AKADEMIA TARNOWSKA

Wydział Politechniczny

Kierunek: ***Informatyka***

Specjalność: Informatyka stosowana

Specjalizacja: Inżynieria systemów inteligentnych

2023/2024

Katarzyna Jaszczur

PRACA INŻYNIERSKA

Asystent dietetyczny

Promotor pracy:

mgr inż. Tomasz Gądek

Tarnów, 2024

Spis treści

1. Wstęp

Przedmiotem niniejszej pracy inżynierskiej była aplikacja internetowa, będąca asystentem dietetycznym. W treści zostaną przedstawione wykorzystane algorytmy, technologie oraz uzyskane efekty.

1.1. Motywacja

Obecnie około 2,6 miliarda ludzi na świecie - czyli około 38 procent populacji boryka się z problemem nadwagi lub otyłości [12]. Jeżeli trend wzrostowy się utrzyma to w ciągu kolejnych 12 lat liczba ta wzrośnie do 4 miliardów, czyli około 51 procent ludności całego świata - szacują eksperci w Światowym Atlasie Otyłości 2023. Zdaniem nich taki wzrost przyczyni się do znacznego obciążenia dla gospodarek, osiągając aż trzy procent światowego PKB. "Globalny ekonomiczny wpływ nadwagi i otyłości sięgnie 4,32 biliona dolarów rocznie do 2035 roku, jeśli nie zwiększą się środki zapobiegawcze i nie poprawi się leczenie" - wskazano.

Aktualnie obserwuje się przyśpieszony wzrost liczby przypadków otyłości u dzieci poniżej 18 roku życia. Prognozuje się wzrost o nawet 100% u chłopców i o 125% u dziewczynek w stosunku do 2020 roku [13]. Organizacja zauważyła też, że największy wzrost przypadków otyłości w najbliższych latach nastąpi w krajach o niskich i średnich dochodach Afryki i Azji, które są również najsłabiej przygotowane do walki z tą przypadłością.

Zdaniem WHO czyli Światowej Organizacji Zdrowia, dwuletni lockdown podczas pandemii przyczynił się do wzrostu liczby przypadków otyłości. W tym okresie wiele osób znacznie ograniczyło aktywność poza domem oraz popadło w złe nawyki żywieniowe.

O nadwadze mówi się, gdy indeks masy ciała (BMI) wynosi powyżej 25, a otyłość ma miejsce, gdy wskaźnik ten jest co najmniej na poziomie 30 [30]. W różnych badaniach udowodniono, że nadmierna masa ciała przyczynia się do zwiększonego ryzyka zachorowania na wiele chorób, w tym nowotwory i choroby serca.

Jednym z pierwszych kroków, które należy podjąć przy walce z otyłością jest staranie się wprowadzić zdrowsze nawyki żywieniowe. Na początku tej drogi pomóc mogą aplikacje internetowe pomagające zaplanować posiłki w zależności od podanych przez nas danych.

1.2. Cel pracy

Celem pracy było zaprojektowanie aplikacji internetowej, której zadaniem będzie pomóc w zrealizowaniu wyznaczonego sobie celu związanego z odżywianiem. Aplikacja będzie promować zdrowe odżywianie poprzez pokazanie użytkownikowi propozycji dziennych jadłospisów mających uzupełnić jego dzienne zapotrzebowanie kaloryczne.

Użytkownikowi po wcześniejszym wprowadzeniu podstawowych informacji na swój temat oraz podaniu kulinarnych preferencji zostanie przedstawione kilka wersji diet.

Inne bardziej szczegółowe informacje dotyczące funkcji aplikacji zostaną opisane w kolejnym rozdziale.

1.3. Zakres pracy

W skład pracy inżynierskiej wchodzą:

- aplikacja serwerowa,
- aplikacja internetowa,
- baza danych.

Wszystkie z wymienionych modułów zostaną szczegółowo przedstawione w dalszej części pracy.

2. Technologie i narzędzia

W rozdziale zostały opisane najważniejsze technologie wykorzystane do utworzenia aplikacji webowej, będącej przedmiotem tej pracy.

2.1 JavaScript

Język JavaScript to język skryptowy, który powstał w 1995 roku jako technologia umożliwiająca dodawanie programów do stron internetowych w przeglądarce Netscape Navigator [2]. Następnie został przyjęty we wszystkich pozostałych przeglądarkach internetowych. To jemu zawdzięczamy Internet w takim stanie, w jakim jest obecnie. Strony z interaktywnymi aplikacjami nie wymagają odświeżenia po każdej wykonanej czynności. Używamy go również na tradycyjnych stronach internetowych, w celu zapewnienia różnych rodzajów interaktywności i innych dodatków.

Zastosowanie JavaScript [3]:

- może poprawić wydajność witryny (dzięki technologii Ajax),
- może posłużyć do naprawienia braków w przeglądarce, na przykład wprowadzenia obsługi nowszych selektorów CSS,
- może być stosowany w urządzeniach przenośnych (w zależności od urządzenia),
- przenosi część zadań z serwera do klienta, odciążając ten pierwszy.

2.2 Node.js

Node.js to otwarte i wieloplatformowe środowisko uruchomieniowe dla języka JavaScript [6]. Został zaprojektowany przez Ryana Dahla, amerykańskiego programistę w 2009 roku.
Od tego czasu projekt zyskał na popularności i stał się istotnym narzędziem w świecie programowania. Znajduje on zastosowanie w niemal każdego rodzaju projekcie.

Node.js korzysta z silnika V8 JavaScript, który stanowi rdzeń przeglądarki Google Chrome. Dzięki temu Node.js charakteryzuje się wysoką wydajnością.

Aplikacja Node.js działa w jednym procesie, co eliminuje konieczność tworzenia nowego wątku dla każdego żądania. W standardowej bibliotece Node.js znajduje się zestaw asynchronicznych funkcji I/O, co pozwala na unikanie blokowania kodu JavaScript.

Biblioteki w Node.js są pisane z wykorzystaniem paradygmatów nieblokujących, co sprawia, że blokowanie staje się wyjątkiem, a nie regułą.

W trakcie operacji I/O (wejścia/wyjścia), takich jak czytanie z sieci, dostęp do bazy danych lub systemu plików, Node.js unika blokowania wątku, co pozwala na efektywne wykorzystanie cykli procesora. Node.js wznawia operację po otrzymaniu odpowiedzi, eliminując potrzebę czekania. Dzięki takiemu podejściu Node.js obsługuje tysiące równoczesnych połączeń na jednym serwerze, bez wprowadzania problemów związanych z zarządzaniem współbieżnością wątków, co może prowadzić do błędów.

2.3 Framework Express

Express.js to popularna biblioteka Node.js typu open source wspomagająca pracę backend developerów [18]. Została wydana w 2010 roku jako darmowe oprogramowanie. Pozwala ona tworzyć proste strony WWW oraz hybrydowe aplikacje webowe, które można uruchamiać w przeglądarce. Cechuje się dużą lekkością i wydajnością oraz służy organizacji projektu po stronie serwera w oparciu o architekturę MVC (Model-View-Controller).

Express.js umożliwia obsługę dużej ilości żądań klientów w czasie rzeczywistym, co sprawia, że często wykorzystywany jest do budowania czatów na żywo. Dodatkowo charakteryzuje się dużą elastycznością, ponieważ pozwala na przetwarzanie żądań w tak zwanej warstwie pośredniej, umożliwiając dowolne umieszczenie żądania w łańcuchu żądań. Umożliwia także m.in.: zarządzanie ciasteczkami, logowanie do serwisu oraz pracę w trybie sesji.

2.4 JSON

JSON (JavaScript Object Notation) reprezentuje prosty format służący do przekazywania danych [7]. Zapisywanie oraz odczytywanie danych w tym formacie jest łatwe do opanowania przez ludzi. Z łatwością odczytują go i generują komputery. Definicja JSON opiera się na podzbiorze języka programowania JavaScript, zgodnie ze Standardem ECMA-262 3rd Edition z grudnia 1999 roku. JSON jest formatem tekstowym, całkowicie niezależnym od języków programowania, ale korzystającym z konwencji znanym programistom korzystającym
z języków z rodziny C, w tym C++, C#, Java, JavaScript, Perl, Python i wiele innych.

