

POLITECHNIKA ŚLĄSKA W GLIWICACH WYDZIAŁ ELEKTRYCZNY

Instytut Elektrotechniki i Informatyki

**PROJEKT INŻYNIERSKI**

**Temat projektu inżynierskiego w języku polskim**

**Temat projektu inżynierskiego w języku angielskim**

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

Studia:. Stacjonarne, I stopnia  
Kierunek: Informatyka  
Specjalność: Inżynieria elektryczna

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

Słownik pojęć - 3 -

Wstęp - 4 -

1 Temat1 - 5 -

1.1 Statystyki otyłości - 5 -

1.2 Sposób liczenia makroskładników - 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 - 16 -

2.5 SourceTree - 16 -

2.6 Semantic UI - 17 -

3 Implementacja - 18 -

3.1 Struktura projektu - 18 -

3.2 Nawigacja - 21 -

3.3 Baza danych - 23 -

3.4 Szyfrowanie danych użytkownika - 25 -

Podsumowanie - 27 -

Literatura - 28 -

Słownik pojęć

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

Sól (salt) – losowe dane dodawane do hasła podczas obliczania funkcji skrótu. Sól zapobiega przed atakami słownikowymi 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, drobnymi kościami, chudymi oraz długimi kończynami, a także wąskimi ramionami.

Endomorfik - typ budowy ciała cechujący się masywną, owalną sylwetką, mający tendencję do tycia po przez 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 rozpiski 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, która pozwala na wyliczenie zapotrzebowania kalorycznego w prosty i wygodny sposób, zapamiętywania naszych postępów oraz planowania bilansu kalorycznego. Możliwe jest dodawanie własnych produktów jak i korzystanie z produktów dodanych przez innych użytkowników korzystających ze 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 zdeterminowania optymalnego spożycia kalorii dla użytkownika. W 2015 roku została wykupiona przez firmę **Under Armor**, która obecnie jest właścicielem serwisu internetowe. Strona posiada także własną aplikację na telefony IOS oraz Android.
* **Lose it** - kolejna strona poświęcona liczeniu zapotrzebowania kalorycznego. Wyróżnia się innowacyjnym pomysłem robienia zdjęć produktów które zostaje przetworzone i dodane do naszej puli kalorycznej.

1. Temat1

W tym rozdziale zostaną poruszone tematy dotyczące kalorii i makroskładników jak i sama przyczyna nagłego zainteresowania dietą przez społeczeństwo w Polsce jak i na całym świecie. Zostaną przedstawione podstawowe metody wyliczania zapotrzebowania kalorycznego, a także podział makroskładników w diecie. Na koniec rozdziału omówiona zostanie słynna typologia Sheldona, która pozwala w przybliżonym stopniu uwzględnić budowę oraz genetykę ciała człowieka wykorzystywaną w wyliczaniu kalorii i makroskładników.

## 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 prezentują się następująco.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 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 tabeli 1.0 z badań przeprowadzonych w latach 2006 i 2007, w województwie świętokrzyskim jako jedynym, prawidłowa waga przekraczała 50%. Przeciętnie 1/3 społeczeństwa cierpiała na nadwagę oraz ponad 10 % na otyłość. Niedowaga w Polsce oscylowała w granicy 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.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 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

W przeciągu dziesięciu lat w naszym kraju procent niedowagi zmniejszył się. Przyczyniło się to także do znacznego przyrostu osób z nadwagą oraz otyłością. Porównując tabelę 1.0 oraz 1.1 i 1.2 można zaobserwować znaczy przyrost otyłości i nadwagi, która wzrosła o około 4%. Niedowaga zmalała o około 1,2 %. Jest to niekorzystne zjawisko występujące w Polsce jak i na całym świecie. Najgorzej pod tym względem sytuacja przedstawia się w krajach najbardziej rozwiniętych. Liderem w rankingu ilości otyłych osób w kraju są Stany Zjednoczone, gdzie aż 1/3 społeczeństwa ma problemy z nadwagą. Przyczyną tego zjawiska jest udoskonalenie świata, skutkuje to brakiem aktywności fizycznej w społeczeństwie.

## Sposób liczenia makroskładników

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 między węglowodanami a tłuszczami. Standardowo zakłada się, że osoba ćwiczące powinna spożywać 2g białka na kg masy ciała.

