatyki

**PROJEKT INŻYNIERSKI**

**Serwis internetowy wspomagający planowanie diety**

**Diet planner website**

Student: **Roger Paul Skrzypczyk**  
Nr albumu: 232415

Studia: Stacjonarne I stopnia  
Kierunek: Informatyka  
Specjalność: Informatyka w systemach elektrycznych

Prowadzący: [dr inż. Bożena Wieczorek](http://eksperci.polsl.pl/eksperci/szczegoly.php?scbpos=&eid=1154&)  
Recenzent: dr inż. Michał LewandowskiSpis treści

Słownik pojęć - 3 -

Wstęp - 4 -

1 Komponowanie właściwej diety - 5 -

1.1 Statystyki otyłości - 5 -

1.2 Zapotrzebowanie na makroskładniki - 8 -

1.3 Zapotrzebowanie kaloryczne - 9 -

1.4 Postrzeganie kalorii - 11 -

1.5 Typologia Sheldona - 12 -

2 Technologie użyte w projekcie - 13 -

2.1 ASP.NET.Core - 13 -

2.2 Visual Studio 2015 - 14 -

2.3 Entity Framework Core - 15 -

2.4 GitHub - 15 -

2.5 SourceTree - 16 -

2.6 Semantic UI - 16 -

3 Implementacja serwisu - 18 -

3.1 Struktura projektu - 18 -

3.2 Nawigacja - 21 -

3.3 Mechanizm zgłaszania produktów - 23 -

3.4 Baza danych - 24 -

3.5 Szyfrowanie danych użytkownika - 26 -

3.6 Korzystanie z serwisu - 27 -

Podsumowanie - 30 -

Literatura - 31 -

Słownik pojęć

Framework - zbiór narzędzi pozwalających na budowę aplikacji i ułatwiający pracę poprzez komponenty oraz biblioteki przeznaczone do wykonywania określonych zadań.

Sól (ang. salt) - losowe dane dodawane do hasła podczas obliczania funkcji skrótu. Sól zapobiega atakom słownikowym na bazę haseł.

TDEE (Total Daily Energy Expenditure) - całkowite dzienne zapotrzebowanie kaloryczne.

BMR (Basal Metabolic Rate) - wskaźnik podstawowej przemiany materii.

TEF (Thermic Effect of Food) - efekt termiczny pożywienia.

TEA - kalorie spalone podczas aktywności fizycznej.

NEAT - kalorie spalane podczas codziennych czynności.

Ektomorfik - typ budowy ciała cechujący się smukłą sylwetką, drobnym kośćcem, chudymi oraz długimi kończynami, a także wąskimi ramionami.

Endomorfik - typ budowy ciała cechujący się masywną, owalną sylwetką, mający tendencję do tycia poprzez wolny metabolizm oraz posiadający wysoki poziom tkanki tłuszczowej.

Mezomorfik - typ budowy ciała cechujący się szerokim rozstawem barków, długimi kończynami oraz wąską talią, co wizualnie przypomina literę „V”. W łatwy sposób pozyskuje mięśnie oraz posiada szybszą regenerację. Typ ten posiadają nieliczni.

Assembly - najmniejsza jednostka wdrożeniowa aplikacji .NET. Może być plikiem dll, albo exe.

Dll (Dynamic Link Library) - [biblioteka współdzielon](https://pl.wikipedia.org/wiki/Biblioteka_współdzielona)a w środowisku [Microsoft Windows](https://pl.wikipedia.org/wiki/Microsoft_Windows), przechowująca [implementacje](https://pl.wikipedia.org/wiki/Implementacja_(informatyka)) różnych [podprogramów](https://pl.wikipedia.org/wiki/Podprogram) programu lub [zasoby programu](https://pl.wikipedia.org/wiki/Zasoby_programu).

Wstęp

Na przełomie kilku lat, zmienił się pogląd ludzi dotyczący zdrowego trybu życia. Zauważalne jest ich samozaparcie, dyscyplina w dążeniu do wymarzonej sylwetki oraz długiego życia. Przyczyniło się to do większego zapotrzebowanie na strony przeznaczone tej dziedzinie. Jednym z głównych czynników zdrowego trybu życia jest dieta, często błędnie postrzegana jako deficyt kaloryczny. Dieta może służyć zarówno utracie jak i przyroście masy ciała. Wyróżniamy kilka sposobów kontroli diety. Możliwe jest zlecenie dietetykowi sporządzenia harmonogramu posiłków lub wyboru gotowych szablonów. Jednym z popularniejszych sposobów w ostatnich latach jest liczenie kalorii oraz makroskładników. Jest to w zupełności wystarczające dla osób, które po prostu chcą zwiększyć lub zmniejszyć masę ciała. Istnieje wiele stron poświęconych liczeniu kalorii od strony merytorycznej jak i praktycznej. Większość posiada nadmiar informacji, co przyczynia się do zniechęcania użytkowników.

Celem projektu jest utworzenie strony internetowej, która pozwala na wyliczenie zapotrzebowania kalorycznego w prosty i wygodny sposób, zapamiętywania naszych dni żywieniowych oraz planowania bilansu kalorycznego. Użytkownik będzie miał możliwość dodawania własnych produktów jak i korzystania z produktów dodanych przez innych użytkowników strony.

Podobnymi projektami są:

* **Po treningu** - polska strona pozwalająca na wyliczania bilansu kalorycznego, zarządzania swoją dietą. Właścicielem strony jest firma **SFD**. Posiada własne forum oraz sklep, które zostały połączone ze stroną **Po treningu**.
* **MyFitnessPal** - amerykańska strona przeznaczona do śledzenia diety oraz ćwiczeń, w celu określenia optymalnego spożycia kalorii. W 2015 roku została wykupiona przez firmę **Under Armor**, która obecnie jest właścicielem serwisu internetowego. Strona posiada także własną aplikację na telefony z systemami operacyjnymi IOS i Android.
* **Lose it** - kolejna strona poświęcona liczeniu zapotrzebowania kalorycznego. Wyróżnia się innowacyjnym pomysłem robienia zdjęć produktów, które zostają przetworzone i dodane do naszej puli kalorycznej.