2.5 HTML

HTML to technologia, służąca do tworzenia stron internetowych [8]. Jest to hipertekstowy język znaczników (HyperText Markup Language). Po rozłożeniu skrótu HTML na czynniki pierwsze, jego poszczególne części oznaczają:

- HyperText (hipertekst) – znany jest także pod nazwą hiperłącza. To rodzaj tekstu zawierający odniesienie do innych stron lub treści. Umożliwia użytkownikowi szybkie przemieszczanie się w sieci za pomocą jednego kliknięcia myszy.
- Markup (znacznik) – narzędzie określane także jako tag. Służy do regulowania wyglądu strony i jej funkcji.
- Language (język) – jak każdy język, HTML opiera się na unikalnej składni i alfabecie.

Warto zauważyć, że HTML nie jest językiem programowania. W porównaniu do nich nie zawiera on logiki programowania, instrukcji warunkowych, obliczeń, obsługi zdarzeń, deklaracji zmiennych, funkcji, modyfikacji ani manipulacji danymi itp. Nie posiada także zdolności do pobierania danych wejściowych ani generowania danych wyjściowych.

2.6 CSS

CSS to język używany do opisu układu elementów na stronie internetowej [9]. Za jego pomocą można definiować różne parametry: kolory, wielkość marginesów i odstępy międzywierszowych, style czcionek i wiele innych. CSS jest odpowiedzialny wyłącznie za prezentację strony. Z poziomu jednego arkusza stylów CSS można kontrolować układ graficzny wielu dokumentów. CSS jest językiem uniwersalnym, obsługiwanym przez większość przeglądarek internetowych. Ułatwia to dostosowanie stron do prawidłowego wyświetlania na urządzeniach klienckich.

2.7 Express session

Express session to moduł w środowisku Node.js i frameworku Express.js, który służy do obsługi sesji użytkowników w aplikacjach internetowych [19]. Sesje są mechanizmem przechowywania informacji o stanie użytkownika między różnymi żądaniami oraz odpowiedziami serwera.

Główne cele to:

- Zachowanie stanu: Gdy użytkownik zaloguje się, informacje o zalogowaniu mogą być przechowywane w sesji, aby aplikacja pamiętała, że użytkownik jest zalogowany.
- Bezpieczeństwo: Dane przechowywane w sesji są przechowywane po stronie serwera, co zapewnia większe bezpieczeństwo niż przechowywanie ich po stronie klienta.
- Zarządzanie sesją: Umożliwia konfigurowanie różnych opcji dotyczących sesji, takich jak czas jej życia.

2.8 REST

REST (Representational State Transfer) to styl architektury oprogramowania, opierający się o zbiór określonych reguł opisujących jak definiowane są zasoby, a także umożliwiających dostęp do nich [10]. Został zaprojektowany przez Roya Fieldinga w 2000 roku. Warto zaznaczyć, że REST to styl architektury oprogramowania, a nie standard [10].

Aby API można nazwać RESTful lub API REST-owym musi ono spełniać następujące założenia:

- Bezstanowość - oznacza to, że każde zapytanie od klienta musi zawierać komplet informacji. Serwer nie przechowuje stanu sesji użytkownika. W architekturze REST nie istnieją takie pojęcia jak stany użytkownika czy sesje. W przypadku, gdy serwer wymaga uwierzytelnionego zapytania, konieczne jest, aby zawierało ono wszystkie niezbędne dane do uwierzytelnienia użytkownika.
- Odseparowanie interfejsu użytkownika od operacji na serwerze - REST API powinno być jedynym wymaganym sposobem na wykonywanie przez klienta zapytań do aplikacji serwerowej. Każde zapytanie od klienta musi mieć wszystkie konieczne informacje do wykonania zapytania lub polecenia. Proces ten działa również w odwrotną stronę, gdy serwer dostarcza klientowi odpowiedzi tylko na zapytania, które zostały mu wysłane.

- Endpointy, czyli adresy zasobów powinny jednoznacznie wskazywać do jakiego zasobu się odwołują. Ze struktury endpointu powinno jednoznacznie wynikać, jakie działanie zostanie podjęte na serwerze.
- REST zakłada wykorzystanie cache, który jest mechanizmem umożliwiającym znacznie szybsze uzyskanie odpowiedzi w przypadku powtarzających się zapytań.
- Separacja warstw oznacza, że należy oddzielić warstwy dostępu do danych, logiki biznesowej oraz prezentacji.
- Podział na aplikacje klient-serwer zwiększa możliwości skalowania, rozszerzalności oraz przenośności.
- Możliwość udostępnienia skryptów wykonywanych użytkownikom. Reguła ta zwana jest kodem na żądanie.

2.9 CRUD

CRUD to standardowy model operacji na danych w aplikacjach webowych [11]. Jest to skrót od czterech podstawowych działań: Create, Read, Update, Delete. Jego celem jest umożliwienie intuicyjnego i łatwego dostępu do danych. Może być wykorzystany do operacji na różnych rodzajach danych, w tym na relacyjnych bazach danych, plikach lub innych źródłach danych.

Działanie	**Rodzaje zapytań w SQL**	**HTTP**
Create	INSERT	PUT/POST
Read	SELECT	GET
Update	UPDATE	POST/PUT/PATCH
Delete	DELETE	DELETE

Tabela 2.1 Rodzaje zapytań w SQL

2.10 MySQL

MySQL jest popularnym open sourcowym systemem zarządzania relacyjną bazą danych [20]. Jest on częścią popularnego pakietu oprogramowania służącego do rozwoju aplikacji webowych zwanego LAMP.

MySQL powstał w roku 1995 na terenie Szwecji. Firma, której właścicielem był Michael Widenius, dystrybuowała MySQL przez pierwsze pięć lat na płatnej licencji i dopiero w 2000 roku przeszła na open source. Od tego czasu MySQL rósł w popularności aż osiągnął swoją obecną pozycję jako drugiego najbardziej popularnego systemu zarządzania bazą danych.

Zalety MySql:

- Skalowalność: MySQL charakteryzuje się efektywną obsługą ogromnej liczby zapytań, radząc sobie nawet z milionem zapytań na sekundę.
- Uniwersalność: MySQL jest dostępny na wielu systemach operacyjnych, takich jak Linux, OS X, Windows, czy nawet na mniej powszechnie używanych platformach, np. OpenBSD czy Symbian
- Ochrona danych: MySQL oferuje mechanizmy ochrony danych takie jak uwierzytelnienie użytkownika, wsparcie dla protokołów SSH czy SSL, granularna struktura dostępu, która pozwala mieć dostęp do tylko wybranych danych.

2.11 bCrypt

BCrypt, będący jednym z najpopularniejszych algorytmów haszowania, jest powszechnie wykorzystywany do zabezpieczania haseł oraz innych poufnych informacji [21].

W 1999 roku Niels Proger opracował algorytm BCrypt, który przedstawił na konferencji USENIX w styczniu tego samego roku w San Antonio w Teksasie.

Konieczność bezpiecznego przechowywania haseł w systemach UNIX stanowiła początek historii algorytmu BCrypt, który powstał jako odpowiedź na zagrożenia wynikające z przechowywania haseł w postaci jawnej lub za pomocą prostych funkcji.

Po wielu miesiącach pracy Proger wprowadził na rynek bCrypt, który był oparty na algorytmie Blowfish, który został zaprojektowany przez Bruce'a Schneiera w 1993 roku. Algorytm Blowfish, będący szyfrem symetrycznym, pozwalał na używanie tego samego klucza do zarówno szyfrowania, jak i deszyfrowania danych. Jednak Proger wykorzystał algorytm w sposób innowacyjny, tworząc funkcję haszującą, która była bezpieczna i wydajna.

2.12 Fetch API

Fetch API jest interfejsem, który pozwala na asynchroniczne pobieranie danych [22]. Fetch API powstało, aby uprościć sposób komunikacji z serwerem. Pozwala na pisanie ładniejszego kodu JavaScript i uniknięcie tak zwanego callback hell.