Dla pierwszej diety ilość tłuszczów w diecie 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.

Dla przykładu diety wysokowęglowodanowej. Osoba ważąca 70 kg i spożywająca 2500 kcal powinna zjeść około 140g białka, 40g tłuszczy i 395g węglowodanów każdego dnia. W przypadku diety wysokotłuszczowej 140g białka, 40g węglowodanów i 198g tłuszczy.

## Zapotrzebowanie kaloryczne.

Występuje kilka sposobów na wyliczenie zapotrzebowania kalorycznego.

Pierwsza, najprostsza metoda określona wzorem w wielu przypadkach się nie sprawdza.

Wzór 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 nie fizyczna, 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.

Wzór 1.1

BMR w zależności od płci, dla kobiet przedstawione wzorem 1.2 i dla mężczyzn wzorem 1.3.

Wzór 1.2

Wzór 1.3

Gdzie waga podawana w kilogramach, wzrost w centymetrach i wiek w latach.

TEA - w zależności od rodzaju treningu. Dla treningu siłowego dodawane jest od 7-9 kcal na minutę w zależności od intensywności i dla aerobowego od 5-10 kcal.

NEAT - odpowiadający typowi budowy ciała dla ektomorfika od 700 do 900kcal, dla mezomorfika od 400 do 500 kcal i dla endomorfika od 200 do 400 kcal.

TEF – pozyskiwany po przez przemnożenie całkowitego wyniku (TDEE) od 6 do 10 %

Przykładowe wyliczenie 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. Dowodami na obalenie teorii kalorii są:

1. Badania prof. Charlesa Libera wykonane w latach 80 za pomocą 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 tą samą pule kalorii, różniącą się od siebie składem makroskładników. Różnica wynikała między rozkładem węglowodanów a 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 kaloria to kaloria.

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

Nie należy na ślepo wierzyć kalorią natomiast nie należy też ich skreślać. Połączenie podejmowania rozsądnych decyzji żywieniowych wraz z obliczaniem spożytych kalorii przyczyni się do osiągnięcia zamierzonych celów dotyczących sylwetki.

## Typologia Sheldona

William Herbert Sheldon, Jr - żył na przełomie XIX i XX wieku, był amerykańskim psychologiem, stworzył topologię, która zakładała, że każdy człowiek należy do jednej z trzech grup budowy ciała.

Typ sylwetki każdego człowieka jest przedstawiony za pomocą trzech cyfr, od 1 do 7, który określa poziom nasilenia w przypadku pierwszej endomorfii, drugiej mezomorfii i trzeciej ektomorfii.

Trzy skrajne typy budowy ciała człowieka.

* 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. 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 z poza 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 ilość 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

Główny założeniem projektu było oparcie się na nowej technologii ASP.NET.Core co przyczyniło się do wyboru specjalnych narzędzi współpracujących z tą technologią.

## ASP.NET.Core

Nowy framework firmy Microsoft, początkowo miał nosić nazwę **ASP.NET 5** jednak żeby nie wprowadzać w błąd użytkownika, iż nowa wersja nie jest aktualizacją wersji 4.6, a nowym wydaniem o numerze 1.0.

Głównymi cechami ASP.NET.Core są:

1. Otwarte oprogramowanie (Open-Source). Jest to bardzo korzystne dla rozwoju projektu. Każdy użytkownik może mieć wpływ na kod źródłowy co przyczyni się do większej niezawodności oraz optymalizacji oprogramowania. Użytkownik nie musi płacić za oprogramowanie i w przypadku, gdy kod źródłowy posiada błędy, może przyczynić się do naprawy.
2. Obsługa multi platformowa dzięki czemu nie tylko użytkownicy Windowsa, ale także IOS i Linuxa będą mogli pracować z frameworkiem.
3. Modularność poprzez paczki NuGet. Umożliwia wstrzykiwanie do projektu tylko tych modułów, które są potrzebne. Możliwość wyłączenia nawet podstawowych funkcjonalności między innymi sesji, MVC czy też używania plików statycznych. Pozwala to na przyśpieszenia pracy jak i kompilacji samego frameworka poprzez usunięcie zbędnego kodu.
4. Możliwość odświeżania strony bez potrzeby przebudowywania solucji po dokonaniu zmian. Pozwala na szybszą pracę programisty a przede wszystkim poprawia komfort programowania. Funkcjonalność ta jest dobrze znana programistą PHP, natomiast dla programistów .NET jest nowością.

