

Changelog

Jest to pomocniczy rozdział opisujący zmiany w kolejnych wersjach pracy wysyłanej do promotora, żeby ułatwić współpracę z promotorem. Zostanie usunięty przed ostatecznym oddaniem pracy.

v3

- Zmiany:
 - Zrezygnowałem z wykorzystania wzorca TemporalObject, bo nie wyrobię się z zaimplementowaniem go. Encje ProductVersion i RecipeVersion zostały zmergowane ze swoimi rodzicami.
 - Poprawiłem interpunkcję w wyliczeniach.
 - Uzupełniłem opis wykorzystywanych technologii
 - Na prośbę promotora dodałem z powrotem screenshooty rozwiązań konkurencyjnych
 - Zmiana sposobu podawania źródła w podpisach obrazków
- Nowości:
 - implementacja - opis architektury
 - implemetnacja - opis dokumentacji
 - opis zakresu implementacji
 - zakończenie
 - wprowadzenie do testów
 - opis testów jednostkowych, integracyjnych, użyteczności
- Do zrobienia:
 - uzupełnić testy
 - prezentacja aplikacji
 - wykorzystanie bazy usda

v2

- Uwzględniono sugestie z komentarzy:
 - poprawiłem błędy wyszczególnione błędy ortograficzne, interpunkcyjne, stylistyczne
 - W kwestii rozwiązań konkurencyjnych i pracy offline;napisałem maile do supportu wszystkich porównywanych rozwiązań z pytaniem czy jest możliwa praca offline. Wszyscy odpisali, że nie. (oczywiście poza aliantem, który działa tylko offline)
 - dodałem zgodę na przetwarzanie danych osobowych w wymaganiach funkcjonalnych i use casach
 - diagramy przypadków użycia powiększyłem o ok 30% a klas o ok 10%
 - dodałem reguły w latex'ie które powinny wyeliminować większość wdów i sierot
 - dodałem wersje przeglądarek/systemów w wymaganiach niefunkcjonalnych
- Zmiany:
 - krótkie streszczenie na stronie tytułowej
 - przegląd rozwiązań konkurencyjnych
 - uzupełniłem opis problemu we wstępie

- założenia projektowe: uzupełniłem słownik
 - diagramy przypadków użycia przerzuciłem do sekcji z wymaganiami funkcjonalnymi w założeniach projektowych
 - poprawiłem scope wymagań funkcjonalnych
 - uprościłem diagramy przypadków użycia
 - uzupełniłem prototypy interfejsu
 - w dodatku (jdl) zamiast całego opisu domeny zostawiłem tylko definicję mikroserwiów. odniesienie do dodatku będzie w rozdziale implementacja
 - kategorie, ograniczenia, reguły i diagramy klas zostały rozdzielone na sekcje dla każdej poddziedziny w celu zwiększenia czytelności
 - wyrzuciłem ze scope'u mało istotne kategorie; głównie z poddziedzin administracyjnej i wizyt
 - zmiana oznaczeń kategorii/graniczeń/reguł z KAT/XXX na KAT/Y/XX, gdzie Y to numer poddziedziny
 - drobne poprawki stylistyczne w całej pracy
- Nowości:
 - stan wiedzy i technik: przegląd literatury dietetycznej, opis architektury mikroserwisów
 - projekt: opis podstawowej architektury systemu
 - projekt: projekt bazy danych - wprowadzenie teoretyczne
 - implementacja: instalacja oprogramowania
 - podrozdział zakres implementacji, w którym wspomniałem o ochronie danych osobowych
 - Podsumowanie:
 - Chciałbym zamknąć już wszystko od początku pracy do końca opisu projektu (tj. do końca rozdziału 3.)
 - Do zrobienia:
 - opisać: architektura systemu, prezentacja aplikacji, dokumentacja kodu, testy, podsumowanie



Politechnika Wrocławska

Wydział Informatyki i Zarządzania
Kierunek studiów: Informatyka

Praca dyplomowa – inżynierska

PROJEKT I IMPLEMENTACJA SYSTEMU DO ZARZĄDZANIA DIETĄ W OPARCIU O ARCHITEKTURĘ MIKROSERWISÓW

Krzysztof Marczyński

słowa kluczowe:
układanie jadłospisów
architektura mikroserwisów
aplikacja webowa

krótkie streszczenie:

Praca przedstawia proces projektowania i implementacji systemu do zarządzania dietą z wykorzystaniem architektury mikroserwisów. Omówiona zostanie wiedza wynikająca z analizy rozwiązań konkurencyjnych, przeglądu literatury domenowej, a także z porównania stylów architektury oprogramowania. Zasadniczym problemem rozwiązywanym przez opracowywany produkt jest układanie jadłospisów przez dietetyka i przydzielanie ich pacjentowi.

Opiekun pracy dyplomowej	dr inż. Michał Szczepanik
	Tytuł/stopień naukowy/imię i nazwisko	ocena	podpis
Ostateczna ocena za pracę dyplomową			
Przewodniczący Komisji egzaminu dyplomowego
	Tytuł/stopień naukowy/imię i nazwisko	ocena	podpis

*Do celów archiwalnych pracę dyplomową zakwalifikowano do:**

- a) kategorii A (akta wieczyste)
- b) kategorii BE 50 (po 50 latach podlegające ekspertyzie)

* niepotrzebne skreślić

pieczętka wydziałowa

Wrocław, rok 2019

Streszczenie

Celem pracy było opracowanie systemu do zarządzania dietą w architekturze mikroserwisów. Aby osiągnąć ten cel przeprowadzono analizę istniejących rozwiązań konkurencyjnych, przedstawiono niezbędną wiedzę domenową oraz porównano popularne style architektury aplikacji. Na podstawie zgromadzonej wiedzy wyszczególniono niezbędne założenia projektowe, zaprojektowano interfejs oraz zdefiniowano kategorie danych wraz z regułami i ograniczeniami ich dotyczącymi. Następnie przedstawiono opis implementacji powstałej na podstawie opracowanego projektu. W implementacji kluczową rolę odegrały języki Java i TypeScript, platforma deweloperska JHipster oraz stos technologii Netflix OSS dla architektury mikroserwisów. Opracowane rozwiązanie może zostać wykorzystane przez dietetyków w celu przeprowadzania kompleksowej obsługi wizyty pacjenta z położeniem szczególnego nacisku na układanie jadłospisów i udostępnianie go pacjentom.

Abstract

The aim of this work was to develop a diet management system based on microservice architecture. To achieve that goal, an analysis of existing competitive solutions was performed, the necessary domain knowledge was presented, and popular application architecture styles were compared. Based on the accumulated knowledge, the necessary design assumptions were specified, the interface was designed, and categories of data were defined along with the rules and restrictions concerning them. Then a description of the implementation based on the developed project was presented. The key role in the implementation was played by languages Java and TypeScript, the JHipster development platform and Netflix OSS technology stack for a microservices architecture. The created solution can be used by dietitians in order to conduct comprehensive service of the patient's visit with particular emphasis on designing the meal plans and sharing them with patients.

Spis treści

Wstęp	1
Opis problemu	1
Cel pracy	2
Zakres pracy	3
1. Stan wiedzy i techniki w zakresie tematyki pracy	4
1.1. Przegląd istniejących rozwiązań konkurencyjnych	4
1.2. Przegląd literatury dietetycznej	10
1.3. Architektura mikroserwisów	12
2. Założenia projektowe	14
2.1. Uwagi wstępne	14
2.2. Słownik pojęć domenowych	14
2.3. Sformułowanie problemu	15
2.4. Pozycjonowanie produktu	16
2.5. Dekompozycja problemu w oparciu o poddziedziny	17
2.6. Podsumowanie użytkowników systemu	17
2.7. Wymagania funkcjonalne	19
2.8. Wymagania niefunkcjonalne	29
3. Projekt	30
3.1. Prototyp interfejsu	30
3.1.1. Poddziedzina administracyjna	31
3.1.2. Poddziedzina produkty	34
3.1.3. Poddziedzina przepisy	37
3.1.4. Poddziedzina jadłospisy	39
3.1.5. Poddziedzina wizyty	41
3.2. Opis podstawowej architektury systemu	47
3.3. Projekt bazy danych	48
3.3.1. Ogólne cechy dziedziny	49
3.3.2. Poddziedzina administracyjna	49
3.3.3. Poddziedzina produkty	52
3.3.4. Poddziedzina przepisy	59
3.3.5. Poddziedzina jadłospisy	66
3.3.6. Poddziedzina wizyty	71
4. Implementacja	78
4.1. Wykorzystywane środowiska i narzędzia programistyczne	78
4.2. Zakres implementacji	80
4.3. Architektura systemu	81

4.3.1. Architektura mikroserwisów	81
4.3.2. Architektura backendu	83
4.3.3. Architektura frontendu	84
4.4. Dokumentacja kodu	84
4.5. Instalacja oprogramowania	86
4.5.1. Wymagania wstępne	86
4.5.2. Instalacja	86
4.6. Prezentacja aplikacji	88
5. Testy	89
5.1. Wprowadzenie	89
5.2. Testy jednostkowe	90
5.3. Testy integracyjne	91
5.4. Testy użyteczności	93
Zakończenie	94
Bibliografia	95
Spis rysunków	98
Spis tabel	100
Spis kodów źródłowych	101
Dodatek A. Konfiguracja mikroserwisów w języku JDL	102

Wstęp

Opis problemu

Przed przejściem do analizy problemu, warto zdefiniować co można rozumieć pod pojęciem diety. Według definicji z Encyklopedii Powszechnej PWN, dieta to "system odżywiania z ustaleniem jakości i ilości pokarmów, dostosowany do potrzeb organizmu"[44]. Opierając się na tej definicji, można opisać system do zarządzania dietą jako system, który będzie ułatwiał dietetykowi układanie jadłospisów dostosowanych do potrzeb żywieniowych organizmu pacjenta, pozwalał na zarządzanie ułożonymi jadłospisami i udostępnianie ich pacjentowi w formie umożliwiającej w jak najprostszy sposób zastosowanie przez pacjenta przygotowanej dla niego diety, a także umożliwiał kontrolę rezultatów stosowania jadłospisu przez pacjenta.

Obserwując trendy występujące we współczesnym społeczeństwie można zauważyc, że zdrowy styl życia stał się modny, a czasem nawet utożsamiany ze statusem społecznym. W związku z tą tendencją coraz więcej osób regularnie uprawia sport, rezygnuje z używek, a także dba o dietę. Analizując przedstawione na rysunku 1 dane dotyczące wyszukiwania hasła "dietetyk" w latach 2004-2019 poprzez narzędzie Google Trends[22] można zauważyc, że popularność wyszukiwania tego hasła w latach 2016-2019 jest ponad 5 krotnie większa niż w roku 2004.

● dietetyk



Polska. Od 01.01.2004 do 12.09.2019. Wyszukiwarka Google.

Rysunek 1: Zainteresowanie hasłem "dietetyk" w ujęciu czasowym (źródło: [22])

Zwiększone zainteresowanie usługami dietetycznymi powoduje zwiększone zapotrzebowanie na wysokiej jakości, nowoczesne narzędzia wspomagające pracę dietetyka. W chwili pisania niniejszej pracy w Polsce popularność zyskało jedynie kilka programów oferujących kompleksowe funkcjonalności potrzebne w codziennej praktyce dietetyka, można więc sądzić, że rynek aplikacji tego typu nie został jeszcze nasycony.

Biorąc pod uwagę powyższe spostrzeżenia, warto w tym miejscu podkreślić, że Światowa Organizacja Zdrowia (ang. World Health Organization - WHO) określiła otyłość (i bardziej ogólnie choroby dietozależne) jako jeden z głównych problemów zdrowia publicznego[70]. Fakt ten podkreśla jak duże brzemię odpowiedzialności spoczywa na ramionach dietetyków, a co za tym idzie jak istotne jest dostarczenie specjalistom narzędzi ułatwiających niesienie pomocy pacjentom.

Cel pracy

Celem pracy jest projekt i budowa platformy do zarządzania dietą w oparciu o architekturę mikroserwisów. Opracowywana platforma będzie obejmowała cały proces zarządzania dietą, czyli przede wszystkim: zebranie przez dietetyka wywiadu żywieniowego od pacjenta, ułożenie przez dietetyka jadłospisu, udostępnienie jadłospisu pacjentowi, ułatwienie pacjentowi stosowania diety poprzez generowanie listy zakupów oraz kontrolę rezultatów diety.

Zakres pracy

Praca w swoim zakresie zawiera opracowanie projektu systemu, w ramach którego, między innymi, omówieni zostaną użytkownicy systemu wraz z ich potrzebami, przeprowadzona zostanie dekompozycja problemu w oparciu o podziedziny, przygotowane zostaną prototypy interfejsu i diagramy UML takie jak diagram przypadków użycia i diagram klas. Omówiona zostanie również podstawowa architektura systemu.

W części praktycznej pracy analizie poddane będą wybrane narzędzia i technologie wykorzystane do implementacji projektu, omówiony zostanie zakres implementacji, a także przedstawiony będzie rezultat tejże implementacji w formie prezentacji działania aplikacji. Przedstawiona zostanie również dokumentacja kodu i testy oraz pokrótko omówiony będzie sposób instalacji systemu z uwzględnieniem wymagań wstępnych.

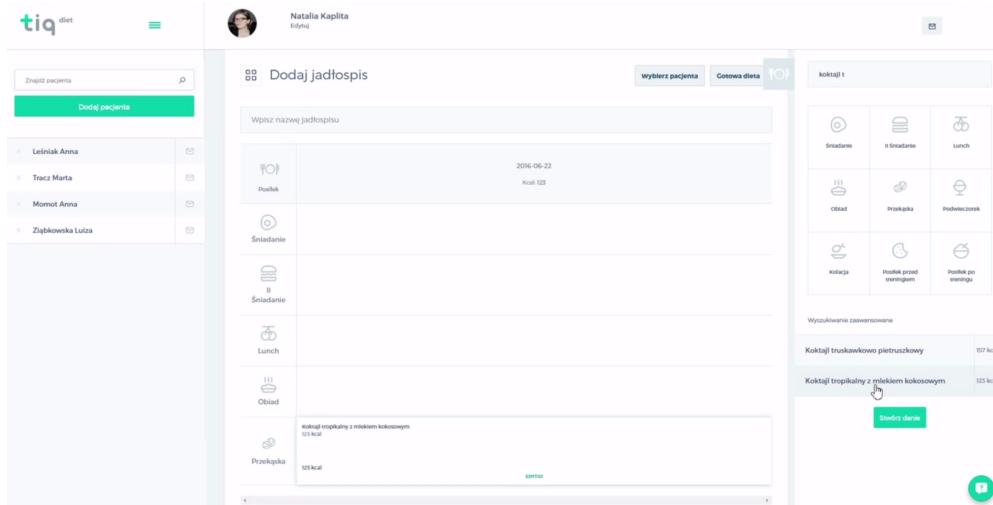
1. Stan wiedzy i techniki w zakresie tematyki pracy

1.1. Przegląd istniejących rozwiązań konkurencyjnych

Na rynku Polskim funkcjonuje zaledwie kilka narzędzi w kompleksowy sposób wspomagających pracę dietetyka. W niniejszej sekcji zostaną omówione systemy cieszące się największą popularnością oraz przedstawione zostanie zbiorcze porównanie ich najważniejszych cech. W przypadku większości z porównywanych programów warunki korzystania z usługi pozwalają na rejestrację w systemie jedynie wykwalifikowanym dietetykom. Z tego względu wszystkie analizowane dane zostały zebrane z publicznie dostępnych źródeł, takich jak strony odpowiednich programów, blogi internetowe oraz filmy promocyjne na platformie YouTube[24]. Dodatkowo, żeby rozwiązać wątpliwości czy dane rozwiązanie konkurencyjne umożliwia korzystanie z kluczowych funkcjonalności bez dostępu do internetu, przeprowadzono stosowną korespondencję z obsługą klienta poszczególnych programów.

- **TiqDiet**

TiqDiet[57] jest wygodnym w użyciu programem, który pozwala dietetykom w intuicyjny sposób projektować jadłospisy i udostępniać je pacjentom. W celu uproszczenia pracy dietetyka, udostępnianych jest wiele szablonów jadłospisów, które łatwo można dostosować do indywidualnych potrzeb pacjenta. Pacjent może odbierać ułożoną dietę poprzez responsywną stronę internetową, aplikację mobilną oraz za pomocą intelligentnego zegarka. Aplikacje mobilne pozwalają ponadto na automatyczne przypominanie pacjentowi m.in. o konieczności zażycia suplementu oraz o konieczności regularnego picia wody. Komunikacja pomiędzy pacjentem, a dietetykiem może odbywać się w czasie rzeczywistym za pomocą zintegrowanego komunikatora. Ponadto dietetyk ma możliwość obserwowania postępów pacjentów w stosowaniu diety, a w razie potrzeby może zalecać wizytę u lekarza czy też zażycie dodatkowych suplementów. Na rysunku 1.1 przedstawiono przykładowy widok edycji planu dnia w aplikacji TiqDiet



Rysunek 1.1: TiqDiet (źródło: [61])

- Kcalmar PRO

Kcalmar[26] jest systemem, którego głównym założeniem jest maksymalne skrócenie czasu potrzebnego na ułożenie programu żywieniowego dopasowanego do potrzeb pacjenta. Zapewnia zaawansowany system podpowiadający ułatwiający projektowanie zbilansowanej diety z wyraźnym oznaczeniem alergenów czy zduplikowanych potraw. Jadłospisy mogą być automatycznie skalowane z automatycznym przeliczeniem miar domowych. Co ciekawe system pozwala również na wyszukiwanie dietetyków w wybranych miastach i filtrowanie ich według typów diet i jednostek chorobowych, w których się specjalizują. Na rysunku 1.2 pokazano widok wygenerowanej listy zakupów dla jadłospisu zaprojektowanego w aplikacji Kcalmar.

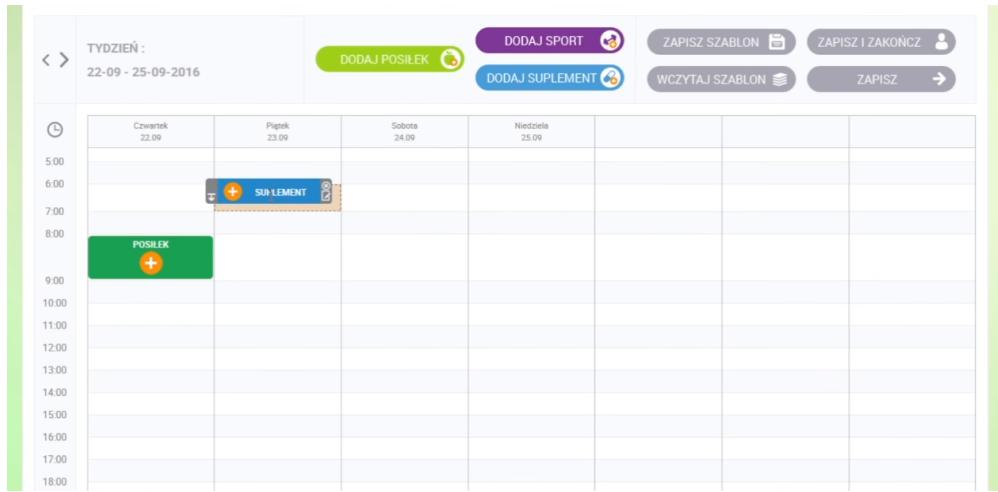
Pacjenci > Michał Kowal > Dieta dla pacjenta Michał Kowal		Pamiętaj, że możesz podzielić się swoimi uwagami klikając w 'Twoja sugestia' z lewej strony ekranu.		
		16-01-2017/22-01-2017		
	GRAFIK	LISTA ZAKUPÓW	KOMENTARZ	OPIS I WŁAŚCIWOŚCI
	73 Produktów	Sortuj Po gramach		Grupowanie
Owoce i Warzywa	Nabiał	Orzechy i ziarna		
Pomarańcza	3400,0g 17 x sztuka	Jogurt naturalny	Orzechy włoskie	
Banan	840,0g 7 x sztuka	Serek wiejski (naturalny)	Tahini	
Pomidór	720,0g 4 x sztuka	Jaja kurze (całe)	Nasiona słonecznika	
Papryka czerwona (świeża)	560,0g 4 x sztuka	Twarzog chudy	Pestki dyni	
Mango	560,0g 2 x sztuka			
Ziemniaki (późne)	560,0g 8 x sztuka	Ryby i owoce morza	Mięso i wyroby miejsne	
Jabłko	450,0g 3 x sztuka	Tunczyk w kawałkach (w sosie własnym)	Filet z piersi kurczaka (bez skóry)	
Pomidory z puszki (krojone)	400,0g 4 x porcja	Łosoś (wędzony)	Połędwiczka z indyka (surowa)	
Cieciorka	375,0g 25 x łyżka	Dorsz (filety bez skóry)	Mięso z udeku kurczaka (bez skóry)	
Brokuł	275,0g 0,6 x sztuka			

Rysunek 1.2: Kcalmar PRO (źródło: [33])

- Dietetyk Pro

Program Dietetyk Pro[3] na tle konkurencji wyróżnia się tym, że poza główną funkcjonalnością układania jadłospisu, abonenci mogą również korzystać ze szkoleń eksperckich i literatury dietetycznej dostępnej w ramach platformy. Dodatkowo dietetycy

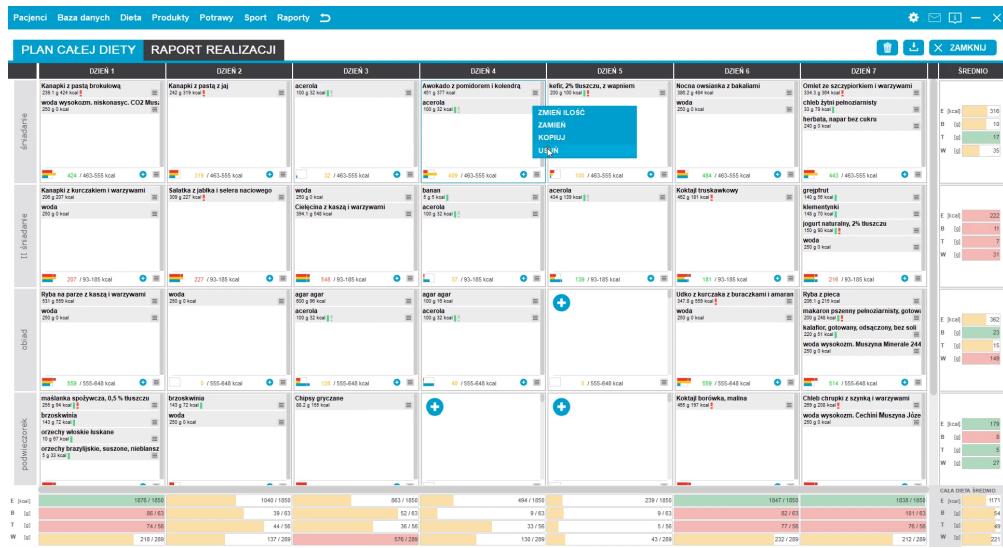
po wykupieniu subskrypcji mogą skorzystać ze zdalnej pomocy z obsługi programu. Ciekawym udogodnieniem jest możliwość szerokiej konfiguracji ekranu startowego, np. poprzez dodanie kalkulatora wartości odżywcznych czy też wyświetlanie listy zaplanowanych wizyt. Spośród porównywanych programów Dietetyk Pro posiada największe bazy produktów i przepisów wyprzedzając konkurencję niemal dwukrotnie. Na rysunku 1.3 przedstawiono widok edycji 4-dniowego jadłospisu w aplikacji Dietetyk Pro



Rysunek 1.3: Dietetyk Pro (źródło: [7])

- **Aliant**

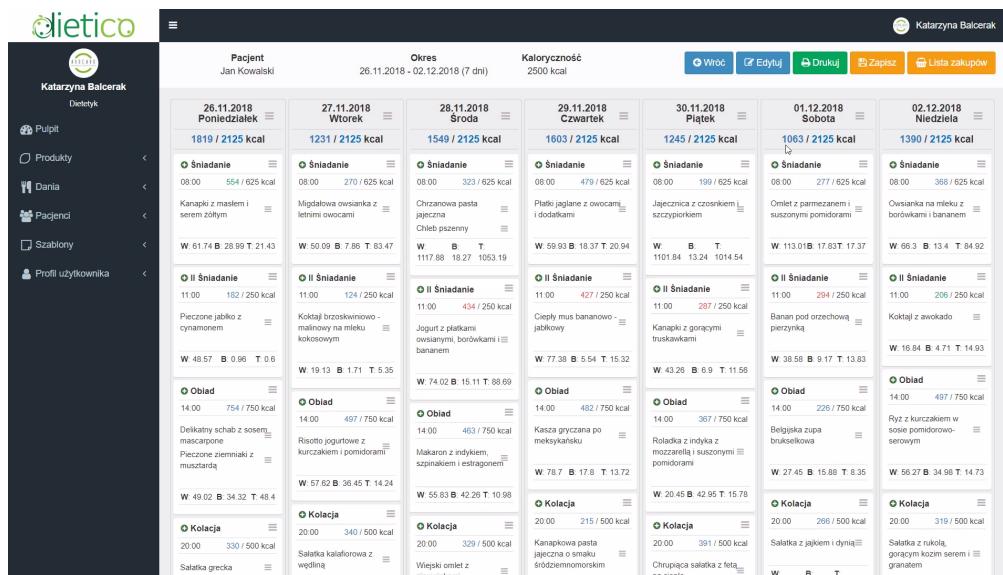
Do grupy klientów docelowych programu Aliant[2] należą zarówno dietetycy jak również trenerzy personalni. Tak jak inne porównywane aplikacje, główną funkcjonalnością programu Aliant jest układanie jadłospisów, jednakże w przeciwieństwie do konkurencji, aplikacja jest dostępna tylko jako aplikacja na platformę Windows. Nie ma możliwości dostępu do systemu przez stronę internetową, jednak możliwe jest automatyczne dodawanie z internetu zewnętrznych baz produktów po zaakceptowaniu ich licencji wykorzystania. Brakuje również zintegrowanego narzędzia do komunikacji z pacjentami, a udostępnianie ułożonego jadłospisu musi odbywać się w całości poza systemem. Na rysunku 1.4 zaprezentowano widok edycji tygodniowego jadłospisu w programie Aliant. Warto zauważyć, że dla każdego dnia w sposób graficzny przedstawiono czy zostały spełnione zaplanowane dzienne normy ilości podstawowych składników odżywcznych.



Rysunek 1.4: Aliant (źródło: [1])

- Dietico

Program Dietico[72] szczerzy się stale powiększaną bazą przepisów i produktów. Pozwala na układanie jadłospisu za pomocą wygodnego interfejsu, a przejrzysty system podpowiedzi pozwala szybko wykryć powtarzające się dania oraz produkty, na które pacjent jest uczulony. Twórcy programu podkreślają, że ich program wyróżnia możliwość uwzględnienia w układanej diecie posiadanego przez pacjenta wyposażenia kuchennego i sezonowych produktów spożywczych. Na rysunku 1.5 przedstawiono widok edycji tygodniowego jadłospisu wraz z graficznymi oznaczeniami produktów, na które pacjent jest uczulony.



Rysunek 1.5: Dietico (źródło: [8])

- Vitme

Program Vitme[60] umożliwia prowadzenie kart pacjentów, projektowanie jadłospisów oraz generowanie wydruków w formacie PDF. Program w porównaniu z konkurencją jest oferowany w bardzo korzystnej cenie oraz posiada bogatą bazę produktów, jednak stosunkowo niewielką bazę przepisów. Do wad produktu można zaliczyć przestarzały i mało przejrzysty interfejs, który może zniechęcić niektórych potencjalnych klientów. Na rysunku 1.6 pokazano przykładowy widok kalendarza wizyt w aplikacji Vitme.

The screenshot shows the Vitme application's interface for creating a new diet. At the top, there is a table with columns: Nazwa diety (Name of diet), Typ (Type), Kcal (kcal), W (g), B (g), T (g), Chol (mg), Blon (g), Sód (mg), and Edytuj (Edit). A row for "Przykładowa Dieta VITME" is shown with values: 3442, 545.52, 143.15, 97.36, 422.78, 234.14, 3125.82, and a green "Edytuj" button. Below this, a green banner says "Dieta została zainportowana". The main area is titled "Nowa dieta" (New Diet) with a "Nowa dieta" button. It shows a dropdown menu set to "Przykładowa Dieta VITME" and "28 dni". Below this, there is a section titled "Wybierz dzień tygodnia lub cały tydzień" (Select a day of the week or the whole week) with a grid of days (1-7). Buttons for "zobacz tydz. 1", "zobacz tydz. 2", "zobacz tydz. 3", and "zobacz tydz. 4" are also present. The main content area is titled "Układanie posiłków" (Arranging meals) and lists meal plans for different times of the day:

- Śniadanie (08:00):** Banan, jogurt naturalny, orzechy (429.00 kcal, 47.60g W, 12.65g B, 21.51g T, 23.53g Chol, 75.45g Blon, 248.30g Sód)
- Śniadanie (10:00):** Czerwony koktajl z buraka (142.00 kcal, 21.71g W, 8.78g B, 3.20g T, 19.80g Chol, 74.85g Blon, 269.38g Sód)
- Obiad (14:00):** Indyk duszony z warzywami i ryżem (544.00 kcal, 97.46g W, 11.34g B, 13.25g T, 0.63g Chol, 12.65g Blon, 89.36g Sód)
- Podwieczorek (17:00):** Filet z kurczaka z warzywami, kasza gryczana (792.00 kcal, 120.52g W, 38.71g B, 20.98g T, 10.76g Chol, 18.66g Blon, 392.22g Sód)

Each meal plan row has a delete ("X") and add ("+") button on the right.

Rysunek 1.6: Vitme (źródło: [67])

W tabeli 1.1 przedstawiono porównanie najważniejszych cech funkcjonalnych, a w tabeli 1.2 cech niefunkcjonalnych 6 istniejących na rynku rozwiązań konkurencyjnych[51]. Warto zwrócić uwagę, że funkcjonalności takie jak możliwość wykorzystania gotowych szablonów diet, wysyłanie diety do pacjenta, przeprowadzanie wywiadu żywieniowego czy automatyczne generowanie listy zakupów nie występują w niektórych spośród analizowanych systemów. Na tej podstawie można wysunąć hipotezę, że te funkcjonalności - mimo iż istotne - nie są kluczowe w systemie wspomagającym pracę dietetyka. Natomiast możliwość układania jadłospisów z wykorzystaniem własnych produktów i przepisów, zapisywanie ich do plików oraz przypisywanie do stosownych kart pacjenta są oczekiwane w tego typu aplikacjach.

Tabela 1.1: Rozwiązania konkurencyjne - cechy funkcjonalne (źródło: opr. wł.)

	TiqDiet	Kcalmar Pro	Dietetyk Pro	Aliant	Dietico	Vitme
Układanie jadłospisów	TAK	TAK	TAK	TAK	TAK	TAK
Gotowe szablony diet	TAK	TAK	TAK	TAK	NIE	NIE
Zapis diety do pliku	TAK	TAK	TAK	TAK	TAK	TAK
Wysyłanie diet do pacjenta	TAK	TAK	TAK	TAK	NIE	TAK
Komunikacja z pacjentem	TAK	TAK	TAK	NIE	NIE	TAK
Karta pacjenta	TAK	TAK	TAK	TAK	TAK	TAK
Wywiad żywieniowy	TAK	TAK	TAK	NIE	TAK	TAK
Lista zakupów	TAK	TAK	TAK	TAK	TAK	NIE
Dodawanie własnych produktów	TAK	TAK	TAK	TAK	TAK	TAK
Dodawanie własnych przepisów	TAK	TAK	TAK	TAK	TAK	TAK

Tabela 1.2: Rozwiązań konkurencyjne - cechy niefunkcjonalne (źródło: opr. wł.)

	TiqDiet	Kcalmar Pro	Dietetyk Pro	Aliant	Dietico	Vitme
Liczba produktów w bazie	1000	1400	6000	3500	900	5000
Liczba gotowych przepisów	200	800	2800	1700	1900	400
Praca offline	NIE	NIE	NIE	TAK	NIE	NIE
Praca online	TAK	TAK	TAK	NIE	TAK	TAK
Aplikacja mobilna dla dietetyka	TAK	TAK	TAK	NIE	NIE	NIE
Aplikacja mobilna dla pacjenta	TAK	TAK	NIE	NIE	NIE	TAK
Dostęp dla pacjenta przez przeglądarkę internetową	TAK	TAK	TAK	NIE	NIE	TAK
Darmowy okres testowy	14dni	14dni	7dni	bezterminowo	14dni	14dni
Cena w abonamencie rocznym	199	1188	246	699	546	219

1.2. Przegląd literatury dietetycznej

W rozdziale 1.1 dokonano przeglądu rozwiązań konkurencyjnych. Na podstawie dokonanej analizy możliwe będzie zdefiniowanie głównych wymagań funkcjonalnych projektowanego systemu, jednakże konieczne jest odwołanie się do literatury dziedzinowej, żeby potwierdzić zasadność przyjętych założeń istotnych z punktu widzenia dietetyki.

Pierwszym rozważanym pojęciem jest podstawowa przemiana materii (PPM). Jest to poziom zapotrzebowania energetycznego organizmu znajdującego się w stanie spoczynku (czyli minimalny poziom zapotrzebowania energetycznego) wyznaczany na podstawie wieku i masy ciała osoby. Aby obliczyć wartość energetyczną posiłku należy wyznaczyć ekwiwalent metaboliczny (MET) podstawowych wartości odżywczych, tj. białek, tłuszczy i węglowodanów[5].

Kolejnym uwzględnianym współczynnikiem jest współczynnik poziomu aktywności fizycznej (ang. Physical Activity Level - PAL). Organizacja Narodów Zjednoczonych do spraw Wyżywienia i Rolnictwa (ang. Food and Agriculture Organization of the United Nations - FAO) definiuje 5 poziomów aktywności fizycznej[14]:

- brak aktywności fizycznej (wartość współczynnika 1,2 - 1,39),
- niska aktywność fizyczna (wartość współczynnika 1,4 - 1,69),
- umiarkowana aktywność fizyczna (wartość współczynnika 1,7 - 1,99),
- wysoka aktywność fizyczna (wartość współczynnika 2 - 2,4),
- bardzo wysoka aktywność fizyczna (wartość współczynnika > 2,4).

Iloczyn PPM i PAL określa stopień całkowitej przemiany materii (CPM)[29]. Przy czym dla prawidłowo zaplanowanej diety, dzienna energia powinna być dostarczana w proporcjach:

- ok 10% z białek,
- ok 60% z węglowodanów,
- ok 30% z tłuszczy.

