МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Кваліфікаційна наукова
праця на правах рукопису

Гаваньо Богдан Іванович

УДК 004.04:004.4:004.67:004.75

ДИСЕРТАЦІЯ

МЕТОДИ ТА ЗАСОБИ ОЦІНЮВАННЯ СТАНУ ЛЮДИНИ В МЕДИЧНИХ КІБЕРФІЗИЧНИХ СИСТЕМАХ

123 – «Комп’ютерна інженерія»

12 – «Інформаційні технології»

Подається на здобуття наукового ступеня доктора філософії

Дисертація містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Гаваньо Б.І.

Науковий керівник:
Кицун Геннадій Васильович
кандидат технічних наук

Львів-2021

АНОТАЦІЯ

Гаваньо Б.І. Методи та засоби оцінювання стану людини в медичних кіберфізичних системах. – Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття наукового ступеня доктора філософії за спеціальністю 123 «Комп'ютерна інженерія» (12 «Інформаційні технології»). – Національний університет «Львівська політехніка», Львів, 2021.

Дисертація присвячена розв'язанню актуального науково-технічного завдання розроблення методів та засобів оцінювання стану людини на основі обробки виміряних життєвих показників сенсорами в медичній кіберфізичній системі.

У вступі обґрунтовано актуальність теми дисертаційного дослідження, сформульовано мету дослідження та науково-технічні завдання, необхідні для її досягнення, показано зв'язок дослідження з науковими програмами та темами, наведено наукову новизну отриманих результатів, їх практичну цінність та особистий внесок здобувача, надано інформацію про апробацію результатів роботи.

В першому розділі проведено аналіз існуючих підходів до побудови медичних кіберфізичних систем, а саме їх структури та архітектурних рішень серверного програмного забезпечення. Аналіз показав, що при проектуванні та реалізації медичних КФС, потрібно враховувати, що медичні дані є великими даними, а їх обробка повинна відбуватись паралельно. Згідно проведеному аналізу визначено архітектурні та функціональні вимоги до медичних КФС. Проведено аналіз принципів зберігання даних в медичних КФС. Постановлено задачі оцінювання стану людини.

У другому розділі запропоновано засоби обробки інформації в медичних кіберфізичних системах, результат виконання функцій яких є оцінюванням стану людини. Запроповано модель обробки інформації в медичних КФС. Для розмежування засобів обробки інформації на окремі незалежні логічні елементи

запропонована модель обробки інформації базується на основі мікросервісної архітектури, де кожен засіб є у вигляді мікросервісу. Запропоновано метод виявлення критичних подій в медичній кіберфізичній системі, який базується на одному з найбільш використовуваних посібників EWS та агрегованій оцінці життєвих показників стану людини. Запропоновано алгоритм класифікації пацієнтів та алгоритм діагностики захворювань.

У третьому розділі, згідно запропонованим моделі та засобам обробки інформації, пропонується структура медичної кіберфізичної системи. З даної структури виділено запропоновану архітектурно-інформаційну модель обробки інформації, яка базується на моделі обробки даних, в основі якої лежить мікросервісна архітектура з використанням API шлюзу. Проведено опис функціональних елементів архітектурно-інформаційної моделі обробки інформації. Чітка взаємодія функціональних елементів в архітектурно-інформаційній моделі обробки інформації є засобос оцінювання стану людини в медичній кіберфізичній системі. Описано бізнес логіку роботи медичної КФС, а саме систему зв’язків та залежностей елементів бізнес-даних та правил обробки цих даних.

У четвертому розділі описано реалізацію серверного програмного забезпечення медичної кіберфізичної системи, яке базується на запропонованій архітектурно-функціональній моделі обробки інформації. Реалізовано засіб автентифікації та авторизації користувачів; сервіс декларування фонових задач; засіб сповіщення про події, використовуючи 2 канали зв’язку; засіб виявлення критичних показників, в основі якого лежить запропонований метод виявлення критичних показників; засіб, який включає адаптовані алгоритми класифікації пацієнтів та діагностики захворювань; засоби обробки та агрегації виміряних показників. Описано принципи організації зберігання даних в реалізованій медичній кіберфізичній системі, з використанням сховищ різного типу, в залежності від потреби засобів обробки інформації. Описано принцип роботи

медичної кіберфізичної системи «HealthyLungs», в якій впроваджувались запропоновані методи, засоби та модель обробки інформації.

Ключові слова: медична кіберфізична система, оцінювання стану людини, життєві показники, обробка та зберігання медичних даних, мікросервісна архітектура, моніторинг стану людини, виявлення критичних показників.

ABSTRACT

Havano B.I. Methods and means of assessing the human condition in medical cyberphysical systems. – Qualification scientific work on the rights of a manuscript.

The thesis for the Philosophy Doctor (Ph.D.) degree in specialty 123 «Computer Engineering» (12 «Information Technology»). – Lviv Polytechnic National University, Lviv, Ukraine, 2021

The thesis is devoted to the solution of the actual scientific and technical task of development of methods and means of assessing the human condition on the basis of processing of the measured vital indicators by sensors in medical cyberphysical systems.

The introduction substantiates the relevance of the topic of dissertation research, formulates the purpose of research and scientific and technical tasks necessary for its achievement, shows the connection of research with scientific programs and topics, presents the scientific novelty of the results, their practical value and personal contribution of the applicant. approbation of work results.

The first section analyzes the existing approaches to the construction of medical cyberphysical systems, namely their structure and architectural solutions of server software. The analysis showed that in the design and implementation of medical CPS, it should be borne in mind that medical data is big data, and their processing should take place in parallel. According to the analysis, the architectural and functional requirements for medical CPS are determined. The analysis of the principles of data storage in medical CPS is carried out. The tasks of assessing the human condition are set.

The second section proposes means of information processing in medical cyberphysical systems, the result of which is to assess the human condition. A model of information processing in medical CPS is proposed. To differentiate the means of information processing into separate independent logical elements, the proposed model of information processing is based on a microservice architecture, where each tool is

in the form of a microservice. A method for detecting critical events in the medical cyberphysical system has been proposed, which is based on one of the most widely used EWS manuals and an aggregated assessment of human vital signs. An algorithm for classifying patients and an algorithm for diagnosing diseases are proposed.

In the third section, according to the proposed model and means of information processing, the structure of the medical cyberphysical system is proposed. From this structure the offered architectural-information model of information processing which is based on model of data processing which is based on microservice architecture with use of API of the gateway is allocated. The description of functional elements of architectural-information model of information processing is carried out. Clear interaction of functional elements in the architectural-information model of information processing is a means of assessing the human condition in the medical cyberphysical system. The business logic of medical CPS is described, namely the system of connections and dependencies of business data elements and rules of data processing.

The fourth section describes the implementation of server software for medical cyberphysical system, which is based on the proposed architectural and functional model of information processing. Implemented a means of authentication and authorization of users; background task declaration service; event notification tool using 2 communication channels; a means of detecting critical indicators, which is based on the proposed method of detecting critical indicators; a mean that includes adapted algorithms for patient classification and disease diagnosis; means of processing and aggregation of measured indicators. The principles of the organization of data storage in the realized medical cyberphysical system, with use of storages of different type, depending on need of means of information processing are described. The principle of operation of the medical cyberphysical system "HealthyLungs" is described, in which the proposed methods, means and model of information processing were implemented.

Key words: medical cyberphysical system, assessment of human condition, vital signs, processing and storage of medical data, microservice architecture, human condition monitoring, detection of critical indicators.