Przez ponad dekadę użycie obiektu XMLHttpRequest było jedynym środkiem do zainicjowania asynchronicznego żądania w JavaScript. Rozwiązanie to jednak miało dużo wad. Fetch API rozwiązuje większość problemów związanych z użyciem XHR. Wprowadza do języka JavaScript obiekty Request, Response, Headers i Body.

Podstawą pracy z fetch API jest użycie metody fetch() do asynchronicznego pobierania danych z serwera. Oprócz używania metody GET do pobierania danych, możemy także korzystać z innych metod HTTP, takich jak POST, PUT, DELETE.

2.13 HTTP

Początki protokołu HTTP sięgają lat 90 XX wieku [14]. W 1991 roku pojawiła się pierwsza wersja protokołu HTTP/0.9, a następnie ewoluowała do HTTP/1.0. Ta wersja umożliwiała podczas jednego połączenia wysyłanie i realizację tylko jednego żądania. Wraz z postępem technologii protokół ten ewoluował, dostosowując się do zmieniających się potrzeb i przekształcając się w obecnie najczęściej stosowany HTTP/1.1.

HTTP to protokół warstwy aplikacji, funkcjonujący w oparciu o model klient-serwer [15]. Klient wysyła żądanie HTTP do serwera, a serwer odpowiada na to żądanie, przesyłając odpowiednie dane. Protokół ten oparty jest na protokole TCP/IP, który umożliwia bezpieczną oraz niezawodną komunikację w sieci.

Podstawowa koncepcja funkcjonowania protokołu HTTP opiera się na przekazywaniu danych poprzez żądania i odpowiedzi [31]. Żądanie składa się z trzech kluczowych składników:

- Metoda - określa rodzaj żądania
- Adres URL - określa lokalizację zasobu, do którego odnosi się żądanie.
- Nagłówki - zawierają dodatkowe informacje o żądaniu, takie jak typ danych, język itp.

Metoda HTTP	**Działanie**
GET	pobieranie danych
POST	przesyłanie danych
PUT	aktualizacja danych
DELETE	usuwanie danych

Tabela 2.2 Rodzaje żądań [opracowanie własne]

Serwer odbiera żądanie i zwraca odpowiednią odpowiedź HTTP. Odpowiedź składa się z trzech głównych elementów:

- Kod stanu - informuje o rezultacie żądania.
- Nagłówki - zawierają dodatkowe informacje o odpowiedzi, takie jak typ danych itp.
- Treść - aktualne dane przesyłane w odpowiedzi.

Kod stanu	**Treść**
200	sukces
301	zasób został przeniesiony na stałe do innego miejsca
404	brak zasobu
500	błąd serwera

Tabela 2.3 Kody stanu [opracowanie własne]

2.14 HTTPS

HTTPS (Hypertext Transfer Protocol Secure) jest to bezpieczny protokół przesyłania danych tekstowych [16]. Bezpieczeństwo danych w transmisji jest osiągane dzięki zastosowaniu protokołów SSL lub TLS. Wszelkie informacje wysyłane między klientem, a serwerem są szyfrowane w obie strony. Protokół SSL/TLS składa się z trzech głównych warstw zabezpieczeń:

- Uwierzytelnienie - dostarcza zabezpieczenie przed atakami kryptograficznymi, które obejmują przechwytywanie komunikatów przesyłanych między serwerem, a klientem.
- Integralność danych - pozwala na wykrywanie zmian i uszkodzeń danych, które mogą wystąpić w trakcie ich przesyłania.
- Szyfrowanie - koduje dane, eliminując obawy użytkownika, że ktoś mógłby monitorować jego działania.

W przypadku HTTP wszystko, co jest wysyłane i odbierane przez ten protokół, ma postać zwykłego tekstu, co sprawia, że jest podatne na przechwycenie. Natomiast w przypadku HTTPS warstwa zabezpieczeń SSL/TLS szyfruje dane, dostarczając im ochrony przed nieautoryzowanym dostępem.

2.15 Body-parser

Body-parser to moduł middleware w środowisku Node.js, który pomaga w parsowaniu danych przesyłanych w request body [23]. W przypadku aplikacji internetowych, przeglądarki mogą przesyłać dane do serwera na różne sposoby na przykład poprzez formularze HTML, JSON i inne. Body-parser ułatwia obsługę tych danych, sprawiając, że są one dostępne łatwo w kodzie aplikacji.

Jego funkcjonalność nie tylko skupia się na parsowaniu danych, ale także na upraszczaniu procesu ich interpretacji i wykorzystania w aplikacji. Dzięki body-parserowi, deweloperzy mogą skoncentrować się na logice biznesowej bez konieczności bezpośredniego zajmowania się detalicznymi aspektami obsługi przesyłanych danych.

2.16 Swagger

Swagger to zestaw narzędzi open-source zbudowanych wokół specyfikacji OpenAPI, które pomagają w projektowaniu, budowaniu oraz dokumentowaniu interfejsów REST API [24]. Plik OpenAPI pozwala na opisanie całego API, a w tym:

- metody uwierzytelniania,
- parametry wejścia i wyjścia dla każdej operacji,
- informacje kontaktowe, licencje.

2.17 JEST

Jest to framework do testowania w języku JavaScript, który jest często używany do testowania kodu frontendowego i backendowego [26]. Jest dostarcza narzędzia do testowania jednostkowego, integracyjnego oraz tworzenia testów akceptacyjnych.

2.18 Supertest

Supertest to biblioteka do testowania API w języku JavaScript/Node.js [27]. Jest oparta na module ‘superagent’ i umożliwia testowanie API poprzez wysyłanie żądań HTTP oraz sprawdzanie odpowiedzi.

3. Architektura systemu

W poniższym rozdziale przedstawiona zostanie architektura, funkcje oraz aktorzy systemu.

3.1 Funkcje systemu

W poniższym podrozdziale zostaną omówione funkcje, które są realizowane przez wybrane części systemu.

3.1.1 Aplikacja webowa

Aplikacja webowa posiada dwóch aktorów:

- użytkownik niezalogowany (gość),
- użytkownik zalogowany.

