

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
УМАНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ САДІВНИЦТВА
Факультет економіки і підприємництва

Кафедра інформаційних технологій

Допущено до захисту

" _____ " _____ 2023 р.

Зав. кафедри, к.т.н., доцент

_____ Лішук Р.І.

КВАЛІФІКАЦІЙНА РОБОТА
МАГІСТРА

Інформаційна система для спостереження за фізіологічним станом
людини

Виконав – студент 2 курсу, групи 21м-кн
освітньої програми "Комп'ютерні науки"
спеціальності 122 Комп'ютерні науки

Лупол Максим Валерійович

Науковий керівник – д.т.н., професор

Кулаков Павло Ігорович

Умань – 2023

Реферат
на кваліфікаційну роботу «Інформаційна система для спостереження за
фізіологічним станом людини»

студента факультету економіки та підприємництва
Уманського національного університету садівництва

Лупола Максима Валерійовича

Структура та обсяг роботи - пояснювальна записка до кваліфікаційної роботи магістра складається з 4 розділів, містить 77 сторінок, 16 рисунків, 5 додатків, використано 42 літературних джерела.

Кваліфікаційна робота магістра присвячена розробці інформаційної системи для спостереження за фізіологічним станом людини. Об'єктом дослідження магістерської роботи є процес моніторингу фізіологічного стану людини. Предметом дослідження є інформаційні системи для реалізації методів контролю енергетичного балансу в організмі людини. Метою роботи є створення інформаційної системи для спостереження за фізіологічним станом людини з метою оцінювання фізичної активності та енергетичних витрат. Наукова новизна отриманих результатів полягає у тому, що дістала подальшого розвитку система самозвіту для обчислення енергетичних витрат організму. Практичне значення одержаних результатів кваліфікаційної роботи магістра полягає в тому, що розроблено зручний мобільний додаток для оцінювання фізичної активності та здійснення контролю енергетичного балансу організму людини.

У першому розділі кваліфікаційної роботи магістра проведено аналітичний огляд предметної області, у другому розділі розроблено архітектуру програмного забезпечення інформаційної системи, у третьому розділі розроблено програмне забезпечення інформаційної системи та розглянуто питання тестування.

Анотація

Кваліфікаційна робота магістра присвячена розробці інформаційної системи для спостереження за фізіологічним станом людини. Проведено аналіз існуючих систем та методів моніторингу фізіологічного стану людини, аналіз готових рішень від різних компаній, доведена актуальність дослідження, проведене обґрунтування розробки, визначені завдання роботи. Практичне значення одержаних результатів кваліфікаційної роботи магістра полягає в тому, що розроблено зручний мобільний додаток для оцінювання фізичної активності та здійснення контролю енергетичного балансу організму людини. Наукова новизна отриманих результатів кваліфікаційної роботи магістра полягає у тому, що дістала подальшого розвитку система самозвіту для обчислення енергетичних витрат організму.

Ключові слова: інформаційна система, фізіологічний стан людини, моніторинг, фізична активність.

Annotation

The master's thesis is devoted to the development of an information system for monitoring the physiological state of a person. An analysis of existing systems and methods of monitoring the physiological state of a person was carried out, an analysis of ready-made solutions from various companies was carried out, the relevance of the research was proven, the justification of the development was carried out, and work tasks were defined. The practical significance of the obtained results of the master's thesis is that a convenient mobile application has been developed for assessing physical activity and monitoring the energy balance of the human body. The scientific novelty of the obtained results of the master's thesis is that the self-report system for calculating the body's energy expenditure has received further development.

Keywords: information system, physiological state of a person, monitoring, physical activity.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
УМАНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ САДІВНИЦТВА

Факультет економіки і підприємництва

Кафедра Інформаційних технологій

Освітня програма Комп'ютерні науки

Спеціальність 122 Комп'ютерні науки

(код, назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри

Ліщук Р.І.

(підпис)

(ініціали, прізвище)

" _____ " _____ 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Студенту Луполу Максиму Валерійовичу

(прізвище, ім'я, по батькові)

1. Тема проекту Інформаційна система для спостереження за
фізіологічним станом людини

керівник роботи Кулаков Павло Ігорович, д.т.н., професор кафедри
інформаційних технологій

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від "28" квітня 2023 року № 01-06/77

2. Строк подання студентом роботи 20.12.2023

3. Вихідні дані до роботи Розробити мобільний додаток інформаційної
системи для спостереження за фізіологічним станом людини, додаток
призначений для операційної системи Android, додаток створити на об'єктно-
орієнтованій мові програмування Java. Додаток повинен забезпечувати
моніторинг фізичної активності та енергетичних витрат людини.

4. Перелік питань, які потрібно розробити Аналітичний огляд предметної
області, розробка архітектури програмного забезпечення інформаційної
системи, розробка програмного забезпечення інформаційної системи,
тестування мобільного застосунку.

5. Перелік графічного матеріалу 16 рисунків, 5 додатків, 42 джерела

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Строк виконання етапів роботи	Примітка
1.	Вибір теми кваліфікаційної роботи	05.09.2022	
2.	Одержання індивідуального завдання	07.09.2022	
3.	Складання календарного плану написання кваліфікаційної роботи	08.09.2022	
4.	Підготовка до виконання кваліфікаційної роботи - підбір та вивчення літератури - участь у виконанні науково-дослідних робіт - виконання курсових робіт - інші заходи	Впродовж I семестру	
5.	Уточнення теми кваліфікаційної роботи та календарного плану-графіка, виходячи зі специфіки базового підприємства, установи	До початку II семестру	
6.	Підготовка: - першого розділу - другого розділу - третього розділу - висновків	09.02.2023 23.03.2023 26.04.2023 10.05.2023	
7.	Доопрацювання кваліфікаційної роботи з урахуванням зауважень наукового керівника	18.05.2023	
8.	Написання та оформлення роботи в остаточному варіанті	25.05.2023	
9.	Перевірка роботи на академічний плагіат	12.06.2023	
10.	Одержання відгуку наукового керівника	13.06.2023	
11.	Попередній захист роботи на кафедрі	14.06.2023	
12.	Затвердження роботи завідувачем кафедри	15.06.2023	
13.	Одержання рецензії	16.06.2023	
14.	Захист кваліфікаційної роботи	19-21.06.2023	

Студент _____ **Лупол М.В.** _____
(підпис) (прізвище та ініціали)

Керівник роботи _____ **Кулаков П.І.** _____
(підпис) (прізвище та ініціали)

Зміст

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ.....	7
ВСТУП.....	8
РОЗДІЛ 1. АНАЛІТИЧНИЙ ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ.....	10
1.1. Сучасні інформаційні технології у медичній галузі	10
1.2. Огляд та аналіз відомих аналогів.....	19
1.3. Постановка задачі.....	30
РОЗДІЛ 2. АРХІТЕКТУРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	
ІНФОРМАЦІЙНОЇ СИСТЕМИ.....	32
2.1. Розробка життєвих циклів	32
2.2. Системні компоненти.....	34
2.3. Перетворення вихідного коду в програму Android	39
РОЗДІЛ 3. РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	
ІНФОРМАЦІЙНОЇ СИСТЕМИ.....	41
3.1. Програмна реалізація основного функціоналу застосунку	41
3.2. Тестування застосунку	46
3.3. Інструкція користувача	47
ВИСНОВОК.....	53
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ.....	55
ДОДАТОК А Схема компонентів застосунку.....	59
ДОДАТОК Б Лістинг DiaryActivity.java.....	60
ДОДАТОК В Лістинг activity_dish.xml	68
ДОДАТОК Г Лістинг Gradle Scripts.....	71
ДОДАТОК Д Лістинг android_calorie_counter.iml.....	73

Перелік умовних позначень, скорочень і термінів

ІТ	Інформаційні технології.
ІС	Інформаційна система.
БД	База даних.
ПЗ	Програмне забезпечення.
ПД	Програмний додаток.
СУБД	Система управління базами даних.

Вступ

Однією з найбільш затребуваних по всьому світу областю є сфера розробки різноманітних інформаційних систем. Щорічно збільшується число додатків, створених безпосередньо для різних платформ. Ринок інформаційних систем неухильно зростає. Наявність програми для смартфонів та комп'ютерів значно підвищує конкурентоспроможність тієї чи іншої сучасної організації на ринку через поширеність використання гаджетів. Досвід використання на практиці показує, що якісні додатки значно зручніше мобільної версії сайту компанії.

Актуальність теми дослідження обумовлена тим, що із поширенням небезпечного вірусу COVID-19 та інших небезпечних хвороб люди почали дельніше вивчати питання охорони власного здоров'я. Через дані події виріс попит на різні системи для моніторингу здоров'я, додатки для відстежування прогресу в різних видах та все, що пов'язане із зміцненням імунітету. З кожним днем вчені відкривають нові методи запобігання різних хвороб. Для ефективного лікування недугів потрібно знати детальну інформацію про пацієнта та його фізичний або фізіологічний стан. Інформаційні технології допомагають лікарю проводити діагностику захворювань, накопичувати і ефективно використовувати отриману інформацію на всіх стадіях процесу лікування.

Об'єкт дослідження кваліфікаційної роботи магістра - процес моніторингу фізіологічного стану людини.

Предмет дослідження кваліфікаційної роботи магістра – інформаційні системи для реалізації контролю енергетичного балансу в організмі людини.

Метою кваліфікаційної роботи магістра є створення інформаційної системи для спостереження за фізіологічним станом людини з метою розширення переліку вихідних даних для оцінювання фізичної активності та енергетичних витрат, що надає можливість більш ефективно підтримувати енергетичний баланс, запобігаючи надлишковій вазі та пов'язаним з нею захворюваннями.

Задачі дослідження:

- проаналізувати актуальні методи моніторингу стану людини та готові рішення, що представлені на ринку;
- на основі проведених досліджень розробити зручну та інтуїтивну інформаційну систему на доступній для будь-якого користувача операційній системі для моніторингу фізіологічного стану людини.
- налаштування ефективної комунікації між пацієнтом та медичним персоналом.

Методи дослідження:

- системний аналіз;
- методи вимірювання фізичної активності;
- об'єктно-орієнтовані методи проектування.

Наукова новизна отриманих результатів полягає у тому, що дістала подальшого розвитку система самозвіту для обчислення енергетичних витрат організму, в якій на відміну від уснуючої використовується більш гнучка система вихідних параметрів, що надає можливість враховувати індивідуальні особливості при контролі фізичної активності.

Практичне значення одержаних результатів полягає в тому, що розроблено простий та зручний у використанні мобільний додаток для операційної системи Android для оцінювання фізичної активності та здійснення контролю енергетичного балансу організму людини.

РОЗДІЛ 1. АНАЛІТИЧНИЙ ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Сучасні інформаційні технології у медичній галузі

Медицина завжди намагалася виміряти різні показники пацієнтів (наприклад, клінічну картину), щоб детально вивчити їх здоров'я та хвороби. Потім інформатика включатиме результати цих вимірювань, експериментів і спостережень у моделі. Останнім часом використання ІТ-інструментів для медицини (таких як машинне навчання) зазнало прискореного зростання, однак усі моделі такої системи є неповними без реальних даних, особливо на основі реєстрів.

Наприклад, дані електронної медичної картки/документації пацієнта (EHR/D) можуть бути надані з достатньою точністю щодо клінічної картини, процедур або супутніх захворювань. Міжнародний стандарт — HL7 (Health Level Seven) забезпечує взаємодію між ІТ-провайдерами та рішеннями. Однак у багатьох країнах більшість медичних даних є аналоговими і їх необхідно оцифрувати (наприклад, обробка зображення та звуку). Крім того, потужні комп'ютерні засоби та переносні пристрої з кількома датчиками зробили можливим збір, обробку та зберігання даних про окремих пацієнтів або групу пацієнтів. Індивідуальні дані можуть бути використані для оцінки ризику певного розладу здоров'я. Наприклад, ризик захворіти інфекційним захворюванням можна розрахувати на основі даних датчиків і анкет. Дані, зібрані від групи потенційно контактних пацієнтів, можна використовувати для оцінки ризику інфекційних захворювань у певній громаді.

Існує занепокоєння щодо Закону про захист даних в ЄС з юридичної точки зору щодо надання такого роду аналізу даних. Потрібно запитувати згоду кожного пацієнта, що може бути дуже важко отримати [1-4].

Основним застосуванням Blockchain в медицині є відстеження ліків у багатьох країнах. Така методика вже застосовувалася в Африці та Південно-Східній Азії для перевірки оригінальності ліків (підроблені ліки там є великою проблемою). Технологія Blockchain також може допомогти у розслідуванні

спалахів, пов'язаних із їжею, де обробка товарів у ланцюжку поставок із характеристиками зберігання або транспортування має вирішальне значення [5]. Об'єднання даних (ОБ) — це багатодоменна галузь, що розвивається, і спрямована на надання даних для розуміння ситуації. У глобальному масштабі виявлення загроз і захист об'єктів є одними з життєвоважливих сфер досліджень ОБ. Системи Fusion мають на меті інтегрувати дані та інформацію датчиків у бази даних, бази знань, контекстну інформацію, завдання користувача тощо, щоб описати ситуації, що динамічно змінюються. У певному сенсі мета злиття інформації полягає в тому, щоб досягти моделювання в реальному часі підмножини світу на основі часткових спостережень за ним.

Здатність об'єднати цифрові дані в корисну інформацію ускладнюється тим фактом, що вхідні дані, отримані з пристрою чи текстові, генеруються в різних форматах, деякі з них неструктуровані. Незалежно від того, чи є інформація неструктурованою за своєю природою, чи відсутні стандарти обміну інформацією (метадані або вони не дотримуються), усе це перешкоджає автоматичній обробці комп'ютерами. Крім того, дані, які потрібно об'єднати, можуть бути неточними, неповними, неоднозначними або суперечливими; воно може бути фальшивим або зіпсованим ворожими заходами. Крім того, велику кількість інформації може бути важко формалізувати, тобто інформацію про зображення. Отже, обмін інформацією часто надто складний. У багатьох випадках існує брак зв'язку між різними джерелами інформації просто тому, що немає механізму для підтримки цього обміну [6-10].