СПИСОК ПУБЛІКАЦІЙ ЗА ТЕМОЮ ДИСЕРТАЦІЇ

1. Melnyk Anatoliy, Morozov Yuriy, Havano Bohdan, Hupalo Petro. Investigation of wireless pulse oximeters for smartphone-based remote monitoring of lung health // Advances in Cyber-Physical Systems. – 2020. – Vol. 5, № 2. – p. 70–76.
2. А. О. Мельник, Ю. В. Морозов, Б. І. Гаваньо, П. А. Гупало. Біомедична кіберфізична система цілодобового моніторингу функцій легень у пацієнтів із COVID-19 // Комп'ютерні системи та мережі. — Львів : Видавництво Львівської політехніки, 2020. — Том 2. — № 1. — С. 1–5.
3. Anatoliy Melnyk, Yurii Morozov, Bohdan Havano, Petro Hupalo HealthSupervisor: Mobile Application for Round-the-Clock Remote Monitoring of the Human Functional State // The 2nd International Workshop on Intelligent Information Technologies & Systems of Information Security (IntelITSIS-2021). CEUR-WS, 2021, Vol-2853, p. 24-37
4. Гаваньо Б. І. Assessing the Human Condition in Medical Cyber-Physical System Based on Microservices Architecture // Досягнення у кіберфізичних системах. – 2021. – vol 6, issue 2. – С. 112–120

Наукові праці, які засвідчують апробацію матеріалів дисертації:

5. А. О. Мельник, Ю. В. Морозов, Б. І. Гаваньо, П. А. Гупало. Мобільні додатки для цілодобового віддаленого моніторингу лікарями функцій легень пацієнтів // Сучасні проблеми і досягнення в галузі радіотехніки, телекомунікацій та інформаційних технологій : тези доповідей X Міжнародної науково-практичної конференції, 07–09 жовтня 2020 р., м. Запоріжжя. – 2020. – С. 83–84
6. Havano B. Problems of privacy and security in cyber physical systems of intellectual houses / Bohdan Havano // Litteris et Artibus : proceedings, 23–25 November 2018 (10th International academic conference "Computer science & engineering 2018"), Lviv. — Lviv : Lviv Politechnic Publishing House, 2018. — P. 58–59.

Наукові праці, які додатково відображають наукові результати дисертації:

7. Havano Bohdan, Kytsun Hennadiy, Tkachyk Oleksandr. Web-server cross-site request forgery protection // Perspectives of science and education : proceedings of the 7th International youth conference, 10th May, 2020 New York, USA. – 2020. – С. 9–16
8. Гаваньо Б. І. Проблеми конфіденційності та безпеки в кіберфізичних системах інтелектуальних будинків // Вісник Національного університету “Львівська політехніка”. Серія: Комп’ютерні системи та мережі. 2018. № 905. С. 49–55.
9. Valerii Hlukhov, Bohdan Havano. FPGA-based Digital Quantum Coprocessor. Advances in Cyber-Physical Systems. Volume 3. Number 2. Lviv Polytechnic National University. 2018. p. 12–31.
10. Valerii Hlukhov, Bohdan Havano. Principles of Digital Quantum Coprocessor Based on a FPGA, which Operates under the Control of a Classical Computer // 2019 9th International Conference on Advanced Computer Information Technologies (ACIT). IEEE, 2019, p. 191-194
11. Патент України на корисну модель №148157, «Біомедична система для цілодобового віддаленого моніторингу уповноваженою особою показників функціонального стану організму клієнта», заявка №u202008053 від 16.12.2020, А. О. Мельник, Ю. В. Морозов, Б. І. Гаваньо, П. А. Гупало.

Зміст

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ 12
ВСТУП 13
РОЗДІЛ 1 19
АНАЛІЗ ТЕХНОЛОГІЙ МЕДИЧНИХ КІБЕРФІЗИЧНИХ СИСТЕМ 19
1.1. Аналіз структури медичної кіберфізичної системи 20
1.2. Архітектурні та функціональні вимоги до медичних кіберфізичних систем 48
1.3. Принципи зберігання даних в медичних кіберфізичних системах 49
1.4. Постановка задачі оцінювання стану людини в медичних кіберфізичних системах 54
1.5. Висновки до розділу 56
РОЗДІЛ 2 57
МЕТОДИ, ЗАСОБИ ТА МОДЕЛЬ ОЦІНЮВАННЯ СТАНУ ЛЮДИНИ В МЕДИЧНІЙ КІБЕРФІЗИЧНІЙ СИСТЕМІ 57
2.1. Засоби обробки інформації в медичних кіберфізичних системах 57
2.2. Модель обробки інформації в медичних кіберфізичних системах 60
2.3. Метод виявлення критичних показників в медичній кіберфізичній системі. 65
2.4. Алгоритм класифікації пацієнтів 68
2.5. Алгоритм діагностики захворювань 71
2.6. Висновки до розділу 75
РОЗДІЛ 3 77
АРХІТЕКТУРНО-ІНФОРМАЦІЙНА МОДЕЛЬ ОЦІНЮВАННЯ СТАНУ ЛЮДИНИ В МЕДИЧНІЙ КІБЕРФІЗИЧНІЙ СИСТЕМІ 77
3.1. Архітектурно-інформаційна модель обробки інформації 77
3.2. Функціональні елементи архітектурно-інформаційної моделі обробки інформації 79

3.3. Бізнес логіка в роботі архітектурно-інформаційної моделі обробки інформації медичної кіберфізичної системи. 94

3.4. Висновки до розділу 105

РОЗДІЛ 4 106

РЕАЛІЗАЦІЯ СЕРВІСІВ МЕДИЧНОЇ КІБЕРФІЗИЧНОЇ СИСТЕМИ ДЛЯ ОЦІНЮВАННЯ СТАНУ ЛЮДИНИ 106

4.1. Архітектура та функціонування програмного забезпечення медичної кіберфізичної системи. 106

4.2. Організація зберігання даних в медичній кіберфізичній системі. 124

4.3. Медична кіберфізична система «HealthyLungs». 129

4.4. Висновки до розділу 135

ВИСНОВКИ 137

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 139

ДОДАТОК А. СПИСОК ПУБЛІКАЦІЙ ЗА ТЕМОЮ ДИСЕРТАЦІЇ 147

ДОДАТОК Б. АКТ ВПРОВАДЖЕННЯ РЕЗУЛЬТАТІВ ДИСЕРТАЦІЙНОГО ДОСЛІДЖЕННЯ 149

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

КФС – Кіберфізична система.

МКФС – Медична кіберфізична система.

ІКТ – Інформаційно-комунікаційні технології.

IoT – Internet of Things. Інтернет речей.

WSAN – Wireless sensor-actuator network. Бездротова сенсор-актуаторна мережа.

SOA – Service-oriented architecture. Сервісно-орієнтована архітектура.

REST – Representational State Transfer. Передача репрезентативного стану.

EMR – Electronic medical record. Електронний медичний запис.

API – Application Programming Interface. Прикладний програмний інтерфейс.

ЕКГ – Електрокардіографія.

BSN – Body sensors network. Мережа біосенсорів.

SQL – Structured query language. Мова структурованих запитів.

СУБД – Система управління базами даних.

HDFS – Hadoop Distributed File System. Розподілена файлова система Hadoop.

БД – База даних.

BFF – Backend for frontend. Паттерн розробки API шлюзу.

JWT – JSON Web Token. Стандарт токена доступу.

SSL – Secure Sockets Layer. Протокол захищених сокетів.

ВСТУП

Актуальність теми дисертації. Кіберфізичні системи (КФС) відносяться до наступного покоління систем ІКТ (інформаційно-комунікаційних технологій), які в основному інтегрують вимірювання, обчислення та комунікації для моніторингу, контролю та взаємодії з фізичними процесами, щоб забезпечити громадян та бізнес інтелектуальними програмами та послугами, наприклад: охорона здоров’я, розумні будинки, Індустрія 4.0 тощо. В останні роки охорона здоров’я стала однією з найважливіших послуг завдяки постійному зростанню її витрат. Це мотивувало великі дослідження з питань охорони здоров’я в медичних кіберфізичних системах, і деякі з цих досліджень були зосереджені на описі програмної архітектури, що лежить в основі цих систем.