1. Komponowanie właściwej diety

## Statystyki otyłości

W Polsce co roku zostają przeprowadzane badania otyłości i nadwagi. Sytuacja w naszym kraju, jak i na całym świecie, pogarsza się. Statystyki dotyczące wagi ludności w Polsce przeprowadzone na przełomie 10 lat zostały zaprezentowane w tab. 1.0 - 1.3.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Województwo | Nadwaga i otyłość | Niedowaga | Prawidłowa waga | Nadwaga | Otyłość | Trudno powiedzieć |
| Śląskie | 44,31 | 3,53 | 44,67 | 32,48 | 11,83 | 0,08 |
| Małopolskie | 41,90 | 3,73 | 48,60 | 33,17 | 8,73 | 5,77 |
| Podkarpackie | 46,04 | 3,29 | 46,52 | 34,20 | 11,84 | 4,15 |
| Lubelskie | 43,10 | 4,74 | 44,47 | 30,51 | 12,59 | 7,68 |
| Mazowieckie | 44,12 | 3,03 | 45,55 | 32,31 | 11,81 | 7,31 |
| świętokrzyskie | 42,38 | 3,61 | 50,58 | 33,22 | 9,16 | 3,44 |
| Opolskie | 47,78 | 2,36 | 42,52 | 33,45 | 14,33 | 7,33 |
| Dolnośląskie | 44,78 | 4,21 | 43,75 | 32,53 | 12,25 | 7,25 |
| Lubuskie | 42,37 | 4,21 | 46,50 | 30,76 | 11,61 | 6,93 |
| Łódzkie | 45,36 | 3,33 | 48,04 | 32,85 | 12,51 | 3,28 |
| Wielkopolskie | 46,29 | 3,41 | 44,60 | 32,51 | 13,78 | 5,70 |
| Kujawsko-Pomorskie | 43,77 | 3,98 | 47,22 | 32,05 | 11,72 | 5,04 |
| Zachodniopomorskie | 43,30 | 3,41 | 44,04 | 30,81 | 12,49 | 9,24 |
| Warmińsko-Mazurskie | 47,17 | 3,56 | 42,41 | 33,70 | 13,47 | 6,86 |
| Podlaskie | 43,22 | 3,52 | 45,63 | 29,97 | 13,25 | 7,63 |
| Pomorskie | 42,52 | 3,09 | 45,03 | 31,07 | 11,45 | 9,36 |

Tabela 1.0 Statystyki w Polsce na przełomie 07.2006-06.2007

W tab. 1.0, zawierającej wyniki badań przeprowadzonych w latach 2006 i 2007, można zaobserwować, że jedynie w woj. świętokrzyskim prawidłową wagę ma ponad 50% ludności. Przeciętnie 1/3 społeczeństwa cierpiała na nadwagę, a ponad 10 % na otyłość. Niedowaga w Polsce oscylowała w granicach 3 i 4 %.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Województwo | Nadwaga i Otyłość | Niedowaga | Prawidłowa waga | Nadwaga | Otyłość | Trudno powiedzieć |
| Śląskie | 48,14 | 2,47 | 40,45 | 36,96 | 11,18 | 8,94 |
| Małopolskie | 40,96 | 2,6 | 48,96 | 30,88 | 10,08 | 7,47 |
| Podkarpackie | 49,69 | 2,06 | 44,98 | 39,5 | 10,19 | 3,27 |
| Lubelskie | 46,54 | 3,17 | 43,72 | 32,61 | 13,93 | 6,57 |
| Mazowieckie | 48,47 | 2,08 | 44,05 | 35,19 | 13,28 | 5,4 |
| świętokrzyskie | 48,33 | 2,82 | 47,41 | 37 | 11,33 | 1,39 |
| Opolskie | 47,04 | 1,27 | 48,34 | 38,06 | 8,98 | 3,34 |
| Dolnośląskie | 49,74 | 2,36 | 44,44 | 36,01 | 13,73 | 3,47 |
| Lubuskie | 49,51 | 2,86 | 42,5 | 32,82 | 16,69 | 5,13 |
| Łódzkie | 53,22 | 1,33 | 41,33 | 40,39 | 12,83 | 4,13 |
| Wielkopolskie | 49,99 | 2,63 | 42,94 | 37,34 | 12,65 | 4,44 |
| Kujawsko-Pomorskie | 51,04 | 2,5 | 41,01 | 37,58 | 13,46 | 5,44 |
| Zachodniopomorskie | 43,97 | 2,85 | 43,21 | 32,41 | 11,56 | 9,97 |
| Warmińsko-Mazurskie | 51,08 | 1,25 | 45,24 | 37,27 | 13,81 | 2,43 |
| Podlaskie | 49,66 | 1,86 | 41,25 | 34,97 | 14,69 | 7,23 |
| Pomorskie | 48,19 | 3,05 | 42,86 | 34,97 | 13,22 | 5,91 |

Tabela 1.1 Statystyki w Polsce na przełomie 07.2011-06.2012

Z tabeli 1.1 wynika, iż w przeciągu 5 lat procent niedowagi zmalał, kosztem wzrostu nadwagi i otyłości.

Zjawisko nadwagi i otyłości występuje zarówno w Polsce jak i na całym świecie. Najgorzej pod tym względem sytuacja przedstawia się w krajach najbardziej rozwiniętych. Liderem w rankingu liczby otyłych osób w kraju są Stany Zjednoczone, gdzie aż 1/3 społeczeństwa ma problemy z nadwagą. Przyczyną tego zjawiska jest rozwój gospodarki i poprawa życiowych standardów, co skutkuje brakiem aktywności fizycznej w społeczeństwie.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Województwo | Nadwaga + Otyłość | Niedowaga | Prawidłowa waga | Nadwaga | Otyłość | Trudno powiedzieć |
| Śląskie | 50,68 | 1,06 | 45,83 | 42,11 | 8,57 | 2,43 |
| Małopolskie | 47,29 | 2,39 | 45,86 | 37,13 | 10,16 | 4,46 |
| Podkarpackie | 53,32 | 2,23 | 42,20 | 40,41 | 12,91 | 2,25 |
| Lubelskie | 45,53 | 3,61 | 47,20 | 34,44 | 11,09 | 3,66 |
| Mazowieckie | 49,49 | 1,48 | 41,13 | 36,01 | 13,48 | 7,91 |
| świętokrzyskie | 54,77 | 3,34 | 40,45 | 46,66 | 8,11 | 1,44 |
| Opolskie | 34,81 | 2,81 | 46,48 | 21,23 | 13,58 | 15,90 |
| Dolnośląskie | 51,93 | 1,71 | 42,80 | 37,58 | 14,35 | 3,56 |
| Lubuskie | 44,87 | 2,70 | 51,18 | 29,02 | 15,85 | 1,25 |
| Łódzkie | 52,62 | 1,18 | 44,13 | 38,49 | 14,13 | 2,07 |
| Wielkopolskie | 52,95 | 3,92 | 38,72 | 38,47 | 14,48 | 4,40 |
| Kujawsko-Pomorskie | 47,99 | 4,08 | 44,96 | 36,78 | 11,21 | 2,98 |
| Zachodniopomorskie | 44,90 | 0,73 | 44,88 | 33,55 | 11,35 | 9,48 |
| Warmińsko-Mazurskie | 45,87 | 1,94 | 49,91 | 26,25 | 19,62 | 2,28 |
| Podlaskie | 46,23 | 2,80 | 47,68 | 32,95 | 13,28 | 3,29 |
| Pomorskie | 51,26 | 1,58 | 40,06 | 36,24 | 15,02 | 7,10 |

