Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп’ютерних наук

Кафедра програмної інженерії

КУРСОВА РОБОТА
ПОЯСНЮВАЛЬНА ЗАПИСКА
з навчальної дисципліни «Архітектура програмного забезпечення»
Тема роботи: Програмна система трекінгу балансу мікроелементів та стану здоров'я людини, що займається спортом

Студент гр. ПЗПІ-18-1 _____ Жиденко І.В.
(підпис)

Керівник роботи _____ доц. Лещинський В.О.
(підпис)

Роботу захищено «__»_____2021 р.
з оцінкою _____

Комісія: _____ доц. Лещинський В.О.
(підпис)

_____ доц. Лещинська І.О.
(підпис)

_____ ст.викл. Сокорчук І.П.
(підпис)

Харків
2021 р.

Харківський національний університет радіоелектроніки

Факультет *комп'ютерних наук* Кафедра *програмної інженерії*

Спеціальність *121 – Інженерія програмного забезпечення*

Курс 3 Семестр 6

Навчальна дисципліна *Архітектура програмного забезпечення*

ЗАВДАННЯ
НА КУРСОВУ РОБОТУ СТУДЕНТОВІ

Жиденку Ігрою Валерійовичу

1. Тема роботи: *«Програмна система трекінгу балансу мікроелементів та стану здоров'я людини, що займається спортом»*
2. Термін узгодження завдання курсової роботи: «01» березня 2021 р.
3. Термін здачі студентом курсової роботи: «27» червня 2021 р.
4. Вихідні дані до проекту (роботи): *У програмній системі передбачити: складання раціонів; підрахунок калорійності та вмісту мікроелементів для визначеної маси продукту; керування пристроями користувача; відображення поточного стану здоров'я та динаміки його зміни за обраний проміжок часу; можливість вказати спожиті продукти під час прийому їжі; появу повідомлень про погіршення стану здоров'я. Використовувати Windows 10, PyCharm 2021.2, PostgreSQL 14.*
5. Зміст пояснювальної записки (перелік питань, що належить розробити): *вступ, аналіз предметної області, постановка задачі, проектування програмного проекту, структура бази даних, кодування проекту, опис розробленої програмної системи, висновки, перелік посилань, додатки.*
6. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень): *UML діаграма прецедентів, ER-діаграма, UML діаграма компонент веб-застосунку, UML діаграма пакетів веб-застосунку, UML діаграма розгортання.*

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів курсової роботи	Термін виконання етапів роботи	Примітка
1	Функціональна специфікація програмного проекту	01.03.2021	виконано
2	Проектування програмного проекту	15.03.2021	виконано
3	Кодування програмного проекту	30.04.2021	виконано
4	Оформлення пояснювальної записки	07.06.2021	виконано
5	Захист курсової роботи	27.06.2021	виконано

Дата видачі завдання «01» березня 2021 р.

Керівник _____ доц. Лещинський В.О.

(підпис)

Завдання прийняв до виконання

ст.гр. ПЗПІ-18-1 _____ Жиденко І.В.

(підпис)

РЕФЕРАТ

Пояснювальна записка до курсової роботи: 55 с., 13 рис., 2 додатки, 7 джерел.

СТАН ЗДОРОВ’Я, ПОКАЗНИК ЗДОРОВ’Я, МЕДИЧНА ДОПОМОГА, ПУЛЬС, САТУРАЦІЯ, МІКРОЕЛЕМЕНТ, ФІТНЕС-БРАСЛЕТ, РАЦІОН, КАЛОРІЙНІСТЬ.

Об’єкт дослідження являє собою галузь охорони здоров’я, насамперед питання стосовно відслідковування поточного стану здоров’я, визначення його показників та сповіщення користувача у разі фіксації значення, яке більше або менше за норму.

Мата даної роботи – створення програмної системи, що дозволить на основі переданих показників здоров’я із пристроїв користувача робити висновок про його поточний стан здоров’я, визначати критичні показники, про що повідомляти його у разі потреби; відображати статистичні дані у візуалізованому вигляді стосовно зафіксованих показників за обраний користувачем проміжок часу; керувати раціонами, вказуючи продукти та їх вміст при кожному прийомі їжі; вказувати спожиті під час прийому їжі продукти та їх відкориговану масу.

Методи розробки back-end складової системи засновуються на застосуванні Python-фреймворку Django, front-end складової – JavaScript-фреймворку React, мобільного застосунку – мови програмування Kotlin, розумного пристрою – апаратної обчислювальної платформи Arduino Mega, запрограмованої в Arduino IDE із використанням мови програмування C. У якості системи керування базами даних обрано PostgreSQL.

Результатом роботи є спроектована та розроблена програмна система відстеження балансу мікроелементів та стану здоров'я людини, яка займається спортом, до її складу входить як серверна, так і клієнтська частини веб-застосунку, мобільний застосунок, Smart Device.

ЗМІСТ

ВСТУП

Немає жодного сумніву, що кожна свідома особа піклується про стан свого здоров’я, адже це є запорукою довгого та щасливого життя. За останнє століття медицина зробила величезний крок вперед, подовживши середню тривалість життя людей не на одне десятиліття [1]. Проте навіть сьогодні значна частина жителів нашої планети помирає від хвороб, які можна було б вилікувати, вчасно звернувши увагу на ті показники здоров’я, що починають відхилятися від норми. Проблемою є те, що зазвичай люди не мають можливості відслідковувати такі показники через відсутність відповідних пристроїв, відвідувати лікаря досить часто бракує часу через досить бурхливий сьогоднішній темп життя [2].

Метою даної роботи є розробка програмної системи, розумні пристрої у складі якої дозволять регулярно отримувати значення необхідних показників здоров’я, а обчислювальні потужності забезпечать детальний їх аналіз та робитимуть висновки, чи є відхилення від норми, та у разі отримання критичних значень негайно сповістять власника пристрою про це, аби зберегти його життя і здоров’я, вчасно прореагувавши на негативні тенденції. Також розроблений програмний продукт має відстежувати динаміку зміни показників здоров’я, візуалізувавши її у вигляді гістограми чи графіку за вибраний користувачем проміжок часу. Велика увага приділяється здоровому харчуванню, тому клієнти програмної системи зможуть складати свої раціони відповідно до калорійності страв та вмісту в них важливих для життя людини мікроелементів.

При виконанні курсової роботи необхідно вирішити наступні завдання: дослідити предметну область відслідковування стану здоров’я, виявити проблеми, що є для неї актуальними, розробити стратегії, які б дозволили їх усунути, обрати найоптимальнішу із них, визначитися із засобами, необхідними для втілення задуманого у життя, виконати безпосереднє кодування програмного продукту.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Аналіз предметної галузі

1.1.1 Аналіз ринку трекінгу стану здоров’я

Тема контролю стану здоров’я є дуже актуальною для людей з слабким здоров’я та для тих осіб, що займаються спортом. Зі зростанням популярності спортзалів, фітнес-залів та інших подібних закладів, у людей, що займаються спортом, з’явилася потреба більш детально та вчасно слідкувати за станом свого здоров’я [3].

Така сама проблема виникає не тільки у пересічних користувачів, а й у самих спорт- та фітнес-залів. Всім вигідно, коли працівники цих залів добре слідкують за своїм здоров’ям та, в разі чого, можуть швидко зреагувати на погіршення його стану [4].

Актуальність роботи полягає у необхідності розробки такої системи, що дозволить за відносно короткий час створити детальний опис стану здоров’я та дати рекомендації щодо його поліпшення у разі потреби.

Необхідність даного продукту пов’язана з тим, що на сьогоднішній день користувачу необхідно витратити багато сил та часу для того, щоб дізнатися детально про свій стан здоров’я.

Таким чином, основними завданнями курсової роботи є поетапна розробка зручних для користувача сервісів для перегляду стану свого здоров’я та рекомендацій щодо його поліпшення.

1.1.2 Аналіз існуючих рішень

На сьогоднішній день на ринку програмних продуктів, пов’язаних з трекінгом стану здоров’я, існує певна кількість продуктів. Однак вони надають

користувачу лише загальну інформацію. Перш за все, існує багато програмних систем для перегляду стану здоров’я, але вони показують лише базову інформацію без детальних досліджень. До них можна віднести такі відомі рішення, як HealthTap, ShopWell, Elevate, Fabulous, Health Pal, Remente, Health and Nutrition Guide & Fitness Calculators, Moodpath [5].

Так як розроблювана програмна система також націлена на здорове харчування, то конкурентами є застосунки, які спеціалізуються на відстеженні калорійності та інших показників, пов’язаних із вживанням їжі, проте є дуже вузькоспеціалізованими і не надають жодної інформації про інші показники здоров’я. Серед таких продуктів можна виділити Nutrients, MyFitnessPal, MyNetDiary, MyPlate Calorie Counter, Nutrition Facts, Calorie Counter & Diet Tracker, Protein Tracker, SuperFood [6].

По-перше, для того, щоб отримувати адекватний результат, потрібно проводити аналізи за допомогою смарт-приладів. Інакше, ви будете отримувати інформацію, яка створена на основі або даних, що самі ж ввели, або недостовірну, на основі ненадійних датчиків телефону, фітнес-браслету, тощо.

По-друге, в таких сервісах існує дуже багато реклами, яка вимикається тільки якщо сплачувати чималі кошти щомісячно.

Окрім цього, всі сервіси для контролю стану здоров’я працюють за різними алгоритмами і тому не видають передбачуваний результат користувачу, який сподівається бачити схожі дані в різних додатках та сервісах [7].

Даний проект використовує спеціальні пристрої, які аналізують фізичні показники людини та передають цю інформацію на сервер. Ризиком для даного проекту буде несправність такого приладу або неточність визначення показників через фізичну активність людини або психологічний стан. Це може призвести до того, що користувач буде отримувати не зовсім точну інформацію. Крім того, ризиком можуть стати конкуренти, які випустять оновлення до своїх додатків з метою розширення функціоналу або взагалі реалізують новий продукт з аналогічними можливостями.

1.2 Постановка задачі

1.2.1 Визначення рішення

Система дозволить користувачам отримувати статистику про кількість мікроелементів та стан здоров’я. Також система буде давати рекомендації щодо раціонів підібраних саме під користувача. Усі зареєстровані користувачі матимуть змогу переглядати усю інформацію та отримувати рекомендації. Як тільки основна аудиторія буде зібрана, нові функції будуть додані до системи.

Система буде перенесена на менш потужні пристрої, щоб охопити більше можливих користувачів. Чим більше користувачів ми отримуємо, тим більшої точності ми можемо досягти. Крім того, це дозволить нам збирати світові статистичні дані, щоб додавати до системи нові раціони та дієти. Чим кращі результати ми маємо, тим краще буде працювати з вуст в уста, що дозволить нам залучити більше клієнтів.