Охорона здоров'я стає однією з найважливіших проблем для урядів, оскільки Всесвітня організація охорони здоров'я попереджає про збільшення витрат на догляд [3], наприклад, в Європі, враховуючи старіння населення. Серцево-судинні або хронічні обструктивні захворювання легень є основними причинами смерті в Європейському Союзі [3]. Також, однією з таких проблем стала пандемія COVID-19. На сьогоднішній день в сфері охорони здоров’я застосовується широкий спектр інформаційних рішень - від клінічних інформаційних систем (включаючи електронні медичні записи EMR, електронне призначення та електронне здоров’я загалом), адміністративних систем та систем управління до інформатики та біоінформатики клінічних досліджень. Серед цих рішень системи моніторингу здоров’я можуть відігравати ключову роль у запобіганні різним захворюванням.

Системи моніторингу розширили свої можливості в так званих кіберфізичних системах. Кіберфізичні системи інтегрують можливості вимірювання, оцінювання, зберігання та комунікації для управління та взаємодії з фізичними процесами. Вбудовані системи контролюють і управляють фізичними процесами в реальному (або в близькому до реального) часі, як правило, з циклами зворотного зв’язку, де фізичні процеси впливають на обчислення і навпаки [45]. Консолідація КФС, частково за сприяння буму в

Інтернеті речей (IoT) [29], великих даних та розвитку бездротових сенсорних та актуаторних мереж (WSAN), створює нові бізнес моделі в широкому діапазоні доменів, включаючи охорону здоров'я. Отже, медичні КФС можуть не тільки запобігати захворюванням, але й активно боротися з ними. З одного боку, медична КФС дозволяє контролювати біосигнали за допомогою мініатюрних носимих датчиків (вони ж біосенсори та датчики тіла), які можуть автоматично виявляти розлад або проблему та генерувати тривогу, яку пацієнт або лікар отримує на своєму мобільному телефоні чи комп’ютері без втручання людини. КФС, як правило, характеризується реалізацією циклів зворотного зв'язку з когнітивними та навчальними можливостями [15], таким чином, що ці системи можуть діяти автономно та швидко без втручання людини. З іншого боку, медична КФС може полегшити негайний доступ лікарів та інших медичних працівників до інформації про пацієнтів та медичних записів, а також доступ до даних, що надходять від біосенсорів, для ефективного прийняття рішень та лікування. З цією подвійною метою в останні роки розповсюджуються рішення медичних КФС, що дозволяють віддалено контролювати хронічних хворих, дозволяти пацієнтам брати активну роль у лікуванні, надаючи інформацію та тренінги, а отже, зменшувати витрати на охорону здоров’я, пов’язані з неодноразовими консультаціями з медичними працівниками. Це розповсюдження рішень призводить до потреби проведення досліджень, спрямованих на вдосконалення побудови медичних кіберфізичних систем.

Всесвітня організація охорони здоров’я вказує, що до 2050 року приблизно 2 мільярди людей становитимуть 60 років і старше, а 80% перебуватимуть у слаборозвинених країнах [14]. [38] підкреслює, що, за даними Всесвітньої організації охорони здоров’я, хронічна обструктивна хвороба легень буде третьою за поширеністю причиною смерті в 2030 році. На даний момент, поширюється поточна пандемія коронавірусної хвороби 2019 (COVID-19), спричинена SARS-CoV-2. Крім того, Європейський Союз заявляє, що основними хронічними патологіями є серцево-судинні захворювання, що разом із респіраторними захворюваннями є основною причиною смерті в Європі. Таким

чином, основною мотивацією систем електронного здоров'я є моніторинг пацієнтів, особливо хворих на хронічні захворювання, лікарями та медичними центрами для надання своєчасної дистанційної допомоги [40]. Крім того, чим більший обсяг інформації про пацієнта, тим більша точність прийняття рішень при діагностиці та призначенні методів лікування. Додатки можуть бути широко спеціалізованими, включаючи контроль серцевого нападу, контроль діабету, астму, хворобу Паркінсона, ожиріння, виявлення нещасних випадків, реабілітацію після втручань, нагадування про лікування, емоційне та когнітивне розпізнавання для полегшення соціальних взаємодій, розташування людей з порушеннями орієнтації та медична допомога у разі стихійного лиха, такого як землетрус. Певний тип систем електронного здоров'я, таких як телемедичні системи, набуває великого значення з різних причин, таких як турбота про старіння населення [46], надання своєчасної діагностики та коригування реабілітаційних методів лікування [39]. «Життєво важливі ознаки - це показники, що відображають фізіологічний стан життєво важливих органів (мозок, серце, легені). Вони негайно виражають функціональні зміни, що відбуваються в організмі, зміни, які в іншому випадку не могли б бути визначені або визначені кількісно» [67] і є основними для оцінювання функціонального стану людини.

На теперішній час розробка та дослідження методів та засобів оцінювання стану людини в медичних кіберфізичних системах є актуальним науковим завданням.

Зв'язок роботи з науковими програмами, планами, темами. Дисертаційна робота відповідає науковому напряму кафедри електронних обчислювальних машин Національного університету "Львівська політехніка": "Питання теорії, проектування та реалізації комп'ютерних систем та мереж, а також комп'ютерних засобів, вузлів, приладів і пристроїв вимірювальних, інформаційних, керуючих, телекомунікаційних та кіберфізичних систем".

Дане дослідження проводилось в рамках роботи над проектом "Applications mobiles pour la surveillance 24h /24 de la fonction pulmonaire chez les

patients atteints de COVID-19” за фінансування грантом L’Agence universitaire de la Francophonie AUF по знаходженню шляхів боротьби з COVID-19.

Мета і задачі дослідження. Метою дисертаційної роботи є розробка та дослідження методів та засобів оцінювання функціонального стану організму людини. Для досягнення поставленої мети необхідне розв’язання наступних задач:

- проаналізувати відомі методи та засоби оцінювання стану людини в медичних кіберфізичних системах;
- запропонувати архітектурно-інформаційну модель обробки інформації в медичних кіберфізичних системах для оцінювання стану людини;
- вдосконалити метод виявлення критичних показників в медичній кіберфізичній системі;
- розробити алгоритми діагностики захворювань в медичній кіберфізичній системі;
- розробити нові засоби оцінювання стану організму людини в медичній кіберфізичній системі;
- практично реалізовати медичну кіберфізичну систему оцінювання стану організму людини.

Об’єктом досліджень є процес оцінювання стану людини в медичній кіберфізичній системі.

Предметом досліджень є методи та засоби оцінювання стану людини в медичних кіберфізичних системах.

Методи досліджень. Методи досліджень базуються на принципах системного аналізу (ієрархічності, декомпозиції та інше). Для розв’язання поставлених у дисертаційній роботі задач використано методи об’єктно-орієнтованого програмування, теорії ймовірностей та обчислювальної математики, теорії комп’ютерних систем і мереж, теорії кіберфізичних систем.

Наукова новизна одержаних результатів.

1. Вперше запропоновано архітектурно-інформаційну модель оцінювання стану людини на основі мікросервісної архітектури в медичних КФС, яка на відміну від відомих моделей оцінювання стану людини, дозволяє одночасно забезпечити масштабування, відмовостійкість та підвищення швидкодії оцінювання стану людини.
2. Вдосконалено існуючі методи та алгоритми оцінювання стану людини в медичних КФС, а саме метод виявлення критичних показників стану людини та алгоритми класифікації пацієнтів і діагностики захворювань, шляхом обчислення агрегованої оцінки життєвих показників стану людини та класифікації пацієнтів на її основі, що дає можливість пришвидшити побудову звязків між кластеризованими пацієнтами.
3. Вперше запропоновано засіб організації виконання сервісів оцінювання стану людини, який дає можливість виконувати ці сервіси незалежно і паралельно один від одного, що дає можливість пришвидшити процес оцінювання стану людини, виконуваючи функції сервісів не очікуючи результатів обробки інформації іншими сервісами.

Практичне значення одержаних результатів.

Розроблені методи та засоби оцінювання стану людини в медичних кіберфізичних системах реалізовані у вигляді програмно-алгоритмічного забезпечення, яке може бути застосовано при створенні медичних кіберфізичних систем, систем автоматизованого управління на основі безкабельних сенсорів.

