

POLITECHNIKA WROCŁAWSKA
WYDZIAŁ ELEKTRONIKI

KIERUNEK: Automatyka i Robotyka (AiR)
SPECJALIZACJA: Technologie informacyjne w systemach auto-
matyki (ART)

**PROJEKT Z BAZ DANYCH
SYSTEM DIET**

System Diet

Diet System

AUTORZY:
Maciej Adamski
Andrzej Krawczuk
Konrad Budzyński

PROWADZACY:
dr inż. Roman Ptak

OCENA PRACY:

Spis treści

| | | |
|----------|--|-----------|
| 1 | Wstępny opis projektu | 3 |
| 1.1 | Cel projektu | 3 |
| 1.2 | Działanie aplikacji | 3 |
| 1.3 | Użytkownicy aplikacji | 3 |
| 2 | Specyfikacja | 5 |
| 2.1 | Architektura bazy danych | 5 |
| 2.2 | Funkcjonalności aplikacji | 7 |
| 2.3 | Cechy нефunkcjonalne technologiczne | 7 |
| 2.4 | Cechy нефunkcjonalne, parametryczne | 7 |
| 3 | Projekt systemu | 9 |
| 3.1 | Projekt bazy danych | 9 |
| 3.1.1 | Model logiczno-fizyczno-koncepcyjny bazy danych | 9 |
| 3.1.2 | Mechanizmy przetwarzania danych | 10 |
| 3.1.3 | Mechanizmy bezpieczeństwa na poziomie bazy danych | 12 |
| 3.2 | Projekt aplikacji użytkownika | 13 |
| 3.2.1 | Architektura aplikacji i schemat jej przypadków użycia | 13 |
| 3.2.2 | Dostęp do bazy danych | 14 |
| 3.2.3 | Algorytm wyliczający zapotrzebowanie kaloryczne | 15 |
| 3.2.4 | Metoda podłączenia do bazy danych | 16 |
| 3.2.5 | Projekt zabezpieczeń na poziomie aplikacji | 16 |
| 4 | Testy | 19 |
| 4.1 | Testy bazy danych | 19 |
| 5 | Aplikacja | 23 |
| 5.1 | Widok dla użytkowników | 23 |
| 5.1.1 | Zwykły użytkownik - obsługa aplikacji | 23 |
| 5.1.2 | Specjalista - dodatkowa strefa | 27 |
| 5.1.3 | Administrator - kolejna strefa | 27 |
| 5.2 | Zabezpieczenia, walidacja i poziomy dostępu | 28 |
| 5.2.1 | Osoba wylogowana | 28 |
| 5.2.2 | Zwykły użytkownik | 29 |
| 5.2.3 | Specjalista | 29 |
| 5.2.4 | Zabezpieczenia danych | 30 |
| 5.2.5 | Walidacja danych | 30 |
| 5.3 | Wnętrze aplikacji - kod źródłowy | 30 |
| 5.3.1 | Podstawowe mechanizmy | 30 |

| | | |
|----------|--|-----------|
| 5.3.2 | Formularze do wprowadzania i edycji danych | 31 |
| 5.3.3 | Dekoratory | 32 |
| 5.3.4 | Algorytm dobierający diety | 32 |
| 5.3.5 | Wykorzystany javascript | 33 |
| 5.4 | Wdrażanie do systemu online | 33 |
| 6 | Podsumowanie | 35 |
| 6.1 | Podsumowanie i wnioski | 35 |

Rozdział 1

Wstępny opis projektu

1.1 Cel projektu

Celem projektu jest stworzenie aplikacji pomagającej w utrzymaniu zdrowego trybu życia, poprzez proponowanie diety niezbędnej do utrzymania prawidłowej masy ciała. Kierowana jest ona dla osób aktywnych fizycznie. Aplikacja ta jest potrzebna, w związku z brakiem czasu na organizację w życiu codziennym, ponieważ pozwoli na przygotowanie diety, bez straty tak cennego w dzisiejszym świecie czasu.

1.2 Działanie aplikacji

Tworzona w projekcie aplikacja musi być przystosowana do przygotowania odpowiedniej dla korzystającej z niej osób diety. W tym celu potrzebne są parametry fizyczne, oraz aktywność osoby, na podstawie których sugerowana będzie dieta. Projekt ma ułatwić organizację zdrowego trybu życia dla korzystającej z niego osoby. Niestety aktualna wersja projektu nie przewiduje przygotowania diet dla osób chorych, lub nietolerujących niektórych składników.

1.3 Użytkownicy aplikacji

System diet jest projektem składającym się z bazy danych oraz aplikacji webowej będąca graficznym interfejsem pomagającym w korzystaniu z owej bazy. Podstawowymi klientami korzystającymi z dostępnych w projekcie usług będą więc osoby będące w stanie połączyć się do sieci, w której będzie on funkcjonował. Ponadto zespół przygotowujący projekt zakłada, że będzie to aplikacja internetowa połączona z bazą danych. Ma ona pomóc w układaniu odpowiedniego planu diety dostosowanego do użytkownika. Potrzebne jest do tego przygotowanie dostępu nie tylko dla klientów, lecz również dla **specjalistów**, mających szerszą wiedzę w zakresie dietetyki, aby byli w stanie odpowiednio edytować dane, a także dla **administratora** mającego wpływ na architekturę systemu.

Rozdział 2

Specyfikacja

2.1 Architektura bazy danych

Baza danych zawarta w projekcie musi jednocześnie zawierać użytkowników aplikacji, ich uprawnienia i dane potrzebne do logowania, a także informacje o produktach, posiłkach czy aktywnościach fizycznych niezbędne do jej funkcjonowania. W związku z tym podzielona będzie na następujące relacje:

- Relacja kont - (tysiące rekordów)
 - id_konta
 - login
 - hasło
 - uprawnienia
- Relacja użytkowników (tysiące rekordów)
 - login
 - imię
 - nazwisko
 - płeć
 - wiek
 - waga
 - wzrost
- Wiersze aktywności - (tysiące rekordów)
 - login
 - id_aktywności
 - tygodniowy czas aktywności

- Relacja aktywności fizycznych (dziesiątki rekordów)

- id.aktywności
- rodzaj aktywności
- godzinowe zużycie energii

- Relacja produktów (setki rekordów)

- id_produktyw
- nazwa_produkty
- typ_produkty
- ilość białka na 100g produktu
- ilość tłuszczu na 100g produktu
- ilość węglowodanów na 100g produktu

- Relacja posiłków (setki rekordów)

- id_posilku
- nazwa_posilku

- Wiersze posiłków (tysiące rekordów)

- id_posilku
- id_produkty
- ilość wykorzystanego produktu

- Relacja diet (tysiące rekordów)

- id_uzytkownika
- posiłek nr 1
- posiłek nr 2
- posiłek nr 3
- posiłek nr 4
- posiłek nr 5

- Relacja diet tygodniowych (tysiące rekordów)

- login
- id_diety
- dzień tygodnia

2.2 Funkcjonalności aplikacji

Do obsługi owej bazy danych niezbędna będzie aplikacja zapewniająca różne funkcjonalności dla użytkowników, w zależności od ich uprawnień. Wśród dostępnych dla wszystkich użytkowników funkcjonalności muszą znaleźć się takie jak:

- Zakładanie, usuwanie i konfiguracja konta
- Wprowadzanie i edycja danych
- Obliczanie zapotrzebowania na składniki odżywcze
- Dobieranie diety zależnie od zapotrzebowania
- Wyświetlanie bazy danych produktów i posiłków
- Dostęp do spersonalizowanej diety

Ponadto musi ona zapewnić następujące funkcjonalności dla kont z większymi uprawnieniami:

- Edytowanie zawartości baz danych związanych z produktami przez specjalistów
- Edytowanie algorytmu dobierającego diety przez specjalistów i ewentualne jego zmiany
- Edytowanie relacji kont przez administratora informatycznego

2.3 Cechy нефunkcjonalne technologiczne

Aplikacja spełniać będzie następujące wymagania нефunkcjonalne, technologiczne:

- Tryb pracy: graficzny
- Dostęp do pracy: siecowy
- System bazodawnowy: MySQL
- Platforma systemowa: Windows10
- Język programowania: Python3
- Środowisko programistyczne: Pycharm Community Edition
- System kontroli wersji: Git

2.4 Cechy нефunkcjonalne, parametryczne

Należy również przeanalizować działanie aplikacji, biorąc pod uwagę bezpieczeństwo jej działania. Jako, że jest to system diet, gdzie każdy użytkownik może liczyć na ułożenie diety nie wymaga ona wielu funkcji zabezpieczających. Wystarczy więc system logowania oparty na loginie oraz hasle. Kolejnym zagrożeniem mogłoby być przeciążenie serwera i systemu. Zakładamy jednak, że na początku w bazie danych będą tabele, z zawartością po około tysiąc rekordów, które zwiększać się będą liniowo, aż do pułapu dziesięciu tysięcy. Takie liczby nie powinny być problemem wzięwszy pod uwagę dostępną w teraźniejszości technologię.