Великий потік даних, як на пристрої, так і на основі тексту, не в змозі обробити і призводить до затримок у часі, додаткових витрат і навіть до невідповідних рішень через відсутність або неповну інформацію.

Ключовим аспектом у сучасних додатках ОБ є відповідна інтеграція всіх типів інформації або знань: даних спостережень, моделей знань (апріорних або індуктивно-засвоєних) і контекстної інформації. Кожна з цих категорій має відмінну природу та потенційну підтримку результату процесу синтезу.

Дані спостережень — це фундаментальні дані про людину, зібрані за допомогою певних засобів спостереження (датчиків будь-якого типу). Ці дані стосуються спостережуваних характеристик людини, які представляють інтерес. Контекст і елементи того, що можна назвати контекстною інформацією, можна визначити як «сукупність обставин, що оточують отримані дані, які потенційно мають значення для їх завершення». Через релевантність даних термоядерний синтез передбачає розробку найкращої можливої оцінки з урахуванням цих сторонніх знань. Ми можемо розглядати контекст як фон, тобто не конкретну сутність, подію чи поведінку, що становить головний інтерес, а ту інформацію, яка впливає на формування найкращої оцінки цих елементів.

Набуті знання: у тих випадках, коли апріорні знання для розробки процесу ОБ не можуть бути сформовані, однією з можливостей є спроба вирізати знання за допомогою онлайн-процесів машинного навчання, що працюють на даних спостережень та інших даних. Це процедурні та алгоритмічні методи для виявлення взаємозв'язків між поняттями, що представляють інтерес, які фіксуються системою (дані датчиків і контекстна інформація). Існує компроміс, пов'язаний із спробою розробки повністю автоматизованих алгоритмічних процесів ОБ для складних проблем, де включення людського інтелекту в певний момент процесу може бути набагато більш розумним вибором.

На даний момент існує безліч проблемно-специфічних рішень термоядерного синтезу, більшість з яких зосереджені на конкретних програмах, що дає специфічні для проблеми результати. Можливість автоматичного об'єднання результатів менших проблем у більший контекст досі відсутня.

Електронні пристрої з низькою потужністю та недорогою ціною можна легко придбати та використовувати. Деякі властивості, такі як температура тіла та пульс, можна виміряти з високою точністю, але розробники намагаються націлюватися на все більше функцій. FDA (Федеральне агентство з лікарських засобів), а також ЕМА (Європейське агентство з лікарських засобів) сертифікують цифрове програмне забезпечення та пристрої для охорони

здоров'я. Більшість пристроїв на ринку не задовольняють мінімальним умовам сертифікації та акредитації. Деякі з них як пристрої для моніторингу якості повітря (наприклад, лазерний датчик пилу PM2.5 менш ніж за 5 доларів) масово використовуються в сильно забруднених містах. Однак жодна медична організація не рекомендує використовувати домашні пристрої через дуже низьку якість даних, і даних, наданих сертифікованими офіційними станціями, має бути достатньо. Більше даних не завжди означає краще, як це показано в цьому прикладі. Деякі проблеми виникають із додатками та програмним забезпеченням для цифрового здоров'я (mhealth). Сьогодні майже для всього можна знайти програму.

Розглянемо прогрес у розпізнаванні зображень за допомогою глибокого навчання смартфоном для вимірювання артеріального тиску за допомогою фотоплетизмографії. Найбільшою перевагою цього методу є простота, тому нічого крім смартфона не потрібно. Однак навіть після калібрування він працює погано для більшості користувачів, є пацієнти, для яких похибка цієї процедури може досягати 30%. Тому програми, що пропонують цю функцію (як-от iCare і HDCH), було вилучено з Google Play і App Store, але їхні еквіваленти все ще дуже популярні, наприклад, у Китаї.

Аналіз і виявлення проблеми зі здоров'ям можна виконувати на мобільному пристрої (телефоні, годиннику, браслеті тощо), використовуючи принципи периферійних обчислень, або в сусідній обчислювальній інфраструктурі. Об'єднання, обробка та аналітика даних датчиків, зібраних з персональних мобільних пристроїв і пристроїв охорони здоров'я, а також інфраструктури розумного міста виконуються на обчислювальних компонентах Edge/Fog, що забезпечує ефективність і мінімальну затримку при виявленні критичних медичних станів, які вимагають оперативних дій. Крім того, це може забезпечити персоналізовану систему охорони здоров'я для загального благополуччя, де люди можуть отримати медичне обслуговування, адаптоване до їхніх потреб [11, 12]. Крім того, така система повинна підтримувати

ефективний збір і аналіз величезної кількості неоднорідних і безперервних даних (Big Data) про здоров'я та діяльність від групи або натовпу користувачів. Зберігання, агрегація, обробка та аналіз великих даних про здоров'я можуть виконуватися в публічній, приватній або гібридній хмарній інфраструктурі. Результати аналізу та видобутку даних надаються лікарям, медичним працівникам, медичним організаціям, фармацевтичним компаніям за допомогою налаштованої візуальної аналітики та додатків на інформаційній панелі. Перед ширшим застосуванням таких персоналізованих систем охорони здоров'я, які безперервно контролюють здоров'я та діяльність людей і належним чином реагують на критичні події та умови, необхідно вирішити низку проблем у сфері досліджень і розробок.

Загальна архітектура великомасштабної розподіленої системи на основі медичних пристроїв IoT та обробки та аналітики Big Data для дистанційного моніторингу здоров'я та діяльності людей наведена на рис. 1.1.

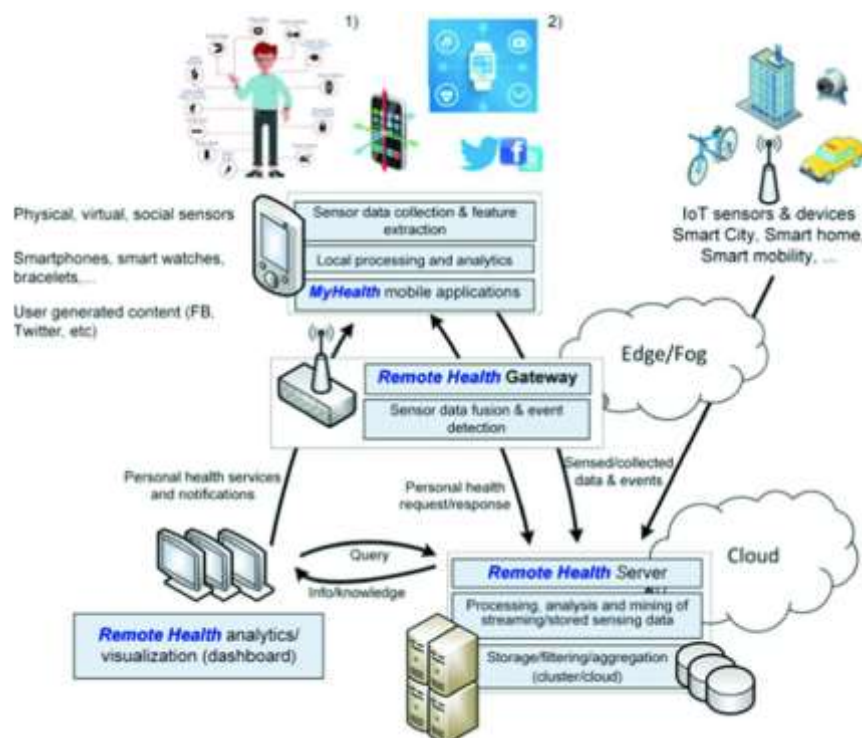


Рисунок 1.1 — Загальна архітектура системи дистанційного моніторингу стану здоров'я та моніторингу людей

Системи дистанційного моніторингу можуть працювати в різних масштабах, забезпечуючи персональне та глобальне зондування. Існує три різні шкали для зондування: особисте, групове та спільнотне. Персональні або індивідуальні системи моніторингу розроблені для окремої особи та часто зосереджені на зборі та аналізі персональних даних. Типові сценарії включають відстеження режиму фізичних вправ користувачів, вимірювання рівня активності або виявлення симптомів, пов'язаних із психологічними розладами. Хоча зібрані дані можуть бути призначені виключно для використання користувачем, обмін ними з медичним працівником є звичайним явищем. Популярними технологіями персонального моніторингу є Google Glass, FitBit і The Nike+ FuelBand [13 - 18].

Якщо окремі особи поділяють спільні інтереси, занепокоєння чи цілі під час участі в програмах моніторингу, вони утворюють групу. Системи групового моніторингу можуть бути популярними в соціальних мережах або підключених групах, де дані можуть вільно ділитися або бути із захистом конфіденційності. Типовими прикладами є додатки для моніторингу здоров'я, включаючи змагання, пов'язані з конкретними цілями: бігова дистанція, втрата ваги, споживання калорій тощо. Якщо кількість людей, які беруть участь у моніторингу здоров'я та активності, є дуже великою, це називається моніторингом спільноти або натовпу. Моделювання та моніторинг натовпу передбачає колективну аналітику даних на благо спільноти. Однак це передбачає співпрацю осіб, які не довірятимуть один одному, підкреслюючи необхідність надійного захисту конфіденційності та, можливо, низький рівень зобов'язань з боку користувачів. Приклади моніторингу спільноти включають відстеження поширення захворювань у регіоні, пошук закономірностей для конкретних захворювань тощо. Вплив масштабування на програми моніторингу ще буде досліджено, але багато дослідницьких проблем пов'язані з обміном інформацією, правом власності на дані, об'єднанням даних, безпекою та

конфіденційність, алгоритми, які використовуються для інтелектуального аналізу даних, надання корисних відгуків, залишаються відкритими.

Усі організації, які збирають, обробляють і зберігають медичні дані, зобов'язані удосконалювати свої стратегії захисту даних відповідно до норм і стандартів міжнародного та місцевого законодавства. Прикладом законів про захист даних, які вимагають США, є Закон про перенесення та підзвітність медичного страхування (HIPAA), примітка 1, яка використовується для відповідності та безпечного впровадження електронних медичних записів (EHR). HIPAA встановлює правила захисту конфіденційних даних пацієнтів у своїх Правилах конфіденційності та безпеки. Згідно з HIPAA, компанія, яка обробляє захищену інформацію про здоров'я, повинна зберігати її в безпечній системі, що стосується фізичної безпеки, безпеки мережі та процесу. Правило безпеки HIPAA визначає вимоги до медичної інформації, яка зберігається або передається в електронній формі. Воно вимагає, щоб організації, на які поширюється дія HIPAA, запроваджували такі засоби захисту:

- забезпечення конфіденційності, цілісності та доступності усіх електронних даних, які вони створюють, отримують, зберігають або передають;
- визначення та захист від очікуваних загроз безпеці або цілісності інформації;
- захист від очікуваного, недозволеного використання або розголошення;
- забезпечення дотримання вимог їхньою робочою силою.

Крім того, воно регулює технічні заходи безпеки під час розміщення конфіденційних даних пацієнтів, включаючи доступ до закладу на місці, а також політику щодо використання робочих станцій, електронних носіїв і всіх процесів передачі, видалення, утилізації та повторного використання електронних носіїв або електронної інформації. Воно забезпечує дотримання авторизованих унікальних ідентифікаторів користувачів, процедур екстреного доступу, автоматичного виходу з системи, шифрування даних і звітів про аудит. Крім

того, необхідні журнали відстеження всіх дій на апаратному та програмному забезпеченні [19 - 20].

Організації охорони здоров'я повинні забезпечити безпеку та доступність даних пацієнтів як для медичних працівників, так і для пацієнтів. Примітка 2 до Загального регламенту захисту даних (GDPR) - це положення законодавства ЄС щодо захисту даних і конфіденційності в межах Європейського Союзу (ЄС) і Європейської економічної зони (ЄЕЗ). Воно також визначає умови експорту персональних даних за межі ЄС та ЄЕЗ. У ньому зазначено:

- права суб'єкта даних;
- роль контролера;
- передача персональних даних третім країнам або міжнародним організаціям;
- незалежні наглядові органи;
- співпраця та послідовність;
- засоби правового захисту, відповідальність і штрафи;
- делеговані акти та імплементаційні акти.

Воно визначає дії, які необхідно вжити для забезпечення обробки в системах, які обробляють медичні дані. Кілька прикладів можуть бути:

- у разі порушення власники даних повинні повідомити органи влади;
- суб'єкти мають право на доступ до своїх даних;
- право бути забутим, яке дає людям право вимагати видалення їх персональних даних;
- конфіденційність включено за замовчуванням;

Право на виправлення даних має право отримати від контролера без невинуватної затримки. Суб'єкт даних має право на доповнення неповних персональних даних, у тому числі шляхом надання додаткової заяви. Кожна організація, яка володіє персональними даними третіх сторін, повинна мати уповноваженого із захисту даних. GDPR накладає суворі штрафи на контролерів і обробників даних за недотримання вимог у розмірі до 20 мільйонів, або 4%

світового річного доходу за попередній фінансовий рік, у разі порушення основних принципів обробки медичних даних.

Щоб отримати необхідний рівень безпеки HIPAA, компанія може піти двома шляхами:

- вибрати авторитетну навчальну компанію HIPAA, яка пропонує сертифікати сертифікації, і самостійно проаналізувати процедури та системи компанії;
- отримати оцінку від незалежного стороннього аудитора, наприклад, Coalfire Systems^{Footnote3} або ComplySmart^{Footnote4}.

Для забезпечення стандартів GRPD компанія може:

- отримати індивідуальний сертифікат GDPR для офіцерів із захисту;
- найняти уповноважених експертів.

Щодо відповідності вимогам HIPAA та GDPR, немає жодної центральної незалежної сторони, яка може сертифікувати компанію. Особливі проблеми GDPR виникають у віддалених системах. По-перше, відповідно до GDPR персональні дані не можуть зберігатися довше, ніж це необхідно. Ось чому необхідно запровадити процедури зберігання, а коли термін дії даних закінчився, їх потрібно видалити із систем. Дані можуть зберігатися в кількох місцях, у кількох постачальників, оброблятися багатьма службами. Повне видалення даних також має враховувати резервні копії та копії даних, що зберігаються на віддаленому обладнанні. Крім того, проблемою може бути реакція на порушення. Повідомлення про порушення мають надаватися власникам даних. Подія порушення має бути чітко визначена. Обробка персональних даних за межами Європейської економічної зони (ЄЕЗ) є наступною проблемою через кілька місць. Контролерам потрібно буде визначити стратегію даних для кількох країн, враховуючи закони про локалізацію. Прозорість процедур безпеки медичних систем або сертифікати третьої сторони необхідні для того, щоб переконати контролерів в якості послуг безпеки. Постачальники медичних послуг повинні підлягати аудиту, щоб забезпечити систему контролю з

конфіденційністю та конфіденційністю за проектом. Слід контролювати медичні послуги, щоб урахувати будь-які зміни в технології та рекомендовані оновлення системи. Вони включають нове обладнання та датчики.