Взаємодіяти із системою користувач зможе наступним чином. Придбавши розумний пристрій, необхідно зайти у веб-застосунок, де на відповідній сторінці увести ідентифікатор пристрою, аби зареєструвати його у системі. Після успішного налаштування пристрій надсилатиме виміряні показники здоров’я на сервер, де визначатиметься, чи належать вони допустимому діапазону. У випадку, якщо певне значення виявилося більшим або меншим за норму, користувач, якщо він увійшов до мобільного застосунку із свого акаунту, отримає сповіщення із детальною інформацією про те, на що необхідно звернути увагу.

Ще одним важливим варіантом використання є створення користувачем індивідуальних раціонів, де він зможе для кожного прийому їжі (сніданок, обід, вечеря, перекуси) вказати, які продукти мають бути спожиті та який їх вміст, при цьому застосунок одразу виводитиме інформацію про калорійність такого раціону та вміст мікроелементів.

1.2.2 Основний функціонал системи

Спроектована та розроблена в ході курсової роботи програмна система виконує наступні основні функції:

1. Створення індивідуального профілю користувача та зберігання у ньому даних.

2. Фіксація розумними пристроями показників здоров’я та надсилання відповідних даних до веб-серверу.

3. Керування раціонами (додавання, редагування, видалення), визначаючи продукти для прийомів їжі у межах доби.

4. Внесення даних про прийом їжі у межах раціону. Хоча раціон і створюється, передбачаючи певний вміст кожного із продуктів, ці дані можуть дещо різнитися від дійсних, тому дозволяється їх змінювати. Проте, якщо певного продукту у раціоні немає для поточного прийому їжі, необхідно створити новий раціон, який би повною мірою відображав наявність усіх компонентів обіду.

5. Ведення статистики поточного стану здоров’я, визначаючи статус відповідності окремих показників нормі.

6. Відстеження динаміки зміни показників здоров’я із часом.

7. Надсилання повідомлення користувачеві про проблеми із здоров’ям у разі потреби.

1.2.3 Потреби, що задовольняє програмний продукт

Програмна система трекінгу балансу мікроелементів та стану здоров'я людини, що займається спортом, покликана задовольнити такі потреби потенційних клієнтів-спортсменів:

– визначення показників здоров’я, які наразі відхиляються від норми, аби попередити захворювання чи запобігти нещасному випадку;

– відсутність необхідності витрачати час і гроші на похід до лікаря, аби дізнатися поточний стан свого здоров’я;

– автоматизоване отримання сповіщень одразу ж, як тільки було зафіксоване відхилення показника від норми;

– зручне ведення власних раціонів із відображенням супутньої інформації про кожен продукт та прийом їжі в цілому.

1.2.4 Бізнес-цілі та критерії успіху

Дана програмна система переслідує наступні бізнес-цілі:

– зекономити клієнту час та зробити процес контролю здоров’я більш комфортним.;

– зменшити шанси виникнення нестачі мікроелементів в організмі.

Програмний продукт вважатиметься успішно реалізованим за умови, що:

– буде отримано 5000 завантажень за перший рік роботи системи;

– буде отримано 50000 активних користувачів у перший рік;

– продукт дійде до точки беззбитковості через 12 місяців;

– система набрере принаймні 60000 завантажень протягом перших 12 місяців.

2 ФОРМУВАННЯ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

Спроектований програмний продукт має складатися із таких частин, як серверна частина, клієнтський веб-застосунок, мобільний застосунок, розумний пристрій. До кожної частини було висунуто низку функціональних вимог, які вона повинна задовольнити.

Серверна частина:

– виконання процесів авторизації та реєстрації користувачів;

– забезпечення аутентифікації за допомогою JWT-токенів;

– автоматизація процесу оновлення SSL-сертифікатів для забезпечення безпечної передачі даних із використанням алгоритмів шифрування у мажах протоколу прикладного рівня HTTPS;

– надання кінцевих точок, що дозволяють взаємодіяти із системою клієнтським застосункам, передаючи дані у форматі JSON;

– здійснення адміністрування (виконання операцій створення, редагування, перегляду та видалення інформації, створення резервних копій налаштувань та даних і відновлення системи до попереднього стану, керування сертифікатами);

– визначення, чи надіслані пристроєм показники відповідають нормі, у випадку негативної відповіді формування рекомендації для користувача;

– формування статистики показань за певний період часу (день, тиждень, місяць).

Клієнтська частина:

– відображення форм авторизації й реєстрації, передбачивши валідацію введених користувачами даних до відповідних полів;

– локалізація (зміна мови інтерфейсу між українською та англійською);

– інтернаціоналізація (зміна формату дати на графіках при відображенні статистики);

– надання списку поточних пристроїв, зареєстрованих користувачем, видалення їх із системи та зміна назви;

– керування раціонами із автоматичним підрахунком калорійності та вмісту мікроелементів для вказаної маси продукту;

– відображення у зручному вигляді усіх показників здоров’я, що відстежуються, а також їх значення для конкретного користувача на основі даних, отриманих із розумного пристрою;

– відображення статистики показників здоров’я у вигляді графіків та діаграм за вибраний проміжок часу;

– відображення інтерфейсу адміністратора.

Мобільний застосунок;

– відображення екрану авторизації;

– локалізація;

– інтернаціоналізація;

– внесення даних про прийом їжі, базуючись на раціоні, при цьому коригуючи вміст кожного продукту у разі потреби;

– відображення інформації про поточний стан здоров’я користувача системи;

– відображення сповіщень із детальною інформацією про те, який показник здоров’я не у нормі, а також його значення.

Розумний пристрій:

– збір показнків здоров’я;

– надсилання відповідних показників на веб-сервер.

3 АРХІТЕКТУРА ТА ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 UML проєктування ПЗ

3.1.1 Створення діаграми прецедентів

З метою визначення сценаріїв, за якими різні користувачі взаємодіють із програмною системою, опису її поведінки у відповідь на таку взаємодію було створено діаграму прецедентів (див. рис. 3.1).

Рисунок 3.1 – Діаграма прецедентів програмної системи

Відповідно до наведеної Use Case діаграми, існує всього два типи користувачів: адміністратор, що здійснює керування системою, та клієнт, який її використовує для відслідковування стану свого здоров’я.

У компетенцію адміністратора входить керування даними (виконання CRUD-операцій), створення резервних копій даних та налаштувань, відновлення системи до стану, збереженого у межах резервної копії, виконання оновлення SSL-сертифікату, якщо він застарів (можна переглянути детальну інформацію про сертифікат, у тому числі і строки його дії).

Клієнт програмної системи може керувати власними пристроями (додавати нові, змінювати інформацію про вже існуючі чи видаляти зайві), із яких надходитимуть показники здоров'я, налаштовувати індивідуальні раціони, визначаючи продукти та їх вміст для сніданку, першого перекусу, обіду, другого перекусу та вечері, отримати дані про останні показники здоров'я (якщо присутні лише застарілі дані, то система вказуватиме, що актуальні дані наразі відсутні), переглянути графіки, що відображають динаміку зміни показників із плином часу, при цьому самостійно треба вибрати, який період часу підлягає аналізу (день, тиждень, місяць), вказати дані стосовно прийому їжі, обравши раціон та безпосередньо прийом їжі та відредагувавши масу кожного із продуктів у разі потреби. Як для адміністратора, так і для клієнта передбачено можливість змінити мову, якою відображатиметься інформація у клієнтських застосунках.

3.1.2 Розробка діаграми розгортання системи

Для відображення компонентів, із яких складається система на фізичному рівні, а саме усіх пристроїв, котрі входять до її складу, а також протоколів взаємодії між ними, було розроблено діаграму розгортання.

Із неї видно, що центральним пристроєм, що зв'язує між собою усі інші складові системи, є back-end сервер, запити до якого обробляє Python-фреймворк Django. Nginx веб-сервер обробляє усі HTTP-запити від веб-сайту та мобільного застосунку, віддаючи одразу статичні файли, проте, коли виникає необхідність

програмно обробити запит, за допомогою WSGI-протоколу відбувається взаємодія із web-сервером Gunicorn, написаному на Python, який і працює вже на стороні Django. При цьому, посередником між веб-застосунком та веб-сервером є front-end сервер, який працює на базі фреймворку React за відповідає за функціонування односторінкового застосунку, вказуючи, яким чином необхідно формувати веб-сторінку відповідно до тих чи інших дій користувача (див. рис. 3.2).

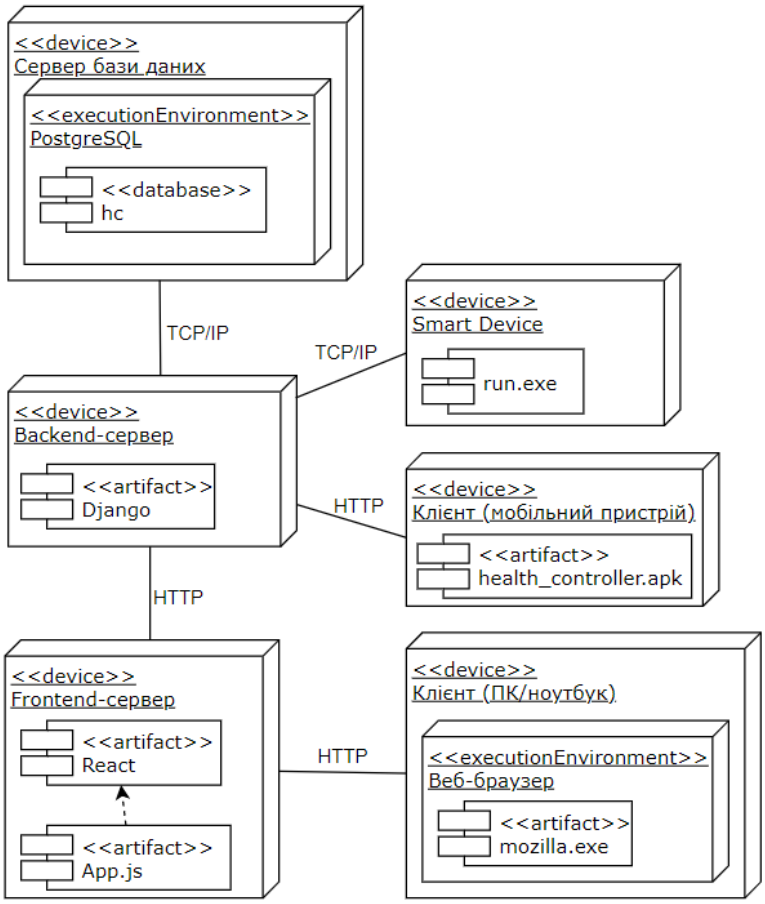

Рисунок 3.2 – Діаграма розгортання програмної системи

Коли виникає необхідність взаємодіяти із базою даних, ініціюється TCP/IP-з’єднання із сервером БД під управлінням СКБД PostgreSQL. Smart-device так само з’єднується із веб-сервером за допомогою протоколу TCP/IP з метою передачі показників здоров’я.