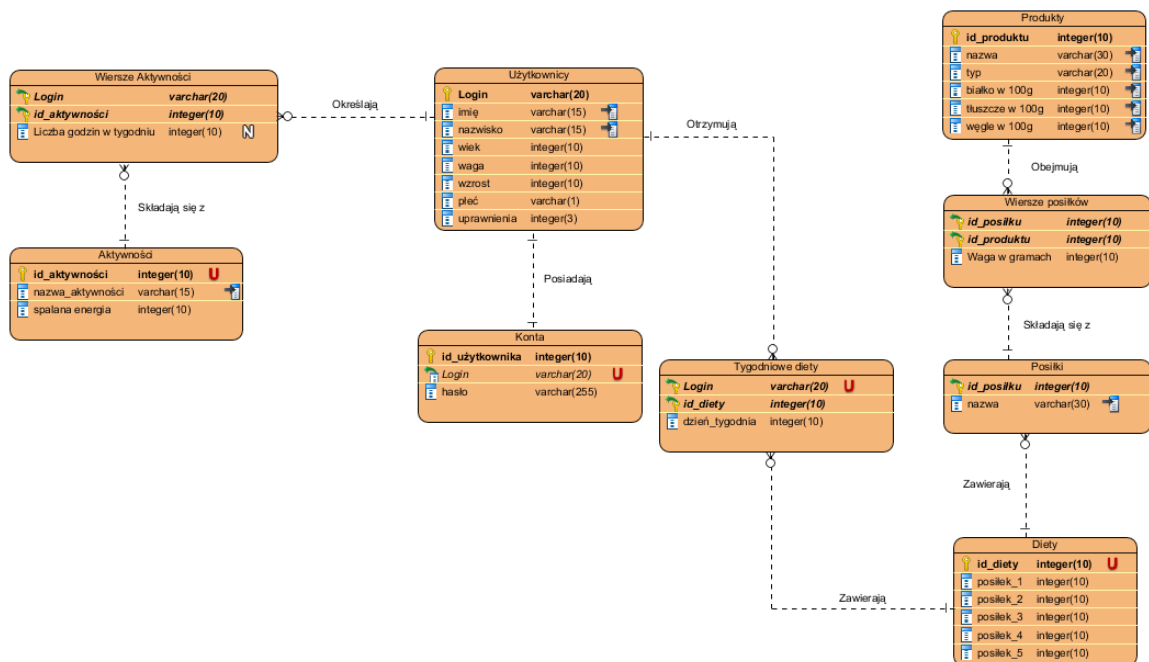
Rozdział 3

Projekt systemu

3.1 Projekt bazy danych

3.1.1 Model logiczno-fizyczno-koncepcyjny bazy danych

Diagram związków encji (ang. ERD - Entity Relationship Diagram) pozwala na modelowanie struktur danych oraz opisuje relacje pomiędzy nimi. Dla bazy danych zawartej w projekcie "System diet" diagram prezentuje się następująco:

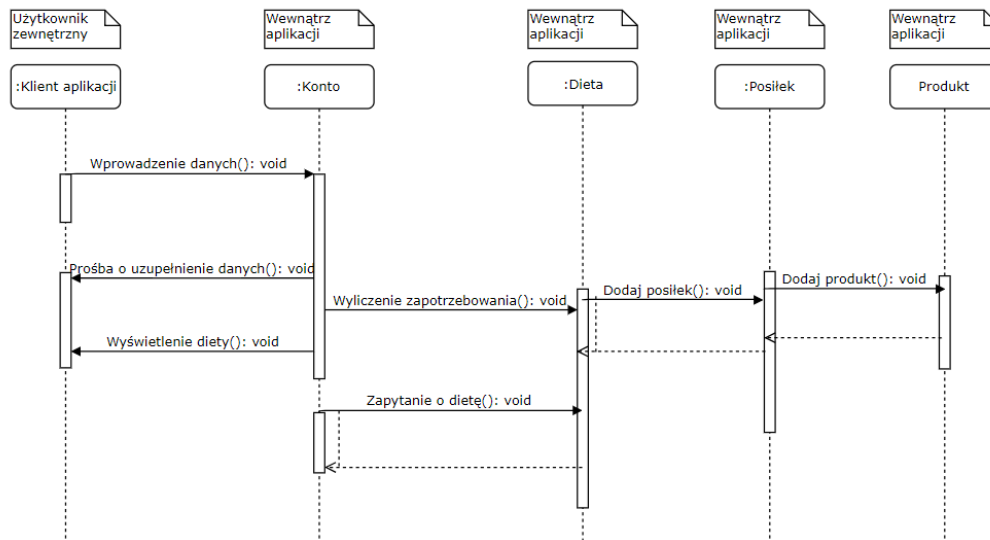


Rysunek 3.1 Model logiczno-fizyczno-konceptualny

Wszystkie modele zawarte są na jednym diagramie, na co pozwalało znormalizowanie schematu bazy danych do trzeciej postaci normalnej.

3.1.2 Mechanizmy przetwarzania danych

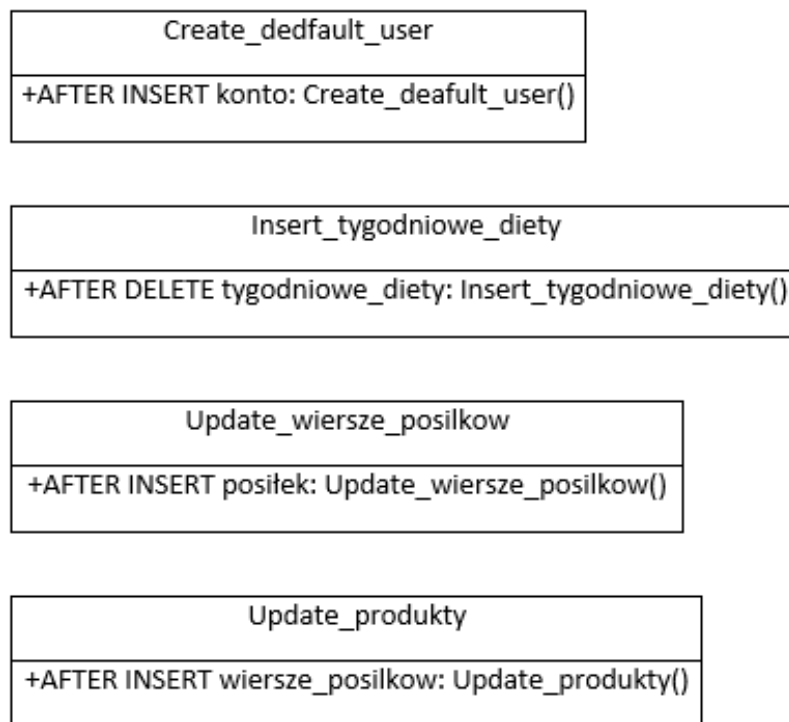
Na potrzeby pokazania ścieżki realizowanej przez aplikację stworzono diagram sekwencyjny, pokazujący poszczególne etapy komunikacji między danymi relacjami, przy wykorzystaniu aplikacji. Diagram ten przedstawia w jaki sposób zwykły użytkownik może wykorzystać aplikację, nie przedstawiając wszystkich jej funkcjonalności. Pozwala to jednak ukazać hierarchię wywołań i poleceń, zakładaną w projekcie, według której jest on realizowany.



Rysunek 3.2 Diagram sekwencyjny

Należy również rozważyć jakie widoki zawarte będą w aplikacji. Już teraz można zdefiniować kilka z nich:

- Widok logowania
 - konto: login
 - konto: hasło
- Widok diety
 - użytkownicy: imię
 - użytkownicy: nazwisko
 - tygodniowe diety: dzień_tygodnia
 - posiłki: nazwa
- Widok parametrów fizycznych i aktywności
 - użytkownicy: wiek
 - użytkownicy: waga
 - użytkownicy: wzrost
 - użytkownicy: aktywności



Rysunek 3.3 Przykładowe wyzwalacze

Przykładowy opis wyzwalacza `CreateDefaultUser()`: Do każdego konta musi być przypisany użytkownik, więc po jego stworzeniu zostaną wprowadzone domyślne dane, określające pola, które nie mogą być zerowe dla użytkownika powiązanego z danym kontem w momencie jego stworzenia.

Początkowo utworzone dane będą generowane automatycznie przy użyciu wbudowanej **sekwencji**, a wszystkie indeksy, zawierane w kluczach głównych będą **inkrementowane automatycznie**. Jedynym kluczem, nie związanym z indeksem jest login, który musi być unikalny, a także niezbędny jest do stworzenia rekordu, będzie to więc jedyny klucz główny, do którego nie będzie wymagana sekwencja.

Warto również pomyśleć o sposobach, które ułatwią wyszukiwanie danych. W tym celu przygotowane zostaną następujące indeksy, niezależnie od kluczy głównych, będących w każdym przypadku będą również indeksami, które wykorzystane będą do wyszukiwania danych:

- Użytkownicy: imię+nazwisko
- Produkty: nazwa+typ
- Produkty: białko+węgle+tłuszcze
- Aktywności: nazwa aktywności
- Posiłki: nazwa

3.1.3 Mechanizmy bezpieczeństwa na poziomie bazy danych

Ze względu na małą wagę niebezpieczeństwa związanego z dostaniem się do danych aplikacji, z której korzystanie będzie darmowe nie podjęto przesadnych środków bezpieczeństwa. Należy jednak zaznaczyć, że w związku z **ogólnym rozporządzeniem o ochronie danych osobowych** zastosowanie pewnych środków jest niezbędne. W związku z tym hasło, które pozwala na logowanie, a tym samym podejrzenie danych osobowych będzie jedynym zaszyfrowanym polem w bazie danych, do czego wykorzystany zostanie funkcja kryptograficzna **SHA256**.