Під час обробки великого набору даних видимість щодо метаданих та анонімізація даних є великою проблемою. Необхідно ретельно перевірити рівень захисту метаданих, відповідні права власності, права на обробку колекцій метаданих, цільове використання метаданих. МОЗ визначає фізичну активність як довільний рух тіла, що виробляється скелетними м'язами, що потребує витрат енергії. Фізична активність стосується будь-якого руху, у навіть під час дозвілля, під час поїздки транспортом, щоб дістатися до місця та назад, або як частина роботи людини. Фізична активність як середньої, так і високої інтенсивності покращує здоров'я [21].

Популярні способи бути фізично активними включають ходьбу, їзду на велосипеді, катання на колесах, спорт, активний відпочинок на природі та ігри, і це можна робити на будь-якому рівні навичок. Науково доведено, що періодична фізична активність допомагає запобігти та лікувати неінфекційні захворювання, такі як діабет, інсульт, хвороби серця та деякі види раку. Це також допомагає підтримувати здорову вагу, запобігти гіпертонії та може покращити психічне здоров'я, якість життя та самопочуття.

1.2 Огляд та аналіз відомих аналогів

Фізична активність визначається як будь-який рух тіла, вироблений скелетними м'язами, який призводить до витрати енергії. Переваги фізичної активності для підтримки здоров'я були добре задокументовані, особливо в профілактиці та лікуванні хронічних захворювань. Тому точне вимірювання фізичної активності та витрат енергії є важливим як для епідеміологічних досліджень, так і в клінічному контексті. Враховуючи велику кількість доступних методів, важливо мати розуміння кожного, особливо коли потрібно вибрати техніку для використання.

Переваги фізичної активності для підтримки здоров'я були добре задокументовані, особливо в профілактиці та лікуванні хронічних захворювань, таких як деякі види раку, діабет 2 типу та серцево-судинні захворювання. У цьому контексті точне вимірювання фізичної активності та енерговитрат є важливим як в епідеміологічних дослідженнях, так і в оцінці ефективності програм втручання. У клінічних умовах оцінка енергетичних витрат дозволяє оцінити потреби пацієнтів у харчових речовинах під час нутритивної підтримки. Фізична активність визначається як будь-який рух тіла, вироблений скелетними м'язами, що призводить до витрат енергії. Важливо підкреслити, що фізична активність і енерговитрати — це 2 різні поняття. Простіше кажучи, фізична активність — це поведінка, яка призводить до підвищення витрат енергії над рівнем спокою. Загальні витрати енергії (TEE) відносяться до загальної кількості енергії, витраченої протягом 24-годинного періоду, і містять 3 основні компоненти: витрати енергії в стані спокою (REE), тепловий ефект їжі (TEF) і витрати енергії під час діяльності (AEE).

REE, найбільша частина TEE, є енергією, необхідною для підтримки основної метаболічної активності, включаючи підтримку температури тіла та підтримку функціонування життєвоважливих органів, таких як мозок, нирки, серце та легені. REE визначається як енергія, витрачена людиною, що голодує, у стані спокою в термонеутральному середовищі. Фактори, що найбільш суттєво впливають на REE, включають склад тіла, стать, температуру тіла, вік, обмеження енергії, а також генетичну та ендокринну систему. Склад тіла: знежирена маса (також звана нежировою масою тіла) є основним визначальним фактором REE, тобто людина з високою знежиреною масою має вищий REE. Стать: REE має тенденцію бути вищим у чоловіків, ніж у жінок, і це може бути пов'язано з вищим відсотком нежирової маси тіла у чоловіків порівняно з жінками. Вік: люди похилого віку мають нижчий REE порівняно з молодими. Було досліджено, що пов'язане з віком зниження REE не залежить від змін у складі тіла, що свідчить про те, що також можуть бути залучені додаткові

метаболическі зміни. Енергетичне обмеження: спроби схуднути шляхом обмеження споживання енергії призводять до зниження REE. REE вимірюється, коли людина, що голодує, відпочиває в комфортних умовах. Тривалість голодування зазвичай становить від 2 до 4 годин. REE трохи вище (приблизно 10%), ніж базові витрати енергії (BEE), що є найнижчими енерговитратами людини, і вимірювання BEE вимагає більш суворих умов. BEE людини визначається, коли людина перебуває в постабсорбційному стані (тобто не приймає їжу принаймні 12 годин), лежить (на спині) і повністю розслаблена (нерухомий стан) — бажано дуже швидко після пробудження від сну в ранок. AEE є найбільш варіабельним серед компонентів TEE, як на внутрішньому, так і на міжособистісному рівні. У людей, які ведуть сидячий спосіб життя, він може становити менше половини AEE, тоді як у дуже активних людей, таких як деякі спортсмени або важкі працівники, він може перевищувати AEE в 2 рази або більше.

TEF, також відомий як індукований дією термогенез (DIT) — це енергія, необхідна для перетравлення, всмоктування, транспортування та метаболізму їжі, зберігання поживних речовин і видалення відходів. Він являє собою збільшення витрат енергії вище REE, яке можна виміряти протягом кількох годин після їжі. TEF оцінюється приблизно як 10% добової TEE.

Техніка прямої калориметрії вимірює швидкість втрати тепла суб'єктом за допомогою калориметра. Це найточніший метод кількісної оцінки швидкості метаболізму, але його використання обмежене високою вартістю. Існує 4 типи прямих калориметрів, а саме «ізоترمичні прямі калориметри» (також відомі як «калориметри теплового потоку або теплопровідності»), які працюють, підтримуючи постійну температуру стінки за допомогою рідини постійної температури (зазвичай вода) в сорочці або ванні, що оточує камеру для тварин, або в мережі мідних трубок, прикріплених до зовнішньої поверхні стіни; «прямі калориметри з радіатором», які складаються з камери, з якої тепло, втрачене суб'єктом, видаляється за допомогою теплообмінника з рідинним

оохолодженням, «калориметри прямої конвекції», які складаються з ізолюваної камери, вентиляованої потоком повітря з відомою швидкістю. Система працює шляхом визначення різниці температур і ентальпії між повітрям, що входить і виходить із ізолюваної камери, і «прямі диференціальні калориметри», які включають 2 ідентичні камери, одна з яких містить суб'єкта, а інша має електричний нагрівач, налаштований на однакове підвищення температури в обох камерах.

Останні досягнення в технології дозволили розробити акселерометри як один із методів вимірювання енергії та витрат фізичної активності. Ці інструменти виявилися надійними, об'єктивними, менш обтяжливими для учасників, універсальними та менш дорогими порівняно з іншими методами оцінки енергії фізичної активності. Сьогодні різні моделі акселерометрів розроблені різними компаніями та доступні на ринку. Для моделей акселерометрів Actigraph і Actical з часом були розроблені різні покоління, як описано в дослідженні Джона та Фрідсона.

Цей метод заснований на вимірюванні прискорення тіла, яке є зміною швидкості в часі і виражається через кратні сили тяжіння. Деякі моделі вимірюють прискорення в 1 площині (одноосьовий), 2 площинах (двовісний) або 3 площинах (триосьовий акселерометр). Акселерометри генерують вихідні дані у формі «відліків» за одиницю часу. Щоб перетворити ці підрахунки в одиниці витрат енергії, були розроблені прогнозні рівняння. Деякі з рівнянь дозволяють розрахувати витрати енергії як метаболічні еквіваленти (MET) або ккал/хв.

Дослідження перевірки цих рівнянь були проведені за допомогою непрямой калориметрії або методу DLW. У недавньому дослідженні Lyden et al. Оцінили точність загальноновживаних рівнянь акселерометра для прогнозування витрат енергії та MET. Їх висновки показали, що поточні рівняння прогнозування акселерометра мають багато обмежень при перекладі показників акселерометра на витрати енергії, особливо при використанні в ряді різноманітних видів діяльності.

Першими розробленими моделями Actigraph є одноосьовий акселерометр, а нещодавно були розроблені нові моделі, які вимірюють прискорення в 3 площинах (триосьові акселерометри). У цих останніх моделях витрати енергії можуть бути розраховані шляхом інтеграції відліків у трьох площинах, а саме вертикальній (Y), горизонтальній праворуч-ліворуч (X) і горизонтальній передньо-задній осі (Z). У цьому випадку витрати енергії передбачають так звану векторну величину. Одне з питань, яке виникло у зв'язку з розробкою нових поколінь акселерометрів, — це порівнянність їх продуктивності з результатами попередніх поколінь. Було показано, що існує порівняння між одновісним і тривісним поколіннями однієї моделі акселерометра, коли використовуються відліки від вертикальної осі, але це не стосується випадку використання відліків VM від тривісного акселерометра.

Монітори серцевого ритму є одними з широко використовуваних об'єктивних інструментів для вимірювання фізичної активності та витрат енергії. Їх використання ґрунтується на передбачуваному зв'язку між частотою серцевих скорочень, інтенсивністю активності та споживанням кисню, оскільки фізична активність змушує серце доставляти більше кисню до м'язових клітин, що тренуються. Дослідження встановили, що частота серцевих скорочень змінюється пропорційно інтенсивності діяльності та споживанню кисню під час фізичних навантажень середньої та інтенсивної. Однак ця кореляція є низькою у випадку сидячої та легкої діяльності, і це залишається одним із обмежень моніторів серцевого ритму. На зв'язок між частотою серцевих скорочень і VO_2 впливають такі фактори, як специфічне використання м'язової маси, тип діяльності, рівень фізичної підготовки та інші фактори, пов'язані з фізичними вправами. Також повідомлялося, що деякі фактори, пов'язані з фізичними навантаженнями, впливають на зв'язок між частотою серцевих скорочень і VO_2 . Дослідження повідомляють про електричні або магнітні перешкоди вимірюванню частоти серцевих скорочень такими пристроями, як комп'ютери, мікрохвильові печі, телевізори та моторизоване тренажерне обладнання, що

призводить до нестабільних показань і втрати даних. Пропозиції щодо зменшення цієї проблеми електричних перешкод включають утримання монітора серцевого ритму на відстані щонайменше 1 метра від електричного кола електричних пристроїв і розміщення приймача частоти серцевих скорочень близько до передавача, таким чином посилюючи сигнал передавача відносно сигналу перешкод. Незважаючи на ці обмеження, монітори серцевого ритму залишаються популярними серед дослідників, оскільки вони мають переваги відносно низької вартості, неінвазивності та універсальності. Їх використання дає об'єктивну та достовірну інформацію про енерговитрати та інтенсивність і тривалість діяльності. Різні дослідження показали ефективність цих інструментів у контрольованих умовах, а також у вільних умовах життя.

Крокоміри використовуються для вимірювання ходьби, яка є однією з найбільш часто виконуваних видів діяльності та сприяє значній частці фізичного АЕЕ в анкетах і журналах фізичної активності. Основний вихід крокомірів – це підрахунок кроків. На додаток до цього можна використовувати крокоміри для оцінки пройденої відстані шляхом множення кількості кроків на довжину кроку. На довжину кроку впливають такі фактори, як швидкість ходьби, зріст, вік і стать. Деякі крокоміри відображають витрати енергії як кілокалорій, але неясно, чи вони відображають валові чи чисті кілокалорій. Було оцінено придатність 10 електронних крокомірів для вимірювання кроків, відстані та витрат енергії та виявили, що ці інструменти є найточнішими для оцінки кроків, менш точними для оцінки відстані та ще менш точними для оцінки кілокалорій. Незважаючи на їх низьку точність, крокоміри мають перевагу низької вартості, і їх можна використовувати як інструменти самоконтролю для людей, які хочуть підтримувати свій PAL.

Найбільш широко використовувані методи самозвіту для оцінки фізичної активності включають такі методи, як анкети щодо фізичної активності, записи та щоденники фізичної активності. Вони використовувалися у великих когортних дослідженнях, які дозволили встановити захисну роль фізичної

активності проти захворювань, включаючи розлади, пов'язані з метаболічним синдромом (резистентність до інсуліну, діабет 2 типу, дисліпідемія, гіпертонія та ожиріння), серцеві та легеневі захворювання (хронічна обструктивна хвороба захворювання легень, ішемічна хвороба серця, хронічна серцева недостатність і переміжна кульгавість), захворювання м'язів, кісток і суглобів (остеоартрит, ревматоїдний артрит, остеопороз, фіброміалгія та синдром хронічної втоми), рак, депресія, астма та діабет 1 типу. Обмеження цих самозвітних методів оцінки фізичної активності полягає в їх низькій точності та надійності, як показано різними дослідженнями валідації з використанням об'єктивних показників фізичної активності або витрат енергії, таких як метод DLW та акселерометри. Проте вони доступніші за рахунок низької вартості порівняно з іншими методами оцінки фізичної активності. Вони надають інформацію про моделі фізичної активності учасників.

Найбільш часто використовувані опитувальники фізичної активності включають Міжнародний опитувальник фізичної активності (IPAQ), 7-денний опитувальник фізичної активності (PAR), опитувальник модифікованої активності (MAQ), попередній тиждень Анкета модифікованої активності (PWMAQ), Анкета недавньої фізичної активності (RPAQ) і Згадування фізичної активності минулого дня (PDPAR). Ці опитувальники становлять низький тягар для суб'єктів, але вони обмежені тим, що вони покладаються на пам'ять учасника про виконану фізичну активність, збільшуючи ризик зміщення пам'яті. Оскільки записи фізичної активності та щоденники вимагають від учасників запису виконаної діяльності, вони мінімізують зміщення пам'яті у випадку, якщо вони виконані вчасно. Однак навантаження на учасників є вищим порівняно з анкетами фізичної активності, а затримка запису може призвести до упередженості пам'яті та реактивності.