3.1.3 UML проектування веб-застосунку

З метою відображення усіх програмних модулів (React-компонентів, css-таблиць, бібліотек), які укупі забезпечують коректне функціонування веб-сайту у складі програмної системи трекінгу стану здоров'я, було створено діаграму компонентів (див. рис. 3.3).

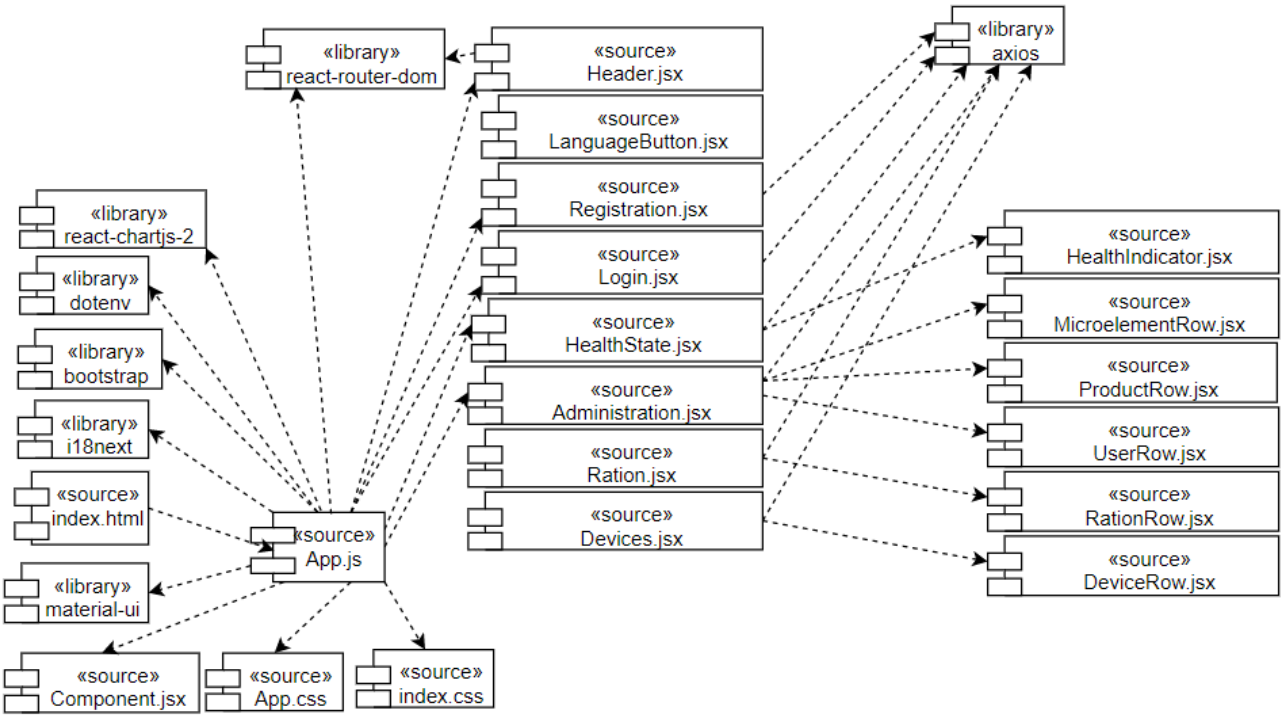

Рисунок 3.3 – Діаграма компонентів веб-застосунку

Виконання програми розпочинається із файлу App.js, який вбудовує увесь згенерований вміст у тег div файлу index.html із ідентифікатором «root». Шапка сайту, кнопка зміни мови, сторінки реєстрації, авторизації, відображення стану здоров'я, адміністрування, раціонів та пристроїв користувача є окремими компонентами React, поміщеними до .jsx файлів, при чому чотири останніх мають у своєму складі таблиці, за рядки яких відповідають окремі компоненти, зберігаючи дані про кожен запис.

У межах застосунку присутні залежності від зовнішніх бібліотек. Усі власні компоненти, окрім шапки сайту та кнопки зміни мови, використовують

axios для виконання HTTP-запитів, react-router-dom дозволяє здійснювати навігацію веб-сайтом, bootstrap містить готові стилі для елементів інтерфейсу, react-chartjs-2 дозволяє будувати графіки, візуалізуючи статистичні дані, i18next здійснює відображення вмісту веб-сайту відповідно до обраної мови.

Усі власноруч створені React-компоненти розподілено по пакетах із різним рівнем вкладеності, як показано на рис. 3.4.

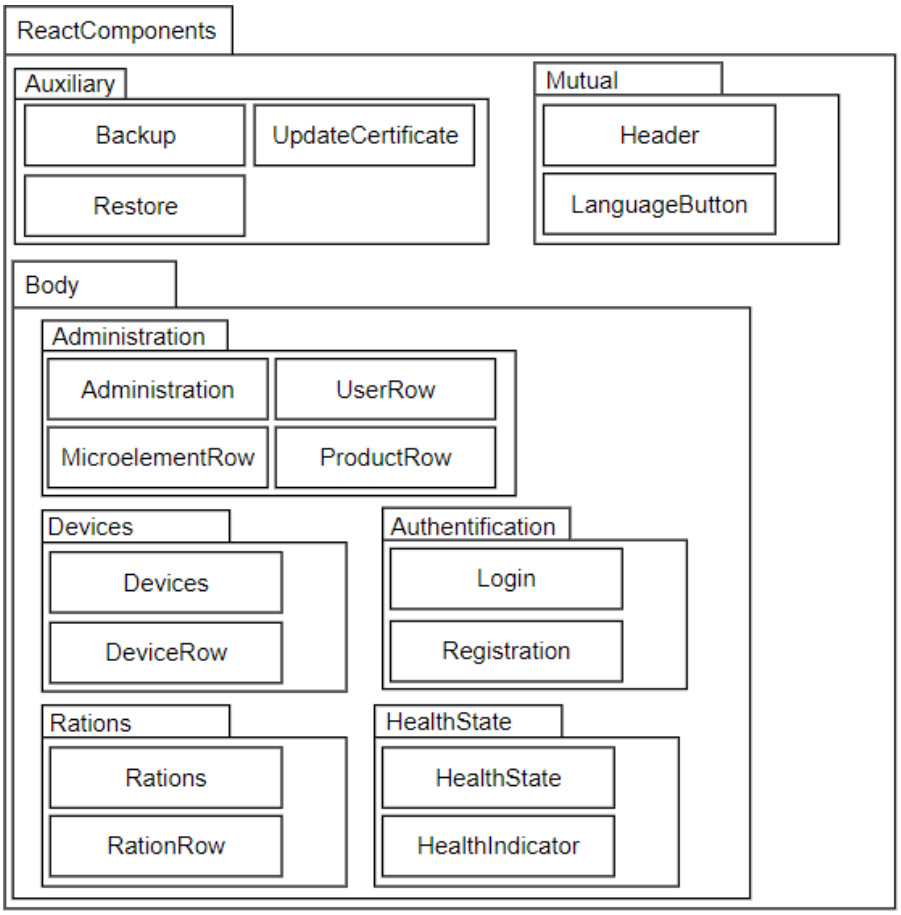

Рисунок 3.4 – Діаграма пакетів веб-застосунку

Усі сторінки містять шапку та мову зміни інтерфейсу, тому їх винесено до окремого пакету Mutual, компоненти, що складають тіло сторінки, винесено до Body, де, залежно від сторінки веб-сайту, вкладено компоненти, що є її складовими. Так, пакет Administration містить усі компоненти, що необхідні при формуванні інтерфейсу сторінки адміністрування: Administration – власне сторінка адміністратора, UserRow, MicroelementRow, ProductRow – компоненти, необхідні для побудови таблиці відповідної сутності.

3.2 Проєктування архітектури ПЗ

3.2.1 Проєктування архітектури серверної частини системи

Серверна частина програмного продукту має двошарову архітектуру: вона містить DAL та BLL. Будуючи веб-застосунок на базі фреймворку Django, застосовано MVC-шаблон. Використовуючи можливості Django ORM, із моделей, які являють собою звичайні Python-класи, відбувається створення таблиць БД. У якості СКБД виступає PostgreSQL.

Кінцеві точки API створюються із використанням Django REST framework. Вони дозволяють іншим застосункам взаємодіяти із сервером шляхом виконання HTTP-запитів, при цьому використовується REST-архітектура. Створені серіалізатори перетворюють об’єкти ORM у зручний для передачі формат JSON. Для цього у rout-ах вказуються вигляди, методи яких поветрають необхідні дані відповідно до бізнес-логіки.

Механізм аутентифікації користувачів відбувається завдяки використанню JWT-токенів, значний функціонал для чого вже вбудовано у бібліотеку Djoser. У процесі авторизації клієнт отримує токен доступу, який потім використовується у заголовках запитів до сервера для контролю доступу до відповідних ресурсів. Коли спливає час дії цього токену, новий можна отримати, застосувавши refresh-токен, також отриманий при авторизації.

Для автоматизації створення резервних копій та відновлення БД використано модуль django-dbbackup.

3.2.2 Проєктування архітектури клієнтської частини системи

Клієнтські застосунки, як мобільний, так і веб-сайт, мають однакову архітектуру, що пов’язано зі спільним призначенням: у зручному для

користувача вигляді відобразити елементи інтерфейсу, що надають користувачеві певну інформацію. Тому архітектура є однорівневою (усі складові системи знаходяться в межах одного пристрою) двошаровою (виділяється шар доступу до даних, який використовується для отримання даних із веб-сервера, а також шар відображення, що безпосередньо генерує код розмітки для створення інтерфейсу користувача). Із патернів використовується Одинак – Authentificator, що є класом, у межах застосунку створюється єдиний його екземпляр, що потім застосовується у різних фрагментах коду. Його призначення – керування JWT-токенами в процесі авторизації та аутентифікації.

3.2.3 Проєктування архітектури розумного пристрою

Архітектура розумного пристрою однорівнева (так само увесь код знаходиться на одному розумному пристрої) двошарова (шар бізнес-логіки та шар доступу до даних).

BLL відповідає за отримання і аналіз показників здоров’я клієнта, мінімальну обробку отриманих даних, а DAL – за їх серіалізацію у формат JSON та відправлення до веб-сервера із використанням протоколу TCP/IP. Така передача відбувається із використанням сокетів.

3.3 Проєктування структури бази даних

Для того, аби показати, які сутності присутні у базі даних, які поля вони мають, їхні первинні та зовнішні ключі, які зв’язки поєднуютьтаблиці між собою, створено ER-діаграму (див. рис. 3.5).