## Visual Studio 2015

ASP.NET.Core jest ściśle powiązany z Visual Studio natomiast nie jest niezbędny i istnieje możliwość pisania aplikacji bez niego. Projekt opiera się na strukturze MVC co przyczyniło się do wyboru tego środowiska programistycznego (IDE) firmy Microsoft, które cały czas jest rozwijane. Głównymi dodatkami do wersji 2015 w odróżnieniu do poprzednich wersji jest

1. Rozbudowa analizy kodu na żywo wraz z podpowiedziami, jest odpowiednikiem resharpera, który jest płatną wtyczką do Visual Studio, jednak zawiera kilka dodatkowych funkcjonalności między innymi analiza kodu w języku Java Script.
2. Poprawa narzędzia diagnostycznego w czasie rzeczywistym podczas debugowania kodu. Pozwala na sprawdzaniu wydajności fragmentu kodu pod względem obciążenia procesora, a także wykorzystania pamięci. Możliwe jest ustawienie punktu przerwania na danym fragmencie wykorzystania procesora.
3. CodeLens dostępny tylko w wersji Professional oraz Enterprise Visual Studio.: Znajduje zmiany w kodzie i umożliwia przegląd historii repozytorium co ułatwia kontakt z członkiem zespołu, który dokonał zmian.
4. Obsługa wielu kont pozwala na łatwy dostęp do zasobów online w przypadku, gdy posiadamy wiele kont programu Visual Studio. Łatwy sposób dodawania oraz usuwania za pomocą menadżera nowego konta.

Obecnie powstaje wersja 2017 RC (Release Candidate). Główny nacisk nowej wersji opiera się na rozwój aplikacji webowych oraz programowania multi-platformowego.

## Entity framework Core

Lekkie, rozszerzalne i multi platformowe narzędzie bazujące na Entity Framework, które jest rozwijane równolegle. Jest napisane na nowo w wersji 1.0, udoskonalone oraz rozbudowane o nowe funkcjonalności względem wersji Entity framework 6.X.

Głównym założenie Core było, aby programista był zaznajomiony i mógł przejść z wersji 6.X na wersję Core 1.0, natomiast wszystkie komponenty bazowe zostały przepisane co przyczyniło się do poprawy wydajności. Niektóre funkcjonalności, które były nieprzydatne zostały usunięte i nie pojawią się w nowej odsłonie, natomiast zostały też dodane elementy, których w wersji 6.X nie ma na przykład:

* Alternatywne klucze (alternate keys)
* Mieszana klient/serwer ewaluacja zapytań LINQ

Niektóre z funkcjonalności jak **leniwe ładowanie** (lazy loading), które polega na wykonaniu zapytania do bazy dotyczących tylko obiektów które aktualnie zostają wykorzystywane, czy też

**connection resiliency**, które dokonuje próby ponownego wysłania zapytania do bazy w przypadku niepowodzenia zostaną dodane w nowszym wydaniu.

Narzędzie te w łatwy sposób umożliwia odtworzenie relacji bazodanowych za pomocą modeli zaimplementowanych na platformie .NET, a także eliminuje większość kodu łączenia z bazą danych którą deweloperzy muszą napisać.

Entity framework Core używa modelu dostawczego co pozwala na wykorzystanie go w wielu różnorodnych bazach.

## GitHub

Kolejnym kluczowym narzędziem ułatwiającym pracę jest narzędzie kontroli wersji. Pozwala na przegląd zmian w historii projektu, możliwość cofnięcia się do poprzedniej wersji, zabezpiecza nas przed utratą oprogramowania w przypadku awarii sprzętu, a także umożliwi prace zdalną bez potrzebny ciągłego przenoszenia projektu w przypadku, gdy pracujemy na kilku maszynach. Kontrola wersji jest niezbędny, gdy prace nad projektem podejmuje zespół.

Najpopularniejszymi narzędziami kontroli wersji są Git i SVN. SVN jest starszy i łatwiejszy w użyciu po przez większy zasób narzędzi, które powstały na przełomie 16 lat. W przypadku, gdy użytkownikami kontroli wersji będą graficy, menadżerowie lub użytkownicy nie będący developerami jest on lepszym wyborem. Git natomiast jest dobrym narzędziem w przypadku zarządzania dużymi projektami z wieloma pracownikami. Kolejną zaletą jest strona GitHub, która zamienia programowanie w działalność społeczną co powoduje, że programowanie staje się przyjemniejsze.