Проаналізуємо платформи та мобільні додатки для моніторингу фізіологічного стану людини.

Thingier.io — це платформа IoT, яка є альтернативою з відкритим кодом. Thingier.io є відносно новим для екосистеми IoT, але широко використовується в різних дослідницьких проектах або навіть для освіти. Вона надає готовий до використання хмарний сервіс для підключення пристроїв до Інтернету для виконання будь-якого дистанційного зондування або активації через Інтернет (див. рис. 1.2). Вона пропонує безкоштовний рівень для підключення обмеженої кількості пристроїв, але також можна встановити програмне забезпечення поза хмарою для приватного керування даними та пристроями, підключеними до платформи, без будь-яких обмежень [22 - 28].

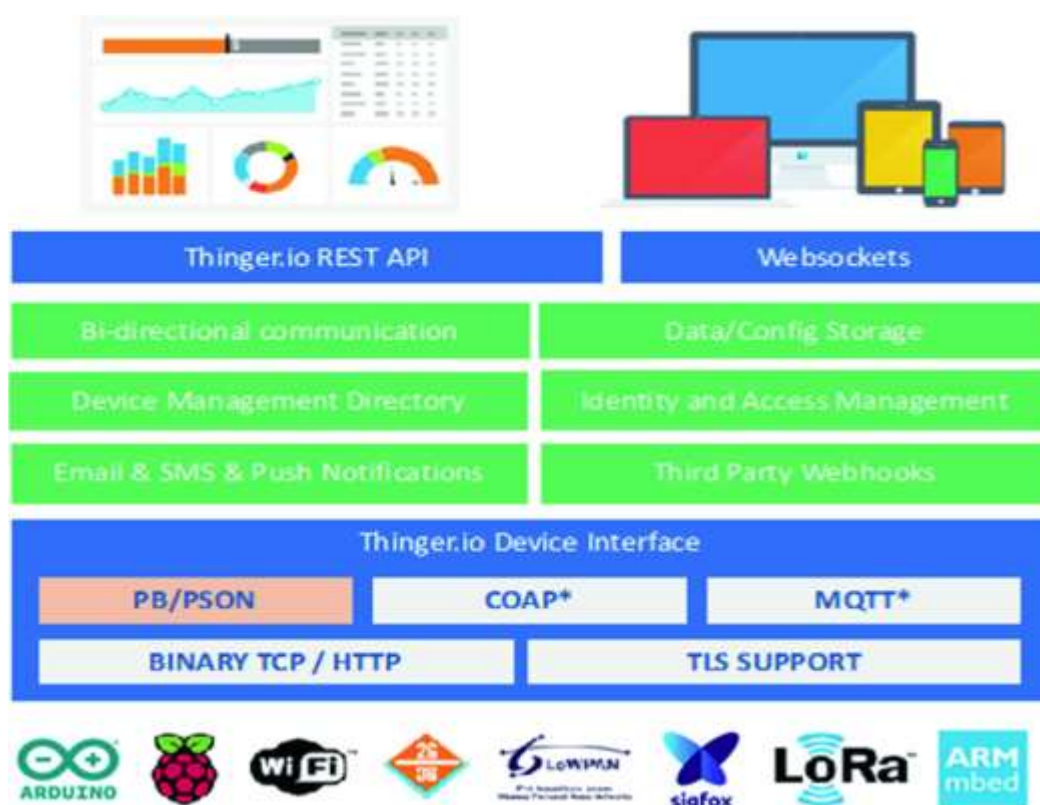


Рисунок 1.2 — Платформа Thingier.io

Ця платформа не залежить від апаратного забезпечення, тому можна підключити будь-який пристрій із підключенням до Інтернету, від пристроїв Arduino, Raspberry Pi, пристроїв Sigfox, рішень Lora через шлюзи або пристроїв

ARM. Платформа надає деякі стандартні функції, такі як реєстр пристроїв, двонаправлений зв'язок у реальному часі, як для зондування, так і для активації, зберігання даних і конфігурації, тому є можливість зберігати дані часових рядів, керування ідентифікацією та доступом (IAM), щоб дозволити стороннім особам отримувати доступ до платформи та ресурсів пристрою через REST/Websocket API, Webhooks сторонніх розробників, щоб пристрої могли легко викликати інші веб-сервіси, надсилати електронні листи, SMS, надсилати дані в інші хмари тощо. Вона також надає веб-інтерфейс для керування всіма ресурсами та створення інформаційних панелей для віддаленого моніторингу. Основною перевагою використання цієї платформи, окрім того, що вона є відкритим кодом, є можливість отримати двонаправлений зв'язок із пристроями в режимі реального часу за допомогою стандартних інтерфейсів REST-API [29, 30].

Таким чином, можна розробити будь-який додаток, наприклад, настільний, мобільний, веб-сервіс, який взаємодіє з пристроями за допомогою добре відомого та перевіреного інтерфейсу на основі REST. Тим часом пристрої можуть використовувати більш ефективні (з точки зору пропускну здатності або обсягу пам'яті) бінарні протоколи для зв'язку з хмарою.

З іншого боку, ця платформа надає клієнтські бібліотеки для підключення кількох найсучасніших пристроїв IoT, таких як Arduino, ESP8266, ESP32, LinkitOne, Texas Instruments CC3200, Libellium Waspnotes, Raspberry Pi тощо. Клієнтські бібліотеки забезпечують комплексний спосіб підключення пристроїв і надсилання інформації в хмару без необхідності мати справу зі складними протоколами IoT [31 - 35].

Libelium MySignals. Це платформа для розробки медичних пристроїв і додатків eHealth. Платформу можна використовувати для розробки веб-додатків eHealth і для тестування власних датчиків для медичних програм. MySignals є прикладом комерційного продукту, який пропонує та підтримує іспанська компанія LibeliumFootnote9. Вона дозволяє вимірювати понад 20 біометричних

параметрів, таких як пульс, частота дихання, кисень у крові, сигнали електрокардіограми, артеріальний тиск, сигнали м'язової електроміографії, рівні глюкози, шкірно-гальванічна реакція, ємність легенів, хвилі хропіння, положення пацієнта, потік повітря та масштаб тіла. Дані, зібрані датчиками, можуть зберігатися в MySignals або сторонній хмарі для візуалізації у зовнішніх веб-додатках і мобільних додатках. Libelium пропонує розробникам API для доступу до інформації [36]. Одним із недоліків є обмеження для надсилання інформації, що надходить із MySignals, на сторонній хмарний сервер за допомогою безпосереднього радіозв'язку WiFi, яке обмежено лише для апаратної версії — ця опція недоступна для версії програмного забезпечення, що працює на вузлі libelium.

FIWARE — це IoT-платформа з відкритим вихідним кодом, яка «комбінує компоненти, які забезпечують підключення до IoT із службами Context Information Management і Big Data у хмарі» [37, 38]. FIWARE використовує досить прості

стандартні API для керування та обміну даними, які можна використовувати для розробки розумних програм. Вона надає розширені можливості хмарного хостингу на основі OpenStack і низку компонентів для функцій, які пропонуються як сервіс. Впровадженню технології FIWARE на ринку електронної охорони здоров'я сприяє проект промислового прискорювача FICHe, запущений Європейською Комісією, і кілька промислових проектів, які були розроблені в контексті FICHe. CardioWeel — це передова система допомоги водієві, яка надає інформацію про стан здоров'я водія за допомогою електрокардіограми людини (ЕКГ), отриманої з рук водія. Ця система розміщена в хмарі FIWARE з використанням доступних віртуальних машин.

Runkeeper — це застосунок для відстеження різноманітних вправ. Runkeeper відстежує ходьбу, біг та будь-яку іншу фізичну активність. Він підходить для людей будь-якого рівня фізичної підготовки, і компанія стверджує, що має спільноту з 50 мільйонів користувачів. Людина може

створювати, зберігати та відкривати нові маршрути. Додаток також використовує GPS для моніторингу темпу та пройденої відстані. Інформаційна панель «Мій план» надає власні плани тренувань на основі відповідей користувача на серію запитань, або людина може вибрати з готових розкладів [39].

Strava — це мобільний застосунок для відстеження їзди на велосипеді та бігу. Strava дозволяє користувачам відстежувати свою подорож на велосипеді та бігу за допомогою GPS [40]. Людина також може приєднатися до викликів, спілкуватися з друзями та стежити за ними, а також ділитися фотографіями через додаток. Програма також сумісна з рядом пристроїв GPS, зокрема:

- телефони;
- годинник;
- монітори пульсу.

Додаток є безкоштовним для використання, але люди можуть отримати доступ до багатьох додаткових функцій, якщо вони платять 7,99 доларів на місяць. До них входять обмін інформацією про місцезнаходження в реальному часі, планування маршруту, встановлення цілей, журнали тренувань і функція Heatmaps, що показує популярні маршрути в поточному місці розташування людини.

Glo пропонує онлайн-заняття з йоги, медитації, пілатесу та фітнесу. Він призначений як для новачків, так і для більш досвідчених людей. Користувач відповідає на кілька запитань, щоб персоналізувати свій досвід, і більше 4000 завантажуваних класів доступні за запитом. Викладачі приїжджають з усього світу [41].

Sworakit — це додаток, який призначений для різних рівнів досвіду. Sworakit підходить для всіх, від початківців до досвідчених спортсменів. Людина може вирішити, чи хоче вона, щоб її тренування було зосереджено на силі, кардіо, йозі, пілатесі чи розтяжці. Вони також можуть вибирати з різноманітних щоденних або щомісячних завдань і ряду колекцій тренувань. Додаток також дозволяє

користувачам вибирати тренування на основі обладнання, фітнес-цілей і рівня впливу. Люди можуть встановити індивідуальний 6-тижневий план, створювати тренування з нуля, відстежувати свою продуктивність і спілкуватися сам-на-сам з особистими тренерами. Також доступні нагадування, музика та розваги для дітей [42].

JEFIT — це мобільний додаток та журнал тренажерного залу відстеження тренувань. JEFIT дозволяє користувачам відстежувати свої тренування та прогрес у важкій атлетиці. За допомогою цього додатка можлива персоналізація, оскільки людина має доступ до понад 1400 вправ з інструкціями. Компанія стверджує, що додаток простий у використанні та вимагає лише одного кліка, щоб зареєструвати навчання. Час відпочинку також легко контролювати, а також доступні аудіо та відео інструкції. Людина також може зосередитися на певних м'язах і отримати вказівки щодо відновлення м'язів.

Charity Miles — це програма, яка допомагає збирати гроші на добрі справи. За кожен миль фізичної активності, яку людина проходить, компанія жертвує гроші на благодійну організацію. Людина може вибрати з понад 50 благодійних організацій. Компанія стверджує, що на сьогодні пожертвувала понад 2 мільйони доларів. Люди можуть брати участь у будь-якому виді фізичної активності, включаючи ходьбу, танці, їзду на велосипеді та біг.

1.3 Постановка задачі

Ціллю магістерської кваліфікаційної роботи є розробка інформаційної системи для спостереження за фізіологічним станом людини. Для її реалізації необхідно здійснити аналітичний огляд способів побудови подібних інформаційних систем, розробити архітектуру інформаційної системи, створити програмне забезпечення для інформаційної системи.

Завданням розробки є створення мобільного додатку, призначеного для операційної системи Android, додаток необхідно створити на об'єктно-орієнтованій мові програмування Java. Розроблюваний додаток повинен бути

простим та зручним у користуванні, забезпечувати моніторинг фізичної активності та енергетичних витрат людини, реалізовувати систему самозвіту для обчислення енергетичних витрат організму, в якій використовується гнучка система вихідних параметрів, що надає можливість враховувати індивідуальні особливості при контролі фізичної активності та здійсненні контролю енергетичного балансу організму людини.

РОЗДІЛ 2. АРХІТЕКТУРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Розробка життєвих циклів

Життєвий цикл застосунку в Android розпочинається, коли користувач натискає на іконку додатку. Такий дотик сприводить до виконання методу onCreate() (рис. 3.1) режим очікування (в термінології Android, активності) додатку.

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    MyMenuActivity.isOpen = true;  
    setContentView(R.layout.activity_my_menu);  
    findViewById(R.id.from_menu_to_diary).setOnClickListener(new ToWindowOnClick(this, DiaryActivity.class));  
    findViewById(R.id.from_menu_to_diary_menu).setOnClickListener(new ToWindowOnClick(this, DiaryActivity.class));  
    findViewById(R.id.from_menu_to_profile).setOnClickListener(new ToWindowOnClick(this, ProfileActivity.class));  
    findViewById(R.id.from_menu_to_dishes).setOnClickListener(new ToWindowOnClick(this, DishActivity.class));  
    findViewById(R.id.from_menu_to_exercises).setOnClickListener(new ToWindowOnClick(this, ExerciseActivity.class));  
    findViewById(R.id.from_menu_to_statistic).setOnClickListener(new ToWindowOnClick(this, StatisticActivity.class));  
}
```

Рисунок 2.1 – Реалізація методу onCreate() у застосунку

Даний метод містить код для відображення на екрані потрібних елементів, які включені в розмітці потрібної активності. Розробник повинен доповнити даний метод кодом, який буде ініціалізувати змінні та об'єкти розмітки до значень, які необхідні, щоб користувач почав взаємодію з застосунком.

Після того, як активність застосунку розпочалася, йде метод onResume(). Цей метод не обов'язково реалізовувати, але це корисно робити, щоб повернути застосунок в стан виконання, у якому він знаходився до переходу в стан паузи. Це, зокрема, відноситься до різних системних служб, які використовуються застосунком (наприклад, служб GPS або камери), перезапуску анімації, а також

до будь-яких інших налаштувань, які необхідні для відновлення роботи користувача з додатком.

Коли користувач вирішує зупинити взаємодію з застосунком, розпочинається гілка його видалення. Ніякий з методів, які запущені на гілці видалення, не є обов'язковими для реалізації.

Проте, вони часто виявляються корисними та тому їх потрібно ретельно розглянути на предмет реалізації. Перший метод, який виконується — це метод `onPause()`. Даний метод повинен використовуватися для призупинки використаних застосунком служб, для зупинки анімації, для збереження необхідної інформації про стан для того, щоб користувач міг відновити користування застосунком з того стану, в якому він його залишив.

Якщо застосунок збирається стати невидимим, викликається на виконання метод `onStop()`. Цей метод повинен гарантувати, що всі важливі дані додатку передані на постійне зберігання для того, щоб вони не загубилися, коли системні ресурси цього додатку будуть перерозподілені між іншим додатками.

Якщо додаток має завершити свою роботу та знищити всі свої дані з оперативної пам'яті, то операційною системою викликається на виконання метод `onDestroy()` (рис. 3.2).