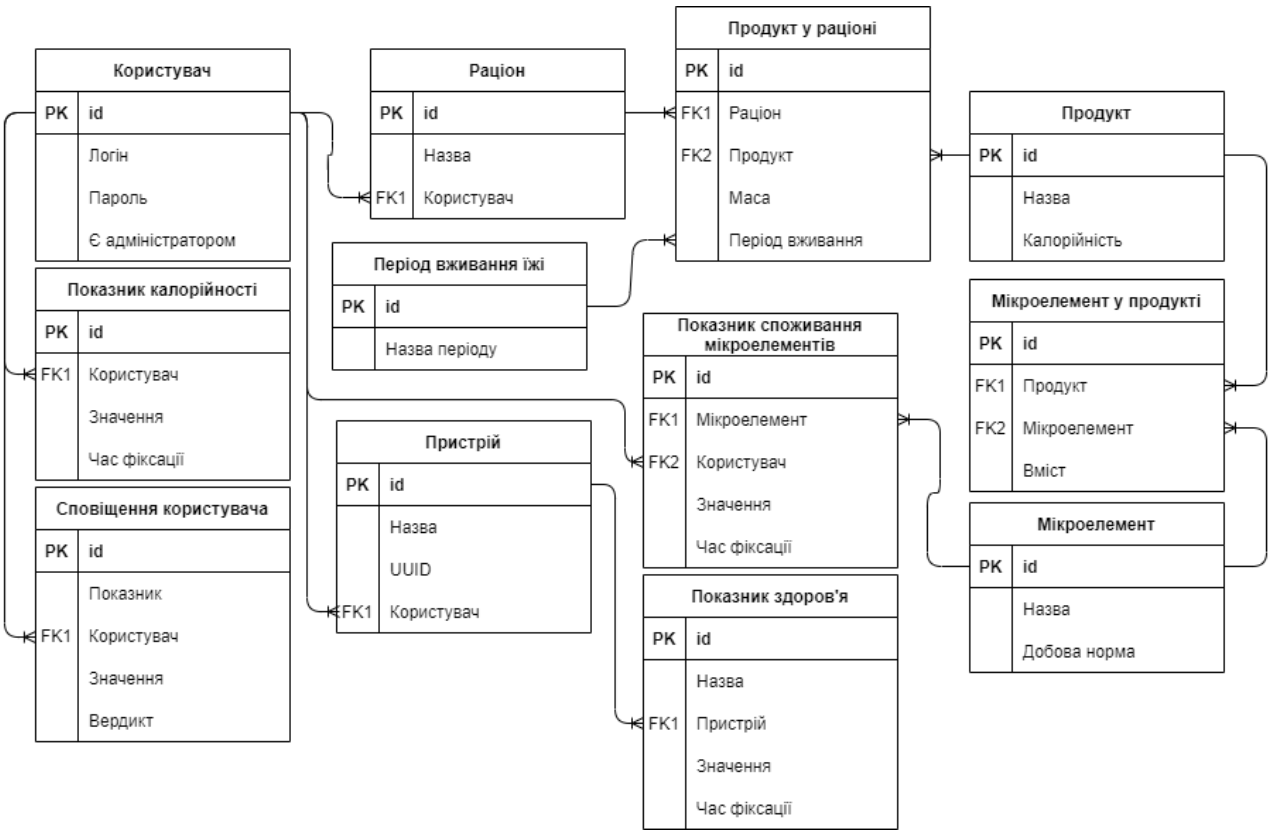

Рисунок 3.5 – ER-діаграма бази даних системи

Із неї видно, що БД має наступні таблиці:

– користувач (адміністратор чи клієнт, що використовує веб-сайт чи мобільний застосунок при взаємодії із системою);

– раціон (вміст продуктів, що мають бути спожиті під час прийомів їжі протягом доби);

– продукт (частина страви чи окрема сутність, що підлягає вживанню людиною);

– продукт у раціоні (визначає продукти, що містяться у певному раціоні);

– мікроелемент (хімічний елемент, присутній в організмах живих істот в низьких концентраціях: тисячні долі відсотка та нижче);

– мікроелемент у продукті (вміст мікроелементу у продукті харчування);

– період вживання їжі (час прийому їжі: сніданок, обід, вечеря, перекус);

– показник калорійності (показники калорійності, що ґрунтуються на вжитих користувачем продуктах);

– показник споживання мікроелементів (показники отриманих із їжі мікроелементів, що ґрунтуються на вжитих користувачем продуктах);

– пристрій (сукупність технічних засобів, що здійснюють отримання показників здоров’я та надсилають відповідні дані на сервер);

– показник здоров’я (показники, що характеризують здоров’я людини);

– сповіщення користувача (інформація про критичні показники користувача).

Зв’язки між таблицями «багато до багатьох»:

– раціон – продукт (один раціон містить багато продуктів, один продукт входить до багатьох раціонів);

– продукт – мікроелемент (в одному продукті може знаходитися декілька мікроелементів, один мікроелемент входить до складу різних продуктів);

– користувач – мікроелемент (у їжі одного користувача знаходиться багато мікроелементів, один мікроелемент знаходиться у їжі багатьох користувачів);

– період вживання їжі – продукт (в межах одного вживання їжі споживаються декілька продуктів, один продукт може бути з’їдено під час різних прийомів їжі).

Іншим видом зв’язку є «один до багатьох»: користувач – раціон (у одного користувача може бути кілька раціонів), користувач – показник калорійності (для одного користувача із плином часу збирається великий обсяг статистичних даних стосовно калорій, спожитих ним), користувач – сповіщення користувача (у одного користувача може бути кілька сповіщень), користувач – пристрій (один користувач може бути власником кількох пристроїв), пристрій – показник здоров’я (один пристрій фіксує багато різних показників здоров’я).

4 ПРОГРАМНІ ЗАСОБИ ДЛЯ РОЗРОБКИ ПРОЕКТУ

Під час розробки серверної частини програмної системи трекінгу балансу мікроелементів та стану здоров'я людини, що займається спортом, було застосовано можливості Python-фреймворку Django 3.2.5. Для створення кінцевих точок в рамках надання клієнтським застосункам програмного інтерфейсу використано модуль Django REST framework (DRF) 3.12.5.

За зберігання даних системи відповідає система керування базами даних Postgres 13. Вона є реляційною, що дозволяє повною мірою реалізувати збереження структурованих даних відповідно до ER-діаграми, створивши необхідні таблиці та зв’язки між ними. Для з’єднання із базою даних у Python-коді використовуються драйвери СКБД, тому необхідно встановити бібліотеку psycopg2-binary 2.9.1. Інструментом, який автоматизує процес аутентифікації за допомогою JSON Web Token-ів, виступає djoser 2. За встановлення SSL-сертифікату на локальний сервер відповідає утиліта mkcert, аби запустити сервер Django для підключення до нього за допомогою безпечного протоколу HTTPS.

При розробці веб-сайту використано фреймворк React, мова розмітки веб-сторінок – HTML, стилі для компонентів інтерфейсу задаються за допомогою CSS. Застосовано низку сторонніх бібліотек, таких як react-bootstrap (для стилізації), axios (для виконання HTTP-запитів), react-chartjs-2 (для побудови графіків).

Мобільний застосунок розрахований на користувачів Android 10, рішення є не кросплатформеним, а нативним, написаним мовою програмування Kotlin 1.4.30. Використано бібліотеки jjwt-api 0.11.2 (для роботи із JWT-токенами), volley 1.2.0 (виконання запитів до веб-сервера), flexbox 3.3.0 (з метою використання гнучких контейнерів для елементів інтерфейсу користувача).

При програмуванні плати Arduino використовувалася мова програмування C++, а також застосовано рішення бібліотек Ethernet, ArduinoJson, Bounce2.

5 ОСОБЛИВОСТІ РЕАЛІЗАЦІЇ ТА ЗАСТОСУВАННЯ ПРОДУКТУ

Основною бізнес-логікою, реалізованою в межах проекту, є аналіз отриманих із розумних пристроїв користувачів різного роду показників здоров’я разом із показниками спожитих калорій та мікроелементів, підрахованих при вказанні даних про прийом їжі. Є наперед визначені показники, що аналізуються, такі як пульс, систолічний та діастолічний тиск, сатурація, кількість кроків, пройдених за день. Для таких показників є верхня та нижня межа, тому програмно визначається, чи передане значення в нормі. Що стосується калорійності та мікроелементів, то в межах запиту до БД отримується агреговане сумарне значення з початку до кінця доби, яке вже у подальшому порівнюється із допустимим діапазоном (див. дод. А).

У тому разі, якщо показник має критичне значення, відповідний запис створюється у таблиці повідомлень користувачів. Пристрій авторизованого користувача час від часу виконує запити до веб-серверу, опитуючи стосовно наявності нових повідомлень. Тому в тому разі, коли користувач має показники, що відхиляються від норми, відповідно до отриманих значень із сервера формується текст повідомлення, яке потім відображається у «шторці». Воно містить поточне значення, вказівку, вище чи нижче допустимих показників воно знаходиться, а також рекомендації стосовно того, як краще відреагувати на ситуацію, що сталася.

До бізнес-логіки також належить підрахунок кількості калорій та вміст мікроелементів автоматично при зміні маси продукту на веб-сайті у списку раціонів та в мобільному застосунку при вказанні інформації про поточний прийом їжі.

Особливість застосування програмного продукту полягає в тому, що без придбання розумного пристрою зникає сенс роботи із системою в цілому. В такому разі увесь функціонал, що буде доступний клієнтові, полягатиме лише у відслідковуванні калорійності та спожитих мікроелементів.

6 ДИЗАЙН ПРОГРАМНОЇ СИСТЕМИ

6.1 Розробка інтерфейсу веб-сайту

При розробці веб-сайту в рамках програмної системи було створено такі сторінки:

– раціони, де є можливість додати новий раціон, увівши його назву, обрати наявний із переліку, після чого відбувається відображення вибраних продуктів для передбачених прийомів їжі (сніданку, обіду, вечері, перекусів), при цьому підраховуючи калорійність при вказаній масі. Дана таблиця передбачає додавання нових страв, редагування та видалення (див. рис. 6.1);

