КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет комп'ютерних наук та кібернетики
Кафедра інтелектуальних програмних систем

Кваліфікаційна робота

на здобуття рівня магістра

за спеціальністю 121 «Інженерія програмного забезпечення»

на тему:

РОЗРОБКА СИСТЕМИ АНАЛІЗУ І КОНТРОЛЮ ЗДОРОВ'Я ЛЮДИНИ НА ОСНОВІ ЖИТТЄВО ВАЖЛИВИХ ПОКАЗНИКІВ

Виконала студентка 2-го курсу магістратури
Юлія ТОКАН _____
(підпис)

Науковий керівник:
к.т.н., доцент
Євген ДЕМКІВСЬКИЙ _____
(підпис)

Засвідчую, що в цій роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Роботу розглянуто й допущено до захисту на засіданні кафедри інтелектуальних програмних систем
« 10 » травня 2023 р., протокол № 9

Завідувач кафедри
О. ПРОВОТАР _____
(підпис)

Київ – 2023

РЕФЕРАТ

Обсяг роботи 60 сторінок, 14 ілюстрацій, 13 таблиць, 15 джерел посилань, 3 додатки.

JAVA, SPRING FRAMEWORK, POSTGRESQL, JAVASCRIPT, HTML, CSS, ВЕБРОЗРОБКА, ЗРОДОВ'Я, ПРОГРАМНИЙ ПРОДУКТ.

Об'єктом роботи є процес відслідковування щоденних звичок здорового харчування та оцінка наповненості раціону нутрієнтами, контролю вживання алергенів, регулярності харчування та режиму споживання води. Предметом роботи є застосунок, що забезпечує можливістю контролювати збалансованість харчування і вплив його на здоров'я людини, моніторити вживання калорій, поживних речовин, алергенів та води.

Метою кваліфікаційної роботи є розробка програмного продукту «EatingWell», який реалізовує функціонал щоденника здорового способу життя та надає можливості відслідковувати калорії та поживні речовини, уникати вживання продуктів та страв, що містять алергени, слідкувати за водним балансом.

Методи розроблення: розробка застосунку, основний функціонал якого ґрунтується на індивідуальному розрахунку щоденної норми калорій, нутрієнтів та аналізу вмісту алергенів у різних продуктах та стравах. Інструментами розроблення є інтегроване середовище розробки IntelliJ IDEA, мови програмування Java, JavaScript, HTML, фреймворк Spring.

Результати роботи та їх новизна: досліджено основні особливості архітектури та розробки застосунку, а також вимоги та програмні інструменти для його створення. Визначено специфіку персоналізованих планів здорового способу життя на основі індивідуальних потреб та особливостей людини. Спроєктовано, розроблено та протестовано застосунок «EatingWell» для

аналізу і контролю здоров’я, враховуючи життєво важливі показники людини з урахуванням актуальних технологій розробки.

Застосунок «EatingWell» може використовуватися для контролю особистих параметрів (наприклад, вага) та здоров’я в цілому. Його також можна використовувати разом з рекомендаціями лікарів, дієтологів і спортивних тренерів для аналізу і контролю здоров’я, оптимізації схуднення або процесу набору м'язової маси. Також він є помічником при різних хворобах, наприклад, цукровий діабет або алергія.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

API – Application Programming Interface;

JDBC – Java DataBase Connectivity;

SQL – Structured Query Language;

HTTP – HyperText Transfer Protocol;

MVC – Model-View-Controller;

JPA – Java Persistence API;

CSS – Cascading Style Sheets;

HTML – HyperText Markup Language;

JS – JavaScript;

JSON – JavaScript Object Notation;

XML – Extensible Markup Language;

БД – База Даних;

СУБД – Система Управління Базами Даних;

IDE – Integrated Development Environment.

ВСТУП

Оцінка сучасного стану об’єкта розробки. Зараз здоровий спосіб життя та правильне харчування стали актуальними темами, оскільки люди все більше усвідомлюють важливість піклування про своє здоров'я. Неправильне харчування та нестача фізичної активності можуть призвести до багатьох захворювань, таких як ожиріння, діабет, серцево-судинні захворювання та інші [1].

Здоровий спосіб життя включає не тільки правильне харчування, але і регулярну фізичну активність, достатній відпочинок і сон, відмову від шкідливих звичок, таких як куріння та вживання алкоголю. Такий спосіб життя допомагає зміцнити імунну систему, покращити стан шкіри, волосся та нігтів, підвищити енергетичний рівень та настрій.

Крім того, здоровий спосіб життя може допомогти запобігти розвитку багатьох захворювань та покращити якість життя. Тому все більше людей прагнуть до такого способу життя і правильного харчування, що призводить до зростання популярності застосунків і сервісів, які допомагають людям стежити за своїм здоров'ям і досягати своїх цілей у цій галузі.

Люди, які мають алергічні реакції або проблеми зі здоров'ям також активно використовують ці програми, щоб відстежувати свої показники здоров'я та знаходити продукти, які не викликають у них алергічних реакцій або не погіршують їхній стан. Деякі застосунки навіть пропонують функціонал для введення інформації про здоров'я або медичні препарати, щоб користувачі могли отримувати персональні рекомендації та поради.

Актуальність роботи та підстави для її виконання. З точки зору еволюції фізичний і психічний розвиток суспільства пройшов дуже швидко і людина просто не встигла фізично та психологічно адаптуватися до нового ритму

життя. Розвиток технологій змушує суспільство концентруватися на роботі, досягненні цілей, особистісному розвитку – це все дається в знаки на стані здоров'я.

З розвитком торгівельних зв'язків люди, які живуть на Півночі, почали імпортувати продуктові товари з Півдня, і навпаки. Отже, люди почали отримувати продукти, до яких їх організм еволюційно ще не встиг адаптуватися. Також стрімкий розвиток отримала індустрія фаст-фуду, швидкої та дешевої їжі, адже це відповідає теперішньому способу життя та запиту суспільства. Такий шалений ритм життя, неправильне харчування і зростаючий рівень стресу призводить до виникнення патологій і змін навантажень на людський організм [2].

Багато хто декілька разів на місяць задається питанням здорового харчування, але просто не має часу та коштів на розробку індивідуального меню та контроль харчування зі спеціалістом. Тож можна зробити висновки, що суспільство потребує застосунок, який зможе контролювати процеси щоденних звичок в харчуванні та підбирати здорові варіанти харчування, відслідковувати регулярність прийомів їжі та водного режиму, контролювати вживання алергенів та оцінювати збалансованість і енергетичну цінність раціону.

Мета й завдання роботи. Метою кваліфікаційної роботи є розробка застосунку «EatingWell» для аналізу і контролю здоров'я людини на основі життєво важливих показників.

Завдання для досягнення цієї мети:

- вивчити основні принципи розробки застосунків;
- здійснити аналіз джерел і матеріалів принципів розробки застосунку для підтримки і контролю здорового способу життя;
- визначити основні технології для проєктування та реалізації застосунку;
- сформувати основні фактори впливу на процес щоденних звичок здорового харчування;

– розробити технічне завдання до програмної системи;
– розробити інтерфейс, дизайн та бізнес-логіку застосунку.

Об'єкт, методи й засоби розробки. Об’єктом розроблення є процес відслідковування та аналізу щоденних звичок здорового способу життя та харчування, оцінка енергетичної наповненості раціону, контролю вживання алергенів, регулярності вживання їжі та водного режиму.

Предметом роботи є застосунок, що забезпечує можливістю контролювати збалансованість харчування і вплив його на здоров’я людини, моніторинг вживання калорій, поживних речовин та води.

Розробці програмного засобу передував аналіз джерел, матеріалів та досліджень, присвячених особливостям щоденників здорового способу життя, а також вивчення, аналіз та порівняння найпоширеніших застосунків для підтримки і контролю здоров’я людини ;

Як інструмент розроблення програмного продукту було обрано IntelliJ IDEA IDE – інтегроване середовище розробки від компанії JetBrains [3].

Можливі сфери застосування. Області використання застосунків для підтримки здорового способу життя можуть включати трекери фітнесу, програми для прийому їжі, моніторингу активності. Також застосунок може використовуватися як засіб для перегляду нутрієнтів, моніторингу ваги, планування правильного харчування, пошуку заходів для здорового способу життя.

РОЗДІЛ 1.

АНАЛІЗ ІСНУЮЧИХ ПРОГРАМНИХ СИСТЕМ

Нині в мережі Інтернет існує значна кількість різних застосунків для аналізу здоров’я людини на основі життєво важливих показників і контролю харчування. Розглянемо найпоширеніші з них в деталях, визначимо їх переваги і недоліки.

1.1 Програма «NutraCheck»

За версією Forbes Health передову позицію в переліку найкращих застосунків для ведення щоденника харчування займає «NutraCheck» [5]. Цей застосунок було розроблено для того, щоб допомогти людям контролювати свій раціон та стежити за якістю харчування. За його допомогою користувачі можуть відстежувати кількість спожитих калорій, білків, жирів та вуглеводів у реальному часі. Застосунок надає інформацію про харчову цінність продуктів та дозволяє вести щоденник харчування.

Однією з головних рис застосунку «NutraCheck» є функція сканування штрих-кодів продуктів. Це дозволяє швидко і легко додавати продукти до щоденника харчування, не витрачаючи час на пошук потрібної інформації в базі даних програми.

Застосунок також пропонує персоналізовані рекомендації щодо харчування на основі даних про користувача, таких як стать, вік, вага та рівень фізичної активності.

Програма доступна як вебсайт, а також як iOS та Android застосунки. Має безкоштовну семиденну пробну версію, а також платну версію з

додатковими функціями, такими як відстеження макронутрієнтів та аналіз споживаних вітамінів та мінералів.

Приклад користувацького інтерфейсу програми «NutraCheck» наведено на рис. 1.1.

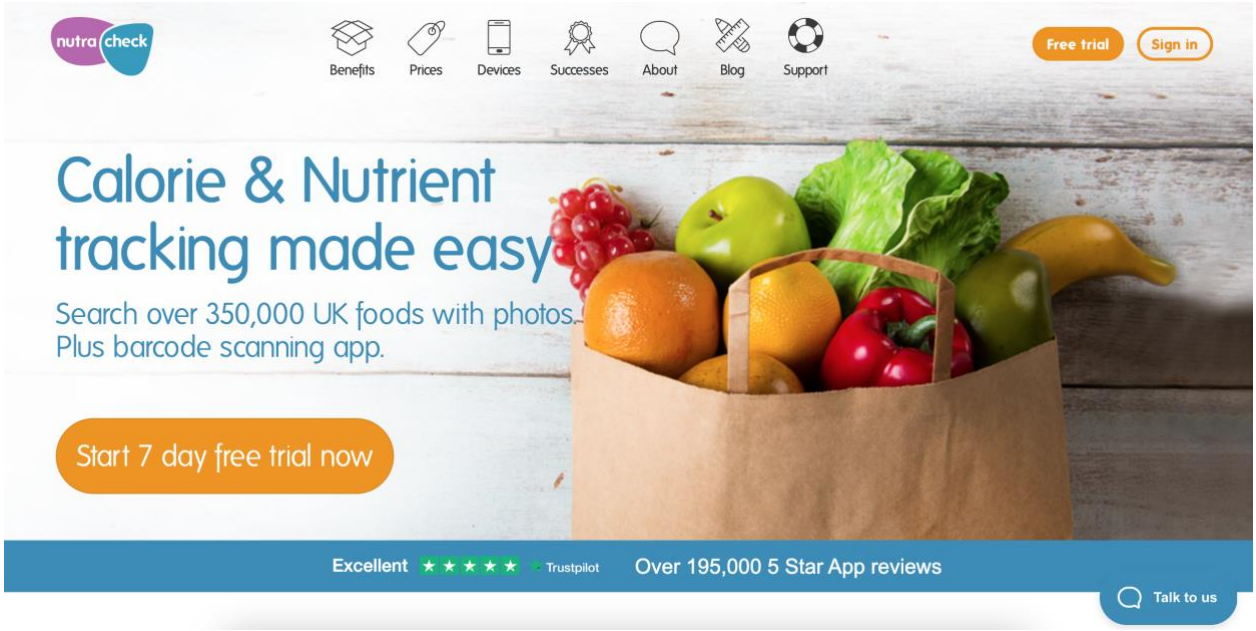

Рисунок 1.1 – Головна сторінка програми «NutraCheck»

1.2 Програма «My Macros + Diet and Calories»

Програма «My Macros + Diet and Calories» – це застосунок для контролю харчування, який допомагає користувачам відстежувати та контролювати споживання калорій, білків, жирів і вуглеводів [6].

Однією з головних особливостей програми є можливість налаштування індивідуальних цілей споживання різних макронутрієнтів. Користувач може задати свої цілі відповідно до рівня фізичної активності, цілей зміни ваги та інших параметрів.