GitHub jest serwerem hostującym nasze repozytorium przy użyciu Git. Pozwala on na dzieleniu się własnym repozytorium z innymi użytkownikami, dostępu do repozytorium innych, a także przechowywania kopii naszego repozytorium za pomocą serwerów Githubowych. Sam GitHub bez Git nie istnieje natomiast Git może wykorzystywać inny serwer.

## SourceTree

Sam GitHub jak i git posiada swoją własną wersję desktopową która pozwala na lepsze zobrazowanie zmian za pomocą grafów, wykonanie poleceń za pomocą przycisków bez konieczności wpisywanie komend z konsoli. Wersję te są darmowe jednak mało czytelne oraz funkcjonalne. Dobrym zamiennikiem wyżej wymienionych aplikacji jest SourceTree, który wspiera zarówno Git jak i SVN.

## Semantic UI

W celu przyśpieszeniu pracy nad projektem, a także poprawy wyglądu warstwy wizualnej aplikacji używa się gotowych frameworków. Najczęściej używanymi są:

* Bootstrap
* Semantic UI
* Foundation

Każdy z nich ma swoje wady i zalety.

Bootstrap jest najbardziej popularnym frameworkiem, został utworzony przez firmę Twitter w roku 2011. Posiada dobrą dokumentację, jest dobrym wyborem dla osób początkujących, jednak z powodu dużych zmian w wyglądzie stron na przełomie kilku lat zawiera wiele klas HTML jak i elementów DOM co może powodować zakłopotanie.

Semantic UI zarówno jak Bootstrap zawiera bogaty opis i jest dobry dla osób zaczynających swoją przygodę z front-endem. Jest przeznaczony do tworzenia prostych, przejrzystych stron, w przypadku próby utworzenia bardziej złożonych i skomplikowanych układów strony Semantic posiada braki i może stać się uciążliwy.

Foundation w odróżnienie od poprzedników jest bardziej złożonym frameworkiem co wymaga większego doświadczenia w użytku.

Semantic UI jest najlepszym rozwiązaniem dla projektu przede wszystkim przez swoją prostotę oraz nazewnictwo klas które używają naturalnego języka.

1. Implementacja

W rozdziale tym zostanie przedstawiona implementacja strony internetowej wraz z architekturą systemu.

## Struktura projektu



Rys3.0 Schemat projektu.

Aplikacja powstała na platformie [ASP.NET Core](https://www.asp.net/core). Architektura projektu opiera się na słynnym wzorcu architektonicznym MVC - Model-Widok-Kontroler. W solucji zawarty jest projekt główny **PersonalTrainer**, zawierający punkt wejściowy aplikacji, wraz z konfiguracją i wszystkimi referencjami do bibliotek zewnętrznych zadeklarowanych w pliku **project.json**. Biblioteka Framework służy do odseparowania strony serwerowej od warstwy GUI. Posiada implementację oraz deklarację serwisów, modeli, atrybutów, zasobów językowych, a także połączeń bazodanowych.

Dodatkowo projekt został podzielony na moduły co w łatwy sposób pozwala na odseparowanie funkcjonalności w dalszym rozwoju projektu, a także w razie potrzeby wymiany całego wyglądu aplikacji po przez przepięcie modułów **WebGUI**.

Solucja przedstawiona na rys.3.0 posiada moduły główne **PersonalTrainerCore** oraz **PersonalTrainerCore.WebGUI**. Pierwszy z nich zawiera kontrolery rdzenne wykorzystywane w kilku modułach. Pozostałe moduły posiadają referencje do modułu głównego. W projekcie zostały także wydzielone moduły widoków, które także posiadają swój moduł główny. Po za modułami rdzennym zostały wydzielone moduły diety odpowiedzialne za zarządzanie logiką i widokami tematyki związanej z dietetyką.

Serwis MVC odpowiadający za dobór kontrolerów i widoków w zależności od wykonywanej akcji. Posiada zaimplementowany mechanizm, który dodaje dodatkowe pliki assembly, w których będą szukane odpowiednie kontrolery i widoki. Dzięki temu w łatwy sposób możemy wpiąć lub wypiąć moduł w zależności od konfiguracji startowej aplikacji. Do dokonania podziału została utworzona klasa pomocnicza, która przechowuje informację dotyczące danego modułu przedstawiona na rysunku 3.1.