Należy również zauważyć, że bezpieczeństwo wiąże się także z integralnością danych. Ze względu na brak dostępu użytkownika bezpośrednio do bazy danych, a jedynie poprzez użycie wyzwalaczy, niezbędne ograniczenia (takie jak wymaganie wpisania liczby dla wieku) zaimplementowane zostaną w aplikacji. W bazie danych zdefiniowane zostanie jedynie, w których polach dopuszczalna jest wartość zerowa.

Podział nadanych poszczególnych uprawnień dla użytkownika, specjalisty oraz admina został przedstawiony w poniższej tabelce.

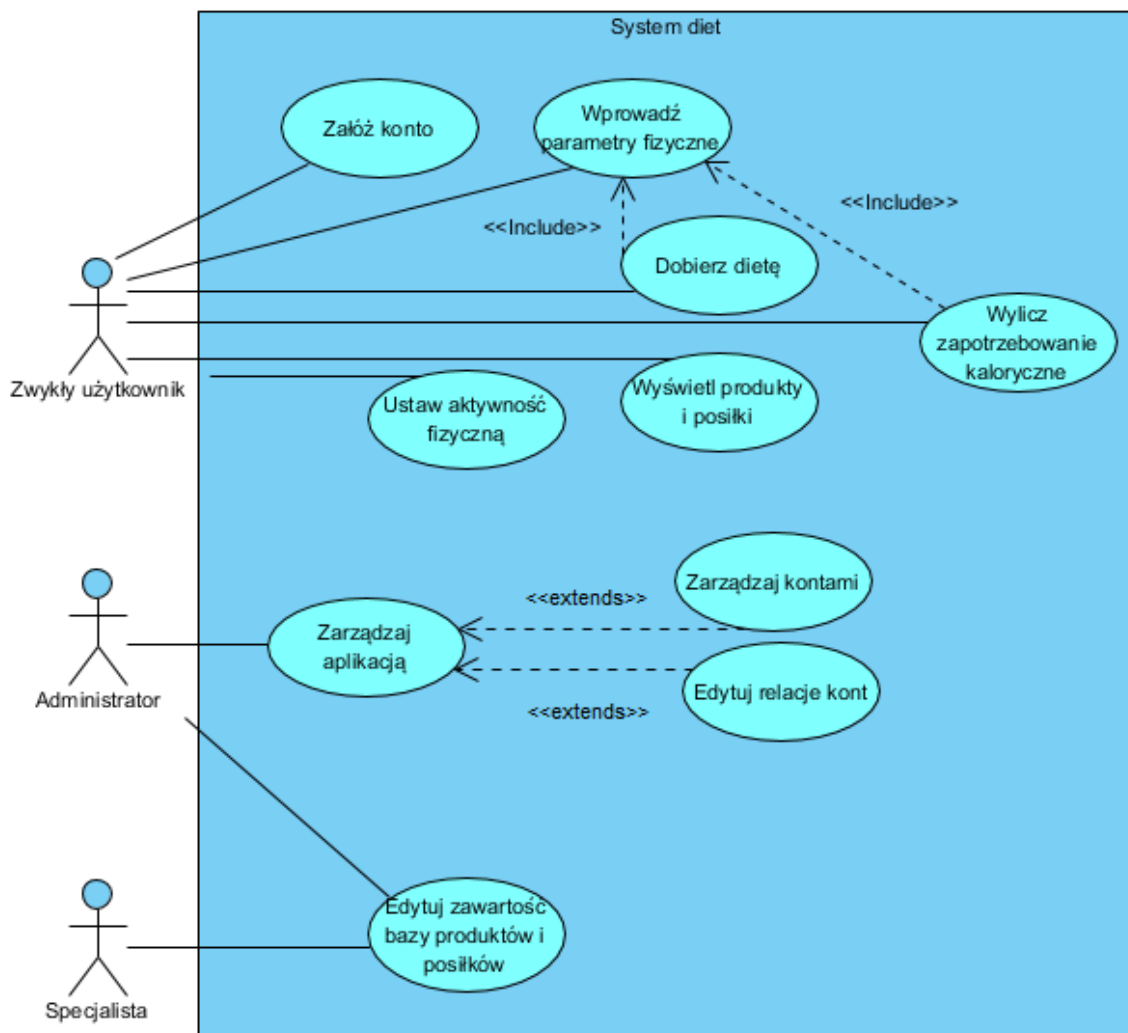
| Dane | Użytkownik | Specjalista | Admin | |
|--------------------------|------------|-------------|-------|------------|
| Konto | C | C | CRUD | C - Create |
| Dane osobowe użytkownika | RU | RU | CRUD | R - Read |
| Aktywności fizyczne | RU | RU | CRUD | U - Update |
| Produkty | R | CRUD | CRUD | D - Delete |
| Posiłki | R | CRUD | CRUD | |
| Diety | R | CRUD | CRUD | |
| Login | C | C | CRUD | |
| Hasło | C | C | CD | |

Rysunek 3.4 Tabela uprawnień

3.2 Projekt aplikacji użytkownika

3.2.1 Architektura aplikacji i schemat jej przypadków użycia

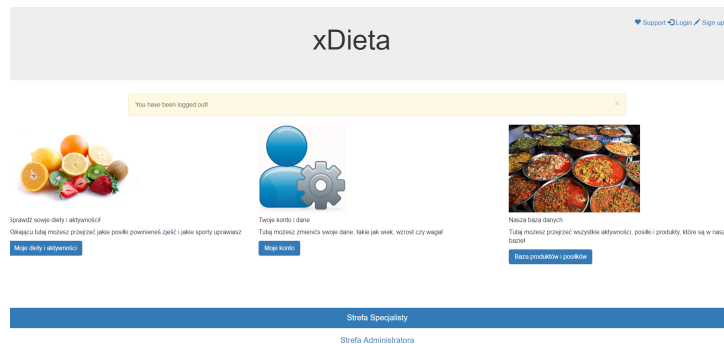
Wykorzystana architektura zakłada "aplikację w jednym". Cała aplikacja stworzona będzie bowiem w jednym projekcie, składającym się z głównego pliku init, oraz innych - widoków, stylów lub agentów rozszerzających jej funkcjonalności. Sama aplikacja jest jednak na tyle mała, że ilość plików jest odpowiednia dla pojedynczego projektu i nie ma potrzeby rozbijania jej na więcej części. Ułatwi to wymianę danych, przekazywanie parametrów a także zarządzanie funkcjonalnościami w niej zawartymi, nawet dotyczącymi różnych jej fragmentów. Możliwości używania aplikacji i jej funkcjonalności przedstawiono poniżej



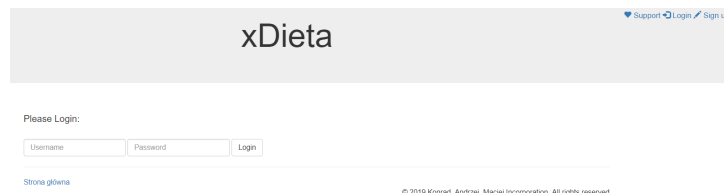
Rysunek 3.5 Diagram przypadków użycia

3.2.2 Dostęp do bazy danych

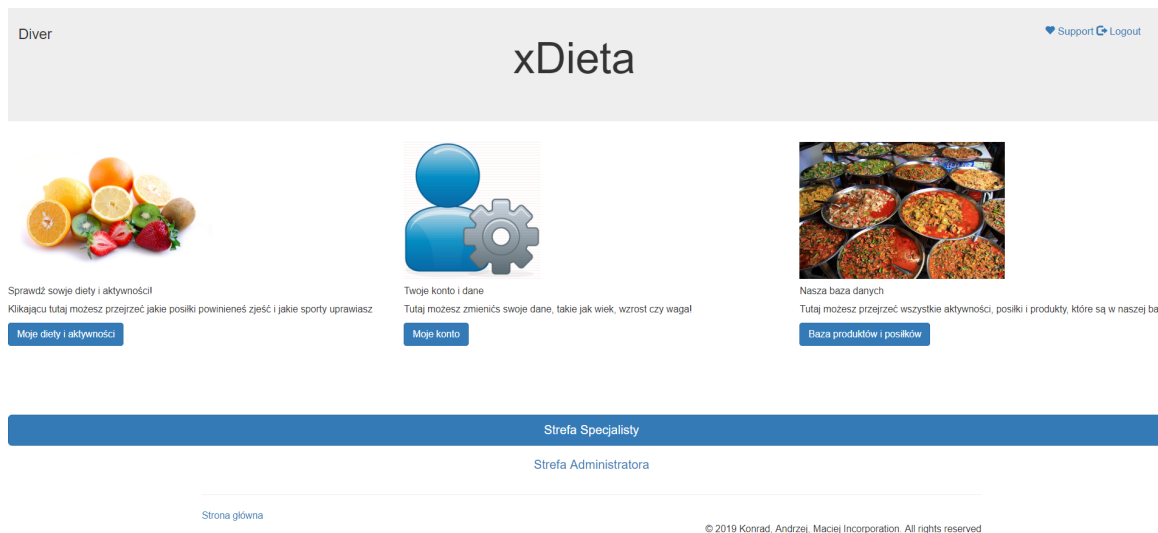
Wstępnie stworzono projekt aplikacji, która pozwoli operować na zakładanej bazie danych. W tym celu stworzono graficzny projekt, który pozwoli w późniejszym etapie utrzymać estetykę wytwarzanego oprogramowania, jednocześnie pozwalając na drobne zmiany w razie problemów z implementacją. Początkowo zakładany interfejs aplikacji dostępu do bazy danych przedstawiono na załączonych poniżej obrazach.