Основні результати теоретичних досліджень дисертації впроваджено в навчальний процес студентів базового напряму “Комп‘ютерна інженерія” Національного університету “Львівська політехніка” у лабораторний практикум з курсу “Автоматизоване проектування комп’ютерних та кіберфізичних систем”; при виконанні науково-дослідницького проекту “Applications mobiles pour la surveillance 24h /24 de la fonction pulmonaire chez les patients atteints de COVID-19” за фінансування AUF.

Особистий внесок здобувача. Основний зміст роботи, всі теоретичні та практичні результати, висновки і дослідження, які представлено до захисту, одержані автором особисто. Роботи [58][59] опубліковані самостійно. У публікаціях, написаних у співавторстві, автору належать: алгоритми класифікації пацієнтів і діагностики захворювань, метод виявлення критичних показників стану людини, засоби автентифікації та авторизації в медичних КФС, застосування квантового комп'ютера для виконання алгоритму класифікації пацієнтів.

Апробація роботи. Основні теоретичні положення та практичні результати дисертаційної роботи доповідалися і обговорювались на семінарах та конференціях: наукових семінарах кафедри «Електронні обчислювальні машини» Національного університету «Львівська політехніка» (2017-2021); 8-ому Міжнародному Форумі Науки "Litteris et Artibus – 2018" (м. Львів, 2018); 9th International conference on advanced computer information technologies (June 5–7, 2019, Ceske Budejovice, Czech Republic); 7th International youth conference "Perspectives of science and education" (Нью Йорк, 15.02.2019); 10-тій Міжнародній науково-практичній конференції «Сучасні проблеми і досягнення в галузі радіотехніки, телекомунікацій та інформаційних технологій» (м. Запоріжжя, 07-09 жовтня 2020).

Публікації. У 11 наукових публікаціях повністю відображені основні результати дисертації, з них: 4 статті у наукових фахових виданнях України; 2 публікації у наукових виданнях, які входять до міжнародних наукометричних баз (з них 1 стаття у науковому періодичному виданні іншої держави); 1 стаття у науковому періодичному виданні України, 3 тези доповідей та матеріали конференцій. Отримано 1 патент на корисну модель.

Структура та обсяг роботи. Повний обсяг дисертації становить 149 сторінок, з яких 120 сторінок основного тексту. Дисертаційна робота складається зі вступу, чотирьох розділів, висновків, списку використаних джерел (77 найменувань) та додатків. Робота містить 31 рисунок, 5 таблиць та 2 додатки.

РОЗДІЛ 1

АНАЛІЗ ТЕХНОЛОГІЙ МЕДИЧНИХ КІБЕРФІЗИЧНИХ СИСТЕМ

Аналіз технологій можна проводити з різних точок зору. Хорошим прикладом є робота Керролла [49], яка представляє модель для оцінки вартості та потенційного впливу медичних КФС з точки зору багатьох зацікавлених сторін. З точки зору програмної інженерії, пропонується оцінити архітектуру програмного забезпечення з наступних причин: архітектури програмного забезпечення є ядром систем; відсутність зосередженості на архітектурі породжує неоптимальні дизайнерські рішення [68]; а неточний архітектурний дизайн призводить до виходу з ладу великих програмних систем [7]. Оскільки архітектура відіграє ключову роль у побудові програмного забезпечення, дослідження архітектури програмного забезпечення для медичних КФС є критичним.

Для визначення вимог до архітектури програмного забезпечення медичних кіберфізичних систем, потрібно виявити, оцінити та інтерпретувати всі наявні відповідні дослідження за допомогою суворої та перевіреної методології [8].

На сьогоднішній день не існує детального огляду архітектур програмного забезпечення медичних КФС, що ускладнює оцінку зрілості сучасних рішень та виявлення тенденцій, прогалин у дослідженнях чи майбутніх вимірів. Як європейські, так і американські ініціативи - наприклад, Програма Європейської комісії "Горизонт 2020" - наголошують на необхідності скласти програму досліджень для медичних КФС, наприклад, щодо ІКТ-рішень для активного та здорового старіння та інновації в галузі електронного здоров’я для розширення можливостей пацієнта.

У міру збільшення кількості життєво важливих ознак, що підлягають моніторингу, ці ознаки є більш складними, і необхідно контролювати складний фізичний процес (наприклад, захворювання та його лікування), з’являються нові рішення. Деякі первинні дослідження описують їх рішення з точки зору кіберфізичних систем [23] [53] [55], Інтернету речей [28] [34] [38] [40] [41] [47]

[51] та інших з точки зору Інтернету майбутнього [44]. Подібно до концепцій КФС та IoT, Інтернет майбутнього (FI) розглядається як система, в якій неоднорідні ресурси передаються один одному для надання послуг реальному світу. Насправді всі ці терміни підкреслюють проблему неоднорідності. У домені КФС / IoT зв'язок відбувається між неоднорідними пристроями (наприклад, датчиками тіла), де велика кількість виробників породила потребу в програмних платформах та проміжних системах, які підтримують інтеграцію таких пристроїв.

1.1. Аналіз структури медичної кіберфізичної системи

Термін кіберфізичні системи виник приблизно в 2006 році, коли він був введений Хелен Гілл у Національному науковому фонді в США [45]. КФС стосується наступного покоління вбудованих систем ІКТ, які взаємопов’язані та спільні (кооперативні та договірні), і які надають користувачам та компаніям широкий спектр інтелектуальних додатків та послуг [15]. З цією метою КФС інтегрує обчислення, зв’язок та управління з об’єктами реального світу та фізичними процесами. КФС має намір відстежувати, контролювати та взаємодіяти з фізичним процесом [45], забезпечуючи ефективні та швидкі петлі зворотного зв'язку між вимірювальними (сенсорами) та виконавчими (актуаторами) механізмами, можливо, з когнітивними та навчальними можливостями [15] та використовуючи автономну поведінку. Щоб відчути фізичне середовище та впливати на нього, КФС часто будуються на сенсорних / акторних мережах, переважно бездротових [3]. КФС все більше і більше взаємопов'язані через IoT [29], який охоплює розширення Інтернету у фізичну сферу, в результаті чого виникає глобальна мережа, яка з'єднує тисячі або навіть мільйони "речей" або розумних об'єктів [37]. Отже, КФС та IoT - це пов’язані терміни: КФС - це більш широкий, більш фундаментальний та довговічний термін, оскільки він не обмежується жодним конкретним впровадженням (наприклад, Інтернетом) [45]. IoT - це мережа для підключення речей, здатних відстежувати та обробляти інформацію для пропонування інтелектуальних

послуг, тоді як КФС включає управління фізичним процесом - таким чином, такі системи взаємодіють з пристроями менеджменту та управління - з Інтернетом як засобом, а не як ядром. Незважаючи на новизну цих концепцій, протягом останніх років КФС / IoT застосовується до безлічі вертикальних доменів, включаючи інтелектуальні мережі, інтелектуальний транспорт, Індустрію 4.0, цифрову охорону здоров'я та розумні міста, які є великими кіберфізичними системами. Загальноприйнята структура КФС виглядає наступним чином (див. Рисунок 1.1) [69].