Rys.3.1 Klasa przechowująca informację o module.

Klasa składa się z nazwy modułu oraz pliku assembly. Plik assembly zostaje wykorzystywany do zasilenia serwisu MVC, nazwa ma na celu zabezpieczenie przed zduplikowaniem modułu o tej samej nazwie.

Do pozyskania wszystkich modułów z solucji została zaimplementowana metoda o nazwie **GetModules**.



Rys.3.2 Metoda pozyskująca informacje o modułach.

Algorytm wydobycia poszczególnych modułów z solucji i wdrożenia ich do serwisu MVC przedstawiony na rys.3.2.

1.Wyszukanie folderu Modules w strukturze aplikacji w projekcie głównym PersonalTrainer.

2.Pobranie z katalogu Modules wszystkich ścieżek modułów.

3.Iteracja pozyskanych ścieżek w celu wyszukania folderów bin.

4.Wydobycie wszystkich plików dll. Znajdujących się w folderach bin.

5.Odczyt z pliku dll assembly zawierających informacje o modułach.

6.Dodanie do listy informacji o modułach które jeszcze nie zostały dodane.

7. Zwrócenie modułów.

## Nawigacja

W aplikacji zastosowana została standardowa obsługa nawigacji pomiędzy stronami za pomocą wbudowanego routingu występującego w serwisie MVC, przy pomocy metody **MapRoute** przedstawionym na rysunku 3.3



Rys.3.3 Metoda zarządzająca routingiem aplikacji.

W metodzie został zdefiniowany szablon wywołujący startowo kontroler Home i akcji Index. Standardowo w aplikacji ASP.NET.Core kontrolery i widoki są poszukiwane w obrębie głównego projektu w folderach **Controllers** oraz **Views**. Ze względu na podział aplikacji na moduły i odseparowanie widoku zostało zadeklarowane rozszerzenie wyszukujące widoków. Nawigacja w poszukiwaniu widoków została skonfigurowana po przez implementację klasy **ModulesViewLocationExpander** zawierającej interfejs **IViewLocationExpander** i przekazana do metody **ViewLocationExapnders** rys 3.4 w konfiguracji aplikacji



Rys.3.4 Konfiguracja wyszukiwania dodatkowych widoków.

Metoda **ExpandViewLocations** przedstawiona na rysunku 3.5 bazuje na wyszukiwanie widoków w obrębie Modułu Core (Rdzennego) jak i modułu, w którym aktualnie znajduje się użytkownik. Przeszukuje zarówno foldery Views jak i foldery Views/Shared posiadające widoki współdzielone dla danego modułu a w przypadku Core dla całej aplikacji.



Rys.3.5 Metoda odpowiedzialna za wskazanie ścieżki wyszukiwania widoków.



Rys.3.6 Metoda odpowiedzialna za wskazanie aktualnego modułu.

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

## Baza danych

Połączenie bazy danych z modelami biznesowymi zostało zrealizowane za pomocą Entity Framework Core. Modele zostają mapowane na poziomie kontekstu. W aplikacji został utworzony pojedynczy kontekst o nazwie **DefaultContext**.

Kontekst jest definiowany podczas startowej konfiguracji aplikacji rysunek 3.7.



Rys.3.7 Ustawienia kontekstu bazodanowego.

Relacje bazodanowe są utworzone przy pomocy atrybutów podpiętych pod szczególne modele tabel między innymi atrybut **Table**, który deklaruje, że dany model odpowiada modelowi tabeli, czy też **Key** definiujący daną właściwość jako identyfikator modelu bazodanowego. Atrybuty są przyjemne w użyciu, natomiast nie można odwzorować każdej zależności za ich pomocą. Do tego celu służy **fluent appi** z 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, nie posiadający referencji do żadnego modułu dzięki czemu pozwala na całkowite wypięcie całej logiki biznesowej z jednego projektu i wpięcie do innego posiadającego całkowicie inna warstwę wizualną aplikacji.