Tabela 1.2 Statystyki w Polsce na przełomie 07.2015-12.2015

Największym problemem w społeczeństwie jest dobór produktów, które często wydają się zdrowe, a w rzeczywistości nimi nie są. Ludzie uważają, że spożywanie produktów o wysokiej zawartości tłuszczów skutkuje przyrostem wagi, przez co sięgają po ich zamienniki. Zamieniają tłuszcze węglowodanami, które w dużych ilościach zostają przetworzone przez organizm człowieka w tłuszcz. Głównymi produktami przyczyniającymi się do nadwagi i otyłości są produkty o wysokiej zawartości cukrów prostych, przede wszystkim napoje gazowane, słodycze, pieczywo jasne i ciemne, a także produkty mocno przetworzone jak chipsy, frytki czy hamburgery.

## Zapotrzebowanie na makroskładniki

Makroskładniki powinny zostać dobrane indywidualnie w zależności od celu i zapotrzebowania energetycznego danej osoby. Podstawowymi makroskładnikami diety są białka, węglowodany i tłuszcze.

Zakłada się, że na:

* 1 gram białka przypada 4 kcal,
* 1 gram węglowodanów przypada 4 kcal,
* 1 gram tłuszczów przypada 9 kcal.

Dla osób ćwiczących podstawowymi dietami wynikającymi z podziału makroskładników są:

* dieta wysokowęglowodanowa,
* dieta wysokotłuszczowa.

Główną różnicą między tymi dietami jest stosunek zawartości węglowodanów do zawartości tłuszczów w diecie. Standardowo zakłada się, że osoba ćwiczące powinna spożywać 2g białka na kg masy ciała.

Dla pierwszej diety, ilość tłuszczów nie powinna przekraczać 50g, natomiast resztę kalorii uzupełniają węglowodany. Dla drugiej diety, ilość białka pozostaje bez zmian, natomiast stosunek węgli do tłuszczów jest odwrotnie proporcjonalny względem pierwszej.

Przykładowo, w diecie wysokowęglowodanowej osoba ważąca 70 kg i spożywająca 2500 kcal powinna zjeść każdego dnia około 140g białka, 40g tłuszczów i 395g węglowodanów. W przypadku diety wysokotłuszczowej - 140g białka, 40g węglowodanów i 198g tłuszczów.

## Zapotrzebowanie kaloryczne

Występuje kilka sposobów obliczania zapotrzebowania kalorycznego [4]. Pierwsza, najprostsza metoda określona wzorem w wielu przypadkach się nie sprawdza.

(1.0)

We wzorze (1.0) **weight** jest odpowiednikiem masy ciała, a **activity** współczynnikiem aktywności fizycznej, która występuje w przedziale od 1.0 do 2.0 i prezentuje się następująco:

1,0 – leżący lub siedzący tryb życia, brak aktywności fizycznej,  
1,2 – praca siedząca, aktywność fizyczna na niskim poziomie,  
1,4 – praca niefizyczna, trening 2 razy w tygodniu,  
1,6 – lekka praca fizyczna, trening 3-4 razy w tygodniu,  
1,8 – praca fizyczna, trening 5 razy w tygodniu,  
2,0 – ciężka praca fizyczna, codzienny trening.

Bardziej złożona metoda, która jest opisana wzorem 1.1, pozwoli nam na przybliżone wyliczenie zapotrzebowania.

(1.1)

BMR jest zależne od płci i zostało przedstawione wzorami (1.2) oraz (1.3), odpowiednio dla kobiet oraz mężczyzn.

(1.2)

(1.3)

Waga podawana jest w kilogramach, wzrost w centymetrach i wiek w latach.

**TEA -** w zależności od rodzaju treningu dodawane są różne ilości kcal wyznaczane intensywnością wykonywanych ćwiczeń. Dla treningu siłowego dodawane jest od 7-9 kcal na minutę, natomiast dla treningu aerobowego od 5-10 kcal.

**NEAT** **–** zależny jest od typu budowy ciała, dla ektomorfika od 700 do 900kcal, dla mezomorfika od 400 do 500 kcal i dla endomorfika od 200 do 400 kcal.

**TEF** **-** równy od 6 do 10 % całkowitego dziennego zapotrzebowania kalorycznego.

Poniżej znajduje się przykładowe wyznaczenie zapotrzebowania kalorycznego dla osoby ważącej 70 kg, w wieku 22 lat o typie ciała endomorfik, która wykonuje pracę fizyczną i trenuje 4 razy w tygodniu po 90 minut dla średniej intensywności:

1. **BMR** =
2. W zależności od celu, przy próbie nabrania lub utraty masy ciała należy do końcowego wyniku dodać lub odjąć 300 kalorii.

## Postrzeganie kalorii

Istnieje wiele kontrowersji dotyczących pomiaru spożytego jedzenia w postaci kcal. Sama jednostka powstała około 130 lat temu, naukowcy spierają się co do jej wiarygodności [5]. Dowodami na obalenie teorii kalorii są:

1. Badania prof. Charlesa Libera wykonane w latach 80 z użyciem alkoholu uważanego za bombę kaloryczną udowodniły, że długotrwałe spożywanie dużych ilości alkoholu nie ma wpływu na masę ciała.
2. Na przełomie 20 lat przeprowadzono badania, które wykazały, że orzechy nie przyczyniają się do nadmiernego przyrostu masy ciała, a ich spożywanie może być pomocne w redukcji masy ciała.
3. W 2003 roku przeprowadzono badania na grupie 50-latków, którzy spożywali tę samą pulę kalorii, różniącą się od siebie ilością konsumowanych makroskładników. Różnica związana była z ilością spożywanych węglowodanów i tłuszczów. Ku zdumieniu wszystkich okazało się, że po 12 tygodniach osoby będące na diecie wysokowęglowodanowej schudły 8 kg, a na tłuszczowej 10 kg.

Są też zwolennicy między innymi prof. Thomas Sanders z King ‘s College London, który uważa, że kalorie są sobie równe.