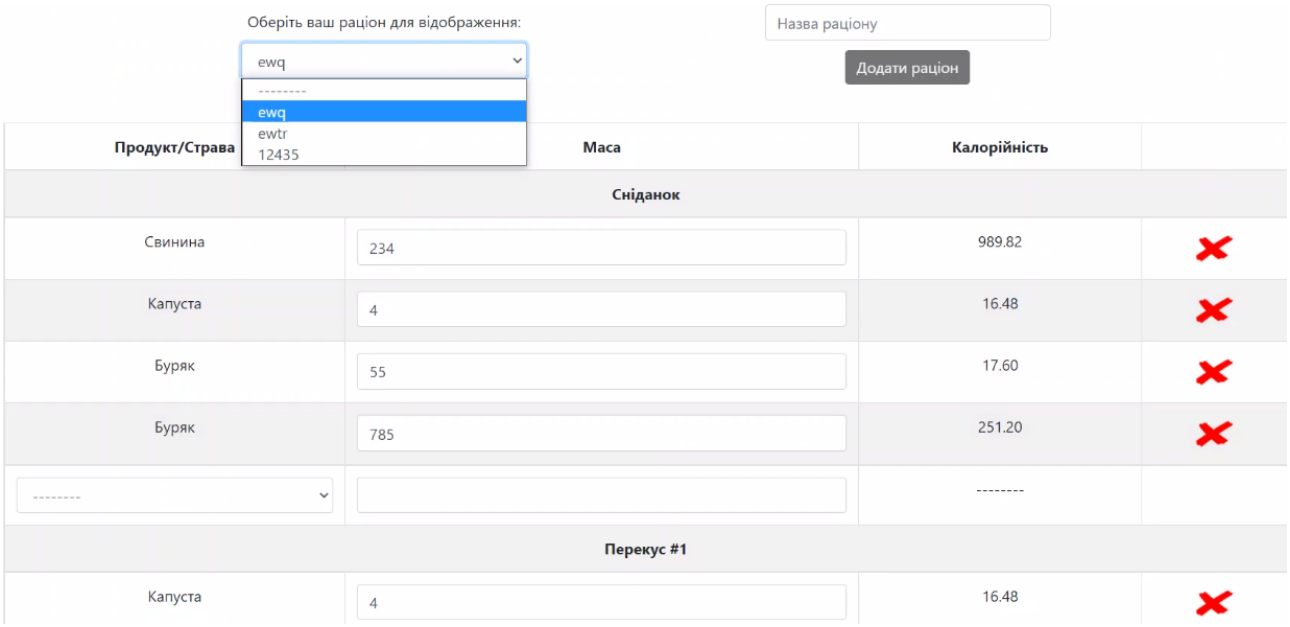

Рисунок 6.1 – Сторінка раціонів

– стан здоров’я. Для таких показників, як пульс, тиск, сатурація, кількість кроків, кількість спожитих калорій відображаються відповідні дані залежно від показника (або останнє значення, або сумарне за добу). Кольором показано стан даного показника (зелений – у нормі, червоний – менше або більше норми, жовтий – актуальні дані відсутні). Нижче на даній сторінці відображено графіки, які демонструють динаміку зміни значень показників із часом (див. рис. 6.2);

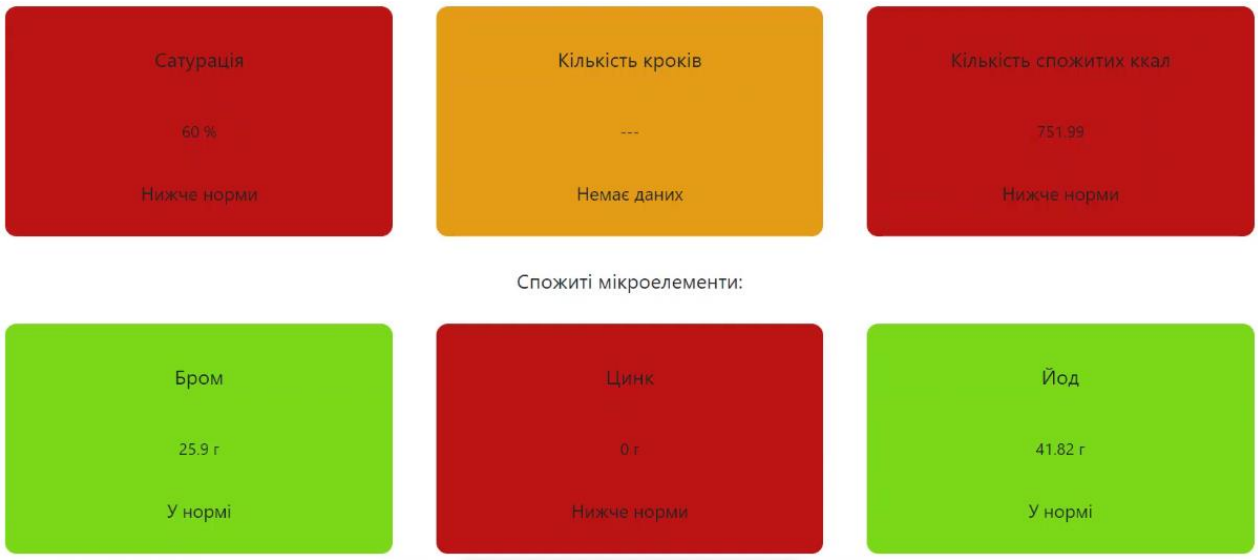

Рисунок 6.2 – Сторінка стану здоров’я

– сторінка пристроїв користувача, де в таблиці можна переглянути зареєстровані розумні девайси, відредагувати інформацію про них, або ж додати нові, увівши зручну для запам’ятовування назву, аби розуміти, із якого пристрою були надіслані дані, а також унікальний ідентифікатор пристрою, що присвоюється йому після виготовлення;

Рисунок 6.3 – Сторінка пристроїв користувача

– сторінка реєстрації, де нові користувачі у поля форми вводять логін, під яким бажають бути розпізнані програмною системою, а також двічі пароль, аби уникнути помилок. Після успішної реєстрації відбувається перенаправлення на сторінку авторизації, у разі невдачі можна ознайомитися із підказкою стосовно того, що необхідно виправити;

– сторінка авторизації, яка пропонує увести логін та пароль, використаний при проходженні процесу реєстрації. У разі введення коректних даних користувач отримає доступ до сторінок відповідно до ролі (див. рис. 6.4).

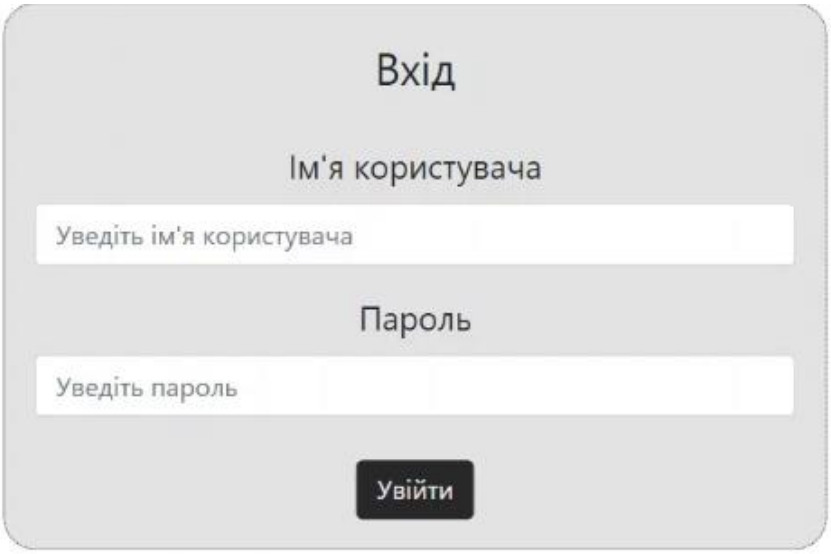

Рисунок 6.4 – Сторінка авторизації

– сторінка адміністратора, де є можливість обрати таблицю бази даних. Після цього в табличному вигляді буде відображено збережені сутності, поля яких можна редагувати чи видаляти, а у разі потреби додати нові сутності. Розміщені у правому нижньому кутку кнопки дозволяють керувати SSL-сертифікатами, створювати резервні копії бази даних, відновлювати стан БД відповідно до резервних копій (див. рис. 6.5).

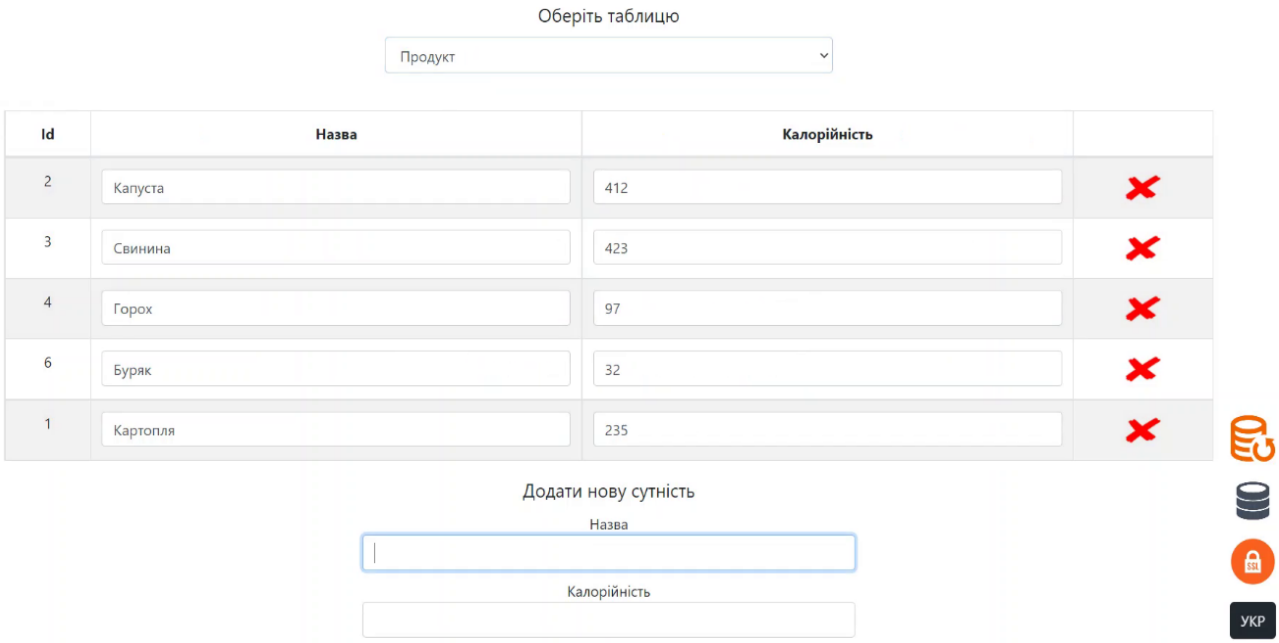

Рисунок 6.5 – Сторінка адміністратора

6.2 Розробка інтерфейсу мобільного застосунку

Мобільний застосунок має наступні екрани:

– екран авторизації, де для отримання доступу до функціоналу застосунку клієнтові потрібно увести логін та пароль, створений при проходженні процесу реєстрації;

– екран стану здоров’я, де відображено показники пульсу, систолічного тиску, діастолічного тиску, сатурації, кількість пройдених протягом доби кроків та спожитих калорій. Нижче можна переглянути обсяг спожитих із їжею мікроелементів. Як і у випадку із веб-сайтом, різними кольорами відображено відповідність даних нормі (див. рис. 6.6);

Рисунок 6.6 – Екран стану здоров’я

– вживання їжі, де обирається раціон користувача, прийом їжі, відповідно до налаштувань раціону відображаються продукти та їх вміст за змовчуванням,

який можна тут же відредагвати та побачити перераховані показники калорійності та вмісту мікроелементів. При натисканні на кнопку «поїсти» ці дані заносяться до БД, що потім враховується при веденні статистики і відстеженні поточного стану здоров’я (див. рис. 6.7).