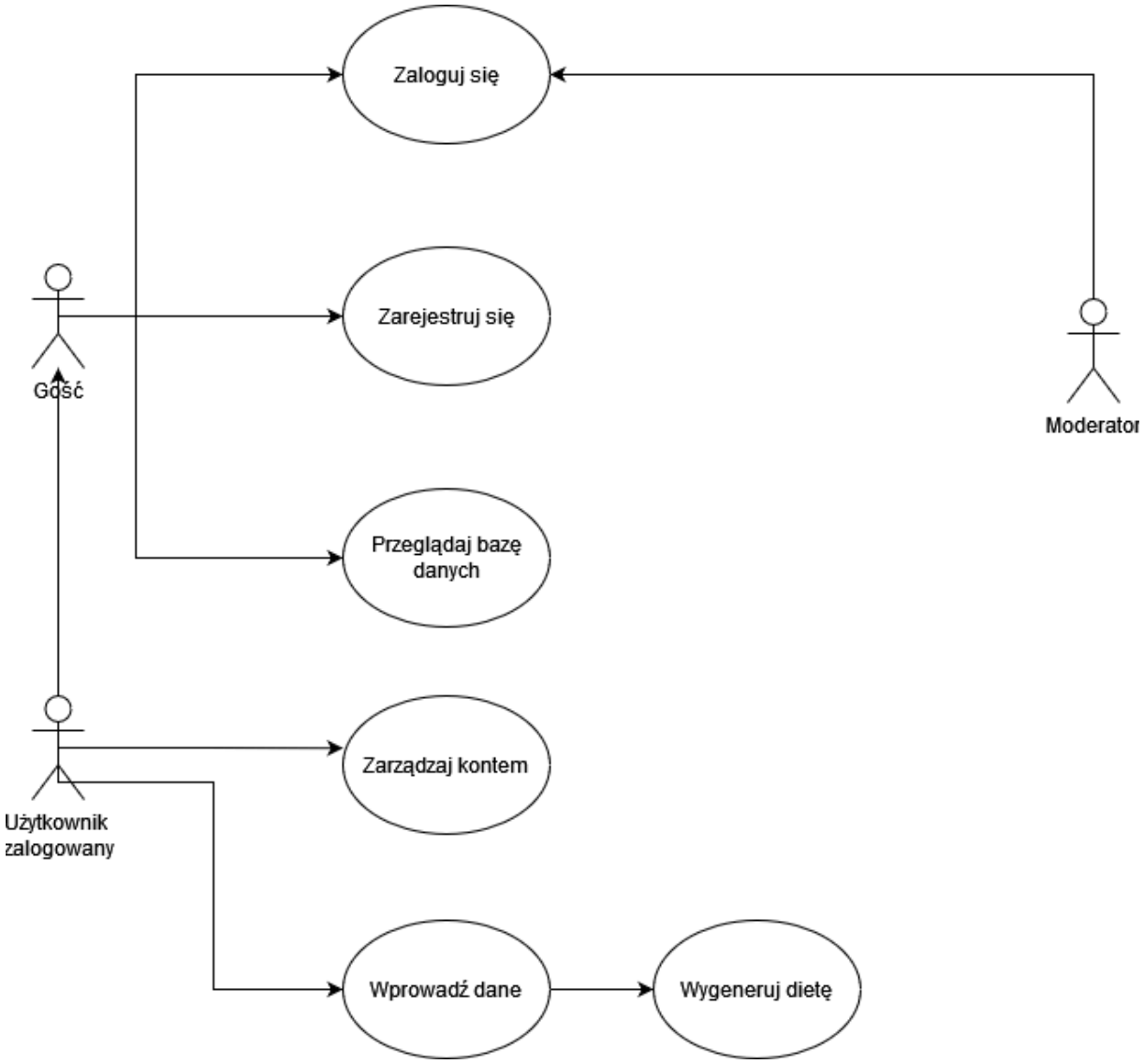
Gość nie ma możliwości skorzystania z żadnej z głównych funkcji serwisu do momentu rejestracji. Po pomyślnym logowaniu, użytkownik zostaje przeniesiony na stronę główną, na której uzyskuje dostęp do wszystkich funkcjonalności serwisu.

Użytkownik zalogowany ma do dyspozycji stronę główną, na której znajduje się pasek menu. Widnieje na nim opcja wygenerowania planu żywieniowego oraz przejścia do bazy danych zawierającej informacje o składnikach i wartościach odżywczych różnych produktów.

W momencie wybrania pierwszej opcji użytkownik zostaje przeniesiony do strony, na której po kolei uzupełnia informacje na temat swojego zdrowia. W rezultacie otrzymuje wygenerowany plan żywieniowy dostosowany do jego potrzeb.

3.1.2 Diagram przypadków użycia

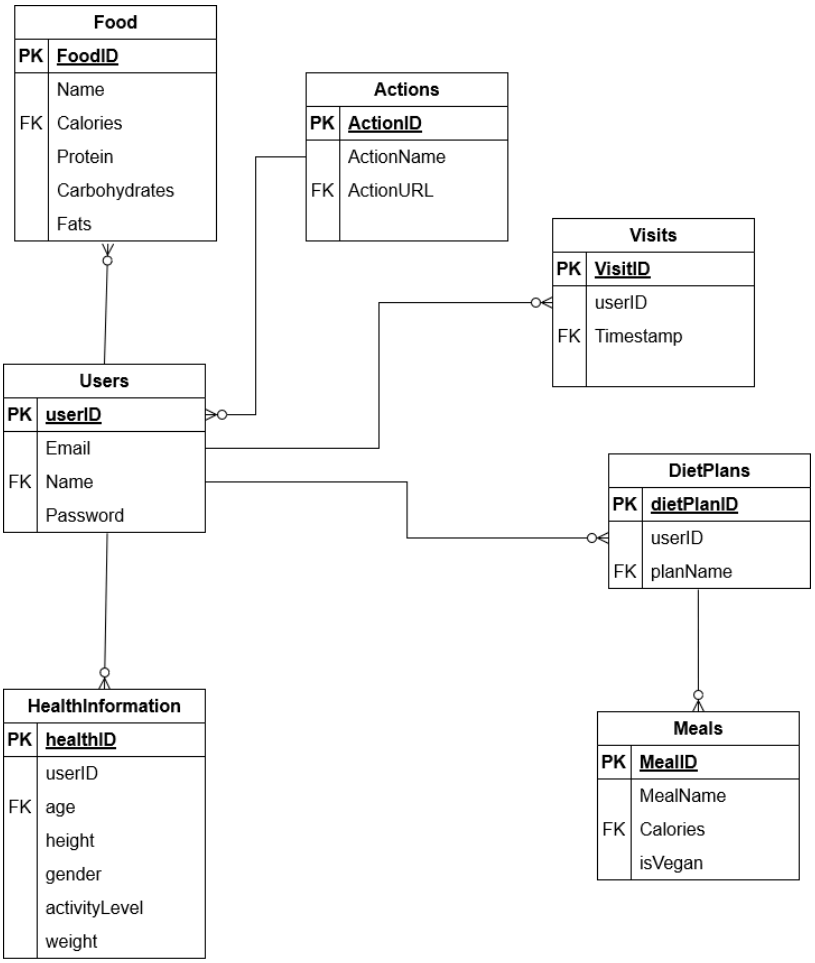
Diagram przypadków użycia czyli diagram, który przedstawia funkcjonalność systemu wraz z jego otoczeniem.



Rys 3.1 Diagram przypadków użycia [opracowanie własne]

3.1.3 Diagram ERD

Diagram ERD (ang. Entity Relationship Diagram), czyli diagram związków encji to graficzny sposób przedstawienia powiązań między encjami w modelu bazy danych.



Rys 3.2 Diagram ERD [opracowanie własne]

Diagram przedstawiony na rysunku 3.2 ukazuje strukturę bazy danych dla asystenta dietetycznego, gdzie tablica “Users” (użytkownicy) stanowi główną encję przechowującą dane logowania. Współdziała ona z tablicą “HealthInformation” (dane zdrowotne), będącą encją, która zbiera dane wprowadzone przez użytkownika w momencie wypełniania formularza potrzebnego do utworzenia spersonalizowanej diety. Połączenie pomiędzy tabelami “Food” (jedzenie), a “Users” (użytkownicy) umożliwia niezalogowanemu użytkownikowi przeglądanie bazy produktów spożywczych wraz z ich wartościami odżywczymi.

Dodatkowo, tabela “Users” (użytkownicy) jest połączona z encjami “Actions” (akcje) oraz “Visits” (wizyty) reprezentującymi aktywność użytkowników na stronie. To powiązanie pozwala na śledzenie i analizę interakcji użytkowników z aplikacją.

Tabela “Users” (użytkownicy) połączona jest również z tabelą “DietPlans” (plany dietetyczne), co umożliwia przypisanie planów dietetycznych do konkretnych użytkowników. Z kolei tabela “DietPlans” (plany dietetyczne) jest połączona z tabelą “Meals” (posiłki), co pozwala dopasowanie posiłków do konkretnego rodzaju planu dietetycznego.