```
protected void onDestroy() {  
    super.onDestroy();  
    db.close(); // close DB  
}
```

Рисунок 2.2 – Реалізація методу `onDestroy()` у застосунку

2.2 Системні компоненти

Об'єкт `Location` представляє географічне розташування, яке може складатися з широти, довготи, позначки часу та іншої інформації, такої як пеленг, висота та швидкість. Існують наступні важливі методи, які можна використовувати з об'єктом `Location` для отримання інформації про місцезнаходження. Щоб отримати поточне місцезнаходження, створюємо клієнт розташування, який є об'єктом `LocationClient`, підключаємо його до служб визначення місцезнаходження за допомогою методу `connect()`, а потім викликаємо його методом `getLastLocation()`. Цей метод повертає останнє місцезнаходження у формі об'єкта `Location`, який містить координати широти та довготи та іншу інформацію, як пояснено вище. Щоб мати функціональність на основі розташування, доведеться реалізувати два інтерфейси:

- `GooglePlayServicesClient.ConnectionCallbacks`;
- `GooglePlayServicesClient.OnConnectionFailedListener`.

Ці інтерфейси забезпечують наступні важливі методи зворотного виклику, які потрібно реалізувати у вашому активності. Далі слід створити клієнт розташування в методі `onCreate()` класу активності, а потім підключити його в `onStart()`, щоб служби визначення місцезнаходження зберігали поточне місцезнаходження, поки діяльність була повністю видимою. Потрібно від'єднати клієнт у методі `onStop()`, щоб, коли програма не відображається, служби визначення місцезнаходження не підтримувала поточне місцезнаходження. Це значною мірою допомагає заощадити заряд акумулятора. Для оновлення місцеположення, окрім вищезазначених інтерфейсів, потрібно буде також реалізувати інтерфейс `LocationListener`. Цей інтерфейс забезпечує наступний метод зворотного виклику, який потрібно реалізувати у класі активності.

Тепер, якщо програмі потрібна висока точність місцезнаходження, вона повинна створити запит про місцезнаходження з `setPriority(int)` у значення `PRIORITY_HIGH_ACCURACY` і `setInterval(long)` у 5 секунд. Можна також використовувати більший інтервал або інші пріоритети, наприклад

PRIORITY_LOW_POWER, щоб запитувати точність рівня "місто" або PRIORITY_BALANCED_POWER_ACCURACY для точності рівня "блок". Діяльність має наполегливо розглянути питання про видалення всіх запитів про місцезнаходження під час переходу у фоновий режим (наприклад, у onPause()) або принаймні змінити запит на більший інтервал і нижчу якість, щоб заощадити енергоспоживання.

Далі використовуємо метод Geocoder.getFromLocation(), щоб отримати адресу для заданої широти та довготи. Цей метод є синхронним і може зайняти багато часу, щоб виконати свою роботу, тому потрібно викликати метод з методу doInBackground() класу AsyncTask.

Для використання AsyncTask має бути підкласом, і підклас замінить метод doInBackground(Params...) для виконання завдання у фоновому режимі, а метод onPostExecute(Result) викликається в потоці інтерфейсу користувача після завершення фонового обчислення та під час відобразити результат. В AsyncTask доступний ще один важливий метод — execute(Params... params), цей метод виконує завдання із зазначеними параметрами.

Нижче наведено вміст файлу основної діяльності MainActivity.java із підключеними методами для виявлення розташування :

```
public class MainActivity extends Activity {
    Button btnShowLocation;

    private static final int REQUEST_CODE_PERMISSION = 2;
    String mPermission = Manifest.permission.ACCESS_FINE_LOCATION;
    // GPSTracker class
    GPSTracker gps;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        try {
```

```

if (ActivityCompat.checkSelfPermission(this, mPermission)
!= MockPackageManager.PERMISSION_GRANTED) {
    ActivityCompat.requestPermissions(this, new String[]{mPermission},
    REQUEST_CODE_PERMISSION);
    // If any permission above not allowed by user, this condition will
    execute every time, else your else part will work
}
} catch (Exception e) {
    e.printStackTrace();
}

```

База даних SQLite — це база даних з відкритим вихідним кодом, яка надається в Android і використовується для зберігання даних у пристрої користувача у формі текстового файлу. Ми можемо виконувати багато операцій з цими даними, наприклад додавати нові дані, оновлювати, читати та видаляти ці дані. SQLite — це автономна база даних, яка локально зберігається на пристрої користувача, і не потрібно створювати жодних з'єднань для підключення до цієї бази даних. Дані зберігаються в базі даних SQLite у вигляді таблиць. Коли дані зберігаються в базі даних SQLite, вони впорядковуються у формі таблиць, схожих на таблиці Excel. Додаємо дозволи на доступ до сховища у файлі AndroidManifest.xml :

```

<uses-permission
android:name="android.permission.READ_EXTERNAL_STORAGE" />

```

Далі переходимо до файлу res > layout > activity_main.xml і додаємо наведений нижче код :

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"

```

```

android:layout_height="match_parent"
android:orientation="vertical"
tools:context=".MainActivity">
<!--Edit text to enter course name-->
<EditText
android:id="@+id/idEdtCourseName"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_margin="10dp"
android:hint="Enter course Name" />
<!--edit text to enter course duration-->
<EditText
android:id="@+id/idEdtCourseDuration"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_margin="10dp"
android:hint="Enter Course Duration" />
<!--edit text to display course tracks-->

```

Далі створюємо новий клас Java для виконання операцій SQLite. Переходимо до файлу java > Ccounter > права кнопка миші > створити > клас Java і називаємо його як DBHandler і додаємо до нього наведений нижче код :

```

public class DBHandler extends SQLiteOpenHelper {
private static final String DB_NAME = "coursedb";
private static final int DB_VERSION = 1;
private static final String TABLE_NAME = "mycourses";
private static final String ID_COL = "id";
private static final String NAME_COL = "name";
private static final String DURATION_COL = "duration";
private static final String DESCRIPTION_COL = "description";

```

```

private static final String TRACKS_COL = "tracks";
public DBHandler(Context context) {
    super(context, DB_NAME, null, DB_VERSION);
}
@Override
public void onCreate(SQLiteDatabase db) {
    String query = "CREATE TABLE " + TABLE_NAME + " ("
        + ID_COL + " INTEGER PRIMARY KEY AUTOINCREMENT, "
        + NAME_COL + " TEXT,"
        + DURATION_COL + " TEXT,"
        + DESCRIPTION_COL + " TEXT,"
        + TRACKS_COL + " TEXT)";
    if (ActivityCompat.checkSelfPermission(this, mPermission)
        != MockPackageManager.PERMISSION_GRANTED) {
        ActivityCompat.requestPermissions(this, new String[]{mPermission},
            REQUEST_CODE_PERMISSION);
    }
    db.execSQL(query);
}

public void addNewCourse(String courseName, String courseDuration, String
    courseDescription, String courseTracks) {
    ContentValues values = new ContentValues();
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    values.put(DESCRIPTION_COL, courseDescription);
    values.put(TRACKS_COL, courseTracks);
    db.insert(TABLE_NAME, null, values);
    db.close();
}

```

2.3 Перетворення вихідного коду в програму Android

Вихідні файли Java перетворюються на файли класів Java компілятором Java. Android SDK містить інструмент під назвою `dx`, який перетворює файли класів Java у файл `.dex` (Dalvik Executable). Усі файли класів програми розміщено в цьому файлі `.dex`. Під час цього процесу перетворення надлишкова інформація у файлах класів оптимізується у файлі `.dex`. Наприклад, якщо той самий рядок знайдено в різних файлах класу, файл `.dex` містить лише одне посилання на цей рядок. Тому ці файли `.dex` набагато менші за розміром, ніж відповідні файли класів. Файл `.dex` та інші ресурси, наприклад зображення та XML-файли, упаковуються у файл `.apk` (пакет Android). Програма `aapt` (Android Asset Packaging Tool) виконує цей крок.

Отриманий файл `.apk` містить усі необхідні дані для запуску програми Android і може бути розгорнуто на пристрої Android за допомогою інструменту `adb`. Починаючи з Android 5.0, Android RunTime (ART) використовується як середовище виконання для всіх програм Android. ART використовує комбінацію компіляції Ahead Of Time і `_Just In Time_`. Під час інсталяції програми на пристрої Android код програми транслюється в машинний код.

Інструмент `dex2oat` бере файл `.dex`, створений ланцюгом інструментів Android, і компілює його у виконуваний і компонований формат (файл ELF). Цей файл містить код `dex`, скомпільований рідний код і метадані. Збереження коду `.dex` дозволяє існуючим інструментам працювати.

Система збірки Gradle розроблена для підтримки складних сценаріїв створення додатків Android:

- мультирозповсюдження: один і той же додаток необхідно налаштувати для кількох клієнтів або компаній;
- multi-apk: підтримка створення кількох арк для різних типів пристроїв із повторним використанням частин коду.

Запускається збірка Gradle за допомогою командного рядка. Для перегляду всіх доступних завдань використовуємо команду `gradlew wrapper` :

gradle build

alternatively speedup second grandle build by holding it in memory

gradle build –daemon

Ця команда створює в папці збірки вихід збірки Gradle. За замовчуванням збірка Gradle створює два файли .apk у папці build/outputs/apk. Для створення та запуску модульних тестів на JVM використовуємо команду gradle test.

Для створення та запуску інструментальних тестів на пристрої Android використовуємо команду gradle connectedCheck.

Для перегляду всіх доступних завдань використовуємо команду gradlew wrapper :

gradle build

alternatively speedup second grandle build by holding it in memory

gradle build –daemon

Ця команда створює в папці збірки вихід збірки Gradle. За замовчуванням збірка Gradle створює два файли .apk у папці build/outputs/apk.

Для створення та запуску модульних тестів на JVM використовуємо команду gradle test.

РОЗДІЛ 3. РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1. Програмна реалізація основного функціоналу застосунку

Після запуску застосунку в першу чергу буде відображатися екран «Меню», через який користувач зможе переміщуватися по іншим екранам. Лістинг екрану «Меню»:

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    MyMenuActivity.isOpen = true;  
    setContentView(R.layout.activity_my_menu);  
    findViewById(R.id.from_menu_to_diary).setOnClickListener(new  
        ToWindowOnClick(this, DiaryActivity.class));  
    findViewById(R.id.from_menu_to_diary_menu).setOnClickListener(new  
        ToWindowOnClick(this, DiaryActivity.class));  
    findViewById(R.id.from_menu_to_profile).setOnClickListener(new  
        ToWindowOnClick(this, ProfileActivity.class));  
    findViewById(R.id.from_menu_to_dishes).setOnClickListener(new  
        ToWindowOnClick(this, DishActivity.class));  
    findViewById(R.id.from_menu_to_exercises).setOnClickListener(new  
        ToWindowOnClick(this, ExerciseActivity.class));  
    findViewById(R.id.from_menu_to_statistic).setOnClickListener(new  
        ToWindowOnClick(this, StatisticActivity.class));  
}
```

У щоденнику користувач буде повинен вибрати спожиті страви та виконані фізичні вправи за день, які будуть вибиратися із переліку вже створених об'єктів. Отже потрібно буде звертатися до бази даних, в якій будуть знаходитися дані об'єкти. Використовуючи дані, які були внесені користувачем,

застосунок обраховуватиме калорії. Також користувач може викликати календар для переходу. Лістинг екрану «Щоденник»:

```
protected void onCreate(Bundle savedInstanceState) {
    profile = Profile.getProfile(this);
    db = DataBase.getDataBase(this);
    setViewDay(new MyDate());
    dishesData =
        DataBase.cursorToArrayList(db.getAllDayDishes(viewDay.getDate()));
    String[] from = new String[]{DataBase.DISH_COLUMN_NAME,
        DataBase.DAYS_DISH_COLUMN_WEIGHT};
    int[] to = new int[]{R.id.db_item_name, R.id.db_item_right_text};
    dishesAdapter = new SimpleAdapter(this, dishesData, R.layout.database_item,
        from, to);
    exercisesData =
        DataBase.cursorToArrayList(db.getAllDayExercises(viewDay.getDate()));
    from = new String[]{DataBase.EXERCISE_COLUMN_NAME,
        DataBase.DAYS_EXERCISE_COLUMN_QUANTITY};
    exercisesAdapter = new SimpleAdapter(this, exercisesData,
        R.layout.database_item, from, to);
}
```

У екрані «Профіль» користувач вносить дані про себе (стать, вага, вік та зріст). Ці дані використовуються для обрахунку калорій, які будуть використовуватися у щоденнику. Для обрахунку використовується спеціальна формула. Лістинг екрану «Профіль»:

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_profile);
    profile = Profile.getProfile(this);
    findViewById(R.id.from_profile_to_menu).setOnClickListener(new
```

```

ToWindowOnClickWithClosing(this, MyMenuActivity.class));
((EditText) findViewById(R.id.profile_age)).setText(profile.getAge() + "");
((EditText) findViewById(R.id.profile_height)).setText(profile.getHeight() + "");
((EditText) findViewById(R.id.profile_weight)).setText(profile.getWeight() +
    "");
((RadioButton)
findViewById(R.id.profile_woman_radio_button)).setChecked(!profile.getGender());
((EditText)
findViewById(R.id.profile_aim_kal_number)).setText(profile.getAimCalorie() +
    "");
optimalCalorieNumber = ((TextView)
findViewById(R.id.profile_optimal_kal_number));
optimalCalorieNumber.setText(profile.calculateCalories() + " " +
getString(R.string.kilocalories));
}