Для зручності відстеження споживаних продуктів «My Macros + Diet and Calories» також надає функцію сканування штрих-кодів продуктів. Користувач може швидко додати продукти в щоденник харчування та побачити інформацію про калорії та макронутрієнти в реальному часі.

Застосунок також пропонує графічні звіти та статистику щодо споживаних калорій та макронутрієнтів за певний період часу. Це дозволяє користувачам краще розуміти свої харчові звички та приймати усвідомленіші рішення про своє харчування.

«My Macros + Diet and Calories» доступно як вебзастосунок, а також на iOS та Android платформах і має безкоштовну версію, а також платну версію з додатковими функціями, такими як автоматичний розрахунок оптимального споживання макронутрієнтів та налаштування індивідуальних цілей для кожного прийому їжі.

Серед недоліків – деякі користувачі вважають, що програма надто складна у використанні, особливо для початківців. Також безкоштовна версія програми має обмежений функціонал, що може бути незручно для користувачів, які хочуть отримати більш детальну інформацію про своє харчування. Крім того, не всі продукти можуть бути знайдені в базі даних, що може призвести до неточних результатів при розрахунку калорій і макронутрієнтів. Приклад користувацького інтерфейсу програми «My Macros + Diet and Calories» наведено на рис. 1.2.

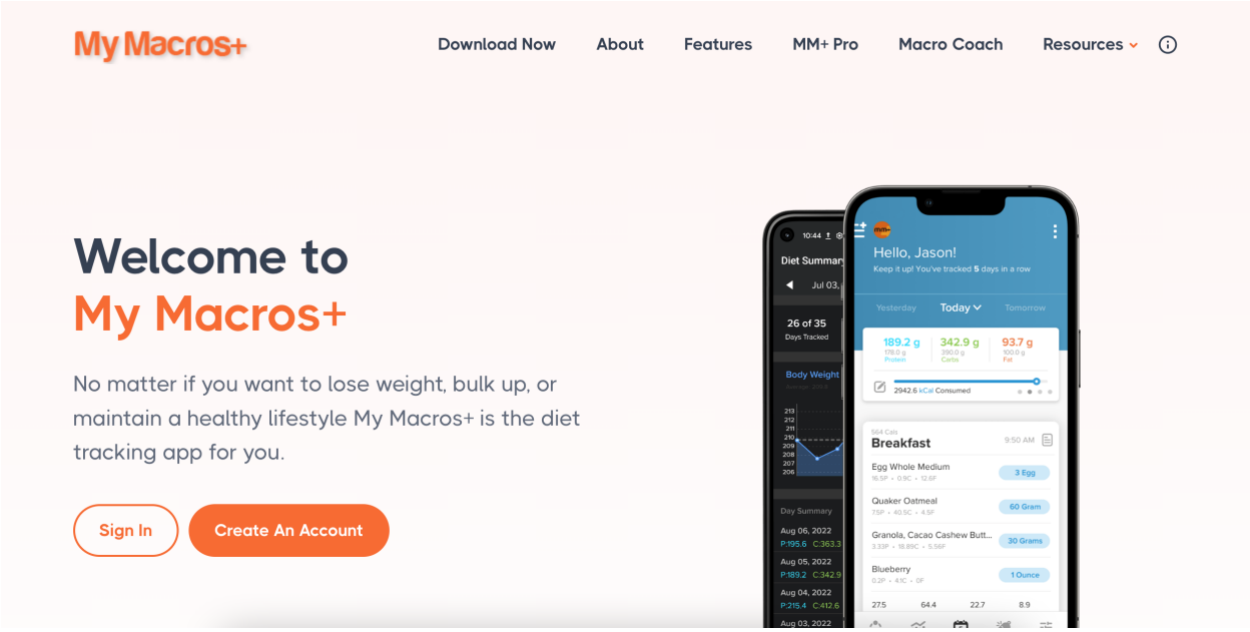

Рисунок 1.2 – Головна сторінка програми «My Macros + Diet and Calories»

1.3 Система «MyFitnessPal»

Система «My FitnessPal» [7] дозволяє розрахувати всі показники, що стосується фігури, ваги і способу життя. Вона є простим і мінімалістичним застосунком, що включає всі найнеобхідніші функції для підтримки здорового способу життя та ведення щоденника харчування.

Розробники стверджують, що база даних системи є найбільшою серед аналогів (близько 11 мільйонів різних страв і продуктів, включаючи ресторанні страви). Додаток пропонує оптимальний функціонал для додавання продуктів і страв, статистику і звіти про динаміку ваги, основні поживні речовини, такі як білки, жири і вуглеводи, а також сканер штрих-кодів.

Ще однією перевагою системи є те, що її можна синхронізувати з вебдодатком, що дозволяє користувачам вести щоденник як з комп'ютера, так

і з мобільного телефону. Система є безкоштовною, але можна придбати розширену підписку, яка надає доступ до додаткового функціоналу. Недоліком є те, що систему не можна синхронізувати з індивідуальними фітнес-трекерами.

Приклад користувацького інтерфейсу системи «My FitnessPal» наведено на рис. 1.3.

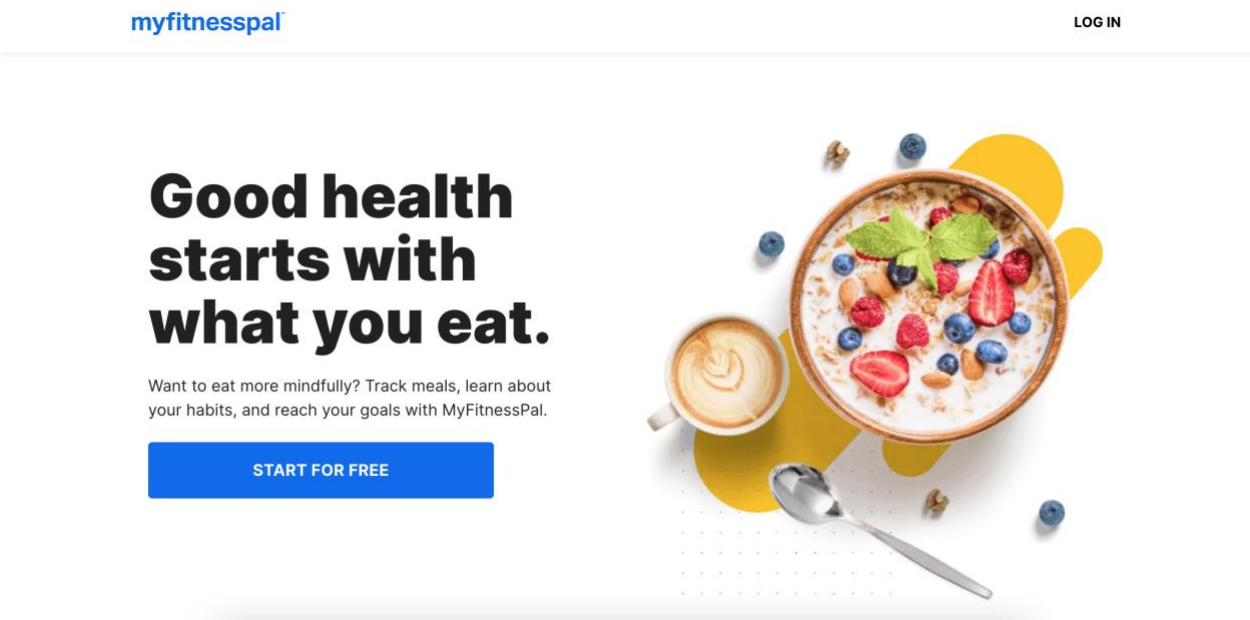

Рисунок 1.3 – Головна сторінка системи «My FitnessPal»

1.4 Система «Nutritionix Track»

«Nutritionix Track» – це ще одна система для відстеження харчування та обліку фізичних активностей.

Однією з особливостей «Nutritionix Track» є його база даних продуктів, яка включає понад 800 000 продуктів, включаючи ресторанну їжу [8]. Користувачі можуть також створювати власні страви та додавати їх до бази

даних. «Nutritionix Track» також має функцію сканування штрих-кодів продуктів, що робить процес внесення даних до щоденника харчування більш зручним та швидким. Ще однією перевагою системи є те, що вона надає більш точну інформацію про поживні властивості продуктів, ніж інші програми. Також, застосунок, у порівнянні з іншими, є простішим у використанні для користувачів, особливо для початківців.

Однак, недоліком «Nutritionix Track» є те, що його база даних продуктів не така широка, як у «MyFitnessPal», і може бути відсутня інформація про деякі продукти. Крім того, безкоштовна версія також має обмежений функціонал, і користувачі повинні платити за доступ до деяких додаткових функцій.

Приклад користувацького інтерфейсу програми «Nutritionix Track» наведено на рис. 1.4.

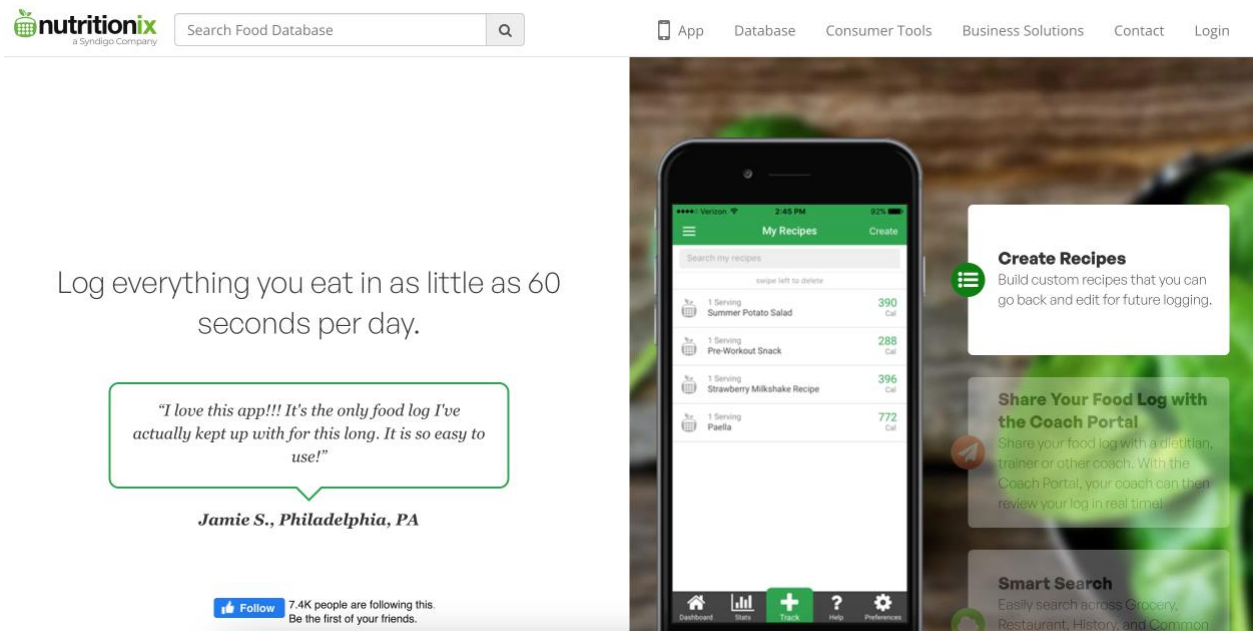

Рисунок 1.4 – Головна сторінка сайту «Nutritionix Track»

1.5 Порівняння розглянутих програмних систем

На основі розглянутої інформації про застосунки можна зробити висновок, що існує різниця в їх функціоналі та інтерфейсі, хоча вони спроєктовані і розроблені для досягнення майже однакової мети. Оскільки інформація про процеси проєктування та розробку архітектури розглянутих застосунків недоступна, можна порівняти їх на основі таких характеристик як зручність користування, базовий функціонал, додатковий функціонал, простота використання, синхронізація з іншими системами, безкоштовне користування (див. табл. 1.1).

Таблиця 1.1 – Порівняння застосунків

Назва	**NutraCheck**	**My Macros + Diet and Calories**	**MyFitnessPal**	**Nutritionix Track**
Зручність користування	+	+/-	+/-	+
Базовий функціонал	+	+	+	+
Додатковий функціонал	+	-	+	+/-
Простота використання	+	+/-	+/-	+
Синхронізація з іншими системами	-	-	+	-
Безкоштовне користування	+/-	+	+/-	+

Спираючись на порівняння характеристик, що наведені в табл. 1.1, можна вважати, що при аналізі даних параметрів кращою програмою є «NutraCheck», навіть не беручи до уваги деякі недоліки. Тому, при розробці власного застосунку буде враховано цей варіант, та будуть виправлені всі недоліки.

Відповідно до дослідження застосунків для аналізу і контролю здоров’я людини на основі життєво важливих показників, що були розглянуті, можна вважати, що варто реалізувати застосунок, який максимально поєднає переваги розглянутих рішень, проте необхідно виправити усі недоліки та проблеми, а також надати додатковий функціонал.