Rysunek 3.6 Panel po uruchomieniu aplikacji bez logowania



Rysunek 3.7 Panel logowania



Rysunek 3.8 Panel po uruchomieniu aplikacji na zalogowanym koncie

| id_uzytkownika | Login | hasło | uprawnienia |
|----------------|-------|---|-------------|
| 37 | Diver | \$5\$rounds=535000\$8LEdwwKPS919/1mt\$XufE5n48LRQRq\$5xB0Mg9mEQqKy2F0hcVwYYWEMA | 2 |
| 38 | Mefoo | \$5\$rounds=535000\$wiZgrfVbSpyeAy\$0xFiGd2xU4L04KT3K9ZNUjwwbWln6X9MOCzRF23uG | 2 |
| 39 | Andre | \$5\$rounds=535000\$K6oLke/9raecsy\$IM5QqCNmB4uVZFQ2gEprQKurIqzFipabBNH0d0B | 1 |

Rysunek 3.9 Strefa Administratora

3.2.3 Algorytm wyliczający zapotrzebowanie kaloryczne

Algorytm wyliczający zapotrzebowanie kaloryczne działa w oparciu o indywidualne parametry fizyczne każdego z użytkowników. W pierwszej fazie liczy całkowitą przemianę materii, następnie zwiększa ją, utrzymuje lub zmniejsza o stałe ustaloną wartość (w zależności od indywidualnego celu żywieniowego), by ostatecznie podzielić dzienne zapotrzebowanie na odpowiednie makroskładniki. Całkowita przemiana materii liczona będzie w dwóch etapach. Pierwszym z nich jest obliczenie BMR (podstawowe zapotrzebowanie kaloryczne). BMR liczony będzie wg następujących wzorów:

dla mężczyzn:

$$[9,99 \times masa_ciala(kg)] + [6,25 \times wzrost(cm)] - [4,92 \times wiek(lata)] + 5$$

dla kobiet:

$$[[9,99 \times masa_ciala(kg)] + [6,25 \times wzrost(cm)] - [4,92 \times wiek(lata)] - 161$$

Następnie do wskaźnika BMR dodane zostanie zapotrzebowanie kaloryczne w zależności od aktywności fizycznych użytkowników, co pozwoli otrzymać pełną informację o tymże zapotrzebowaniu dziennym. W ostatnim etapie algorytm przelicza zapotrzebowanie kaloryczne na makroskładniki tak, by można było na ich podstawie dobrać odpowiednią dietę. W omawianym projekcie algorytm dobierał będzie wartości w następujący sposób:

- 25% zapotrzebowania stanowią tłuszcze (1 gram tłuszczu to około 9kcal)
- 1.8 gram białka na każdy kilogram masy ciała (1 gram białka to około 4kcal)
- pozostałe zapotrzebowanie stanowią węglowodany (1 gram węglowodanów to około 4kcal)

W celu lepszego zrozumienia działania ostatniej części algorytmu zaprezentowany zostanie przykład obliczenia zapotrzebowania na makroskładniki:

Zakładając, że obliczone zapotrzebowanie po uwzględnieniu aktywności fizycznej wynosi 1800kcal na dzień, a osoba waży 75 kilogramów i planuje utrzymać wagę:

- tłuszcze: $25\% \times 1800kcal = 450kcal$; $\frac{450kcal}{9\frac{kcal}{gram}} = 50gram$
- białko: $75kg \times 1.8\frac{gram}{kg} = 135gram$
- węglowodany: $135gram \times 4\frac{kcal}{gram} = 540kcal$; $1800kcal - 540kcal - 450kcal = 810kcal$;
 $\frac{810kcal}{4\frac{kcal}{gram}} \approx 200gram$

Podsumowując powyższy przykład, makroskładniki dla osoby ważącej 75kg i zapotrzebowaniu kalorycznym na poziomie 1800kcal na dzień wynoszą:

- tłuszcze: 50 gram/dzień
- białko: 135 gram/dzień
- węglowodany: 200 gram/dzień

3.2.4 Metoda podłączenia do bazy danych

Projekt aplikacji tworzony jest w języku Python3, w związku z czym do podłączenia się do bazy danych stworzonej przy użyciu MySQL wykorzystana zostanie paczka **MySQLConnector**. Pozwala ona na połączenie z bazą danych, dzięki wykorzystaniu protokołu **OBD**C (Open-Database Connectivity). Wysoki poziom języka Python3, a także użycie tej biblioteki w dużym stopniu zapewni niezbędną integralność danych. Zaprogramowane zostaną jednak również warunki w funkcjach wiążących się z wyzwalaczami, które nie sprawdzą jedynie zgodności typów, lecz również zawartość edytowanych rekordów.

3.2.5 Projekt zabezpieczeń na poziomie aplikacji

Zabezpieczenia w aplikacji będzie można podzielić na dwa obszary. Będą to zabezpieczenia po stronie klienta, jeszcze na stronie - sprawdzane podczas wprowadzania danych. Sprawdzać one będą zgodność danych z wymogami, a także obecność niebezpiecznych skryptów, mogących zaszkodzić aplikacji. Drugim poziomem zabezpieczeń będą zabezpieczenia po stronie serwera, w razie, gdyby przy użyciu JQuery, lub innych środków

użytkownik obszedł zabezpieczenia po stronie klienta. Przeprowadzana będzie walidacja danych, dla których zastosowane będą odpowiednie filtry. Przykładowy filtr: imię musi składać się z od 3 do 20 liter, bez znaków specjalnych i liczb.

Oprócz przy wprowadzaniu danych, pojawią się również zabezpieczenia przy wyświetlaniu widoków. Wykorzystane zostaną do tego gotowe funkcje, sprawdzające, czy spełnione są niezbędne wymagania (słowa kluczowe Required), dotyczące logowania danego użytkownika, a dane wykorzystywane do tworzenia widoku pobrane zostaną z bazy danych w zależności od jego id, więc nawet w źródle strony nie będzie dostępu do danych innych użytkowników.

W celu ochrony danych klienta również przed administratorem wykorzystane zostaną gotowe funkcje szyfrujące, nie pozwalające na poznanie hasła użytkownika, nawet przy dostępie do relacji kont.

Rozdział 4

Testy

4.1 Testy bazy danych

Po zaprojektowaniu bazy danych przeszliśmy do jej tworzenia. Wszystkie tablice, klucze główne, klucze obce, indeksy, widoki zostały stworzone w środowisku MAMP na serwerze lokalnym.

Wszystkie operacje edycji bazy danych zostały napisane w języku Python za pomocą biblioteki `mysql.connector`, która pozwala na połączenie z bazą danych i jej edycję za pomocą MySQL.

Na bazie danych z przykładowymi danymi zostały wykonane testy:

- Test czasu wyszukiwania w małej tabeli 70 rekordów

```
start = timeit.timeit()
find_meal_name('142')
end = timeit.timeit()
print(end - start)
```

Rysunek 4.1 Polecenie wyszukiwania

```
3.43999999999989994e-05
```

Rysunek 4.2 Czas wyszukiwania w sekundach

- Test indeksu UNIQUE

```
insert account(("mefioo",None,1))
```

Rysunek 4.3 Próba dodania konta o nazwie już zajętej

```
1062 (23000): Duplicate entry 'mefioo' for key 'LoginKonto'
```

Rysunek 4.4 Komunikat błędu

- Test zmiany stanu bazy po komendzie INSERT

| id_uzytkownika | Login | haslo | uprawnienia |
|----------------|--------------|-----------------|-------------|
| 1 | xd | beka | 0 |
| 2 | awd | awdawdawda | 0 |
| 3 | dwadawdawdaw | awdawdawdawdawd | 1 |
| 31 | mefioo | xd | 2 |
| 33 | Andre | xddd | 1 |
| 34 | UserLogin5 | None | 1 |
| 35 | UserLogin4 | UserPassword | 2 |
| 36 | Mama | None | 1 |
| 41 | NULL | None | 1 |