Rys.3.8 Schemat bazy danych

Kluczową tabelą w schemacie bazy danych przedstawiony na rysunku 3.8 jest użytkownik (User). Jest on powiązany z większością tabel. Posiada on relację 1-1 z tabelą reprezentującą szczegóły użytkownika (UserDetails), czy też celami ustawionymi przez właściciela konta (UserGoal). Dodatkowo występują dwie relację 1- ∞. Pierwsza z nich zachodzi między dziennymi wpisami w dzienniku żywieniowym (DayFoodDiary), a użytkownikiem, który ma możliwość zawarcia wielu dni żywieniowych. Druga relacja 1-  ∞ występuje między produktami (Product), a użytkownikiem, który analogicznie jak w przypadku dni żywieniowych ma możliwość dodania wielu produktów. Dodatkowo każdy produkt zawiera informację dodatkowe (ProductDetails) w relacji 1-1, a między produktami a dniami żywieniowymi występuje relacja ∞- ∞ przez co powstała tabela pomocnicza (DiaryProduct) mająca na celu umożliwić powstawania relacji.

## Szyfrowanie danych użytkownika

Kolejnym ważnym elementem w każdej aplikacji internetowej, która umożliwia tworzenie kont jest ich zabezpieczenie. W tym celu, w bazie nie przechowuje się haseł, a funkcje hashujące. W momencie tworzenia konta generowana jest funkcja hashująca za pomocą wybranego algorytmu hashującego. Podczas próby zalogowania się na konto, hasło poddane zostaje hashowaniu, z bazy danych pobierana jest funkcja hashująca, która zostaje porównana z nową funkcją wygenerowaną podczas podania hasła.

Poniżej przedstawiony zostaje 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 rysunku 3.9 w celu zaszyfrowania hasła poprzez wbudowaną klasę **RandonNumberGenerator** w frameworku .Net



Rys.3.9 Metoda tworząca sól

1. Dodanie do hasła soli poprzez doklejenie bajtów, a następnie wykorzystane funkcji hashujące SHA512 wbudowanej w .Net na połączonych bajtach przedstawione jest na rys. 3.10.



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

1. Zapis hashu oraz soli w bazie danych

Algorytm walidacji hasła podczas próby logowania prezentuje się następująco

1. Pobranie z bazy danych hashu oraz soli dla podanego użytkownika.
2. Wygenerowanie hashu za pomocą funkcji hashujące za pośrednictwem pobranych parametrów.
3. Porównanie hashu przechowywanego w bazie danych z wygenerowanym w czasie próby logowania przez użytkownika

.

Podsumowanie

Głównym założeniem projektu, było stworzenie przejrzystej i prostej w obsłudze strony internetowej opartej o technologię ASP.NET.Core. Strona miała umożliwiać prowadzenie własnego dziennika żywieniowego poprzez wyliczanie 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 spełniony, nie obyło się bez problemów, największym z nich okazał się responsywny widok oparty o język JavaScript, jQuery oraz AJAX. Sam język JavaScript uznawany jest przez wielu jako najbardziej problematyczny język głównie ze względu na to, że nie jest językiem silnie typowanym co przyczynia się do cięższego zrozumienia kodu i braku wychwycenia błędów z poziomu kompilatora. Kolejnym problemem był podział projektu na moduły, który ostatecznie zakończył się powodzeniem. Dużą wadą ASP.NET.Core jest niekompletna dokumentacja jak i brak licznych książek i materiałów pomocniczych. Wbudowane wstrzykiwanie 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.

Literatura

1. Artykuł dotyczący postrzeganie kalorii. [Online]. Dostępny w internecie: http://oczymlekarze.pl/zdrowy-styl-zycia/dieta/1417-kaloria-kalorii-nierowna
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. Metody wyliczania kalorycznego [Online] Dostępny w internecie http://www.fabrykasily.pl/porady-trenerow/zapotrzebowanie-kaloryczne-wyliczenie
5. Porównanie systemów kontroli wersji GIT i SVN [Online] Dostępny w internecie http://software-engineer-training.com/git-vs-svn-which-is-better/
6. Najlepsze frameworki warstwy wizualnej [Online] Dostępne w internecie https://www.keycdn.com/blog/front-end-frameworks/#1-Bootstrap
7. Dokumentacja ASP.NET.Core [Online] Dostępne w internecie https://docs.microsoft.com/en-us/aspnet/core/
8. Lynn Beighley Head First SQL.

wydawnictwo O’Reily