Prof. Alan Jackson z Uniwersytetu w Southampton uważa, że obliczenia dokładnego zapotrzebowania kalorycznego dla danej osoby jest trudne, jednak aktualny sposób na obliczanie kalorii sprawdza się i jest w zupełności wystarczający.

Nie należy całkowicie opierać planu żywieniowego na zapotrzebowaniu kalorycznym. Są one przydatnym elementem diety i w połączeniu z podejmowaniem rozsądnych decyzji żywieniowych przyczynią się do osiągnięcia zamierzonych celów dotyczących sylwetki.

## Typologia Sheldona

William Herbert Sheldon, Jr - żył na przełomie XIX i XX wieku, był amerykańskim psychologiem, zaproponował typologię dzielącą ludzi, w zależności od typu sylwetki, na trzy grupy [8].

Typ sylwetki każdego człowieka jest przedstawiony za pomocą trzech cyfr, od 1 do 7, które określają poziom nasilenia endomorfii, mezomorfii i ektomorfii.

Trzy skrajne typy budowy ciała człowieka to:

* typ ektomorficzny (ektomorfik) – numer 117, jest to osoba wysoka, smukła,
* typ mezomorficzny (mezomorfik) – numer 171, jest to osoba postawna, umięśniona,
* typ endomorficzny (endomorfik) – numer 711, jest to osoba niska, krępa.

Typy te cechują się wysokim natężeniem jednej z cech kosztem pozostałych [7]. W przyrodzie typy te są rzadko spotykane, większość ludzi należy do typów mieszanych. W teorii istnieje 7³ = 343 możliwych typów budowy ciała człowieka. W praktyce liczba ta zaokrągla się do około 80 typów.

System Sheldona jest krytykowany za nieprawidłowe odwzorowanie typów budowy ludności spoza Europy. Brakuje stopni dla skrajnych moezomorfików czy też ektomorfików. Typologia zakłada, że somatotyp jest niezmienny i nawet głodzony endomorfik nie stanie się ektomorfikiem, komórki tłuszczowe ulegną pomniejszeniu się natomiast ich liczba się nie zmieni. Powstały także stereotypy oparte na trzech skrajnych typach budowy ciała. Endomorficy z założone są wolni oraz leniwi, mezomorficy zazwyczaj są postrzegani jako osoby popularne i ciężko pracujące, ektomorficy jako inteligentni, uprawiający sporty długodystansowe między innymi bieg maratoński.

1. Technologie użyte w projekcie

## ASP.NET.Core

Głównym założeniem projektu było użycie nowej technologii firmy Microsoft **ASP.NET.Core,** co przyczyniło się do wyboru specjalnych narzędzi współpracujących z tą technologią [6]. Początkowo, framework firmy Microsoft miał nosić nazwę **ASP.NET 5** lub **ASP.NET.vNEXT**. Stwierdzono, iż nowa wersja nie jest aktualizacją swoich poprzedników, a projektem nowej generacji. Ostatecznie w momencie wydania wersji 1.0 zmieniono nazwę na **ASP.NET.Core**.

Głównymi cechami **ASP.NET.Core** są:

1. Otwarte oprogramowanie (ang. Open-Source). Ta cecha jest niezwykle korzystna dla rozwoju projektu. Każdy użytkownik może przyczynić się do zmiany kodu źródłowego, co skutkuje większą niezawodnością oraz optymalizacją oprogramowania. Użytkownik nie jest zobowiązany do zapłaty za oprogramowanie i w przypadku, gdy kod źródłowy posiada błędy, może pomóc w ich naprawie.
2. Obsługa multi platformowa, dzięki której nie tylko użytkownicy systemu Windows, ale także IOS i Linuxa mogą pracować z frameworkiem.
3. Modularność poprzez paczki NuGet. Właściwość ta umożliwia wstrzykiwanie do projektu tylko niezbędnych modułów aplikacji, a także wyłączenie podstawowych funkcjonalności, między innymi sesji, MVC, czy też plików statycznych, dzięki czemu kompilacja i praca frameworka zostaje przyspieszona.
4. Możliwość odświeżania strony bez potrzeby przebudowywania solucji po dokonaniu zmian, przyspiesza to pracę programisty i poprawia komfort programowania. Funkcjonalność ta jest dobrze znana programistom PHP, w odróżnieniu od programistów .Net, dla których jest rozwiązaniem innowacyjnym.

## Visual Studio 2015

**ASP.NET.Core** jest ściśle powiązany ze zintegrowanym środowiskiem programistycznym o nazwie **Visual Studio** i można go używać w implementacji od wersji 2013. Sam framework Core nie wymaga środowiska programistycznego, dzięki wbudowanym komendom **dotnet restore, dotnet run oraz dotnet watch**, pozwalającym na budowę projektu. W projekcie została zawarta struktura MVC, wspierana przez **Visual Studio**, co przyczyniło się do wyboru tego środowiska programistycznego.

Głównymi dodatkami do wersji 2015 w odróżnieniu do jego poprzedników są:

1. Rozbudowa analizy kodu w czasie rzeczywistym, będąca odpowiednikiem płatnej wtyczki do **Visual Studio** o nazwie **resharper**.
2. Poprawa narzędzia diagnostycznego w czasie rzeczywistym, które podczas analizy kodu umożliwia podgląd wydajności wybranych fragmentów kodu pod kątem obciążenia procesora, a także wykorzystania pamięci.
3. **CodeLens** dostępny tylko w wersji **Professional** oraz **Enterprise** **Visual Studio**, prezentuje historię zmian dla metod oraz klas, ułatwiając kontakt z członkami zespołu, odpowiedzialnymi za dokonane zmiany.
4. Obsługa wielu kont pozwala na łatwy dostęp do zasobów online w przypadku, gdy posiadamy wiele kopii programu **Visual Studio;** upraszcza to dodawanie oraz usuwanie kont za pomocą menadżera.

Obecnie powstaje wersja 2017 RC (ang. Release Candidate). Główny nacisk nowej wersji został położony na rozwój aplikacji webowych oraz programowania multiplatformowego.

## Entity Framework Core

Entity Framework Core jest lekkim, rozszerzalnym i multiplatformowym narzędziem bazującym na równolegle rozwijanym projekcie Entity Framework. Core w wersji 1.0 jest przepisanym, udoskonalonym oraz rozbudowanym o nowe funkcjonalności względem swojego poprzednika w wersji 6.X. Głównym założeniem Core było, aby programista był zaznajomiony z frameworkiem i mógł przejść z wersji 6.X do wersji Core 1.0. Wszystkie komponenty bazowe zostały przepisane, co skutkuje poprawą wydajności oraz usunięciem zbędnych funkcjonalności, na rzecz nowych, które nie występują w wersji 6.X. Są to między innymi:

* alternatywne klucze (ang. alternate keys),
* ewaluacja zapytań LINQ.