Rysunek 4.5 Tabela przed INSERT

```
insert_account(("Test","password",1))
```

Rysunek 4.6 Polecenie INSERT

| id_uzytkownika | Login | haslo | uprawnienia |
|----------------|--------------|-----------------|-------------|
| 1 | xd | beka | 0 |
| 2 | awd | awdawdawda | 0 |
| 3 | dwadawdawdaw | awdawdawdawdawd | 1 |
| 31 | mefioo | xd | 2 |
| 33 | Andre | xddd | 1 |
| 34 | UserLogin5 | None | 1 |
| 35 | UserLogin4 | UserPassword | 2 |
| 36 | Mama | None | 1 |
| 41 | NULL | None | 1 |
| 42 | Test | password | 1 |

Rysunek 4.7 Tabela po poleceniu INSERT

- Testy poprawności argumentów komend

```
insert_activity(('grucht','string'))
1366 (HY000): Incorrect integer value: 'string' for column 'spalana_energia' at row 1
```

Rysunek 4.8 Polecenie i komunikat dla niepoprawnych danych(zmienna string zamiast int)

```
insert_account(("loginloginloginloginloginlogin", 'passwd', 1))
1406 (22001): Data too long for column 'Login' at row 1
```

Rysunek 4.9 Polecenie i komunikat dla niepoprawnych danych(za dużo znaków)

```
insert_diet((None, 1, 1, 1, 1))
1048 (23000): Column 'posilek_1' cannot be null
```

Rysunek 4.10 Polecenia i komunikat, nazwa nie może być wartości NULL

```
start = timeit.timeit()
find_meal_name('593')
end = timeit.timeit()
print(end - start)
```

Rysunek 4.11 Polecenie wyszukiwania

```
0.000170200000000000923
```

Rysunek 4.12 Czas wyszukiwania

- Test czasu wyszukiwania dla tabeli 587 rekordów
- Test usuwania rekordu

| + Options | | | | | id_uzytkownika | Login | haslo | uprawnienia |
|--------------------------|------|------|--------|--|----------------|-----------------|-----------|-------------|
| <input type="checkbox"/> | Edit | Copy | Delete | | 4 | IronMan | ironman | 0 |
| <input type="checkbox"/> | Edit | Copy | Delete | | 5 | DrStrange | drstrange | 0 |
| <input type="checkbox"/> | Edit | Copy | Delete | | 6 | Thanos | thanos | 0 |
| <input type="checkbox"/> | Edit | Copy | Delete | | 7 | SpiderMan | spiderman | 0 |
| <input type="checkbox"/> | Edit | Copy | Delete | | 8 | CzłowiekKapusta | hehe | 0 |

Rysunek 4.13 Tabela przed poleceniem DELETE

```
delete_account(4)
```

Rysunek 4.14 Polecenie DELETE

| | | | | | id_uzytkownika | Login | haslo | uprawnienia |
|--------------------------|------|------|--------|--|----------------|-----------------|-----------|-------------|
| <input type="checkbox"/> | Edit | Copy | Delete | | 5 | DrStrange | drstrange | 0 |
| <input type="checkbox"/> | Edit | Copy | Delete | | 6 | Thanos | thanos | 0 |
| <input type="checkbox"/> | Edit | Copy | Delete | | 7 | SpiderMan | spiderman | 0 |
| <input type="checkbox"/> | Edit | Copy | Delete | | 8 | CzłowiekKapusta | hehe | 0 |

Rysunek 4.15 Tabela po poleceniu DELETE

Testy uprawnień zostaną przeprowadzone dopiero z poziomu interfejsu, ze względu na to, że utrata danych nie wiąże się z poważnymi konsekwencjami a tym samym, tak jak wcześniej już zdefiniowano - kwestie bezpieczeństwa rozwiązane zostaną w aplikacji. Należy więc powiedzieć, że sama aplikacja wiąże się z bazą danych z poziomu korzenia i może ją dowolnie edytować a wszystkie restrykcje dotyczące uprawnień wprowadzone zostaną dopiero w niej. Rozwiązanie to nie jest najbezpieczniejszym, jednak ze względu na małe zagrożenie i konsekwencje utraty danych taki poziom zabezpieczeń uznany został za wystarczający.

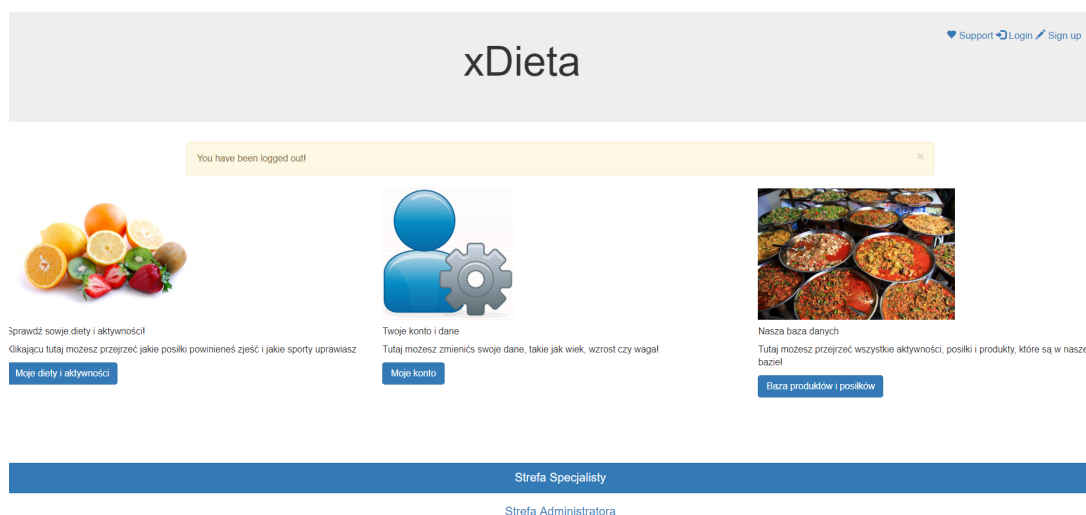
Rozdział 5

Aplikacja

5.1 Widok dla użytkowników

5.1.1 Zwyczajny użytkownik - obsługa aplikacji

Aby zacząć pracę z aplikacją, niezbędne jest jej otwarcie i dostanie się do podstawowego panelu, z którego można się zarejestrować. Panel ten wygląda w sposób następujący:



Rysunek 5.1 Podstawowy panel aplikacji

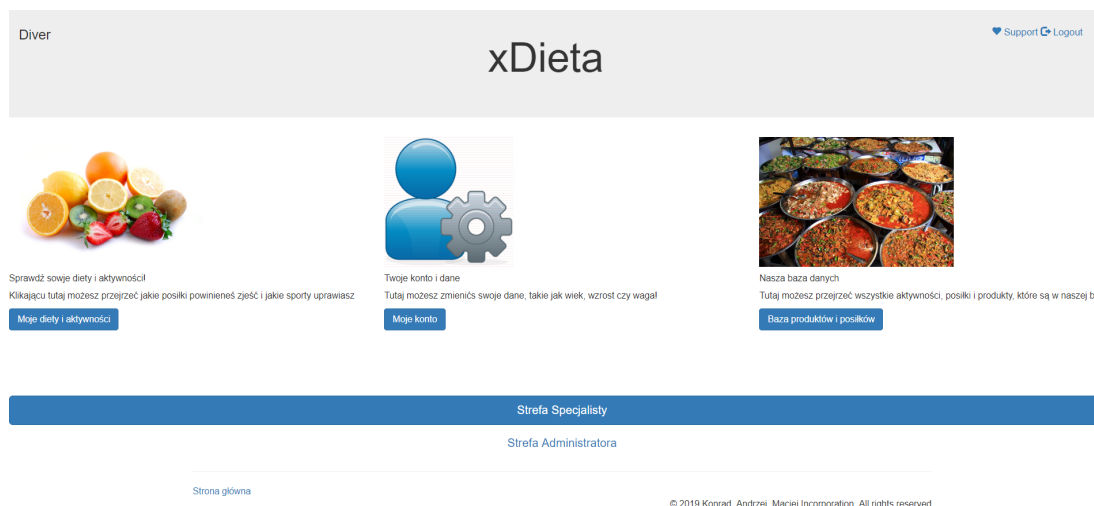
Aby się zarejestrować należy kliknąć na przycisk *Sign up* znajdujący się w panelu nawigacyjnym w prawym górnym rogu. Jego naciśnięcie przekieruje użytkownika do panelu rejestracyjnego, w którym wymagane jest uzupełnienie trzech pól - loginu, hasła oraz powtórzenia hasła, co będzie niezbędne do stworzenia własnego konta. Należy również wyrazić zgodę na politykę prywatności strony, poprzez zaznaczenie dostępnego w tym panelu pola wyboru. Jest to niezbędne do założenia konta. Po założeniu konta użytkownik zostaje przekierowany z powrotem do menu głównego, lecz może zobaczyć zmianę w panelu nawigacyjnym. Teraz zamiast pól *Sign up* oraz *Log in* znajduje się tam pole *Log out*. Po rejestracji bowiem następuje automatyczne zalogowanie. Można również zwrócić uwagę na wyświetlony login użytkownika w lewym górnym rogu panelu nawigacyjnego. Aby zacząć korzystać z aplikacji użytkownik powinien wypełnić swoje dane, co może zrobić w dostępnej na środku podstawowego panelu karcie *Moje konto*. Wchodząc we właściwości swojego konta, użytkownik ma szansę na wprowadzenie danych takich jak imię, nazwisko,

Register

Akceptuję nowe warunki polityki prywatności (Zaktualizowane 26.05.2018)

☒

Rysunek 5.2 Okno rejestracji



Rysunek 5.3 Podstawowy panel aplikacji po zalogowaniu

wiek, waga, wzrost czy płeć, które, poza pierwszymi dwoma, niezbędne są do ustalania odpowiedniej diety, co jest celem działania aplikacji. W tym miejscu użytkownik może również zmienić swoje hasło na nowe, przy pomocy którego będzie się logował. W celu zmiany hasła nie jest wymagane podanie starego hasła, w związku z potrzebą jego znania do zalogowania się. Wszystkie wypisane powyżej operacje zatwierdza się przyciskiem *Submit*, który potwierdza chęć wykonania operacji.