Wymienione wyżej składniki tworzą najważniejszą grupę składników, tzw grupę składników energetycznych, ale należy zwrócić uwagę, że do prawidłowego funkcjonowania organizmu definiuje się ok 40 innych niezbędnych składników należących do grup składników budulcowych (głównie jod, wapń, lipidy, fosfor, żelazo i siarka) i regulujących (głównie witaminy, błonnik oraz mikro- i makroelementy), które również powinny być uwzględnione podczas układania zbilansowanej diety[5].

Podczas układania jadłospisu uwzględnia się podstawowe typy diet[18]:

- podstawowa,
- kleikowa,
- papkowata,
- bogatopotasowa,
- niskosodowa,
- niskocholesterolowa,
- bezglutenowa,
- bogatoresztkowa,
- łatwostrawna,
- ubogotłuszcza,
- bogatobiałkowa,
- ubogobiałkowa,
- bogatoenergetyczna,
- ubogoenergetyczna.

Jak zauważono wyżej, podstawowym kryterium potrzebnym do skomponowania odpowiednio zbilansowanej diety jest odpowiednie dobranie wartości odżywczej produktów spożywczych. Wartości te mogą być uzyskane z tabel składu i wartości odżywczej żywności. Na rynku polskim tabele takie są odpłatnie udostępniane przez polski Instytut Żywności i Żywienia (IŻŻ)[34], jednakże licencja wspomnianego opracowania nie zezwala na wykorzystanie danych zawartych w zestawieniu bez wykupienia odpowiedniego abonamentu[27].

Podobne rozwiązanie w języku angielskim oferuje Departament Rolnictwa Stanów Zjednoczonych (ang. United States Department of Agriculture - USDA) udostępniając całkowicie za darmo do dowolnego użytku narodową bazę danych wartości odżywczych dla standardowych odwołań (ang. National Nutrient Database for Standard Reference)[65] potocznie nazywana "bazą USDA". Dane są udostępniane w formie pliku bazy Microsoft Access. Baza zawiera:

- ponad 7 tysięcy produktów spożywczych,
- ponad 600 tysięcy wartości odżywczych,
- ok 100 definicji wartości odżywczych,
- ponad 14 tysięcy miar domowych,
- ok 25 kategorii produktów spożywczych.

Na koniec należałoby rozważyć wskaźniki pozwalające ocenić wpływ diety. Podstawowe wyznaczane wartości to wskaźnik masy ciała (ang. Body Mass Index - BMI), stosunek obwodu talii do obwodu bioder (ang. Waist to Hip Ratio - WHR) oraz ilość tkanki tłuszczowej w organizmie. W celu wyznaczenia ilości tkanki tłuszczowej w organizmie przeprowadza się pomiar fałdów skórno-tłuszczowych lub stosuje się bioelektryczną metodę impedancji (ang. bioelectrical impedance analysis - BIA). Metoda BIA polega na przepuszczeniu przez organizm ładunku elektrycznego o natężeniu poniżej 1mA. Metoda ta jest nieinwazyjna i bezpieczne jest jej stosowanie w każdym stanie zdrowia pacjenta. Pozwala nie tylko na wyznaczenie ilości tkanki tłuszczowej, ale także ilości wody w organizmie, wapnia w kościach i masy mięśniowej[5].

1.3. Architektura mikroserwisów

Projektowanie złożonych systemów informatycznych umożliwiających bezproblemowe jednoczesne korzystanie przez miliony użytkowników jest zadaniem niebanalnym. W klasycznym podejściu, implementowano systemy w architekturze monolitycznej. Aplikacja napisana w takiej architekturze jest samowystarczalna w kontekście jej zachowania. Może komunikować się z zewnętrznymi usługami lub źródłami danych w celu wykonania operacji, ale logika biznesowa potrzebna do wykonania każdej operacji jest w całości zawarta w obrębie aplikacji. W przypadku wystąpienia potrzeby skalowania horyzontalnego takiej aplikacji, konieczne jest powielanie całej aplikacji na każdym z serwerów[37].

Architektura mikroserwisów, zgodnie z tym co sugeruje nazwa, skupia się na budowaniu aplikacji będącej zbiorem niewielkich, luźno powiązanych serwisów komunikujących się ze sobą na przykład za pomocą protokołu HTTP czy AMQP. Serwisy implementowane i wdrażane są niezależnie od siebie[62]. Efektem projektowania niezależnych serwisów jest skalowanie tylko serwisów, które tego wymagają, co pozwala na optymalne wykorzystanie zasobów[55].

Architektura monolityczna ma wiele zalet[52], spośród których do najważniejszych należą:

- prostota implementacji,
- możliwość łatwego przeprowadzania radykalnych zmian w programie,
- prostota testowania,
- prostota wdrażania aplikacji na środowisko produkcyjne,
- prostota skalowania aplikacji.

Martin Fowler podkreśla, że w przypadku wielu aplikacji architektura monolityczna jest jak najbardziej wystarczająca, a w przypadku gdy system jest wystarczająco złożony, żeby użycie mikroserwisów przyniosło realny zysk, zwykle lepiej jest zacząć od monolitu, a następnie przeprowadzić migrację do architektury mikroserwisów poprzez wydzielenie modułów w obrębie monolitu i późniejsze przekształcanie ich w niezależne serwisy[16].

W przypadku aplikacji monolitycznej łatwo jest doprowadzić do sytuacji w której poszczególne moduły są ze sobą ściśle powiązane, co zwykle ma bardzo negatywny wpływ na wydajność aplikacji, utrudnia wprowadzanie zmian w kodzie i prowadzi do występowania trudnych do wykrycia błędów w implementacji. Sytuację, w której w aplikacji powstaje dużo przypadkowych powiązań i zależności Vaughn Vernon określił mianem "Wielkiej Kuli Błota"[66].

Do głównych zalet zastosowania mikroserwisów należy stosowanie luźnego powiązania serwisów, co poniekąd wymusza, żeby zależności pomiędzy serwisami były bardziej przemyślane i lepiej zaprojektowane. Z pomocą we właściwym zaprojektowaniu serwisów i zależności między nimi przychodzi strategiczne wzorce DDD. Jednym z takich wzorców jest dekompozycja w oparciu o poddziedziny[12]. Dziedzina systemu jest dzielona na poddziedziny poprzez zdefiniowanie przestrzeni problemów biznesowych w obrębie względnie niezależnych obszarów specjalizacji. W przypadku architektury mikroserwisów można wyznaczyć poszczególne serwisy poprzez zdefiniowanie poddziedzin systemu i zaprojektowanie serwisu dla każdej z nich[52].

2. Założenia projektowe

2.1. Uwagi wstępne

W niniejszym rozdziale opisano wizję systemu, który będzie wspomagał zarządzanie dietą.

Zalogowani dietetycy będą mogli zarządzać produktami, ich wartościami odżywczymi oraz miarami domowymi. Korzystając ze zdefiniowanych produktów dietetycy będą mogli tworzyć przepisy, a następnie, w ramach jadłospisu, dodawać do planów posiłków przepisy i pojedyncze produkty.

Dietetycy będą mogli również zarządzać pacjentami i ich wizytami. W ramach wizyty dietetyk będzie mógł przeprowadzić wywiad żywieniowy, zebrać pomiary ciała pacjenta i przydzielić pacjentowi jadłospis.

2.2. Słownik pojęć domenowych

Na podstawie rozważań z rozdziału 1 sporządzono następującą listę definicji domenowych istotną z punktu widzenia projektu:

- BIA - metoda impedancji bioelektrycznej wykorzystywana do analizy składu ciała
- BMI - wskaźnik masy ciała
- CPM - całkowita przemiana materii
- Dieta - sposób odżywiania uwzględniający potrzeby organizmu, ze zdefiniowaną ilością i jakością pożywienia
- Dietetyk - specjalista w dziedzinie dietetyki
- Jadłospis - plan posiłków zdefiniowany na określoną liczbę dni z uwzględnieniem określonych wymagań
- Karta pacjenta - karta przedstawiająca przebieg współpracy dietetyka z pacjentem
- MET - ekwiwalent metaboliczny
- Miara domowa - definicja pospolitej miary, takiej jak np. łyżeczka w gramach
- Pacjent - klient dietetyka
- PAL - współczynnik aktywności fizycznej
- Podstawowe wartości odżywcze - energia, białko, tłuszcze, węglowodany
- Pomiary ciała - pomiary ciała pacjenta przeprowadzane przez dietetyka
- Posiłek - posiłek jest przydzielany do jadłospisu; zawiera produkty i przepisy
- PPM - podstawowa przemiana materii

- Produkt
 - produkt spożywczy, dla którego specyfikowane są wartości odżywcze i miary domowe
- Przepis
 - opis składników i kroków przygotowania dania
 - semantyczny podział przepisu, np. sernik może mieć sekcje związane z przygotowaniem ciasta, nadzienia i polewy
- Sekcja przepisu
 - opis składników i kroków przygotowania dania
 - semantyczny podział przepisu, np. sernik może mieć sekcje związane z przygotowaniem ciasta, nadzienia i polewy
- Baza USDA
 - baza produktów udostępniana przez Departament Rolnictwa Stanów Zjednoczonych
 - ilość elementu takiego jak np. węglowodanów albo białka w 100g produktu
- Wartość odżywcza
 - konkretna wizyta pacjenta
 - wywiad przeprowadzany z pacjentem uwzględniający jego nawyki żywieniowe, nietolerancje, choroby, przyjmowane leki, itp.
- Wizyta
 - wspomaganie procesu układania odpowiednio zbilansowanego jadłospisu i kontrola rezultatów stosowania ułożonego jadłospisu
- Wywiad żywieniowy
 - wspomaganie procesu układania odpowiednio zbilansowanego jadłospisu i kontrola rezultatów stosowania ułożonego jadłospisu
- Zarządzanie dietą
 - wspomaganie procesu układania odpowiednio zbilansowanego jadłospisu i kontrola rezultatów stosowania ułożonego jadłospisu

2.3. Sformułowanie problemu

W tabeli 2.1 przedstawiono sformułowanie rozważanego w pracy problemu wraz z jego wpływem i propozycją pomyślnego rozwiązania.

Tabela 2.1: Sformułowanie problemu (źródło: opr. wł.)

Problem	Problem z ręcznym układaniem jadłospisu i kontrolą rezultatów stosowania jadłospisu przez pacjenta.
Dotyczy	Dietetyków
Wpływ problemu	<ul style="list-style-type: none"> • Dietetyk poświęca dużo czasu na wyszukiwanie informacji o każdym produkcie, którego potrzebuje wykorzystać w układanym jadłospisie. • Dietetyk poświęca dużo czasu na obliczanie wartości odżywcznych w każdym przepisie. • Dietetyk poświęca dużo czasu na obliczanie wartości odżywcznych w każdym jadłospisie. • Dietetyk ma problem z przeliczeniem miar domowych produktów na gramy. • Dietetyk musi sam obliczać i kontrolować zmiany BMI pacjenta.
Pomyślne rozwiązanie	<ul style="list-style-type: none"> • Będzie zwalniało dietetyka z konieczności obliczania wartości odżywcznych dla przepisów i jadłospisów. • Będzie ułatwiało dietetykowi przekazywanie ułożonego jadłospisu pacjentów. • Będzie ułatwiało dietetykowi kontrolowanie rezultatów stosowania diety.

2.4. Pozycjonowanie produktu

W tabeli 2.2 przedstawiono pozycjonowanie opracowywanego produktu względem rynku produktów wspomagających pracę dietetyka.

Tabela 2.2: Pozycjonowanie produktu (źródło: opr. wł.)

Dla	Dietetyka
Który	Chce łatwiej zarządzać dietą
Nazwa produktu	Aplikacja webowa wspomagająca pracę dietetyka "Dietify"
Który	Skraca czas potrzebny na ułożenie i zarządzanie jadłospisami
Inaczej niż	Kalkulator kalorii
Nasz produkt	Skupia się na układaniu i udostępnianiu jadłospisów oraz kontroli rezultatów ich stosowania

2.5. Dekompozycja problemu w oparciu o poddziedziny

Na podstawie wywiadu z dietetykiem, analizy rozwiązań konkurencyjnych oraz opierając się na opisany w rozdziale 1.3 wzorcu dekompozycji problemu w oparciu o poddziedziny dla omawianej aplikacji wspomagania zarządzania dietą można wyszczególnić następujące poddziedziny:

- **poddziedzina administracyjna** - służąca jako brama aplikacji, pozwalająca na zarządzanie użytkownikami i administrowanie aplikacją,
- **poddziedzina produkty** - skupiająca się na zarządzaniu produktami spożywczymi, ich wartościami odżywczymi i miarami domowymi,
- **poddziedzina przepisy** - pozwalająca na zarządzanie przepisami, w tym przypisywanie do przepisów produktów,
- **poddziedzina jadłospisy** - pozwalająca na zarządzanie jadłospisami, w tym przypisywanie do jadłospisów produktów i przepisów,
- **poddziedzina wizyty** - skupiająca się na całościowym zarządzaniu wizytami pacjenta w obrębie karty pacjenta, a w szczególności przypisywaniem do wizyty jadłospisów, przeprowadzaniem wywiadu żywieniowego oraz zbieraniem pomiarów ciała pacjenta.

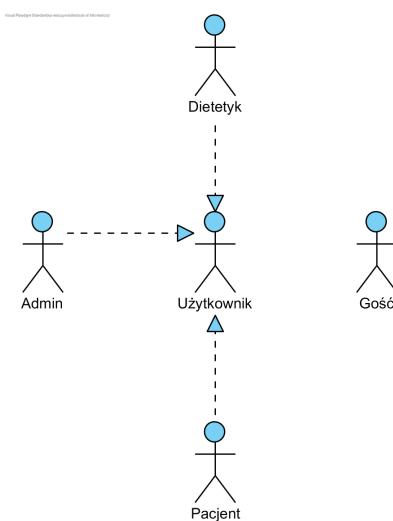
2.6. Podsumowanie użytkowników systemu

W tabeli 2.3 przedstawiono podsumowanie użytkowników projektowanego systemu, ich krótki opis oraz ich podstawowe odpowiedzialności związane z korzystaniem z systemu.

Tabela 2.3: Użytkownicy (źródło: opr. wł.)

Nazwa	Opis	Odpowiedzialności
Gość	Niezalogowany użytkownik	<ul style="list-style-type: none"> • Zakłada konto użytkownika. • Wyświetla stronę główną.
Pacjent	Klient dietetyka	<ul style="list-style-type: none"> • Otrzymuje ułożony jadłospis.
Dietetyk	Specjalista w dziedzinie dietetyki	<ul style="list-style-type: none"> • Używa założonego konta. • Wprowadza, edytuje i usuwa produkty, przepisy i jadłospisy. • Zarządza wizytami pacjenta.
Administrator	Osoba zarządzająca działaniem aplikacji	<ul style="list-style-type: none"> • Przydzielanie i odbieranie użytkownikom uprawnień. • Zarządzanie definicjami wartości odżywczych, typami diet, typami posiłków, typami dań i wyposażeniem kuchennym.

Na rysunku 2.1 przedstawiono relacje pomiędzy użytkownikami korzystając z notacji diagramu przypadków użycia języka UML.



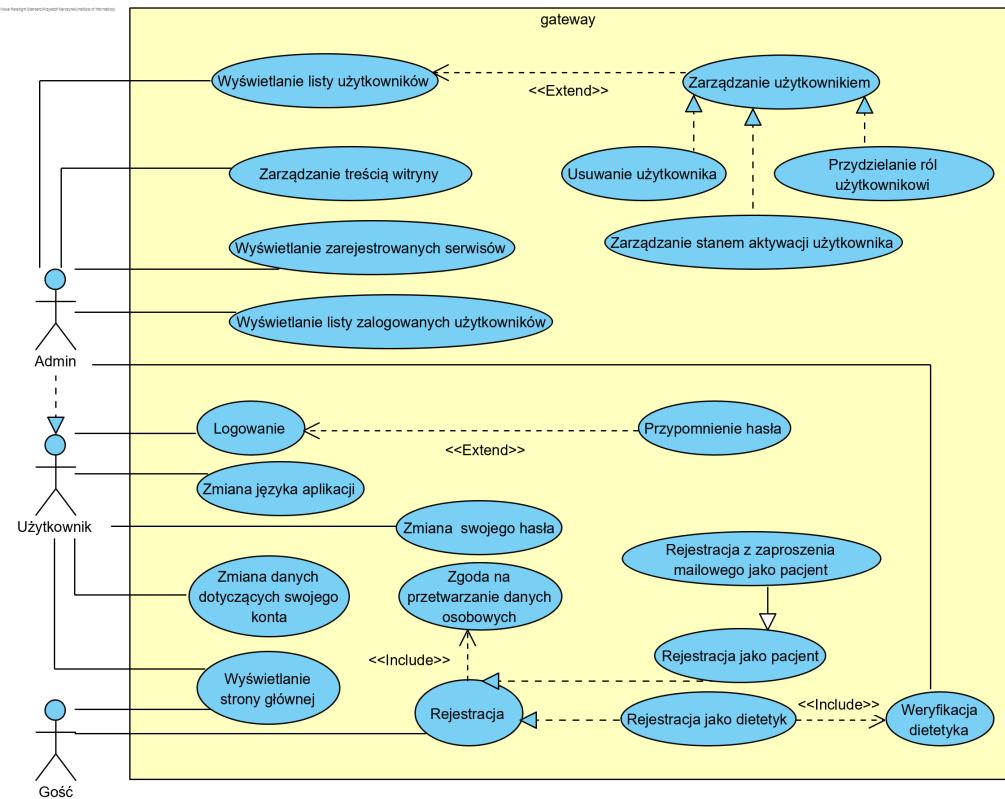
Rysunek 2.1: Diagram przypadków użycia - użytkownicy (źródło: opr. wł.)

2.7. Wymagania funkcjonalne

W tabelach 2.4, 2.5, 2.6, 2.7 i 2.8 przedstawiono wymagania funkcjonalne dla poddziedzin systemu w postaci zestawienia potrzeb użytkowników systemu z cechami związanymi z realizacją danej potrzeby. Następnie wymagania dla konkretnych poddziedzin sformalizowano w postaci diagramów przypadków użycia języka UML odpowiednio na rysunkach 2.2, 2.3, 2.4, 2.5 i 2.6.

Tabela 2.4: Wymagania funkcjonalne - poddziedzina administracyjna (źródło: opr. wł.)

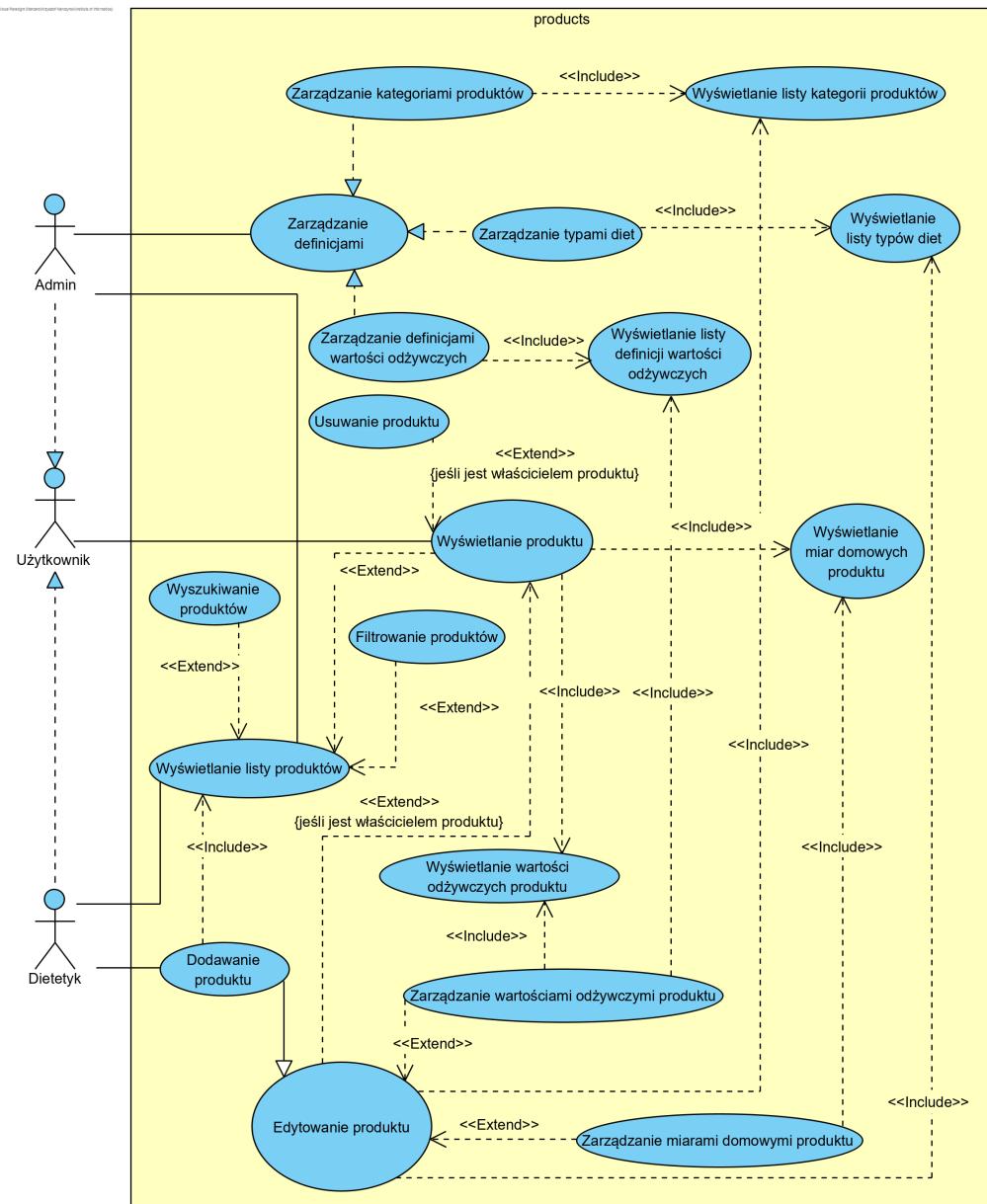
Potrzeby	Cechy
Administrator potrzebuje widzieć listę użytkowników	<ul style="list-style-type: none">Przydzielanie i odbieranie użytkownikom uprawnień.
Użytkownik potrzebuje korzystać ze swojego konta	<ul style="list-style-type: none">Logowanie do systemu.Przypomnienie hasła.Zarządzanie swoimi danymi osobowymi.
Użytkownik chce przeglądać witrynę w swoim języku	<ul style="list-style-type: none">Obsługa witryny w wielu językach.
Gość potrzebuje korzystać z systemu	<ul style="list-style-type: none">Zakładanie konta użytkownika.Wyrażenie zgody na przetwarzanie swoich danych osobowych.



Rysunek 2.2: Diagram przypadków użycia - poddziedzina administracyjna (źródło: opr. wł.)

Tabela 2.5: Wymagania funkcjonalne - poddziedzina produkty (źródło: opr. wł.)

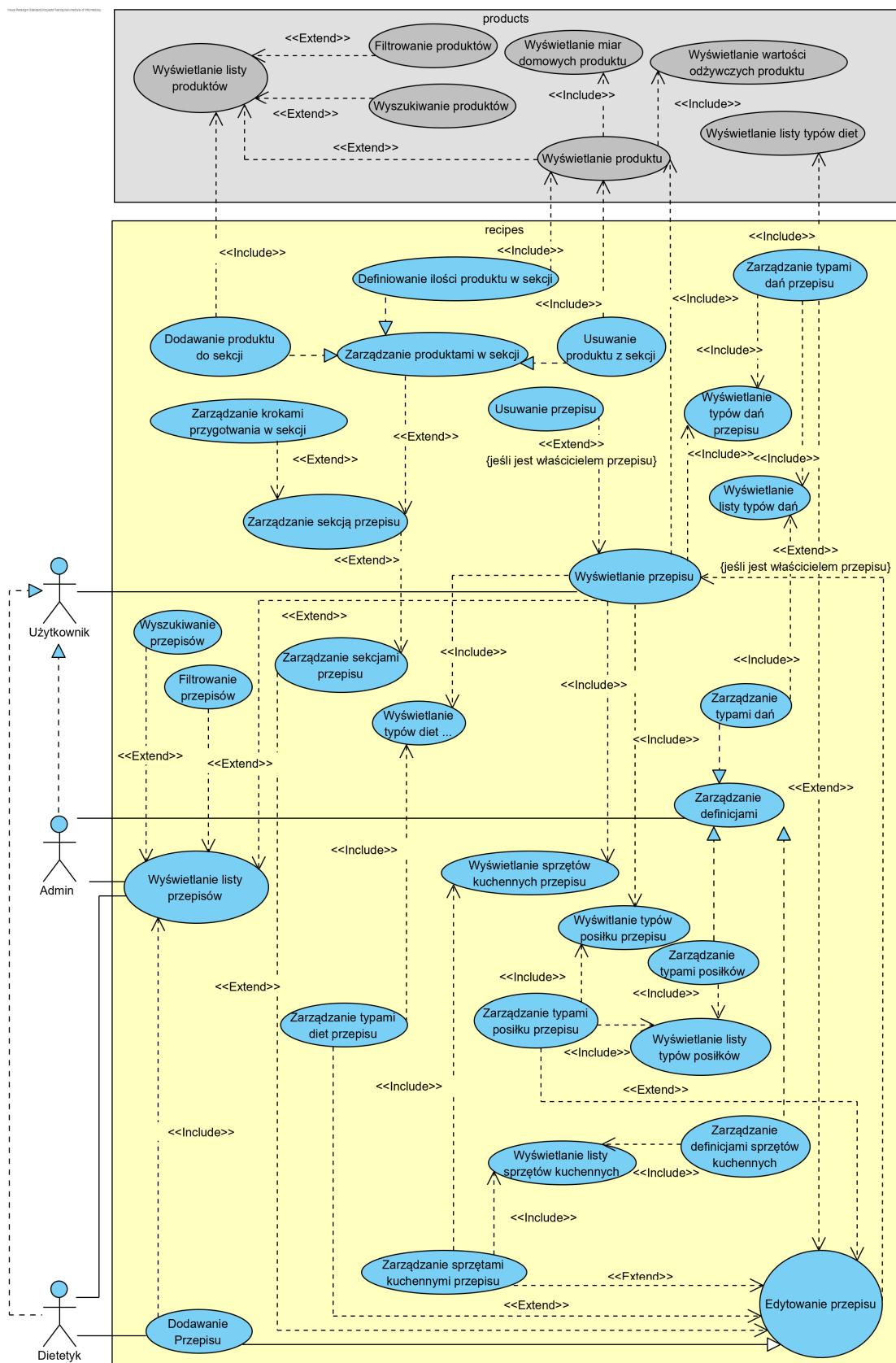
Potrzeby	Cechy
Administrator potrzebuje zarządzać definicjami niezbędnymi w produktach	<ul style="list-style-type: none"> • Zarządzanie definicjami wartości odżywcznych. • Zarządzanie kategoriami produktów. • Zarządzanie rodzajami diet.
Dietetyk potrzebuje widzieć listę produktów	<ul style="list-style-type: none"> • Wyszukiwanie produktów. • Filtrowanie produktów. • Dodawanie nowych produktów.
Dietetyk potrzebuje zarządzać szczegółami produktu	<ul style="list-style-type: none"> • Edytowanie i usuwanie produktów. • Definiowanie wartości odżywcznych dla produktu. • Definiowanie miar domowych dla produktu. • Przypisywanie produktu do kategorii i podkategorii. • Definiowanie do jakich typów diet produkt nadaje się, a do jakich nie.



Rysunek 2.3: Diagram przypadków użycia - poddziedzina produkty (źródło: opr. wł.)

Tabela 2.6: Wymagania funkcjonalne - poddziedzina przepisy (źródło: opr. wł.)

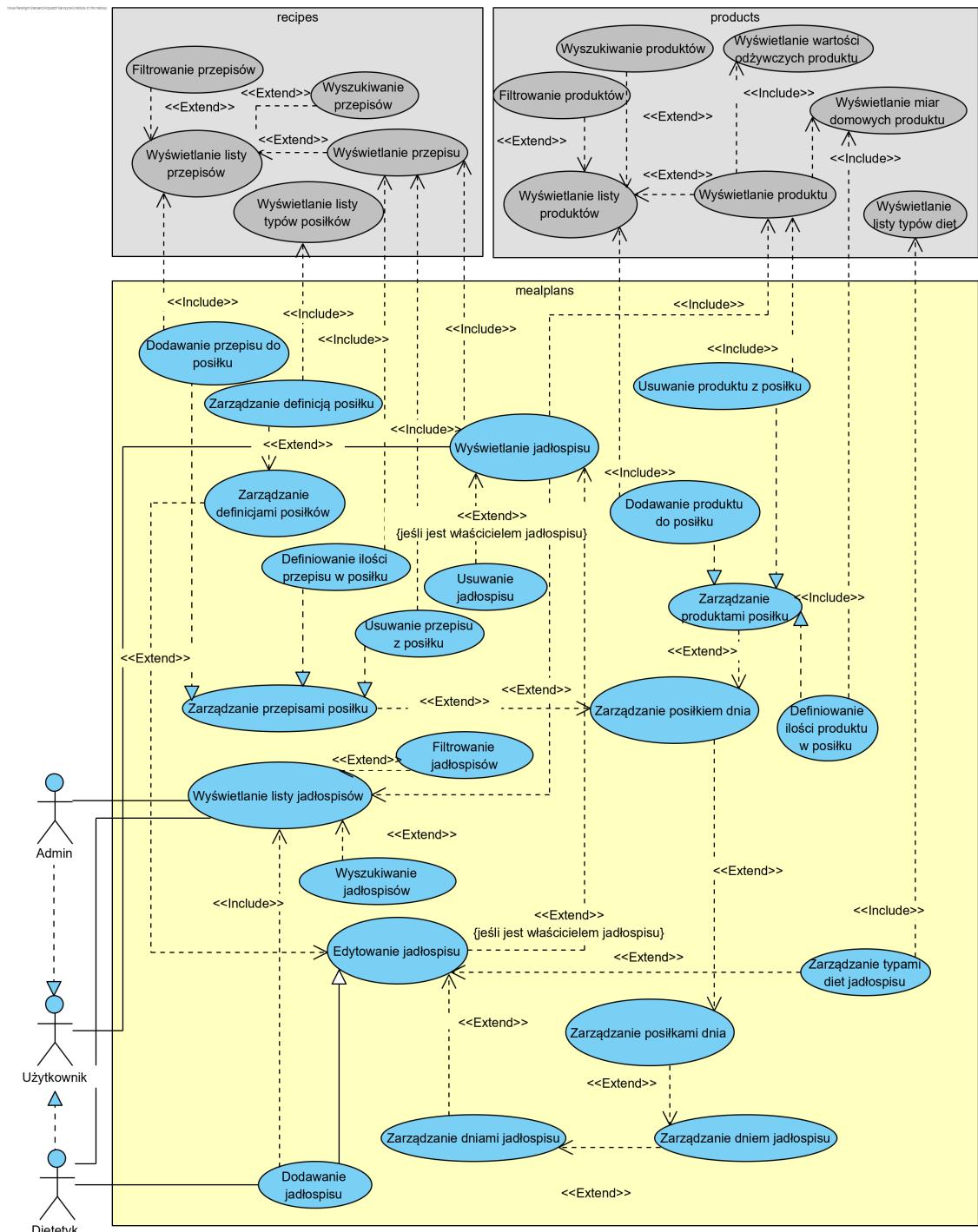
Potrzeby	Cechy
Administrator potrzebuje zarządzać definicjami niezbędnymi w przepisach	<ul style="list-style-type: none"> • Zarządzanie typami posiłków. • Zarządzanie typami dań. • Zarządzanie definicjami wyposażenia kuchennego.
Dietetyk potrzebuje widzieć listę przepisów	<ul style="list-style-type: none"> • Wyszukiwanie przepisów. • Filtrowanie przepisów. • Dodawanie nowych przepisów.
Dietetyk potrzebuje zarządzać szczegółami przepisu	<ul style="list-style-type: none"> • Edytowanie i usuwanie przepisów. • Dodawanie wielu sekcji do przepisu. • Dodawanie do każdej sekcji listy składników. • Dodawanie do każdej sekcji sposobu przygotowania. • Dodawanie zdjęcia dania do przepisu. • Definiowanie czasu przygotowania posiłku.



Rysunek 2.4: Diagram przypadków użycia - podzdiedzina przepisy (źródło: opr. wł.)

Tabela 2.7: Wymagania funkcjonalne - poddziedzina jadłospisy (źródło: opr. wł.)