РОЗДІЛ 2.

ПРИЗНАЧЕННЯ І ЦІЛІ РОЗРОБКИ ЗАСТОСУНКУ

2.1 Призначення застосунку

Задачею даної кваліфікаційної роботи є реалізація застосунку, що надає інструменти та ресурси для контролю та покращення свого здоров’я, враховуючи переваги на недоліки уже готових рішень. Крім того, він може надавати інформацію про харчові цінності різних продуктів, а також рекомендації щодо складання здорового раціону харчування. Розробка повинна задовольняти наступним вимогам:

- надавати зручні інтерфейс та середовище для постійного використання застосункау;
- надавати доступ до інформації;
- надавати можливість здійснювати контроль і аналіз складових здорового способу життя, що задовольняють індивідуальні потреби;
- надавати можливість відслідковувати вживання алергенів або чутливість, яку користувач можете мати до різних продуктів та страв;
- надавати можливість слідкувати за харчовими звичками;
- надавати можливість стежити за прогресом у досягненні будь-яких цілей з покращення стану здоров’я;
- надавати можливість аналізувати споживання їжі, кількість поживних речовин і калорій;
- надавати можливість контролювати кількість спожитої води;
- надавати можливість відстежувати активно проведений час.

Проаналізувавши існуючі рішення та способи їх використання, можна зробити висновок, що доцільно буде розробити продукт зі зручним інтерфейсом, достатнім функціоналом та з простим управлінням. Така система є дуже затребуваною та корисною, оскільки забезпечує користувачам зрозумілий та зручний доступ.

Тому очікується отримати застосунок з можливістю контролю і аналізу здоров’я, харчування та вживання поживних речовин, балансу рідини у організмі та моделювання рекомендацій на основі даних про здоров’я та фізичну активність користувача. Такий застосунок призначений допомогти людям стежити за своїм харчуванням, контролювати життєво важливі показники і покращити своє здоров'я.

2.2 Цілі розробки застосунку

Цілі розробки застосунку «EatingWell»:

– організації щоденника здорового способу життя з підтримкою підрахунку калорій та поживних речовин;
– забезпечення розрахунку добової норми калорій на основі індивідуальних параметрів користувача;
– надання списку продуктів та страв, що є безпечними та не несуть шкоди для здоров’я;
– стеження за раціоном харчування та споживання їжі, що задовольняють індивідуальні харчові потреби;
– забезпечення підтримки водного балансу;
– надання можливості вести облік прийомів їжі;

– надання рекомендацій щодо харчування на основі даних про користувача, таких як стать, вік, вага та рівень фізичної активності;
– надання зручної і зрозумілої графіки, яка допоможе покращити якість харчування та здорового способу життя.

Застосунок допоможе вести облік спожитих калорій і орієнтуватися у виборі страв та продуктів, що є корисними для користувача. На основі даних про спожиті поживні речовини та воду система покаже скільки користувач вже спожив і скільки ще залишилося до його індивідуальної добової норми. Денна норма нутрієнтів та води розраховується на початку, враховуючи індивідуальні параметри і цілі, такі як стать, вік, вага та рівень фізичної активності.

Також застосунок допоможе з пошуком рецептів та складанням здорового раціону. Увесь функціонал застосунку представлений на use-case діаграмі (див. додаток А).

2.3 Вимоги до застосунку

2.3.1 Вимоги до застосунку в цілому

Застосунок «EatingWell» повинен бути щоденником харчування, де користувач може записувати всі спожиті продукти, страви та кількість випитої води, на основі цієї інформації має вестись розрахунок споживаних калорій, білків, жирів, вуглеводів. Це представлено у вигляді таблиці або графіка. Застосунок повинен надавати інформацію про рецепти страв та їх зручний пошук.

В свою чергу необхідна реєстрація користувача та створення профілю, де користувач може вказати свій ріст, вагу, вік, рівень фізичної активності та інші параметри, які допоможуть визначити його норму калорій та поживних речовин. Тут він може вказати про свої харчові особливості, такі як алергії. Після реєстрації користувач повинен пройти автентифікацію та авторизацію перед використанням застосунку для підтвердження особистості та отримання прав доступу.

Користувачу доступна інформація про харчові цінності різних продуктів та страв, яка подана у вигляді каталогу продуктів, і включає інформацію про калорійність, вміст білків, жирів, вуглеводів, вітамінів та мінералів. Користувач може додавати власні продукти чи страви, вказуючи необхідну інформацію про них. Також існує користувач-адміністратор, який має доступ до статистичних даних (список усіх продуктів та страв).

Система має надавати рекомендації щодо складання здорового раціону харчування на основі даних про здоров'я та фізичну активність користувача. Це представлено у вигляді списку продуктів та рецептів, які відповідають потребам користувача. Інструменти для відстеження прогресу, такі як графіки та діаграми, допоможуть користувачеві бачити свій прогрес та мотивувати його на подальше покращення свого здоров’я.

2.3.2 Вимоги до функціоналу програми

Перелік основного функціоналу, що доступний адміністратору, наведено в табл. 2.1.

Таблиця 2.1 – Перелік основних функцій, що доступні адміністратору

Функція	**Опис**

Керування даними про продукти та страви	Додавання нового продукту чи страви (вказуючи усі параметри та показники), що будуть доступні для всіх користувачів системи
	Редагування інформації про продукт чи страву, зміна деяких їх параметрів
	Видалення інформації про продукт чи страву

Перелік основного функціоналу, що доступний користувачам, наведено в табл. 2.2.

Таблиця 2.2 – Перелік основних функцій, що доступні користувачу

Функція	**Опис**
Авторизація	Введення обов’язкової інформації, такої як логін та пароль
	Створення профілю, де користувачу слід вказати свій ріст, вагу, вік, рівень фізичної активності
	Редагування профілю
	Вихід з системи
Налаштування наявних продуктів та страв	Додавання нового продукту чи страви, вказуючи необхідну інформацію про них, що будуть доступними тільки даному користувачу
	Перегляд усіх продуктів та страв
Керування раціоном	Додавання інформації про споживання їжі (продукт чи страва, кількість або вага, час)
	Видалення інформації про споживання їжі

Відстеження прогресу	Відображення графіків та діаграм, що надають статистику по кількості вжитих калорій та поживних речовин за деякий проміжок часу
Керування рецептами	Пошук рецептів, що є безпечними та не несуть загрози для здоров’я
Контроль водного балансу	Додавання випитої води
	Відображення графіку випитої води та процентного співвідношення з денною нормою

2.3.3 Технічні вимоги

Необхідно розробити застосунок, що надає такий функціонал:

- Реалізувати систему на основі предметної області застосунку.
- Дані, що фігурують у застосунку, зберігати в БД PostgreSQL. Для доступу до даних використовувати API.
- Використовуючи мову програмування Java та фреймворк Spring реалізувати функціонал серверної частини застосунку.
- Використовуючи мови програмування JavaScript, HTML та CSS реалізувати функціонал клієнтської частини застосунку.
- Реалізувати роботу з кирилицею, в тому числі і при зберіганні інформації в базу даних.
- Створити документацію до коду.
- Застосунок має бути покрито Unit-тестами, а також слід провести модульне тестування та тестування навантаженням системи.
- Слід використовувати Log4j для обробки подій та помилок в системі.

- Необхідно забезпечити підтримку транзакцій та пагінації.
- Застосунок має коректно обробляти та реагувати на помилки і виключення різного роду.
- Реалізувати систему авторизації і аутентифікації.
- Використовувати сесії і фільтри у розробці бізнес логіки.
- Застосунок повинен коректно відображатися в сучасних інтернет-браузерах.
- На довільні помилкові дії користувача, що пов’язані з введенням хибних даних, уникнення заповнення обов’язкових полів в формах та інші, які можуть бути оброблені програмою, слід генерувати відповідні повідомлення про помилки.

РОЗДІЛ 3.

ВИБІР ПРОГРАМНИХ ЗАСОБІВ РОЗРОБКИ

Розглянемо засоби розробки, інструменти та технології, які дозволяють оптимальним чином розробити програмний продукт і задовольнити описані раніше вимоги. Було прийнято рішення брати до уваги і аналізувати найбільш поширені засоби в своїй категорії, оскільки проєкт обмежений невеликим терміном розробки.

3.1 Вибір мови програмування серверної частини

Важливим аспектом розробки інформаційних систем є проектування архітектури, оскільки це може вплинути на ефективність, масштабованість, безпеку, швидкість розробки та сумісність з іншими технологіями. Якщо помилки будуть допущені на етапі проектування і виявлені пізніше, їх виправлення буде дуже дорогим.

Саме вибір інструментів, мов, технологій та бібліотек програмування допоможе забезпечити оптимальну роботу серверної частини, зменшити кількість помилок та вразливостей, а також зробити процес розробки більш ефективним та продуктивним. При виборі мови програмування слід враховувати потреби проекту, його масштаб та специфіку, а також здатність програміста працювати з обраною мовою. Крім того, широкий спектр можливостей обраних бібліотек і технологій полегшує розширення, покращення та модифікацію системи.

Існують вимоги щодо вибору інструментів, мов програмування, технологій та бібліотек для реалізації системи:

– клієнт-серверна архітектура: застосунок повинен реалізовувати архітектуру клієнт-сервер, тому мова програмування повинна максимально спростити обмін запитами між сервером та клієнтом;
– робота з даними: застосунок працює з великим об’ємом даних, тому потрібно обрати механізм взаємодії з БД, що є максимально гнучким та прозорим. Це зменшить кількість коду програми та полегшить підтримку застосунку у майбутньому;
– наявність ресурсів: деякі мови програмування можуть вимагати більше ресурсів для запуску програми, тому необхідно враховувати наявність ресурсів на сервері;
– швидкість виконання: деякі мови програмування можуть бути швидше та ефективніші в роботі з базами даних та іншими функціями.

Java була розроблена в 90-х роках і досі є найпопулярнішою мовою програмування. З самого початку вона розроблялась в першу чергу для написання вебзастосунків, тому її технології надають розробнику широкий спектр можливостей для розробки вебсервісів, включаючи бібліотеки сторонніх розробників, які використовуються при роботі з Java. Вона є об’єктно орієнтованою і може працювати на будь-якій платформі, що робить її досить функціональною [9].

Мова програмування Java має багато переваг у порівнянні з іншими популярними мовами програмування високого рівня а саме:

– зрозумілий синтаксис: мова має синтаксис, що легко читається. Це дозволяє полегшити розуміння логіки роботи програми і прискорити процес написання коду;
– кросплатформна незалежність: завдяки JVM, неважливо, на якому пристрої чи під якою ОС буде запускатися програма;
– строга типізація даних: це означає, що тип даних змінних та параметрів буде відомий в момент компіляції. Така особливість

значно підвищує надійність коду та зменшує час, що витрачається на виявлення помилок, які проявляються на етапі компіляції;

– автоматичне управління пам’яттю: ця особливість дозволяє не турбуватися про виділення пам’яті та її подальше звільнення;
– велика кількість фреймворків, що працюють на базі мови програмування Java: наприклад, Spring та Hibernate.

3.2 Фреймворк Spring

З урахуванням вимог, що описані вище, для взаємодії з клієнтською частиною застосунку необхідно організувати обмін за принципом запит-відповідь, при цьому запити клієнта і відповіді сервера повинні бути у форматі HTTP-запитів. Такий функціонал реалізований у фреймворку Spring (див. рис. 3.1).

Spring Framework – універсальний додаток з відкритим кодом, який широко використовується в парі з Java для вирішення великої кількості різних задач [10]. Він є одним з найпоширеніших середовищ розробки, які використовується для розробки вебзастосунків.

Spring включає в себе велику кількість модулів, які можуть бути використані для вирішення тих чи інших проблем. Так, наприклад, Inversion of Control контейнер дозволяє конфігурувати компоненти застосунку та керувати життєвим циклом Java-об’єктів, Spring MVC використовують для створення всього, що пов’язано з мережею. Часто використовують також Spring Security та Spring Data модулі. Вони необхідні для створення авторизації і автентифікації користувача та взаємодії з поширеними ORM (Object-Relational Mapping) рішеннями і JDBC відповідно.