W nowym wydaniu zostaną dodane funkcjonalności takie jak:

* **leniwe ładowanie** (ang. lazy loading), polegające na wykonaniu zapytania do bazy danych dotyczącego obiektów aktualnie wykorzystywanych z pominięciem elementów nieużywanych,
* **connection resiliency**, które w przypadku niepowodzenia wykonuje ponowne zapytanie do bazy danych.

Entity Framework Core upraszcza sposób odtworzenia relacji bazodanowych za pomocą modeli zaimplementowanych na platformie .NET, a także eliminuje większość kodu łączenia z bazą danych, tworzonego przez deweloperów. Entity framework Core używa modelu dostawczego, co pozwala na wykorzystanie go w różnorodnych bazach danych.

## GitHub

Kolejnym, kluczowym elementem ułatwiającym pracę jest narzędzie kontroli wersji, pozwalające na podgląd zmian w historii projektu oraz powrót do poprzedniej wersji. To zabezpiecza twórców aplikacji przed utratą oprogramowania w przypadku awarii sprzętu i umożliwia pracę zdalną bez potrzebny ciągłego przenoszenia projektu, w przypadku pracy na wielu maszynach. Kontrola wersji jest niezbędna, gdy prace nad projektem podejmuje zespół.

Najpopularniejszymi narzędziami kontroli wersji są **Git** i **SVN** [9]. **SVN** jest starszy i prostszy w użyciu poprzez większy zasób narzędzi, które powstały na przełomie 16 lat. W przypadku, gdy użytkownikami kontroli wersji są graficy, menadżerowie lub użytkownicy nie będący developerami, jest on korzystniejszy. **Git** natomiast jest najlepszym rozwiązaniem w przypadku zarządzania dużym projektem w licznym zespole. Jego kolejną zaletą jest strona **GitHub**, która zamienia programowanie w działalność społeczną. Strona jest serwerem udostępniającym repozytorium przy użyciu **Gita**, pozwalająca na współużytkowanie zasobów umieszczonych na serwerze, a także przechowywanie kopii naszego repozytorium za pomocą serwerów Githubowych.

## SourceTree

Sam **GitHub** jak i **Git** posiadają swoją własną wersję desktopową prezentującą zmiany w historii projektu przy użyciu grafów oraz umożliwiają wykonanie poleceń za pomocą przycisków, bez konieczności wpisywania komend ręcznie. Wersję te są darmowe, jednak nieczytelne i posiadające braki w funkcjonalności względem aplikacji konkurencyjnych. Dobrym zamiennikiem wyżej wymienionych aplikacji jest **SourceTree**, który wspiera zarówno **Git** jak i **SVN**. Główną zaletą **SourceTree** jest przeszukiwanie systemu, w celu wyznaczenia i dodania używanych repozytoriów.

## Semantic UI

W celu przyśpieszenia pracy nad projektami, a także poprawy wyglądu warstwy wizualnej aplikacji, używa się gotowych frameworków. Najczęściej używanymi są [3]:

* Bootstrap,
* Semantic UI,
* Foundation.

**Bootstrap** jest najbardziej popularnym frameworkiem; został utworzony przez firmę Twitter w roku 2011. Posiada rozległą dokumentację, jest dobrym wyborem dla osób początkujących, jednak z powodu dużych zmian w wyglądzie stron, zawiera wiele klas HTML jak i elementów DOM, co może wprowadzać zamieszanie w kodzie programistów.

**Semantic UI** podobnie jak **Bootstrap** zawiera bogaty opis, czym skłania ku sobie osoby zaczynające swoją przygodę z programowaniem interfejsu użytkownika. Jest przeznaczony do tworzenia prostych, przejrzystych stron. W przypadku próby utworzenia bardziej złożonych i skomplikowanych układów, **Semantic** posiada braki i staje się uciążliwy dla programisty.

**Foundation** w odróżnieniu od poprzedników jest bardziej złożonym frameworkiem i wymaga większego doświadczenia w użyciu.

**Semantic UI** jest najlepszym rozwiązaniem dla projektu przede wszystkim przez swoją prostotę oraz przyjętą konwencję nazewnictwa klas, która jest intuicyjna i w praktyce przyswaja się o wiele szybciej niż w przypadku konkurencyjnych frameworków. Został on wybrany z pośród trzech wyżej wymienionych narzędzi.

1. Implementacja serwisu

## Struktura projektu

Cała architektura projektu bazuje na wzorcu projektowym Model-Widok-Kontroler (MVC). W solucji zawarty jest projekt główny o nazwie **PersonalTrainer**, mieszczący w sobie punkt wejściowy aplikacji, wraz z konfiguracją i wszystkimi referencjami do bibliotek zewnętrznych zadeklarowanych w pliku **project.json**. Biblioteka Framework posiada implementację oraz deklarację serwisów, modeli, atrybutów, zasobów językowych, a także połączeń bazodanowych i służy do oddzielenia strony biznesowej od warstwy wizualnej.

Dodatkowo, projekt został podzielony na moduły, co upraszcza rozbudowę funkcjonalności oraz zmianę wyglądu aplikacji w dalszym rozwoju projektu.



Rys. 3.0 Schemat projektu

Solucja przedstawiona na rys. 3.0 zawiera moduły główne **PersonalTrainerCore** oraz **PersonalTrainerCore.WebGUI**, pełniące rolę modułów bazowych dla widoków i kontrolerów, które w swojej implementacji posiadają rdzenne komponenty wykorzystywane w obrębie całej aplikacji. Poza modułami rdzennymi zostały wydzielone moduły diety, odpowiedzialne za zarządzanie logiką i widokami tematyki związanej z dietetyką, oraz moduły panelu administracyjnego.

Serwis MVC odpowiadający za dobór kontrolerów i widoków, posiada wbudowany mechanizm wyszukiwania ich za pomocą plików assembly. Dzięki temu mechanizmowi można zadeklarować moduły, których dołączenie do projektu będzie zależne od konfiguracji startowej aplikacji. W celu przechowania informacji o danym module, została utworzona klasa pomocnicza przedstawiona na rys. 3.1.



Rys. 3.1 Klasa przechowująca informację o module.

Klasa składa się z nazwy modułu oraz pliku assembly. Plik assembly zostaje wykorzystany do zasilenia serwisu MVC, natomiast nazwa modułu zabezpiecza przed możliwością zduplikowania go.

W celu pozyskania wszystkich modułów z solucji, zaimplementowana została metoda o nazwie **GetModules**.