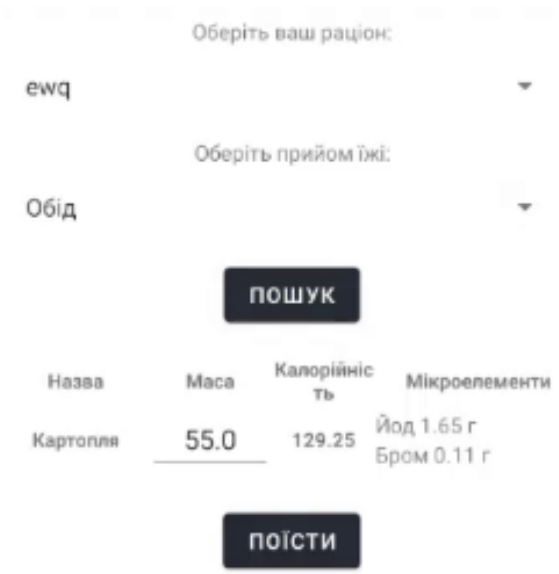

Рисунок 6.7 – Екран вживання їжі

У випадку, якщо із веб-сервера буде отримано дані про показники здоров’я, значення яких є поза нормою, веб-застосунок згенерує повідомлення, що відобразиться у «шторці» (див. рис. 6.8).

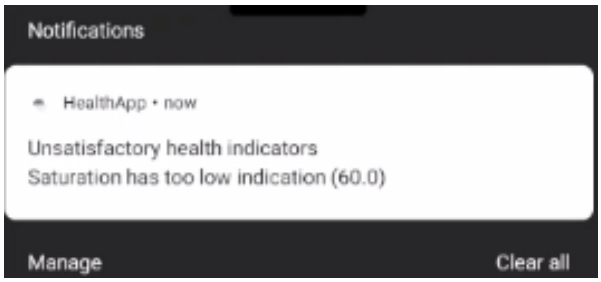

Рисунок 6.8 – Спливаюче повідомлення

Як веб-сайт, так і мобільний застосунок підтримує локалізацію: можна перемикатися між українською та англійською мовою інтерфейсу. У рамках підтримки інтернаціоналізації відбувається зміна формату дат.

ВИСНОВКИ

У рамках курсової роботи проаналізовано декілька предметних областей на наявність важливих проблем, обрано найактуальнішу із них, детально проаналізовано ринок, ризики, конкурентів та їхні програмні продукти, визначено критерії успіху, досягши яких можна з упевненістю сказати, що поставлену задачу успішно виконано, спланово архітектуру, складові частини (веб-сервер, веб-застосунок, мобільний застосунок, розумний пристрій), визначено інструменти для реалізації системи та безпосередньо виконано її кодування. Створено детальну специфіацію спроектованої системи (див. дод. Б). У результаті роботи створено програмну система трекінгу балансу мікроелементів та стану здоров'я людини, що займається спортом.

При плануванні архітектури визначено кількість рівнів (кількість фізичних пристроїв, у межах яких виконано розгортання програмного продукту), шарів (логічного поділу системи на частини відповідно до призначення, які відокремлені одна від одної). Вказано протоколи передачі даних між частинами системи, а також формат, який використовується при серіалізації. За допомогою графічних засобів та мови UML розроблено низку діаграм, які відображають, яким чином користувач може взаємодіяти із системою, із яких складових (фізичних, логічних) вона складається, у яких станах вона може знаходитися та яку структуру мають дані, що зберігаються у реляційній базі даних.

Для написання програмного коду використано такі IDE, як PyCharm, WebStorm, Arduino IDE, для керування базою даних – програмний інструмент PgAdmin. Створена програма дозволяє відстежувати поточний стан здоров’я людини завдяки фіксації показників здоров’я розумним пристроєм та аналізу отриманих даних на стороні веб-сервера. Велика увага приділяється здоровому харчуванню, тому підтримується створення раціонів та фіксування процесу споживання їжі. Програмна система підтримує інтерфейс адміністрування, веб-сайт та мобільний застосунок підтримують інтернаціоналізацію й локалізацію.

ПЕРЕЛІК ПОСИЛАНЬ

1. Авцын А. П., Жаворонков А. А. Микроэлементозы человека: этиология, классификация, органопатология. СПб.: Питер, 1991. 285 с.

2. Коломийцева М. Г., Габович Р. Д. Микроэлементы в медицине. М.: Медицина, 1971. 406 с.

3. Максимова Т. М. Современное состояние, тенденции и перспективные оценки здоровья населения. Смоленск: Смоленск Плюс, 2009. 288 с.

4. Бабенко Г. А. Микроэлементозы человека: патогенез, профилактика, лечение. М.: Медицина, 2001. 316 с.

5. Тутельян В. А., Спиричев В. Б. Микронутриенты в питании здорового и больного человека. М.: Медицина, 2002. 412 с.

6. Пшендин П. И. Рациональное питание спортсменов. СПб.: Олимп СПб, 2003. 341 с.

7. Скальный А. В. Химические элементы в физиологии и экологии человека. М.: ОНИКС 21 век, 2004. 218 с.

ДОДАТОК А

Реалізація оцінки стану здоров’я клієнта

```
class UserStateView(View):

    indications = {
        'pulse': [60, 90],
        'systolic_pressure': [100, 130],
        'diastolic_pressure': [70, 90],
        'saturation': [94, 100],
        'steps': [10000, 100000000],
    }

    def get(self, request, *args, **kwargs):
        user = models.User.objects.get(id=request.GET.get('user'))
        current_datetime = datetime.datetime.now()
        last_datetime = current_datetime -
            datetime.timedelta(days=1)

        res = {
            k: self.get_indication_verdict(
                k, user, last_datetime, v[0], v[1]
            )
            for k, v in UserStateView.indications.items()
        }

        calories_count = self.get_calories_count(
            user, last_datetime
        )
        if calories_count is None:
            calories_count = 0
        res['calories'] = self.check_value(
            calories_count, 1800, 2600
        )

        res['microelements'] = self.get_microelement_count(
            user, last_datetime
        )

        return HttpResponse(dumps(res, ensure_ascii=False))

    def get_indication_verdict(
            self, indication, user, last_datetime,
            min_value, max_value
        ):
        last_appropriate_time = self.get_indication_last_time(
            indication, user, last_datetime
        )
        if not last_appropriate_time:
            return -100, -1
```

```
        last_appropriate_time_value = \
            self.get_indication_last_time_value(
                indication, user, last_appropriate_time
            )
        return self.check_value(
            last_appropriate_time_value, min_value, max_value
        )

    @staticmethod
    def get_indication_last_time(indication, user, last_datetime):
        return models.HealthIndications.objects.filter(
            name=indication,
            device__user=user,
            datetime__gt=last_datetime
        ).aggregate(Max('datetime'))['datetime__max']

    @staticmethod
    def get_indication_last_time_value(
            indication, user, indication_last_time
        ):
        return models.HealthIndications.objects.filter(
            name=indication,
            device__user=user,
            datetime=indication_last_time
        ).values_list('indication', flat=True)[0]

    @staticmethod
    def check_value(current_value, min_value, max_value):
        if current_value < min_value:
            return -1, current_value
        elif current_value > max_value:
            return 1, current_value
        else:
            return 0, current_value

    @staticmethod
    def get_last_day_boundaries(last_datetime):
        yesterday_date = last_datetime.date()
        yesterday = datetime.datetime(
            yesterday_date.year,
            yesterday_date.month,
            yesterday_date.day
        )
        today = yesterday + datetime.timedelta(days=1)
        tomorrow = today + datetime.timedelta(days=1)
        return today, tomorrow

    def get_calories_count(self, user, last_datetime):
        today, tomorrow = \
            self.get_last_day_boundaries(last_datetime)
        return models.CaloriesIndications.objects.filter(
            datetime__gte=today,
            datetime__lte=tomorrow,
```

```
            user=user
        ).aggregate(Sum('indication'))['indication__sum']

    def get_microelement_count(self, user, last_datetime):
        today, tomorrow = \
            self.get_last_day_boundaries(last_datetime)
        microelements = list(
            models.Microelement.objects.all()
            .values_list('id', 'daily_norm', 'name')
        )
        res = {}

        for microelement in microelements:
            microelement_yesterday_value = \
                models.Microelement.objects.filter(
                microelementindications__datetime__gte=today,
                microelementindications__datetime__lte=tomorrow,
                microelementindications__user=user,
                id=microelement[0]
            ).aggregate(Sum(
                 'microelementindications__indication'
            ))['microelementindications__indication__sum']

            if microelement_yesterday_value is None:
                microelement_yesterday_value = 0
            res[microelement[2]] = self.check_value(
                microelement_yesterday_value,
                microelement[1],
                10000000
            )

        return res
```

ДОДАТОК Б

Специфікація ПЗ

Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп’ютерних наук

Кафедра програмної інженерії

СПЕЦИФІКАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

програмної системи трекінгу балансу мікроелементів та стану здоров'я людини, що займається спортом

Студент гр. ПЗПІ-18-1 _____ Жиденко І. В.
(підпис)

Харків
2021 р.

ЗМІСТ

1 ВСТУП

1.1 Огляд ПЗ

У специфікації докладно описано, які функціональні й нефункціональні вимоги було висунуто до програмного продукту, що повинен значно спростити процес відслідковування поточного стану здоров’я, максимально швидко виявляти відхилення від норми та повідомляти про це користувача, аби уникнути хвороб та ускладнень.

Архітектура програмної системи передбачає наявність таких складових, як веб-сервер, веб-застосунок та мобільний застосунок, розумний пристрій, що отримує показники здоров’я клієнта та надсилає їх до веб-сервера, звідки їх зможуть шляхом виконання запитів отримати інші частини. Це свідчить про те, що усі складові пов’язані між собою та взаємодіють у ході роботи.

1.2 Мета

Специфікація програмної системи трекінгу балансу мікроелементів та стану здоров'я людини, що займається спортом, була розроблена з метою документування загального опису, конкретних вимог (функціональних та нефункціональних), характеристики потенційних користувачів, на яких розрахована програмна система, факторів, від яких залежить її робота та успішне введення в експлуатацію.

В рамках опису функціональних вимог детально розглянуто, які саме функціональні можливості мають бути реалізовані у програмному продукті, а при визначенні нефункціональних вимог розглянуто низку властивостей, що описують безпеку, потративність, апаратне забезпечення, необхідне для коректної роботи.

1.3 Межі

Готова програмна система повинна надавати користувачеві можливість відстежувати стан його здоров’я на основі наперед визначених характеристик, таких як пульс, систолічний та діастолічний тиск, сатурація, кількість пройдених за добу кроків та обсяг спожитих калорій. Важливою особливістю системи є надання інформації про отримані із їжею мікроелементи, наявність у достатній кількості в організмі людини яких дозволяє уникнути розвитку значної кількості хвороб. Для усіх показників здоров’я, що підлягають аналізу, має бути доступна статистика, розрахована за чітко визначені проміжки часу (доба, тиждень, місяць), али можливості самостійно вказати кількість днів не передбачено. Такі дані мають бути візуалізовані у вигляді графіка чи діаграми, на осі абсцис відкладаються дати, ординат – зафіксовані значення, до яких застосовано агрегуючу функцію (середнє значення чи максимальне). Додавання власних показників здоров’я не є можливим, що пов’язано із специфікою реалізації розумного пристрою.