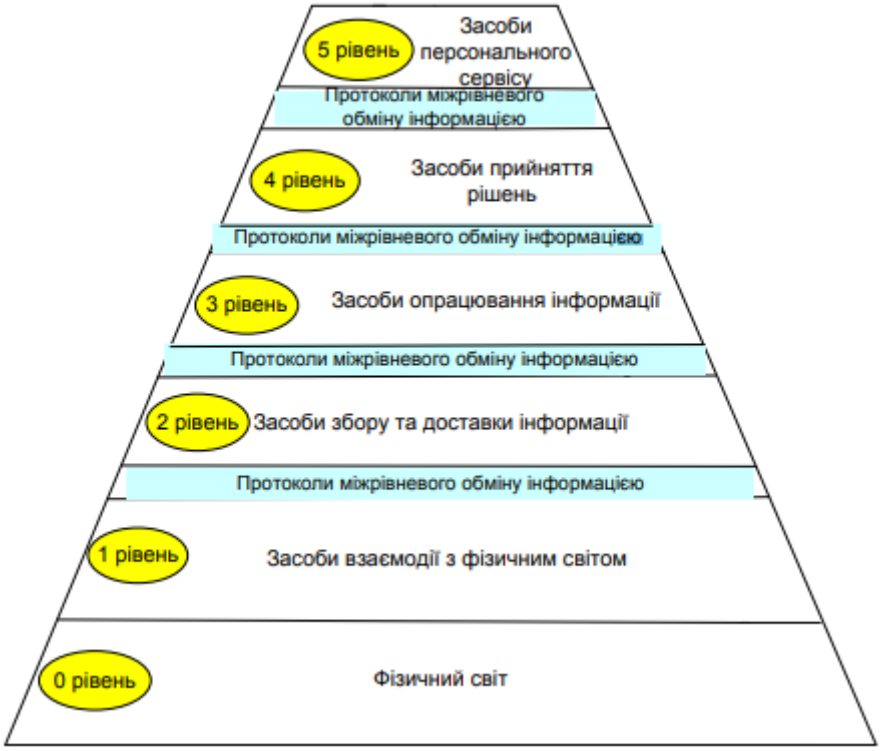

Рисунок 1.1. Багаторівнева базова платформа кіберфізичних систем.

Зосередження уваги на цифровій охороні здоров’я, Connected Health [14] [43] або eHealth (у різних її варіантах, таких як телемедицина, телездоров’я та mHealth), стосується медичних послуг та інформації, метою якої є покращення здоров’я, яка надається або покращується через Інтернет та супутні технології. Це спрямовано на зменшення витрат порівняно з традиційною медичною послугою. Зокрема, телемедицина визначається як використання ІКТ для

надання клінічних послуг на відстані, тоді як mHealth має на меті ту ж мету, але підтримується мобільними пристроями (стільниковими телефонами або цифровими персональними асистентами). Ці терміни ближчі до того, що ми називаємо медичними КФС, але вони не однакові. Медичні КФС - це перетин електронного здоров'я та пов'язаних з ним варіантів із КФС та IoT, які головним чином зосереджуються на моніторингу біосигналів для контролю фізичних процесів та надання розумних медичних послуг. Порівняно з eHealth, медична КФС включає набагато більше фізичних компонентів, і компоненти медичної КФС обмінюються інформацією між собою. КФС має на меті покращити якість медичної допомоги, надаючи спостереження за пацієнтами в режимі реального часу, за якими можна спостерігати та контролювати на відстані та які зможуть отримати швидке реагування на надзвичайні ситуації.

Швидка поява на ринку все менш і менш нав'язливих, мініатюрних (медичних) носимих сенсорів (відомих як біосенсори та датчики тіла), зробила можливим дистанційне спостереження за пацієнтами та відкрила двері для безлічі нових бізнес моделей у галузі охорони здоров'я. Основною функцією датчиків тіла є контроль за життєвими показниками пацієнтів; потім ці дані обробляються та аналізуються програмною системою, яка візуалізує інформацію для лікарів, щоб вони могли надати віддалену допомогу. Найбільш поширеними біосигналами, що підлягають моніторингу, є вібрації, пульс, сатурація киснем крові, звуки, гази, рух або відсутність руху, температура тіла, кров'яний тиск, частота серцевих скорочень, електрокардіограма, гальванічна реакція шкіри (потовиділення) та положення тіла. Найдосконаліші системи можуть надати допомогу в процесі прийняття рішень під час визначення діагнозу та його відповідного лікування, а також можуть забезпечити обгрунтування та автономну поведінку.

Відповідно до [33], медичні кіберфізичні системи класифікуються за додатками (допоміжні, керовані), архітектурою (інфраструктура, потреба даних, система композиції), вимірюванням (тип датчика, метод, параметр), управління

даними (інтеграція, зберігання, обробка даних), обчислення (моделювання, моніторинг), зв'язок (планування, протокол), безпека (конфіденційність, шифрування) та контролем / активацією (прийняття рішень, механізм). Враховуючи важливість архітектури для приховування важливих рішень та характеристик систем, у цій роботі основна увага приділяється архітектурі програмного забезпечення для медичних КФС, а саме програмним системам, які обробляють, аналізують, контролюють та візуалізують біосигнали для надання розумних медичних послуг, що є 2,3 та 4 рівнями структури КФС (див. Рис. 1.1).

1.1.1. Аналіз архітектури програмного забезпечення медичних кіберфізичних систем.

Архітектури програм описують структуру програмних систем шляхом приховування деталей низького рівня та абстрагування важливих функцій високого рівня [2], що відповідає як функціональним, так і нефункціональним вимогам. Проектування, специфікація та аналіз структури програмно-інтенсивних систем стали критичними питаннями при розробці програмного забезпечення [6], з’явившись як рішення для проектування та розробки великих та складних програмних систем. Насправді вже давно визнано, що архітектура має сильний вплив на життєвий цикл системи і є критично важливим елементом успішного розвитку, а також успішної еволюції програмно-інтенсивних систем [10] [13].

Зважаючи на важливість рівня архітектури розробки систем, потрібен надійний консенсус з точним визначенням архітектури системи. Стандарт IEEE 1471-2000 (2007) [15] та наступний стандарт ISO/IEC/IEEE 42010 (2011) забезпечують основу для архітектур програмно-інтенсивних систем з точки зору стандартизації елементів та практик для опису архітектури. Набір концепцій, пов'язаних з архітектурами програмного забезпечення, визначених стандартом IEEE 1471-2000, використовується наступним чином. Система охоплює окремі додатки, системи, підсистеми, системи систем, лінійки продуктів або екосистеми. Система має одну або декілька зацікавлених сторін. Зацікавлена

сторона системи - це особа, команда або організація, що мають інтереси в системі [15]. Фундаментальна організація системи проявляється через її архітектуру: компоненти системи, їх взаємозв'язок між собою та навколишнім середовищем, а також принципи, що керують розробкою та еволюцією системи. Компонент розглядається як чорний ящик, що показує високий рівень інкапсуляції та абстракції, а також взаємодії, з якими він обмежений через свої інтерфейси. Опис архітектур, як правило, організовується в один або кілька видів - наприклад, модель подання 4 + 1 Крухтена [3]. Модель подання - це представлення цілої системи з точки зору відповідного комплексу проблем зацікавлених сторін системи [15]. Нарешті, архітектурний стиль передбачає шаблони та принципи високого рівня, які зазвичай використовуються для додатків, тобто багаторазові рішення загальнопоширених проблем в архітектурі програмного забезпечення в певному контексті. Тому стиль описує типи компонентів разом із набором обмежень щодо того, як їх можна комбінувати. Відповідно до [68], архітектурні стилі можна класифікувати за комунікацією (наприклад, SOA (Service-oriented Architecture), message bus), розгортанням (наприклад, клієнт / сервер, багаторівнева архітектура (N-Tier), програмне забезпечення багатосторонніх програм (multitenant cloud) та структурою (наприклад, component-based, layered architecture).

Service-oriented Architecture (SOA). Сервісно-орієнтована архітектура пропонує переваги вільного з'єднання сервісу та його постачальника, пристосованості за допомогою композиції, організації та інкапсуляції сервісу для інтеграції у високо розподілену систему [48]. SOA прописує, як обчислювальні суб'єкти взаємодіють таким чином, що запитувач знає лише, що таке робота сервісу і як її запитувати; і сервіс єдиний, який знає про його реалізацію.

Багато компаній знаходять, що зробити їх додатки високо доступними, масштабованими, модифікованими та гнучкими все ще залишається складним завданням. Мікросервіси - це новий архітектурний стиль для розвитку розподілених систем, який має намір вирішити ці проблеми. «Додаток для

мікросервісів розкладається на незалежні компоненти, які називаються мікросервісами, які працюють спільно, щоб забезпечити загальну функціональність програми» [48]. Це називається компонентизацією через послуги [42]. Принцип архітектури мікросервісів подібний до принципу Unix: робіть одне і робіть це добре [42]. Кожна мікрослужба має чітко визначені контракти (як правило, RESTful) щодо інших мікросервісів для комунікації та обміну даними. Оскільки мікросервіси можуть бути розгорнуті незалежно один від одного, вони можуть масштабуватися незалежно, їх легко замінити та оновити. Це підтримує швидку / рухливу та надійну еволюцію програми.