```

У розділі «Страви» знаходиться список страв, які створює, редагує та видаляє сам користувач. При редагуванні чи створення нової страви з'являється спеціальний екран для цих дій. Всі об'єкти заносяться в базу даних застосунку. Значення калорій та назву страви вводить користувач. Лістинг екрану «Страви»:

```

protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_dish);
    findViewById(R.id.from_dishes_to_menu).setOnClickListener(new
    ToWindowOnClickWithClosing(this, MyMenuActivity.class));
    db = DataBase.getDataBase(this);
    data=DataBase.cursorToArrayList(db.getDishes());
    String[] from = new String[] {DataBase.DISH_COLUMN_NAME,
    DataBase.DISH_COLUMN_CALORIES_PER_100_GM }; // columns names

```

```

int[] to = new int[] { R.id.db_item_name, R.id.db_item_right_text}; // places to
write (View id)
sAdapter = new SimpleAdapter(this, data , R.layout.database_item, from, to);
listView= findViewById(R.id.dish_list_view);
listView.setAdapter(sAdapter);
listView.setChoiceMode(ListView.CHOICE_MODE_SINGLE);
listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
@Override
public void onItemClick(AdapterView<?> parent, View view, int position, long id)
{
selectedElementId = id;
}
});

```

Розділ «Вправи» складається із списку фізичних активностей, які можна також редагувати, створювати та редагувати. Всі об'єкти також заносяться у базу даних застосунку. Аналогічно, як у попередньому розділі, для створення та редагування вправ викликається окремий екран, де користувач змінює загальну формулу обрахунку калорій для усіх вправ. Лістинг екрану «Вправи»:

```

protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_exercise);
findViewById(R.id.from_exercises_to_menu).setOnClickListener(new
ToWindowOnClickWithClosing(this, MyMenuActivity.class));
db = DataBase.getDataBase(this);
data = DataBase.cursorToArrayList(db.getExercises());
String[] from = new String[]{DataBase.EXERCISE_COLUMN_NAME};
int[] to = new int[]{R.id.db_item_name};
sAdapter = new SimpleAdapter(this, data, R.layout.database_item, from, to);
listView = (ListView) findViewById(R.id.exercise_list_view);

```

```

listView.setAdapter(sAdapter);
listView.setChoiceMode(ListView.CHOICE_MODE_SINGLE);
listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position, long id)
    {
        selectedElementId = id;
    }
});
}

```

У розділі «Статистика» візуалізуються витрачені калорії із кожної страви, які були вибрані у щоденнику. Лістинг екрану «Статистика»:

```

protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_statistic);
    findViewById(R.id.from_statistic_to_menu).setOnClickListener(new
    ToWindowOnClickWithClosing(this, MyMenuActivity.class));
    barChart = findViewById(R.id.statistic_histogram);
    db = DataBase.getDataBase(this);
    profile = Profile.getProfile(this);
    showDiagram();
}

private void showDiagram() {
    switch (diagramID) {
        case DAY_DIAGRAM:
            showDayDiagram();
        case MONTH_DIAGRAM:
            showMonthDiagram();
        break;
    }
}

```

```
case YEAR_DIAGRAM:  
showYearDiagram();  
break;  
}  
}
```

3.2 Тестування застосунку

Було протестовано здатність інформаційної системи моніторингу рахувати кроки, пройдені користувачем та визначення швидкості при ходьбі за допомогою мобільного телефону із відключеною службою GPS. Результат показано на рисунку 3.1.

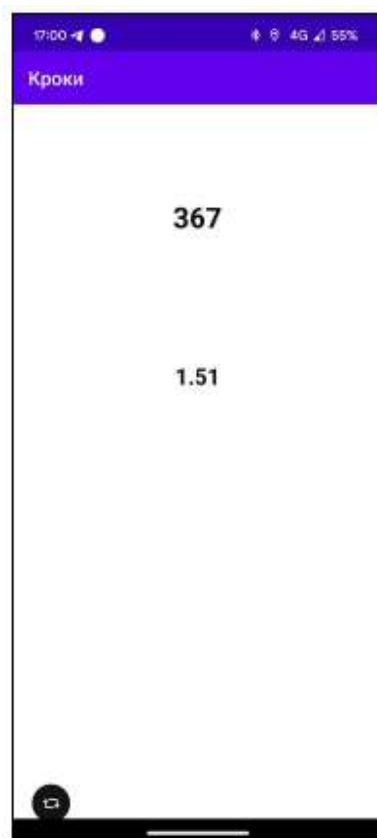


Рисунок 3.1 — Результат підрахунку кроків та визначення швидкості

Дана функція працювати лише на пристроях, які мають у своїй специфікації відповідні GPS модулі. Інший функціонал мобільного застосунку

було протестовано в середовищі розробки Android Studio за допомогою віртуального девайсу (див. рис. 3.2).

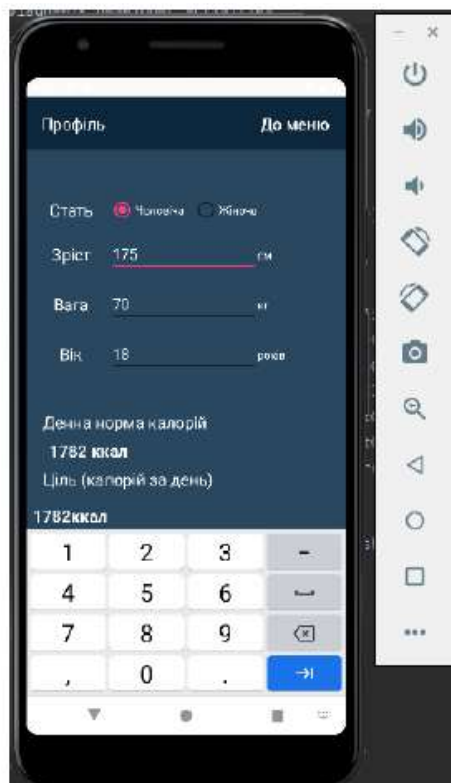


Рисунок 3.2 — Тестування функціонування за допомогою віртуального девайсу

3.3 Інструкція користувача

Користувач має володіти мобільним пристроєм під керуванням операційної системи Android не нижче версії 4.1. Додаток інсталюється на мобільний пристрій користувача через відкриття “apk” файлу та його автоматичного налаштування.

Користування функціоналом інформаційної системи розпочинається з початкового екрану «Меню». Тут знаходяться кнопки із посиланнями на усі інші екрани (див. рис. 3.3).



Рисунок 3.3 — Вигляд екрану «Меню»

На екрані «Щоденник» користувач системи заповнює дані щодо спожитих та витрачених калорій (рис. 3.4).



Рисунок 3.4 — Вигляд екрану «Щоденник»

На екрані “Страви” знаходиться створена база даних різних продуктів із заздалегіть підрахованими калоріями. Дану базу можна поповнювати та змінювати вже збережені дані (рис. 3.5).



Рисунок 3.5 — Екран вибору страв

Щоб додати виконану фізичну вправу, потрібно натиснути на жовту кнопку. Потім потрібно вибрати вправу із списку та вказати час та кількість виконання. Для підтвердження натиснути на зелену кнопку. Для відміни натиснути червону кнопку (рис. 3.6).



Рисунок 3.6 – Екран вибору вправ

Для переключення днів натиснути на поле із датою та вибрати потрібний день у календарі, що з'явився. Найнижче знаходиться поле для поміток користувача. На екрані «Профіль» користувач вносить свої дані для подальшого обрахунку калорій. У поля вік, зріст та вага користувач вводить числа, а навпроти статі вибирає один із двох варіантів (рис. 3.7).

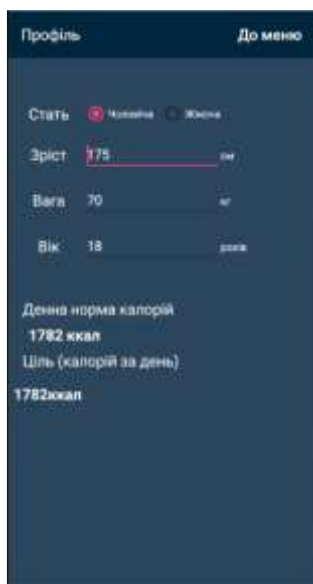


Рисунок 3.7 – Вигляд екрану «Профіль»

На екрані «Страви» міститься список страв користувача. Для видалення страви потрібно натиснути на червоний хрест, вибрати страву та повторно натиснути на ту саму кнопку (рис. 3.8).

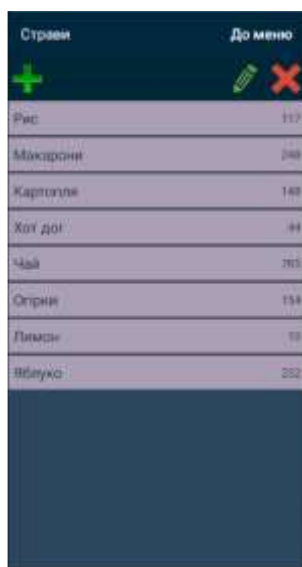


Рисунок 3.8 – Вигляд екрану «Страви»

Щоб внести зміни в існуючі страви, потрібно натиснути на зелений олівець, вибрати страву та повторно натиснути на кнопку. З'явиться екран для редагування страви, де можна змінити кількість калорій на 100 грам. А щоб створити нову страву, потрібно натиснути на зелений хрестик. Тоді з'явиться такий самий екран, що був для редагування (рис. 3.9).



Рисунок 3.9 – Вигляд екрану створення та редагування страв

На екрані «Вправи» міститься список фізичних вправ користувача. Для видалення страви потрібно натиснути на червоний хрест, вибрати вправу та повторно натиснути на ту саму кнопку (рис. 3.10).



Рисунок 3.10 – Вигляд екрану «Вправи»

Щоб внести зміни в існуючі фізичні вправи, потрібно натиснути на зелений олівець, вибрати вправу та повторно натиснути на кнопку. З'явиться екран для редагування вправи, де можна змінити формулу для розрахунку витрачених калорій. А щоб створити нову вправу, потрібно натиснути на зелений хрестик. Тоді з'явиться такий самий екран, що був для редагування (рис. 3.11).



Рисунок 3.11 – Вигляд екрану створення та редагування фізичних вправ

На екрані «Статистика» демонструються спожиті калорії за проміжок часу. Кожна страва представляється окремо для зручного порівняння (рис. 3.12).

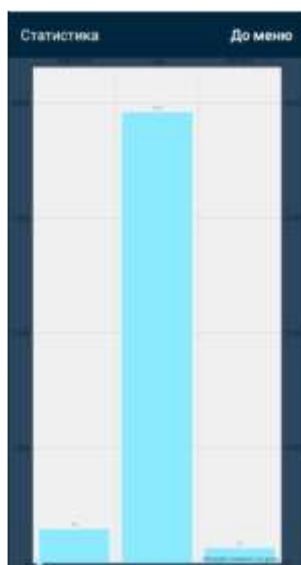


Рисунок 3.12 – Екран «Статистика»

ВИСНОВОК

У результаті виконання кваліфікаційної роботи магістра був проведений аналіз існуючих систем та методів моніторингу фізіологічного людини, а також аналіз готових рішень від різних компаній. Була доведена актуальність дослідження існуючих систем та методів, проведене обґрунтування розробки методу для інформаційної системи моніторингу фізіологічного стану людини, визначена задача роботи.

На основі результатів аналізу була розроблена інформаційна система моніторингу фізіологічного стану людини та проведений аналіз розробленого методу. У ході виконання роботи було проведене тестування використання нової інформаційної системи моніторингу фізіологічного стану людини у різних умовах та доведена її конкурентноспроможність. Для практичного застосування було розроблено спеціальний Android застосунок, який спрощує збір необхідних даних.

За результатами кваліфікаційної роботи магістра можна зробити висновок, що поставлена мета - аналіз сучасного стану і тенденцій розвитку світових розробок у сфері моніторингу фізіологічного стану людини та розробка програмного продукту для відстеження необхідних показників була виконана.

Актуальність теми проведеного дослідження зумовлена тим, що із поширенням небезпечних хвороб люди почали дельніше вивчати питання охорони власного здоров'я. Інформаційні технології допомагають лікарю проводити діагностику захворювань, накопичувати і ефективно використовувати отриману інформацію на всіх стадіях процесу лікування.

Об'єктом дослідження кваліфікаційної роботи магістра є процес моніторингу фізіологічного стану людини. Предметом дослідження кваліфікаційної роботи магістра є інформаційні системи для реалізації методів контролю енергетичного балансу в організмі людини. Метою кваліфікаційної роботи магістра є створення інформаційної системи для спостереження за фізіологічним станом людини з метою розширення переліку вихідних даних для

оцінювання фізичної активності та енергетичних витрат, що надає можливість більш ефективно підтримувати енергетичний баланс, запобігаючи надлишковій вазі та пов'язаним з нею захворюваннями.

Завдяки виконанню завдань дослідження проаналізовано актуальні методи моніторингу стану людини та готові рішення, що представлені на ринку, розроблено зручну та інтуїтивну інформаційну систему на доступній для будь-якого користувача операційній системі для моніторингу фізіологічного стану людини.

Наукова новизна отриманих результатів полягає у тому, що дістала подальшого розвитку система самозвіту для обчислення енергетичних витрат організму, в якій на відміну від уснуючої використовується більш гнучка система вихідних параметрів, що надає можливість враховувати індивідуальні особливості при контролі фізичної активності. Практичне значення одержаних результатів полягає в тому, що розроблено простий та зручний у використанні мобільний додаток для операційної системи Android для оцінювання фізичної активності та здійснення контролю енергетичного балансу організму людини.