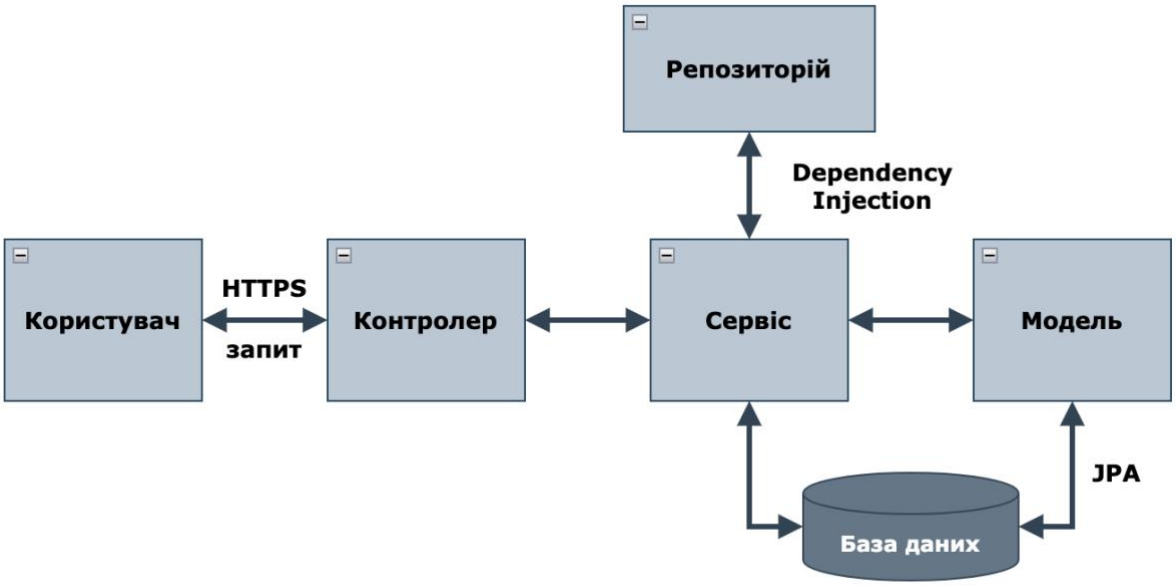

Рисунок 3.1 – Структура та модулі Spring застосунку

3.3 Вибір технології взаємодії з базою даних

Оскільки функціонал програми взаємодіє з базою даних, архітектура на стороні сервера повинна бути спроектована так, щоб максимально спростити роботу з базою даних, залишаючи при цьому можливість для подальшої модернізації, розширення або вдосконалення програмного продукту.

JDBC – це стандартний прикладний інтерфейс для взаємодії програм мовою Java з різними реляційними базами даних. Бібліотека JDBC надає набір класів та методів для підключення до бази даних, виконання SQL-запитів та отримання результатів, що розташовані у пакеті java.sql, що є компонентом Java SE. Для використання JDBC необхідно мати драйвер для відповідної бази даних. Основні компоненти бібліотеки JDBC: DriverManager, Connection, Statement, ResultSet [10].

3.4 База даних PostgreSQL

PostgreSQL – це об'єктно-реляційна система керування базами даних, яка підтримує безліч функцій і можливостей, таких як транзакції, процедури, що зберігаються, розширюваність і багато іншого [11]. Вона є однією з найвикористовуваніших систем керування базами даних у світі та використовується багатьма великими компаніями, такими як Apple, Cisco, Fujitsu та іншими.

Основні переваги PostgreSQL:

- надійність та цілісність даних: PostgreSQL забезпечує високий рівень надійності та цілісності даних завдяки використанню механізму транзакцій та контролю цілісності даних;
- розширюваність: PostgreSQL дозволяє створювати власні типи даних, функції та оператори, що робить його дуже гнучким і розширюваним;
- підтримка SQL: PostgreSQL повністю підтримує стандарт SQL і має множину розширень для роботи з даними;
- висока продуктивність: PostgreSQL має безліч оптимізації для роботи з великими обсягами даних і забезпечує високу продуктивність при роботі з ними;
- безкоштовність та відкритість: PostgreSQL є вільною та відкритою системою, що дозволяє використовувати її безкоштовно та вносити зміни до вихідного коду;
- підтримка багатьох операційних систем: PostgreSQL підтримує безліч операційних систем, включаючи Windows, Linux, MacOS та інші;
- збережені процедури: PostgreSQL підтримує збережені процедури, що дозволяє створювати більш складні програми та зменшити кількість звернень до бази даних;

В цілому, PostgreSQL – це дуже потужна та гнучка система керування базами даних, яка може бути використана для широкого спектру застосунків та проєктів.

3.5 Фреймворк Hibernate

Hibernate - це фреймворк для роботи з базами даних, який забезпечує об'єктно-реляційне відображення між об'єктами Java та таблицями бази даних. ORM - це підхід до роботи з базами даних, при якому об'єкти програми відображаються у вигляді таблиць у базі даних, а зв'язки між об'єктами реалізуються за допомогою зв'язків між таблицями.

Hibernate дозволяє розробникам працювати з базами даних на вищому рівні абстракції, не звертаючись безпосередньо до SQL. Це означає, що розробники можуть працювати з об'єктами Java, а Hibernate автоматично генерує SQL-запити для створення таблиць та збереження даних у них.

Одним з головних переваг Hibernate є можливість використовувати анотації для визначення структури таблиць та зв'язків між ними. Це дозволяє розробникам зосередитись на логіці програми, а не на деталях бази даних.

Hibernate підтримує різні стратегії кешування, що дозволяє покращити продуктивність додатку. Кешування - це процес збереження результатів запитів до бази даних в пам'яті, щоб уникнути повторного виконання запитів, якщо дані не змінилися. Hibernate також забезпечує можливість використання транзакцій для безпечної роботи з базою даних.

Hibernate є одним з найбільш популярних фреймворків ORM для розробки Java-додатків та широко використовується у великих проектах. Використання Hibernate дозволяє розробникам зосередитись на розробці

функціональності програми, а не на деталях бази даних, що забезпечує більш швидку та ефективну розробку.

3.6 Вибір мови програмування клієнтської частини

Клієнтська частина застосунку реалізує інтерфейс користувача, формує запити до сервера і обробляє відповіді від нього. Графічний інтерфейс користувача відображається в браузері. Користувач працює з застосунком саме через браузер, використовуючи посилання і функціонал вебсторінок.

HTML – це мова розмітки, яка використовується для створення вебсторінок. HTML дозволяє визначити структуру документа, включаючи заголовки, пункти, списки, таблиці та інші елементи. CSS – це мова стилів, яка використовується для оформлення вебсторінок. CSS дозволяє визначити кольори, шрифти, розміри та інші властивості елементів HTML.

Водночас HTML та CSS забезпечують можливість створення красивих та функціональних вебсторінок. HTML визначає структуру сторінки, а CSS – її зовнішній вигляд [12].

Для надання графічному інтерфейсу динамічності, використовуються додаткові бібліотеки та технології: скрипти JavaScript, Thymeleaf, а також вбудовані компоненти, створені за допомогою Java, Flash та інших.

Основна особливість мови JavaScript полягає в тому, що при використанні цієї мови елементи середовища відображення можуть бути змінені під час перегляду застосунку, тому вебсторінку не буде перезавантажено. Наприклад, за допомогою JavaScript можна змінити колір фону вебсторінки, замінити зображення, інтегровані у вебсторінку, створити нове вікно відображення, або відобразити повідомлення [13].

РОЗДІЛ 4.

РЕАЛІЗАЦІЯ ЗАСТОСУНКУ

4.1 Опис організації інформаційної бази

4.1.1 Логічна структура бази даних

СУБД, що використовується в застосунку, повинна надавати реляційну модель роботи з даними. Вона має свою структуру (таблиці) (див. додаток Б та табл. 4.1), що спрощує взаємодію з даними та їх зберігання.

Також при проєктуванні реляційної БД виділено такі етапи:

- визначення переліку таблиць та зв'язків між ними;
- визначення переліку стовпців, їх типів, ключів кожної таблиці, встановлення зв'язків між таблицями через зовнішні ключі;
- встановлення індексування у таблицях;
- встановлення обмежень цілісності для таблиць та зв'язків;
- нормалізація таблиць, коригування переліку таблиць та зв'язків.

Підсумовуючи вимоги, описані вище, обрано базу даних PostgreSQL. Так ця об'єктно-реляційна СУБД дозволить повністю сконцентруватися на розробці застосунку, уникаючи проблеми, що виникають при збереженні даних.

База даних складається з наступних таблиць, що представлені у табл. 4.1.

Таблиця 4.1 – Таблиці бази даних

Номер	**Таблиця**	**Опис**
1	users	Таблиця, що зберігає інформацію для автентифікації користувачів

2	clients	Таблиця, що зберігає інформацію про індивідуальні параметри користувачів
3	products	Таблиця, що зберігає інформацію про продукти та страви
4	meals	Таблиця, що зберігає інформацію про вживання їжі
5	eatPeriods	Таблиця, що зберігає інформацію про період споживання їжі
6	roles	Таблиця, що зберігає інформацію про ролі користувачів
7	genders	Таблиця, що зберігає інформацію про гендер користувача
8	nutritionGoals	Таблиця, що зберігає інформацію про типи мети використання додатку
9	activities	Таблиця, що зберігає інформацію про види активності
10	water	Таблиця, що зберігає інформацію про вживання води

4.1.2 Опис таблиць бази даних

Таблиці – це основа БД, де зберігаються всі дані. Спочатку необхідно спроєктувати структуру кожної таблиці, враховуючи формат виводу, запити і звіти, які будуть отримані при роботі з БД.

У таблицях 4.2-4.10 наведено інформацію про будову та конфігурацію таблиць БД (ім’я поля, тип та опис поля).

Структура таблиці «Користувач» представлена наступним чином (див. табл. 4.2):

Таблиця 4.2 – Опис таблиці Користувач

Атрибут	**Тип**	**Опис**
id	integer	Ідентифікатор (primary key)
loginName	varchar	Логін користувача
password	varchar	Пароль
roleId	integer	Ідентифікатор ролі (foreign key)
clientId	integer	Ідентифікатор клієнта (foreign key)

Таблиця складається з таких полів як Ідентифікатор, Логін, Пароль, Роль та Ідентифікатор клієнта. Таблиця «Користувач» містить зв’язок з таблицею «Клієнт» по ідентифікатору клієнта, що зберігає детальні дані, що стосуються кожного користувача (див. табл. 4.3).

Таблиця 4.3 – Опис таблиці Клієнт

Атрибут	**Тип**	**Опис**
id	integer	Ідентифікатор (primary key)
name	varchar	Ім’я користувача
genderId	varchar	Ідентифікатор статі (foreign key)
age	integer	Вік користувача
height	double	Ріст користувача
weight	double	Вага користувача
nutritionGoalId	integer	Ідентифікатор цілі (foreign key)
activityId	integer	Ідентифікатор активності (foreign key)
calories	integer	Денна норма калорій
protein	double	Денна норма білків
fats	double	Денна норма жирів

carbohydrates	double	Денна норма вуглеводів

Структура таблиці «Продукт» представлена наступним чином (див. табл. 4.4). Дана таблиця має зв’язок з таблицею «Користувач» по ідентифікатору користувача, так як виражає дані про продукт, що додається до системи окремим користувачем.

Таблиця 4.4 – Опис таблиці Продукт

Атрибут	**Тип**	**Опис**
id	integer	Ідентифікатор (primary key)
userId	integer	Ідентифікатор користувача (foreign key)
name	varchar	Ім’я продукту чи страви
calories	integer	Кількість калорій на 100 г.
protein	double	Кількість білків на 100 г.
fats	double	Кількість жирів на 100 г.
carbohydrates	double	Кількість вуглеводів на 100 г.
common	boolean	Загальнодоступність

Таблиця «Раціон» описує вживання їжі користувачем (див. табл. 4.5) і має зв’язки з таблицею «Користувач» по ідентифікатору користувача, «Продукт» по ідентифікатору продукту та з таблицею «Період» по ідентифікатору періоду (див. табл. 4.6).

Таблиця 4.5 – Опис таблиці Раціон

Атрибут	**Тип**	**Опис**
id	integer	Ідентифікатор (primary key)
userId	integer	Ідентифікатор користувача (foreign key)

productId	integer	Ідентифікатор продукту (foreign key)
weight	integer	Вага спожитого продукту чи страви
eatPeriodId	integer	Ідентифікатор періоду вживання їжі (foreign key)
date	timestamp	Дата

Таблиця 4.6 – Опис таблиці Період

Атрибут	**Тип**	**Опис**
id	integer	Ідентифікатор (primary key)
period	varchar	Назва періоду

Таблиці «Роль», «Гендер» та «Ціль використання» описують параметри клієнта та мають зв’язки з таблицею «Користувач» по своїм ідентифікаторам (див. табл. 4.7-4.9).

Таблиця 4.7 – Опис таблиці Роль

Атрибут	**Тип**	**Опис**
id	integer	Ідентифікатор (primary key)
role	varchar	Назва ролі

Таблиця 4.8 – Опис таблиці Гендер

Атрибут	**Тип**	**Опис**
id	integer	Ідентифікатор (primary key)
gender	varchar	Назва гендеру

Таблиця 4.9 – Опис таблиці Ціль використання

Атрибут	**Тип**	**Опис**
id	integer	Ідентифікатор (primary key)

goal	varchar	Назва цілі використання застосунку
coefficient	double	Коефіцієнт для розрахунку калорійності раціону

Структура таблиці «Вживання води» представлена наступним чином (див. табл. 4.10). Дана таблиця має зв’язок з таблицею «Користувач» по ідентифікатору користувача, так як відображає дані про споживання води, коли користувач додає інформацію про випиту воду.