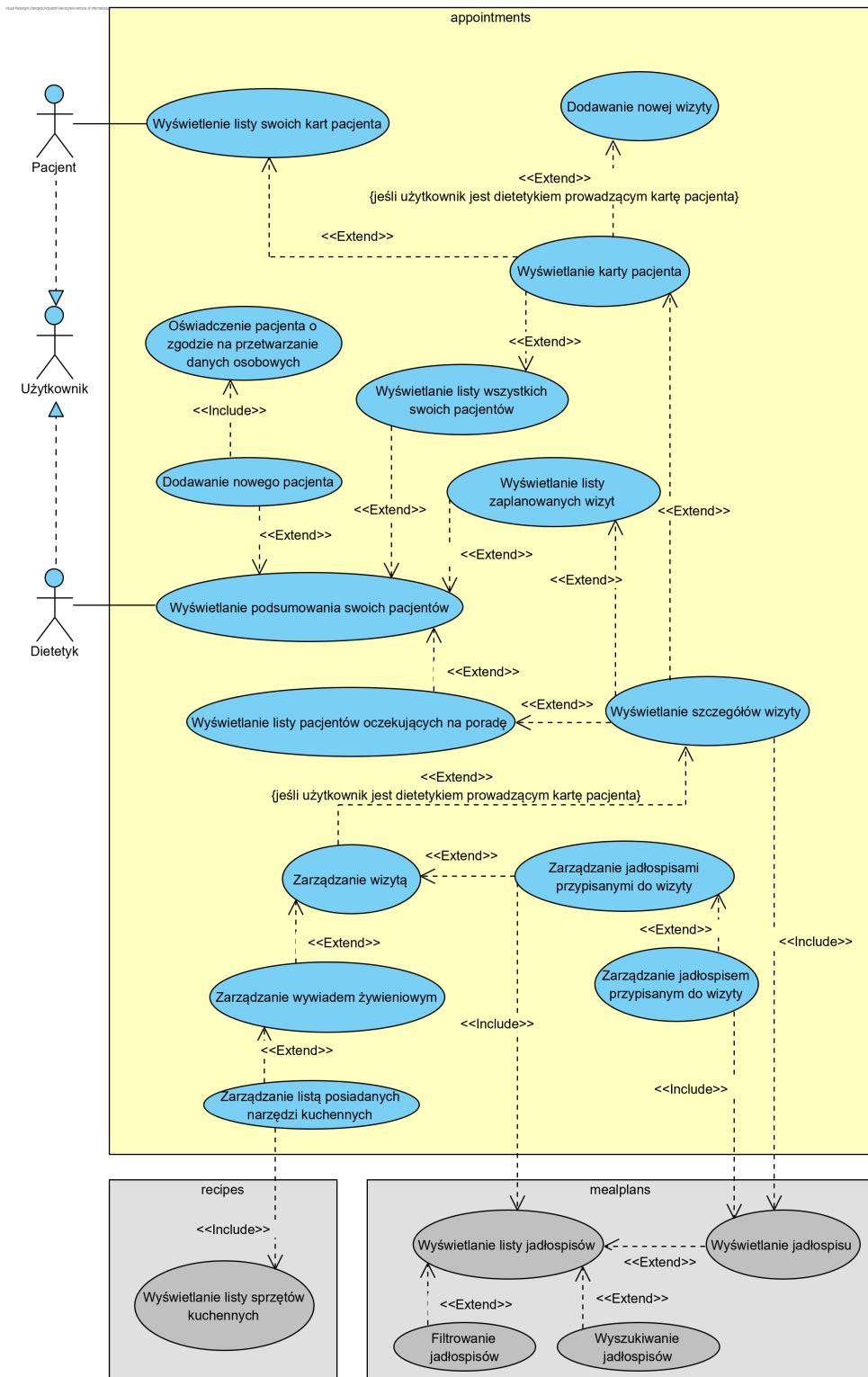
Potrzeby	Cechy
Dietetyk potrzebuje widzieć listę jadłospisów	<ul style="list-style-type: none"> • Wyszukiwanie jadłospisów. • Filtrowanie jadłospisów. • Dodawanie nowych jadłospisów.
Dietetyk potrzebuje zarządzać szczegółami jadłospisu	<ul style="list-style-type: none"> • Dodawanie, edytowanie i usuwanie jadłospisów. • Definiowanie liczby dni na które będzie układany jadłospis. • Definiowanie liczby posiłków dziennie. • Definiowanie planowanego czasu każdego z posiłków. • Definiowanie procentowego udziału podstawowych wartości odżywczych w każdym posiłku. • Definiowanie posiłków w jadłospisie. • Dodawanie produktów i przepisów do posiłków.



Rysunek 2.5: Diagram przypadków użycia - podzdiedzina jadłospisy (źródło: opr. wł.)

Tabela 2.8: Wymagania funkcjonalne - poddziedzina wizyty (źródło: opr. wł.)

Potrzeby	Cechy
Dietetyk potrzebuje wyświetlać listę swoich pacjentów	<ul style="list-style-type: none"> • Wyszukiwanie pacjentów. • Wyświetlanie listy znalezionych pacjentów. • Wyświetlanie listy umówionych wizyt. • Wyświetlanie listy oczekujących porad. • Dodawanie nowych pacjentów.
Dietetyk potrzebuje zarządzać kartą pacjenta	<ul style="list-style-type: none"> • Wyświetlanie i edytowanie podstawowych informacji pacjenta. • Wyświetlanie listy wizyt pacjenta. • Wyświetlanie listy oczekujących porad pacjenta. • Dodawanie nowej wizyty pacjenta.
Dietetyk potrzebuje wyświetlać szczegóły wizyty pacjenta	<ul style="list-style-type: none"> • Wyświetlanie i edytowanie szczegółów wizyty pacjenta. • Zarządzanie pomiarami ciała pacjenta przypisanymi do wizyty. • Zarządzanie wywiadem żywieniowym przypisanym do wizyty. • Zarządzanie jadłospisem przydzielonym do wizyty.
Pacjent potrzebuje otrzymywać dietę	<ul style="list-style-type: none"> • Udostępnianie pacjentowi jadłospisu.
Pacjent chce mieć wgląd w swoją kartę	<ul style="list-style-type: none"> • Logowanie do konta utworzonego w serwisie. • Dodawanie kart pacjenta do swojego konta po udostępnieniu ich przez dietetyka.



Rysunek 2.6: Diagram przypadków użycia - podziedzina wizyty (źródło: opr. wł.)

2.8. Wymagania niefunkcjonalne

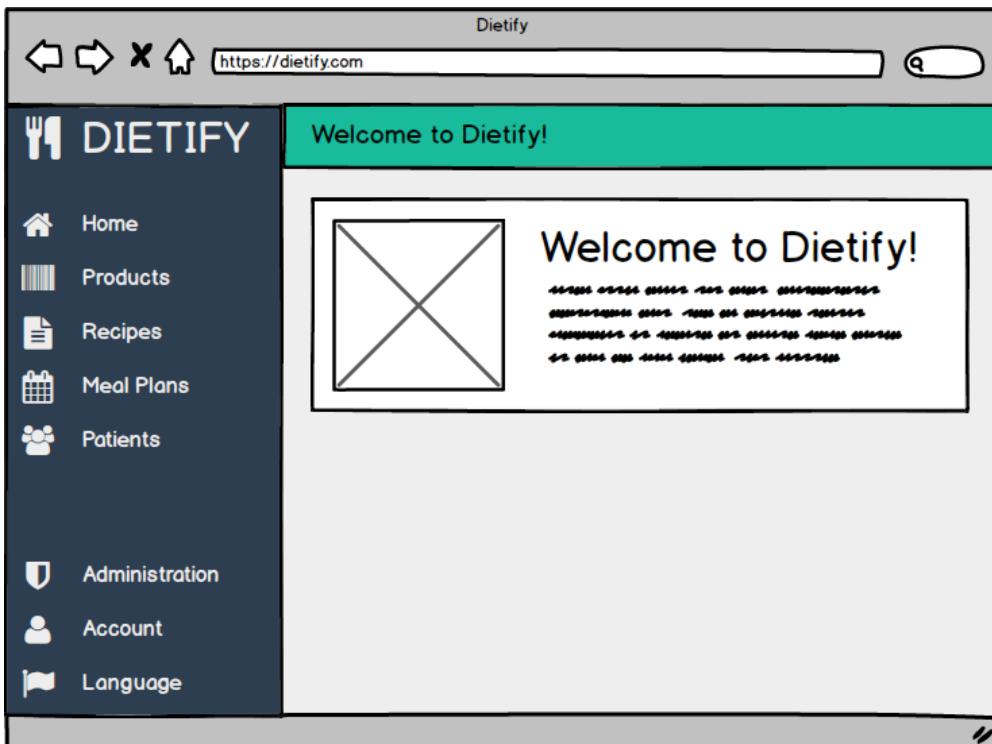
- System działa poprawnie w przeglądarkach Google Chrome 76, Mozilla Firefox 69 i Opera 63.
- System działa na urządzenia mobilnych korzystających z systemu Android 9 i iOS 12.
- System jest dostępny w polskiej i angielskiej wersji językowej.
- System ma czytelny i minimalistyczny interfejs.
- Aplikacja webowa jest w pełni responsywna i wygodna do używania na ekranach od 5 do 30 cali.
- Aplikacja jest oparta na architekturze mikroserwisów.

3. Projekt

3.1. Prototyp interfejsu

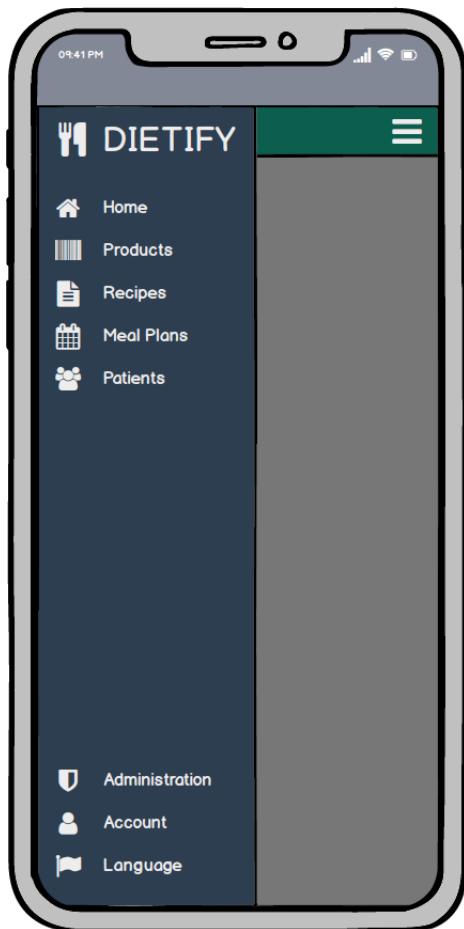
Opierając się zdobytej wiedzy dotyczącej potrzeb użytkowników aplikacji oraz na wymaganiach funkcjonalnych służących do zaspokojenia tych potrzeb opracowano prototypy kluczowych widoków interfejsu użytkownika.

Zanim przedstawione zostaną prototypy dla poddziedzin warto wpierw omówić szablon widoku, który będzie stosowany w obrębie całej witryny. Na rysunku 3.1 przedstawiona została strona startowa witryny. U góry strony znajduje się belka zawierająca hasło przewodnie danego widoku. Po lewej stronie umieszczony został panel nawigacyjny, u góry którego widnieje logo i nazwa systemu. Na rzecz projektu roboczo przyjęto jako logo rysunek sztućców, a "Dietify" jako nazwę projektowanego systemu. Poniżej są odnośniki do strony startowej oraz do głównych widoków związanych z każdą z poddziedzin. Na dole panelu nawigacyjnego znajdują się łącza do zmiany języka, ustaleń użytkownika oraz, jeśli zalogowany użytkownik ma uprawnienia administratora, do widoku administracyjnego. W głównej części widoku widoczna jest właściwa treść bieżącej strony. W przypadku omawianej strony startowej jest to wiadomość powitalna oraz obrazek związany z dietetyką.



Rysunek 3.1: Prototyp interfejsu - strona startowa (źródło: opr. wł.)

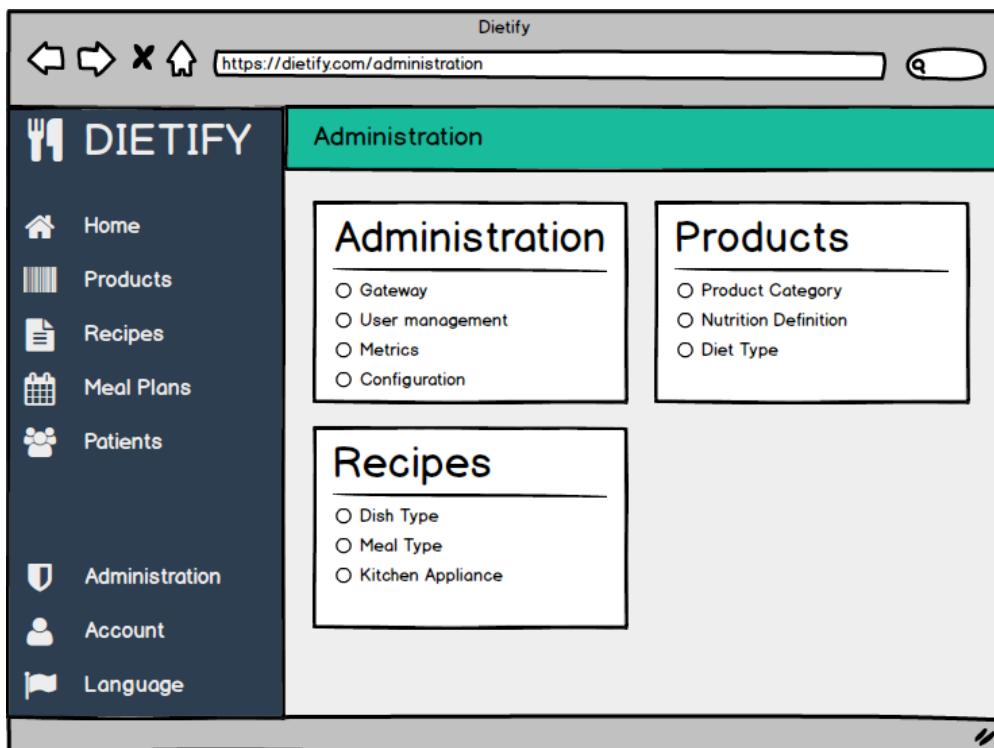
Podobnie szablon strony będzie wyglądał na urządzeniu mobilnym, co przedstawiono na rysunku 3.2, przy czym ze względu na niewielki rozmiar ekranu urządzeń mobilnych, panel nawigacyjny jest zwijany, a w celu jego wyświetlenia koniecznie jest naciśnięcie przycisku znajdującego się po prawej stronie belki, która jest u góry strony.



Rysunek 3.2: Prototyp interfejsu - układ strony na urządzeniu mobilnym (źródło: opr. wł.)

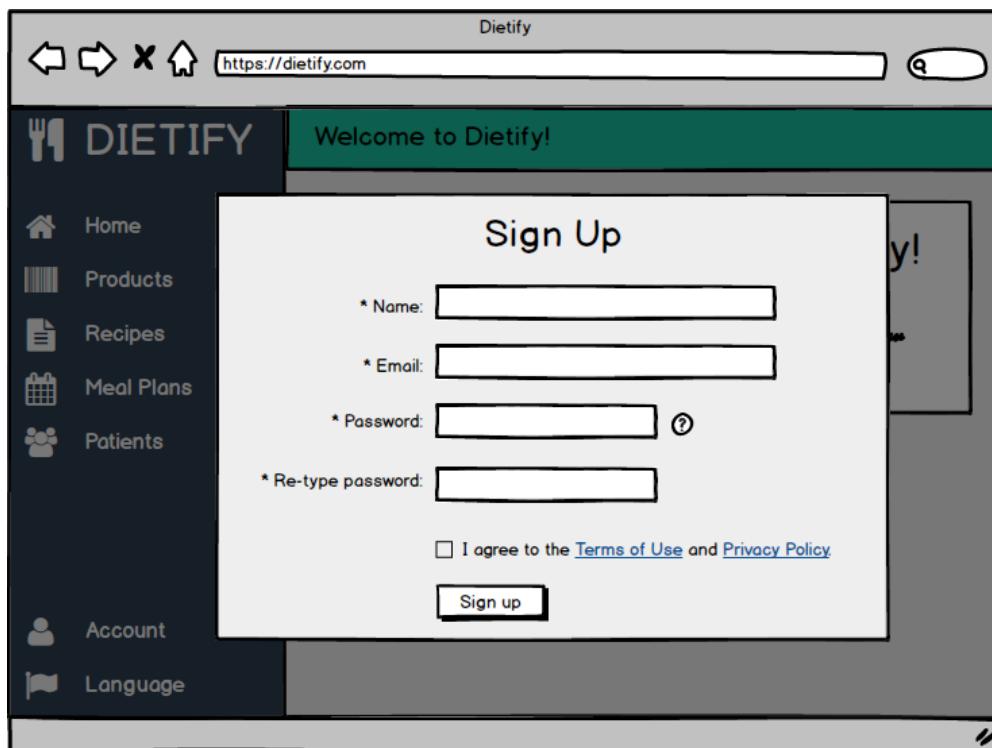
3.1.1. Poddziedzina administracyjna

Na rysunku 3.3 przedstawiono podstawowy widok administratora zawierający łącza do treści zarządzanych przez administratora w poszczególnych poddziedzinach.

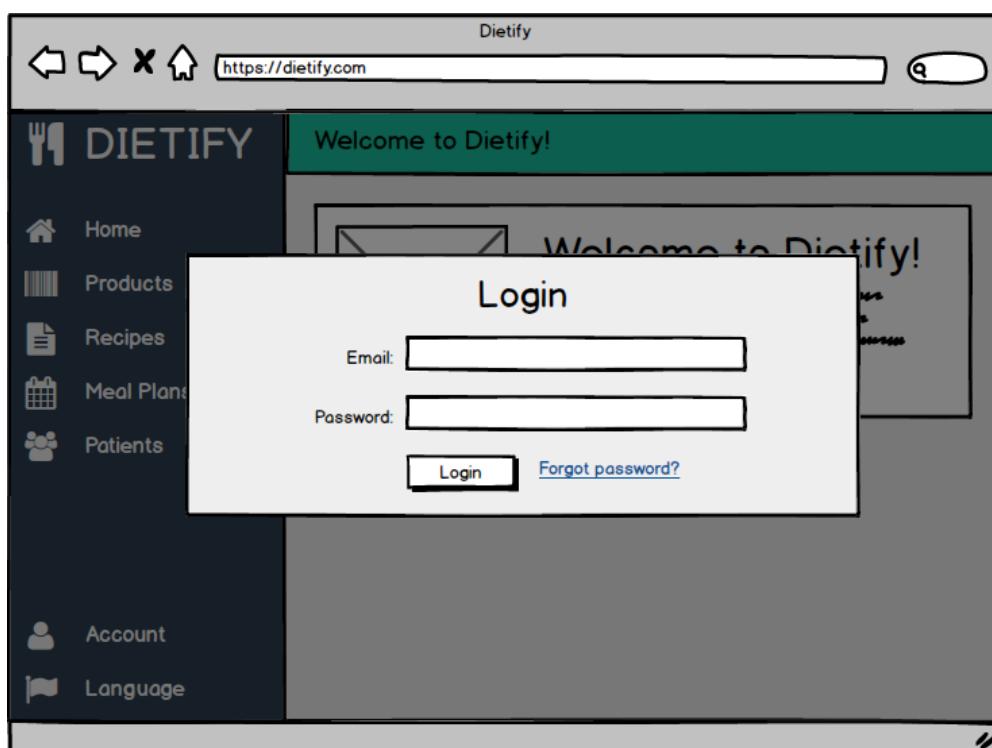


Rysunek 3.3: Prototyp interfejsu - widok administratora (źródło: opr. wł.)

Poddziedzina administracyjna jest związana przede wszystkim z zarządzaniem użytkownikami, przy czym należy również wziąć pod uwagę rejestrację (rysunek 3.4) i logowanie (rysunek 3.5) użytkowników do systemu. Warto zauważyć, że podczas rejestracji użytkownik wyraża zgodę na przetwarzanie swoich danych osobowych poprzez akceptację polityki prywatności.



Rysunek 3.4: Prototyp interfejsu - rejestrowanie do systemu (źródło: opr. wł.)



Rysunek 3.5: Prototyp interfejsu - logowanie do systemu (źródło: opr. wł.)

Widok zarządzania użytkownikami (rysunek 3.6) jest dostępny tylko dla administratora Z poziomu tego widoku możliwe jest zarządzanie uprawnieniami użytkowników, zmiana statusu ich kont oraz tworzenie nowych użytkowników.

The screenshot shows a web browser window for 'Dietify' at the URL <https://dietify.com/administration/user-management>. The left sidebar has a dark blue background with white icons and text: DIETIFY logo, Home, Products, Recipes, Meal Plans, Patients, Administration, Account, Language. The main area has a light gray background. A green header bar says 'Users'. Below it, a white table lists three users:

ID	Login	Email	Status	Profiles
1	admin	admin@localhost	activated	ROLE_ADMIN, ROLE_USER
2	dietitian	dietitian@localhost	activated	ROLE_DIETITIAN, ROLE_USER
3	patient	patient@localhost	activated	ROLE_PATIENT, ROLE_USER

A 'Create new user' button is located in the top right corner of the user list area.

Rysunek 3.6: Prototyp interfejsu - zarządzanie użytkownikami (źródło: opr. wł.)

3.1.2. Poddziedzina produkty

Na rysunku 3.7 przedstawiono prototyp widoku listy produktów. Z poziomu tego widoku możliwe jest wyszukiwanie i filtrowanie produktów, wyświetlanie szczegółów produktu oraz tworzenie nowego produktu. Do tego widoku prowadzi odnośnik "Products" z panelu nawigacyjnego.

The screenshot shows a web-based application interface for managing products. On the left, a dark sidebar menu includes links for Home, Products, Recipes, Meal Plans, Patients, Account, and Language. The main content area has a green header bar with the word 'Products'. Below this is a sub-header 'Products' with a 'Create new Product' button. A search bar with a magnifying glass icon is present. To the left of the main table, there are dropdown menus for 'Language' (set to English) and 'Subcategory'. The main area displays a table of product data with columns: Name, Subcategory, Lang, Energy, Protein, Fat, and Carbs. The table lists four items: Acerola (Fruits), Asparagus (Vegetables), Cheese (Diary), and Egg (Diary). The Egg entry has numerical values: EN, 357, 84.08, 0.32, 4.51. At the bottom of the table is a navigation bar with buttons for page numbers (1, 2, 3, ..., 20, >, >>).

Name	Subcategory	Lang	Energy	Protein	Fat	Carbs
Acerola	Fruits	EN	23	0.4	0.3	4.8
Asparagus	Vegetables	EN	20	2.2	0.12	3.88
Cheese	Diary	EN	353	21.4	28.74	2.34
Egg	Diary	EN	357	84.08	0.32	4.51

Rysunek 3.7: Prototyp interfejsu - lista produktów (źródło: opr. wł.)

Na rysunkach 3.8 i 3.9 są widoczne odpowiednio prototypy widoków edycji i szczegółów produktu. Oba widoki zawierają takie same pola, jednak w przypadku formularza edycji wartości odżywcze i miary domowe znajdują się w oddzielnych zakładkach, natomiast w widoku szczegółów produktu te sekcje są przedstawione jako karty. Dodatkowo w widoku szczegółów podstawowe wartości odżywcze zostały podsumowane w formie diagramu kołowego. Na uwagę zasługuje fakt, że widok edycji produktu jest tożsamy z widokiem tworzenia nowego produktu.

Dietify

<https://dietify.com/product/new>

DIETIFY

Products

Create or Edit Product

Basic Nutrition Data Household Measures

Description

Source

Category: Diary

Subcategory: Milk

Suitable Diets

Unsuitable Diets

Basic Nutritions

Energy

Fat

Protein

Carbohydrates

Language: English

Cancel Save

Rysunek 3.8: Prototyp interfejsu - dodawanie nowego lub edycja istniejącego produktu
(źródło: opr. wł.)

Dietify

<https://dietify.com/product/1/view>

DIETIFY

Products

Acerola

Edit Delete

Source

Author

Nutrition Data

Calories	100g
Protein	100g
Fat	100g
Carbohydrates	100g
Energy	100g
Protein	100g
Fat	100g
Carbohydrates	100g

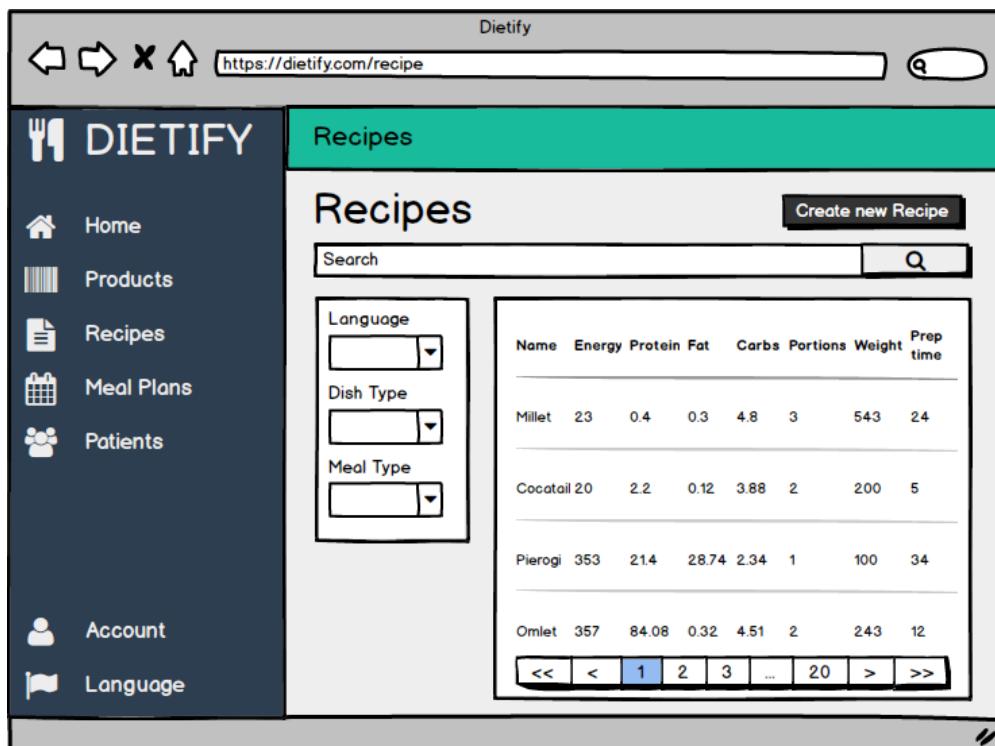
Household Measures

100g	100g

Rysunek 3.9: Prototyp interfejsu - szczegóły produktu (źródło: opr. wł.)

3.1.3. Poddziedzina przepisy

Na rysunku 3.10 przedstawiono prototyp widoku listy przepisów. Z poziomu tego widoku możliwe jest wyszukiwanie i filtrowanie przepisów, wyświetlanie szczegółów przepisu oraz tworzenie nowego przepisu. Do tego widoku prowadzi odnośnik "Recipes" z panelu nawigacyjnego.



The screenshot shows a web-based application interface for 'Dietify'. The top navigation bar includes standard browser controls (back, forward, search) and a URL bar showing 'https://dietify.com/recipe'. Below the header is a dark sidebar with icons and labels: Home, Products, Recipes (selected), Meal Plans, Patients, Account, and Language. The main content area has a teal header 'Recipes' and a 'Create new Recipe' button. A search bar with a magnifying glass icon is positioned above a table. The table has columns for Name, Energy, Protein, Fat, Carbs, Portions, Weight, and Prep time. It lists four items: Millet, Cocatail 20, Pierogi, and Omlet, with detailed nutritional information. On the left side of the table are three dropdown filters: Language, Dish Type, and Meal Type. At the bottom of the table is a pagination control with buttons for '<<', '<', '1' (highlighted in blue), '2', '3', '...', '20', '>', and '>>'.

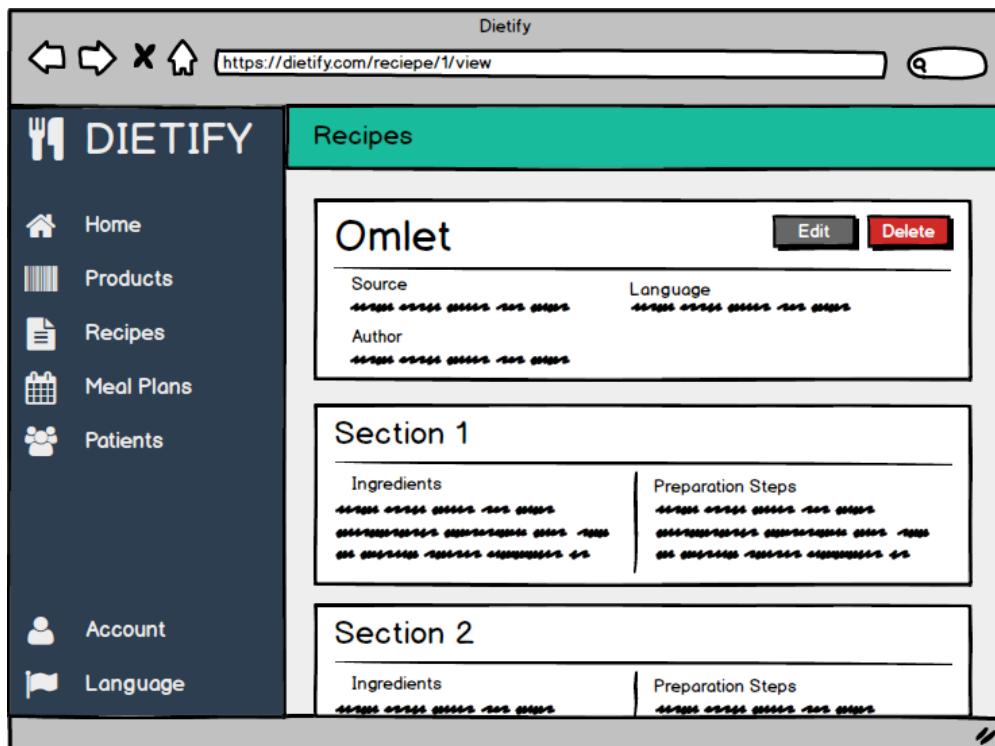
Name	Energy	Protein	Fat	Carbs	Portions	Weight	Prep time
Millet	23	0.4	0.3	4.8	3	543	24
Cocatail 20	2.2	0.12	3.88	2		200	5
Pierogi	353	21.4	28.74	2.34	1	100	34
Omlet	357	84.08	0.32	4.51	2	243	12

Rysunek 3.10: Prototyp interfejsu - lista przepisów (źródło: opr. wł.)

Rysunek 3.11 przedstawia prototyp widoku edycji przepisu. Oprócz definiowania podstawowych informacji dotyczących przepisu, formularz pozwala na dodawanie nowych sekcji, a w każdej sekcji na dodawanie i usuwanie składników i kroków przygotowania. Podczas dodawania nowego składnika zostanie wyświetlony modal zawierający listę produktów, analogiczną jak na rysunku 3.7. Na uwagę zasługuje fakt, że widok edycji przepisu jest tożsamy z widokiem tworzenia nowego przepisu. Po zapisaniu edytowanego przepisu zostanie wyświetlony widok szczegółów przepisu, którego prototyp widoczny jest na rysunku 3.12.



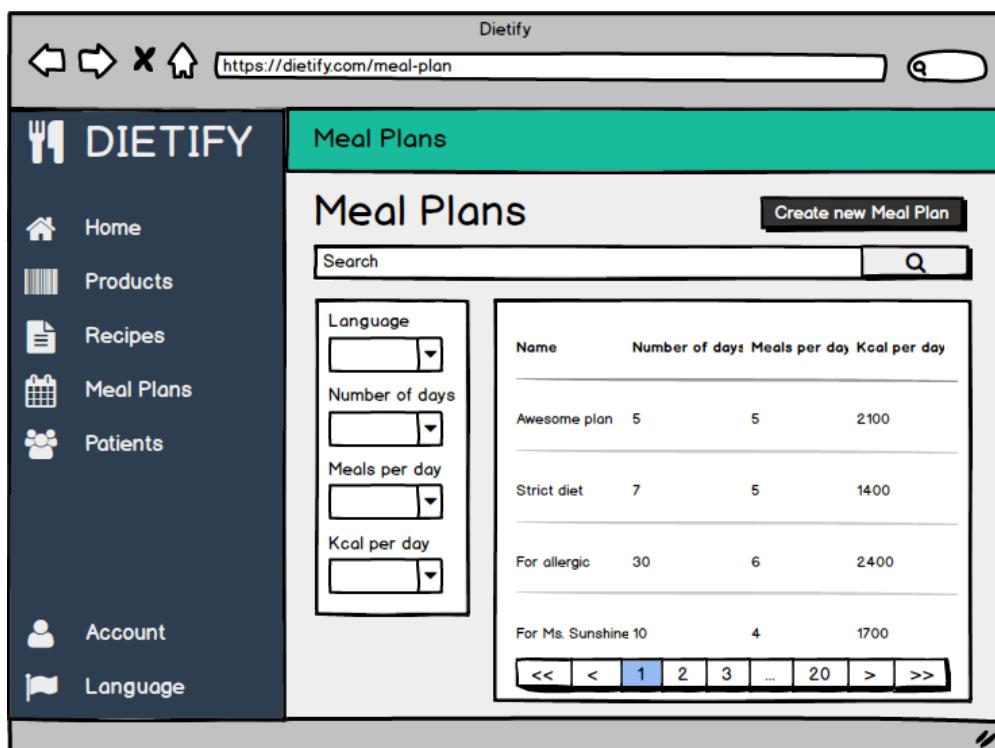
Rysunek 3.11: Prototyp interfejsu - dodawanie nowego lub edycja istniejącego przepisu
(źródło: opr. wł.)



Rysunek 3.12: Prototyp interfejsu - szczegóły przepisu (źródło: opr. wł.)

3.1.4. Poddziedzina jadłospisy

Na rysunku 3.13 przedstawiono prototyp widoku listy jadłospisów. Z poziomu tego widoku możliwe jest wyszukiwanie i filtrowanie jadłospisów, wyświetlanie szczegółów produktu oraz tworzenie nowego jadłospisu. Do tego widoku prowadzi odnośnik "Meal Plans" z panelu nawigacyjnego.



The screenshot shows a web-based application interface for Dietify. On the left, there's a dark sidebar with icons and labels: Home, Products, Recipes, Meal Plans (which is the active tab), Patients, Account, and Language. The main area has a green header bar with the Dietify logo and the title 'Meal Plans'. Below it, there's a search bar with a magnifying glass icon and a 'Create new Meal Plan' button. To the left of the main content, there's a sidebar with dropdown menus for 'Language', 'Number of days', 'Meals per day', and 'Kcal per day'. The main content area displays a table of meal plans:

Name	Number of days	Meals per day	Kcal per day
Awesome plan	5	5	2100
Strict diet	7	5	1400
For allergic	30	6	2400
For Ms. Sunshine 10	4	4	1700

At the bottom of the table, there are navigation buttons: '<<', '<', '1' (highlighted in blue), '2', '3', '...', '20', '>', and '>>'.

Rysunek 3.13: Prototyp interfejsu - lista jadłospisów (źródło: opr. wł.)

Podczas układania jadłospisu dietetyk musi wpierw zdefiniować podstawowe właściwości jadłospisu, co widoczne jest na prototypie widoku zakładki ustawień jadłospisu przedstawionym na rysunku 3.14. Po uzupełnieniu liczby dni i liczby posiłków w ciągu dnia, a także definicji posiłków możliwe jest przejście do zakładki kalendarza, której prototyp widoczny jest na rysunku 3.15. Kalendarz jest automatycznie dostosowywany do liczby dni i liczby posiłków w ciągu dnia. W polach odpowiadających odpowiednim posiłkom w ciągu dnia wyświetlane są produkty i przepisy składające się na posiłek. Po kliknięciu w takie pole możliwe będzie wyświetlenie bardziej szczegółowego widoku posiłku oraz dodawanie produktów po wyświetleniu modala zawierającego listę produktów, analogiczną jak na rysunku 3.7 i dodawanie przepisów po wyświetleniu modala zawierającego listę przepisów, analogiczną jak na rysunku 3.10. Jeśli w zakładce właściwości uzupełniono informacje o procentowej dystrybucji energii na posiłki i na podstawowe wartości odżywcze oraz zdefiniowano oczekiwana dzienną całkowitą oczekiwana wartość energetyczną posiłków to w widoku kalendarza w kolumnie "Summary" widoczne będzie ikonograficzne podsumowanie czy nałożone ograniczenia zostały spełnione. Użytkownik będzie miał możliwość wyświetlić szczegóły tego podsumowania, które udzielą szerszych informacji pozwalających na skorygowanie planu posiłków. Na uwagę zasługuje fakt, że widok edycji jadłospisu jest tożsamy z widokiem tworzenia nowego jadłospisu.