Нарешті, REST - це програмний архітектурний стиль Всесвітньої павутини (WWW), який забезпечує узгоджений набір обмежень для проектування веб-сервісів у розподіленій системі гіпермедіа, що може призвести до більш ефективної та ремонтопридатної архітектури [20]. Отже, REST - це архітектура, яка частіше використовується для створення API веб-служб (Прикладний програмний інтерфейс), яка може використовуватися будь-яким пристроєм або клієнтом, що реалізує HTTP (протокол передачі гіпертекстових документів). Це робить REST простішим, ніж попередні альтернативи, такі як SOAP (Простий протокол доступу до об'єкта) та XML-RPC (віддалений виклик процедури).

Шина повідомлень (Message bus). Архітектура, орієнтована на обробку повідомлень, передбачає використання орієнтованого на повідомлення підходу проміжного програмного забезпечення для централізації зв'язку між програмними системами або підсистемами таким чином, що деякі системи генерують повідомлення, а багато інших систем споживають ці повідомлення. Проміжне програмне забезпечення, орієнтоване на повідомлення, може реалізовувати черги повідомлень, публікувати / підписуватись або пересилати повідомлення.

Багаторівнева архітектура

Багаторівнева архітектура розділяє функціональність на окремі сегменти, будучи рівнем, розташованим на фізично окремому комп'ютері. Клієнт-серверні архітектури або 2-рівневі архітектури розділяють систему на дві програми, де клієнт робить запити на сервер. У багатьох випадках сервер є базою даних з логікою додатків, представленою як збережені процедури [68]. Рівні, як правило, є монолітними, оскільки вони реалізують різноманітні функції, які об’єднані в єдиний пакет, розгорнутий на апаратному забезпеченні, попередньо масштабованому для пікових навантажень [48].

Cloud Computing – Multi-tenant SaaS architecture. Хмарні обчислення - це нова модель, яка забезпечує зручний мережевий доступ на вимогу до спільного пулу конфігурованих обчислювальних ресурсів (наприклад, мереж, серверів, сховищ, програм, платформ та послуг), які можна швидко забезпечити та випустити з мінімальними зусиллями управління. або взаємодії постачальника послуг. Постачальники хмарних обчислень, такі як Microsoft, Amazon та Google, пропонують різноманітні обчислювальні послуги, побудовані на основі власної інфраструктури, якими керують спеціальні глобально розподілені центри обробки даних, що забезпечують високу доступність, стійкість та масштабованість. Хмара сприймає всі виконані завдання як «послугу»: Інфраструктура як послуга (IaaS); Платформа як послуга (PaaS); та Програмне забезпечення як послуга (SaaS). Багатокористувацький SaaS - це архітектурний стиль, заснований на модульному дизайні, в якому один екземпляр програми обслуговує декількох користувачів (орендарів). Орендарям може бути дозволено налаштовувати деякі частини програми для реалізації своїх індивідуальних та різноманітних вимог [37].

1.1.2. Існуючі архітектури програмного забезпечення для медичних кіберфізичних систем та їх основні характеристики

Для визначення основних характеристик існуючих архітектур ПЗ для медичних кіберфізичних систем, використано схему характеристик, визначену на рис. 1.2. Тому опис та порівняння існуючих архітектур програмного забезпечення для

медичних кіберфізичних систем базується на таких категоріях: зацікавлені сторони, основні функціональні особливості, архітектурні стилі та моделі, компоненти, реалізація технологій, проблеми сумісності та інших основних нефункціональних особливостях та атрибутах якості.

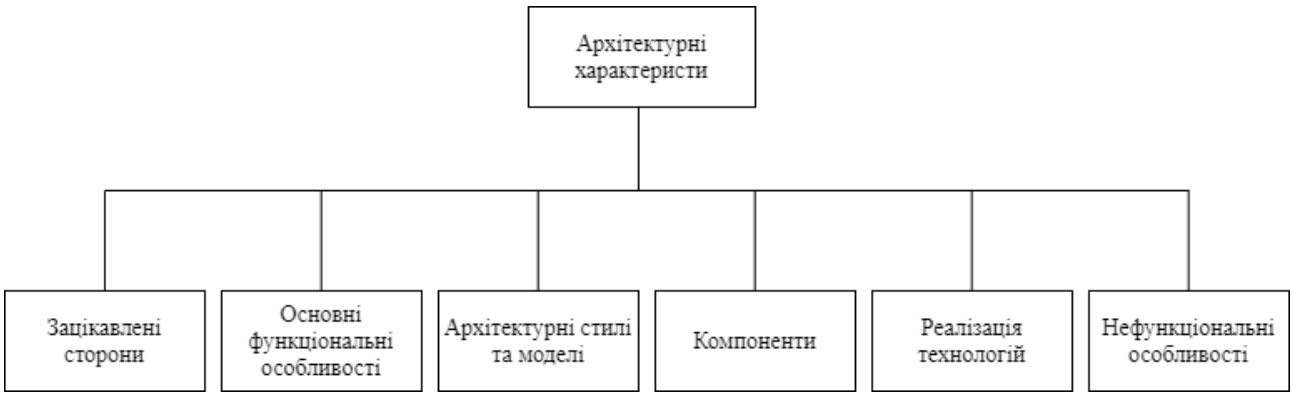

Рисунок 1.2. Схема характеристики архетіктери програмного забезпечення

Зацікавлені сторони

Платформа EcoHealth [40] визначає набір зацікавлених сторін, які беруть участь у системі: виробники пристроїв, які розробляють драйвери пристроїв, сумісні з API EcoHealth, щоб зробити можливим зв'язок між пристроями та платформою; лікарі / клініцисти, які постійно контролюють дані, зібрані від пацієнтів за допомогою платформи EcoHealth, і використовують цю інформацію для покращення діагностики та реагування на надзвичайні ситуації; пацієнти, які надають інформацію платформі через підключені датчики тіла; та системних адміністраторів користувачів та діяльності лікарні. Архітектура, запропонована в [34] [41] [46], спрощує визначення зацікавлених сторін, орієнтуючись лише на пацієнтів та лікарів. У цій роботі виробники пристроїв не розглядаються, оскільки підхід заснований на використанні сенсорів вбудовааних у смартфонах, а отже, драйвери вже вбудовані. Окрім пацієнтів, лікарів та адміністраторів, [38] також визначає соціальних працівників та сімейних опікунів, які мають повний доступ до інформації про пацієнта в будь-який час і в будь-якому місці. Відповідно до [46], виробники пристроїв не розглядаються, оскільки архітектура, описана в [38], підтримує лише обмін даними між

біосенсорами та мобільним пристроєм за допомогою стандарту бездротової технології Bluetooth. Платформа BodyCloud [2] [39], крім пацієнтів та лікарів, включає нових зацікавлених сторін: хмарних постачальників та розробників. Ця платформа надає розробникам веб-абстракції для програмування для швидкого розвитку та розгортання програм BSN (Body sensor network) у широкому діапазоні доменів, включаючи моніторинг охорони здоров'я, управління надзвичайними ситуаціями, моніторинг стану та спостереження за поведінкою.

На відміну від попередніх систем, [23] зосереджується лише на інтерфейсі для автоматизації зчитування життєво важливих показників в електронних медичних записах (EMR); таким чином, єдиним клієнтом є медичний центр, а кінцевим споживачем є медичний персонал. Подібно до [39], [23] включає нову зацікавлену сторону: розробника / супровідника як передбачуваного одержувача читабельності, пристосованості та чітко визначеного інтерфейсу. [26] вважає лікарів, патологоанатомів, фармацевтів, медсестер та родичів зацікавленими сторонами системи. [36] включає медсестер, які надають допомогу в машині швидкої допомоги, та лікарів, які контролюють життєво важливі показники та віддалено аналізують справу. [35], який представляє мобільний додаток для електрокардіограми для спортсменів, включає лікарів, фізіотерапевтів та спортивних тренерів. Система iSenior [31] пропонує рішення для спостереження за людьми похилого віку, всередині будинку або поза ним; таким чином, зацікавленими сторонами, яких описує ця робота, є доглядачі, системні адміністратори та пацієнти. Нарешті, [54] зосереджується на впровадженні гуманоїдних роботів для догляду за діабетиками, пропонуючи наглядачам набір послуг для спостереження за своїми пацієнтами.