Має бути розроблена сторінка адміністрування, що містить таблиці користувачів, продуктів та мікроелементів. Особисті дані користувачів не повинні бути доступні адміністратору.

1.4 Посилання

1. Data Access Layer : веб-сайт. URL: https://metanit.com/sharp/mvc5/23.7.php (дата обращения: 03.08.2021).

2. What is an API? : web-site. URL: https://www.mulesoft.com/resources/api/what-is-an-api (accessed: 03.08.2021).

3. Reading: Random Access Memory : web-site. URL: https://www.avast.com/c-what-is-ram-memory (accessed: 03.08.2021).

4. Введение в ORM (Object Relational Mapping) : веб-сайт. URL: http://internetka.in.ua/orm-intro/ (дата обращения: 03.08.2021).

1.5 Означення та абревіатури

Нижче наведено тлумачення абревіатур й термінів, що зустрічаються у тексті даного документу:

– DAL (data access layer) – це шар комп'ютерної програми, який надає спрощений доступ до даних, що зберігаються в постійному сховище будь-якого типу, такому як реляційна база даних [1];

– API (application programming interface) – це набір способів і правил, за якими різні програми спілкуються між собою і обмінюються даними [2];

– RAM (random access memory) – пам'ять ЕОМ, призначена для зберігання коду та даних програм під час їхнього виконання. У сучасних комп'ютерах оперативна пам'ять переважно представлена динамічною пам'яттю з довільним доступом DRAM [3];

– ORM (Object-relational mapping) – це технологія програмування, яка дозволяє перетворювати несумісні типи моделей в ООП, зокрема, між сховищем даних і об'єктами програмування. ORM використовується для спрощення процесу збереження об'єктів в реляційну базу даних і їх вилучення, при цьому ORM сама піклується про перетворення даних між несумісними станами [4].

2 ЗАГАЛЬНИЙ ОПИС

2.1 Перспективи ПЗ

Програмна система трекінгу балансу мікроелементів та стану здоров’я людини, що займається спортом, має розроблятися із розрахунком на те, що інтеграція із жодним іншим програмним продуктом не передбачається, тобто вона є в цілому самодостатньою. Відсутні сервіси, API яких би використовувалися для отримання певної інформації. Наприклад, нормальні значення для показників здоров’я занесено до програмного коду і не мають отримуватися ззовні.

Програмний продукт містить чотири основні частини: бек-енд сервер, фронт-енд сервер, мобільний додаток та смарт-девайс. Обмін даними між ними забезпечується завдяки кінцевим точкам, наданим веб-сервером.

2.2 Функції ПЗ

Серед функцій, що повинні бути реалізовані перед уведенням програмного продукту в експлуатацію, варто виділити:

1. Створення власного профілю користувача та зберігання у ньому даних.
2. Фіксація розумними пристроями показників здоров’я та їх передача до веб-серверу.
3. Керування раціонами (додавання, редагування, видалення), визначаючи продукти для прийомів їжі у межах доби.
4. Внесення даних про прийом їжі у межах раціону. Хоча раціон і створюється, передбачаючи певний вміст кожного із продуктів, ці дані можуть дещо різнитися від дійсних, тому дозволяється їх змінювати. Проте, якщо певного продукту у раціоні немає для поточного прийому їжі, необхідно

створити новий раціон, який би повною мірою відображав наявність усіх компонентів обіду.

5. Ведення статистики поточного стану здоров’я, визначаючи статус відповідності окремих показників нормі.

6. Відстеження динаміки зміни показників здоров’я із часом (за добу, тиждень, місяць).

7. Надсилання повідомлень користувачеві про проблеми із здоров’ям у разі виявлення відхилань значень показників від норми.

2.3 Характеристики користувачів

Розробка проекту у першу чергу розрахована на використання спортсменами, котрі мають слідкувати за станом здоров’я, кількістю спожитих калорій та мікроелементів. Але інша частина функціоналу, не пов’язана із харчуванням, актуальна для широкого кола людей, насамперед, для тих, хто має проблеми із серцево-судинною та дихальною системами, тому ще одна велика група клієнтів – пенсіонери.

Що стосується першої категорії, спортсменів, то це особи переважно віком від 16 до 35 років, у більшій мірі чоловічої статі, проте тенденції останніх років показують, що значно збільшується відсоток і жінок. Освіта середня, адже люди, що займаються розумовою працею, набагато рідше є спортсменами і не мають багато часу на це. Рівень доходу – середній та високий, що пов’язано із вартістю розумного пристрою. Передбачається, що користувачі користуватимуться програмним продуктом постійно, адже важливо завжди турбуватися про стан власного здоров’я.

Коли мова йде про людей, для яких характерні певні види захворювань, то найчастіше це є люди віком більш, ніж 50 років, освіта не має значення, рівень доходу – високий чи середній, використання також є постійним.

2.4 Загальні обмеження

Існує ряд факторів, що впливають на систему, тим самим слугуючи джерелом обмежень. Серед них у першу чергу є необхідність стабільного підключення до мережі Інтернет, адже саме із її допомогою відбувається комунікація між окремими частинами продукту. До того ж, передача даних повинна відбуватися у рамках протоколу HTTPS, що забезпечує їх шифрування, а тому і значно підвищує безпечність цього процесу. Для забезпечення роботи цього механізму як на стороні веб-сервера, так і front-end застосунку має бути встановлено SSL-сертифікат.

Передбачений формат передачі даних – JSON, тому серіалізація зі сторони back-end серверу та десеріалізація клієнтськими застосунками має передбачати наявність програмних модулів, здатних працювати саме із цим форматом, адже підтримка жодного іншого не передбачається.

Під час розробки серверної частини програмної системи трекінгу балансу мікроелементів та стану здоров'я людини, що займається спортом, має бути застосовано можливості Python-фреймворку Django. Для створення кінцевих точок в рамках надання клієнтським застосункам програмного інтерфейсу варто використовувати модуль Django REST framework.

За зберігання даних системи відповідатиме система керування базами даних PostgreSQL. Для з’єднання із базою даних у Python-коді необхідно використати драйвери СКБД, тому треба встановити бібліотеку psycopg2-binary. Інструментом, який автоматизує процес аутентифікації за допомогою JSON Web Token-ів, має бути djoser 2. За встановлення SSL-сертифікату на локальний сервер повинна відповідати утиліта mkcert, аби запустити сервер Django для підключення до нього за допомогою безпечного протоколу HTTPS.

При розробці веб-сайту необхідно використати фреймворк React, мову розмітки веб-сторінок HTML, стилі для компонентів інтерфейсу мають задаватися за допомогою CSS. Необхідно застосувати низку сторонніх бібліотек,

таких як react-bootstrap (для стилізації), axios (для виконання HTTP-запитів), react-chartjs-2 (для побудови графіків).

Мобільний застосунок повинен бути розрахований на користувачів Android 10, рішення є не кросплатформеним, а нативним, написаним мовою програмування Kotlin. Передбачається використання бібліотек jjwt-api (для роботи із JWT-токенами), volley (виконання запитів до веб-сервера), flexbox (з метою використання гнучких контейнерів для елементів інтерфейсу користувача).

При програмуванні плати Arduino використовувалася мова програмування C++, а також застосовано рішення бібліотек Ethernet, ArduinoJson, Bounce2.

2.5 Припущення та залежності

Нормально функціонуюча програмна система припускає наявність у користувачів принаймні одного розумного пристрою, розгорнутий та успішно працюючий веб-сервер та фронт-енд сервер, а також наявність доступу до Інтернету для кожної із вище перелічених сторін. Не менш важливим є безперебійність електропочтачання, адже усі електричні пристрої потребують живлення від електромережі.

Кількість випущених розумних пристроїв значною мірою залежить від капіталовкладень третіх сторін, адже проект реалізується у межах стартапу і немає можливості за власні кошти фінансувати масове виготовлення таких пристроїв. До того ж, коректність отриманих значень показників здоров’я буде задовільною лише у випадку правильності кріплення його до тіла людини. Для отримання точної статистики стосовно спожитих калорій та мікроелементів необхідно після кожного прийому їжі вноОити до мобільного застосунку дані про спожиті продукти.

3 КОНКРЕТНІ ВИМОГИ

3.1 Вимоги до зовнішніх інтерфейсів

3.1.1 Інтерфейс користувача

Розглядаючи інтерфейс користувача, у першу чергу варто розглянути розміри екранів, які дозволять коректно працювати із продуктом, вмістивши усі необхідні UI-компоненти. Мінімальні розміри монітору комп’ютера повинні складати 1024x768, а екрану смартфону – 480x800.

Усі клієнтські застосунки повинні передбачати можливість зміни мови інтерфейсу. Мінімальним набором доступних мов є українська та англійська, проте для залучення ширшої аудиторії передбачається його розширення. Також необхідно передбачити функціонал, пов’язаний із інтернаціоналізацією, намасперед, механізм відображення дати відповідно до формату, пов’язаного із вибраною мовою.

Для веб-сайту обов’язковою вимогою є наявність шапки, у якій, окрім логотипу сайту, має міститися навігаційне меню, що дозволить потрапити до будь-якої сторінки веб-застосунку. Кнопка зміни мови має бути завжди в області видимості користувача, тому її необхідно зафіксувати у нижньому правому кутку. Усі елементи керування (кнопки, випадаючі списки, поля введення) повинні бути однотипно стилізовані і гармонійно поєднуватися один з одним у рамках темної теми. Сторінка адміністратора повинна мати зафіксовані кнопки створення резервної копії, відновлення налаштувань та даних, оновлення сертифікатів поряд із кнопкою зміни мови інтерфейсу.

Вгорі екрану мобільного застосунку повинні розташовуватися елементи навігації між різними екранами, а також кнопка зміни мови. До моменту завершення отримання необхідних даних із сервера необхідно показувати анімацію, що свідчить про процес завантаження, аби користувач не думав, що відповідний список порожній.

3.1.2 Апаратний інтерфейс

Нормальна робота back-end та front-end серверів передбачає наявність 64-розрядного процесора архітектури x86, мінімально допустима кількість ядер складає 8. Багатоядерність використовується задля одночасного обслуговування декількох клієнтів, адже кожен із них опрацьовується в окремому потоці. Рекомендовано використовувати рішення, які підтримують технологію Hyper-threading, адже віртуальні ядре ще більшою мірою здатні розвантажити процесор. Ще однією важливою характеристикою є частота і її значення повинно починатися від 3 ГГц. Мобільний пристрій, на якому відбуватиметься запуск застосунку, повинен мати багатоядерний процесор із частотою від 1 ГГц.

3.1.3 Програмний інтерфейс

Вимоги до програмного інтерфейсу здебільшого зумовлені системними вимогами до мов програмування, фреймворків, бібліотек, які мають бути використані в процесі розробки.