W każdym momencie można to jednak przerwać wykonywaną operację, wracając do panelu głównego, co możliwe jest poprzez naciśnięcie w pasie dolnym strony linku *Strona główna*. Wciśnięcie tego przycisku nigdy nie powoduje wprowadzania zmian do bazy danych, jest to przerwanie wszystkich podjętych czynności. Po wprowadzeniu danych osobowych użytkownik może przejrzeć dostępne w bazie produkty, posiłki i aktywności, klikając na kartę po prawej stronie panelu podstawowego aplikacji o nazwie *Baza produktów i posiłków*. Dane te udostępnione są w formie przejrzystych tabel, które można sortować po dowolnej kolumnie, wspierają funkcje wyszukiwania, a także wyboru ilości pokazywanych wierszy na jednej stronie tabeli. Aby przełączać między poszczególnymi tabelami wystarczy klikać na dostępne nad aktualnie wyświetlaną tabelą przyciski: *Aktywności*, *Posiłki*, *Produkty*. Każdy przycisk odpowiada za wyświetlenie pasującej jego nazwie tabeli. Aby powrócić do panelu głównego znów można kliknąć przyciskiem myszy na znajdujący się na dole strony link *Strona główna*. Ostatnią dostępną dla użytkownika kartą jest karta *Moje diety i aktywności*. Po naciśnięciu na przycisk przypięty do tej karty użytkownik

Wprowadź lub uaktualnij swoje dan

Konrad

Budzyński

22

72

179

Password

Zaznacz to pole, jeśli jesteś mężczyzną

☐

Submit

Rysunek 5.4 Okno zmiany danych użytkownika

Tabele

Aktywności

Posiłki

Produkty

| id_produktu | nazwa | białko_w_100g | tluszcz_w_100g | wegle_w_100g |
|-------------|---------------------------|---------------|----------------|--------------|
| 6410 | Agar (suszony) | 7 | 0 | 73 |
| 6291 | Algi nori (arkusze) | 30 | 2 | 0 |
| 109 | Amarantus, ziarno | 16 | 7 | 59 |
| 127 | Ananas | 0 | 0 | 12 |
| 4403 | Ananas, plastry w syropie | 0 | 0 | 20 |
| 9522 | Anyż (mielony) | 20 | 15 | 35 |
| 189 | Arbuz | 1 | 0 | 8 |
| 56 | Awokado | 2 | 15 | 4 |
| 93 | Bakłażan | 1 | 0 | 4 |
| 89 | Banan | 1 | 0 | 22 |

Showing 1 to 10 of 428 rows 10 rows per page

1 2 3 4 5 ... 43

Rysunek 5.5 Baza danych dostępnych produktów

przekierowany jest do nowej strony, w której może przejrzeć zasugerowane dla niego diety, a także swoje aktywności fizyczne i w razie czego ma dostęp do ich edycji. Należy zauważyć, że ponad tabelą, w której znajdują się diety i aktywności użytkownika ma on dostęp do czterech przycisków odpowiadających odpowiednim działaniom. Przyciski *Dodaj dietę* oraz *Usuń dietę* włączają algorytm, który wylicza, a następnie ustala dietę na cały tydzień, lub ją usuwa. Dzięki temu użytkownik może łatwo i szybko zmieniać diety. Nieco inaczej działają przyciski *Dodaj aktywność* oraz *Usuń aktywność*. Przyciski te przekierowują użytkownika do kolejnej strony, gdzie wybierając z listy może dodać nową aktywność do swojego planu, podając ilość godzin jej uprawiania tygodniowo w przypadku dodania, lub wybrać samą nazwę ćwiczenia w celu jej usunięcia ze swojego planu. Wykorzystanie w tym miejscu rozwijanej listy rozwiązuje wszelkie problemy z poprawnym nazewnictwem danych z bazy danych. Użytkownik zawsze może się wylogować i zalogować. Po zalogowaniu pojawia się mała wiadomość na środku strony o poprawnym zalogowaniu, a także login

Dodaj aktywność
Usuń aktywność
Dodaj dietę
Usuń dietę

Tabele

[Twoje Aktywności](#)
[Twoja dieta](#)

| Imię | Nazwisko | dzien_tygodnia | Sniadanie | Drugie Sniadanie | Obiad | Podwieczorek | Kolacja |
|--------|-----------|----------------|--------------------------------------|--|---|---|--|
| Konrad | Budzyński | 1 | Farsz z kurczakiem do tortilli | Gulasz wołowy z kaszą jęczmienną ze śmietaną | Zapiekane bataty, buraki i ziemniaki z dipem czosnkowym | Bezglutenowy zakwas na barszcz biały | Paleo chia pudding z granatem |
| Konrad | Budzyński | 2 | Paleo chleb | Frittata z cukinią i serem feta | Bezglutenowy chleb jaglano - gryczany | Wytrawne muffiny jaglane | Kogel mogel |
| Konrad | Budzyński | 3 | Zapiekanka ziemniaczano-warzywna | Zupa hiszpańska z ciecierzycą | Makaron z serem i brokulem | Jaglanka z bananem | Tarta z porami |
| Konrad | Budzyński | 4 | Goląbki z mięsem z kurczaka | Zapiekane plastry batatów | "Ryz" z kalafiora z mięsem mielonym | Pierś z kurczaka w plastrach selera z surówką z kalafiora i kaszą | Pieczony dorsz z komosą i pieczonym burakiem |
| Konrad | Budzyński | 5 | Chrupiąca sałatka z selera naciowego | Kefir z malinami | Naleśniki z mąki gryczanej z twarogiem i prażonymi jabłkami (składniki na 3 porcje) | "Makaron" z cukinii z wołowiną | Łosoś w papilotach z kaszą gryczaną |
| Konrad | Budzyński | 6 | Kasza jaglana z bananem i | Warzywne słupki | Zupa chrzanowa z topinamburem | Owsianka na ciepło ze śliwkami | Pancakes z czekoladą |

Rysunek 5.6 Tabele diet i aktywności

Dodaj aktywność, możesz dodać tylko aktywności o takich nazwach jakie są w bazie

Trucht ▼

Liczba godzin

Submit

Rysunek 5.7 Dodawanie aktywności do swojego planu

w lewym górnym rogu panelu nawigacyjnego. W panelu podstawowym na dole znajdują się jeszcze jednak dwa niesprawdzone przyciski, jednak w związku z umieszczonymi na nich napisami *Strefa Specjalisty* oraz *Strefa Administratora* można się spodziewać, że zwykły użytkownik nie będzie posiadał do nich dostępu.

5.1.2 Specjalista - dodatkowa strefa

Kolejnym zdefiniowanym użytkownikiem aplikacji jest specjalista - osoba odpowiedzialna za bazę danych dotyczącą produktów. W związku z tym ma specjalista możliwość ingerencji w relację produktów, w swojej strefie, co widać na *Rysunku 5.8*

Insert Product

Rysunek 5.8 Strefa specjalisty - dodanie produktu

5.1.3 Administrator - kolejna strefa

Kolejnym poziomem użytkownika aplikacji jest administrator, który odpowiedzialny jest za usuwanie kont, zmienianie haseł czy uprawnień a także ingerencje w tabelę dostępnych aktywności. Wszystkie te możliwości znajdują się w strefie administratora, która jest dostępna tylko dla użytkowników o odpowiednich uprawnieniach. Na *5.9* widać, że

Dodaj aktywność

Zmień uprawnienia

Usuń konto/zmień hasło

Użytkownicy

| id_uzytkownika | Login | haslo | uprawnienia |
|----------------|--------|---|-------------|
| 37 | Diver | \$5\$rounds=535000\$8LEDkwKPS919/TmfSjXuFe5n46LRQqSjxB0Mg9mEQqXyZF0hcVwYWIEMA | 2 |
| 38 | Mefloo | \$5\$rounds=535000\$wiZgrdVoSpyeAy\$OxFIGdc2xUs4Lo4KT3K9ZNUwwbWm9X9MOCzRFZ3uG | 2 |
| 39 | Andre | \$5\$rounds=535000\$SiEoLke/9raecsjy\$IM5QqCNm9J4uVZF00gEpQKurlquZFipabBNH0d019 | 1 |
| 41 | Kasia | \$5\$rounds=535000\$gRyCuvBpi1M2#5Y\$F03RPHIL7n3gGrkx504en.4F9r2L_yqqrV5j21hyR8 | 0 |