Основні функціональні особливості (оцінювання)

Платформа EcoHealth [40] надає лікарям управління всією інформацією про своїх пацієнтів, а також даними, доданими до них датчиками тіла, медичними картами та повідомленнями про проблеми, симптоми чи аномалії життєво важливих показників, виміряних за допомогою датчиків. EcoHealth підтримує

неоднорідність датчиків, а отже, неоднорідність керованих даних, якщо для цих датчиків існують (сумісні з EcoHealth) драйвери.

Система, представлена в [46], підтримує моніторинг хворих на паркінсонізм за допомогою мобільного додатка, здатного автоматично виявляти рухові розлади, що називаються замерзанням ходи (ЗХ), за допомогою набору датчиків на смартфонах та генерувати сигнал тривоги, коли ЗХ виявлено. Ця система управляє дуже певними типами даних, такими як довжина кроку, частота кроків та індекс замерзання, за допомогою певного типу датчиків на борту смартфонів. Система також підтримує клініцистів, надаючи веб-додаток, який дозволяє їм проконсультуватися з усією інформацією про епізоди ЗХ своїх пацієнтів, а також оновити фармакологічну та реабілітаційну терапію пацієнтів. Пацієнти також можуть бачити свої фармакологічні та реабілітаційні методи лікування, і вони отримують сповіщення, як тільки відбулася зміна, яка стосується будь-якої терапії.

Подібно до [46], система моніторингу хронічних хворих, описана в [38], підтримує ранню діагностику за допомогою розумного мобільного пристрою, здатного обробляти дані з датчиків Bluetooth, таких як пульсоксиметри, акселерометри та датчики температури тіла. Пристрій також може активувати сигнали тривоги, які надсилатимуться вихователям та медичному персоналу хронічного пацієнта. Тому і [46], і [38], на відміну від [40], підтримують внутрішній і зовнішній моніторинг. Система [38] також підтримує соціальних працівників та сімейних опікунів, надаючи веб-програму для отримання інформації про відповідного пацієнта. Нарешті, система надає утиліту для пошуку (на основі природної мови) медичної інформації в медичних джерелах.

Платформа BodyCloud [39] підтримує управління та зберігання великих обсягів потоків даних, що надходять із сенсорних мереж тіла, які відстежують стан здоров'я пацієнтів (наприклад, серцеві напади, діабет та астма), а також обробку та онлайн/офлайн аналіз даних для підтримки лікарів у прийнятті рішень. Дані передаються в режимі реального часу лікарям, які відповідають за

конкретного пацієнта, і вони можуть надавати інструкції пацієнтам через сигнали тривоги. Ця платформа пропонує розробникам абстрагування аналізу даних на основі робочого процесу, тобто розробники можуть налаштувати та модифіковувати спосіб обробки та аналізу даних. Подібно до [39], медична КФС, запропонована в [55], управляє великими обсягами даних. Система підтримує збір даних через набір вузлів даних, і ці дані надсилаються в хмару через адаптери, аналіз у режимі реального часу та офлайн, обробка алгоритмів видобутку даних та машинне навчання.

Knowledge-Aware Service-Oriented (KASO) проміжне програмне забезпечення [22] було розгорнуто з двома основними функціями в сценарії охорони здоров'я: відстеження пацієнтів та медичного персоналу через віртуальний периметр навколо санаторіїв за допомогою браслетного вузла та моніторинг біосигналів через два біомедичні датчики , датчик монітора серця та датчик температури тіла. Крім того, по всій території санаторію було розгорнуто набір проміжних вузлів для моніторингу температури, вологості та освітлення навколишнього середовища, щоб зробити висновки як з даних біомедичних датчиків, так і з датчиків навколишнього середовища. Проміжне програмне забезпечення пропонує механізми виявлення для динамічної інтеграції нових пристроїв у сценарій охорони здоров'я, розуміння контексту для динамічного та ефективного розподілу ресурсів та програмування абстракцій для швидкого прототипування сценаріїв на основі датчиків та виконавчих механізмів.

[23] має на меті автоматизувати зчитування життєво важливих показників за допомогою біосенсорів для існуючих систем електронних медичних записів (EMR). З цією метою [23] розробляє та впроваджує кіберфізичний інтерфейс, який фіксує життєво важливі ознаки для типу зустрічі за допомогою набору спеціальних форм життєвих ознак, які збирають конкретні медичні дані.

Система e-SURAKSHAK [26] здатна здійснювати в режимі реального часу, безперервний та автономний моніторинг життєвих показників пацієнтів. Ці життєво важливі ознаки походять від «бездротових вбудованих Інтернет-

пристроїв» (WEID) і періодично зберігаються в електронній базі даних та флеш-пам'яті. Крім того, ця система реалізує веб-графічний графічний інтерфейс для візуалізації вимірювань у реальному часі, можна налаштувати WEID для періодичних вимірювань. Він також включає сповіщення для повідомлення медичного персоналу про проблеми з підключенням до мережі.

Система моніторингу пацієнтів [36] підтримує моніторинг стану пацієнта, потокове передавання відео та аудіозв'язок через VoIP від швидкої допомоги при екстреному виклику. Автори пропонують програму для моніторингу життєво важливих показників пацієнта без затримки передачі даних, тим самим дозволяючи лікарю розпочати аналіз та лікування, поки пацієнт все ще перебуває у машині швидкої допомоги. Ці послуги доступні через два графічні інтерфейси, один для лікаря в лікарні, який показує отримання даних у реальному часі з медичних сенсорів, а також забезпечує голосовий та відео зв’язок, а другий для персоналу швидкої допомоги, який відображає інформацію про місцезнаходження пацієнта.

Додаток ЕКГ для Android [35] надає пацієнтам візуалізацію хвиль на електрокардіограмі. Пацієнт використовує електроди ЕКГ, а виміряні дані надсилаються на мобільні пристрої та зберігаються у приватній хмарі пацієнта або у певній медичній хмарі.

Система iSenior [31] підтримує моніторинг, оповіщення та запит про допомогу. Пацієнт похилого віку має пристрій персонального моніторингу (пристрій для моніторингу похилого віку), який дозволяє вихователям контролювати біосигнали пацієнта в режимі реального часу, аналізувати історичні дані, знаходити людей на випадок, якщо їм потрібна допомога, і отримувати сповіщення. Система має такі дві функції: збір та зберігання даних: сенсор генерує інформацію, яка зберігається в базі даних і може бути переглянута через веб-інтерфейс користувача; та управління сигналізацією: система дозволяє лікарям встановлювати сигнали тривоги над сенсорами за допомогою правил щодо параметрів, що підлягають спостереженню. Коли

досягнуте порогове значення, визначене лікарем, будильник надсилає SMS-повідомлення або електронне повідомлення на заздалегідь визначену групу стільникових телефонів та адресу електронної пошти та відображає сигнал тривоги у веб-інтерфейсі. Система налаштована на надсилання попередження у разі розряду батареї; локалізація: система дозволяє знаходити користувачів та відстежувати їхні траєкторії на Картах Google; віддалений доступ: система реалізує автентифікацію та доступ до функціональних можливостей системи; профілі користувачів: система реалізує три профілі: доглядачі, які мають дозвіл на перевірку та аналіз даних, системний адміністратор з доступом до всієї інформації та кінцеві користувачі (пацієнти); автоконфігурація: система дозволяє реєструвати пристрої, додаючи їх до бази даних, коли вони ввімкнені.