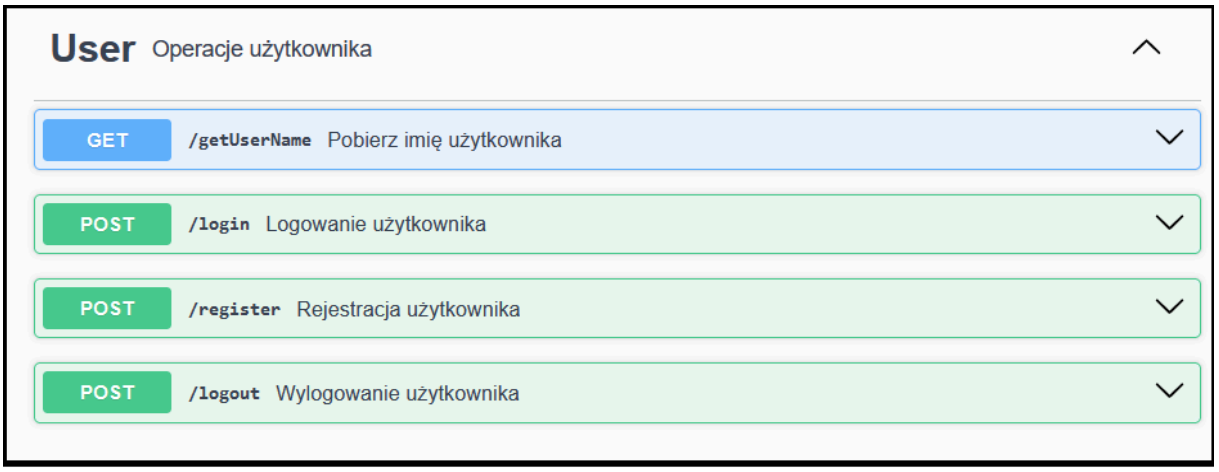
3.1.4 Dokumentacja API

Serwer przetwarza zapytania od aplikacji webowej za pomocą REST API. Podejście to opiera się na obsłudze zapytań przez kontrolery, które są endpointami dostępnymi dla aplikacji klienta poprzez protokół HTTP.

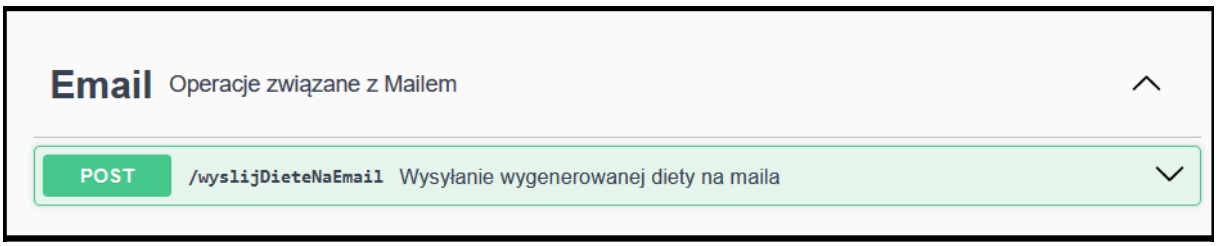
Wyróżniamy cztery główne metody zapytań REST:

- GET - używane do pobierania zasobów.
- POST - służy do tworzenia nowego zasobu.
- PUT - wykorzystywane do modyfikacji istniejącego zasobu.
- DELETE - używane do usuwania istniejącego zasobu.

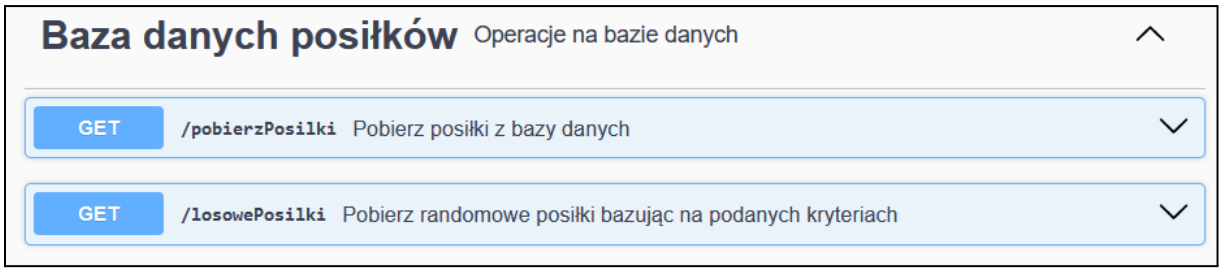
Na rysunkach 3.3, 3.4 oraz 3.5 przedstawiony jest schemat endpointów wygenerowanych za pomocą narzędzia Swagger.



Rys 3.3 Operacje użytkownika [opracowanie własne]



Rys 3.4 Operacje związane z wysyłką email [opracowanie własne]



Rys 3.5 Operacje na bazie danych [opracowanie własne]

3.1.5 Aplikacja serwerowa

Aplikacja serwerowa ma za zadanie przyjmować zapytania od aplikacji internetowej. Jej podstawowym zadaniem jest autoryzacja użytkowników i przesyłanie ich danych.

Dla użytkowników nieautoryzowanych:

- rejestracja użytkownika,
- logowanie użytkownika,
- dostęp do bazy danych posiłków.

Dla użytkowników autoryzowanych:

- dostęp do spersonalizowanych ustawień,
- dostęp do bazy danych posiłków.

3.1.6 Baza danych

Baza danych pełni kluczową rolę w przechowywaniu, zarządzaniu i udostępnianiu danych w systemie. Stanowi ona centralne repozytorium, w którym przechowywane są precyzyjnie zorganizowane informacje dotyczące składników oraz wartości odżywczych. Pozwala na efektywne gromadzenie i kontrolowanie informacji z których korzysta cała aplikacja.

Dla użytkowników nieautoryzowanych aplikacja oferuje dostęp do informacji na temat składników i wartości odżywczych różnych produktów.

Użytkownikom autoryzowanym aplikacja oferuje również dostęp do informacji o składnikach i wartościach odżywczych produktów oraz szerszy dostęp do różnego rodzaju potraw.

4. Interfejsy użytkownika

W tym rozdziale przedstawione zostały najważniejsze interfejsy użytkownika aplikacji wraz z opisami.

4.1 Ekran powitalny

Rys 4.1 Ekran powitalny [opracowanie własne]

Na rysunku 4.1 przedstawiony został widok ekranu ukazującego się użytkownikowi po pierwszym wejściu na stronę. Funkcje, które dostępne są z poziomu tego widoku to:

- przejście do ekranu logowania oraz rejestracji,
- przejście do aplikacji jako użytkownik niezalogowany.

Użytkownik niezalogowany nie uzyskuje dostępu do wygenerowania własnego planu dietetycznego.

4.2 Baza danych

Baza Danych

Wartości Odżywcze

Nazwa	Kalorie	Białko	Węglowodany	Tłuszcze
Ananas	50	0.5	13	0.1
Arbuz	30	0.6	8	0.2
Bakłażan	25	1	6	0.2
Bakłażan	25	1	6	0.2
Brokuł	34	2.8	7	0.4
Brokuł	34	2.8	7	0.4
Cebula	40	1.1	9.3	0.1
Cynamon	247	3.2	80	1.2
Cytryna	29	1.1	9	0.3
Czosnek	149	6.4	33	0.5
Dynia	26	1	7	0.1
Gruszka	57	0.4	15	0.2
Jabłko	52	0.3	14	0.2

Rys 4.2 Baza danych [opracowanie własne]

Na rysunku 4.2 przedstawiony został widok bazy danych. Jest to jedyny ekran dostępny dla niezalogowanego użytkownika. Z tego miejsca możliwe jest wciąż przejście do ekranu logowania oraz rejestracji.

Sama baza danych przedstawia informacje o składnikach i wartościach odżywczych różnych produktów z opcją sortowania po kliknięciu na wybraną kategorie w tabeli.

4.3 Rejestracja

4.3.1 Ekran rejestracji

Rys 4.3 Ekran rejestracji [opracowanie własne]

Na rysunku 4.3 przedstawiony został ekran rejestracji użytkownika.

4.3.2 Ekran rejestracji - błąd

Rys 4.4 Błąd rejestracji [opracowanie własne]

Na rysunku 4.4 przedstawiony został widok okna rejestracji w momencie, gdy użytkownik wprowadzi adres e-mail, który istnieje już w bazie danych. Możliwe jest wtedy szybkie przejście do ekranu logowania lub zmiana wcześniej wpisanego adresu e-mail na inny.

4.4 Logowanie

4.4.1 Ekran logowania

Rys 4.5 Ekran logowania [opracowanie własne]

Na rysunku 4.5 przedstawiony został widok ekranu logowania.