У першому розділі кваліфікаційної роботи магістра проведено аналітичний огляд предметної області, у другому розділі розроблено архітектуру програмного забезпечення інформаційної системи, у третьому розділі розроблено програмне забезпечення інформаційної системи та розглянуто питання тестування розробленої інформаційної системи.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Information System for Monitoring Personal Health Status Based on Big Data Analysis [Електронний ресурс] – Режим доступу до ресурсу: <https://ceurws.org/Vol-2917/paper44.pdf>
2. Internet of things-enabled real-time health monitoring system using deep learning [Електронний ресурс] – Режим доступу до ресурсу: <https://link.springer.com/article/10.1007/s00521-021-06440-6>
3. Remote patient monitoring: a comprehensive study [Електронний ресурс] – Режим доступу до ресурсу: <https://link.springer.com/article/10.1007/s12652-017-0598-x>
4. Medical Data Processing and Analysis for Remote Health and Activities Monitoring [Електронний ресурс] – Режим доступу до ресурсу: https://link.springer.com/chapter/10.1007/978-3-030-16272-6_7
5. A Detailed Research on Human Health Monitoring System Based on Internet of Things [Електронний ресурс] – Режим доступу до ресурсу: <https://www.hindawi.com/journals/wcmc/2021/5592454/>
6. eValuate - A Sports Analytics mHealth App [Електронний ресурс] – Режим доступу до ресурсу: <https://www.divaportal.org/smash/get/diva2:1451496/FULLTEXT01.pdf>
7. Physical activity [Електронний ресурс] – Режим доступу до ресурсу: <https://www.who.int/news-room/fact-sheets/detail/physical-activity>
8. A Focused Review of Smartphone Diet-Tracking Apps: Usability, Functionality, Coherence With Behavior Change Theory, and Comparative Validity of Nutrient Intake and Energy Estimates [Електронний ресурс] – Режим доступу до ресурсу: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6543803/>
9. Measurement Methods for Physical Activity and Energy Expenditure: a Review [Електронний ресурс] – Режим доступу до ресурсу: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5426207/>

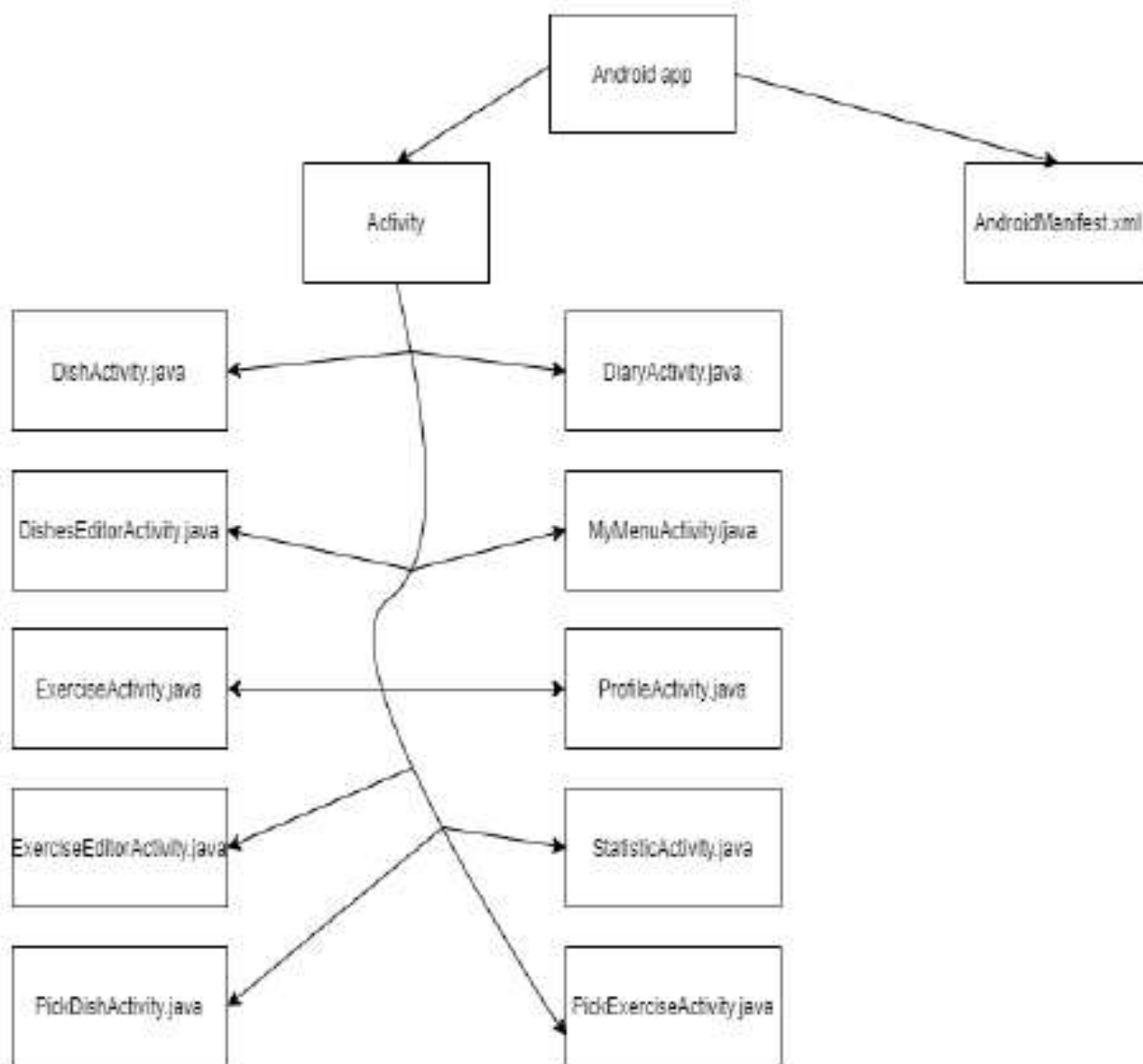
10. Multi-sensory Cyber-Physical Therapy System for Elderly Monitoring [Электронный ресурс] – Режим доступа до ресурсу: https://link.springer.com/chapter/10.1007/978-3-319-39949-2_9
11. Calorie counting smart phone apps: Effectiveness in nutritional awareness, lifestyle modification and weight management among young Indian adults [Электронный ресурс] – Режим доступа до ресурсу: https://www.researchgate.net/publication/Calorie_counting_smart_phone_apps_Effectiveness_in_nutritional_awareness_lifestyle_modification_and_weight_management_among_young_Indian_adults
12. How are Calories Calculated on the ELEMNT and Wahoo Fitness App? [Электронный ресурс] – Режим доступа до ресурсу: <https://support.wahoofitness.com/hc/en-us/articles/204280754-How-are-Calories-Calculated-on-the-ELEMNT-and-Wahoo-Fitness-App>
13. Operating System Market Share Worldwide [Электронный ресурс] – Режим доступа до ресурсу: <https://gs.statcounter.com/os-market-share>
14. What is an operating system? [Электронный ресурс] – Режим доступа до ресурсу: <https://www.techtarget.com/whatis/definition/operating-system-OS>
15. Windows [Электронный ресурс] – Режим доступа до ресурсу: <https://www.computerhope.com/jargon/w/windows.htm>
16. Linux [Электронный ресурс] – Режим доступа до ресурсу: <https://www.computerhope.com/jargon/l/linux.htm>
17. macOS [Электронный ресурс] – Режим доступа до ресурсу: <https://www.computerhope.com/jargon/m/macOS.htm>
18. MacOS [Электронный ресурс] – Режим доступа до ресурсу: <https://www.britannica.com/technology/Mac-OS>
19. What is an Android Operating System & Its Features [Электронный ресурс] – Режим доступа до ресурсу: <https://www.elprocus.com/what-is-android-introductionfeatures-applications/>

20. What Is iOS? – A Brief Introduction [Електронний ресурс] – Режим доступу до ресурсу: <https://www.emizentech.com/blog/what-is-ios.html>
21. Java is a powerful general-purpose programming language. [Електронний ресурс]. – Режим доступу: <https://www.programiz.com/javaprogramming>
22. «CPU vs. GPU Rendering: Which Should You Choose and Why?». IncrediBuild. <https://www.incredibuild.com/blog/cpu-vs-gpu-rendering-whichshould-you-choose-and-why> (accessed Nov. 21, 2022).
23. Федотенко М. Розробка мобільних додатків. Перші кроки.
24. Герберт Шилдт. Java 8. Посібник для початківців 2016. – 669 с.
25. Fundamentals of Testing [Електронний ресурс]. – 2021. – Режим доступу до ресурсу: <https://developer.android.com/training/testing/fundamentals>
26. Software R. Understanding The Difference Between MVC, MVP and MVVM Design Patterns [Електронний ресурс] / Rishabh Software. – 2021. – Режим доступу до ресурсу:
<https://www.linkedin.com/pulse/understanding-difference-between-mvc-mvpmvvm-design-rishabh-software/>.
25. Getting started with WebStorm [Електронний ресурс] : echo / Режим доступу: <https://www.jetbrains.com/help/webstorm/getting-started-withwebstorm.html>
26. Visual Studio Code Tips and Tricks [Електронний ресурс]. – Режим доступу: до ресурсу: <https://code.visualstudio.com/docs/getstarted/tips-and-tricks>
27. How to Use Sublime Text: A Quick Overview [Електронний ресурс]. – Режим доступу до ресурсу: <https://kinsta.com/blog/how-to-use-sublime-text/>
28. Що таке прогресивні веб-додатки [Електронний ресурс] - Режим доступу до ресурсу: <https://smile-ukraine.com/ua/pwa/introduction>
29. Web-додатки – переваги і недоліки [Електронний ресурс] - Режим доступу до ресурсу: https://mydiv.net/arts/view-web-dodatki_perevagy_i_nedliki.html
30. Розробка Web-додатків [Електронний ресурс] - Режим доступу до ресурсу: <http://clashdash/directions/web-applications>

31. Що таке Progressive Web App та в чому їх переваги [Електронний ресурс] - Режим доступу до ресурсу: <https://timeweb.com/community/articles/chto-takoeprogressive-web-apps>
32. Все що необхідно знати про Progressive Web App [Електронний ресурс] - Режим доступу до ресурсу: <https://habr.com/company/wrike/blog/481240/>
33. Методи життєвого циклу [Електронний ресурс] - Режим доступу до ресурсу: <https://dev-gang/article/metody-cikla-komponenta-vreactjs-m3v6725v7q/>
34. Node.js Documentation [Електронний ресурс] - Режим доступу до ресурсу: <https://nodejs.org/dist/latest-v14.x/docs/api/>
35. TypeScript [Електронний ресурс] - Режим доступу до ресурсу: <https://www.typescriptlang.org/>
36. Тестування ПЗ [Електронний ресурс] - Режим доступу: https://uk.wikipedia.org/wiki/Тестування_програмного_забезпечення.
37. Single-page application vs. multiple-page application [Електронний ресурс] - Режим доступу до ресурсу: <https://medium.com/@NeotericEU/single-pageapplication-vs-multiple-page-application-2591588efe58>
38. Xcode [Електронний ресурс] // Режим доступу: <https://en.wikipedia.org/wiki/Xcode>
39. Kotlin [Електронний ресурс] // Режим доступу: <https://uk.wikipedia.org/wiki/Kotlin>
40. Android studio [Електронний ресурс] // Режим доступу: https://uk.wikipedia.org/wiki/Android_Studio
41. Андерс Йоранссон Ефективне використання потоків в операційній системі Android
42. Harris Drucker, Yann Le Cun. Improving Generalization Performance Using Back-propagation / Harris Drucker, Yann Le Cun // IEEE Transactions on Neural Networks. - 1992. - Vol.3, №5 - p. 991-997.

ДОДАТОК А

Схема компонентів застосунку



ДОДАТОК Б

Лістинг DiaryActivity.java

```

public class DiaryActivity extends AppCompatActivity {
    Profile profile;
    Day viewDay;
    DataBase db;
    SimpleAdapter dishesAdapter;
    ArrayList<Map<String, Object>> dishesData;
    SimpleAdapter exercisesAdapter;
    ArrayList<Map<String, Object>> exercisesData;
    private static final int DATE_DIALOG = 1;
    private static final int VIEW_DISHES_DIALOG = 2;
    private static final int VIEW_EXERCISES_DIALOG = 3;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_diary);
        findViewById(R.id.from_diary_to_menu).setOnClickListener(new
        ToWindowOnClickWithClosing(this, MyMenuActivity.class));
        findViewById(R.id.diary_set_date).setOnClickListener(new
        View.OnClickListener() {
            @Override
            public void onClick(View v) {
                showDialog(DATE_DIALOG);
            }
        });
        findViewById(R.id.diary_left_arrow).setOnClickListener(new
        View.OnClickListener() {

```

```

@Override
public void onClick(View v) {
    changeViewDay(viewDay.getPreviousDay(db).getDate());
}
});
findViewById(R.id.diary_right_arrow).setOnClickListener(new
View.OnClickListener() {
@Override
public void onClick(View v) {
    changeViewDay(viewDay.getNextDay(db).getDate());
}
});
findViewById(R.id.diary_dish_add_button).setOnClickListener(new
ToWindowOnClick(this, PickDishActivity.class) {
@Override
public void onClick(View v) {
    PickDishActivity.date = viewDay.getDate();
    super.onClick(v);
}
});
findViewById(R.id.diary_exercise_add_button).setOnClickListener(new
ToWindowOnClick(this, PickExerciseActivity.class) {
@Override
public void onClick(View v) {
    PickExerciseActivity.date = viewDay.getDate();
    super.onClick(v);
}
});
profile = Profile.getProfile(this);

```

```

db = DataBase.getDataBase(this);
setViewDay(new MyDate()); // set current date
dishesData =
DataBase.cursorToArrayList(db.getAllDayDishes(viewDay.getDate()));
String[] from = new String[]{DataBase.DISH_COLUMN_NAME,
DataBase.DAYS_DISH_COLUMN_WEIGHT}; //columns names
int[] to = new int[]{R.id.db_item_name, R.id.db_item_right_text}; // places to
write (View id)
dishesAdapter = new SimpleAdapter(this, dishesData, R.layout.database_item,
from, to);
exercisesData =
DataBase.cursorToArrayList(db.getAllDayExercises(viewDay.getDate()));
from = new String[]{DataBase.EXERCISE_COLUMN_NAME,
DataBase.DAYS_EXERCISE_COLUMN_QUANTITY}; //columns names
exercisesAdapter = new SimpleAdapter(this, exercisesData,
R.layout.database_item, from, to);
}
@Override
protected Dialog onCreateDialog(int id) {
AlertDialog.Builder adb;
switch (id) {
case DATE_DIALOG:
Calendar calendar = Calendar.getInstance();
calendar.setTime(viewDay.getDate());
return new DatePickerDialog(this, myCallBack,
calendar.get(Calendar.YEAR), calendar.get(Calendar.MONTH),
calendar.get(Calendar.DAY_OF_MONTH));
case VIEW_DISHES_DIALOG:
adb = new AlertDialog.Builder(this);