Таблиця 4.10 – Опис таблиці Вживання води

Атрибут	**Тип**	**Опис**
id	integer	Ідентифікатор (primary key)
userId	integer	Ідентифікатор користувача (foreign key)
volume	integer	Об’єм випитої води

4.2 Розробка застосунку

Створення застосунку «EatingWell» проходило в декілька етапів:

1. Планування: визначення цілей, завдань та вимог до застосунку, аналіз конкурентів та цільової аудиторії, розробка концепції та дизайну.
2. Проектування: створення структури застосунку, вибір технологій, визначення функціональності та інтерфейсу, створення макетів та прототипів.
3. Розробка: написання коду, створення бази даних, інтеграція функціональності та дизайну, тестування та налагодження.

4. Наповнення контентом: створення текстів, зображень та відео для сайту, оптимізація контенту для пошукових систем.
5. Тестування та запуск: перевірка працездатності сайту на різних пристроях та браузерах, виправлення помилок, розміщення на хостингу.
6. Підтримка та розвиток: оновлення контенту, додавання нових функцій та покращень, аналіз результатів роботи сайту, оптимізація для покращення показників.

В наступних розділах можна знайти детальну інформацію про кожен з цих етапів.

4.3 Постановка завдання

Провівши аналіз предметної області можна виділити наступні завдання:

- автентифікація та авторизація користувачів;
- розрахунок добової норми калорій кожного користувача з урахуванням його параметрів (вік, зростання, вага, рівень активності);
- можливість користувача вести записи про своє харчування, включаючи інформацію про споживані продукти, кількість калорій та час прийому їжі;
- можливість встановити цілі споживання калорій і відстежувати їх досягнення;
- надавання рекомендацій щодо складання здорового раціону харчування на основі переваг користувача;

- можливість вводити алергени та уподобання в їжі, щоб застосунок міг рекомендувати страви, які не викликають алергічних реакцій та відповідатимуть смаковим уподобанням;
- забезпечувати можливістю створення персоналізованих планів харчування на основі індивідуальних потреб та цілей користувача.
- відображати статистику споживання калорій за день, тиждень та місяць;
- зберігати дані та відображати статистику водного балансу;
- можливість для адміністратора переглядати дані всіх продуктів та страв та керувати ними.

Підсумовуючи вимоги можна зробити висновок, що застосунок матиме зручний та простий у використанні інтерфейс користувача та функціонал, доступний з будь якого комп’ютера або мобільного пристрою із доступом до інтернету.

4.4 Структура застосунку

Програма, що є реалізацією даного застосунку, структурно складається з двох частин: клієнтської і серверної. В цілому, ці дві частини працюють у тісній взаємодії одна з одною, забезпечуючи користувачеві зручний інтерфейс та швидкий доступ до даних.

Обидві частини взаємодіють між собою за допомогою протоколу HTTP та передачі даних у форматі JSON або XML. Клієнтська частина надсилає запити на серверну частину, яка обробляє запити та повертає результати. Клієнтська частина потім обробляє ці результати та відображає їх на екрані для користувача.

4.4.1 Розробка серверної частини

Spring Framework, або просто Spring – один із найпопулярніших фреймворків для створення вебзастосунків на Java. [14]. Загальна структура Spring застосунку може бути представлена у вигляді тришарової архітектури: шар представлення (контролери та веб-сторінки), шар сервісів (бізнес-логіка) та шар доступу до даних (репозиторії). Структуру проєкту навдено у додатку В.

Розглянемо кожну частину програми окремо. Почнемо огляд з класів-моделей. Вони є шаблонними та описують структуру, за якою зберігаються дані.

Оскільки вони за структурою є звичайними Java-класами, то для роботи з полями класу необхідні конструктори, гетери та сетери. Для того, щоб спростити написання та читабельність коду було використано плагін Lombok, який за допомогою анотацій легко створює та інтегрує конструктори, гетери та сетери у код. Так анотація @Getter згенерує гетери, а анотація @Setter – сетери. За допомогою анотації @NoArgsConstructor можна створити конструктор без параметрів, а з @AllArgsConstructor – конструктор з усіма параметрами. Всі ці анотації можна об’єднати та замінити однією анотацією @Data, що є дуже зручною розробкою та економить багато часу.

Усі класи відповідають предметній області бази даних та описують відповідні таблиці та зв’язки. Так у пакеті entity зберігаються такі класи: Client, EatPeriod, Gender, Meals, NutritionGoal, Product, User, UserRole, Water.

Використовуючи анотацію @Entity, система визначає що цей клас є сутністю (entity bean). Такий клас повинен мати конструктор за замовчуванням (порожній конструктор).

Анотації в Spring дозволяють визначити структуру таблиць бази даних та зв'язки між ними. Для цього використовуються анотації @Table, @Column, @OneToOne, @OneToMany, @ManyToOne та @JoinColumn. Анотація @Table вказує на таблицю, з якою пов'язана ця сутність. Анотація @Column використовується для визначення стовпців таблиці та їх типів даних. Для визначення зв'язків між таблицями використовуються анотації @OneToOne, @OneToMany та @ManyToOne. Анотація @OneToOne вказує на зв'язок один до одного між двома таблицями, а анотації @OneToMany та @ManyToOne – на зв'язок один до багатьох та багато до одного відповідно. Анотація @JoinColumn використовується для визначення стовпця, який використовується для зв'язку між таблицями.

Наступним, не менш важливим, кроком було створення класів-сервісів. Вони є бізнес-логікою роботи застосунку та є проміжним прошарком між класами-контролерами та класами-репозиторіями. Тобто вони приймають запити від контролерів, та роблять свої запити до репозиторіїв.

Кожен сервіс відповідає за певну логіку, що об’єднує певні задачі. Наприклад, логіку обробки автентифікації та авторизації клієнта ми можемо винести в окремий сервіс, який буде в собі містити всі необхідні методи, які використовуються під час роботи застосунку.

Анотація @Service помічає клас як сервіс. При розробці продукту було створено такі сервіси: ClientService, EatPeriodService, GenderService, MealsService, NutritionGoalService, ProductService, UserDetailsService, UserService, UserRoleService, WaterService. Усі вони розділені за деякою логікою та кожен відповідає за свою предметну область.

Методи цих класів використовують в собі класи-репозиторії, які дозволяють взаємодіяти з базою даних. Це реалізовано за допомогою механізму зв’язування Spring, а саме анотацію – @Autowired.

Наступний прошарок – класи-репозиторії. Вони призначені для взаємодії з базою даних та надають можливість автоматизувати цю взаємодію. Такі класи визначаються анотацією @Repository. Ці класи відловлюють SQL помилки, оскільки вони напряму комунікують з базою даних.

За допомогою можливостей Spring Data JPA, було створено методи для взаємодії з базою даних. Потужність JPA дозволяє інтерпретувати в SQL запити ім’я методів. Також за допомогою анотації @Query можна виконувати більш складні SQL запити. Також ці класи успадковують клас JpaRepository, що додає за замовченням всі базові запити до бази даних.

При розробці було створено такі класи-репозиторії: ClientRepository, EatPeriodRepository, GenderRepository, MealsRepository, NutritionGoalRepository, ProductRepository, UserRepository, UserRoleRepository, WaterRepository.

Наступним прошарком були класи-контролери. Вони визначені анотацією @Controller. Головною особливістю є те, що дані контролери перехоплюють та оброблюють HTTP запити. Було вказано тип запиту за допомогою анотацій: @GetMapping, @PostMapping, @DeleteMapping, @PatchMapping, що відповідають методам класу. Анотації @RestController та @RequestMapping вказують який URL буде оброблювати даний клас.

У пакеті controller зберігаються такі класи: AdminController, ClientController, CustomErrorController, InfoController, LocaleController, SignController, UserController, ProductController.

Методи цих класів використовують в собі класи-сервіси, які реалізують основну бізнес-логіку. Це реалізовано за допомогою механізму зв’язування Spring, а саме анотацію – @Autowired.

4.4.2 Розробка клієнтської частини

Клієнтська частина застосунку – це те, що бачить користувач на екрані свого пристрою. Вона відповідає за інтерфейс та взаємодію з користувачем. Клієнтська частина застосунку написана мовами HTML, CSS та JavaScript.

Клієнтська частина веб-програми написана мовою JavaScript, яка виконується у браузері користувача. Вона відповідає за взаємодію з користувачем та надсилання запитів на серверну частину. Також вона використовує HTML і CSS для створення інтерфейсу користувача. JavaScript відповідає за динамічну зміну вмісту сторінки та обробку дій користувача, а HTML і CSS визначають зовнішній вигляд і структуру елементів на сторінці. Разом вони утворюють основу застосунку.

Клієнтська частина може також використовувати різні плагіни та бібліотеки для поліпшення інтерфейсу користувача. Застосунок використовує Bootstrap для створення красивих і адаптивних форм та компонентів.

4.5 Розгортання та перенесення застосунку на віддалений сервер

Для розміщення застосунку на віддаленому сервері було обрано платформу Heroku. Це хмарна платформа, яка дозволяє розробникам розгортати та масштабувати свої застосунки.

Для розгортання програми у середовищі Heroku необхідно створити обліковий запис, завантажити код програми на сервер, налаштувати базу даних та інші параметри оточення. Heroku також надає інструменти для моніторингу та управління застосунком.

Перенесення програми на Heroku було виконано завдяки інтеграції з Git і автоматичному масштабуванню ресурсів залежно від навантаження. Також код програми був збережений у системі для керування та контролю версіями GitHub.

4.6 Тестування застосунку

Даний розділ описує процес перевірки та оцінки функціональності, продуктивності та безпеки застосунку після завершення його розробки. Воно включає різні етапи, такі як модульне тестування, інтеграційне тестування, функціональне тестування, тестування продуктивності і тестування безпеки.

Модульне тестування – це тестування кожного окремого модуля програми, щоб переконатися у його правильній роботі. Зазвичай використовуються автоматичні тести, які перевіряють код на відповідність певним критеріям.

Інтеграційне тестування – це тестування взаємодії між різними модулями програми. Воно дозволяє виявити помилки, які можуть виникнути під час інтеграції різних компонентів.

Функціональне тестування – це тестування функціональності застосунку відповідно до його вимог. Воно дозволяє переконатися в тому, що програма виконує свої функції правильно і відповідає очікуванням користувачів.

Тестування навантаженням – це тестування швидкості та надійності програми при різних навантаженнях. Воно дозволяє виявити вузькі місця у застосунку та оптимізувати його для більш ефективної роботи.

Тестування безпеки – це тестування програми на наявність вразливостей, які можуть призвести до порушення безпеки даних

користувачів. Воно дозволяє виявити вразливості та усунути їх до того, як застосунок буде відкритий до користування.

Всі ці етапи тестування можуть бути автоматизовані за допомогою спеціальних інструментів, таких як Selenium, JMeter, Postman та інші. Це дозволяє прискорити процес тестування та знизити ймовірність помилок. Також важливо проводити тестування на реальних пристроях та браузерах, щоб переконатися у правильній роботі програми на різних платформах та пристроях.

4.6.1 Модульне тестування

Модульне тестування є важливою частиною процесу розробки застосунків та допомагає швидко виявляти та виправляти помилки в коді, зменшує час та витрати на налагодження програми та підвищує його якість.

Під час розробки застосунку було проведено модульне тестування. Були протестовані окремі компоненти (модулі) з метою виявлення та усунення помилок та дефектів. Кожен модуль тестувався окремо, незалежно від інших компонентів програми.

Для проведення модульного тестування використовувалися спеціальні фреймворки та інструменти, такі як JUnit та Mockito. Інтегроване середовище тестування JUnit є засобом для модульного тестування клієнтських Java-застосунків [15].

4.6.2 Інтеграційне тестування

Інтеграційне тестування є важливою частиною процесу розробки застосунків і допомагає створити більш надійне і стабільне програмне забезпечення. Це також дозволяє переконатися в правильній роботі застосунку в умовах реального використання. На відміну від модульного тестування, інтеграційне тестування проводиться на рівні системи в цілому. Воно дозволяє перевірити, що всі компоненти програми працюють правильно разом, забезпечуючи правильну передачу даних і виконання функцій.

В ході інтеграційного тестування проводяться різні типи тестів, такі як тестування сумісності, тестування інтерфейсів, тестування продуктивності та ін.

Для проведення інтеграційного тестування даного застосунку використовувався спеціальний інструмент Selenium WebDriver. За результатами тестування помітних недоліків чи помилок не було знайдено. Відображення застосунку в кожному браузері було коректне, тому можна зробити висновок, що даний застосунок сумісний з сучасними браузерами і працює відповідно до вимог.