Rysunek 5.9 Strefa administratora

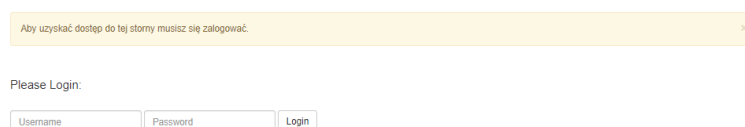
strefa ta jest podzielona na cztery sekcje. Pierwsze trzy dotyczą zmian wprowadzanych w

bazie danych. Administrator może dodać aktywność podając jej nazwę a także spalaną na godzinę energię, zmienić uprawnienia konta, poprzez podanie loginu a także nowego poziomu uprawnień. Ostatnim z tych trzech bloków jest blok służący do resetowania hasła lub usuwania konta. W momencie, kiedy pole hasło zostaje puste konto jest usuwane, jednak gdy wprowadza się tam domyślne dane "password" wtedy hasło konta zostaje zrestartowane, a użytkownik może je zmienić po zalogowaniu. W ostatniej części strefy administratora znajduje się tabela kont wraz ze wszystkimi danymi. W związku z *Ogólnym Rozporządzeniem o Ochronie Danych* administrator nie ma dostępu do prywatnych danych użytkowników aplikacji, takich jak imię czy nazwisko, a widoczne przez niego hasła w tabeli są zaszyfrowane. Jest to zgodne z wcześniejszymi założeniami projektowymi, wypisanymi w poprzednich rozdziałach.

5.2 Zabezpieczenia, walidacja i poziomy dostępu

5.2.1 Osoba wylogowana

Najniższy poziom dostępu zaoferowany jest dla osoby nieposiadającej konta w bazie danych. Osoba taka może wyświetlać jedynie panel podstawowy aplikacji oraz rejestrację. Podczas próby włączenia dowolnej innej funkcji aplikacji niezalogowany użytkownik dostaje informację, o braku dostępu do danej strony i zostaje przekierowany do strony logowania, wraz z prośbą o zalogowanie, tak jak na *Rysunku 5.10* Komunikat ten wywo-

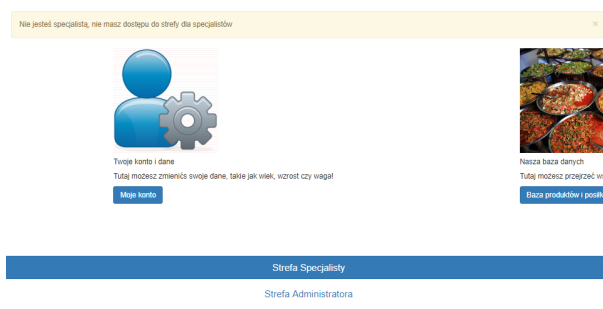


Rysunek 5.10 Dowolna usługa - osoba niezalogowana

ływany jest niezależnie od sposobu, w jaki użytkownik próbuje dostać się na niedostępną dla niego stronę - czy to używając dostępnych przycisków, czy wpisując adres strony w wierszu przeglądarki. Weryfikacja ta zrobiona jest w warstwie aplikacji (backendu), co utrudnia możliwość jej obejścia poprzez wykorzystanie metod javascriptowych.

5.2.2 Zwykły użytkownik

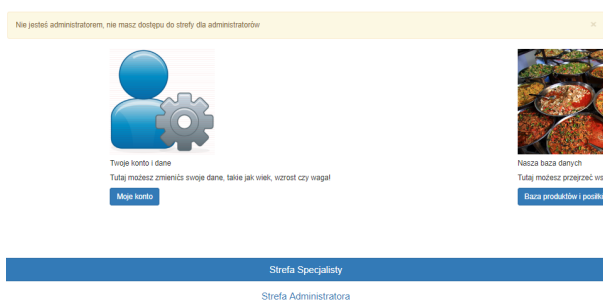
Zwykły użytkownik ma dostęp do większości usług dostępnych w aplikacji. Jego jedynymi ograniczeniami są strefy specjalisty i administratora. Kiedy zwykły użytkownik próbuje się do nich dostać Otrzymuje informacje o tym, że nie ma odpowiednich uprawnień, tak jak jest to pokazane na *Rysunku 5.11*



Rysunek 5.11 Strefa specjalisty - brak uprawnień

5.2.3 Specjalista

Specjalista ma dostęp do tych samych usług, co zwykły użytkownik, jednak ma również dostęp do strefy specjalisty. Kiedy jednak klika na przycisk przekierowujący do strefy admina, lub próbuje się do niej dostać poprzez wpisanie linku w przeglądarce dostaje informacje ukazana na *Rysunku 5.12*



Rysunek 5.12 Strefa administratora - brak uprawnień

5.2.4 Zabezpieczenia danych

Zgodnie z ustaleniami jedyną niezbędną do zaszyfrowania informacją jest hasło, którego znajomość umożliwia dostanie się do konta użytkownika bez wiedzy jego posiadacza, a tym samym poznanie jego danych bez jego wiedzy. Aby temu zapobiec wykorzystano szyfrowanie SHA256, co można było zobaczyć w załączonym wcześniej widoku panelu administratora.

5.2.5 Walidacja danych

W aplikacji rozszerzono walidację danych, która realizowana była już wcześniej, podczas implementacji bazy danych. W momencie, w którym dane nie spełniają odpowiednich wymagań aplikacja informuje o tym użytkownika, dla każdego dostępnego w aplikacji formularzu. Informacje o błędach wyglądają tak, jak na przykładowym *Rysunku 5.13*

Rysunek 5.13 Walidacja danych w aplikacji

5.3 Wnętrze aplikacji - kod źródłowy

5.3.1 Podstawowe mechanizmy

Aplikacja pisana jest w języku Python3 z użyciem biblioteki Flask. Aby z niej korzystać wystarczy więc napisać proste funkcje, które będą miały zdefiniowane ścieżki i wykorzystywały stworzone szablony w języku HTML, z wykorzystaniem CSS oraz javascriptu. Bazowym kodem tworzącym widok na podstawie szablonu HTML jest kod pokazany w *Rysunku 5.14* Warto również zauważyć, że każda sesja aplikacji (można mieć otwartych

```
@app.route("/")
@app.route('/main/')
@app.route('/homepage/')
def homepage(request):
    return render_template("main.html")
```

Rysunek 5.14 Podstawowa funkcja renderująca widok

wiele sesji na raz - bowiem z aplikacji może korzystać wielu użytkowników) ma swoje własne dane, zapisane w słowniku o nazwie *session*. Znajdują się w nim: login użytkownika, jego poziom uprawnień a także informacja o tym, czy jest on zalogowany.


```
session['logged_in'] = True
session['username'] = request.form['username']
session['permission'] = permission
flash('#udało się zalogować #') + str(session['username'])
```

Rysunek 5.15 Dane zawarte w sesji

5.3.2 Formularze do wprowadzania i edycji danych

Aby stworzyć formularze, które współpracują z bazą danych i przekazują dane z kodu źródłowego. W tym celu wykorzystano bibliotekę wtforms3, dostępną w języku python. Udostępnia ona bazowe formularze, a także przekazywanie informacji między wierzchnią warstwą aplikacji a jej wewnętrznym kodem źródłowym. W celu lepszego wykorzystania owych formularzy stworzone zostały własne klasy. Formularze te umożliwiają jednocześnie

```
class UserActivitiesForm(Form):
    name = StringField('Imię', [validators.Length(min=2, max=20), validators.DataRequired()])
    surname = StringField('Nazwisko', [validators.DataRequired()])
    password = PasswordField('Hasło')
    age = IntegerField('Wiek', [validators.NumberRange(18, 200, "Niepoprawny wiek"), validators.DataRequired()])
    weight = IntegerField('Waga', [validators.NumberRange(5, 450, "Niepoprawna waga"), validators.DataRequired()])
    height = IntegerField('Wzrost', [validators.NumberRange(5, 450, "Niepoprawny wzrost"), validators.DataRequired()])
    sex = BooleanField('Zaznacz to pole, jeśli jesteś mężczyzną')
```

Rysunek 5.16 Formularz w języku python

przeprowadzanie walidacji danych i przygotowania komunikacji błędu w razie nie spełnienia wymaganej formy danych. Ważna jest jednak również umiejętność obsługi danych w szablonie HTML:

```
<body>
<div class="container">
<div id="form" id="main" id="main">
<div>
<div class="form-group">
<input type="text" class="form-control" style="" placeholder="Imię" name="name" value="{{form.name}}">
</div>
</div>
<div class="form-group">
<input type="text" class="form-control" style="" placeholder="Nazwisko" name="surname" value="{{form.surname}}">
</div>
</div>
<div class="form-group">
<input type="password" class="form-control" style="" placeholder="Hasło" name="password" value="{{form.password}}">
</div>
</div>
```