Rysunek 3.14: Prototyp interfejsu - dodawanie nowego lub edycja istniejącego jadłospisu - zakładka ustawień (źródło: opr. wł.)



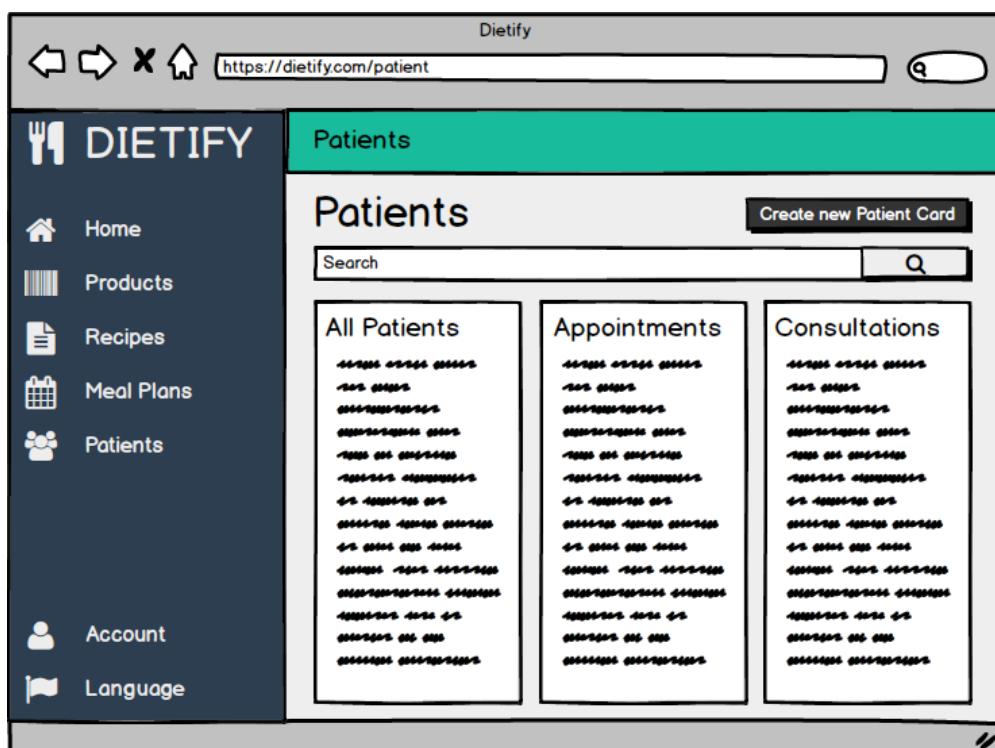
Rysunek 3.15: Prototyp interfejsu - dodawanie nowego lub edycja istniejącego jadłospisu - zakładka kalendarza (źródło: opr. wł.)

3.1.5. Poddziedzina wizyty

Na rysunku 3.16 przedstawiono prototyp widoku listy pacjentów i wizyt. Dietetyk będzie widział 3 kategorie wpisów:

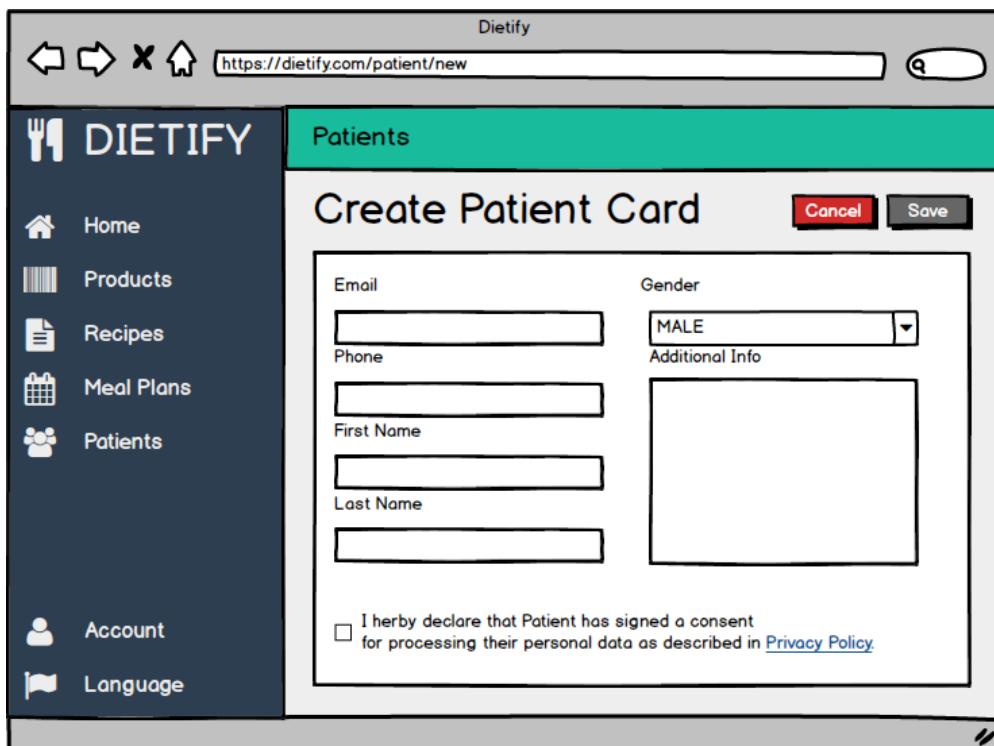
- lista wszystkich pacjentów
- lista zbliżających się wizyt
- lista wizyt, które się odbyły, ale które nadal oczekują na konsultację (np. na komentarz dietetyka, albo na przydzielenie jadłospisu)

Z poziomu tego widoku możliwe jest tworzenie nowej karty pacjenta, wyszukiwanie pacjentów, wyświetlenie karty pacjenta, wyświetlenie szczegółów wizyt nadchodzących i oczekujących na konsultacje. Należy zauważyć, że dietetyk będzie widział tylko karty pacjentów, które sam prowadzi. Do tego widoku prowadzi odnośnik "Patients" z panelu nawigacyjnego.



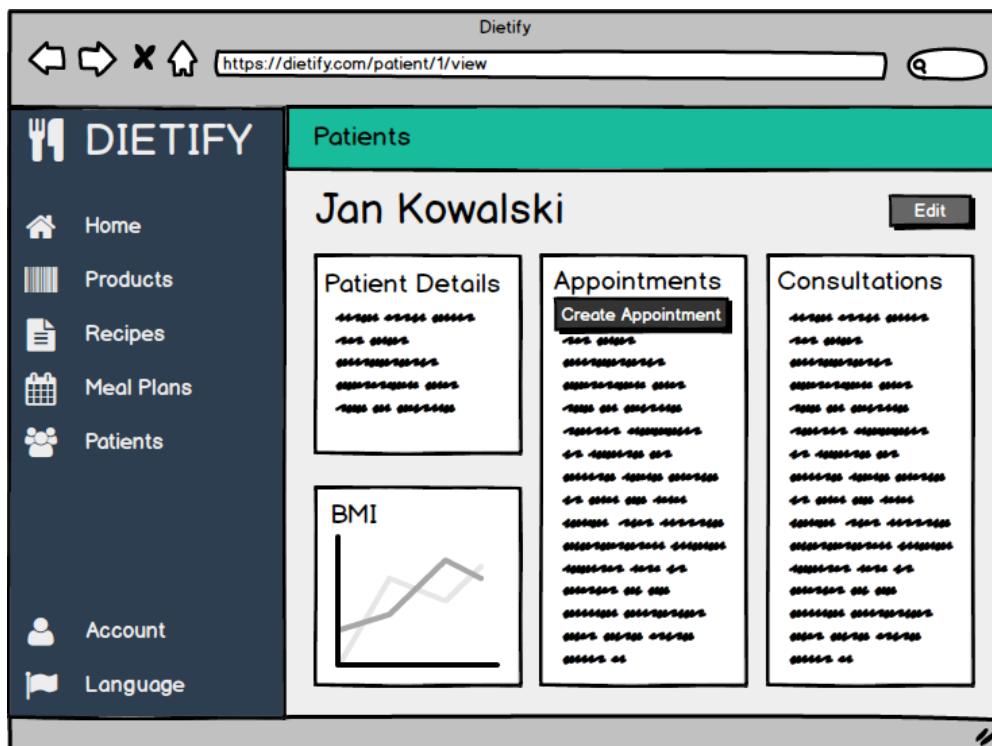
Rysunek 3.16: Prototyp interfejsu - lista wizyt (źródło: opr. wł.)

Rysunek 3.17 przedstawia prototyp widoku tworzenia nowej karty pacjenta. Na szczególną uwagę zasługuje fakt, że formularz poza polami służącymi do zdefiniowania podstawowych danych pacjenta zawiera pole, w którym dietetyk deklaruje się, że uzyskał od pacjenta pisemną zgodę na przetwarzanie jego danych osobowych.



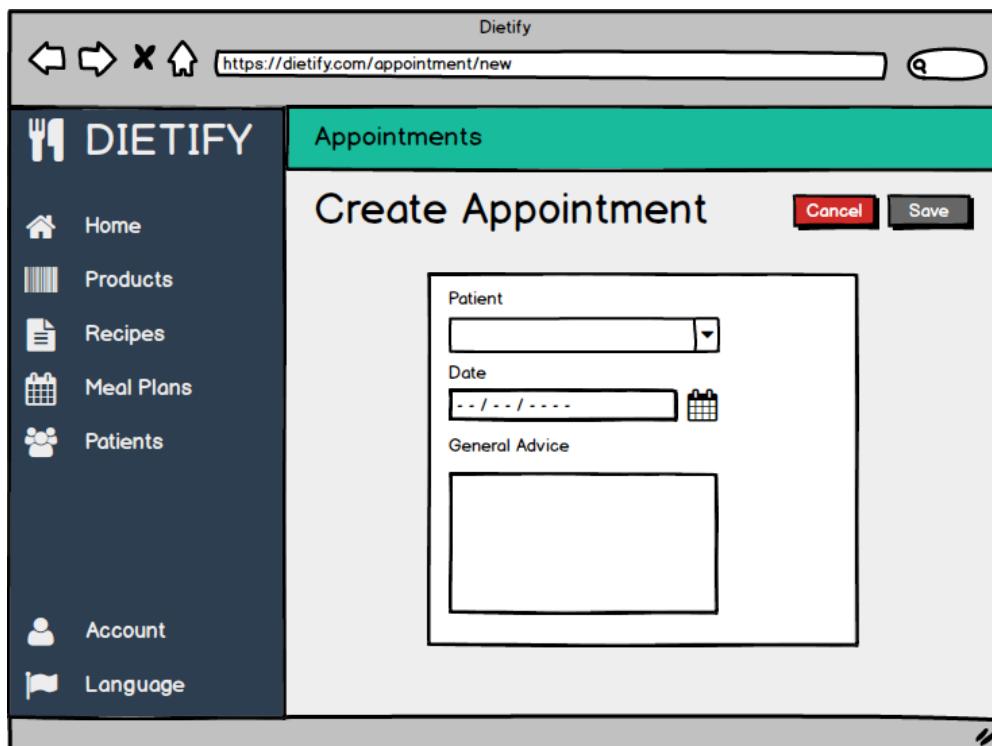
Rysunek 3.17: Prototyp interfejsu - dodawanie nowej karty pacjenta (źródło: opr. wł.)

Prototyp szczegółów karty pacjenta został przedstawiony na rysunku 3.18. Poza podstawowymi danymi pacjenta w obrębie tego widoku wyświetlany jest wykres BMI, utworzony na podstawie pomiarów przeprowadzanych w ramach kolejnych wizyt pacjenta. Z poziomu tego widoku możliwe jest edytowanie karty pacjenta, utworzenie nowej wizyty oraz przeglądanie listy wszystkich wizyt pacjenta oraz listy wizyt oczekujących na konsultację.



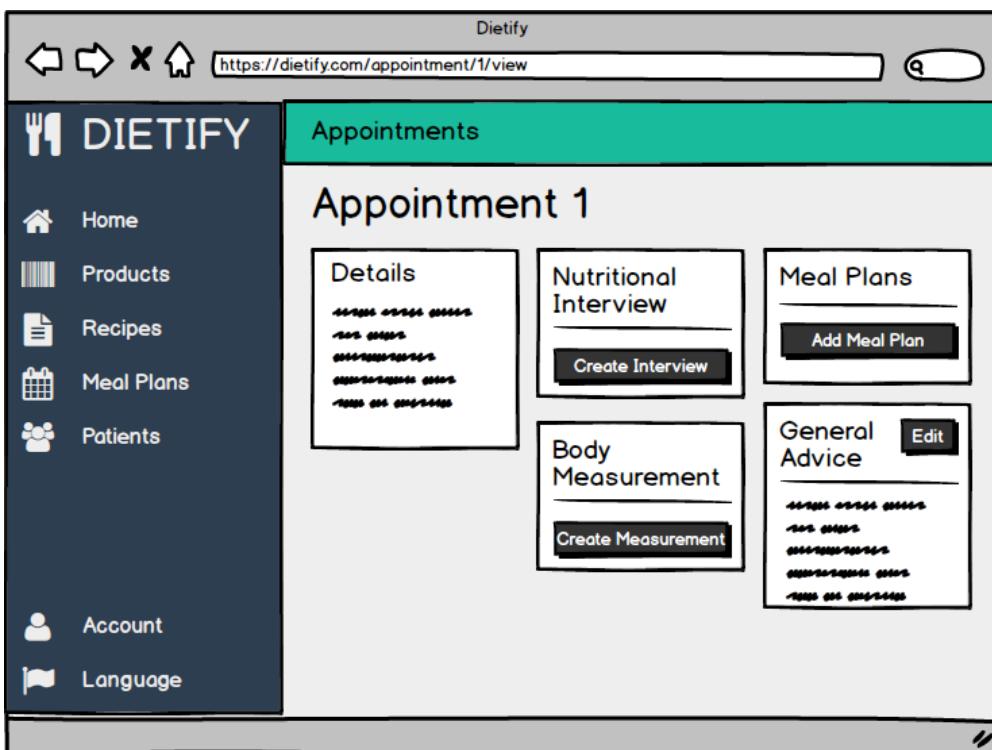
Rysunek 3.18: Prototyp interfejsu - szczegół karty pacjenta (źródło: opr. wł.)

Na rysunku 3.19 przedstawiono prototyp widoku tworzenia nowej wizyty.



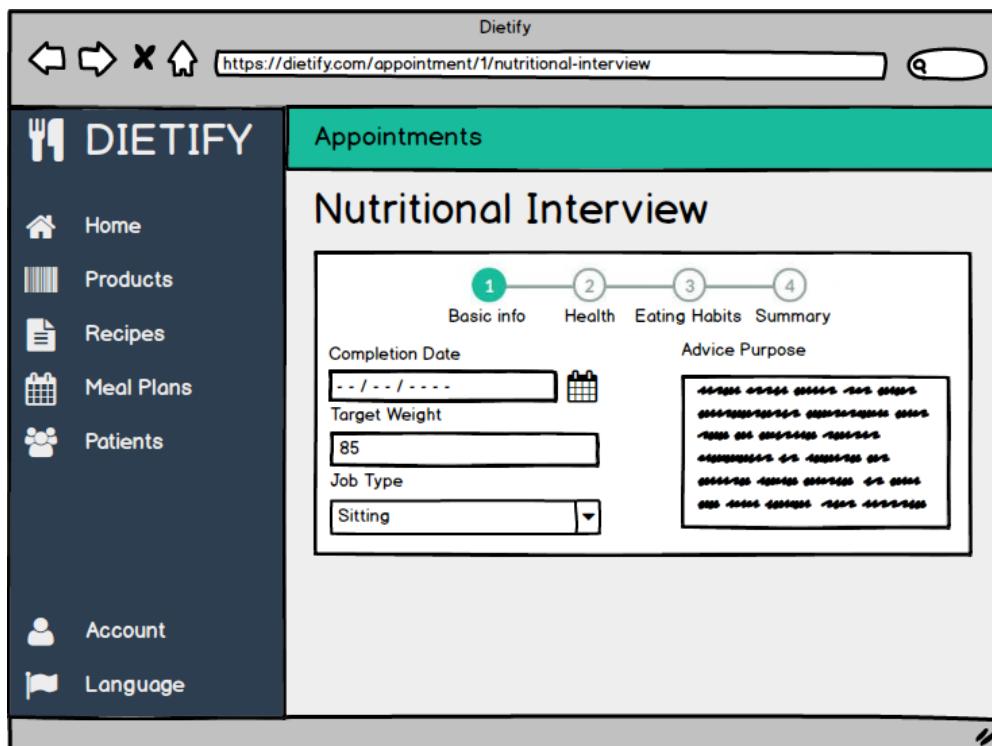
Rysunek 3.19: Prototyp interfejsu - dodawanie nowej wizyty (źródło: opr. wł.)

Rysunek 3.20 przedstawia prototyp widoku szczegółów wizyty. Z poziomu tego widoku możliwe jest zarządzanie przypisanymi do wizyty jadłospisami, pomiarami ciała i wywiadem żywieniowym, a także zmiana statusu wizyty.

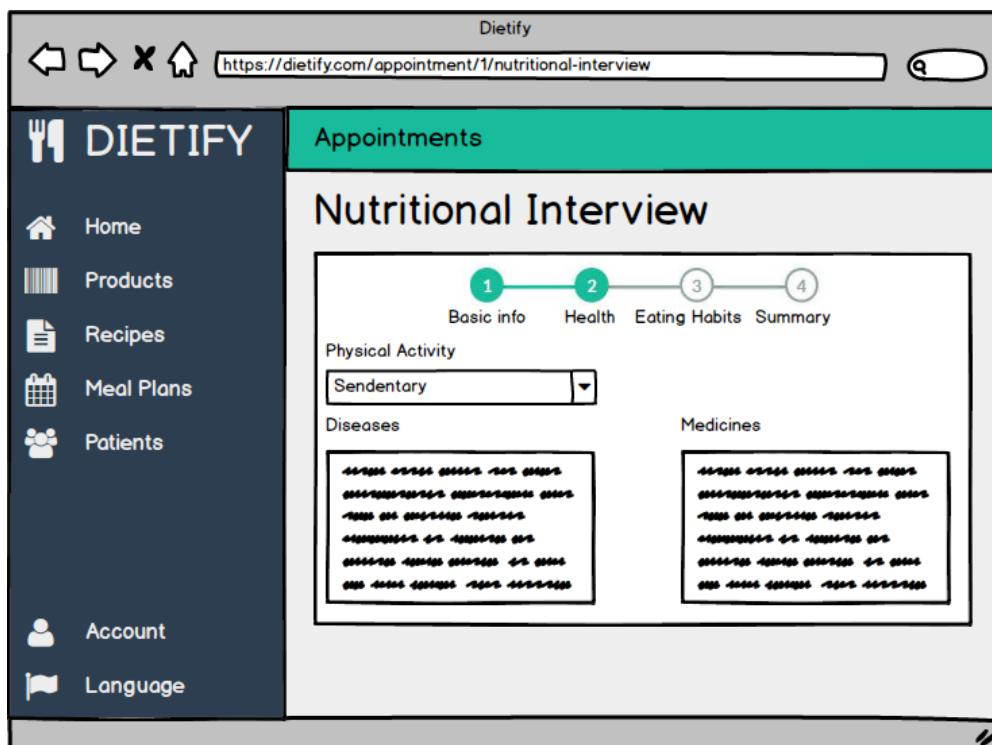


Rysunek 3.20: Prototyp interfejsu - szczegóły wizyty (źródło: opr. wł.)

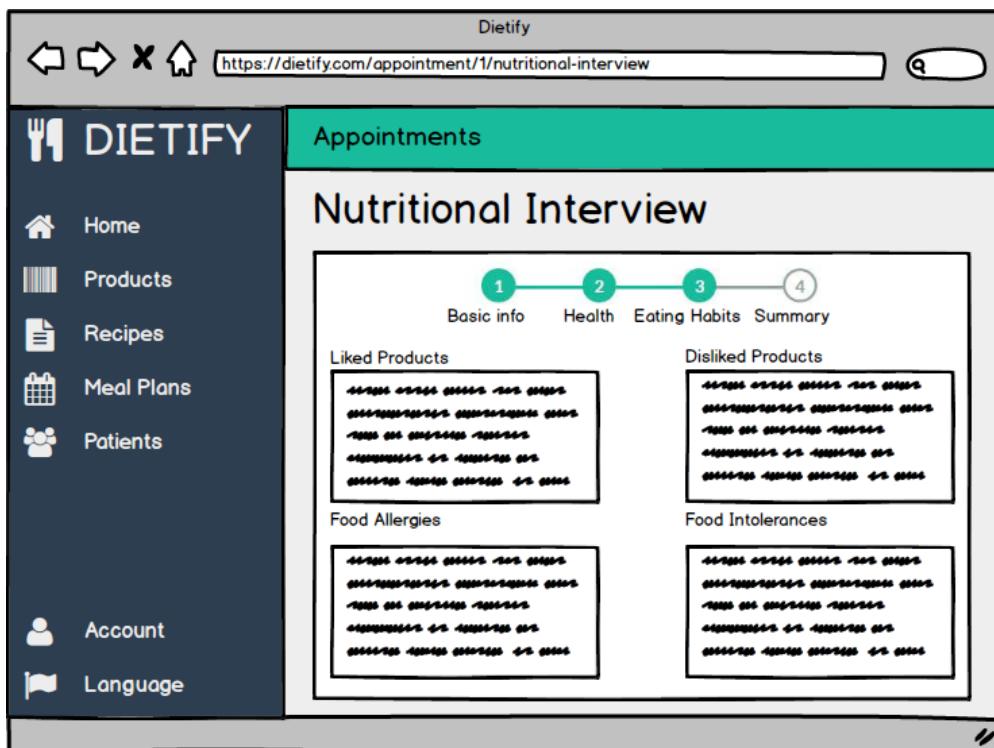
Wywiad żywieniowy podzielony został na kilka etapów. Zbieranie podstawowych informacji dotyczących stanu pacjenta zostało przedstawione na rysunku 3.21, szczegółowe informacje dotyczące zdrowia na rysunku 3.22, a nawyki żywieniowe na rysunku 3.23.



Rysunek 3.21: Prototyp interfejsu - wizyta - wywiad żywieniowy - krok 1 (źródło: opr. wł.)



Rysunek 3.22: Prototyp interfejsu - wizyta - wywiad żywieniowy - krok 2 (źródło: opr. wł.)



Rysunek 3.23: Prototyp interfejsu - wizyta - wywiad żywieniowy - krok 3 (źródło: opr. wł.)

Na rysunku 3.24 przedstawiono prototyp widoku edycji pomiarów ciała. Formularz został podzielony na 2 sekcje:

- podstawowa - podstawowe, wymagane pomiary ciała
- zaawansowana - pomiary, które powinny zostać przeprowadzone metodą BIA

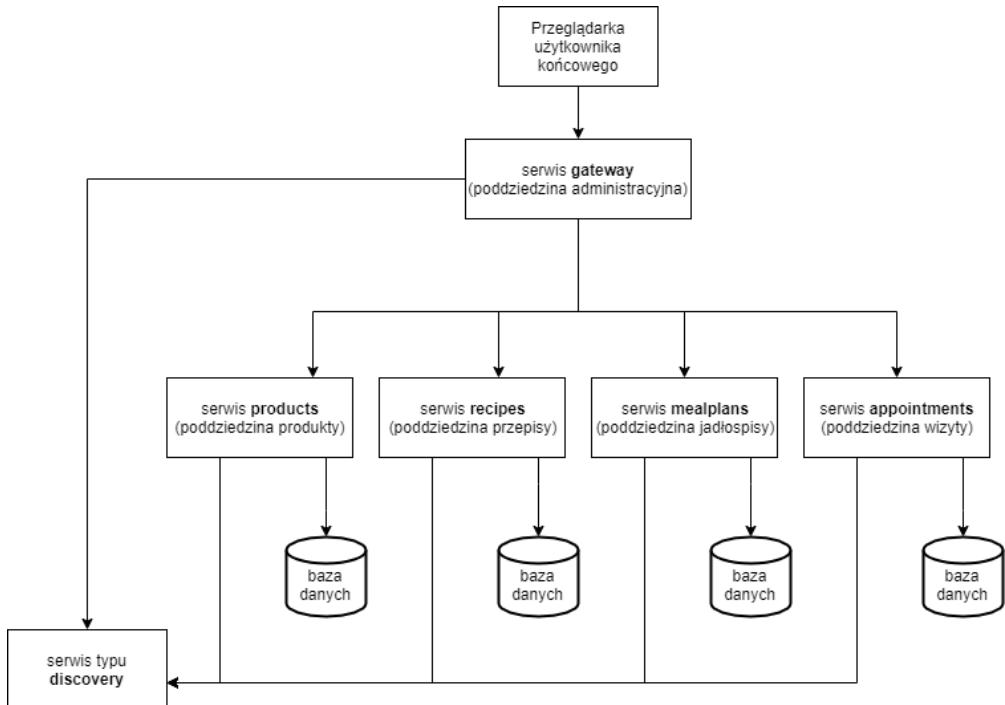
The screenshot shows a web-based application interface for 'Dietify'. The top navigation bar includes standard browser controls (back, forward, search) and a URL bar showing 'https://dietify.com/appointment/1/body-measurements'. The main header 'Appointments' is followed by a sub-header 'Body Measurements'. On the left, a dark sidebar menu lists 'DIETIFY' logo, 'Home', 'Products', 'Recipes', 'Meal Plans', 'Patients', 'Account', and 'Language'. The main content area contains two sections: 'Basic' and 'Advanced'. The 'Basic' section includes fields for 'Completion Date' (with a calendar icon), 'Height', 'Weight', and 'Waist'. The 'Advanced' section includes fields for 'Percent of fat tissue', 'Percent of water', 'Muscle mass', 'Calcium in bones', 'Basic metabolism', and 'Metabolic age'. At the bottom right are 'Cancel' and 'Save' buttons.

Rysunek 3.24: Prototyp interfejsu - wizyta - pomiary ciała (źródło: opr. wł.)

3.2. Opis podstawowej architektury systemu

Jak wspomniano w rozdziale 1.3, dekompozycja w oparciu o poddziedziny pozwala na proste wyznaczenie mikroserwisów poprzez zaprojektowanie serwisu dla każdej poddziedziny. Każdy mikroserwis potrzebuje przechowywać dane w bazie danych, która będzie odseparowana od baz innych mikroserwisów przynajmniej na poziomie logicznym. Instancje mikroserwisów mogą być tworzone dynamicznie i w efekcie mogą być dostępne pod różnymi adresami, należy więc uwzględnić mechanizm pozwalający na automatyczną detekcję bieżących lokalizacji mikroserwisów. Zadanie takie może być realizowane przez serwis typu discovery. Dodatkowo potrzebne jest rozwiązanie umożliwiające odbieranie żądań użytkownika i przekierowywanie ich do odpowiednich serwisów z uwzględnieniem obciążenia (ang. load balancing). Do tego celu służyć może serwis działający jako brama aplikacji (ang. gateway), przy czym warto zauważyć, że brama aplikacji powinna również być odpowiedzialna za autoryzację żądań użytkowników, dlatego też poddziedzina administracyjna będzie zaimplementowana w ramach serwisu gateway.

Na rysunku 3.25 przedstawiono diagram podstawowej architektury uwzględniający powyższe rozważania.



Rysunek 3.25: Podstawowa architektura systemu (źródło: opr. wł.)

3.3. Projekt bazy danych

Podstawą projektowanego systemu jest przetwarzanie danych związanych z domeną zarządzania dietą, dlatego niezmiernie istotne jest zapewnienie trwałości i spójności danych. Cechy takie może zapewnić relacyjna baza danych dzięki mechanizmowi transakcji[36]. Jednakże w przypadku architektury mikroserwisów, w której każdy serwis korzysta z własnego schematu bazy danych, spójność będzie gwarantowana jedynie w obrębie jednego serwisu, gdyż więzy integralności nie będą definiowane pomiędzy danymi znajdującymi się w bazach różnych serwisów. Aby rozwiązać ten problem, dla danych definiowanych przez użytkownika końcowego (dietetyka), do których odniesienia będą przechowywane w serwisie, który nie jest ich właścicielem, dodana zostanie flaga blokująca możliwość usuwania i edycji obiektu używanego przez inny serwis, która będzie aktywowana w momencie, w którym zerwis nie będący właścicielem danych zaczyna z nich korzystać.

W dalszej części niniejszego podrozdziału zostanie przeprowadzone projektowanie konceptualne, w ramach którego dla każdej rozważanej poddziedziny zdefiniowane zostaną kategorie (obiekty) oraz związane z nimi reguły funkcjonowania i ograniczenia dziedzinowe[36], które następnie zostaną podsumowane w formie modelu informacyjnego wyrażonego za pomocą diagramu klas języka UML.

3.3.1. Ogólne cechy dziedziny

Reguły funkcjonowania

Reguły ogólne

REG/0/01 Przedmiot kompozycji podlega takim samym zasadom dostępu co właściciel kompozycji pod warunkiem, że przedmiot kompozycji nie definiuje własnych reguł dostępu.

Ograniczenia dziedzinowe

Ograniczenia ogólne

OGR/0/01 Wszystkie **id** muszą być unikalne.

OGR/0/02 Wszystkie **id** są wymagane.

OGR/0/03 Wszystkie **id** są liczbami całkowitymi dodatnimi tworzonymi przez SZBD za pomocą autonumerowania.

OGR/0/04 Wszystkie atrybuty **language** są wymagane.

OGR/0/05 Wszystkie **language** są ciągami znaków o długości 2 znaków spełniającymi normę ISO 639-1.

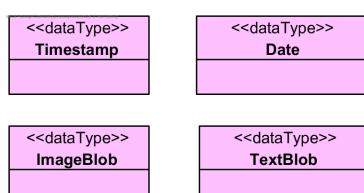
OGR/0/06 Wszystkie **stemple czasowe** są w formacie YYYY:MM:DD HH:MI:SS.

OGR/0/07 Wszystkie **daty** są w formacie YYYY:MM:DD.

OGR/0/08 Ciągi znaków bez dodatkowych ograniczeń mogą zawierać dowolne znaki dopuszczalne w systemie kodowania UTF-8.

Model informacyjny

Na rysunku 3.26 przedstawiono definicje niestandardowych typów danych potrzebnych przy prezentacji modelu informacyjnego.



Rysunek 3.26: Diagram klas - typy danych (źródło: opr. wł.)

3.3.2. Poddziedzina administracyjna

Kategorie

KAT/1/01 User (Użytkownik)

Opis: Konto użytkownika aplikacji. Każdy zalogowany użytkownik musi mieć konto użytkownika.

Atrybuty:

- **id** - identyfikator

• login	- login użytkownika
• passwordHash	- reprezentacja hasła utworzona przez nałożenie na hasło funkcji skrótu
• firstName	- imię użytkownika
• lastName	- nazwisko użytkownika
• email	- adres e-mail
• image	- zdjęcie profilowe użytkownika
• activated	- flaga pokazująca czy konto użytkownika zostało aktywowane
• language	- język użytkownika w postaci kodu ISO 639-1
• activationKey	- klucz wymagany podczas aktywacji konta użytkownika
• resetKey	- klucz wymagany podczas resetowania hasła do konta użytkownika
• createdDate	- data utworzenia konta
• resetDate	- data ostatniego resetowania hasła do konta
• lastModifiedDate	- data ostatniej modyfikacji konta

KAT/1/02 Authority (Rola)

Opis: Rola użytkownika od której zależy zakres uprawnień użytkownika.

Atrybuty:

- | | |
|--------|--------------|
| • name | - nazwa roli |
|--------|--------------|

Reguły funkcjonowania

Reguły dla KAT/1/01 User

- REG/1/01** Użytkownik (**KAT/1/01 User**) musi mieć przynajmniej jedną rolę (**KAT/1/02 Authority**).
- REG/1/02** Użytkownik (**KAT/1/01 User**) może mieć wiele ról (**KAT/1/02 Authority**).
- REG/1/03** Użytkownik (**KAT/1/01 User**) nie musi mieć autora (**KAT/1/01 User**).
- REG/1/04** Użytkownik (**KAT/1/01 User**) może mieć maksymalnie jednego autora (**KAT/1/01 User**).
- REG/1/05** Użytkownik (**KAT/1/01 User**) nie musi mieć ostatniego edytora (**KAT/1/01 User**).
- REG/1/06** Użytkownik (**KAT/1/01 User**) może mieć maksymalnie jednego ostatniego edytora (**KAT/1/01 User**).
- REG/1/07** Gość może dodawać nowego użytkownika (**KAT/1/01 User**).
- REG/1/08** Użytkownik może wyświetlać, edytować i usuwać swoje dane użytkownika (**KAT/1/01 User**).
- REG/1/09** Dietetyk może wyświetlać dane *Pacjenta*, którego kartotekę prowadzi.
- REG/1/10** Administrator może wyświetlać i usuwać dane użytkownika (**KAT/1/01 User**).

Reguły dla KAT/1/02 Authority

- REG/1/11** Administrator może dodawać, wyświetlać, edytować i usuwać dane roli (**KAT/1/02 Authority**).