4.4.2 Ekran logowania - błąd

Rys 4.6 Błąd logowania [opracowanie własne]

Na rysunku 4.6 przedstawiono widok okna logowania po nieprawidłowym wprowadzeniu adresu e-mail lub hasła. W tej sytuacji użytkownik ma trzy dostępne opcje: po pierwsze, może przejść bezpośrednio do ekranu rejestracji, co jest wygodną alternatywą w przypadku chęci założenia nowego konta. Po drugie, istnieje opcja zmiany wprowadzonych wcześniej danych na poprawne, umożliwiając użytkownikowi ponowną próbę logowania bez konieczności powrotu do głównego ekranu logowania. Alternatywą trzecią jest podanie adresu e-mail, skorzystanie z opcji "zapomniałem hasła" i otrzymanie nowego, wygenerowanego hasła na podany adres e-mail.

4.5 Strona główna

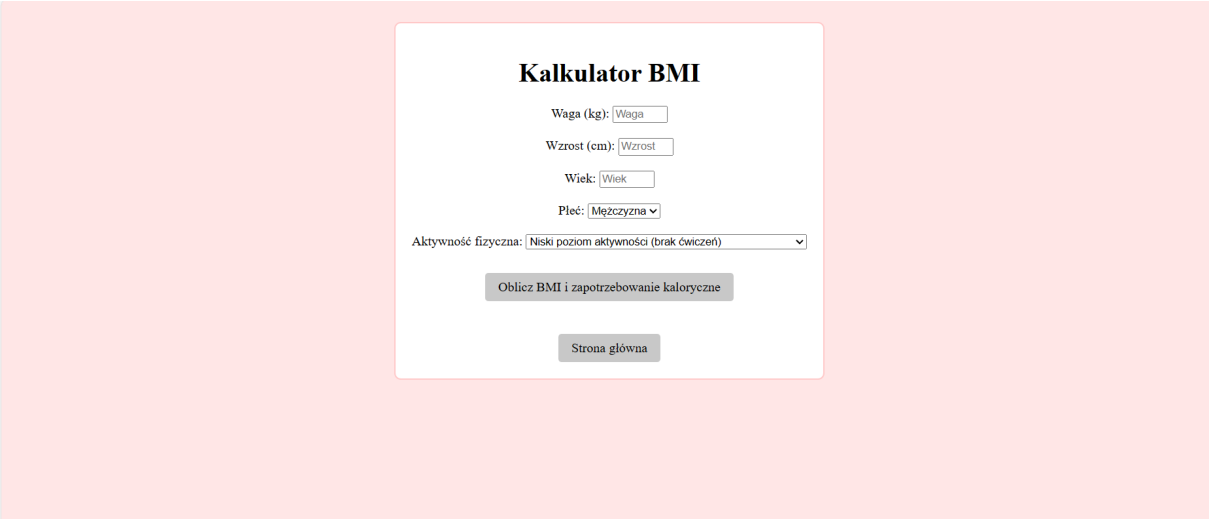
Rys 4.7 Strona główna [opracowanie własne]

Użytkownik po zalogowaniu dostaje dostęp do głównego ekranu aplikacji. Funkcje dostępne z poziomu tego widoku to:

- przejście do generatora planu dietetycznego,
- przejście do bazy danych zawierającej informacje o składnikach i wartościach odżywczych różnych produktów,
- możliwość wylogowania się.

4.6 Kalkulator BMI

4.6.1 Kalkulator BMI - formularz



Rys 4.8 Formularz BMI [opracowanie własne]

Na rysunku 4.8 przedstawiony został widok kalkulatora BMI oraz zapotrzebowania kalorycznego użytkownika [17]. Ten przejrzysty interfejs stanowi bramę do dokładnych obliczeń związanych z indeksem masy ciała oraz podstawowym zapotrzebowaniem kalorycznym.

Do obliczenia podstawowego zapotrzebowania kalorycznego używamy metody Harrisa-Benedicta. Jest ona jedną z najpopularniejszych metod ze względu na uwzględnienie współczynnika aktywności fizycznej.

Zastosowane zostały dwa różne równania w zależności od płci użytkownika:

- Dla mężczyzn

$$BMR = 66 + (13.7 \times waga) + (5 \times wzrost \times 100) - (6.76 \times wiek)$$

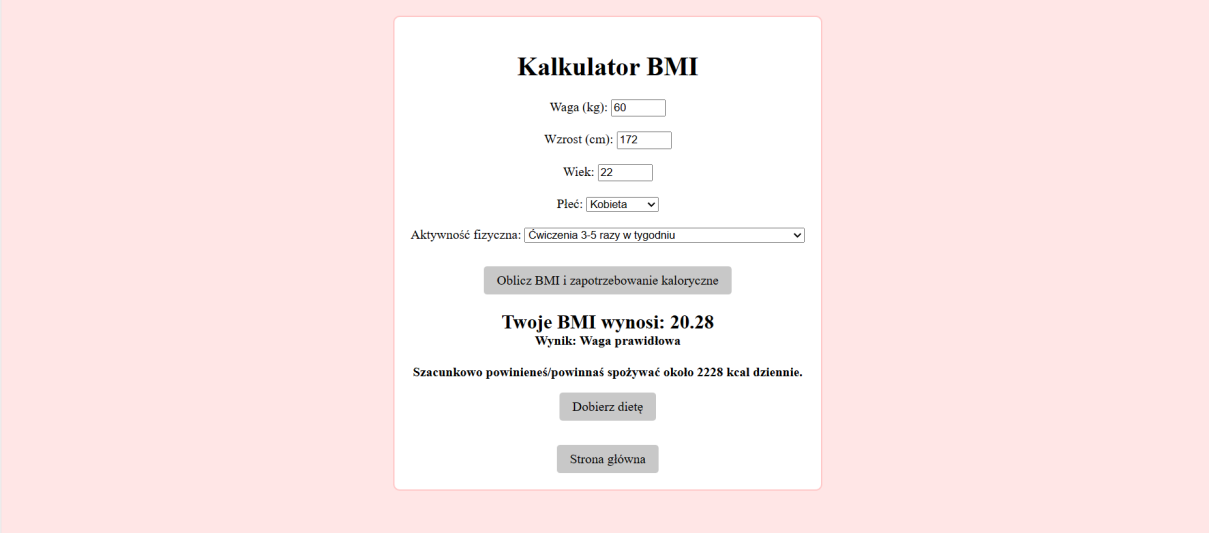
- Dla kobiet

$$BMR = 655 + (9.6 \times waga) + (1.8 \times wzrost \times 100) - (4.7 \times wiek)$$

Następnie oblicza się całkowitą kaloryczność, mnożąc wynik BMR przez odpowiedni współczynnik aktywności. Prezentują się one następująco:

- Niski poziom aktywności = 1.2,
- Ćwiczenia 1-3 razy w tygodniu = 1.375,
- Ćwiczenia 3-5 razy w tygodniu = 1.55,
- Ćwiczenia 6-7 razy w tygodniu = 1.725,
- Aktywność bardzo wysoka (ćwiczenia i praca fizyczna) = 1.9.

4.6.2 Kalkulator BMI - obliczanie BMI oraz zapotrzebowania kalorycznego



Rys 4.9 Kalkulator BMI i wyniki [opracowanie własne]

W przedstawionym na rysunku 4.9 interfejsie widoczny jest kalkulator BMI, a także wyniki, które ujawniają się po kliknięciu opcji “Oblicz BMI i zapotrzebowanie kaloryczne”. Ten funkcjonalny panel umożliwia użytkownikowi wprowadzenie wszystkich niezbędnych danych dotyczących masy ciała i wzrostu, a następnie prezentuje informacje dotyczące jego stanu zdrowia.

Po wprowadzeniu danych, użytkownik otrzymuje szczegółowe rezultaty, w tym wskaźnik BMI oraz informacje odnośnie zapotrzebowania kalorycznego. Dodatkowo, w momencie uzyskania wyników, odblokowana zostaje opcja dostosowania diety. Użytkownik może skorzystać z tej funkcji, aby uzyskać spersonalizowane zalecenia żywieniowe, dostosowane do jego indywidualnych potrzeb zdrowotnych.

4.7 Wygenerowane diety

Rys 4.10 Wygenerowana dieta [opracowanie własne]

Na rysunku 4.10 przedstawiony został widok wygenerowanych diet. Na podstawie wcześniej wprowadzonych danych, aplikacja tworzy dwie spersonalizowane diety jak najbardziej zgodne z zapotrzebowaniem kalorycznym użytkownika.