Rys. 3.2 Metoda pozyskująca informacje o modułach.

Algorytm pobrania poszczególnych modułów z solucji przedstawiony na rys. 3.2 prezentuje się następująco:

1. Wyszukanie folderu Modules w projekcie głównym **PersonalTrainer**.

2. Pobranie z katalogu Modules wszystkich ścieżek modułów.

3. Iteracyjne przeszukiwanie ścieżek w celu wyszukania folderów bin.

4. Pozyskanie wszystkich plików o rozszerzeniu dll znajdujących się w folderach bin.

5. Odczyt z dll pliku assembly zawierającego podstawowe informacje o module.

6. Dodanie modułów do listy przy użyciu klasy **ModuleInfo**.

7. Zwrócenie kolekcji modułów.

## Nawigacja

W aplikacji zastosowano standardową obsługę nawigacji między żądaniami (rys. 3.3), przy użyciu wbudowanej metody **MapRoute** występującej w serwisie MVC. Skonfigurowano ją za pomocą szablonu startowego, który podczas uruchomienia strony internetowej wywołuje kontroler **Home** i akcje **Index**.



Rys. 3.3 Metoda zarządzająca nawigacją aplikacji.

Standardowo w aplikacji ASP.NET.Core kontrolery i widoki są wyszukiwane w obrębie głównego projektu w folderach **Controllers** oraz **Views**. Ze względu na podział aplikacji na moduły i oddzielenie części wizualnej od biznesowej, zadeklarowane zostało rozszerzenie przeszukujące moduły widoków, poprzez implementację klasy **ModulesViewLocationExpander** przekazanej do metody **ViewLocationExapnders (**rys. 3.4) w konfiguracji aplikacji.



Rys. 3.4 Konfiguracja wyszukiwania dodatkowych widoków.

Metoda **ExpandViewLocations** przedstawiona na rys. 3.5 bazuje na wyszukiwaniu widoków w obrębie modułu rdzennego oraz modułu, w którym aktualnie znajduje się użytkownik. Przeszukane zostają foldery Views, zawierające implementację widoków i kontrolerów oraz View/Shared posiadające widoki współdzielone dla danego modułu.



Rys. 3.5 Metoda odpowiedzialna za wskazanie ścieżki wyszukiwania widoków.

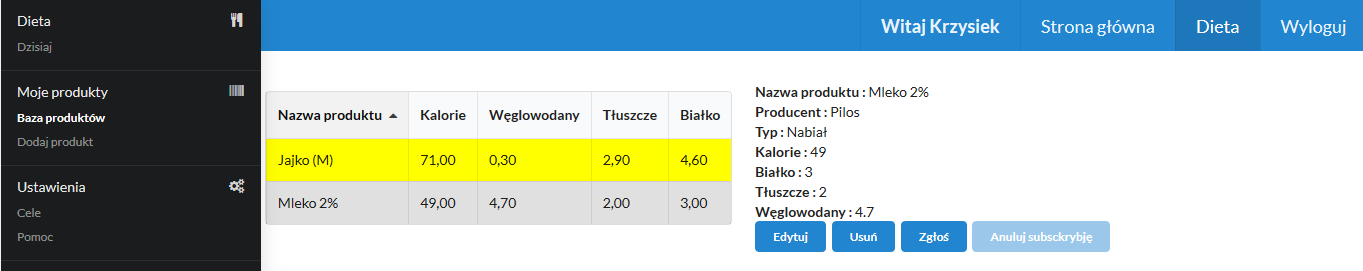
Metoda **PopulateValues** przedstawiona na rys. 3.6 pozwala na zapisanie w zmiennej kontekstowej aktualnego modułu, w którym znajduje się użytkownik. Zmienna zostaje wykorzystana w celu wyszukania widoku w metodzie **ExpandViewLocations**.



Rys. 3.6 Metoda odpowiedzialna za wskazanie aktualnego modułu.

## Mechanizm zgłaszania produktów

Jedną z podstawowych funkcjonalności aplikacji jest dodawanie produktów. Każdy użytkownik ma możliwość dodawania własnych jak i korzystania z gotowych produktów dostarczonych przez innych użytkowników systemu. Nowo utworzony artykuł jest prywatny, przez co tylko twórca może z niego korzystać (rys. 3.7). Ma to na celu zabezpieczenie aplikacji przed dodawaniem nieprawidłowych produktów zawierających błędne informacje na temat nazwy, kalorii i makroskładników. Aby móc udostępniać własne produkty, należy skorzystać z mechanizmu zgłaszania, polegającego na zgłaszaniu produktów do listy artykułów spożywczych oczekujących na rozpatrzenie. Do tej listy mają dostęp wyłącznie administratorzy strony internetowej. W trakcie oczekiwania na zatwierdzenie bądź odrzucenie produktu, użytkownik zgłaszający artykuł posiada do niego pełne prawa. W przypadku, gdy produkt został zatwierdzony przez administratora, staje się własnością systemu, natomiast twórca artykułu traci prawo do usuwania lub edytowania elementu. W momencie zatwierdzenia elementu, administrator ma możliwość wycofania go, co skutkuje odzyskaniem uprawnień przez twórcę produktu i pozwala na ponowne zgłoszenie do systemu.



Rys. 3.7 Lista produktów dodanych przez użytkownika.

## Baza danych

Połączenie bazy danych z modelami biznesowymi zostało zrealizowane przy użyciu frameworka **Entity Framework Core,** za pomocą konwersji zdefiniowanej w kontekście, której zadaniem było utworzenie elementom bazy ich odpowiedników modelowych. W aplikacji stworzono pojedynczy kontekst o nazwie **DefaultContext**, który został zadeklarowany podczas startowej konfiguracji aplikacji (rys. 3.7).