Ograniczenia dziedzinowe

Ograniczenia dla KAT/1/01 User

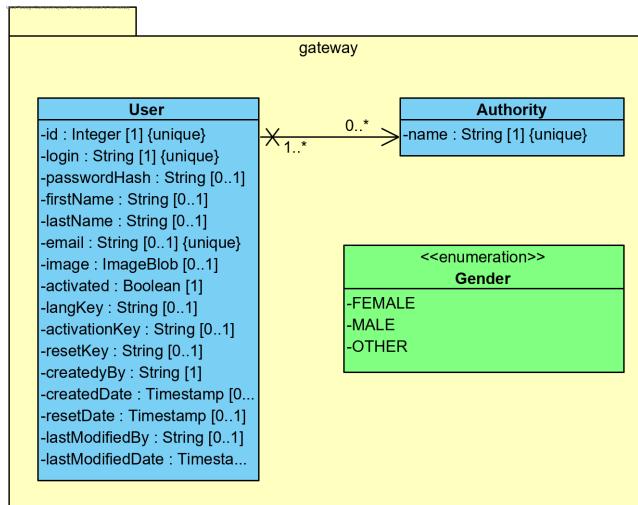
- OGR/1/01** Atrybut **login** jest wymagany.
- OGR/1/02** Atrybut **passwordHash** jest wymagany.
- OGR/1/03** Atrybut flagę **activated** jest wymagany.
- OGR/1/04** Atrybut **createdDate** jest wymagany.
- OGR/1/05** Atrybut **login** ma unikalną wartość.
- OGR/1/06** Atrybut **email** ma unikalną wartość.
- OGR/1/07** Atrybut **login** jest ciągiem znaków składającym się z liter, cyfr i dodatkowo mogącym zawierać znaki ".", "-", "@" o długości od 1 do 50 znaków.
- OGR/1/08** Atrybut **passwordHash** jest ciągiem znaków o długości 60 znaków.
- OGR/1/09** Atrybut **firstName** jest ciągiem znaków o długości do 50 znaków.
- OGR/1/10** Atrybut **lastName** jest ciągiem znaków o długości do 50 znaków.
- OGR/1/11** Atrybut **email** jest ciągiem znaków o długości od 5 do 254 znaków.
- OGR/1/12** Atrybut **activated** jest typem logicznym.
- OGR/1/13** Atrybut **image** jest ciągiem znaków o długości do 256 znaków tworzącym poprawny adres URL.
- OGR/1/14** Atrybut **activationKey** jest ciągiem znaków o długości 20 znaków.
- OGR/1/15** Atrybut **resetKey** jest ciągiem znaków o długości 20 znaków.
- OGR/1/16** Atrybut **resetDate** jest stemplem czasowym.
- OGR/1/17** Atrybut **createdDate** jest stemplem czasowym.
- OGR/1/18** Atrybut **lastModifiedDate** jest stemplem czasowym.

Ograniczenia dla KAT/1/02 Authority

- OGR/1/19** Atrybut **name** jest wymagany.
- OGR/1/20** Atrybut **name** ma unikalną wartość.
- OGR/1/21** Atrybut **name** jest ciągiem znaków składającym się z liter i znaków "_" o długości od 1 do 255 znaków.

Model informacyjny

Na rysunku 3.27 przedstawiono model informacyjny dla poddziedziny administracyjnej w formie diagramu klas języka UML.



Rysunek 3.27: Diagram klas - poddziedzina administracyjna (źródło: opr. wł.)

3.3.3. Poddziedzina produkty

Kategorie

KAT/2/01 Product (Produkt)

Opis: Wersja produktu. Podczas każdej edycji powstaje nowa wersja. Wersji nie można usunąć, można ją jedynie zastąpić nową wersją.

Atrybuty:

- | | |
|---------------|--|
| • id | - identyfikator |
| • createdDate | - czas utworzenia wersji |
| • description | - krótki opis produktu w języku produktu |
| • source | - źródło produktu, jeśli produkt jest importowany; preferowany format adresu URL |
| • isFinal | - flaga określająca czy produkt jest dostępny do edycji czy jest zablokowany |
| • language | - język tłumaczenia w postaci kodu ISO 639-1 |

KAT/2/02 ProductBasicNutritionData (Podstawowe Składniki Odżywcze Produktu)

Opis: Podstawowe składniki odżywcze produktu.

Atrybuty:

- | | |
|-----------------|---|
| • id | - identyfikator |
| • energy | - energia w kilokaloriach (kcal) na 100 gramów produktu |
| • protein | - białko w gramach na 100 gramów produktu |
| • fat | - tłuszcze w gramach na 100 gramów produktu |
| • carbohydrates | - węglowodany w gramach na 100 gramów produktu |

KAT/2/03 NutritionData (Wartość Odżywczna)

Opis: Wartość wartości odżywczej dla konkretnego produktu.

Atrybuty:

- id - identyfikator
- nutritionValue - ilość składnika odżywczego w jednostkach wyspecyfikowanych w definicji wartości odżywczej

KAT/2/04 NutritionDefinition (Definicja Wartości Odżywczej)

Opis: Definicja wartości odżywczej.

Atrybuty:

- id - identyfikator
- tag - krótki znacznik reprezentujący wartość odżywczą
- description - krótki opis wartości odżywczej w języku angielskim
- units - jednostki wykorzystywane do pomiaru wartości odżywczej, np. "g", "kcal", "ml"
- decimalPlaces - liczba miejsc dziesiętnych do których wartość składnika odżywczego powinna być zaokrąglana

KAT/2/05 NutritionDefinitionTranslation (Tłumaczanie Definicji Wartości Odżywczej)

Opis: Tłumaczanie definicji wartości odżywczej.

Atrybuty:

- id - identyfikator
- translation - przetłumaczony opis definicji wartości odżywczej
- language - język tłumaczenia w postaci kodu ISO 639-1

KAT/2/06 HouseholdMeasure (Miara Domowa)

Opis: Miara domowa produktu z masą w gramach.

Atrybuty:

- id - identyfikator
- description - krótki opis miary domowej w języku produktu, np. "szklanka" lub "łyżeczka"
- gramsWeight - masa w gramach 1 jednostki miary domowej
- isVisible - flaga określająca czy miara jest widoczna podczas wyświetlania produktu

KAT/2/07 ProductSubcategory (Podkategoria Produktu)

Opis: Podkategoria produktu.

Atrybuty:

- id - identyfikator
- description - krótki opis podkategorii w języku produktu

KAT/2/08 ProductCategory (Kategoria Produktu)

Opis: Główna kategoria produktu.

Atrybuty:

- id - identyfikator
- description - krótki opis kategorii produktu w języku angielskim

KAT/2/09 ProductCategoryTranslation (Tłumaczenie Kategorii Produktu)

Opis: Tłumaczenie kategorii produktu.

Atrybuty:

- | | |
|---------------|--|
| • id | - identyfikator |
| • translation | - przetłumaczona nazwa kategorii produktu |
| • language | - język tłumaczenia w postaci kodu ISO 639-1 |

KAT/2/10 DietType (Typ Diety)

Opis: Typ diety.

Atrybuty:

- | | |
|--------|--|
| • id | - identyfikator |
| • name | - krótki opis typu diety w języku angielskim |

KAT/2/11 DietTypeTranslation (Tłumaczenie Typu Diety)

Opis: Tłumaczenie typu diety.

Atrybuty:

- | | |
|---------------|--|
| • id | - identyfikator |
| • translation | - tłumaczenie nazwy typu diety |
| • language | - język tłumaczenia w postaci kodu ISO 639-1 |

Reguły funkcjonowania

Reguły dla KAT/2/01 Product

- REG/2/01** Produkt (**KAT/2/01 Product**) nie musi mieć zdefiniowanego autora (**KAT/1/01 User**).
- REG/2/02** Produkt (**KAT/2/01 Product**) może mieć maksymalnie jednego autora (**KAT/1/01 User**).
- REG/2/03** Produkt (**KAT/2/01 Product**) musi być przypisana do dokładnie jednych podstawowych wartości odżywcznych (**KAT/2/02 ProductBasicNutrition-Data**).
- REG/2/04** Produkt (**KAT/2/01 Product**) nie musi mieć zdefiniowanych żadnych wartości odżywcznych (**KAT/2/03 NutritionData**).
- REG/2/05** Produkt (**KAT/2/01 Product**) może mieć zdefiniowane wiele wartości odżywcznych (**KAT/2/03 NutritionData**).
- REG/2/06** Produkt (**KAT/2/01 Product**) nie musi mieć zdefiniowanych żadnych miar domowych (**KAT/2/06 HouseholdMeasure**).
- REG/2/07** Produkt (**KAT/2/01 Product**) może mieć zdefiniowane wiele miar domowych (**KAT/2/06 HouseholdMeasure**).
- REG/2/08** Produkt (**KAT/2/01 Product**) musi należeć do dokładnie jednej podkategorii (**KAT/2/07 ProductSubcategory**).
- REG/2/09** Produkt (**KAT/2/01 Product**) nie musi mieć przypisanego żadnego odpowiedniego typu diety (**KAT/2/10 DietType**).
- REG/2/10** Produkt (**KAT/2/01 Product**) może mieć przypisanych wiele odpowiednich typów diety (**KAT/2/10 DietType**).
- REG/2/11** Produkt (**KAT/2/01 Product**) nie musi mieć przypisanego żadnego nieodpowiedniego typu diety (**KAT/2/10 DietType**).

- REG/2/12** Produkt (**KAT/2/01 Product**) może mieć przypisanych wiele nieodpowiednich typów diety (**KAT/2/10 DietType**).
- REG/2/13** *Dietetyk* może wyświetlać publiczne produkty (**KAT/2/01 Product**).
- REG/2/14** *Dietetyk* może dodawać, wyświetlać, edytować i usuwać własne produkty (**KAT/2/01 Product**).
- REG/2/15** *Administrator* może wyświetlać i usuwać produkty (**KAT/2/01 Product**).
- REG/2/16** Produkt (**KAT/2/01 Product**) oznaczony jako ostateczny nie może być usunięty ani edytowany.

Reguły dla KAT/2/02 ProductBasicNutritionData

- REG/2/17** Podstawowe wartości odżywcze produktu(**KAT/2/02 ProductBasicNutritionData**) muszą być przypisane do dokładnie jednego produktu (**KAT/2/01 Product**).
- REG/2/18** Podstawowe wartości odżywcze produktu (**KAT/2/02 ProductBasicNutritionData**) są przedmiotem kompozycji ze strony produktu (**KAT/2/01 Product**).

Reguły dla KAT/2/03 NutritionData

- REG/2/19** Wartość odżywca (**KAT/2/03 NutritionData**) musi być przypisana do dokładnie jednego produktu (**KAT/2/01 Product**).
- REG/2/20** Wartość odżywca (**KAT/2/03 NutritionData**) musi być przypisana do dokładnie jednej definicji wartości odżywczej (**KAT/2/04 NutritionDefinition**).
- REG/2/21** Wartość odżywca (**KAT/2/03 NutritionData**) jest przedmiotem kompozycji ze strony produktu (**KAT/2/01 Product**).

Reguły dla KAT/2/04 NutritionDefinition

- REG/2/22** Definicja wartości odżywczej (**KAT/2/04 NutritionDefinition**) nie musi mieć zdefiniowanego żadnego tłumaczenia (**KAT/2/05 NutritionDefinitionTranslation**).
- REG/2/23** Definicja wartości odżywczej (**KAT/2/04 NutritionDefinition**) może mieć zdefiniowanych wiele tłumaczeń (**KAT/2/05 NutritionDefinitionTranslation**).
- REG/2/24** *Dietetyk* może wyświetlać definicję wartości odżywczej (**KAT/2/04 NutritionDefinition**).
- REG/2/25** *Administrator* może dodawać, wyświetlać, edytować i usuwać definicję wartości odżywczej (**KAT/2/04 NutritionDefinition**).

Reguły dla KAT/2/05 NutritionDefinitionTranslation

- REG/2/26** Tłumaczenie definicji wartości odżywczej (**KAT/2/05 NutritionDefinitionTranslation**) musi być przypisane do dokładnie jednej definicji wartości odżywczej (**KAT/2/04 NutritionDefinition**).
- REG/2/27** Tłumaczenie definicji wartości odżywczej (**KAT/2/05 NutritionDefinitionTranslation**) jest przedmiotem kompozycji ze strony definicji wartości odżywczej (**KAT/2/04 NutritionDefinition**).

Reguły dla KAT/2/06 HouseholdMeasure

- REG/2/28** Miara domowa (**KAT/2/06 HouseholdMeasure**) musi być przypisana do dokładnie jednego produktu (**KAT/2/01 Product**).
- REG/2/29** Miara domowa (**KAT/2/06 HouseholdMeasure**) jest przedmiotem kompozycji ze strony produktu (**KAT/2/01 Product**).

Reguły dla KAT/2/07 ProductSubcategory

- REG/2/30** Podkategoria produktu (**KAT/2/07 ProductSubcategory**) musi być przypisana do co najmniej jednego produktu (**KAT/2/01 Product**).
- REG/2/31** Podkategoria produktu (**KAT/2/07 ProductSubcategory**) może być przypisana do wielu produktów (**KAT/2/01 Product**).
- REG/2/32** Podkategoria produktu (**KAT/2/07 ProductSubcategory**) musi być przypisana do dokładnie jednej kategorii (**KAT/2/08 ProductCategory**).
- REG/2/33** *Dietetyk* może dodawać i wyświetlać podkategorię produktu (**KAT/2/07 ProductSubcategory**).
- REG/2/34** *Administrator* może wyświetlać podkategorię produktu (**KAT/2/07 ProductSubcategory**).

Reguły dla KAT/2/08 ProductCategory

- REG/2/35** Kategoria produktu (**KAT/2/08 ProductCategory**) nie musi mieć przypisanego żadnego tłumaczenia (**KAT/2/09 ProductCategoryTranslation**).
- REG/2/36** Kategoria produktu (**KAT/2/08 ProductCategory**) może mieć przypisanych wiele tłumaczeń (**KAT/2/09 ProductCategoryTranslation**).
- REG/2/37** *Dietetyk* może wyświetlać kategorię produktu (**KAT/2/08 ProductCategory**).
- REG/2/38** *Administrator* może dodawać, wyświetlać, edytować i usuwać kategorię produktu (**KAT/2/08 ProductCategory**).

Reguły dla KAT/2/09 ProductCategoryTranslation

- REG/2/39** Tłumaczenie kategorii produktu (**KAT/2/09 ProductCategoryTranslation**) musi być przypisane do dokładnie jednej kategorii (**KAT/2/08 ProductCategory**).
- REG/2/40** Tłumaczenie kategorii produktu (**KAT/2/09 ProductCategoryTranslation**) jest przedmiotem kompozycji ze strony kategorii (**KAT/2/08 ProductCategory**).

Reguły dla KAT/2/10 DietType

- REG/2/41** Typ diety (**KAT/2/10 DietType**) nie musi mieć zdefiniowanego żadnego tłumaczenia (**KAT/2/11 DietTypeTranslation**).
- REG/2/42** Typ diety (**KAT/2/10 DietType**) może mieć zdefiniowanych wiele tłumaczeń (**KAT/2/11 DietTypeTranslation**).
- REG/2/43** *Dietetyk* może wyświetlać typ diety (**KAT/2/10 DietType**).
- REG/2/44** *Administrator* może dodawać, wyświetlać, edytować i usuwać typ diety (**KAT/2/10 DietType**).

Reguły dla KAT/2/11 DietTypeTranslation

- REG/2/45** Tłumaczenie typu diety (**KAT/2/11 DietTypeTranslation**) musi być przypisane do dokładnie jednego typu diety (**KAT/2/10 DietType**).
REG/2/46 Tłumaczenie typu diety (**KAT/2/11 DietTypeTranslation**) jest przedmiotem kompozycji ze strony typu diety (**KAT/2/10 DietType**).

Ograniczenia dziedzinowe

Ograniczenia dla KAT/2/01 Product

- OGR/2/01** Atrybut **createdDate** jest wymagany.
OGR/2/02 Atrybut **description** jest wymagany.
OGR/2/03 Atrybut **isFinal** jest wymagany.
OGR/2/04 Atrybut **createdDate** jest stemplem czasowym.
OGR/2/05 Atrybut **description** jest ciągiem znaków o długości od 1 do 255 znaków.
OGR/2/06 Atrybut **source** jest ciągiem znaków o długości od 1 do 255 znaków.
OGR/2/07 Atrybut **isFinal** jest typu logicznego.

Ograniczenia dla KAT/2/02 ProductBasicNutritionData

- OGR/2/08** Atrybut **energy** jest wymagany.
OGR/2/09 Atrybut **protein** jest wymagany.
OGR/2/10 Atrybut **fat** jest wymagany.
OGR/2/11 Atrybut **carbohydrates** jest wymagany.
OGR/2/12 Atrybut **energy** jest liczbą rzeczywistą nie mniejszą niż 0.
OGR/2/13 Atrybut **protein** jest liczbą rzeczywistą nie mniejszą niż 0.
OGR/2/14 Atrybut **fat** jest liczbą rzeczywistą nie mniejszą niż 0.
OGR/2/15 Atrybut **carbohydrates** jest liczbą rzeczywistą nie mniejszą niż 0.

Ograniczenia dla KAT/2/03 NutritionData

- OGR/2/16** Atrybut **nutritionValue** jest wymagany.
OGR/2/17 Atrybut **nutritionValue** jest liczbą rzeczywistą nie mniejszą niż 0.

Ograniczenia dla KAT/2/04 NutritionDefinition

- OGR/2/18** Atrybut **tag** jest wymagany.
OGR/2/19 Atrybut **description** jest wymagany.
OGR/2/20 Atrybut **units** jest wymagany.
OGR/2/21 Atrybut **decimalPlaces** jest wymagany.
OGR/2/22 Atrybut **tag** ma unikalną wartość.
OGR/2/23 Atrybut **tag** jest ciągiem znaków o długości od 1 do 20 znaków.
OGR/2/24 Atrybut **description** jest ciągiem znaków o długości od 1 do 255 znaków.
OGR/2/25 Atrybut **units** jest ciągiem znaków o długości od 1 do 10 znaków.
OGR/2/26 Atrybut **decimalPlaces** jest liczbą całkowitą nie mniejszą niż 0.

Ograniczenia dla KAT/2/05 NutritionDefinitionTranslation

- OGR/2/27** Atrybut **translation** jest wymagany.
OGR/2/28 Atrybut **translation** jest ciągiem znaków o długości od 1 do 255 znaków.

Ograniczenia dla KAT/2/06 HouseholdMeasure

- OGR/2/29** Atrybut **description** jest wymagany.
- OGR/2/30** Atrybut **gramsWeight** jest wymagany.
- OGR/2/31** Atrybut **isVisible** jest wymagany.
- OGR/2/32** Atrybut **description** jest ciągiem znaków o długości od 1 do 255 znaków.
- OGR/2/33** Atrybut **gramsWeight** jest liczbą rzeczywistą nie mniejszą niż 0.
- OGR/2/34** Atrybut **isVisible** jest typu logicznego.

Ograniczenia dla KAT/2/07 ProductSubcategory

- OGR/2/35** Atrybut **description** jest wymagany.
- OGR/2/36** Atrybut **description** jest ciągiem znaków o długości od 1 do 255 znaków.

Ograniczenia dla KAT/2/08 ProductCategory

- OGR/2/37** Atrybut **description** jest wymagany.
- OGR/2/38** Atrybut **description** ma unikalną wartość.
- OGR/2/39** Atrybut **description** jest ciągiem znaków o długości od 1 do 255 znaków.

Ograniczenia dla KAT/2/09 ProductCategoryTranslation

- OGR/2/40** Atrybut **translation** jest wymagany.
- OGR/2/41** Atrybut **translation** jest ciągiem znaków o długości od 1 do 255 znaków.

Ograniczenia dla KAT/2/10 DietType

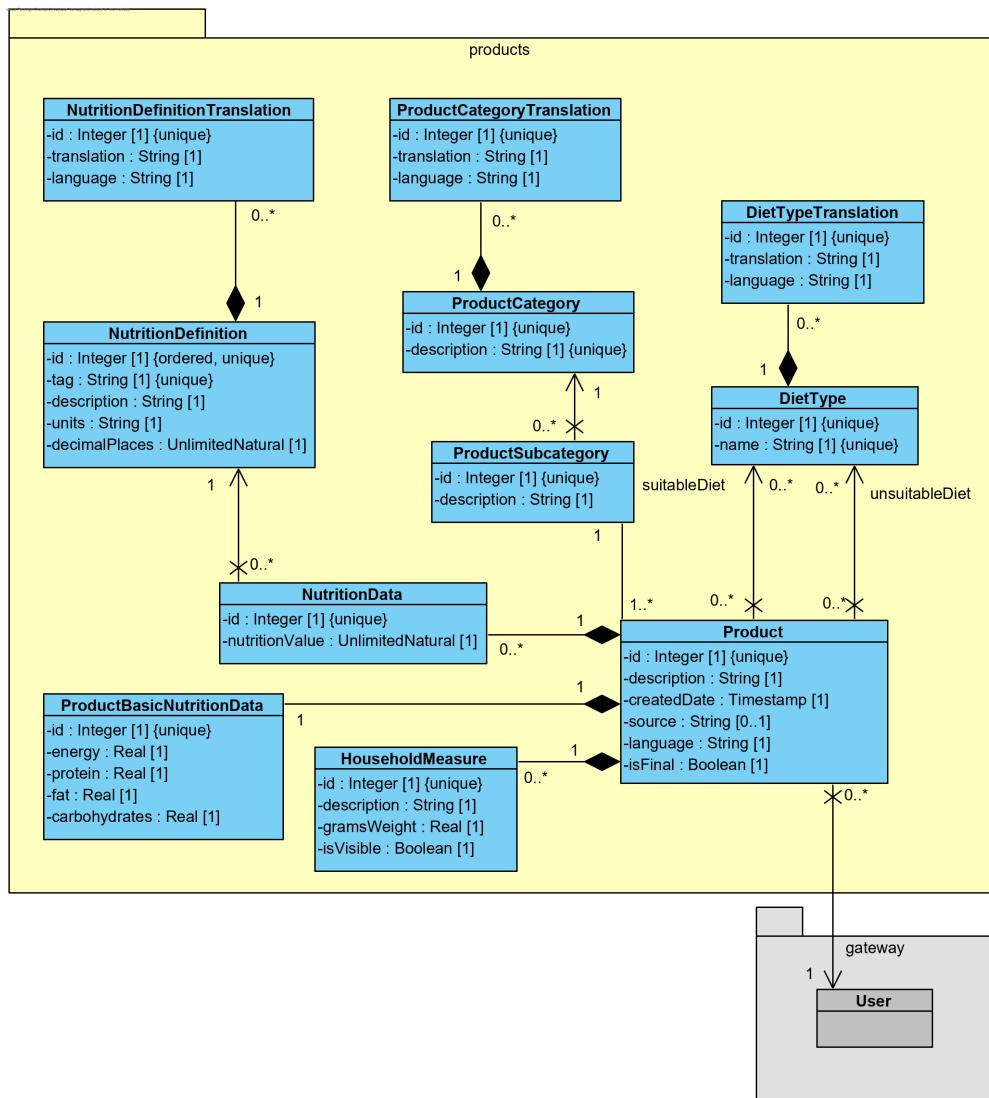
- OGR/2/42** Atrybut **name** jest wymagany.
- OGR/2/43** Atrybut **name** ma unikalną wartość.
- OGR/2/44** Atrybut **name** jest ciągiem znaków o długości od 1 do 255 znaków.

Ograniczenia dla KAT/2/11 DietTypeTranslation

- OGR/2/45** Atrybut **translation** jest wymagany.
- OGR/2/46** Atrybut **translation** jest ciągiem znaków o długości od 1 do 255 znaków.

Model informacyjny

Na rysunku 3.28 przedstawiono model informacyjny dla poddziedziny produkty w formie diagramu klas języka UML.



Rysunek 3.28: Diagram klas - poddziedzina produkty (źródło: opr. wł.)

3.3.4. Poddziedzina przepisy

Kategorie

KAT/3/01 Recipe (Przepis)

Opis: Wersja przepisu. Podczas każdej edycji powstaje nowa wersja. Wersji nie można usunąć, można ją jedynie zastąpić nową wersją.

Atrybuty:

- | | |
|--------------------------|--|
| • id | - identyfikator |
| • editTimestamp | - czas utworzenia wersji |
| • name | - nazwa przepisu w języku przepisu |
| • preparationTimeMinutes | - średni czas potrzebny na całkowite przygotowanie przepisu, zdefiniowany w minutach |
| • numberOfPortions | - liczba porcji dla których przepis jest zdefiniowany |

- image - opcjonalne zdjęcie dania przygotowanego na podstawie przepisu
- totalGramsWeight - całkowita masa w gramach dania przygotowanego z przepisu
- isFinal - flaga określająca czy przepis jest dostępny do edycji czy jest zablokowany
- language - język tłumaczenia w postaci kodu ISO 639-1

KAT/3/02 RecipeBasicNutritionData (Podstawowe Wartości Odżywcze Przepisu)

Opis: Podstawowe wartości odżywcze przepisu.

Atrybuty:

- | | |
|-----------------|---|
| • id | - identyfikator |
| • energy | - energia w kilokaloriach (kcal) na 100 gramów posiłku przygotowanego z użyciem produktów w przepisie |
| • protein | - białko w gramach na 100 gramów posiłku przygotowanego z użyciem produktów w przepisie |
| • fat | - tłuszcz w gramach na 100 gramów posiłku przygotowanego z użyciem produktów w przepisie |
| • carbohydrates | - węglowodany w gramach na 100 gramów posiłku przygotowanego z użyciem produktów w przepisie |

KAT/3/03 RecipeSection (Sekcja Przepisu)

Opis: Sekcja przepisu, np. sernik może mieć 3 sekcje odpowiednio dla ciasta, masy i polewy.

Atrybuty:

- | | |
|---------------|------------------------------------|
| • id | - identyfikator |
| • sectionName | - nazwa przepisu w języku przepisu |

KAT/3/04 ProductPortion (Porcja Produktu)

Opis: Porcja produktu wykorzystywana w przepisie.

Atrybuty:

- | | |
|----------|---|
| • id | - identyfikator |
| • amount | - ilość produktu w jednostkach miary domowej lub w gramach jeśli miara domowa nie jest zdefiniowana |

KAT/3/05 PreparationStep (Krok Przygotowania)

Opis: Krok przygotowania w przepisie.

Atrybuty:

- | | |
|-------------------|--|
| • id | - identyfikator |
| • ordinalNumber | - liczba porządkowa kroku przygotowania |
| • stepDescription | - w miarę możliwości krótki opis kroku przygotowania |

KAT/3/06 KitchenAppliance (Sprzęt Kuchenny)

Opis: Definicja sprzętu kuchennego.

Atrybuty:

- | | |
|------|-----------------|
| • id | - identyfikator |
|------|-----------------|

- name - nazwa sprzętu kuchennego w języku angielskim

KAT/3/07 KitchenApplianceTranslation (Tłumaczenie Sprzętu Kuchennego)

Opis: Tłumaczenie sprzętu kuchennego.

Atrybuty:

- | | |
|---------------|--|
| • id | - identyfikator |
| • translation | - przetłumaczona nazwa sprzętu kuchennego |
| • language | - język tłumaczenia w postaci kodu ISO 639-1 |

KAT/3/08 DishType (Typ Dania)

Opis: Typ dania, np. sałatka lub zupa.

Atrybuty:

- | | |
|---------------|---------------------------------------|
| • id | - identyfikator |
| • description | - opis typu dania w języku angielskim |

KAT/3/09 DishTypeTranslation (Tłumaczenie Typu Dania)

Opis: Tłumaczenie typu dania.

Atrybuty:

- | | |
|---------------|--|
| • id | - identyfikator |
| • translation | - przetłumaczona nazwa typu dania |
| • language | - język tłumaczenia w postaci kodu ISO 639-1 |

KAT/3/10 MealType (Typ Posiłku)

Opis: Typ posiłku, np. śniadanie lub obiad.

Atrybuty:

- | | |
|--------|--|
| • id | - identyfikator |
| • name | - nazwa typu posiłku w języku angielskim |

KAT/3/11 MealTypeTranslation (Tłumaczenie Typu Posiłku)

Opis: Meal type translation.

Atrybuty:

- | | |
|---------------|--|
| • id | - identyfikator |
| • translation | - przetłumaczona nazwa typu posiłku |
| • language | - język tłumaczenia w postaci kodu ISO 639-1 |

Reguły funkcjonowania

Reguły dla KAT/3/01 Recipe

REG/3/01 Przepis (**KAT/3/01 Recipe**) nie musi mieć zdefiniowanego autora (**KAT/1/01 User**).

REG/3/02 Przepis (**KAT/3/01 Recipe**) może mieć maksymalnie jednego autora (**KAT/1/01 User**).

REG/3/03 Przepis (**KAT/3/01 Recipe**) musi mieć dokładnie jedne podstawowe wartości odżywcze przepisu (**KAT/3/02 RecipeBasicNutritionData**).

REG/3/04 Przepis (**KAT/3/01 Recipe**) musi mieć przynajmniej jedną sekcję (**KAT/3/03 RecipeSection**).

- REG/3/05** Przepis (**KAT/3/01 Recipe**) może mieć wiele sekcji (**KAT/3/03 RecipeSection**).
- REG/3/06** Przepis (**KAT/3/01 Recipe**) nie musi mieć przypisanego żadnego sprzętu kuchennego (**KAT/3/06 KitchenAppliance**).
- REG/3/07** Przepis (**KAT/3/01 Recipe**) może mieć przypisanych wiele sprzętów kuchennych (**KAT/3/06 KitchenAppliance**).
- REG/3/08** Przepis (**KAT/3/01 Recipe**) nie musi mieć przypisanego żadnego typu dania (**KAT/3/08 DishType**).
- REG/3/09** Przepis (**KAT/3/01 Recipe**) może mieć przypisanych wiele typów dań (**KAT/3/08 DishType**).
- REG/3/10** Przepis (**KAT/3/01 Recipe**) nie musi mieć przypisanego żadnego typu posiłku (**KAT/3/10 MealType**).
- REG/3/11** Przepis (**KAT/3/01 Recipe**) może mieć przypisanych wiele typów posiłków (**KAT/3/10 MealType**).
- REG/3/12** Przepis (**KAT/3/01 Recipe**) nie musi mieć przypisanego żadnego odpowiedniego typu diety (**KAT/2/10 DietType**).
- REG/3/13** Przepis (**KAT/3/01 Recipe**) może mieć przypisanych wiele odpowiednich typów diety (**KAT/2/10 DietType**).
- REG/3/14** Przepis (**KAT/3/01 Recipe**) nie musi mieć przypisanego żadnego nieodpowiedniego typu diety (**KAT/2/10 DietType**).
- REG/3/15** Przepis (**KAT/3/01 Recipe**) może mieć przypisanych wiele nieodpowiednich typów diety (**KAT/2/10 DietType**).
- REG/3/16** *Dietetyk* może wyświetlać publiczne przepisy (**KAT/3/01 Recipe**).
- REG/3/17** *Dietetyk* może dodawać, wyświetlać, edytować i usuwać własne przepisy (**KAT/3/01 Recipe**).
- REG/3/18** *Administrator* może wyświetlać i usuwać przepisy (**KAT/3/01 Recipe**).
- REG/3/19** Przepis (**KAT/3/01 Recipe**) oznaczony jako ostateczny nie może być edytowany ani usuwany.

Reguły dla KAT/3/02 RecipeBasicNutritionData

- REG/3/20** Podstawowe wartości odżywcze przepisu (**KAT/3/02 RecipeBasicNutritionData**) muszą być przypisane do dokładnie jednego przepisu (**KAT/3/01 Recipe**).
- REG/3/21** Podstawowe wartości odżywcze przepisu (**KAT/3/02 RecipeBasicNutritionData**) są przedmiotem kompozycji ze strony przepisu (**KAT/3/01 Recipe**).

Reguły dla KAT/3/03 RecipeSection

- REG/3/22** Sekcja przepisu (**KAT/3/03 RecipeSection**) musi być przypisana do dokładniej jednego przepisu (**KAT/3/01 Recipe**).
- REG/3/23** Sekcja przepisu (**KAT/3/03 RecipeSection**) musi mieć przypisaną przynajmniej jedną porcję produktu (**KAT/3/04 ProductPortion**).
- REG/3/24** Sekcja przepisu (**KAT/3/03 RecipeSection**) może mieć przypisanych wiele porcji produktu (**KAT/3/04 ProductPortion**).
- REG/3/25** Sekcja przepisu (**KAT/3/03 RecipeSection**) musi mieć przypisany przynajmniej jeden krok przygotowania (**KAT/3/05 PreparationStep**).

- REG/3/26** Sekcja przepisu (**KAT/3/03 RecipeSection**) może mieć zdefiniowanych wiele kroków przygotowania (**KAT/3/05 PreparationStep**).
REG/3/27 Sekcja przepisu (**KAT/3/03 RecipeSection**) jest przedmiotem kompozycji ze strony przepisu (**KAT/3/01 Recipe**).

Reguły dla KAT/3/04 ProductPortion

- REG/3/28** Porcja produktu (**KAT/3/04 ProductPortion**) musi być przypisana do dokładnie jednej sekcji przepisu (**KAT/3/03 RecipeSection**).
REG/3/29 Porcja produktu (**KAT/3/04 ProductPortion**) musi mieć przypisany dokładnie jeden produkt (**KAT/2/01 Product**).
REG/3/30 Porcja produktu (**KAT/3/04 ProductPortion**) nie musi mieć przypisanej miary domowej (**KAT/2/06 HouseholdMeasure**).
REG/3/31 Porcja produktu (**KAT/3/04 ProductPortion**) może mieć przypisaną maksymalnie jedną miarę domową (**KAT/2/06 HouseholdMeasure**).
REG/3/32 Porcja produktu (**KAT/3/04 ProductPortion**) jest przedmiotem kompozycji ze strony sekcji przepisu (**KAT/3/03 RecipeSection**).

Reguły dla KAT/3/05 PreparationStep

- REG/3/33** Krok przygotowania (**KAT/3/05 PreparationStep**) musi być przypisany do dokładnie jednej sekcji przepisu (**KAT/3/03 RecipeSection**).
REG/3/34 Krok przygotowania (**KAT/3/05 PreparationStep**) jest przedmiotem kompozycji ze strony sekcji przepisu (**KAT/3/03 RecipeSection**).

Reguły dla KAT/3/06 KitchenAppliance

- REG/3/35** Sprzęt kuchenny (**KAT/3/06 KitchenAppliance**) nie musi mieć zdefiniowanego żadnego tłumaczenia (**KAT/3/07 KitchenApplianceTranslation**).
REG/3/36 Sprzęt kuchenny (**KAT/3/06 KitchenAppliance**) może mieć zdefiniowanych wiele tłumaczeń (**KAT/3/07 KitchenApplianceTranslation**).
REG/3/37 *Pacjent* może wyświetlać sprzęt kuchenny (**KAT/3/06 KitchenAppliance**).
REG/3/38 *Dietetyk* może wyświetlać sprzęt kuchenny (**KAT/3/06 KitchenAppliance**).
REG/3/39 *Administrator* może dodawać, wyświetlać, edytować i usuwać sprzęt kuchenny (**KAT/3/06 KitchenAppliance**).

Reguły dla KAT/3/07 KitchenApplianceTranslation

- REG/3/40** Tłumaczenie sprzętu kuchennego (**KAT/3/07 KitchenApplianceTranslation**) musi być przypisane do dokładnie jednego sprzętu kuchennego (**KAT/3/06 KitchenAppliance**).
REG/3/41 Tłumaczenie sprzętu kuchennego (**KAT/3/07 KitchenApplianceTranslation**) jest przedmiotem kompozycji ze strony sprzętu kuchennego (**KAT/3/06 KitchenAppliance**).

Reguły dla KAT/3/08 DishType

- REG/3/42** Typ dania (**KAT/3/08 DishType**) nie musi mieć zdefiniowanego żadnego tłumaczenia (**KAT/3/09 DishTypeTranslation**).