4.6.3 Тестування навантаженням

Тестування навантаженням застосунку є процесом, який дозволяє визначити, як система поводиться при певних навантаженнях. Цей вид тестування вимірює реакцію застосунку на різні рівні навантаження, такі як кількість користувачів, запитів на сервер, обсяги даних тощо.

Основна мета тестування навантаженням полягає у забезпеченні високої продуктивності і доступності застосунку під час максимального

навантаження. Це може бути особливо важливо для вебдодатків, які отримують велику кількість відвідувачів або обробляють великий обсяг даних.

Після проведення тестування навантаженням можуть бути отримані різні результати, такі як час відповіді сервера, кількість запитів на сервер, обсяги даних, які передаються тощо. Ці результати можуть бути використані для виявлення проблем з продуктивністю веб-застосунку та для покращення його роботи під час максимального навантаження.

Тестування навантаженням проводилося шляхом створення симульованого навантаження на вебзастосунок. Це було зроблено за допомогою Apache JMeter. Цей інструмент дозволяє створити велику кількість запитів на сервер, щоб перевірити, як застосунок реагує на таке навантаження.

РОЗДІЛ 5.

ОГЛЯД ЗАСТОСУНКУ

Коли користувач вперше потрапляє до застосунку «EatingWell», він бачить головну сторінку (див. рис. 5.1), де перед ним відкривається багато функціоналу. Перш за все, користувач може увійти або зареєструватися. Також він може перейти на сторінку з рецептами, вибрати зручну йому мову, або передивитися різну цікаву додаткову інформацію.

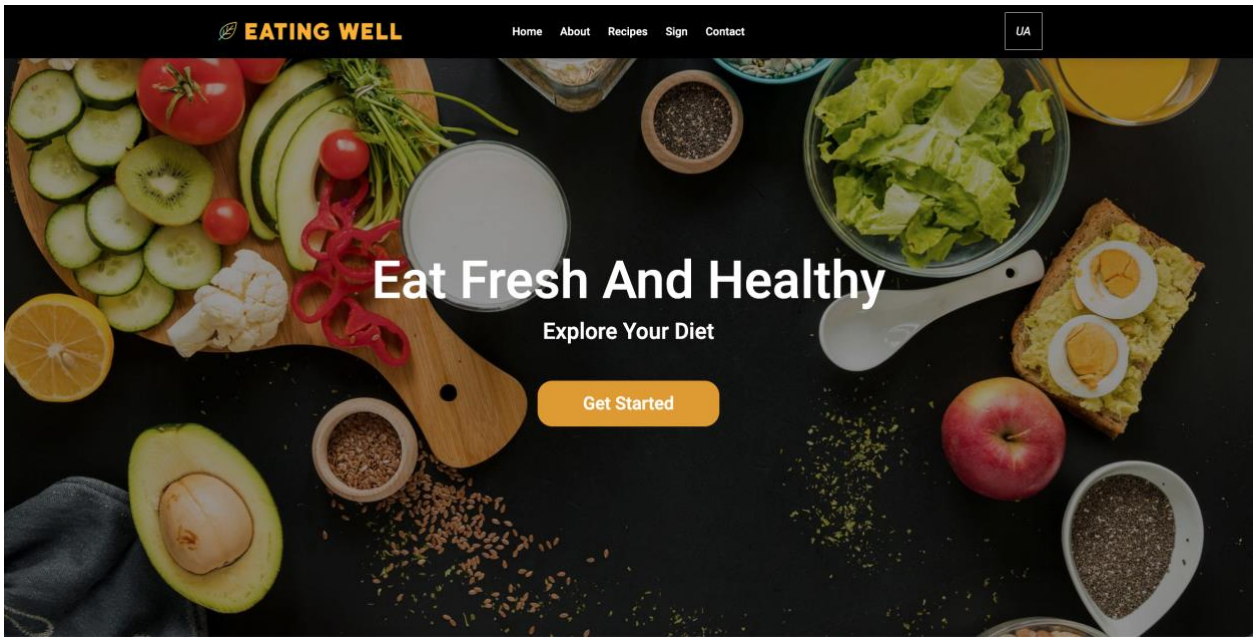

Рисунок 5.1 – Головна сторінка застосунку

Далі слідує процес авторизації або автентифікації (див. рис. 5.2). При реєстрації користувачу слід ввести свою електронну пошту та пароль. Також треба повторно ввести пароль для перевірки на сумісність. Після введення інформації вона перевіряється на коректність: електронна адреса повинна мати коректну структуру, а пароль відповідати вимогам (довжина 7-20 символів, обов’язково містити велику літеру та цифру). Якщо інформація є

некоректною, то користувач отримає червоне сповіщення які саме дані є хибними. Обробка продовжиться тільки якщо усі дані введені правильно.

Теж саме відбувається при спробі увійти до застосунку. Користувачу буде запропоновано ввести електронну пошту і пароль. Якщо вони дійсні, то користувач потрапить до свого кабінету.

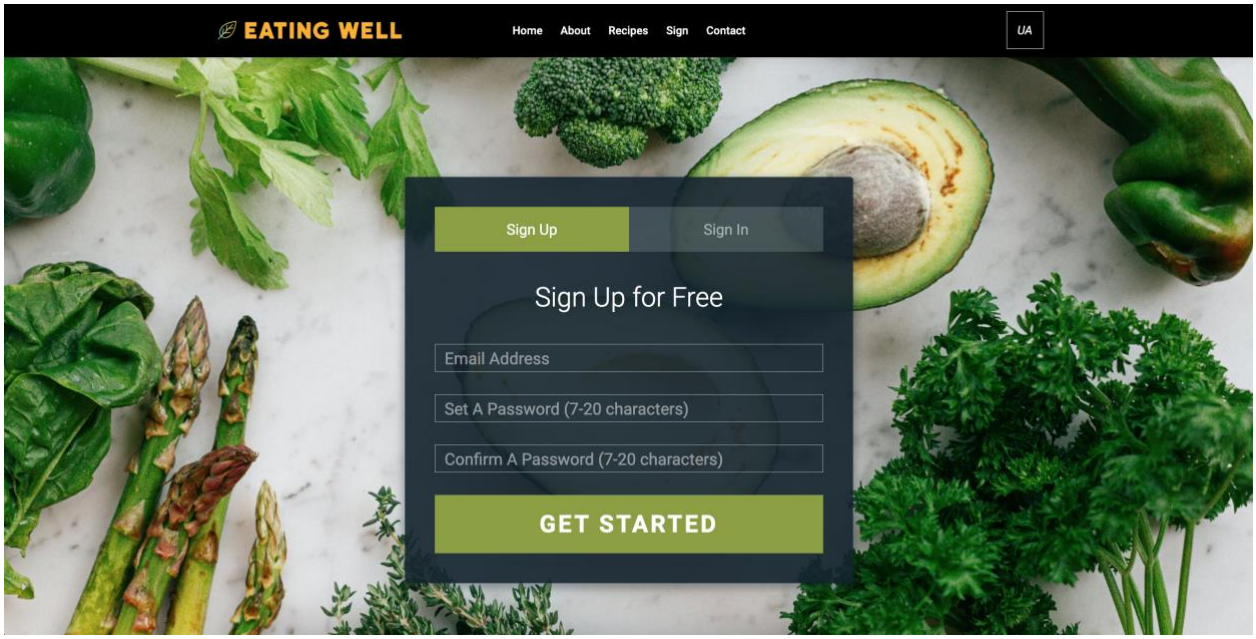

Рисунок 5.2 – Сторінка реєстрації користувача

Після реєстрації користувачу буде запропоновано ввести персональну інформацію про свої параметри, спосіб життя та мету використання застосунку (див. рис. 5.3). На основі цих даних буде побудовано подальші рекомендації щодо споживання їжі та калорійності. Це дозволить користувачу більш продуктивно використовувати застосунок та досягати своєї цілі. Проте це є необов’язково, оскільки система може встановити ці значення за замовчуванням (будуть визначені за середньостатистичними параметрами).

Ця інформація може бути додана або змінена у будь-який час користувачем через особистий кабінет.

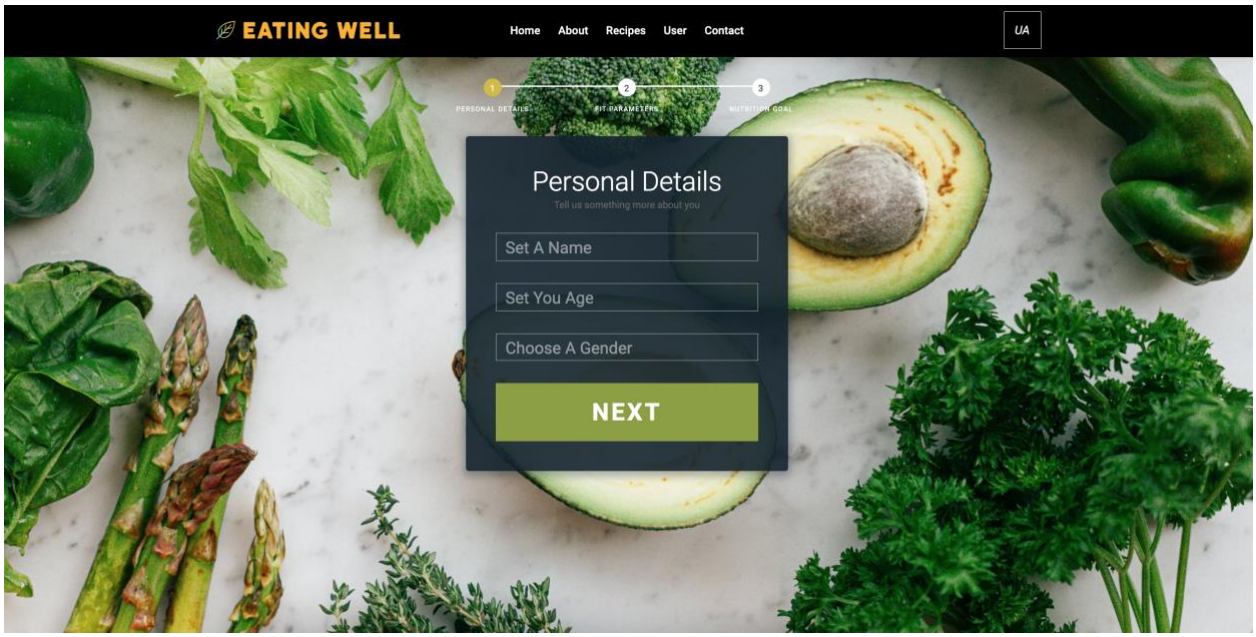

Рисунок 5.3 – Сторінка для введення персональної інформації

Після успішної авторизації користувачу відкривається увесь основний функціонал. Оскільки основна мета застосунку – ведення щоденника харчування, то користувач, використовуючи сторінку Денний раціон (див. рис. 5.4), може заповнювати споживання їжі. Для цього треба обрати продукт або страву зі списку, вказати кількість в грамах, а також вибрати час споживання (сніданок, обід, вечеря чи перекус). Усі поля є обов’язковими для введення, кількість продукту не може бути від’ємною. Ця інформація буде додана до загальної таблиці.

Якщо ця інформація додана, то її можна редагувати, або видалити на сторінці, використавши необхідні інструменти.

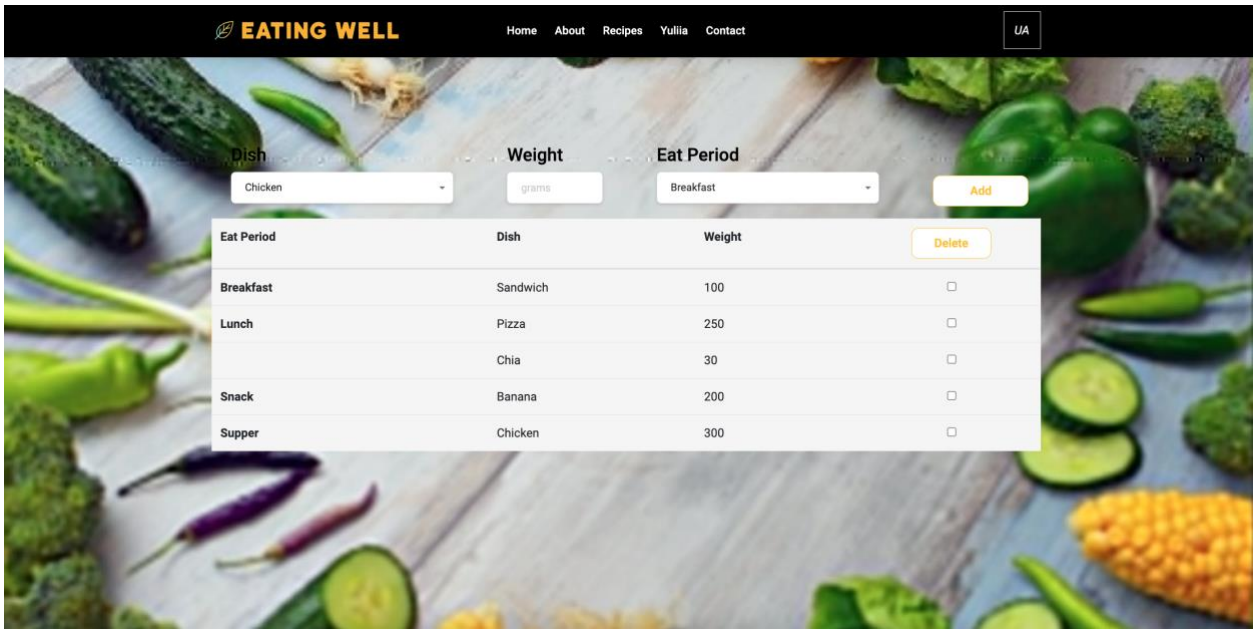

Рисунок 5.4 – Сторінка для заповнення щоденного раціону

Якщо користувач не знайшов продукту або страву, то він може додати власну одиницю (див. рис. 5.5). У спливаючому вікні йому слід надати інформацію про продукт або страву: ім’я, кількість калорій, білків, жирів, вуглеводів на 100 грам. Уся інформація у процесі введення повинна бути коректною для успішного завантаження до системи. Цей продукт або страва буде доступною тільки для даного користувача. Проте існує інша можливість у адміністратора – додавання продукту або страви для усіх користувачів.