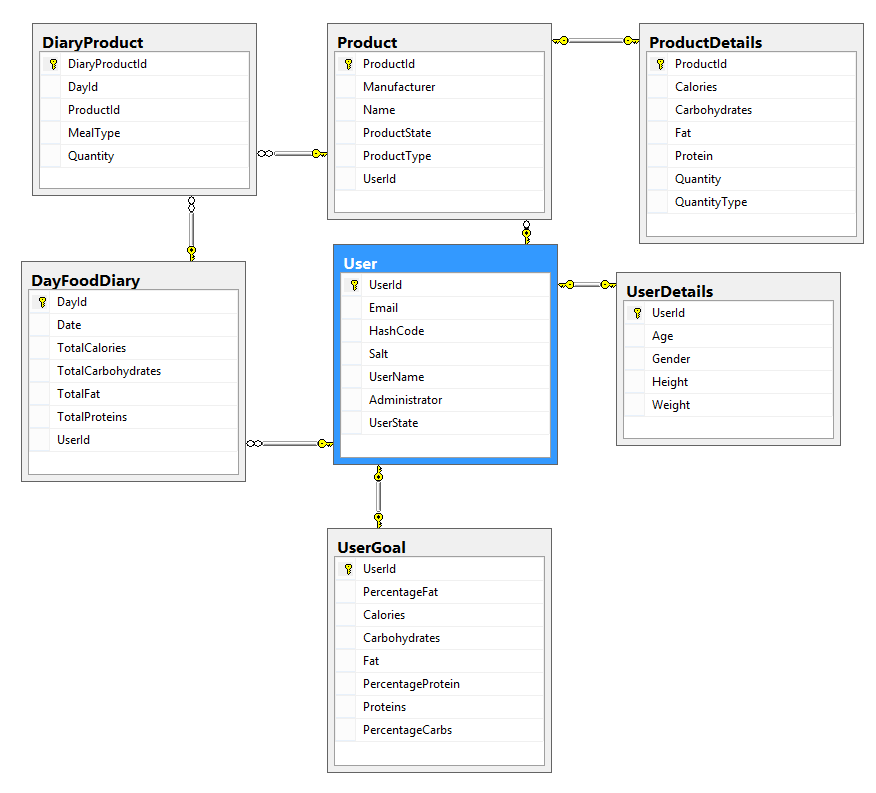
Rys. 3.7 Ustawienia kontekstu bazodanowego.

Relacje bazodanowe zawarte w projekcie w głównej mierze zostają wygenerowane za pomocą atrybutów przypisanych poszczególnym modelom tabel. Głównymi atrybutami wykorzystanymi w projekcie są:

* **Required -** służący do oznaczenia pola tabeli jako wymagane,
* **Table** **-** generujący tabelę na podstawie klasy, do której został przydzielony,
* **Key** **-** definiujący daną właściwość, jako identyfikator modelu,
* **ForeignKey -** definiujący daną właściwość, jako identyfikator pomocniczy modelu, służący do łączenia ze sobą dwóch tabel.

Atrybuty są prostsze w użyciu, natomiast nie pozwalają na odwzorowanie każdej zależności. Do tego celu służy **fluent appi** wbudowane w Entity framework, które jest wykorzystywane z poziomu kontekstu w momencie tworzenia relacji.

Cała logika biznesowa wraz z połączeniem z bazą danych oraz atrybutami i zasobami językowymi znajduje się w bibliotece Framework. Jest ona niezależną biblioteką, nieposiadającą powiązań z żadnym modułem, dzięki czemu pozwala na przeniesienie całej logiki aplikacji do nowego, zawierającego inną strukturę projektu.



Rys. 3.8 Schemat bazy danych

Kluczową tabelą w schemacie bazy danych przedstawionym na rys. 3.8 jest użytkownik (User), który został powiązany z większością tabel w projekcie. Posiada on dwie relację „jeden do jednego” z tabelami „szczegóły użytkownika” (UserDetails) oraz „cel użytkownika” (UserGoal), a także dwie relację „jeden do wielu” z produktami (Product) i dniami żywieniowymi (DayFoodDiary) dodawanymi przez użytkowników systemu. Dodatkowo każdy produkt zawiera informację dodatkowe (ProductDetails), między produktami a dniami żywieniowymi występuje relacja „wielu do wielu”, przez co powstała tabela pomocnicza (DiaryProduct) umożliwiająca stworzenie powiązań [1].

## Szyfrowanie danych użytkownika

Kolejnym ważnym element aplikacji sieciowej, odpowiedzialnej za zarządzanie kontami jest ich zabezpieczenie [2]. W tym celu, w bazie nie przechowuje się haseł, a funkcje skrótu, które zostają wygenerowane w momencie tworzenia lub logowania do konta. Funkcja ta zostaje utworzona za pomocą wybranego algorytmu hashującego, przy użyciu hasła podanego przez użytkownika oraz soli stworzonej w momencie generowania funkcji.

Poniżej przedstawiony został algorytm zapisu hasła w bazie danych w momencie rejestracji użytkownika:

1. Wprowadzenie hasła przez użytkownika systemu podczas rejestracji.
2. Wygenerowanie soli za pomocą metody przedstawionej na rys. 3.9, w celu zaszyfrowania hasła poprzez wbudowaną klasę **RandonNumberGenerator** w frameworku .Net.



Rys. 3.9 Metoda tworząca sól

1. Dodanie hasła do soli poprzez połączenie bajtów, a następnie wykorzystanie funkcji skrótu SHA512 na połączonych bajtach (rys. 3.10).
2. Zapis funkcji hashującej oraz soli w bazie danych.



Rys. 3.10 Metoda generująca funkcję skrótu przy użyciu hasła i soli.

Algorytm walidacji hasła podczas próby logowania prezentuje się następująco:

1. Pobranie z bazy danych funkcji skrótu oraz soli dla podanego użytkownika.
2. Wygenerowanie funkcji skrótu przy użyciu hasła podanego przez użytkownika systemu.
3. Porównanie dwóch funkcji i zwróceniu flagi w zależności od zgodności danych.

## Korzystanie z serwisu

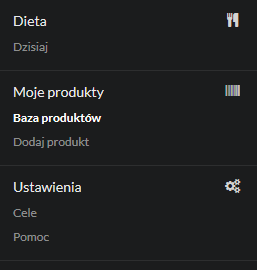
Przy starcie aplikacji pojawia się możliwość zalogowania lub rejestracji, przy której przyszły użytkownik serwisu zobowiązany jest do podania danych takich jak:

* nazwa użytkownika,
* adres e-mail,
* hasło zabezpieczające konto,
* potwierdzenie hasła,
* płeć,
* wzrost,
* waga,
* wiek.

Do utworzenia konta wymagane są niestandardowe dane dotyczące wzrostu i wagi, które będą wykorzystane w dalszym rozwoju aplikacji. W momencie, gdy posiadamy już istniejące konto, logowanie odbywa się poprzez wpisanie nazwy użytkownika oraz hasła.

Po zalogowaniu użytkownik ma możliwość przejścia do strony głównej, panelu diety lub do panelu administracyjnego w przypadku, gdy posiada uprawnienia administracyjne. W panelu diety znajdują się główne funkcjonalności aplikacji, które zostały pogrupowane i przydzielone do trzech grup przez lewy panel (rys. 3.11). Grupy te prezentują się następująco:

* Dieta - odpowiedzialna za zarządzanie dniami żywieniowymi,
* Moje produkty - pełniąca funkcję zarządzania produktami użytkownika,
* Ustawienia - określająca cele użytkownika oraz przedstawiająca instrukcję obsługi panelu diety.