REG/3/43 Typ dania (**KAT/3/08 DishType**) może mieć zdefiniowanych wiele tłumaczeń (**KAT/3/09 DishTypeTranslation**).

REG/3/44 *Dietetyk* może wyświetlać typ dania (**KAT/3/08 DishType**).

REG/3/45 *Administrator* może dodawać, wyświetlać, edytować i usuwać typ dania (**KAT/3/08 DishType**).

Reguły dla KAT/3/09 DishTypeTranslation

REG/3/46 Tłumaczenie typu dania (**KAT/3/09 DishTypeTranslation**) musi być przypisane do dokładnie jednego typu dania (**KAT/3/08 DishType**).

REG/3/47 Tłumaczenie typu dania (**KAT/3/09 DishTypeTranslation**) jest przedmiotem kompozycji ze strony typu dania (**KAT/3/08 DishType**).

Reguły dla KAT/3/10 MealType

REG/3/48 Typ posiłku (**KAT/3/10 MealType**) nie musi mieć zdefiniowanego żadnego tłumaczenia (**KAT/3/11 MealTypeTranslation**).

REG/3/49 Typ posiłku (**KAT/3/10 MealType**) może mieć zdefiniowanych wiele tłumaczeń (**KAT/3/11 MealTypeTranslation**).

REG/3/50 *Pacjent* może wyświetlać typ posiłku (**KAT/3/10 MealType**).

REG/3/51 *Dietetyk* może wyświetlać typ posiłku (**KAT/3/10 MealType**).

REG/3/52 *Administrator* może dodawać, wyświetlać, edytować i usuwać typ posiłku (**KAT/3/10 MealType**).

Reguły dla KAT/3/11 MealTypeTranslation

REG/3/53 Tłumaczenie typu posiłku (**KAT/3/11 MealTypeTranslation**) musi być przypisane do dokładnie jednego typu posiłku (**KAT/3/10 MealType**).

REG/3/54 Tłumaczenie typu posiłku (**KAT/3/11 MealTypeTranslation**) jest przedmiotem kompozycji ze strony typu posiłku (**KAT/3/10 MealType**).

Ograniczenia dziedzinowe

Ograniczenia dla KAT/3/01 Recipe

OGR/3/01 Atrybut **name** jest wymagany.

OGR/3/02 Atrybut **preparationTimeMinutes** jest wymagany.

OGR/3/03 Atrybut **numberOfPortions** jest wymagany.

OGR/3/04 Atrybut **totalGramsWeight** jest wymagany.

OGR/3/05 Atrybut **isFinal** jest wymagany.

OGR/3/06 Atrybut **editTimestamp** jest stemplem czasowym.

OGR/3/07 Atrybut **name** jest ciągiem znaków o długości od 1 do 255 znaków.

OGR/3/08 Atrybut **preparationTimeMinutes** jest liczbą całkowitą nie mniejszą niż 0.

OGR/3/09 Atrybut **numberOfPortions** jest liczbą rzeczywistą nie mniejszą niż 0.

OGR/3/10 Atrybut **image** jest zdjęciem o maksymalnym rozmiarze 5000000 bajtów.

OGR/3/11 Atrybut **totalGramsWeight** jest liczbą rzeczywistą nie mniejszą niż 0.

OGR/3/12 Atrybut **isFinal** jest typu logicznego.

Ograniczenia dla KAT/3/02 RecipeBasicNutritionData

OGR/3/13 Atrybut **energy** jest wymagany.

- OGR/3/14** Atrybut **protein** jest wymagany.
- OGR/3/15** Atrybut **fat** jest wymagany.
- OGR/3/16** Atrybut **carbohydrates** jest wymagany.
- OGR/3/17** Atrybut **energy** jest liczbą całkowitą nie mniejszą niż 0.
- OGR/3/18** Atrybut **protein** jest liczbą całkowitą nie mniejszą niż 0.
- OGR/3/19** Atrybut **fat** jest liczbą całkowitą nie mniejszą niż 0.
- OGR/3/20** Atrybut **carbohydrates** jest liczbą całkowitą nie mniejszą niż 0.

Ograniczenia dla KAT/3/03 RecipeSection

- OGR/3/21** Atrybut **sectionName** jest ciągiem znaków o długości od 1 do 255 znaków.

Ograniczenia dla KAT/3/04 ProductPortion

- OGR/3/22** Atrybut **amount** jest wymagany.
- OGR/3/23** Atrybut **amount** jest liczbą rzeczywistą nie mniejszą niż 0.

Ograniczenia dla KAT/3/05 PreparationStep

- OGR/3/24** Atrybut **ordinalNumber** jest wymagany.
- OGR/3/25** Atrybut **ordinalNumber** jest liczbą całkowitą nie mniejszą niż 1.
- OGR/3/26** Atrybut **stepDescription** jest ciągiem znaków.

Ograniczenia dla KAT/3/06 KitchenAppliance

- OGR/3/27** Atrybut **name** jest wymagany.
- OGR/3/28** Atrybut **name** ma unikalną wartość.
- OGR/3/29** Atrybut **name** jest ciągiem znaków o długości od 1 do 255 znaków.

Ograniczenia dla KAT/3/07 KitchenApplianceTranslation

- OGR/3/30** Atrybut **translation** jest wymagany.
- OGR/3/31** Atrybut **translation** jest ciągiem znaków o długości od 1 do 255 znaków.

Ograniczenia dla KAT/3/08 DishType

- OGR/3/32** Atrybut **description** jest wymagany.
- OGR/3/33** Atrybut **description** ma unikalną wartość.
- OGR/3/34** Atrybut **description** jest ciągiem znaków o długości od 1 do 255 znaków.

Ograniczenia dla KAT/3/09 DishTypeTranslation

- OGR/3/35** Atrybut **translation** jest wymagany.
- OGR/3/36** Atrybut **translation** jest ciągiem znaków o długości od 1 do 255 znaków.

Ograniczenia dla KAT/3/10 MealType

- OGR/3/37** Atrybut **name** jest wymagany.
- OGR/3/38** Atrybut **name** ma unikalną wartość.
- OGR/3/39** Atrybut **name** jest ciągiem znaków o długości od 1 do 255 znaków.

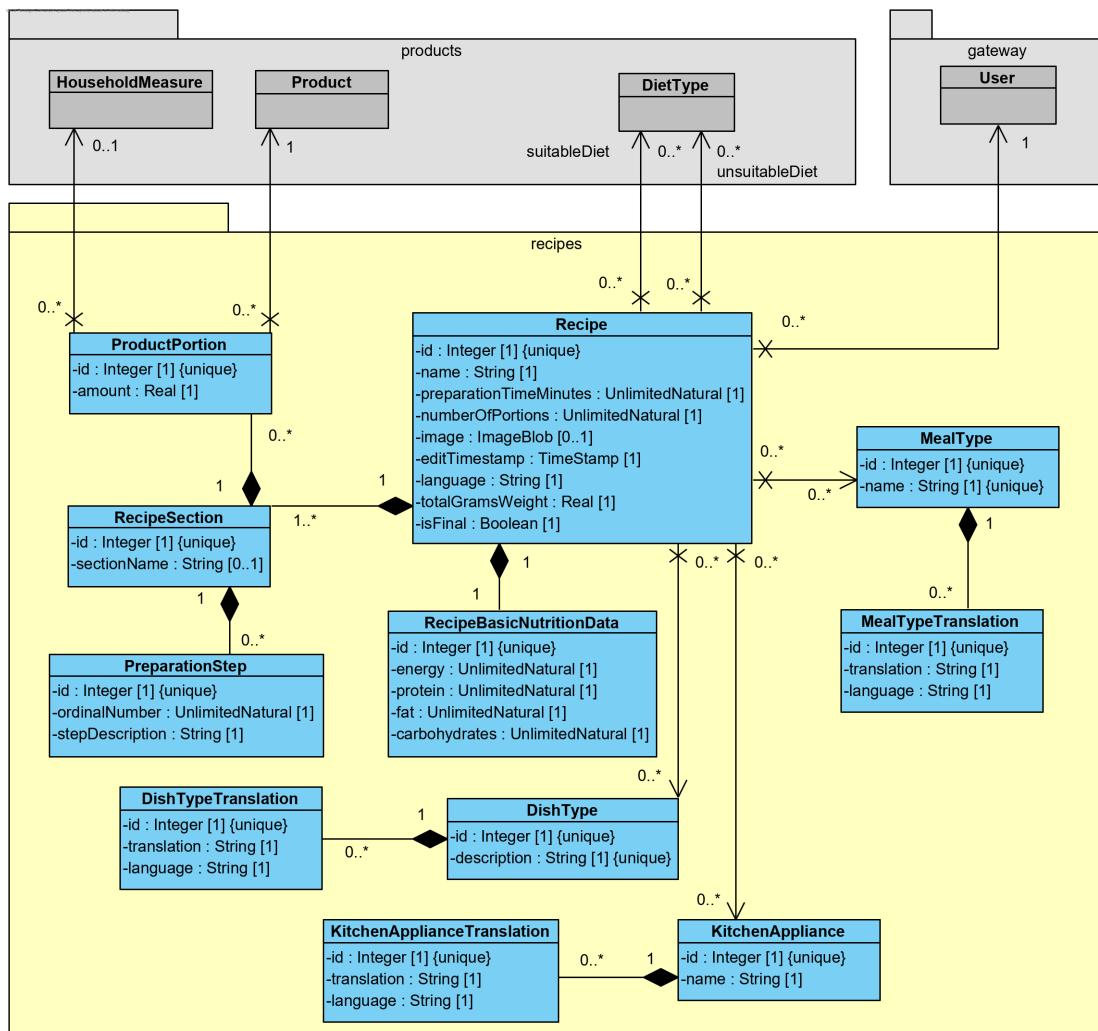
Ograniczenia dla KAT/3/11 MealTypeTranslation

OGR/3/40 Atrybut **translation** jest wymagany.

OGR/3/41 Atrybut **translation** jest ciągiem znaków o długości od 1 do 255 znaków.

Model informacyjny

Na rysunku 3.29 przedstawiono model informacyjny dla poddziedziny przepisy w formie diagramu klas języka UML.



Rysunek 3.29: Diagram klas - poddziedzina przepisy (źródło: opr. wł.)

3.3.5. Poddziedzina jadłospisy

Kategorie

KAT/4/01 MealPlan (Jadłospis)

Opis: Jadłospis; plan posiłków z podziałem na dnie i posiłki.

Atrybuty:

• id	- identyfikator
• creationTimestamp	- czas utworzenia jadłospisu
• editTimestamp	- czas ostatniej edycji jadłospisu
• name	- nazwa jadłospisu
• isFinal	- flaga określająca czy jadłospis jest edytowalny
• language	- język tłumaczenia w postaci kodu ISO 639-1
• numberOfWorkdays	- liczba dni planu
• numberOfMealsPerDay	- liczba posiłków w ciągu dnia
• totalDailyEnergy	- całkowita oczekiwana energia w ciągu dnia w kaloriach
• percentOfProtein	- procent białka w całkowitej dziennej energii
• percentOfFat	- procent tłuszcza w całkowitej dziennej energii
• percentOfCarbohydrates	- procent węglowodanów w całkowitej dziennej energii

KAT/4/02 MealPlanDay (Dzień Jadłospisu)

Opis: Dzień w jadłospisie.

Atrybuty:

• id	- identyfikator
• ordinalNumber	- numer porządkowy dnia

KAT/4/03 Meal (Posiłek)

Opis: Posiłek.

Atrybuty:

• id	- identyfikator
• ordinalNumber	- numer porządkowy posiłku

KAT/4/04 MealRecipe (Przepis Posiłku)

Opis: Przepis przypisany do posiłku.

Atrybuty:

• id	- identyfikator
• amount	- ilość przepisu w gramach

KAT/4/05 MealProduct (Produkt Posiłku)

Opis: Produkt przypisany do posiłku.

Atrybuty:

• id	- identyfikator
• amount	- ilość produktu w jednostkach miary domowej lub w gramach jeśli miara domowa nie jest zdefiniowana

KAT/4/06 MealDefinition (Definicja Posiłku)

Opis: Definicja posiłku wykorzystywana do określenia właściwości każdego posiłku w ciągu dnia.

Atrybuty:

• id	- identyfikator
• ordinalNumber	- dzienny numer porządkowy posiłku

- timeOfMeal
 - typowy czas posiłku w formacie 24h w postaci: HH:mm
- percentOfEnergy
 - część dziennej całkowitej dziennej energii w procentach

Reguły funkcjonowania

Reguły dla KAT/4/01 MealPlan

- REG/4/01** Jadłospis (**KAT/4/01 MealPlan**) musi mieć przypisany przynajmniej jeden dzień (**KAT/4/02 MealPlanDay**).
- REG/4/02** Jadłospis (**KAT/4/01 MealPlan**) może mieć przypisanych maksymalnie 31 dni (**KAT/4/02 MealPlanDay**).
- REG/4/03** Jadłospis (**KAT/4/01 MealPlan**) musi mieć przypisaną przynajmniej jedną definicję posiłku (**KAT/4/06 MealDefinition**).
- REG/4/04** Jadłospis (**KAT/4/01 MealPlan**) może mieć przypisanych maksymalnie 10 definicji posiłków (**KAT/4/06 MealDefinition**).
- REG/4/05** Jadłospis (**KAT/4/01 MealPlan**) nie musi mieć przypisanego żadnego odpowiedniego typu diety (**KAT/2/10 DietType**).
- REG/4/06** Jadłospis (**KAT/4/01 MealPlan**) może mieć przypisanych wiele odpowiednich typów diety (**KAT/2/10 DietType**).
- REG/4/07** Jadłospis (**KAT/4/01 MealPlan**) nie musi mieć przypisanego żadnego nieodpowiedniego typu diety (**KAT/2/10 DietType**).
- REG/4/08** Jadłospis (**KAT/4/01 MealPlan**) może mieć przypisanych wiele nieodpowiednich typów diety (**KAT/2/10 DietType**).
- REG/4/09** Jadłospis (**KAT/4/01 MealPlan**) musi mieć dokładnie jednego autora (**KAT/1/01 User**).
- REG/4/10** Dietetyk może wyświetlać publiczne jadłospisy (**KAT/4/01 MealPlan**).
- REG/4/11** Dietetyk może dodawać, wyświetlać, edytować i usuwać własne jadłospisy (**KAT/4/01 MealPlan**).
- REG/4/12** Administrator może wyświetlać i usuwać jadłospisy (**KAT/4/01 MealPlan**).

Reguły dla KAT/4/02 MealPlanDay

- REG/4/13** Dzień jadłospisu (**KAT/4/02 MealPlanDay**) musi być przypisany do dokładnie jednego jadłospisu (**KAT/4/01 MealPlan**).
- REG/4/14** Dzień jadłospisu (**KAT/4/02 MealPlanDay**) nie musi mieć przypisanego żadnego posiłku (**KAT/4/03 Meal**).
- REG/4/15** Dzień jadłospisu (**KAT/4/02 MealPlanDay**) może mieć przypisanych maksymalnie 10 posiłków (**KAT/4/03 Meal**).
- REG/4/16** Dzień jadłospisu (**KAT/4/02 MealPlanDay**) jest przedmiotem kompozycji ze strony jadłospisu (**KAT/4/01 MealPlan**).

Reguły dla KAT/4/03 Meal

- REG/4/17** Posiłek (**KAT/4/03 Meal**) musi być przypisany do dokładnie jednego dnia jadłospisu (**KAT/4/02 MealPlanDay**).
- REG/4/18** Posiłek (**KAT/4/03 Meal**) nie musi mieć przypisanego żadnego produktu (**KAT/4/05 MealProduct**).

- REG/4/19** Posiłek (**KAT/4/03 Meal**) może mieć przypisanych wiele produktów (**KAT/4/05 MealProduct**).
- REG/4/20** Posiłek (**KAT/4/03 Meal**) nie musi mieć przypisanego żadnego przepisu (**KAT/4/04 MealRecipe**).
- REG/4/21** Posiłek (**KAT/4/03 Meal**) może mieć przypisanych wiele przepisów (**KAT/4/04 MealRecipe**).
- REG/4/22** Posiłek (**KAT/4/03 Meal**) jest przedmiotem kompozycji ze strony dnia jadłospisu (**KAT/4/02 MealPlanDay**).

Reguły dla **KAT/4/04 MealRecipe**

- REG/4/23** Przepis posiłku (**KAT/4/04 MealRecipe**) musi być przypisany do dokładnie jednego posiłku (**KAT/4/03 Meal**).
- REG/4/24** Przepis posiłku (**KAT/4/04 MealRecipe**) musi mieć przypisany dokładnie jeden przepis (**KAT/3/01 Recipe**).
- REG/4/25** Przepis posiłku (**KAT/4/04 MealRecipe**) jest przedmiotem kompozycji ze strony posiłku (**KAT/4/03 Meal**).

Reguły dla **KAT/4/05 MealProduct**

- REG/4/26** Produkt posiłku (**KAT/4/05 MealProduct**) musi być przypisany do dokładnie jednego posiłku (**KAT/4/03 Meal**).
- REG/4/27** Produkt posiłku (**KAT/4/05 MealProduct**) musi mieć przypisany dokładnie jeden produkt (**KAT/2/01 Product**).
- REG/4/28** Produkt posiłku (**KAT/4/05 MealProduct**) nie musi mieć przypisanej żadnej miary domowej (**KAT/2/06 HouseholdMeasure**).
- REG/4/29** Produkt posiłku (**KAT/4/05 MealProduct**) musi mieć przypisaną maksymalnie jedną miarę domową (**KAT/2/06 HouseholdMeasure**).
- REG/4/30** Produkt posiłku (**KAT/4/05 MealProduct**) jest przedmiotem kompozycji ze strony posiłku (**KAT/4/03 Meal**).

Reguły dla **KAT/4/06 MealDefinition**

- REG/4/31** Definicja posiłku (**KAT/4/06 MealDefinition**) musi być przypisana do dokładnie jednego jadłospisu (**KAT/4/01 MealPlan**).
- REG/4/32** Definicja posiłku (**KAT/4/06 MealDefinition**) musi mieć przypisany dokładnie jeden typ posiłku (**KAT/3/10 MealType**).
- REG/4/33** Definicja posiłku (**KAT/4/06 MealDefinition**) jest przedmiotem kompozycji ze strony jadłospisu (**KAT/4/01 MealPlan**).

Ograniczenia dziedzinowe

Ograniczenia dla **KAT/4/01 MealPlan**

- OGR/4/01** Atrybut **creationTimestamp** jest wymagany.
- OGR/4/02** Atrybut **editTimestamp** jest wymagany.
- OGR/4/03** Atrybut **isFinal** jest wymagany.
- OGR/4/04** Atrybut **numberOfDays** jest wymagany.
- OGR/4/05** Atrybut **numberOfMealsPerDay** jest wymagany.
- OGR/4/06** Atrybut **totalDailyEnergy** jest wymagany.
- OGR/4/07** Atrybut **percentOfProtein** jest wymagany.

- OGR/4/08** Atrybut **percentOfFat** jest wymagany.
- OGR/4/09** Atrybut **percentOfCarbohydrates** jest wymagany.
- OGR/4/10** Atrybut **creationTimestamp** jest stemplem czasowym.
- OGR/4/11** Atrybut **editTimestamp** jest stemplem czasowym.
- OGR/4/12** Atrybut **name** jest ciągiem znaków o długości od 1 do 255 znaków.
- OGR/4/13** Atrybut **isFinal** jest typu logicznego.
- OGR/4/14** Atrybut **numberOfDays** jest liczbą całkowitą nie mniejszą niż 1 i nie większą niż 30.
- OGR/4/15** Atrybut **numberOfMealsPerDay** jest liczbą całkowitą nie mniejszą niż 1 i nie większą niż 10.
- OGR/4/16** Atrybut **totalDailyEnergy** jest liczbą całkowitą nie mniejszą niż 1.
- OGR/4/17** Atrybut **percentOfProtein** jest liczbą całkowitą nie mniejszą niż 0 i nie większą niż 100.
- OGR/4/18** Atrybut **percentOfFat** jest liczbą całkowitą nie mniejszą niż 0 i nie większą niż 100.
- OGR/4/19** Atrybut **percentOfCarbohydrates** jest liczbą całkowitą nie mniejszą niż 0 i nie większą niż 100.
- OGR/4/20** Suma wartości atrybutów **percentOfProtein**, **percentOfFat**, **percentOfCarbohydrates** nie może przekraczać 100

Ograniczenia dla KAT/4/02 MealPlanDay

- OGR/4/21** Atrybut **ordinalNumber** jest wymagany.
- OGR/4/22** Atrybut **ordinalNumber** jest liczbą całkowitą nie mniejszą niż 1.

Ograniczenia dla KAT/4/03 Meal

- OGR/4/23** Atrybut **ordinalNumber** jest wymagany.
- OGR/4/24** Atrybut **ordinalNumber** jest liczbą całkowitą nie mniejszą niż 1.

Ograniczenia dla KAT/4/04 MealRecipe

- OGR/4/25** Atrybut **amount** jest wymagany.
- OGR/4/26** Atrybut **amount** jest liczbą całkowitą nie mniejszą niż 0.

Ograniczenia dla KAT/4/05 MealProduct

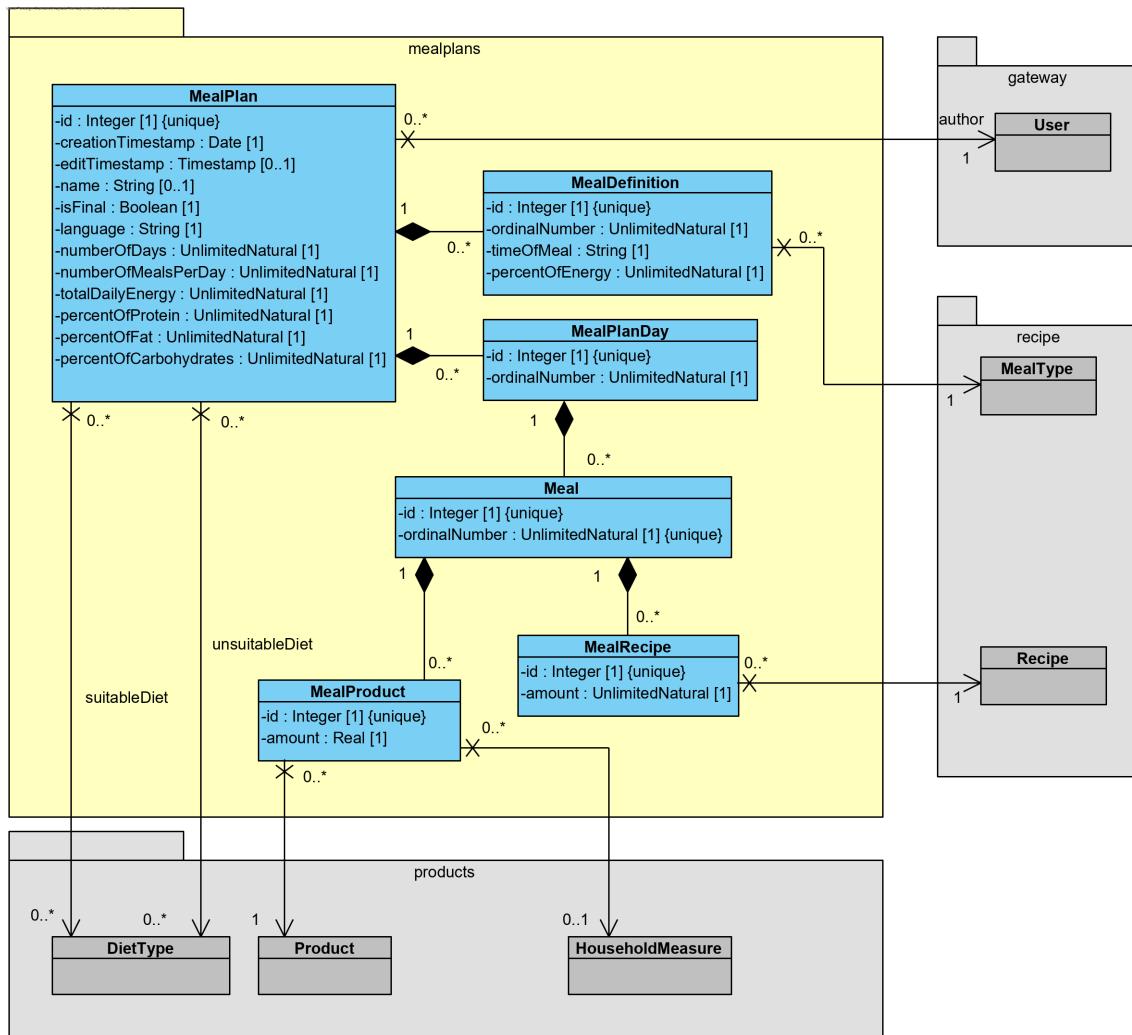
- OGR/4/27** Atrybut **amount** jest wymagany.
- OGR/4/28** Atrybut **amount** jest liczbą rzeczywistą nie mniejszą niż 0.

Ograniczenia dla KAT/4/06 MealDefinition

- OGR/4/29** Atrybut **ordinalNumber** jest wymagany.
- OGR/4/30** Atrybut **timeOfMeal** jest wymagany.
- OGR/4/31** Atrybut **percentOfEnergy** jest wymagany.
- OGR/4/32** Atrybut **ordinalNumber** jest liczbą całkowitą nie mniejszą niż 1.
- OGR/4/33** Atrybut **timeOfMeal** jest ciągiem znaków w postaci HH:MI.
- OGR/4/34** Atrybut **percentOfEnergy** jest liczbą całkowitą nie mniejszą niż 0 i nie większą niż 100.
- OGR/4/35** Suma wartości wszystkich atrybutów **percentOfEnergy** w jednym jadłospisie musi być równa 100

Model informacyjny

Na rysunku 3.30 przedstawiono model informacyjny dla poddziedziny jadłospisy w formie diagramu klas języka UML.



Rysunek 3.30: Diagram klas - poddziedzina jadłospisy (źródło: opr. wł.)

3.3.6. Poddziedzina wizyty

Kategorie

KAT/5/01 Appointment (Wizyta)

Opis: Wizyta dietetyczna.

Atrybuty:

- | | |
|--------------------|----------------------------|
| • id | - identyfikator |
| • appointmentDate | - data i godzina wizyty |
| • appointmentState | - stan wizyty |
| • generalAdvice | - ogólna porada po wizycie |

KAT/5/02 PatientCard (Karta Pacjenta)

Opis: Karta pacjenta.

Atrybuty:

- | | |
|-------------------------|--|
| • id | - identyfikator |
| • creationDate | - data rejestracji pacjenta do dietetyka |
| • patientLastName | - nazwisko pacjenta |
| • patientFirstName | - imię pacjenta |
| • patientGender | - płeć pacjenta |
| • patientEmail | - adres mailowy pacjenta |
| • patientPhone | - numer telefonu pacjenta |
| • additionalPatientInfo | - dodatkowe informacje pacjenta |

KAT/5/03 BodyMeasurement (Pomiar Ciała)

Opis: Pomiar ciała.

Atrybuty:

- | | |
|----------------------|---|
| • id | - identyfikator |
| • completionDate | - data przeprowadzenia pomiaru |
| • height | - wzrost pacjenta; razem z wagą wykorzystywany do obliczania współczynnika BMI |
| • weight | - waga pacjenta; razem z wzrostem wykorzystywany do obliczania współczynnika BMI |
| • waist | - obwód pasa pacjenta |
| • percentOfFatTissue | - procent tkanki tłuszczowej w ciele pacjenta; norma dla kobiet: 16-20; norma dla mężczyzn: 15-18 |
| • percentOfWater | - procent wody w ciele pacjenta; norma dla kobiet: 45-60; norma dla mężczyzn: 50-65 |
| • muscleMass | - masa tkanki mięśniowej w ciele pacjenta w kilogramach |
| • physicalMark | - ocena fizyczna; norma: 5 |
| • calciumInBones | - poziom wapnia w kościach pacjenta w kilogramach; norma: 2.4kg |
| • basicMetabolism | - podstawowy metabolizm w kilokaloriach |
| • metabolicAge | - wiek metaboliczny w latach |
| • visceralFatLevel | - poziom tłuszcza trzewnego; norma: 1-12 |

KAT/5/04 NutritionalInterview (Wywiad Żywieniowy)

Opis: Wywiad żywieniowy.

Atrybuty:

- | | |
|--------------------|--|
| • id | - identyfikator |
| • completionDate | - czas przeprowadzenia wywiadu |
| • targetWeight | - docelowa waga pacjenta w kilogramach |
| • advicePurpose | - cel wizyty podsumowujący co pacjent pragnie osiągnąć poprzez terapię dietetyczną |
| • physicalActivity | - poziom aktywności fizycznej pacjenta |
| • diseases | - choroby pacjenta |
| • medicines | - leki przyjmowane przez pacjenta |
| • jobType | - typ pracy pacjenta |

- | | |
|--|---|
| <ul style="list-style-type: none"> • likedProducts • dislikedProducts • foodAllergies • foodIntolerances | <ul style="list-style-type: none"> - produkty spożywcze, które pacjent lubi - produkty spożywcze, których pacjent nie lubi - produkty spożywcze na które pacjent jest uczulony - nietolerancje pokarmowe pacjenta |
|--|---|

KAT/5/05 CustomNutritionalInterviewQuestion (Niestandardowe Pytanie Wywiadu Żywieniowego)

Opis: Niestandardowe pytanie wywiadu żywieniowego.

Atrybuty:

- | | |
|---|--|
| <ul style="list-style-type: none"> • id • ordinalNumber • question • answer | <ul style="list-style-type: none"> - identyfikator - numer porządkowy pytania - pytanie - odpowiedź na pytanie |
|---|--|

KAT/5/06 AssignedMealPlan (Przypisany Jadłospis)

Opis: Przypisany jadłospis.

Atrybuty:

- | | |
|---|--|
| <ul style="list-style-type: none"> • id • assigmentTime | <ul style="list-style-type: none"> - identyfikator - czas przypisania jadłospisu |
|---|--|

Reguły funkcjonowania

Reguły dla KAT/5/01 Appointment

- REG/5/01** Wizyta (**KAT/5/01 Appointment**) musi być przypisana do dokładnie jednej karty pacjenta (**KAT/5/02 PatientCard**).
- REG/5/02** Wizyta (**KAT/5/01 Appointment**) nie musi mieć przypisanego żadnych pomiarów ciała (**KAT/5/03 BodyMeasurement**).
- REG/5/03** Wizyta (**KAT/5/01 Appointment**) może mieć przypisane maksymalnie jedne pomiary ciała (**KAT/5/03 BodyMeasurement**).
- REG/5/04** Wizyta (**KAT/5/01 Appointment**) nie musi mieć przypisanego żadnego wywiadu żywieniowego (**KAT/5/04 NutritionalInterview**).
- REG/5/05** Wizyta (**KAT/5/01 Appointment**) może mieć przypisany maksymalnie jeden wywiad żywieniowy (**KAT/5/04 NutritionalInterview**).
- REG/5/06** Wizyta (**KAT/5/01 Appointment**) nie musi mieć przypisanego żadnego jadłospisu (**KAT/5/06 AssignedMealPlan**).
- REG/5/07** Wizyta (**KAT/5/01 Appointment**) może mieć przypisanych wiele jadłospisów (**KAT/5/06 AssignedMealPlan**).
- REG/5/08** *Pacjent* może wyświetlać swoją wizytę (**KAT/5/01 Appointment**).
- REG/5/09** *Dietetyk* może dodawać nową wizytę (**KAT/5/01 Appointment**).
- REG/5/10** *Dietetyk* może wyświetlać i edytować swoją wizytę (**KAT/5/01 Appointment**).

Reguły dla KAT/5/02 PatientCard

- REG/5/11** Karta pacjenta (**KAT/5/02 PatientCard**) nie musi mieć przypisanej żadnej wizyty (**KAT/5/01 Appointment**).
- REG/5/12** Karta pacjenta (**KAT/5/02 PatientCard**) może mieć przypisanych wiele wizyt (**KAT/5/01 Appointment**).

- REG/5/13** Karta pacjenta (**KAT/5/02 PatientCard**) nie musi mieć przypisanego żadnego konta pacjenta (**KAT/1/01 User**).
- REG/5/14** Karta pacjenta (**KAT/5/02 PatientCard**) może mieć przypisaną maksymalnie jedno konto pacjenta (**KAT/1/01 User**).
- REG/5/15** Karta pacjenta (**KAT/5/02 PatientCard**) musi mieć przypisanego dokładnie jednego dietetyka (**KAT/1/01 User**).
- REG/5/16** *Pacjent* może wyświetlać swoją kartę pacjenta (**KAT/5/02 PatientCard**).
- REG/5/17** *Dietetyk* może dodawać nową kartę pacjenta (**KAT/5/02 PatientCard**).
- REG/5/18** *Dietetyk* może wyświetlać i edytować karty pacjenta (**KAT/5/02 PatientCard**), którymi zarządza.

Reguły dla KAT/5/03 BodyMeasurement

- REG/5/19** Pomiary ciała (**KAT/5/03 BodyMeasurement**) muszą być przypisane do dokładnie jednej wizyty (**KAT/5/01 Appointment**).
- REG/5/20** Pomiary ciała (**KAT/5/03 BodyMeasurement**) są przedmiotem kompozycji ze strony wizyty (**KAT/5/01 Appointment**).

Reguły dla KAT/5/04 NutritionalInterview

- REG/5/21** Wywiad żywieniowy (**KAT/5/04 NutritionalInterview**) musi być przypisany do dokładnie jednej wizyty (**KAT/5/01 Appointment**).
- REG/5/22** Wywiad żywieniowy (**KAT/5/04 NutritionalInterview**) nie musi mieć przypisanego żadnego niestandardowego pytania (**KAT/5/05 CustomNutritionalInterviewQuestion**).
- REG/5/23** Wywiad żywieniowy (**KAT/5/04 NutritionalInterview**) może mieć przypisanych wiele niestandardowych pytań (**KAT/5/05 CustomNutritionalInterviewQuestion**).
- REG/5/24** Wywiad żywieniowy (**KAT/5/04 NutritionalInterview**) nie musi mieć przypisanych żadnych posiadanych sprzętów kuchennych (**KAT/3/06 KitchenAppliance**).
- REG/5/25** Wywiad żywieniowy (**KAT/5/04 NutritionalInterview**) może mieć przypisanych wiele posiadanych sprzętów kuchennych (**KAT/3/06 KitchenAppliance**).
- REG/5/26** Wywiad żywieniowy (**KAT/5/04 NutritionalInterview**) jest przedmiotem kompozycji ze strony wizyty (**KAT/5/01 Appointment**).