Rysunek 5.17 Obsługa formularza ze strony HTML

5.3.3 Dekoratory

W celu obsługi uprawnień przygotowano wygodne w użyciu dekoratory, korzystające z danych zawartych w każdej z sesji. Dekoratory w języku Python są funkcjami, sprawdzającymi pewne warunki przed wykonaniem funkcji docelowych.

```
def is_specialist(f):
    @wraps(f)
    def wrap(*args, **kwargs):
        if session['permission'] >= 1:
            return f(*args, **kwargs)
        else:
            flash("Nie jesteś specjalistą, nie masz dostępu do strony dla specjalistów")
            return redirect(url_for('homepage'))
    return wrap

def is_admin(f):
    @wraps(f)
    def wrap(*args, **kwargs):
        if session['permission'] == 2:
            return f(*args, **kwargs)
        else:
            flash("Nie jesteś administratorem, nie masz dostępu do strony dla administratorów")
            return redirect(url_for('homepage'))
    return wrap
```

Rysunek 5.18 Dekoratory w języku python

5.3.4 Algorytm dobierający diety

Kolejną częścią programu jest wbudowany algorytm dobierający diety. W tym celu musi on liczyć zapotrzebowanie na energię, a także liczyć energię z każdego posiłku. Wykorzystywana jest do tego wiedza, o ilość kalorii przypadających na jeden gram tłuszczu, węgli czy białka. Część działania tego algorytmu, współpracującego z bazą danych przedstawiono poniżej:

```
def find_wfc_by_id(id):
    cursor.execute(
        "SELECT bialko_w_100g, tluszcz_w_100g, wegla_w_100g FROM produkty WHERE id_produkta = {}".format(id))
    result = cursor.fetchall()
    try:
        return result
    except Exception as e:
        print(e)
        pass

def return_calories_from_ingredients(data):
    return data[0][0]**4 + data[0][1]**4 + data[0][2]**4

def count_calories_in_product_by_product_id(id):
    makro = find_wfc_by_id(id)
    return return_calories_from_ingredients(makro)

def count_calories_in_meal_by_meal_id(id):
    ingredients = find_row_meal_by_id(id)
    sum = 0
    for ingredient in ingredients:
        sum = sum + count_calories_in_product_by_product_id(ingredient)
    return sum
```

Rysunek 5.19 Algorytm liczący kalorie danego posiłku

```
arr = []
for ingredient in ingredients:
    data = find_wfc_by_id(ingredient)
    w = data[0][0]
    t = data[0][1]
    c = data[0][2]
    arr.append(w, t, c)
return arr

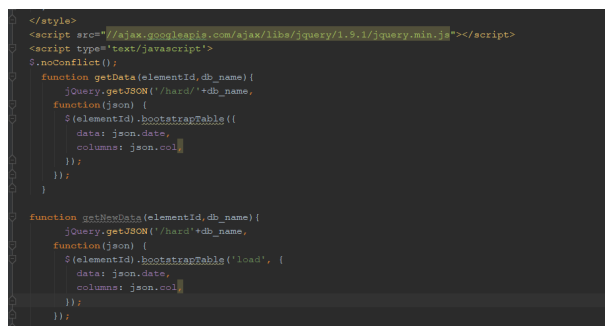
def get_diet(login):
    personal_data = find_for_algorithm(login)
    #personal_data = personal_data
    activities = find_activity_id_and_time_per_week_by_login(login)
    calories = 0
    for activity, hours in activities:
        calories = calories + int(find_energy_by_id(activity)) * hours

    #personal_data
    sex_factor = -16 if personal_data[0][5] == 'k' else 5
    daily_energy = 9.99*personal_data[0][3] + 6.25*personal_data[0][4] - 4.92*personal_data[0][2] + sex_factor
    whole_daily_energy = daily_energy * calories/7
    energy_per_meal = whole_daily_energy/5
```

Rysunek 5.20 Algorytm liczący zapotrzebowanie na wartość energetyczną

5.3.5 Wykorzystany javascript

W celu renderowania i obsługi bazy danych i wyświetlania bazy produktów wykorzystano funkcje javascriptowe napisane własnoręcznie, które wywoływały funkcję z kodu wysyłającą zapytanie do bazy danych. Przykładowa funkcja javascriptowa przedstawiona została na *Rysunku 5.21*



```
</style>
<script src="//ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js"></script>
<script type="text/javascript">
$.noConflict();
function getData(elementId,db_name){
    jQuery.getJSON("/hard/"+db_name,
        function(json) {
            $(elementId).bootstrapTable({
                data: json.data,
                columns: json.col1
            });
        });
}

function getNewData(elementId,db_name){
    jQuery.getJSON("/hard/"+db_name,
        function(json) {
            $(elementId).bootstrapTable('load', {
                data: json.data,
                columns: json.col1
            });
        });
}
```

Rysunek 5.21 Algorytm liczący zapotrzebowanie na wartość energetyczną

5.4 Wdrażanie do systemu online

W związku z tym, iż System Diet jest aplikacją Webową pisaną w języku Python3 nie ma żadnego problemu z wdrożeniem go i włączeniem na serwerze zdalnym. Wystarczy do tego dowolna maszyna Wirtualna, Docker lub podobne oprogramowania, realizujące systemy operacyjne, na których pojawia się system Linux. System Linux obsługuje bowiem typowe przeglądarki i ma wbudowany język Python3, wykorzystany do napisania tej aplikacji. Aby aplikację można było uruchomić na serwerze zdalnym wystarczy więc skopiować jej pliki oraz bazę danych i nie trzeba nawet konfigurować środowiska. Jedynym problemem mógłby być brak bibliotek takich jak: Flasj, Wtforms3 czy jinja2, jednak zainstalowanie każdej z nich wiąże się z wpisaniem jednej komendy w terminalu linuxowym (np. pip3 install flask). W celu sprawdzenia możliwości wdrożenia aplikacja uruchomiona została na systemie operacyjnym Raspbian i Linux i w obu przypadkach działała bez zarzutów.

Rozdział 6

Podsumowanie

6.1 Podsumowanie i wnioski

Podczas projektu zmagano się z wieloma dziedzinami techniki, ale nie tylko - tworzono bazę danych, projektowano stronę internetową a także sięgnięto po wiedzę dietetyczną, w której autorzy projektu nie są specjalistami. Każda z tych gałęzi wymagała sporego nakładu pracy, ze względu na niekompletną wiedzę, a także na konieczność spełnienia wszystkich założeń funkcjonalnych oraz нефункциональных.

Należy jednak stwierdzić, że wszystkie założenia postawione w trakcie fazy projektowania zostały zrealizowane w ostatecznej wersji projektu. Nie było konieczności rezygnacji z niektórych komponentów. Spowodowane to było dużą determinacją oraz samorozwojem w trakcie pracy nad produktem.

Pierwszy projekt oparty o bazę danych realizowany przez grupę trzyosobową pozwolił na zdobycie doświadczenia pracy w zespole a także poznanie różnych problemów z tym związanych.

Pierwszymi problemami było mieszczanie się w terminach oddawania kolejnych faz projektów, co spowodowane było brakiem wiedzy w danych dziedzinach. W trakcie nauki ciężko było realizować kolejne etapy na czas - co skutkowało opóźnieniem. Jednak mimo wszystko cały projekt udało się dostarczyć w wyznaczonym terminie.

Kolejnym problemem były części projektu, nad którymi wymagana była praca wspólna - część dotycząca projektowania i tworzenia założeń. Aby ją zrealizować autorzy projektu nie mogli podzielić się pracą, lecz zobowiązani byli do znalezienia wspólnego czasu.

Działająca aplikacja mimo spełniania wszystkich wymagań funkcjonalnych i нефункциональных nadal może być rozbudowywana. Brak specjalistycznej wiedzy na temat dietetyki sprawia, że wykorzystany algorytm doboru diet i obliczania zapotrzebowania kalorycznego może wciąż być poprawiany i rozwijany w konsultacji z osobami mającymi kompetencje w wyżej wymienionej dziedzinie.

Po zdobyciu doświadczenia można również stwierdzić iż sam projekt bazy danych, a także sposób jej obsługi można poprawić i w razie potrzeby zwiększyć jej wydajność - aczkolwiek nie jest to obecnie wymagane, ze względu na stosunkowo niewielką liczbę danych przechowywanych w bazie. Ze względu na potrzeby kursu cała aplikacja oraz baza danych działają lokalnie, jednak po testach na RaspberryPi3 można stwierdzić iż umieszczenie ich na stronie ogólnodostępnej nie stanowi problemu.

Ostatecznie należy zaznaczyć iż przygotowanie działającej aplikacji podłączonej do bazy danych w takim wykonaniu, z brakiem początkowej wiedzy jest dużym sukcesem.

Bibliografia

- [1] Dokumentacja języka python - <https://docs.python.org/3/>
- [2] Dokumentacja biblioteki Flask - <http://flask.pocoo.org/docs/1.0/>
- [3] Żywnościowa baza danych Kcalmar - <https://kcalmar.com/>
- [4] Kalkulator energetyczny <https://www.fabrykasily.pl/bmr>
- [5] Książka poszerzająca wiedzę o pythonie - Python notes for professionals
- [6] Wzór obliczania zapotrzebowania kalorycznego - <https://wformie24.poradnikzdrowie.pl>