Istnieje również możliwość wysłania wygenerowanej diety na e-maila podanego przy logowaniu.

5. Testy

Test jednostkowy to rodzaj testu oprogramowania, który sprawdza pojedynczą jednostkę kodu [25]. Celem testów jest sprawdzenie, czy jednostka działa zgodnie z ich oczekiwaniami w izolacji od innych części systemu.

Test integracyjny to rodzaj testu oprogramowania, który ocenia poprawność współpracy pomiędzy modułami systemu lub jego komponentami [29].

5.1 Testy API i wykorzystane narzędzia

W tym podrozdziale zostaną przedstawione wyniki wykonanych testów jednostkowych oraz integracyjnych dla najważniejszych funkcji systemu.

Do wykonania testów skorzystano z technologii Jest oraz Supertest. Służą one do wykonywania testowania w języku JavaScript kodu frontendowego i backendowego.

5.2 Testy funkcji

5.2.1 Test endpointu pobierającego posiłki

Test endpointu "/pobierzPosiłki" tworzy imitację żądania GET na adres "http://localhost:3000/pobierzPosilki" za pomocą biblioteki "nock", zwracając symulowane dane z bazy danych. Następnie test wysyła żądanie GET na ten sam endpoint, oczekując odpowiedzi o statusie 200 oznaczającej, że żądanie zostało pomyślnie przetworzone i serwer zwrócił odpowiedź. Dodatkowo, sprawdzane jest, czy serwer zwrócił odpowiedź w formacie JSON. W rezultacie test potwierdza poprawność funkcjonowania endpointu "/pobierzPosiłki".

```
test('Test endpointu /pobierzPosilki', async () => {
  nock('http://localhost:3000')
    .get('/pobierzPosilki')
    .reply(200, [{ /*dane z bazy danych */ }]);
  await request(app)
    .get('/pobierzPosilki')
    .expect(200)
    .expect('Content-Type', /json/)
    .then(response => {
      expect(response.body).toHaveLength(166);
    });
});
```

Listing 5.1 Test żądania pobierającego posiłki z bazy danych [opracowanie własne]

5.2.2 Test endpointu pobierającego nazwę użytkownika

Test ocenia poprawność działania endpointu "/getUserName", oczekując, że zwróci on status HTTP 200 oznaczający, że żądanie zostało pomyślnie przetworzone i serwer zwrócił odpowiedź. Dodatkowo, sprawdzane jest, czy serwer zwrócił odpowiedź w formacie JSON.

Po wysłaniu żądania GET, sprawdza czy odpowiedź zawiera wartość "UserName". Następnie oczekuje, że wartość tego pola będzie równa "null" (użytkownik niezalogowany), co weryfikuje poprawność działania endpointu.

```
test('Test endpointu /getUserName', async () => {
  await request(app)
    .get('/getUserName')
    .expect(200)
    .expect('Content-Type', /json/)
    .then(response => {
      expect(response.body).toHaveProperty('userName');
      expect(response.body.userName).toBeNull();
    });
});
```

Listing 5.2 Test żądania pobierającego nazwę użytkownika z bazy danych [opracowanie własne]

5.2.3 Test endpointu logowania

Test endpointu "/login" sprawdza poprawność procesu logowania. Na początku, za pomocą funkcji request wysyłane jest żądanie POST na endpoint "/login", przekazując dane logowania. Test oczekuje odpowiedzi ze statusem 302 oznaczającym przekierowanie.

Następnie wysyłane jest żądanie GET na stronę "/start.html" oraz sprawdzane, czy otrzymana strona ma typ zawartości HTML. Dzięki funkcji then test analizuje treść strony, upewniając się, że zawiera ona oczekiwane frazy.

```
test('Test endpointu /login', async () => {
  const { app } = getApp();
  await request(app)
    .post('/login')
    .send({ email: 'test@example.com', password: 'testpassword' })
    .expect(302)
    .expect('Location', '/start.html')
    .then(response => {
      return request(app)
        .get('/start.html')
        .expect('Content-Type', /html/)
        .then(finalResponse => {
          expect(finalResponse.text).toContain('Witaj!');
          expect(finalResponse.text).toContain('Udane zarejestrowanie');
        });
    });
});
```

Listing 5.3 Test żądania logującego użytkownika [opracowanie własne]

5.2.4 Test endpointu rejestracji

Test endpointu "/register" ocenia poprawność procesu rejestracji użytkownika. W pierwszym kroku poprzez funkcję request wysyłane jest żądanie POST z danymi tekstowymi. Oczekuje się, że odpowiedź ma status 302 oznaczający przekierowanie i kieruję się na stronę index.html.

Kolejny krok testu polega na wysyłaniu żądania GET na adres przekierowania. Test sprawdza, czy strona ma odpowiedni typ zawartości. Za pomocą funkcji then analizuje treść tej strony, upewniając się, że zawiera wymagane frazy.

W wyniku tych kroków, test potwierdza, że proces rejestracji działa poprawnie.

```
test('Test endpointu /register', async () => {
  const { app } = getApp();
  await request(app)
    .post('/register')
    .send({ email: 'test@example.com', name: 'Test User', password: 'testpassword' })
    .expect(302)
    .expect('Location', '/index.html?registered=true')
    .then(response => {
      return request(app)
        .get(response.header['location'])
        .expect('Content-Type', /html/)
        .then(finalResponse => {
          expect(finalResponse.text).toContain('Asystent Dietetyczny');
          expect(finalResponse.text).toMatch(/<!DOCTYPE html>/);
        });
    });
});
```

Listing 5.4 Test żądania rejestrującego użytkownika [opracowanie własne]

5.2.5 Test endpointu wylogowywania

Test endpointu “/logout” sprawdza poprawność procesu wylogowywania się. Na początku wysyłane jest żądanie POST na endpoint “/logout” i oczekuje odpowiedzi. Następnie test sprawdza, czy w nagłówku odpowiedzi istnieje ustawione ciasteczko sesji.

Dalej analizowane jest czy ciasteczko sesji jest obecne w nagłówku. Jeśli nie jest obecne,oznacza to, że wylogowanie udało się. Oczekuję się wtedy statusu odpowiedzi 200 (OK). W przeciwnym razie, jeśli ciasteczko sesji nadal istnieje, oznacza to, że wystąpił problem i test oczekuje, że status odpowiedzi nie wyniesie 302, czyli przekierowania.

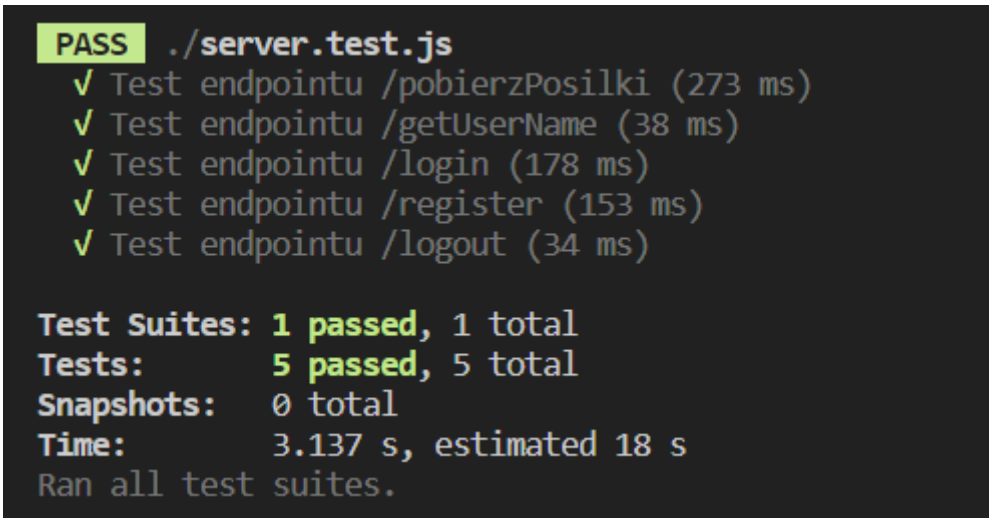
```
test('Test endpointu /logout', async () => {
  const response = await request(app)
    .post('/logout');

  const setCookieHeader = response.header['set-cookie'];
  expect(setCookieHeader).not.toBeFalsy();
  if (!setCookieHeader.includes('connect.sid')) {
    expect(response.status).toBe(200);
  } else {
    expect(response.status).not.toBe(302);
  }
});
```

Listing 5.5 Test żądania wylogowania użytkownika [opracowanie własne]

5.3 Wynik testów

Wszystkie testy zostały zaimplementowane zgodnie z oczekiwaniami. Otrzymane wyniki potwierdzają poprawność funkcji systemu w zakresie testowania endpointów. Przy każdym teście zastosowano symulację danych, co pozwoliło na sprawdzenie reakcji systemu na różne scenariusze.