Reguły dla KAT/5/05 CustomNutritionalInterviewQuestion

- REG/5/27** Niestandardowe pytanie żywieniowe (**KAT/5/05 CustomNutritionalInterviewQuestion**) musi być przypisane do dokładnie jednego wywiadu żywieniowego (**KAT/5/04 NutritionalInterview**).
- REG/5/28** Niestandardowe pytanie żywieniowe (**KAT/5/05 CustomNutritionalInterviewQuestion**) jest przedmiotem kompozycji ze strony wywiadu żywieniowego (**KAT/5/04 NutritionalInterview**).

Reguły dla KAT/5/06 AssignedMealPlan

- REG/5/29** Przypisany jadłospis (**KAT/5/06 AssignedMealPlan**) musi mieć przydzieloną dokładnie jedną wizytę (**KAT/5/01 Appointment**).

- REG/5/30** Przypisany jadłospis (**KAT/5/06 AssignedMealPlan**) musi mieć przydzielony dokładnie jeden jadłospis (**KAT/4/01 MealPlan**).
- REG/5/31** Przypisany jadłospis (**KAT/5/06 AssignedMealPlan**) jest przedmiotem kompozycji ze strony wizyty (**KAT/5/01 Appointment**).

Ograniczenia dziedzinowe

Ograniczenia dla **KAT/5/01 Appointment**

- OGR/5/01** Atrybut **appointmentDate** jest wymagany.
- OGR/5/02** Atrybut **appointmentState** jest wymagany.
- OGR/5/03** Atrybut **appointmentDate** jest stemplem czasowym.
- OGR/5/04** Atrybut **appointmentState** jest typu wyliczeniowego i może przyjmować wartości "PLANNED", "CANCELED", "TOOK_PLACE", "COMPLETED".
- OGR/5/05** Atrybut **generalAdvice** jest ciągiem znaków.

Ograniczenia dla **KAT/5/02 PatientCard**

- OGR/5/06** Atrybut **creationDate** jest wymagany.
- OGR/5/07** Atrybut **patientLastName** jest wymagany.
- OGR/5/08** Atrybut **patientFirstName** jest wymagany.
- OGR/5/09** Atrybut **patientGender** jest wymagany.
- OGR/5/10** Atrybut **creationDate** jest stemplem czasowym.
- OGR/5/11** Atrybut **patientLastName** jest ciągiem znaków o długości od 1 do 255 znaków.
- OGR/5/12** Atrybut **patientFirstName** jest ciągiem znaków o długości od 1 do 255 znaków.
- OGR/5/13** Atrybut **patientGender** jest typu wyliczeniowego i może przyjmować wartości "FEMALE", "MALE", "OTHER".
- OGR/5/14** Atrybut **patientEmail**- jest ciągiem znaków o długości od 5 do 254 znaków.
- OGR/5/15** Atrybut **patientPhone**- jest ciągiem znaków o długości od 1 do 50 znaków.
- OGR/5/16** Atrybut **additionalPatientInfo** jest ciągiem znaków.

Ograniczenia dla **KAT/5/03 BodyMeasurement**

- OGR/5/17** Atrybut **completionDate** jest wymagany.
- OGR/5/18** Atrybut **height** jest wymagany.
- OGR/5/19** Atrybut **weight** jest wymagany.
- OGR/5/20** Atrybut **waist** jest wymagany.
- OGR/5/21** Atrybut **completionDate** jest stemplem czasowym.
- OGR/5/22** Atrybut **height** jest liczbą całkowitą.
- OGR/5/23** Atrybut **weight** jest liczbą całkowitą.
- OGR/5/24** Atrybut **waist** jest liczbą rzeczywistą.
- OGR/5/25** Atrybut **percentOfFatTissue** jest liczbą rzeczywistą nie mniejszą niż 0 i nie większą niż 100.
- OGR/5/26** Atrybut **percentOfWater** jest liczbą rzeczywistą nie mniejszą niż 0 i nie większą niż 100.
- OGR/5/27** Atrybut **muscleMass** jest liczbą rzeczywistą.
- OGR/5/28** Atrybut **physicalMark** jest liczbą rzeczywistą.

- OGR/5/29** Atrybut **calciumInBones** jest liczbą rzeczywistą.
- OGR/5/30** Atrybut **basicMetabolism** jest liczbą całkowitą.
- OGR/5/31** Atrybut **metabolicAge** jest liczbą rzeczywistą.
- OGR/5/32** Atrybut **visceralFatLevel** jest liczbą rzeczywistą.

Ograniczenia dla KAT/5/04 NutritionalInterview

- OGR/5/33** Atrybut **completionDate** jest wymagany.
- OGR/5/34** Atrybut **targetWeight** jest wymagany.
- OGR/5/35** Atrybut **advicePurpose** jest wymagany.
- OGR/5/36** Atrybut **physicalActivity** jest wymagany.
- OGR/5/37** Atrybut **completionDate** jest stemplem czasowym.
- OGR/5/38** Atrybut **targetWeight** jest liczbą całkowitą.
- OGR/5/39** Atrybut **advicePurpose** jest ciągiem znaków.
- OGR/5/40** Atrybut **physicalActivity** jest typu wyliczeniowego i może przyjmować wartości "EXTREMELY_INACTIVE", "SEDENTARY", "MODERATELY_ACTIVE", "VIGOROUSLY_ACTIVE", "EXTREMELY_ACTIVE".
- OGR/5/41** Atrybut **diseases** jest ciągiem znaków.
- OGR/5/42** Atrybut **medicines** jest ciągiem znaków.
- OGR/5/43** Atrybut **jobType** jest typu wyliczeniowego i może przyjmować wartości "SITTING", "STANDING", "MIXED".
- OGR/5/44** Atrybut **likedProducts** jest ciągiem znaków.
- OGR/5/45** Atrybut **dislikedProducts** jest ciągiem znaków.
- OGR/5/46** Atrybut **foodAllergies** jest ciągiem znaków.
- OGR/5/47** Atrybut **foodIntolerances** jest ciągiem znaków.

Ograniczenia dla KAT/5/05 CustomNutritionalInterviewQuestion

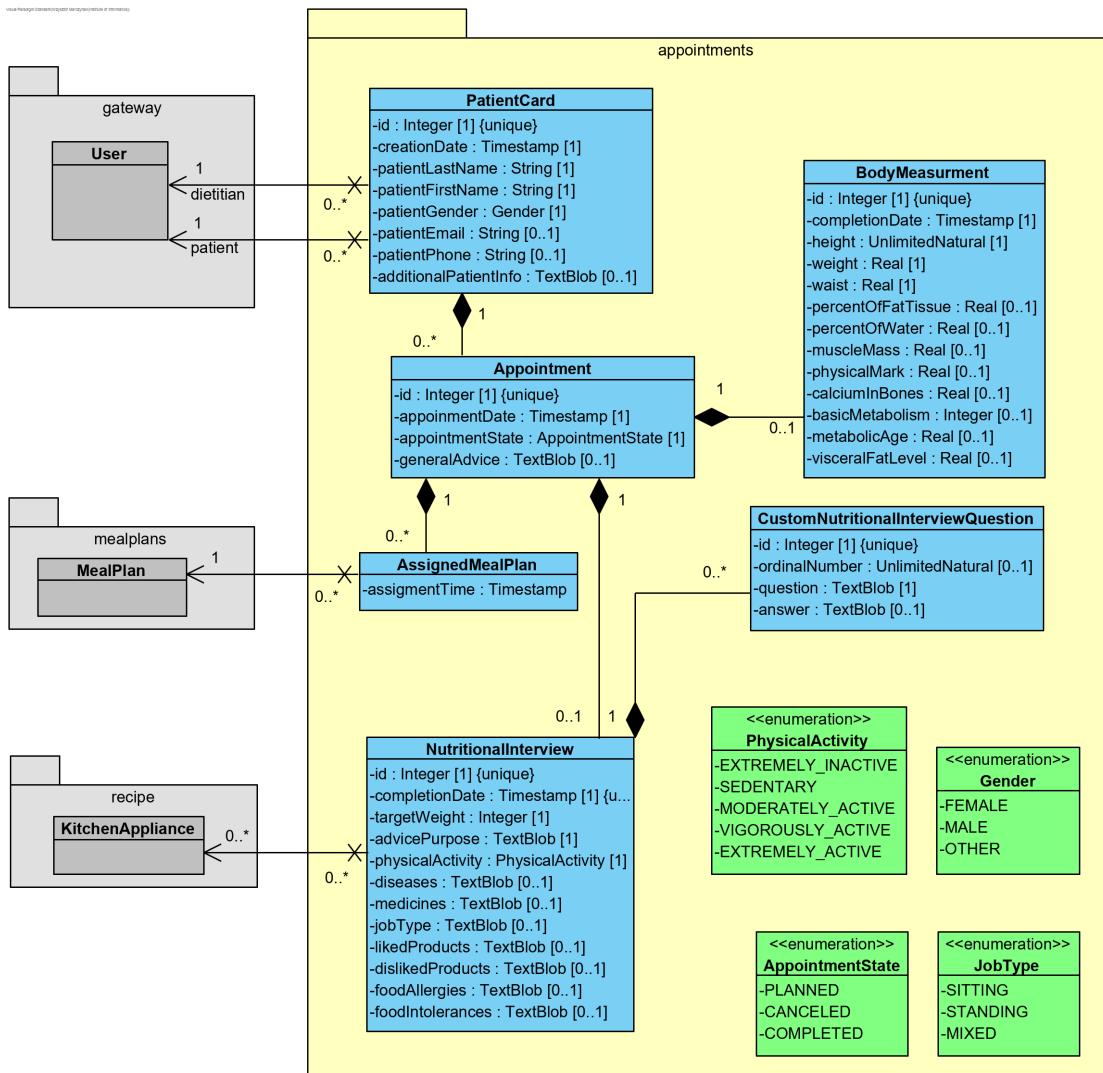
- OGR/5/48** Atrybut **question** jest wymagany.
- OGR/5/49** Atrybut **ordinalNumber** jest liczbą całkowitą nie mniejszą niż 1.
- OGR/5/50** Atrybut **question** jest ciągiem znaków.
- OGR/5/51** Atrybut **answer** jest ciągiem znaków.

Ograniczenia dla KAT/5/06 AssignedMealPlan

- OGR/5/52** Atrybut **assigmentTime** jest wymagany.
- OGR/5/53** Atrybut **assigmentTime** jest stemplem czasowym.

Model informacyjny

Na rysunku 3.31 przedstawiono model informacyjny dla poddziedziny wizyty w formie diagramu klas języka UML.



Rysunek 3.31: Diagram klas - poddziedzina wizyty (źródło: opr. wł.)

4. Implementacja

4.1. Wykorzystywane środowiska i narzędzia programistyczne

Podczas wyboru języków programowania, z użyciem których miał zostać zaimplementowany system, postawiono następujące kryteria:

- ścisła kontrola typów,
- dobre wsparcie dla paradymatu programowania obiektowego,
- niezależność języka od platformy,
- bogaty ekosystem.

Wybrane języki spełniające te kryteria to:

- w warstwie backendu Java[46] - opracowany przez Sun Microsystems język kompilowalny do kodu bajtowego, który jest wykonywany na maszynie wirtualnej,
- w warstwie frontendu Typescript[38] - opracowany przez Microsoft język otwartoźródłowy kompilowalny do języka JavaScript[69].

Powyższy wybór zaowocował decyzją o zastosowaniu Angulara[21] jako wiodącej frontendowej platformy programistycznej (ang. framework) i Springa[50] jako wiodącej backendowej platformy programistycznej. Wspomniane platformy cieszą się bardzo dużą popularnością, a ich dojrzałość sprawia, że znajdują zastosowanie zarówno w niewielkich aplikacjach jak i w systemach klasy enterprise.

Podczas projektu witryny internetowej kod, który jest wykonywany po stronie przeglądarki zwykle jest napisany w technologiach HTML, CSS[68] i JavaScript[69]. Jak już wspomniano zamiast języka JavaScript wykorzystano TypeScript, natomiast zamiast CSS postanowiono wykorzystać SASS[4], który rozszerza funkcjonalność CSS.

System został zaprojektowany tak, żeby wykorzystać cechy relacyjnych baz danych, więc podczas wyboru systemu zarządzania bazą danych pod uwagę wzięto tylko relacyjne bazy danych. Rozważano przede wszystkim systemy PostgreSQL[47] i MySQL[59]. Z punktu widzenia funkcjonalności potrzebnych w implementowanej aplikacji oba systemy systemy zarządzania relacyjną bazą danych (ang. Relational Database Management System - RDBMS) wypadają równie dobrze, jednakże ostatecznie wybrano PostgreSQL ze względu na mniej restrykcyjną licencję wykorzystania systemu nawet w rozwiązaniach komercyjnych o zamkniętym kodzie. Dodatkowo w celu implementacji ewolucyjnego projektowania bazy danych[15] postanowiono wykorzystać bibliotekę Liquibase[6] do zarządzania zmianami schematu bazy.

Do implementacji architektury mikroserwisów postanowiono wykorzystać stos technologii Netflix OSS[40]. Jest to dojrzały, prosty do wykorzystania w implementacji stos technologii mikroserwisowych, dla którego możliwa jest ścisła integracja ze Springiem poprzez wykorzystania projektu Spring Cloud Netflix[49]. W skład tego stosu wchodzą przede wszystkim:

- Eureka[39] - serwis typu discovery,
- Zuul[42] - serwis zapewniający dynamiczne przekierowywanie żądań z bramy aplikacji do poszczególnych mikroserwisów,
- Ribbon[41] - serwis zapewniający równoważenie obciążenia podczas wyboru mikroserwisu, który odpowie na żądanie.

Jako serwis typu discovery postanowiono wykorzystać JHipster Registry[31], który jest oparty na serwisie Eureka i dodatkowo zapewnia metryki kompatybilne z ekosystemem aplikacji tworzonych w oparciu o generator JHipster.

Ze względu na wykorzystanie architektury mikroserwisowej zdecydowano się na uwierzytelnianie użytkowników z użyciem tokenów JWT[32], ponieważ jest to bezstanowy mechanizm, który można bezproblemowo wykorzystywać w środowisku rozproszonym.

W celu przyspieszenia rozwoju aplikacji postanowiono wykorzystać generator szkieletu aplikacji JHipster[30]. Korzystając z autorskiego języka domenowego JHipster możliwe jest zdefiniowanie konfiguracji systemu mikroserwisowego w oparciu o stos technologii Netflix OSS oraz encji przypisanych do konkretnych serwisów. Wykorzystaną konfigurację przedstawiono w dodatku A na listingu A.1, jednakże w celu zwiększenia czytelności postanowiono pominąć opis encji, które zostały bezpośrednio oparte na kategoriach opisanych w rozdziale 3.3. Na podstawie zdefiniowanej konfiguracji wygenerowane zostały szkielety serwisów zapewniające zarządzanie infrastrukturą mikroserwisów, uwierzytelnianie i autoryzację użytkowników oraz prowadzenie podstawowych operacji CRUD na encjach.

Aby zapewnić możliwie wysoką jakość oprogramowania konieczne jest przetestowanie czy kod działa w oczekiwany sposób. Do implementacji testów jednostkowych i integracyjnych po stronie backendu postanowiono wykorzystać bibliotekę JUnit[58] i platformę Mockito[13]

Dodatkowo postanowiono wykorzystać następujące narzędzia niezwiązane bezpośrednio z implementacją:

- Docker[9] - system konteneryzacji pozwalający uprościć proces wdrażania aplikacji z wykorzystaniem konfiguracji niezależnej od środowiska,
- Docker Compose[10] - narzędzie upraszczające zarządzanie wielokontenerowym środowiskiem aplikacji skonteneryzowanych,
- Git[54] - rozproszony system kontroli wersji wykorzystywany do zarządzania zmianami w kodzie,
- Gitlab Pipelines[19] - narzędzie wspomagające proces ciągłej integracji.

4.2. Zakres implementacji

W wyniku analizy założeń projektowych przedstawionych w rozdziale 2 stwierdzono, że osiągnięcie minimalnego funkcjonalnego stanu produktu (ang. Minimum Viable Product - MVP) wymaga zaimplementowania całego modelu domeny omówionego w projekcie bazy danych w rozdziale 3.3. Natomiast w kwestii dostępu do systemu, w celu osiągnięcia MVP wystarczy, żeby bezpośredni dostęp mieli tylko dietetycy i administratorzy, a pacjenci będą otrzymywali skomponowane diety na adres mailowy, który podali dietetykowi.

Ze względu na przetwarzanie wrażliwych danych osobowych, konieczne jest zawarcie w witrynie internetowej polityki prywatności. Na rzecz osiągnięcia MVP w przygotowywanej implementacji postanowiono wykorzystać szablon udostępniany przez firmę ogicom.pl[43]. Jednakże należy podkreślić, że autor niniejszej pracy inżynierskiej nie posiada wykształcenia prawniczego i należy traktować przedstawioną politykę prywatności jedynie jako wersję roboczą, a przed komercyjnym wdrożeniem systemu należałoby zasięgnąć porady w kancelarii prawnej oferującej doradztwo prawne z zakresu przetwarzania i ochrony danych osobowych.

Aby zapewnić użytkownikom aplikacji podstawową bazę produktów spożywczych postanowiono wykorzystać bazę USDA omówioną w rozdziale 1.2. Baza jest udostępniana w formie pliku bazy Microsoft Access. Żeby umożliwić wykorzystanie tej bazy w tworzonym programie, z pomocą narzędzi dostępnych w programie Microsoft Access przygotowano pliki w formacie CSV dla następujących kategorii:

- **KAT/2/01 Product,**
- **KAT/2/02 ProductBasicNutritionData,**
- **KAT/2/03 NutritionData,**
- **KAT/2/04 NutritionDefinition,**
- **KAT/2/05 NutritionDefinitionTranslation,**
- **KAT/2/06 HouseholdMeasure,**
- **KAT/2/07 ProductSubcategory,**
- **KAT/2/08 ProductCategory,**
- **KAT/2/09 ProductCategoryTranslation,**

Przygotowane pliki są wczytywane do bazy za pomocą Liquibase. Przykładowy skrypt ładujący dane z pliku CSV został przedstawiony na listingu 4.1.

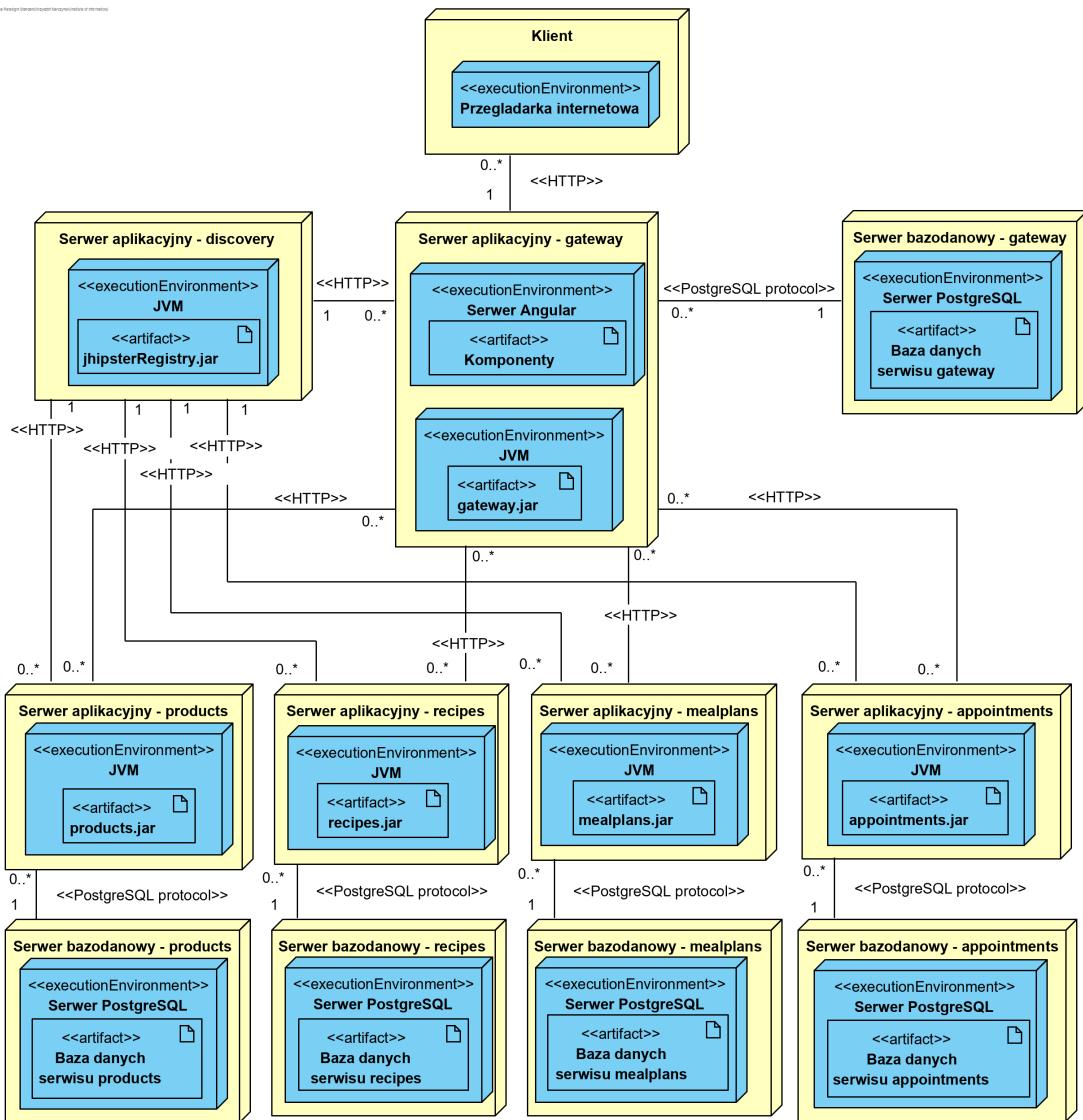
```
1 <changeSet id="201902130001" author="kmarczynski" context="usda">
2   <loadData
3     →   file="config/liquibase/usda_sr_db/household_measure.csv"
4     quotchar=''' separator=";" tableName="household_measure">
5     <column header="product_id" name="product_id" type="numeric"/>
6     <column header="measure_description" name="description"
7       →   type="string"/>
8     <column header="grams_weight" name="grams_weight"
9       →   type="numeric"/>
10    <column header="is_visible" name="is_visible" type="boolean"/>
11  </loadData>
12 </changeSet>
```

Listing 4.1: Skrypt ładujący dane z pliku CSV (źródło: opr. wł.)

4.3. Architektura systemu

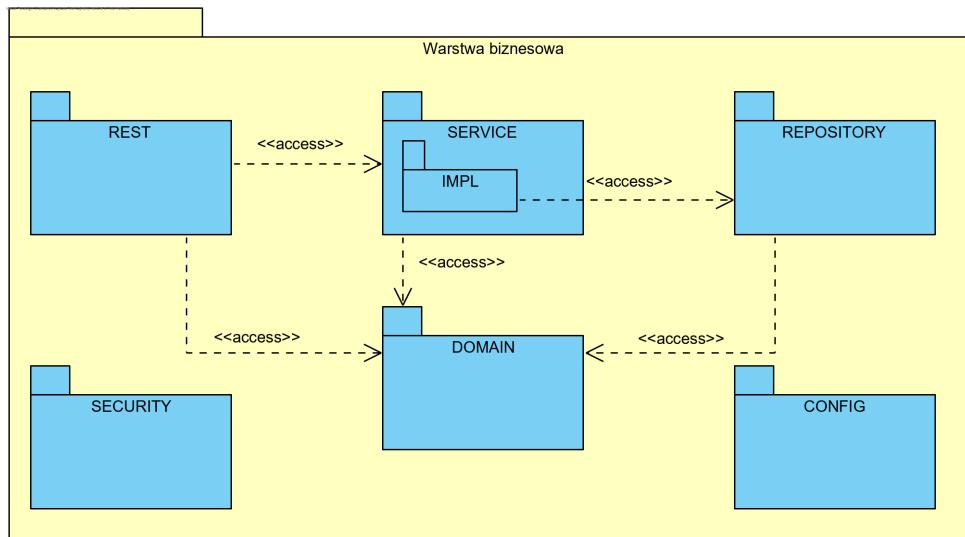
4.3.1. Architektura mikroserwisów

Na rysunku 4.1 przedstawiono diagram rozmieszczenia prezentujący fizyczne rozmieszczenie komponentów systemu. Mikroserwisy mogą mieć wiele instancji, które po uruchomieniu rejestrują się w serwisie JHipster Registry za pomocą protokołu HTTP. Serwis Gateway z pomocą mechanizmu równoważenia obciążenia komunikuje się z innymi serwisami za pomocą protokołu HTTP. Klient końcowy za pomocą przeglądarki internetowej łączy się z systemem za pomocą protokołu HTTP. Żądania klienta są obsługiwane przez jednąinstancję serwisu Gateway, a serwis Gateway może obsługiwać żądania wielu klientów. Warto zauważyć, że serwis JHipster Registry jest pojedynczym punktem awarii (ang. Single point of failure), gdyż wszystkie mikroserwisy korzystają z pojedynczej instancji tego serwisu. Komunikacja z serwerami bazodanowymi odbywa się za pomocą protokołu PostgreSQL. W celu uniknięcia niespójności danych przyjęto założenie, że może istnieć tylko jedna instancja bazy danych dla konkretnego serwisu.



Rysunek 4.1: Diagram rozmieszczenia (źródło: opr. wł.)

4.3.2. Architektura backendu



Rysunek 4.2: Podstawowy diagram pakietów warstwy biznesowej mikroserwisów (źródło: opr. wł.)

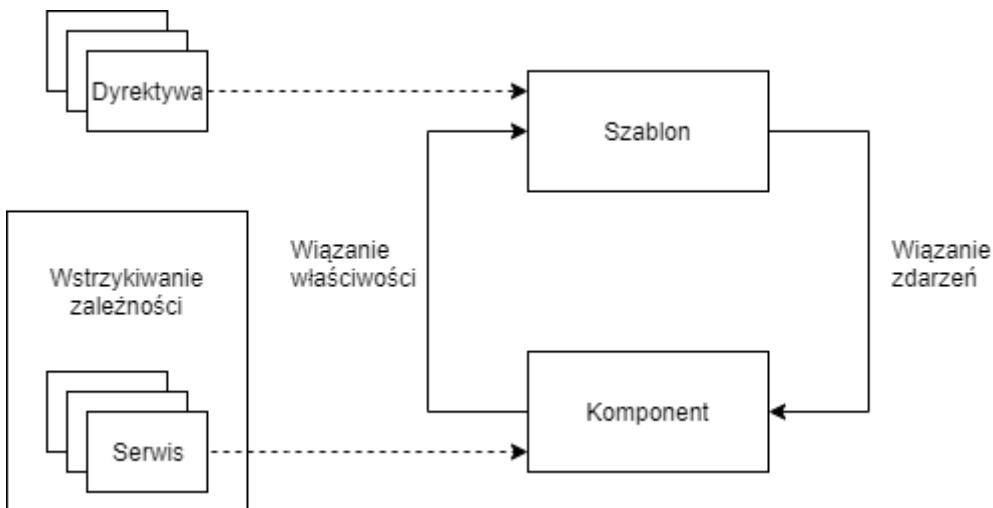
Na rysunku 4.2 przedstawiono diagram pakietów obrazujący konwencję zastosowaną we wszystkich mikroserwisach podczas implementowania kodu backendu. Pakiet "REST" jest odpowiedzialny za odbieranie żądań przychodzących do aplikacji i odpowiadanie na nie. W celu wykonania akcji biznesowej wywoływanie są funkcje z klas pakietu "SERVICE", który agreguje całą logikę biznesową aplikacji. Żeby ułatwić testowanie jednostkowe z wykorzystaniem Mockito w warstwie serwis postanowiono zaimplementować wzorzec projektowy Most[17] i oddzielić interfejsy serwisów od ich implementacji. Aby realizować niektóre akcje biznesowe konieczne jest odwołanie się do bazy danych. Za wszystkie operacje na bazie danych odpowiadają klasy z pakietu "REPOSITORY". We wszystkich wymienionych pakietach możliwe jest korzystanie z pakietu "DOMAIN" zawierającego definicje encji i typów wyliczeniowych.

Dodatkowo zdefiniowane są dwa pakiety nie związane z realizacją akcji biznesowych:

- "SECURITY" - odpowiedzialny za zabezpieczanie aplikacji,
- "CONFIG" - odpowiedzialny za zapewnienie konfiguracji aplikacji.

Klasy z pakietów "REST", "SERVICE.IMPL", "REPOSITORY", "SECURITY" i "CONFIG" zdefiniowane są jako komponenty Spring (ang. Spring Bean) przez co platforma Spring zarządza ich czasem życia i zapewnia mechanizm wstrzykiwania zależności[71].

4.3.3. Architektura frontendu



Rysunek 4.3: Ogólna architektura angulara (źródło: opr. wł.)

Architektura frontendu jest ściśle związaną a architekturą promowaną na platformie Angular opartej na komponentach, co przedstawiono na rysunku 4.3. Podstawową jednostką budulcową są komponenty, które za pomocą wiązania właściwości i zdarzeń oddziałują z szablonami w celu prezentacji treści użytkownikowi końcowemu. Komponenty mogą korzystać z serwisów dostępnych z pomocą mechanizmu wstrzykiwania zależności.

4.4. Dokumentacja kodu

Podstawowa dokumentacja kodu została napisana przy użyciu komentarzy w stylu kompatybilnym z generatorem dokumentacji JavaDoc[45]. Przykładowy komentarz przedstawiono na listingu 4.2, a fragment wygenerowanej dokumentacji na rysunku 4.4.

```
1 /**
2  * Short description of measure in language of a~product, e.g.
3  * ~ "cup" or "tea spoon"
4 */
5 @NotNull
6 @Size(min = 1, max = 255)
7 private String description;
```

Listing 4.2: Komentarz w stylu JavaDoc (źródło: opr. wł.)

The screenshot shows a JavaDoc interface with the following details:

- Navigation Bar:** OVERVIEW PACKAGE CLASS TREE DEPRECATED INDEX HELP
- Search Bar:** SEARCH: Search X
- Class Information:**
 - ALL CLASSES
 - SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD
 - Package:** pl.marczynski.dietify.products.domain
 - Class HouseholdMeasure**
- Java Declaration:**

```
java.lang.Object
pl.marczynski.dietify.products.domain.HouseholdMeasure
```
- Implemented Interfaces:**

```
All Implemented Interfaces:
java.io.Serializable
```
- Annotations:**

```
@Entity
public class HouseholdMeasure
extends java.lang.Object
implements java.io.Serializable
```
- Description:** A HouseholdMeasure.
- See Also:** Serialized Form
- Field Summary:** A table showing the fields of the HouseholdMeasure class.

Fields	Modifier and Type	Field	Description
	private @NotNull @Size(min=1,max=255) java.lang.String	description	Short description of measure in language of a product, e.g.
	private @NotNull @DecimalMin("0") java.lang.Double	gramsWeight	Grams weight of 1 unit of specified measure
	private java.lang.Long	id	
	private @NotNull java.lang.Boolean	isVisible	Flag specifying if measure is visible on presentation layer.
	private static long	serialVersionUID	

Rysunek 4.4: Przykładowy fragment dokumentacji JavaDoc (źródło: opr. wł.)

Dodatkowo wykorzystano narzędzie Swagger[56] do automatycznego generowania dokumentacji punktów końcowych interfejsu programowania aplikacji (ang. Application Programming Interface Endpoints - API Endpoints) na podstawie implementacji, czego rezultat zaprezentowano na rysunku 4.5.

The screenshot shows the Swagger UI interface for a 'Diet Type Resource'. At the top, it says 'diet-type-resource : Diet Type Resource'. Below that, there are buttons for 'Show/Hide', 'List Operations', and 'Expand Operations'. A 'GET' button is highlighted, pointing to the endpoint '/api/diet-types'. To the right of the endpoint is a link 'getAllDietTypes'. The main content area shows a 'Response Class (Status 200) OK' section. It includes tabs for 'Model' (selected) and 'Example Value'. The 'Example Value' tab displays a JSON array representing diet types, with one item expanded to show its internal structure. Below this, there's a 'Response Content Type' dropdown set to '*/*'. Under 'Response Messages', there's a table with columns 'HTTP Status Code', 'Reason', 'Response Model', and 'Headers'. It lists three error codes: 401 (Unauthorized), 403 (Forbidden), and 404 (Not Found). A 'Try it out!' button is located at the bottom left of this section.

Rysunek 4.5: Przykładowy fragment dokumentacji Swagger (źródło: opr. wł.)

4.5. Instalacja oprogramowania

4.5.1. Wymagania wstępne

Przed przystąpieniem do wykonywania kolejnych kroków należy się upewnić, że następujące narzędzia są zainstalowane:

- Open JDK 11 (<https://adoptopenjdk.net/?variant=openjdk11>),
- Node.js 10 lub nowsza wersja LTS (<https://nodejs.org/en/>),
- Docker 19.03 + Docker Compose 2 (<https://docs.docker.com/install/>).

4.5.2. Instalacja

Aby zbudować i uruchomić projekt z wykorzystaniem Dockera należy z poziomu głównego katalogu projektu wykonać polecenia przedstawione na listingu 4.3.

```
1 cd gateway
2 npm install
3 sh gradlew bootJar -Pprod jibDockerBuild
4 cd ../products
5 sh gradlew bootJar -Pprod jibDockerBuild
6 cd ../recipes
7 sh gradlew bootJar -Pprod jibDockerBuild
8 cd ../mealplans
9 sh gradlew bootJar -Pprod jibDockerBuild
10 cd ../appointments
11 sh gradlew bootJar -Pprod jibDockerBuild
12 cd ../docker-compose
13 sh docker-compose up
```

Listing 4.3: Skrypt komplilujący wszystkie mikroserwisy i uruchamiający aplikację na Dockerze (źródło: opr. wł.)

Alternatywnie, dla celów deweloperskich można zastosować uproszczony proces nie wykorzystujący Dockera. W tym celu należy najpierw uruchomić JHipster Registry wykonując polecenie z poziomu głównego katalogu projektu wykonając polecenia przedstawione na listingu 4.4.

```
1 cd service-discovery
2 java -jar ./jhipster-registry-5.0.2.jar
   --spring.profiles.active=dev
   --spring.security.user.password=admin
   --spring.cloud.config.server.composite.0.type=git
   --spring.cloud.config.server.composite.0.uri=
      https://github.com/jhipster/jhipster-registry-sample-config
```

Listing 4.4: Uruchamianie JHipster Registry (źródło: opr. wł.)

Następnie z poziomu katalogu każdego z serwisów (gateway, products, recipes, mealplans, appointments) należy wykonać polecenie uruchamiające Gradle Wrapper przedstawione na listingu 4.5.

```
1 ./gradlew
```

Listing 4.5: Uruchamianie Gradle Wrapper (źródło: opr. wł.)