Після додавання інформація про продукт або страву може бути змінена або видалена.

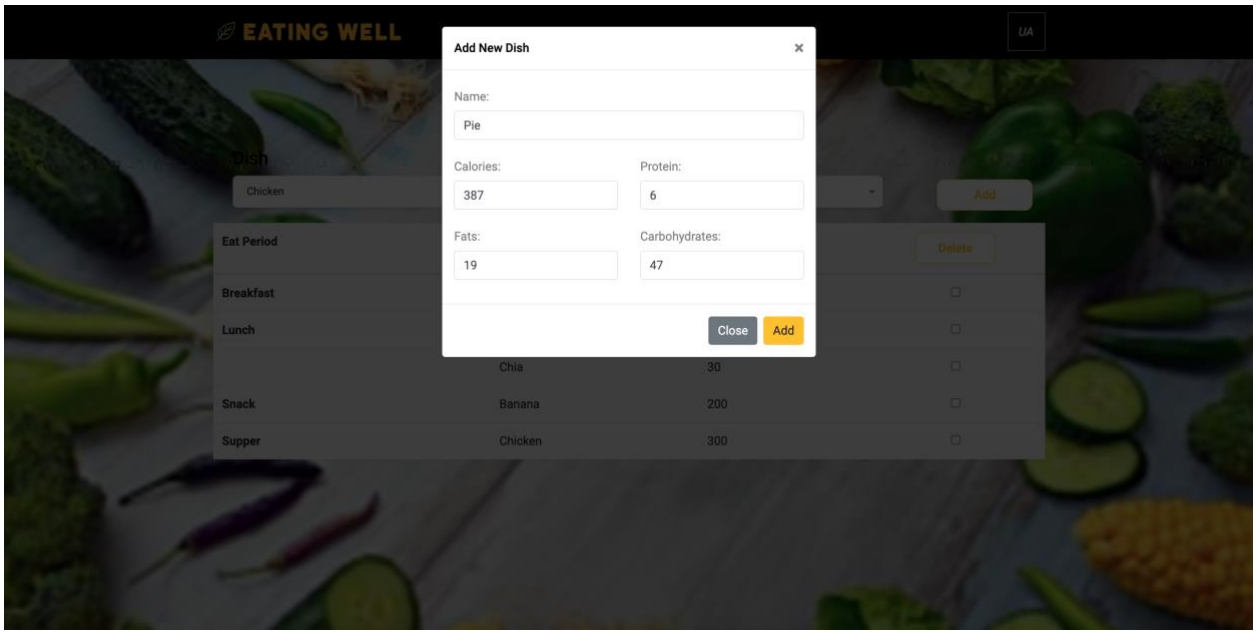

Рисунок 5.5 – Сторінка для додавання нового продукту чи страви

Будь-який зареєстрований користувач може слідкувати за своїм водним балансом. На сторінці вживання води (див. рис. 5.6), він може додати випиту склянку води. Відразу він може побачити як зростає кількість його випитої води, процентне співвідношення з денною нормою, а також кількість, яку ще треба спожити.

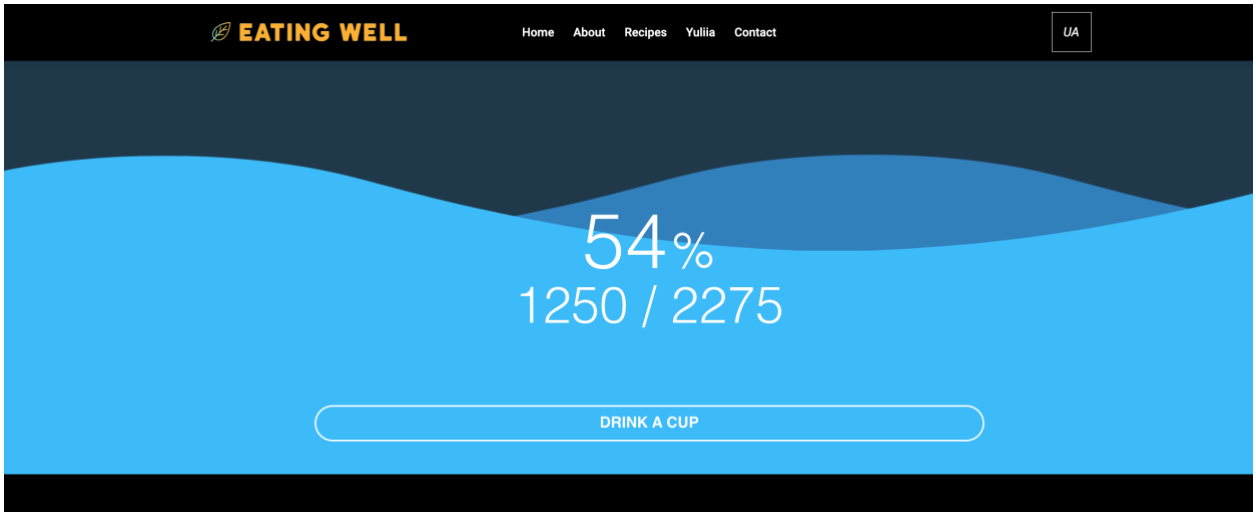

Рисунок 5.6 – Сторінка для введення інформації про випиту воду

На основі заповнення даних про раціон користувача будуються графіки та діаграми для зручного користування та аналізу прогресу за день або тиждень. Існує декілька різних діаграм (див. рис. 5.7): вони інформують користувача про спожиту кількість калорій, жирів, білків та вуглеводів. А також наявний графік, що зображує споживання калорій за останній тиждень.

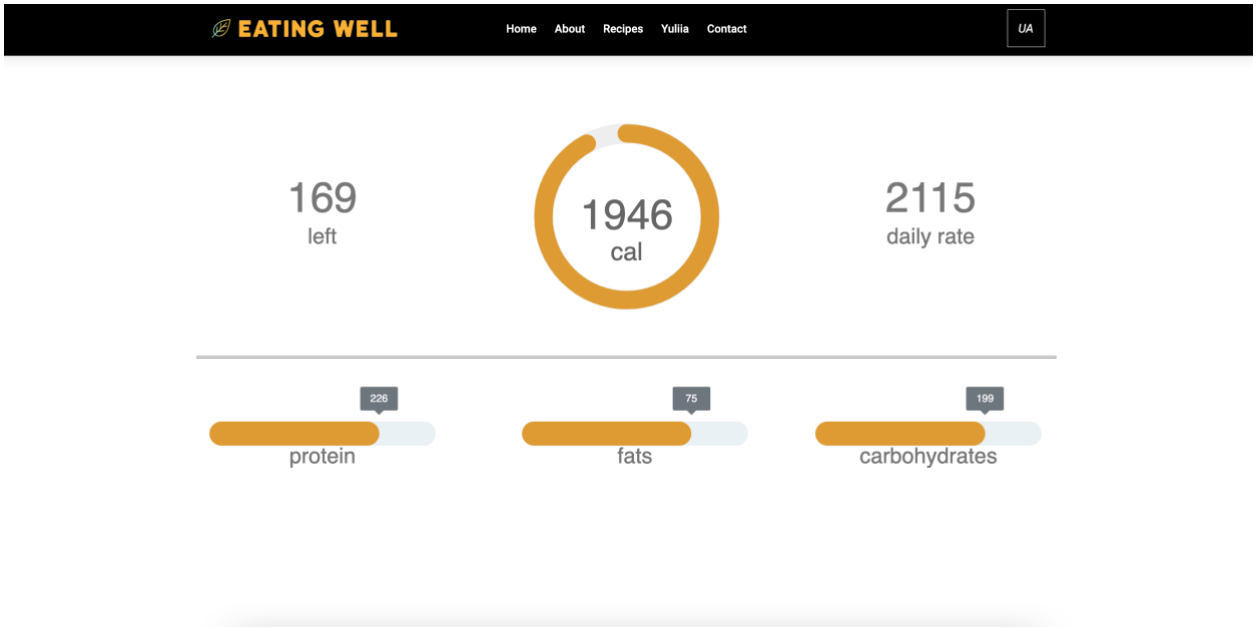

Рисунок 5.7 – Аналіз раціону користувача

Для будь-якого користувача також доступний список з різноманітними рецептами (див. рис. 5.8). Він може знаходити різні цікаві йому рецепти. Також існує фільтр, щоб враховувати різні алергени при пошуку. Користувач може обрати декілька параметрів для фільтрації. Це дозволяє відкинути рецепти, що є небезпечними для користувача.

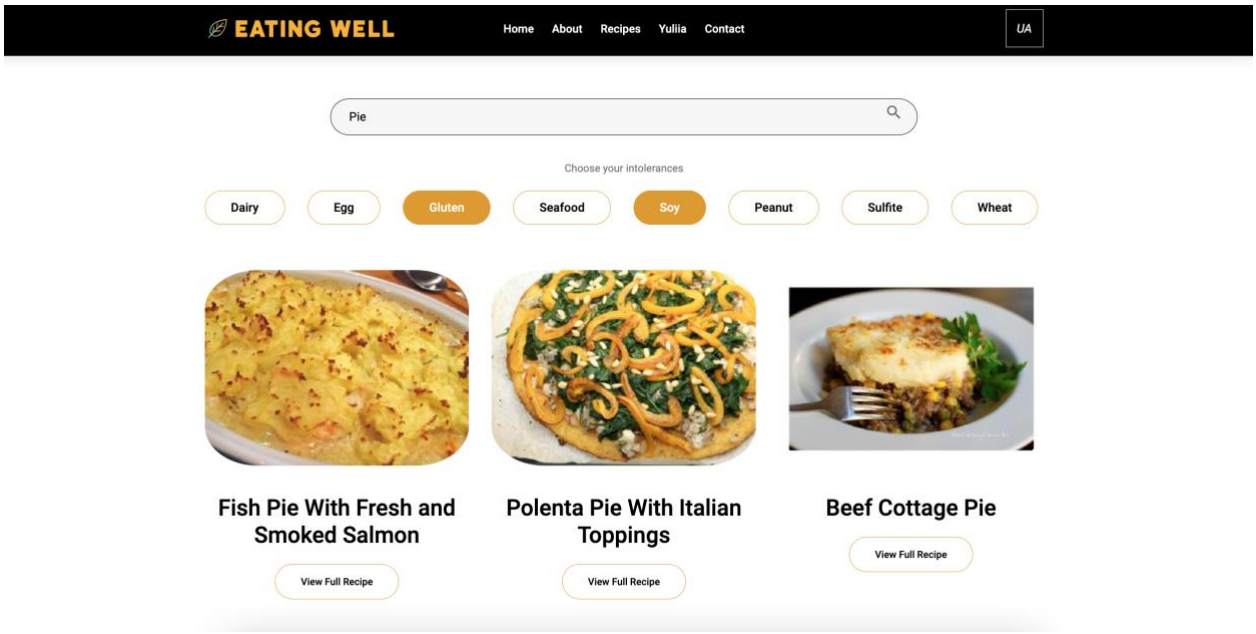

Рисунок 5.8 – Сторінка з рецептами

Також існує декілька сторінок з загальною інформацією про систему «EatingWell» і контактною інформацією, куди користувач зможе звернутися, якщо виникнуть якісь проблеми або будуть побажання (див. рис. 5.9).