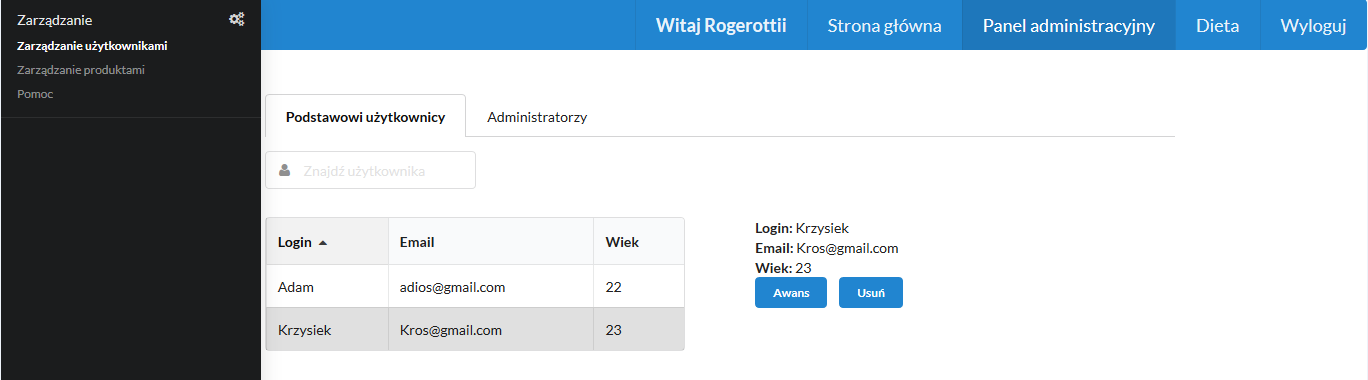
Rys. 3.11 Panel nawigacyjny występujący w module diety.

Panel administracyjny dzieli się na trzy typy funkcjonalności, takie jak:

* zarządzanie produktami,
* zarządzanie użytkownikami,
* pomoc.

„Zarządzanie produktami” jest funkcjonalnością umożliwiającą administratorom wgląd do listy artykułów oczekujących, rozpatrzenie zgłoszenia oraz podjęcie decyzji o przyjęciu lub odrzuceniu żywności.

Dzięki tej funkcjonalności w serwisie nie będą upubliczniane artykuły zawierające błędne informacje na temat nazwy, makroskładników oraz kalorii. „Zarządzanie użytkownikami” (rys. 3.12) daje administratorowi możliwość usunięcia konta użytkownika z serwisu, mianowania na administratora bądź zwolnienia z pełnionej funkcji. Ostatnią funkcjonalnością panelu administracyjnego jest „Pomoc”, w której zawarte są wszystkie informacje na temat panelu.



Rys. 3.12 Panel zarządzania użytkownikami.

Podsumowanie

Głównym założeniem projektu było stworzenie przejrzystej i prostej w obsłudze strony internetowej opartej o technologię ASP.NET.Core. Strona miała na celu umożliwienie prowadzenia własnego dziennika żywieniowego poprzez wyliczenie spożytych kalorii i makroskładników. Miała pozwalać na ustawianie progu kalorycznego i zarządzania własnymi produktami oraz korzystania z produktów dodanych przez innych użytkowników systemu. Cel ten został osiągnięty, nie obyło się jednak bez problemów. Największym z nich okazał się dynamiczny widok oparty o język JavaScript, jQuery oraz AJAX. Sam język JavaScript uznawany jest przez wielu jako najbardziej problematyczny język głównie ze względu na brak silnego typowania, co powoduje, że kod jest mało czytelny, a kompilator nie jest w stanie wykryć wielu błędów. Kolejnym problemem był podział projektu na moduły, który ostatecznie zakończył się powodzeniem. Dużą wadą ASP.NET.Core jest niekompletna dokumentacja, jak i brak literatury oraz materiałów pomocniczych.

Wbudowane wstrzykiwanie zależności ułatwiło pracę z kodem, pozwoliło na proste odseparowanie od siebie niezależnych struktur zarządzania logiką biznesową oraz umożliwiło pozyskanie serwisów niezależnych, dostarczonych przez użytkowników jak i firmę Microsoft za pomocą paczek NuGet. Praca z systemem kontroli wersji zdała swój egzamin w projekcie jednoosobowym, w momentach awarii aplikacji przyśpieszała pracę poprzez archiwizację poprzedniego programu. Założenia projektowe zostały w pełni zrealizowane dzięki słusznym wyborom technologii, które dobrze dopasowane usprawniły pracę nad projektem.

Pomysł na temat projektu inżynierskiego wynikał z zapotrzebowania oraz rosnącego zainteresowania stronami dietetycznymi. Praca nie wyczerpała w pełni tematu i ma możliwość dalszego rozwoju zarówno od strony graficznej jak i nowych funkcjonalności. Struktura aplikacji została dobrana w sposób usprawniający rozwój projektu poprzez dołączanie nowych funkcjonalności jako odseparowanych od reszty modułów.

Literatura

1. Lynn Beighley – „Head First SQL”. Wydawnictwo O’Reily
2. Adam Freeman, Allen Jones – „Programming .NET Security”.

Wydawnictwo O’Reily

1. Cody Arsenault - Najlepsze frameworki warstwy wizualnej [Online] Dostępne w internecie https://www.keycdn.com/blog/front-end-frameworks/#1-Bootstrap
2. Mariusz Janecki – Metody wyliczania kalorycznego [Online]

Dostępny w internecie http://www.fabrykasily.pl/porady-trenerow/zapotrzebowanie-kaloryczne-wyliczenie

1. Tony Edwards - Artykuł dotyczący postrzeganie kalorii. [Online].

Dostępny w internecie: http://oczymlekarze.pl/zdrowy-styl-zycia/dieta/1417-kaloria-kalorii-nierowna

1. Dokumentacja ASP.NET.Core [Online] Dostępne w internecie https://docs.microsoft.com/en-us/aspnet/core/
2. Opis skrajnych typów ciała człowieka [Online]. Dostępny w internecie http://www.sfd.pl/Ektomorfik,\_Endomorfik\_i\_Mezomorfik\_Sylwetka\_prawd%C4%99\_Ci\_powie\_-t236043.html
3. Typologia Sheldona [Online]. Dostępny w internecie https://pl.wikipedia.org/wiki/Typologia\_Sheldona
4. Porównanie systemów kontroli wersji GIT i SVN [Online]

Dostępny w internecie http://software-engineer-training.com/git-vs-svn-which-is-better