Po uruchomieniu wszystkich serwisów aplikacja będzie dostępna pod adresem *localhost:8080*.

4.6. Prezentacja aplikacji

(podstawowy opis poruszania się po aplikacji, zrzuty ekranu z kilku najważniejszych widoków)

(filmik na youtube'ie?)

5. Testy

5.1. Wprowadzenie

Tworząc oprogramowanie istotne jest żeby akcje biznesowe i inne operacje wykonywane w systemie były realizowane w określony sposób i przynosiły oczekiwane rezultaty. Wiąże się z tym pojęcie zapewniania jakości, czyli przede wszystkim "spełnienie lub przekroczenie wymagań klienta"[73]. Podstawowym narzędziem pozwalającym na zapewnienie jakości oprogramowania jest przeprowadzanie testów. Według sylabusa Międzynarodowej Rady Kwalifikacji Testów Oprogramowania (ang. International Software Testing Qualifications Board - ISQB)[28] testowanie przeprowadza się w celu:

- znajdowania i zapobiegania błędom,
- zdobywania pewności odnośnie poziomu jakości,
- sprawdzania czy akcje biznesowe realizowane są w oczekiwany sposób,
- zapewniania informacji dotyczących bieżącego stanu implementacji.

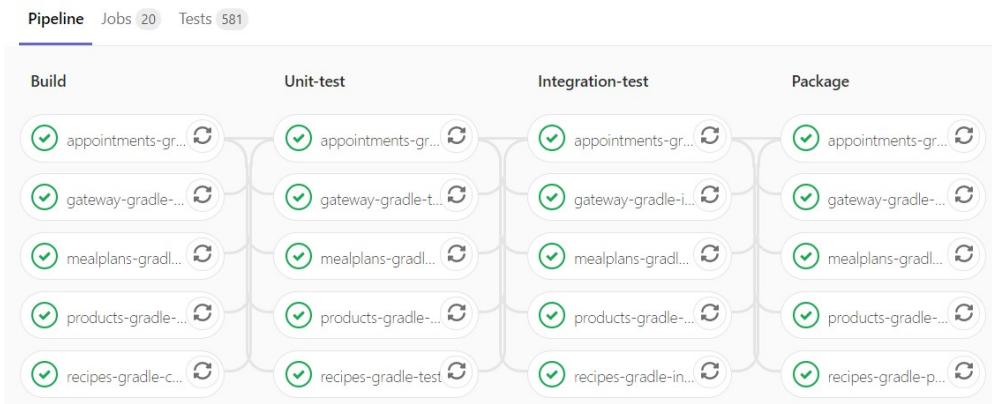
W ramach niniejszej pracy opisane zostaną następujące rodzaje testów:

- **jednostkowe** - testujące logikę biznesową,
- **integracyjne** - testujące punkty końcowe API,
- **użyteczności** - testy przeprowadzane z użytkownikami końcowymi sprawdzające użyteczność systemu.

Pierwsze dwa rodzaje testów mogą być przeprowadzane automatycznie, dlatego postanowiono wdrożyć rozwiązanie zapewniające ciągłą integrację (ang. Continuous Integration - CI) z wykorzystaniem platformy Gitlab CI[19]. Dzięki temu w momencie zwracania kodu do repozytorium automatycznie wykonywany jest zdefiniowany proces CI (ang. pipeline), zawierający następujące etapy:

- **build** - komplikacja kodu,
- **unit tests** - wykonanie testów jednostkowych,
- **integration tests** - wykonanie testów integracyjnych,
- **package** - pakowanie skompilowanego kodu do plików *.jar.

Rezultat wykonania procesu CI na platformie Gitlab CI został przedstawiony na rysunku 5.1.



Rysunek 5.1: Gitlab Pipeline (źródło: [20])

5.2. Testy jednostkowe

Robert C. Martin stwierdził, że dobrze napisane testy jednostkowe zwiększą elastyczność kodu produkcyjnego, ułatwiają wprowadzanie zmian w kodzie i pozwalają szybko wykryć zaistniałe błędy[35].

Przedmiotem testów jednostkowych jest logika akcji biznesowych przeprowadzanych w obrębie warstwy serwisów. Należy jednak zauważyć, że wątpliwą wartość przynosi testowanie funkcji, których jedyną odpowiedzialnością jest przekierowanie żądania z warstwy punktów końcowych API do warstwy repozytoriów.

Testy jednostkowe, jak nazwa wskazuje, powinny testować jednostkę taką jak funkcja i powinny daną jednostkę testować w izolacji od zależności zewnętrznych[48]. Aby osiągnąć taką izolację, zastosowano platformę Mockito dzięki czemu możliwe jest tworzenie makiet dla zależności zewnętrznych i definiowanie rezultatów obcowania z nimi. W rezultacie osiągnięta może być ścisła kontrola nad przepływem informacji w obrębie testowanej jednostki.

Na listingu 5.1 przedstawiony został przykładowy test jednostkowy.

```
1  @RunWith(MockitoJUnitRunner.class)
2  public class ExampleUnitTest {
3      @Mock private UserService userService;
4      @Mock private ProductRepository productRepository;
5      @Mock private CacheManager cacheManager;
6      @Mock private ProductSubcategoryService
7          ↳ productSubcategoryService;
8      @Mock EntityManager entityManager;
9      @InjectMocks private ProductServiceImpl productService;
10     private User user;
11     private Product product;
12
13     @Before
14     public void setup() {
15         long id = 1;
16         this.user = UserCreator.createEntity();
17         this.user.setId(id);
18         this.product = ProductCreator.createEntity(entityManager);
19         this.product.setId(id);
20         this.product.setAuthor(user);
21         when(userService.getCurrentUser())
22             .thenReturn(Optional.of(this.user));
23         when(productRepository.findOneWithEagerRelationships(any()))
24             .thenReturn(Optional.of(this.product));
25     }
26
27     @Test
28     public void authorShouldBeAbleToEditOwnProduct() {
29         //when
30         productService.save(product);
31         //then
32         Mockito.verify(productRepository,
33             ↳ times(1)).saveAndFlush(this.product);
34     }
35 }
```

Listing 5.1: Przykładowy test jednostkowy (źródło: opr. wł.)

5.3. Testy integracyjne

W przeciwieństwie do testów jednostkowych testy integracyjne nie są wykonywane w całkowitej izolacji[48]. Testowaniu podlegają punkty końcowe API z wykorzystaniem rzeczywistego połączenia z bazą danych. Na potrzeby testów tworzona jest instancja osadzonej bazy danych H2[25] oraz inicjalizowany jest kontekst aplikacji Spring Boot.

Celem przeprowadzania tego typu testów jest sprawdzenie czy punkty końcowe API w poprawny sposób obsługują przychodzące żądania z uwzględnieniem przeprowadzenia akcji biznesowych w warstwie serwisów i perzystencji w warstwie repozytoriów.

Na listingu 5.2 przedstawiony został przykładowy test integracyjny.

```
1  @SpringBootTest(classes = RecipesApp.class)
2  public class ExampleIntegrationTest {
3      @Autowired private DishTypeRepository dishTypeRepository;
4      @Autowired private DishTypeService dishTypeService;
5      @Autowired private DishTypeSearchRepository
6          → mockDishTypeSearchRepository;
7      @Autowired private MappingJackson2HttpMessageConverter
8          → jacksonMessageConverter;
9      @Autowired private PageableHandlerMethodArgumentResolver
10         → pageableArgumentResolver;
11      @Autowired private ExceptionTranslator exceptionTranslator;
12      @Autowired private EntityManager em;
13      @Autowired private Validator validator;
14
15      private MockMvc restDishTypeMockMvc;
16
17      @BeforeEach
18      public void setup() {
19          MockitoAnnotations.initMocks(this);
20          final DishTypeResource dishTypeResource = new
21              → DishTypeResource(dishTypeService);
22          this.restDishTypeMockMvc =
23              → MockMvcBuilders.standaloneSetup(dishTypeResource)
24                  .setCustomArgumentResolvers(pageableArgumentResolver)
25                  .setControllerAdvice(exceptionTranslator)
26                  .setConversionService(createFormattingConversionService())
27                  .setMessageConverters(jacksonMessageConverter)
28                  .setValidator(validator).build();
29      }
30
31      @Test
32      @Transactional
33      public void getNonExistingDishType() throws Exception {
34          // Get the dishType
35          restDishTypeMockMvc.perform(get("/api/dish-types/{id}",
36              → Long.MAX_VALUE))
37              .andExpect(status().isNotFound());
38      }
39  }
```

Listing 5.2: Przykładowy test integracyjny (źródło: opr. wł.)

5.4. Testy użyteczności

Oprogramowanie zwykle nie jest tworzone, żeby istniało w próżni. Należy więc zbadać w jakim stopniu aplikacja może być używana przez rzeczywistych użytkowników. W tym celu przeprowadzane są testy użyteczności[53].

W ramach realizacji tej grupy testów, 7 potencjalnych użytkowników (tj. osoby studiujące dietetykę lub zajmujące się profesjonalnie dietetyką) zostało poproszonych o zasymulowanie przeprowadzania kompleksowej wizyty pacjenta i wyrażenie opinii o używalności systemu poprzez odpowiedź na pytania ustandaryzowanego formularza Skali Używalności Systemu (ang. System Usability Scale - SUS)[64].

Korzystając z SUS, uczestnik badania udziela odpowiedzi na 10 pytań w pięciostopniowej skali od "Bardzo się zgadzam" do "Bardzo się nie zgadzam". Rezultaty następnie są konwertowane na wartość liczbową z zakresu 0-4, a suma otrzymanych punktów jest mnożona przez 2.5, żeby otrzymać wynik w skali 0-100. Wynik powyżej 68 punktów uznawany jest za ponadprzeciętny. Na rysunku 5.2 przedstawiony został wykorzystany kwestionariusz SUS.

Participant ID: _____ Site: _____ Date: ___/___/___

System Usability Scale

Instructions: For each of the following statements, mark one box that best describes your reactions to the website *today*.

	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
1. I think that I would like to use this website frequently.	<input type="checkbox"/>				
2. I found this website unnecessarily complex.	<input type="checkbox"/>				
3. I thought this website was easy to use.	<input type="checkbox"/>				
4. I think that I would need assistance to be able to use this website.	<input type="checkbox"/>				
5. I found the various functions in this website were well integrated.	<input type="checkbox"/>				
6. I thought there was too much inconsistency in this website.	<input type="checkbox"/>				
7. I would imagine that most people would learn to use this website very quickly.	<input type="checkbox"/>				
8. I found this website very cumbersome/awkward to use.	<input type="checkbox"/>				
9. I felt very confident using this website.	<input type="checkbox"/>				
10. I needed to learn a lot of things before I could get going with this website.	<input type="checkbox"/>				

Rysunek 5.2: Kwestionariusz SUS (źródło: [63])

(wyniki sus)

Zakończenie

Rezultatem wykonanych prac jest platforma wspomagająca zarządzanie dietą spełniająca kryteria biznesowe postawione w fazie analizy wymagań. Dietetycy mogą wykorzystać opracowane rozwiązanie do ułatwienia układania i udostępniania swoim pacjentom jadłospisów, a także do kontroli rezultatów stosowania diety przez pacjentów.

Należy jednak wziąć pod uwagę, że oprogramowanie to ma na celu przede wszystkim ułatwienie pracy dietetyka i zakłada się, że użytkownicy korzystający z aplikacji będą mieli specjalistyczną wiedzę w dziedzinie dietetyki.

Oprogramowanie zostało zaprojektowane w sposób ułatwiający skalowanie, a zastosowane wzorce i dobre praktyki programistyczne sprawią, że wdrażanie nowych funkcjonalności w systemie nie będzie stanowiło problemu. Do funkcjonalności, które mogłyby zostać wdrożone w systemie w przyszłości można zaliczyć przede wszystkim:

- Zapewnienie możliwości korzystania z aplikacji bez połączenia z internetem poprzez dodanie funkcjonalności progresywnej aplikacji webowej (ang. Progressive Web App - PWA)[23].
- Zapewnienie możliwości instalacji aplikacji na urządzeniach mobilnych poprzez integrację z platformą Ionic[11].
- Umożliwienie pacjentom korzystania z aplikacji.
- Zapewnienie możliwości komunikacji pacjenta z dietetykiem z wykorzystaniem komunikatora zintegrowanego z opracowywanym systemem.

Wdrożenie aplikacji komercyjnie i szersza współpraca z ekspertami domenowymi z dziedziny dietetyki z pewnością pokazałyby wiele innych możliwości rozwoju opracowanej platformy, jednakże może to wymagać zainwestowania znacznych środków finansowych.

Bibliografia

- [1] Aliant Kalkulator Dietetyczny, *Kreator jadłospisu*, <https://www.youtube.com/watch?v=De2Acb1BPTU&t=380>. 2018. Dostęp 31.10.2019.
- [2] Anmarsoft, *Aliant Kalkulator dietetyczny*, <https://aliant.com.pl>. Dostęp 13.09.2019.
- [3] AURA GROUP, *Dietetyk Pro*, <https://program.dietetykpro.pl/>. Dostęp 13.09.2019.
- [4] Catlin, H., Weizenbaum, N., Eppstein, C., Anne, J., *Sass*, <https://sass-lang.com/>. Dostęp 10.09.2019.
- [5] Ciborowska, H., Rudnicka, A., *Dietetyka. Żywienie zdrowego i chorego człowieka*, 4 wyd. (PZWL, Warszawa, 2019).
- [6] Datical, *Liquibase 3.6.3*, <https://www.liquibase.org/>. Dostęp 10.05.2019.
- [7] dietetykpro, *DietetykPro cz.4 - układanie jadłospisu 1/2*, <https://www.youtube.com/watch?v=y8M3m38AStE&t=140>. 2016. Dostęp 31.10.2019.
- [8] Dietico, *Dietico.pl - Wydruk jadłospisu z programu*, <https://www.youtube.com/watch?v=hgVIOQJwb-c&t=21>. 2018. Dostęp 31.10.2019.
- [9] Docker Inc., *Docker*, <https://www.docker.com/>. Dostęp 10.09.2019.
- [10] Docker Inc., *Docker Compose*, <https://docs.docker.com/compose/>. Dostęp 10.09.2019.
- [11] Drifty Co., *Ionic*, <https://ionicframework.com/>. Dostęp 19.11.2019.
- [12] Evans, E., *Domain-Driven Design. Zapanuj nad złożonym systemem informatycznym* (Helion, Gliwice, 2015).
- [13] Faber, S., *Mockito 3*, <https://site.mockito.org/>. Dostęp 10.09.2019.
- [14] Food and Agriculture Organization of the United Nations, *Energy requirements of adults*, <http://www.fao.org/3/y5686e/y5686e07.htm>. Dostęp 28.10.2019.
- [15] Fowler, M., *Evolutionary Database Design*, <https://martinfowler.com/articles/evodb.html>. Dostęp 18.11.2019.
- [16] Fowler, M., *MonolithFirst*, <https://martinfowler.com/bliki/MonolithFirst.html>. Dostęp 27.10.2019.
- [17] Gamma, E., Helm, R., Johnson, R., Vlissides, J., *Wzorce projektowe. Elementy oprogramowania obiektowego wielokrotnego użytku* (Helion, Gliwice, 2010).
- [18] Głabska, D., Kozłowska, L., Lange, E., Włodarek, D., *Dietoterapia* (PZWL, Warszawa, 2017).
- [19] Gitlab, *Creating and using CI/CD pipelines*, <https://docs.gitlab.com/ee/ci/pipelines.html>. Dostęp 10.11.2019.
- [20] Gitlab, *Dietify - Pipeline - 99334505*, <https://gitlab.com/kp.marczynski/dietify/pipelines/99334505>. Dostęp 28.11.2019.
- [21] Google, *Angular 7.2.4*, <https://angular.io>. Dostęp 10.05.2019.
- [22] Google, *Google Trends*, <https://trends.google.pl>. Dostęp 10.09.2019.
- [23] Google, *Progressive Web Apps*, <https://developers.google.com/web/progressive-web-apps>. Dostęp 19.11.2019.
- [24] Google, *YouTube*, <https://youtube.com>. Dostęp 10.09.2019.
- [25] H2 Group, *H2 Database Engine*, <https://www.h2database.com>. Dostęp 20.11.2019.
- [26] HERMAX sp. z o.o., *Kcalmar PRO*, <https://kcalmar.com/>. Dostęp 13.09.2019.
- [27] Instytut Żywności i Żywienia, *Regulamin dostępu do bazy "Wartość odżywcza produktów spożywczych i potraw"*, <http://www.izz.waw.pl/warto-odywcza-produktow-spojwczych-i-potraw/regulamin-dostpu-do-bazy>. Dostęp 28.10.2019.

- [28] ISTQB, *Foundation Level Syllabus*, <https://www.istqb.org/downloads/syllabi/foundation-level-syllabus.html>. Dostęp 20.12.2019.
- [29] Jarosz, M., Bułhak-Jachymczyk, B., *Normy żywienia człowieka. Podstawy prewencji otyłości i chorób niezakaźnych* (PZWL, Warszawa, 2008).
- [30] JHipster, *JHipster 6.1.2*, <https://www.jhipster.tech/>. Dostęp 10.09.2019.
- [31] JHipster, *The JHipster Registry*, <https://www.jhipster.tech/jhipster-registry/>. Dostęp 19.11.2019.
- [32] Jones, M., Bradley, J., Sakimura, N., *JSON Web Token*, <https://tools.ietf.org/html/rfc7519>. Dostęp 10.09.2019.
- [33] Kcalmar, *Kcalmar.pro - funkcja oznaczania alergenów, produktów nietolerowanych, lubianych i nielubianych*, https://www.youtube.com/watch?v=e1S8NOQ_oEY&t=58. 2017. Dostęp 31.10.2019.
- [34] Kunachowicz, H., Przygoda, B., Nadolna, I., Iwanow, K., *Tabele składu i wartości odżywczej żywności* (PZWL, Warszawa, 2017).
- [35] Martin, R.C., *Czysty kod. Podręcznik dobrego programisty* (Helion, Gliwice, 2010).
- [36] Mazur, H., Mazur, Z., *Projektowanie relacyjnych baz danych* (Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław, 2004).
- [37] Microsoft, *Common web application architectures*, <https://docs.microsoft.com/en-us/dotnet/architecture/modern-web-apps-azure/common-web-application-architectures>. Dostęp 27.10.2019.
- [38] Microsoft, *TypeScript 3.4*, <https://www.typescriptlang.org/>. Dostęp 10.05.2019.
- [39] Netflix Inc., *Eureka*, <https://github.com/Netflix/eureka>. Dostęp 10.09.2019.
- [40] Netflix Inc., *Netflix OSS*, <https://netflix.github.io/>. Dostęp 10.09.2019.
- [41] Netflix Inc., *Ribbon*, <https://github.com/Netflix/ribbon>. Dostęp 10.09.2019.
- [42] Netflix Inc., *Zuul Proxy*, <https://github.com/Netflix/zuul>. Dostęp 10.09.2019.
- [43] Ogiom.pl, *Wzór polityki prywatności*, <https://www.ogicom.pl/dokumenty/ogicom-wzor-polityka-prywosci.rtf>. Dostęp 11.11.2019.
- [44] Opracowanie zbiorowe, *Encyklopedia Powszechna PWN*, tom 7 (Wydawnictwo Naukowe PWN, Warszawa, 2009).
- [45] Oracle, *JavaDoc*, <https://docs.oracle.com/javase/8/docs/technotes/tools/windows/javadoc.html>. Dostęp 19.11.2019.
- [46] Oracle Corporation, *Java 8 SE*, <https://docs.oracle.com/javase/8/docs/>. Dostęp 10.05.2019.
- [47] Oracle Corporation, *MySQL 8.0*, <https://www.mysql.com/>. Dostęp 27.10.2019.
- [48] Osherove, R., *Testy jednostkowe. Świat niezawodnych aplikacji*, 2 wyd. (Helion, Gliwice, 2014).
- [49] Pivotal Software, *Spring Cloud Netflix*, <https://spring.io/projects/spring-cloud-netflix>. Dostęp 10.11.2019.
- [50] Pivotal Software, *Spring Framework 5*, <https://spring.io/>. Dostęp 10.05.2019.
- [51] Redakcja dietetycy.org.pl, *Programy dla dietetyków 2018 – porównanie*, <https://dietetycy.org.pl/programy-dla-dietetykow-2018-porownanie/>. Dostęp 28.10.2019.
- [52] Richardson, C., *Microservices Patterns* (Packt Publishing, Birmingham, 2018).
- [53] Roman, A., *Testowanie i jakość oprogramowania* (Wydawnictwo Naukowe PWN, Warszawa, 2015).
- [54] Scott Chacon, *Git*, <https://git-scm.com/>. Dostęp 10.09.2019.
- [55] Sharma, S., *Mastering Microservices with Java 9*, 2 wyd. (Packt Publishing, Birmingham, 2017).
- [56] Smartbear, *Swagger*, <https://swagger.io/>. Dostęp 19.11.2019.
- [57] Spotbeans sp. z o.o., *TiqDiet*, <https://tiqdiet.com/pl/>. Dostęp 13.09.2019.
- [58] The JUnit Team, *JUnit 5*, <https://junit.org/junit5/>. Dostęp 10.05.2019.
- [59] The PostgreSQL Global Development Group, *PostgreSQL 11.5*, <https://www.postgresql.org/>. Dostęp 10.09.2019.
- [60] Thomasworks, *Vitme*, <https://vitme.pl/>. Dostęp 13.09.2019.

- [61] TiqDiet, *TiqDiet - tworzenie jadłospisu*, <https://www.youtube.com/watch?v=M0mr6oNHBXA&t=61>. 2016. Dostęp 31.10.2019.
- [62] de la Torre, C., Wagner, B., Rousos, M., *.NET Microservices: Architecture for Containerized .NET Applications* (Microsoft, Redmond, 2019).
- [63] Tullis, T., Albert, B., *System Usability Scale Form*, <https://www.measuringux.com/SUS.pdf>. Dostęp 20.11.2019.
- [64] U.S. Department of Health and Human Services, *System Usability Scale*, <https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html>. Dostęp 20.11.2019.
- [65] USDA, *USDA National Nutrient Database for Standard Reference, Legacy*, <http://www.ars.usda.gov/Services/docs.htm?docid=8964>. Dostęp 28.10.2019.
- [66] Vernon, V., *DDD. Kompendium Wiedzy* (Helion, Gliwice, 2018).
- [67] Vitme - Programy dla Dietetyków i Branży Spożywczej, *VITME - Program dla Dietetyków i Trenerów Personalnych*, <https://www.youtube.com/watch?v=xxD2q9rcusY&t=112>. 2018. Dostęp 31.10.2019.
- [68] W3C, *HTML and CSS*, <https://www.w3.org/standards/webdesign/htmlcss>. Dostęp 10.09.2019.
- [69] W3C, *JavaScript Web APIs*, <https://www.w3.org/standards/webdesign/script>. Dostęp 10.11.2019.
- [70] Walczak, M., Krasowska-Walczak, G., *Dietetyk na rynku usług medycznych w Polsce i wybranych krajach*, Zdrowie Publiczne i Zarządzanie. 2015, tom 13 (2): 204–215.
- [71] Walls, C., *Spring w akcji*, 4 wyd. (Helion, Gliwice, 2015).
- [72] XLTEAM, *Dietico*, <https://dietico.pl/>. Dostęp 13.09.2019.
- [73] Zmitrowicz, K., *Jakość projektów informatycznych. Rozwój i testowanie oprogramowania* (Helion, Gliwice, 2015).

Spis rysunków

1.	Zainteresowanie hasłem "dietetyk" w ujęciu czasowym (źródło: [22])	2
1.1.	TiqDiet (źródło: [61])	5
1.2.	Kcalmar PRO (źródło: [33])	5
1.3.	Dietetyk Pro (źródło: [7])	6
1.4.	Aliant (źródło: [1])	7
1.5.	Dietico (źródło: [8])	7
1.6.	Vitme (źródło: [67])	8
2.1.	Diagram przypadków użycia - użytkownicy (źródło: opr. wł.)	18
2.2.	Diagram przypadków użycia - poddziedzina administracyjna (źródło: opr. wł.)	20
2.3.	Diagram przypadków użycia - poddziedzina produkty (źródło: opr. wł.) . .	22
2.4.	Diagram przypadków użycia - poddziedzina przepisy (źródło: opr. wł.) . . .	24
2.5.	Diagram przypadków użycia - poddziedzina jadłospisy (źródło: opr. wł.) . .	26
2.6.	Diagram przypadków użycia - poddziedzina wizyty (źródło: opr. wł.) . . .	28
3.1.	Prototyp interfejsu - strona startowa (źródło: opr. wł.)	30
3.2.	Prototyp interfejsu - układ strony na urządzeniu mobilnym (źródło: opr. wł.) .	31
3.3.	Prototyp interfejsu - widok administratora (źródło: opr. wł.)	32
3.4.	Prototyp interfejsu - rejestrowanie do systemu (źródło: opr. wł.)	33
3.5.	Prototyp interfejsu - logowanie do systemu (źródło: opr. wł.)	33
3.6.	Prototyp interfejsu - zarządzanie użytkownikami (źródło: opr. wł.)	34
3.7.	Prototyp interfejsu - lista produktów (źródło: opr. wł.)	35
3.8.	Prototyp interfejsu - dodawanie nowego lub edycja istniejącego produktu (źródło: opr. wł.)	36
3.9.	Prototyp interfejsu - szczegóły produktu (źródło: opr. wł.)	36
3.10.	Prototyp interfejsu - lista przepisów (źródło: opr. wł.)	37
3.11.	Prototyp interfejsu - dodawanie nowego lub edycja istniejącego przepisu (źródło: opr. wł.)	38
3.12.	Prototyp interfejsu - szczegóły przepisu (źródło: opr. wł.)	38
3.13.	Prototyp interfejsu - lista jadłospisów (źródło: opr. wł.)	39
3.14.	Prototyp interfejsu - dodawanie nowego lub edycja istniejącego jadłospisu - zakładka ustawień (źródło: opr. wł.)	40
3.15.	Prototyp interfejsu - dodawanie nowego lub edycja istniejącego jadłospisu - zakładka kalendarza (źródło: opr. wł.)	40
3.16.	Prototyp interfejsu - lista wizyt (źródło: opr. wł.)	41
3.17.	Prototyp interfejsu - dodawanie nowej karty pacjenta (źródło: opr. wł.) . .	42
3.18.	Prototyp interfejsu - szczegóły karty pacjenta (źródło: opr. wł.)	43
3.19.	Prototyp interfejsu - dodawanie nowej wizyty (źródło: opr. wł.)	43
3.20.	Prototyp interfejsu - szczegóły wizyty (źródło: opr. wł.)	44
3.21.	Prototyp interfejsu - wizyta - wywiad żywieniowy - krok 1 (źródło: opr. wł.)	45

3.22. Prototyp interfejsu - wizyta - wywiad żywieniowy - krok 2 (źródło: opr. wł.)	45
3.23. Prototyp interfejsu - wizyta - wywiad żywieniowy - krok 3 (źródło: opr. wł.)	46
3.24. Prototyp interfejsu - wizyta - pomiary ciała (źródło: opr. wł.)	47
3.25. Podstawowa architektura systemu (źródło: opr. wł.)	48
3.26. Diagram klas - typy danych (źródło: opr. wł.)	49
3.27. Diagram klas - poddziedzina administracyjna (źródło: opr. wł.)	52
3.28. Diagram klas - poddziedzina produkty (źródło: opr. wł.)	59
3.29. Diagram klas - poddziedzina przepisy (źródło: opr. wł.)	66
3.30. Diagram klas - poddziedzina jadłospisy (źródło: opr. wł.)	71
3.31. Diagram klas - poddziedzina wizyty (źródło: opr. wł.)	77
4.1. Diagram rozmieszczenia (źródło: opr. wł.)	82
4.2. Podstawowy diagram pakietów warstwy biznesowej mikroserwisów (źródło: opr. wł.)	83
4.3. Ogólna architektura angulara (źródło: opr. wł.)	84
4.4. Przykładowy fragment dokumentacji JavaDoc (źródło: opr. wł.)	85
4.5. Przykładowy fragment dokumentacji Swagger (źródło: opr. wł.)	86
5.1. Gitlab Pipeline (źródło: [20])	90
5.2. Kwestionariusz SUS (źródło: [63])	93

Spis tabel

1.1. Rozwiązania konkurencyjne - cechy funkcjonalne (źródło: opr. wł.)	9
1.2. Rozwiązania konkurencyjne - cechy niefunkcjonalne (źródło: opr. wł.)	10
2.1. Sformułowanie problemu (źródło: opr. wł.)	16
2.2. Pozycjonowanie produktu (źródło: opr. wł.)	17
2.3. Użytkownicy (źródło: opr. wł.)	18
2.4. Wymagania funkcjonalne - poddziedzina administracyjna (źródło: opr. wł.)	19
2.5. Wymagania funkcjonalne - poddziedzina produkty (źródło: opr. wł.)	21
2.6. Wymagania funkcjonalne - poddziedzina przepisy (źródło: opr. wł.)	23
2.7. Wymagania funkcjonalne - poddziedzina jadłospisy (źródło: opr. wł.)	25
2.8. Wymagania funkcjonalne - poddziedzina wizyty (źródło: opr. wł.)	27

Spis kodów źródłowych

4.1	Skrrypt ładujący dane z pliku CSV (źródło: opr. wł.)	81
4.2	Komentarz w stylu JavaDoc (źródło: opr. wł.)	84
4.3	Skrypt komplilujący wszystkie mikroserwisy i uruchamiający aplikację na Dockerze (źródło: opr. wł.)	87
4.4	Uruchamianie JHipster Registry (źródło: opr. wł.)	87
4.5	Uruchamianie Gradle Wrapper (źródło: opr. wł.)	87
5.1	Przykładowy test jednostkowy (źródło: opr. wł.)	91
5.2	Przykładowy test integracyjny (źródło: opr. wł.)	92
A.1	Definicja mikroserwisów w języku JDL (źródło: opr. wł.)	105

A. Konfiguracja mikroserwisów w języku JDL

```
1 // <<START gateway.jh>>
2 application {
3     config {
4         baseName gateway
5         packageName pl.marczynski.dietify.gateway
6
7         applicationType gateway
8         serverPort 8080
9
10        authenticationType jwt
11        buildTool gradle
12        serviceDiscoveryType eureka
13
14        databaseType sql
15        devDatabaseType h2Disk
16        prodDatabaseType postgresql
17        cacheProvider hazelcast
18        enableHibernateCache true
19
20        enableTranslation true
21        nativeLanguage en
22        languages [en, pl]
23
24        useSass true
25        clientFramework angularX
26        //clientTheme flatly
27        //clientThemeVariant primary
28    }
29    entities *
30 }
31
32 deployment {
33     deploymentType docker-compose
34
35     appsFolders [gateway, products, recipes, mealplans, appointments]
36     dockerRepositoryName "dietify"
37     monitoring elk
38 }
39
40 service * with serviceImpl
41 noFluentMethod *
```

```

42 // <<END gateway.jh>>
43
44 // <<START products.jh>>
45 application {
46     config {
47         baseName products
48         packageName pl.marczynski.dietify.products
49
50         applicationType microservice
51         serverPort 8081
52
53         authenticationType jwt
54         buildTool gradle
55         serviceDiscoveryType eureka
56
57         databaseType sql
58         devDatabaseType h2Disk
59         prodDatabaseType postgresql
60         cacheProvider hazelcast
61         enableHibernateCache true
62
63         enableTranslation true
64         nativeLanguage en
65         languages [en, pl]
66     }
67     entities Product, ProductSubcategory, ProductCategory,
68     ↳ ProductCategoryTranslation, NutritionData, NutritionDefinition,
69     ↳ NutritionDefinitionTranslation, HouseholdMeasure, DietType,
70     ↳ DietTypeTranslation, ProductBasicNutritionData
71 }
72 // <<END products.jh>>
73
74 // <<START recipes.jh>>
75 application {
76     config {
77         baseName recipes
78         packageName pl.marczynski.dietify.recipes
79
80         applicationType microservice
81         serverPort 8082
82
83         authenticationType jwt
84         buildTool gradle
85         serviceDiscoveryType eureka
86
87         databaseType sql
88         devDatabaseType h2Disk
89         prodDatabaseType postgresql
90         cacheProvider hazelcast
91         enableHibernateCache true

```

```

90     enableTranslation true
91     nativeLanguage en
92     languages [en, pl]
93
94     testFrameworks [gatling, cucumber]
95 }
96 entities Recipe, MealType, MealTypeTranslation, DishType,
97   ↳ DishTypeTranslation, KitchenAppliance, KitchenApplianceTranslation,
98   ↳ RecipeSuitableForDiet, RecipeUnsuitableForDiet, RecipeSection,
99   ↳ ProductPortion, PreparationStep, RecipeBasicNutritionData
100 }
101 // <<END recipes.jh>>
102
103 // <<START mealPlans.jh>>
104 application {
105     config {
106         baseName mealplans
107         packageName pl.marczynski.dietify.mealplans
108
109         applicationType microservice
110         serverPort 8083
111
112         authenticationType jwt
113         buildTool gradle
114         serviceDiscoveryType eureka
115
116         databaseType sql
117         devDatabaseType h2Disk
118         prodDatabaseType postgresql
119         cacheProvider hazelcast
120         enableHibernateCache true
121
122         enableTranslation true
123         nativeLanguage en
124         languages [en, pl]
125     }
126     entities MealPlan, MealDefinition, MealPlanSuitableForDiet,
127       ↳ MealPlanUnsuitableForDiet, MealPlanDay, Meal, MealRecipe, MealProduct
128 }
129 // <<END mealPlans.jh>>
130
131 // <<START appointments.jh>>
132 application {
133     config {
134         baseName appointments
135         packageName pl.marczynski.dietify.appointments
136
137         applicationType microservice
138         serverPort 8084
139
140         authenticationType jwt

```

```
137     buildTool gradle
138     serviceDiscoveryType eureka
139
140     databaseType sql
141     devDatabaseType h2Disk
142     prodDatabaseType postgresql
143     cacheProvider hazelcast
144     enableHibernateCache true
145
146     enableTranslation true
147     nativeLanguage en
148     languages [en, pl]
149 }
150 entities Appointment, BodyMeasurement, PatientCard, NutritionalInterview,
151   ↳ AssignedMealPlan, OwnedKitchenAppliance,
152   ↳ CustomNutritionalInterviewQuestion, AppointmentEvaluation
153 }
154 // <<END appointments.jh>>
```

Listing A.1: Definicja mikroserwisów w języku JDL (źródło: opr. wł.)