Rys 5.1 Wynik uruchomienia testów jednostkowych [opracowanie własne]

6. Podsumowanie

Celem pracy inżynierskiej było utworzenie aplikacji webowej, pełniącej funkcję asystenta dietetycznego. Wszystkie z wymagań,które zostały opisane w pierwszym rozdziale zostały zrealizowane.

W chwili obecnej prace, które zostały wykonane nad aplikacją sprawiły, że:

- Aplikacja umożliwia użytkownikom łatwe wprowadzenie swojego aktualnego stanu zdrowia oraz wybranie preferencji żywieniowych. Każdy użytkownik może szybko i wygodnie skonfigurować personalizowane ustawienia zdrowotne
- Użytkownik ma możliwość wygenerować własny plan żywieniowy dostosowany do swoich indywidualnych preferencji. Ponadto, aplikacja umożliwia użytkownikowi powtórne wygenerowanie posiłków, co pozwala na elastyczne dostosowywanie diety do własnych, zmieniających się potrzeb.
- Użytkownik ma dostęp do bazy danych zawierającej informacje o składnikach i wartościach odżywczych różnych produktów.
- Użytkownik ma możliwość na udostępnienie swojej spersonalizowanej diety poprzez wysłanie jej na adres email podany przy procesie logowania.
- Interfejs aplikacji charakteryzuje się spójnością i jednolitością, co sprawia, że poruszanie się pomiędzy różnymi funkcjonalnościami jest niezwykle proste. Przyczynia się to do niezwykle intuicyjnego korzystania z aplikacji.
- Obsługa została dostosowana do różnego typu użytkowników, niezależnie od ich doświadczenia z technologią.

Aplikacja została zaopatrzona w solidne fundamenty , które umożliwiają jej ciągły rozwój. Elastyczna struktura aplikacji sprawia, że jest ona gotowa by w każdym momencie poszerzyć ją o nowe funkcjonalności lub rozwinąć już te istniejące. Dzięki temu podejściu aplikacja jest skalowalna i gotowa na ewentualne zmiany.

Pierwszym etapem rozwoju aplikacji mogłoby być zwiększenie ilości diet, które użytkownik jest w stanie wybrać. Obecnie do wyboru jest opcja wegańska i niewegańska, ale jest to element warty poszerzenia w przyszłości.

Niniejsza praca umożliwiła rozwój umiejętności zdobytych podczas studiów w obszarze projektowania aplikacji internetowych. Badanie aplikacji skoncentrowanej na dietetyce pozwoliło zgłębić wiedzę w zakresie zdrowego stylu życia. Proces tworzenia aplikacji

pozwolił na praktyczne zastosowanie zdobytej wiedzy, która stanowi istotny element współczesnego społeczeństwa stającego się coraz bardziej zainteresowanym zdrowymi nawykami żywieniowymi.

Bibliografia

[1] Eric Freeman, Elisabeth Robson. *Programowanie w JavaScript. Rusz głową!*

[2] Haverbeke Marijn *Zrozumieć JavaScript. Wprowadzenie do programowania* Wydanie III

[3] Larry Ullman *Nowoczesny język JavaScript Kiedyś 6 teraz 3*

[4] Marcin Domański *Kurs Javascript dla superbohaterów* [Online]

https://kursjs.pl/ dostęp 10.2023

[5] JavaScript - dokumentacja techniczna [Online]

https://developer.mozilla.org/pl/docs/Web/JavaScript/ dostęp 10.2023

[6] Node.js [Online]

https://nodejs.org/en/learn/getting-started/introduction-to-nodejs dostęp 10.2023

[7] JSON [Online] https://www.json.org/json-pl.html dostęp 10.2023

[8] HTML [Online]

https://udigroup.pl/blog/jezyk-html-co-to-jest-do-czego-sluzy-jak-wyglada/

dostęp 10.2023

[9] CSS [Online] https://futurecollars.com/slownik_it/css-definicja/ dostęp 10.2023

[10] REST [Online] https://devszczepaniak.pl/wprowadzenie-do-rest-api/ dostęp 11.2023

[11] CRUD [Online]

https://boringowl.io/blog/jak-uzywac-crud-w-aplikacjach-webowych dostęp 11.2023

[12]Motywacja [Online]

https://www.theguardian.com/society/2023/mar/02/more-than-half-of-humans-on-track-to-be-overweight-or-obese-by-2035-report dostęp 12.2023

[13] Motywacja [Online]

https://tvn24.pl/ciekawostki/otylosc-i-nadwaga-ponad-polowa-ludzkosci-bedzie-otyla-do-2035-roku-eksperci-wskazali-kto-tyje-najszybciej-6791583 dostęp 12.2023

[14] HTTP [Online]

https://jakwybrachosting.pl/co-to-jest-protokol-http/ dostęp 12.2023

[15] HTTP [Online]

https://widoczni.com/blog/najwiekszy-slownik-marketingu-internetowego-w-polsce/protokol-http/ dostęp 12.2023

[16] HTTPS [Online] https://semcore.pl/slownik/https/ dostęp 12.2023

[17] Kalkulator BMI [Online]

https://fit.poradnikzdrowie.pl/diety-i-zywienie/zdrowe-odzywianie/bmr-jak-obliczyc-zapotrzebowanie-kaloryc dostęp 01.2024

[18] Express.js [Online]

https://boringowl.io/tag/express-js dostęp 01.2024

[19] Express session [Online]

https://medium.com/@zakiazizi1841992/express-js-session-explained-super-easy-with-example-908d8bc4bef9 dostęp 01.2024

[20] MySQL [Online] https://vavatech.pl/technologie/bazy-danych/mysql dostęp 01.2024

[21] bCrypt [Online]

https://davidburdelak.pl/blog/bcrypt-bezpieczne-przechowywanie-hasla-w-bazie-danych dostęp 01.2024

[22] Fetch API [Online] https://devenv.pl/fetch-api/ dostęp 01.2024

[23] Body-parser [Online] https://www.geeksforgeeks.org/body-parser-middleware-in-node-js/ dostęp 01.2024

[24] Swagger [Online] https://swagger.io/docs/specification/about/ dostęp 01.2024

[25]Testy jednostkowe [Online]

https://learn.microsoft.com/pl-pl/visualstudio/javascript/unit-testing-javascript-with-visual-studio?view=vs-2022&tabs=jest dostęp 01.2024

[26] Jest [Online] jestjs.io [Online] dostęp 01.2024

[27] Supertest [Online] https://github.com/ladjs/supertest [Online] dostęp 01.2024

[28] Nagłówek [Online]

https://www.flaticon.com/free-icon/healthy-food_2460233 dostęp 01.2024

[29] Testy integracyjne [Online]

https://boringowl.io/blog/wykorzystanie-jest-w-testowaniu-javascript-zrozumienie-kluczowych-korzysci

[30] Motywacja [Online]

https://www.zwrotnikraka.pl/otylosc-nadmierna-masa-ciala-czynniki-ryzyka-rozwoju-choroby-nowotworowej/

[31] HTTP [Online]

https://widoczni.com/blog/najwiekszy-slownik-marketingu-internetowego-w-polsce/protokol-http/

Spis rysunków

Spis tabel

Spis listingów