Веб-сервер повинен працювати на базі операційної системи Windows 7 чи вище, альтернативним варіантом є розгортання на Ubuntu 16.04. Для роботи із кодом, написаним мовою програмування Python, варто встановити її інтерпретатор.

Веб-застосунок може бути розгорнуто на Windows 8.1 чи вище, необхідною є наявність встановленого Node.js, а також пакетного менеджера yarn, що дозволяє отримувати сторонні програмні модулі.

Що стосується ПК клієнта, то він може працювати під ОС Windows XP та вище, має бути встановлено браузер, що підтримує JavaScript. Мобільний пристрій повинен працювати як мінімум на базі Android 4.4.

3.1.4 Комунікаційний протокол

Так як програмна система складається із декількох частин, що мають взаємодіяти між собою, то важливо однозначно визначити вимоги до комунікаційного протоколу.

Центральною частиною виступає веб-сервер, який і може слугувати посередником між усіма іншими складовими. Для отримання даних із бази даних та значень показників здоров'я із розумного пристрою використовується протокол TCP/IP. Коли ж клієнт за допомогою веб-сайту чи мобільного застосунку звертається до кінцевих точок, визначених на веб-сервері, дані передаються в межах HTTPS, виконуючи GET-, POST-, PUT-, PATCH- та DELETE-запити. Формат, що використовується при передачі даних між клієнтом та сервером – JSON.

3.1.5 Обмеження пам'яті

При розрахунку необхідного обсягу пам'яті слід відштовхуватися від того, що програмна система повинна бути спроможна обслуговувати до 50 тисяч користувачів одночасно, тому використання лише одного сервера вважається досить проблематичним. У зв'язку із цим було визначено, що при уведенні програмної системи в експлуатацію програмне забезпечення має бути розгорнуто на двох комп'ютерах, кожен із яких оперує 32 гігабайтами оперативної пам'яті. Для зберігання даних у БД необхідно мати 30 ТБ пам'яті на жорсткому диску, що пов'язано із інтенсивним надходженням нових даних показників здоров'я користувачів постійно.

Користувачі мобільного застосунку повинні мати від 2 ГБ RAM та 38 МБ вільного місця у внутрішньому сховищі для встановлення програми.

3.1.6 Операції

Серед операцій, виконання яких передбачено програмною системою, основна частина зосереджена на сторінці адміністратора. До них належить виконання CRUD-операцій (створення, перегляд, оновлення та видалення) над сутностями таблиць бази даних, таких як користувачі, продукти, мікроелементи. Також сюди варто віднести створення резервних копій, відновлення налаштувань та даних користувачів, здійснення керування сертифікатами, зміна мови інтерфейсу.

Проте клієнт також виконує низку операцій. До них належить створення та виконання маніпуляцій із раціонами, керування списком зареєстрованих пристроїв.

3.1.7 Функції ПЗ

Програмна система складається із чотирьох незалежних одна від одної частин, кодна із яких спеціалізується на вирішенні власного кола задач. Нижче наведено функції, котрі мають бути реалізовані для із них;

Серверна частина:

– виконання процесів авторизації та реєстрації користувачів;

– забезпечення аутентифікації за допомогою JWT-токенів;

– автоматизація процесу оновлення SSL-сертифікатів для забезпечення безпечної передачі даних із використанням алгоритмів шифрування у мажах протоколу прикладного рівня HTTPS;

– надання кінцевих точок, що дозволяють взаємодіяти із системою клієнтським застосункам, передаючи дані у форматі JSON;

– здійснення адміністрування (виконання операцій створення, редагування, перегляду та видалення інформації, створення резервних копій налаштувань та даних і відновлення системи до попереднього стану, керування сертифікатами);

– визначення, чи надіслані пристроєм показники відповідають нормі, у випадку негативної відповіді формування рекомендації для користувача;

– формування статистики показань за певний період часу (день, тиждень, місяць).

Клієнтська частина:

– відображення форм авторизації й реєстрації, передбачивши валідацію введених користувачами даних до відповідних полів;

– локалізація (зміна мови інтерфейсу між українською та англійською);

– інтернаціоналізація (зміна формату дати на графіках при відображенні статистики);

– надання списку поточних пристроїв, зареєстрованих користувачем, видалення їх із системи та зміна назви;

– керування раціонами із автоматичним підрахунком калорійності та вмісту мікроелементів для вказаної маси продукту;

– відображення у зручному вигляді усіх показників здоров’я, що відстежуються, а також їх значення для конкретного користувача на основі даних, отриманих із розумного пристрою;

– відображення статистики показників здоров’я у вигляді графіків та діаграм за вибраний проміжок часу;

– відображення інтерфейсу адміністратора.

Мобільний застосунок;

– відображення екрану авторизації;

– локалізація;

– інтернаціоналізація;

– внесення даних про прийом їжі, базуючись на раціоні, при цьому коригуючи вміст кожного продукту у разі потреби;

– відображення інформації про поточний стан здоров’я користувача системи;

– відображення сповіщень із детальною інформацією про те, який показник здоров’я не у нормі, а також його значення.

Розумний пристрій:

– збір показнків здоров’я;

– надсилання відповідних показників на веб-сервер.

3.1.8 Припущення й залежності

Припущення:

– користувач зацікавлений у отриманні інформації про свій стан здоров’я;

– кожен користувач має розумний пристрій для коректної роботи програмної системи;

– користувачі мають безперебійне з’єднання із мережею Інтернет;

– користувачі зацікавлені у отриманні рекомендацій та оповіщень щодо стану здоров’я, кількості мікроелементів та раціонів.

Залежності:

– від потужності клієнтських комп’ютерів та мобільних пристроїв;

– від точності виміряних розумним пристроєм значень показників здоров’я;

– від обсягу інвестицій;

– від обсягу виготовлених розумних пристроїв.

3.2 Властивості ПЗ

Оцінюючи якість програмного забезпечення, було визначено властивості, за якими і відбувалися розрахунки. Кожна властивість характеризується набором метрик, значення яких за правилами Байєсівських мереж безпосередньо впливають на отриманий результат для кожної із наступних властивостей:

– ефективність: 0,74;

– портативність: 0,93;

– зручність застосування: 0,85;

– супроводжуваність: 1;

– надійність: 0,67;

– функціональність: 1.

3.3 Атрибути ПЗ

3.3.1 Надійність

З метою забезпечення високої надійності програмної системи було вжито ряд заходів. У першу чергу, аби помилки у програмному коді не стали причиною зупинки роботи сервера при виникненні виключних ситуацій, у рамках реалізації наскрізної функціональності повинно бути розроблено потужний механізм обробки виключень. При цьому передбачено ведення логування подібного роду випадків, що дозволить переглянути журнал, проаналізувати потенційно небезпечні ситуації та усунути причини їх виникнення.

При виникненні різноманітних помилок при роботі клієнта із користувацьким інтерфейсом має бути забезпечено виведення інформативних сповіщень про помилки, проте подробиці внутрішнього влаштування системи не повинні висвітлюватися.

3.3.2 Доступність

Програмна система трекінгу балансу мікроелементів та стану здоров’я людини, що займається спортом, повинна бути доступна із будь-якої точки планети 24/7. Це має досягатися завдяки стабільному підключенню усіх складових системи до Інтернету. Важливою умовою є також безперебійність електропостачання.

Для того, аби можна було із упевненістю сказати, що програмний продукт є дійсно доступним, необхідно забезпечити можливість продовження роботи із того місця, де користувач завершив попередній сеанс.

3.3.3 Безпека

Безпечність системи визначається застосуванням протоколу HTTPS, який передбачає шифрування даних, що передаються. Тому на машинах, на яких буде розгорнуто програмний продукт, має бути встановлено SSL-сертифікат, а адміністратор повинен мати можливість у будь-який момент часу здійснити його оновлення. На стороні сервера мають відловлюватися SQL-ін’єкції, що автоматично відбувається при використанні Django ORM.

3.3.4 Супроводжуваність

Система, що підлягає розробці, повинна містити у своєму складі серверну частину, клієнтський веб-застосунок, мобільний застосунок, розумний пристрій. Для усіх модулів кожної із частин мають бути написані unit-тести, покриття коду

якими повинно бути не менш як 65%. Проте використанням лише даного виду тестування не є достатнім. Аби перевірити, яким чином усі перевірені модулі взаємодіють між собою, необхідно провести інтеграційне тестування. Окрім цього, наприкінці розробки кожної робочої версії програмного продукту має проводитися приймальне та димове тестування.

Підтримка проекту у подальшому передбачає дотримання правил написання чистого коду, адже програмний код має гарно читатися, щоб потім можна було швидко збагнути, що саме він виконує.

3.3.5 Переносимість

Написання програмного коду відбувається мовами програмування, що підтримуються усіма сучасними платформами, тому перенесення програм, написаних на Python чи JavaScript, із Windows на Linux чи MacOS не виглядає проблемою. Що стосується Android, то використовується бібліотека підтримки ранніх версій, тому застосунок має коректно працювати більш ніж на 98% пристроїв під управлінням даної ОС. Забезпечується адаптивність інтерфейсу користувача, тому він може підлаштуватися під різні розміри екранів.

Для підтримки переносимості у контексті баз даних використовується Django ORM, що використовує об’єкти Python, потім перетворюючи їх у відповідні сутності обраної бази даних.

3.3.6 Продуктивність

Вимоги до продуктивності програмного продукту постулюють, що користувач повинен отримати відповідь від графічного інтерфейсу застосунку не

довше, ніж за 0,7 с. При цьому під час очікування завантаження необхідно відображати відповідну анімацію, яка б надавала йому упевненості у тому, що система не втратила працездатності і певні дії нею виконуються.

З метою уникнення простою веб-застосунку під час очікування HTTP-відповідей від сервера, необхідно використовувати засоби асинхронного програмування. Це дозволить продовжувати виконувати певну корисну роботу, а коли буде отримано відповідь – обробити її належним чином.

3.4 Вимоги бази даних

У якості системи керування базами даних необхідно використовувати PostgreSQL, проте використання об’єктно-реляційних відображень дозволить абстрагуватися від конкретної бази даних, маніпулюючи класами мови програмування Python. З’єднання і обмін інформацією із БД має відбуватися відповідно до стеку протоколів TCP/IP. Для безпосереднього з’єднання Python-проекту із сервером Postgres повинен використовуватися модуль psycopg2. При з’єднанні вказується рядок, що містить ім’я користувача, хост, ім’я користувача та його пароль.

Усі таблиці бази даних повинні мати ідентифікатор (вводиться сурогатний ключ). Таке рішення значно спрощує вибірку даних при виконанні запитів у межах архітектурного стилю REST, на якому й заснована робота back-end складової.

При використанні пов’язаних таблиць важливо виконувати каскадне видалення. Усі поля таблиць є обов’язковими, тому встановлення обмеження типу NOT NULL не допускається. У базі даних можуть зберігатися дані таких типів, як varchar, integer, float, uuid, datetime, boolean.