```

```

adb.setTitle(getString(R.string.eaten_dishes));
adb.setAdapter(dishesAdapter, null);
return adb.create();
case VIEW_EXERCISES_DIALOG:
adb = new AlertDialog.Builder(this);
adb.setTitle(getString(R.string.exercises_per_day));
adb.setAdapter(exercisesAdapter, null);
return adb.create();
}
return super.onCreateDialog(id);
}

@Override
protected void onPrepareDialog(int id, Dialog dialog) {
super.onPrepareDialog(id, dialog);
switch (id) {
case DATE_DIALOG:
Calendar calendar = Calendar.getInstance();
calendar.setTime(viewDay.getDate());
((DatePickerDialog) dialog).updateDate(calendar.get(Calendar.YEAR),
calendar.get(Calendar.MONTH), calendar.get(Calendar.DAY_OF_MONTH));
break;
case VIEW_DISHES_DIALOG:
break;
case VIEW_EXERCISES_DIALOG:
break;
}
}

DatePickerDialog.OnDateSetListener myCallBack = new
DatePickerDialog.OnDateSetListener() {

```

```

public void onDateSet(DatePicker view, int year, int monthOfYear,
int dayOfMonth) {
    String newDate = dayOfMonth + "." + (monthOfYear + 1) + "." + year;
    try {
        changeViewDay(new MyDate(Day.format.parse(newDate)));
    } catch (ParseException e) {
        Toast.makeText(DiaryActivity.this, getString(R.string.invalid_date),
        Toast.LENGTH_SHORT).show();
    } catch (NullPointerException e) {
        Toast.makeText(DiaryActivity.this, getString(R.string.invalid_year),
        Toast.LENGTH_SHORT).show();
    }
}

};

public void onViewDayDishes(View v) {
    showDialog(VIEW_DISHES_DIALOG);
}

public void onViewDayExercises(View v) {
    showDialog(VIEW_EXERCISES_DIALOG);
}

private void changeViewDay(MyDate date) {
    saveViewDay();
    setViewDay(date);
}

private void saveViewDay()//закончить
{
    viewDay.setRecord(((EditText)
    findViewById(R.id.diary_record)).getText().toString());
    for (Map<String, Object> dish : dishesData)

```

```

viewDay.addDish(db.getDish((String)
dish.get(DataBase.DISH_COLUMN_NAME)), Integer.parseInt((String)
dish.get(DataBase.DAYS_DISH_COLUMN_WEIGHT)));
for (Map<String, Object> exercise : exercisesData)
viewDay.addExercise(db.getExercise((String)
exercise.get(DataBase.EXERCISE_COLUMN_NAME)), Integer.parseInt((String)
exercise.get(DataBase.DAYS_EXERCISE_COLUMN_TIME)),
Integer.parseInt((String)
exercise.get(DataBase.DAYS_EXERCISE_COLUMN_QUANTITY)));
db.saveDay(viewDay);
}

private void setViewDay(MyDate date) {
setViewDay(Day.getDayByDate(db, date));
}

private void setViewDay(Day day) {
viewDay = day;
if (dishesData != null) {
dishesData.clear();
dishesData.addAll(DataBase.cursorToArrayList(db.getAllDayDishes(viewDay.getDate(
)))));
dishesAdapter.notifyDataSetChanged();
}

if (exercisesData != null) {
exercisesData.clear();
exercisesData.addAll(DataBase.cursorToArrayList(db.getAllDayExercises(viewDay.getDate(
)))));
exercisesAdapter.notifyDataSetChanged();
}

```

```

    }
    ((TextView)
    findViewById(R.id.diary_weekday)).setText(Day.getDayOfWeekByDate(day.getDate()
    ));
    ((EditText)
    findViewById(R.id.diary_set_date)).setText(Day.format.format(day.getDate()));
    ((EditText) findViewById(R.id.diary_record)).setText(day.getRecord());
    updateCaloriesRow();
    }
    protected void onRestart() {
    dishesData.clear();
    dishesData.addAll(DataBase.cursorToArrayList(db.getAllDayDishes(viewDay.getDate(
    ))));
    dishesAdapter.notifyDataSetChanged();
    exercisesData.clear();
    exercisesData.addAll(DataBase.cursorToArrayList(db.getAllDayExercises(viewDay.get
    Date())));
    exercisesAdapter.notifyDataSetChanged();
    updateCaloriesRow();
    super.onRestart();
    }
    private void updateCaloriesRow() {
    Integer receivedCalories = viewDay.getReceivedCalories();
    Integer spentCalories = viewDay.getSpentCalories(profile.getWeight());
    ((TextView)
    findViewById(R.id.diary_calorie_get)).setText(receivedCalories.toString());

```

```
((TextView)
findViewById(R.id.diary_calorie_spend)).setText(spentCalories.toString());
((TextView) findViewById(R.id.diary_calorie_need)).setText(((Integer)
(profile.getAimCalorie() + spentCalories - receivedCalories)).toString());
findViewById(R.id.diary_calorie_get).requestLayout();
findViewById(R.id.diary_calorie_spend).requestLayout();
}
@Override
super.onDestroy();
}
```

ДОДАТОК В

Лістинг activity_dish.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    style="@style/background_style"
    tools:context="com.vladnamik.developer.activities.DishActivity">
    <RelativeLayout
        style="@style/header_style">
        <TextView
            style="@style/header_text"
            android:text="@string/dishes">
        </TextView>
        <Button
            style="@style/header_button"
            android:id="@+id/from_dishes_to_menu"
            android:text="@string/to_menu">
        </Button>
    </RelativeLayout>
    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="60dp"
        android:orientation="horizontal"
        android:background="@color/my_db_menu_color"
        android:id="@+id/db_dish_menu">
        <ImageButton
            android:layout_width="40dp"
            android:layout_height="40dp"
            android:background="@null"
```

```

android:scaleType="centerInside"
android:layout_marginLeft="8dp"
android:layout_marginStart="8dp"
android:layout_centerVertical="true"
android:id="@+id/db_dish_add"
android:onClick="onCreateDish"
android:src="@drawable/add_item"
android:contentDescription="@string/create" />

```

Лістинг Г.2 — activity_my_menu.xml

```

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    tools:context="com.vladnamik.developer.activities.MyMenuActivity"
    app:layout_behavior="@string/appbar_scrolling_view_behavior"
    style="@style/background_style">
    <RelativeLayout
        style="@style/header_style">
        <TextView
            style="@style/header_text"
            android:text="@string/menu">
        </TextView>
        <Button
            style="@style/header_button"
            android:id="@+id/from_menu_to_diary"
            android:text="@string/to_diary">
        </Button>
    </RelativeLayout>
    <ScrollView
        android:layout_width="match_parent"

```

```

    android:layout_height="match_parent"
    android:id="@+id/scroll_view_menu">
    <LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:orientation="horizontal">
    <View
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_weight=".16"/>
    <LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:layout_weight=".1">
    <Button
    android:id="@+id/from_menu_to_diary_menu"
    style="@style/menu_button"
    android:text="@string/diary"
    >
    </Button>

```

ДОДАТОК Г

Лістинг Gradle Scripts

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 27
    buildToolsVersion "27.0.3"
    defaultConfig {
        applicationId "com.vladnamik.developer"
        minSdkVersion 15
        targetSdkVersion 27
        versionCode 1
        versionName "1.0"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguardrules.
            pro'
        }
    }
}

dependencies {
    api fileTree(dir: 'libs', include: ['*.jar'])
    testImplementation 'junit:junit:4.12'
    api 'com.android.support:appcompat-v7:27.1.1'
    api 'com.android.support:design:27.1.1'
}

buildscript {
    repositories {
```

```
mavenLocal()
maven { url 'https://maven.google.com' }
mavenCentral()
jcenter()
google()
}
dependencies {
    classpath 'com.android.tools.build:gradle:3.1.2'
}
}
allprojects {
    repositories {
        mavenLocal()
        maven { url 'https://maven.google.com' }
        mavenCentral()
        jcenter()
        google()
    }
}
```

ДОДАТОК Д

Лістинг android_calorie_counter.iml

```

<?xml version="1.0" encoding="UTF-8"?>
<module external.linked.project.id=":app"
external.linked.project.path="$MODULE_DIR$"
external.root.project.path="$MODULE_DIR$/.." external.system.id="GRADLE"
external.system.module.group="MyApplication"
external.system.module.version="unspecified" type="JAVA_MODULE"
version="4">
<component name="FacetManager">
<facet type="android-gradle" name="Android-Gradle">
<configuration>
<option name="GRADLE_PROJECT_PATH" value=":app" />
</configuration>
</facet>
<facet type="android" name="Android">
<configuration>
<option name="SELECTED_BUILD_VARIANT" value="debug" />
<option name="ASSEMBLE_TASK_NAME" value="assembleDebug" />
<option name="COMPILE_JAVA_TASK_NAME"
value="compileDebugSources" />
<afterSyncTasks>
<task>generateDebugSources</task>
</afterSyncTasks>
<option name="ALLOW_USER_CONFIGURATION" value="false" />
<option name="MANIFEST_FILE_RELATIVE_PATH"
value="/src/main/AndroidManifest.xml" />
<option name="RES_FOLDER_RELATIVE_PATH" value="/src/main/res" />

```

```

<option name="RES_FOLDERS_RELATIVE_PATH"
value="file://$MODULE_DIR$/src/main/res" />
<option name="ASSETS_FOLDER_RELATIVE_PATH"
value="/src/main/assets" />
</configuration>
</facet>
</component>
<component name="NewModuleRootManager"
LANGUAGE_LEVEL="JDK_1_7">
<output url="file://$MODULE_DIR$/build/intermediates/classes/debug" />
<output-test url="file://$MODULE_DIR$/build/intermediates/classes/test/debug" />
<exclude-output />
<content url="file://$MODULE_DIR$">
<sourceFolder url="file://$MODULE_DIR$/build/generated/source/apt/debug"
isTestSource="false" generated="true" />
<sourceFolder url="file://$MODULE_DIR$/build/generated/source/r/debug"
isTestSource="false" generated="true" />
<sourceFolder url="file://$MODULE_DIR$/build/generated/source/aidl/debug"
isTestSource="false" generated="true" />
<sourceFolder
url="file://$MODULE_DIR$/build/generated/source/buildConfig/debug"
isTestSource="false" generated="true" />
<sourceFolder url="file://$MODULE_DIR$/build/generated/source/rs/debug"
isTestSource="false" generated="true" />
<sourceFolder url="file://$MODULE_DIR$/build/generated/res/rs/debug"
type="java-resource" />
<sourceFolder url="file://$MODULE_DIR$/build/generated/res/resValues/debug"
type="java-resource" />
<sourceFolder

```

```

url="file://$MODULE_DIR$/build/generated/source/apt/androidTest/debug"
isTestSource="true" generated="true" />
<sourceFolder
url="file://$MODULE_DIR$/build/generated/source/r/androidTest/debug"
isTestSource="true" generated="true" />
<sourceFolder
url="file://$MODULE_DIR$/build/generated/source/aidl/androidTest/debug"
isTestSource="true" generated="true" />
<sourceFolder
url="file://$MODULE_DIR$/build/generated/source/buildConfig/androidTest/debug
"
isTestSource="true" generated="true" />
<sourceFolder
url="file://$MODULE_DIR$/build/generated/source/rs/androidTest/debug"
isTestSource="true" generated="true" />
<sourceFolder
url="file://$MODULE_DIR$/build/generated/res/rs/androidTest/debug"
type="javatest-
resource" />
<sourceFolder
url="file://$MODULE_DIR$/build/generated/res/resValues/androidTest/debug"
type="java-test-resource" />
<sourceFolder url="file://$MODULE_DIR$/build/generated/source/apt/test/debug"
isTestSource="true" generated="true" />
<sourceFolder url="file://$MODULE_DIR$/src/debug/res" type="java-resource" />
<sourceFolder url="file://$MODULE_DIR$/src/debug/resources"
type="javaresource"
/>
<sourceFolder url="file://$MODULE_DIR$/src/debug/assets" type="java-resource"

```

```

/>
<sourceFolder url="file://$MODULE_DIR$/src/debug/aidl" isTestSource="false" />
<sourceFolder url="file://$MODULE_DIR$/src/debug/java" isTestSource="false"
/>
<sourceFolder url="file://$MODULE_DIR$/src/debug/rs" isTestSource="false" />
<sourceFolder url="file://$MODULE_DIR$/src/debug/shaders"
isTestSource="false" />
<sourceFolder url="file://$MODULE_DIR$/src/testDebug/res" type="java-
testresource"
/>
<sourceFolder url="file://$MODULE_DIR$/src/testDebug/resources" type="javatest-
resource" />
<sourceFolder url="file://$MODULE_DIR$/src/testDebug/assets" type="java-
testresource"
/>
<sourceFolder url="file://$MODULE_DIR$/src/testDebug/aidl"
isTestSource="true" />
<sourceFolder url="file://$MODULE_DIR$/src/testDebug/java"
isTestSource="true" />
<sourceFolder url="file://$MODULE_DIR$/src/testDebug/rs" isTestSource="true"
/>
<sourceFolder url="file://$MODULE_DIR$/src/testDebug/shaders"
isTestSource="true" />
<sourceFolder url="file://$MODULE_DIR$/src/androidTestDebug/res"
type="javatest-
resource" />
<sourceFolder url="file://$MODULE_DIR$/src/androidTestDebug/resources"
type="java-test-resource" />
<sourceFolder url="file://$MODULE_DIR$/src/androidTestDebug/assets"

```

```
type="java-test-resource" />  
<sourceFolder url="file://$MODULE_DIR$/src/androidTestDebug/aidl"  
isTestSource="true" />  
<sourceFolder url="file://$MODULE_DIR$/src/androidTestDebug/java"  
isTestSource="true" />  
<sourceFolder url="file://$MODULE_DIR$/src/androidTestDebug/rs"  
isTestSource="true" />  
<sourceFolder url="file://$MODULE_DIR$/src/androidTestDebug/shaders"  
isTestSource="true" />
```