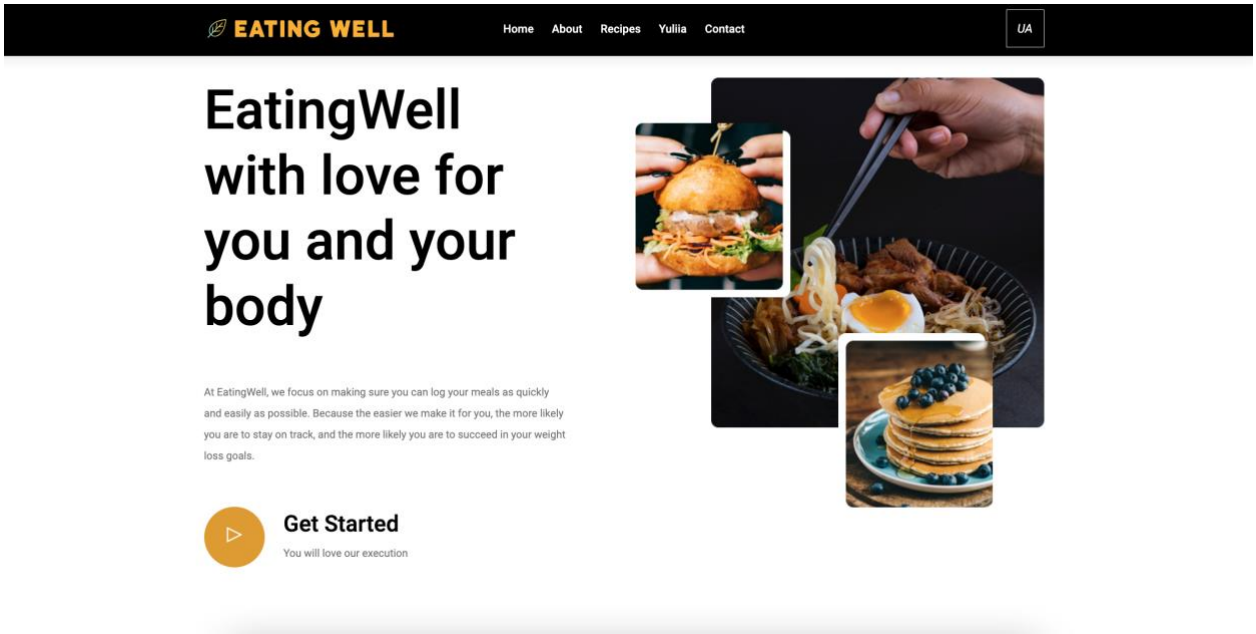

Рисунок 5.9 – Сторінка з загальною інформацією про систем

ВИСНОВКИ

В даній роботі було досліджено основні принципи та засоби розробки застосунків. Визначено на основі аналізу предметної області та існуючих рішень основні вимоги до архітектури, функціоналу та дизайну застосунку. І як результат, було розроблене технічне завдання до системи. Також проаналізовано використання різних технологій для проєктування та розробки програмного продукту.

За час виконання кваліфікаційної роботи на здобуття рівня магістра було досягнуто поставлені цілі: досліджено системи, призначені для підтримки здоров'я, проведено аналіз джерел і матеріалів принципів розробки застосунку для підтримки і контролю здорового способу життя, досліджено використання різних підходів та технологій для проєктування та реалізації застосунків, розроблено технічне завдання до програмного продукту, розроблено інтерфейс, дизайн та бізнес-логіку застосунку, реалізовано клієнт-серверний застосунок «EatingWell», що надає інструменти та ресурси для контролю, аналізу та покращення свого здоров'я, враховуючи переваги на недоліки уже готових рішень. Отримані результати підтверджують новизну даної роботи.

Для розробки клієнтської частини було обрано HTML та CSS – для побудови та відображення графічного наповнення сторінок, JavaScript для динамічної зміни вмісту сторінки та обробку дій користувача. У свою чергу для серверної частини була обрана Java, як мова програмування для розробки бізнес-логіки, разом з фреймворком Spring. У результаті програма була протестована, використовуючи різні вхідні дані.

У майбутньому основна увага буде зосереджена на вдосконаленні та доопрацюванні функціоналу застосунку. Також можливе використання даної розробки в контексті інших сервісів, імплементуючи туди дану логіку.

На момент написання кваліфікаційної роботи код застосунку розміщений у вебсервісі для спільної розробки програмного забезпечення GitHub і відкритий до вдосконалення та розширення для сторонніх розробників. Застосунок також розміщено на віддаленому сервері Heroku.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Виноградов П. А. Фізична культура і здоровий образ життя / П. А. Виноградов. – М. : Думка, 2001. – 287 с.
2. М. П. Горобей, О. В. Осадчий. Загальна теорія здоров'я / М. П. Горобей, О. В. Осадчий. – Чернігів : ЧНТУ, 2017. – 210 с.
3. JetBrains [Електронний ресурс] – Режим доступу до ресурсу: https://www.jetbrains.com.
4. Крокфорд Дуглас. JavaScript. Сильні сторони, 2016. — 176 с.
5. NutraCheck [Електронний ресурс] – Режим доступу до ресурсу: https://www.nutracheck.co.uk/Home.
6. My Macros + Diet and Calories [Електронний ресурс] – Режим доступу до ресурсу: https://getmymacros.com/.
7. My Fitness Pal [Електронний ресурс] – Режим доступу до ресурсу: https://www.myfitnesspal.com.
8. Nutritionix Track [Електронний ресурс] – Режим доступу до ресурсу: https://www.nutritionix.com/app.
9. Bloch J. Effective Java, Third Edition / Josh Bloch., 2017. – 416 с.
10. Craig Walls. Spring in Action / Craig Walls., 2018. – 520 с.
11. PostgreSQL [Електронний ресурс]: Вікіпедія. Вільна енциклопедія. – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/PostgreSQL.
12. Elisabeth Robson, Eric Freeman. Head First HTML with CSS & XHTML., 2010. — 656 с.
13. JavaScript Official Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://devdocs.io/javascript/.
14. Spring Framework [Електронний ресурс] – Режим доступу до ресурсу: https://spring.io/.
15. Unit тестування з JUnit. – Режим доступу до ресурсу: http://devcolibri.com/864.

ДОДАТКИ

Додаток А. Use-Case діаграма системи

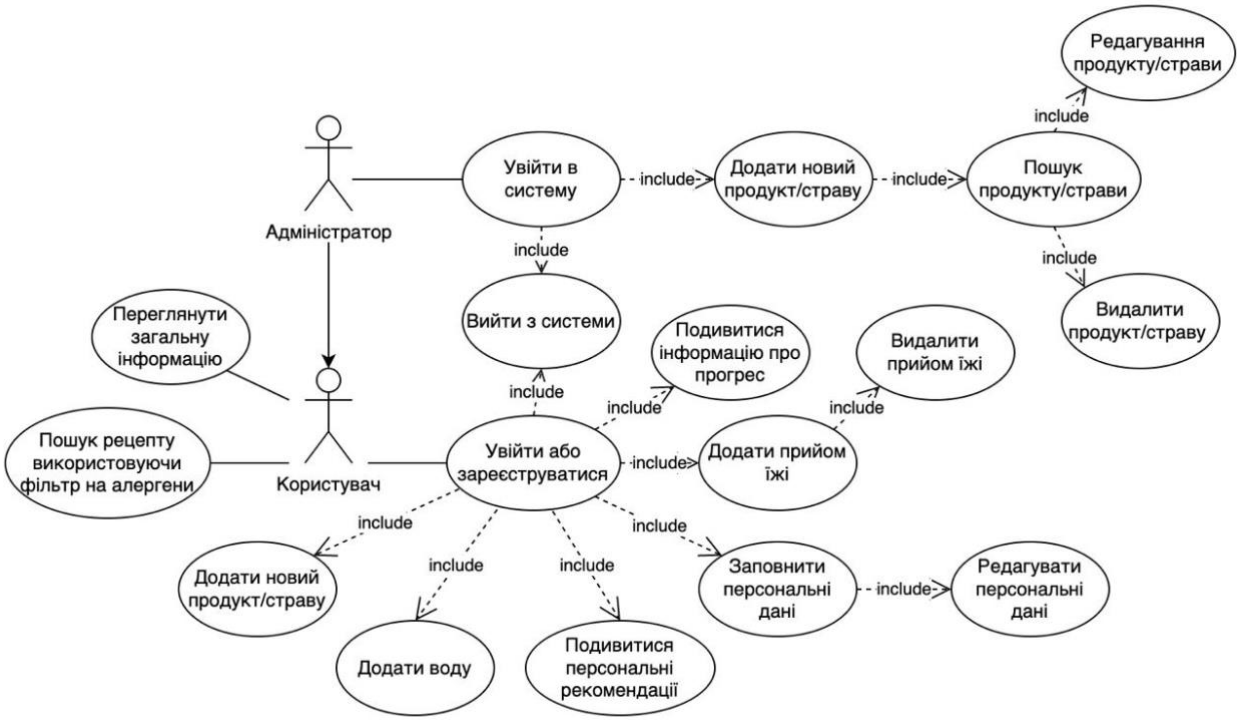

Рисунок А.1 – Use-case діаграма

Додаток Б. Діаграма бази даних системи

Рисунок Б.1 – Діаграма бази даних

Додаток В. Структура програми системи

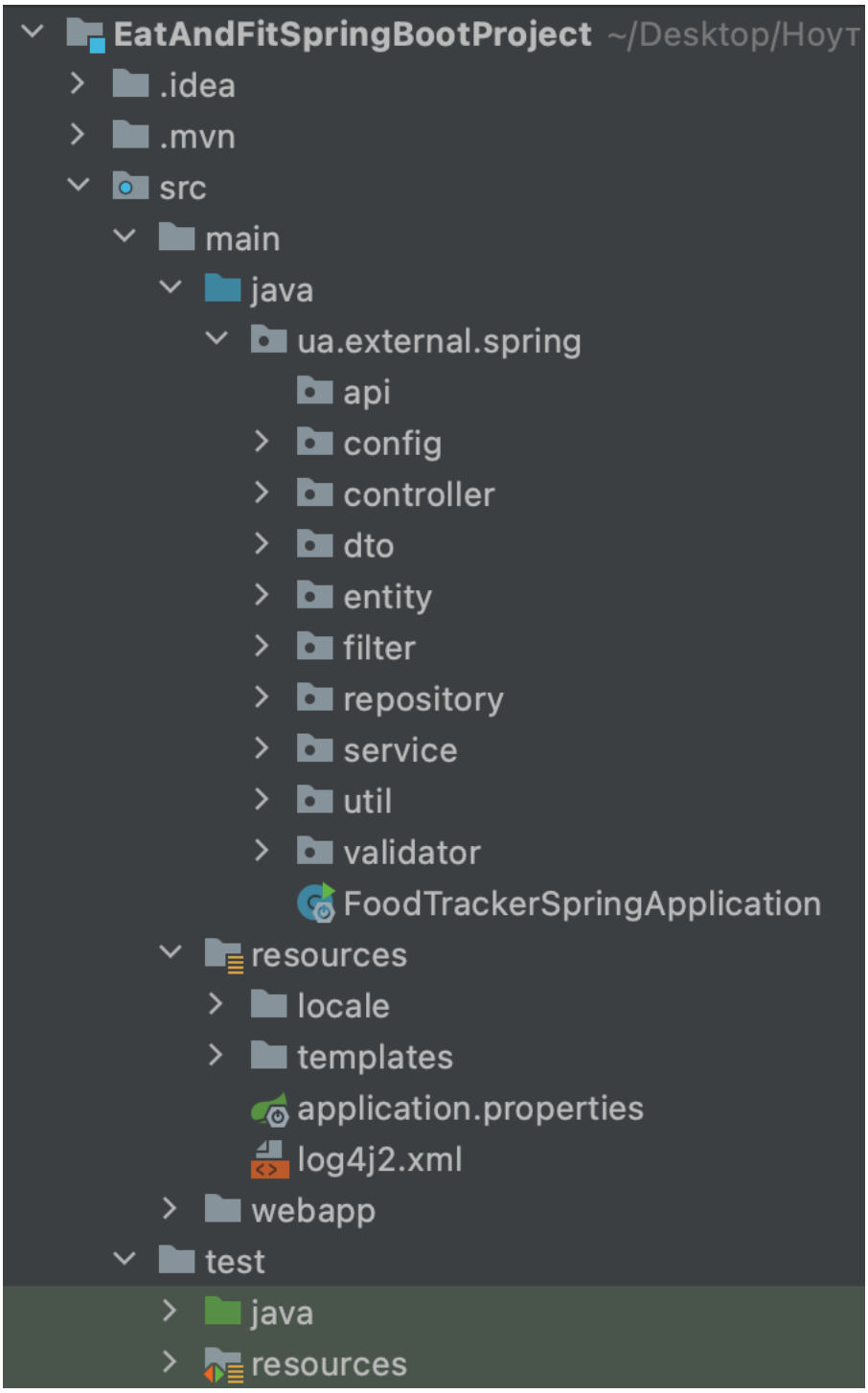

Рисунок В.1 – Структура програми