Структура Activity as a Service [53] побудована на платформі BodyCloud [39], що дозволяє збирати, обчислювати та зберігати дані датчиків, програмування аналізу даних на основі робочого процесу та візуалізацію результатів. Автори цієї системи визначають «Діяльність як послугу» як «повноцінну кіберфізичну систему для підтримки розпізнавання та моніторингу людської діяльності в громаді, в Інтернеті та поза мережею» [53]. Абстракції високого рівня BodyCloud спеціалізуються на визначенні трьох моделей залежно від (попередньої) обробки, що виконується локально або в хмарі. Таким чином, три моделі - це повна хмара (необроблені дані надсилаються в хмару, яка виконує всю необхідну обробку), змішана хмара (є деяка попередня обробка на стороні тіла перед тим, як дані надсилаються в хмару), і повністю локальний (дані збираються та обробляються повністю з боку тіла, а потім надсилаються в хмару лише для тривалого зберігання та історичної візуалізації). У [53] автори застосовують абстракції програмування до програм моніторингу людини для моніторингу постави та активності та підрахунку кроків (включаючи автентифікацію, геолокацію та обчислення відсоткового часу, витраченого на кожну виконану діяльність), оцінку фізичної енергії (оцінювач ккал) , автоматичне виявлення падіння (включаючи систему сигналізації, яка підключається до Facebook і

надсилає сповіщення заздалегідь визначеним друзям, як родичам чи доглядачам), та розумну підтримку інвалідного візка (розпізнавання постави тіла в інвалідному кріслі).

[54] включає гуманоїдних роботів для підтримки надзвичайних ситуацій при лікуванні діабету. Платформа надає такі функції: інтерфейс доступу до програми, який забезпечує уніфікований доступ до програм-центрів управління захворюваннями, коли він отримує запит; перевірка вхідних запитів; автентифікація за допомогою сеансу HTTPS, файлів cookie або ключів доступу, які - якщо це дозволено - перенаправляють запит до сервісу; рішення сервісу, яке генерує екземпляр сервісу; авторизація, при якій блок авторизації перевіряє, чи має об’єкт дозвіл на доступ до сервісу, та використання функцій сервісу.

[41] представляє систему під назвою CardioNet, яка включає систему моніторингу, що полегшує дистанційний нагляд за пацієнтами із серцево-судинними захворюваннями за допомогою певних портативних пристроїв. Ця система визначає онтологію для семантичної інтерпретації отриманих медичних даних, а також для безперебійного обміну даними між автономними структурами (кабінети загальної практики, лікарні, лабораторії, відділення невідкладної допомоги, медичний персонал та пацієнти). Онтологію можна пристосувати для лікування інших захворювань. CardioNet дозволяє пацієнтам і лікарям взаємодіяти трьома способами: в Інтернеті (пацієнти та лікарі взаємодіють через веб-інтерфейс), поза мережею (система забезпечує асинхронний зв'язок між цими учасниками) та безпосередньо обличчям до обличчя. Крім того, система дозволяє дистанційну консультацію та ведення записів пацієнтів.

На відміну від інших пропозицій, в яких моніторинг більше зосереджений на життєво важливих показниках, VIRTUS [28] пропонує контролювати рухи пацієнта, тобто підраховувати кількість кроків та визначати пози протягом дня. За допомогою цієї інформації можна проаналізувати проміжок часу між діями та реакцію пацієнта після введення препарату.

[47] пропонує телемоніторинг параметрів, пов’язаних зі здоров’ям - швидкості та напрямку руху, пульсу та насичення крові киснем - що дозволяє користувачам отримувати доступ до оброблених даних із будь-якого пристрою, що має доступ до Інтернету (включаючи ПК та планшети). Нарешті, проект [51], на відміну від інших, реалізує «Інтелектуальне будівництво» для обробки великих обсягів даних, отриманих від датчиків, підключених до тіла (наприклад, зап’ястя, щиколотки, серця та грудної клітки).

Архітектурні стилі та моделі

Більшість досліджуваних систем описують архітектури програм для медичних кіберфізичних систем з використанням моделі розгортання (відома як фізична модель) за допомогою архітектурних стилів багаторівневої архітектури (N-Tier), клієнт-серверної архітектури (Client-server) та хмарних обчислень (cloud computing); модель структури (модель реалізації) через компонентно-орієнтовану архітектуру (component-based) та багатошарову архітектуру (layered architecture); та комунікаційна модель (модель обробки) через сервісно-орієнтовану архітектуру (SOA), шину повідомлень, подійно-орієнтовану архітектуру та архітектуру REST.

Архітектура, описана в [40], є трирівневою архітектурою: мережа бездротових датчиків (рівень сприйняття) взаємодіє з набором компонентів, які називаються драйверами (зазвичай називаються рівнями шлюзу), які згодом обмінюються даними через Інтернет з платформою EcoHealth (прикладний рівень). Модель реалізації архітектури платформи EcoHealth [40] складається з набору модулів (сервіси проміжного програмного забезпечення). Модуль підключення пристроїв інтегрує фізичні пристрої (бездротові датчики тіла) до платформи за допомогою драйверів, які різні виробники розробляють відповідно до визначеного API (API EcoHealth). Ці драйвери можуть отримати доступ до даних шляхом постійного опитування або за запитом після повідомлення. Вони побудовані на принципах REST і засновані на протоколі HTTP (сумісний з парадигмою мережі речей). Таким чином, датчики тіла являють собою веб-

ресурси, які надають доступ, контроль і моніторинг через Інтернет. Модуль обробки даних реєструє дані, що надсилаються до EcoHealth через запити HTTP PUT від драйверів. Модуль зберігання включає реляційну базу даних та файлову систему для зберігання всіх даних, що використовуються в EcoHealth, які можуть скористатися перевагами хмарних інфраструктур зберігання. Модуль загальних служб складається з набору загальних послуг, таких як безпека та сповіщення (наприклад, механізми сповіщення на основі подій, які повідомляють про проблеми, симптоми або аномалії вимірюваних життєвих показників). Нарешті, модуль візуалізації та управління забезпечує веб-інтерфейс для надання зацікавленим сторонам (лікарям, пацієнтам та системним адміністраторам) доступу до основних функціональних можливостей.

Архітектура, описана в [46], також є трирівневою: на борту смартфонів є датчики, а смартфони виступають як шлюзи, які надсилають дані до основного компонента, тобто до клінічного віддаленого сервера. Вид реалізації архітектури складається з наступних компонентів: додаток для смартфона, база даних та веб-програма. Додаток для смартфона відстежує параметри ходи, виявляє епізоди замерзлої ходи, генерує сигнали тривоги у відповідь на них, зберігає ці дані в локальній базі даних і періодично завантажує ці дані в центральну базу даних клінічного віддаленого сервера.

Архітектура, описана в [38], як [40] та [46], представляє три рівні: датчики життєво важливих знаків та їх зв'язок складають рівень сприйняття; розумний мобільний пристрій складає рівень шлюзу, який фіксує життєві показники датчиків і передає дані на медичну платформу; і нарешті, ця медична платформа становить прикладний рівень. Модель реалізації прикладного рівня має такі модулі: розумний мобільний пристрій, який, крім фіксації життєво важливих показників від датчиків і надсилання цієї інформації на медичну платформу, реалізує систему генерації правил, здатну обробляти проміжні дані та генерувати тривожні повідомлення направляти до опікунів пацієнта або медичного персоналу; модулі зберігання та обробки даних, які здатні управляти великими

обсягами даних; платформа взаємодії та обміну повідомленнями для доставки інформації до всіх зацікавлених сторін за допомогою SMS, голосу та технології PUSH; платформа веб-сайту, що дозволяє отримати інформацію про пацієнта; і шукач інформації, пов’язаної зі здоров’ям, у різних медичних джерелах.

Платформа BodyCloud [39] - це чотирирівнева архітектура. Найнижчий рівень - це рівень сприйняття, який складається з мережі сенсорів тіла, який, у свою чергу, складається з набору сенсорних вузлів, які вимірюють біосигнали та надсилають необроблені або попередньо оброблені дані координатору. Вузли датчиків надають функції, які є більш досконалими, ніж попередні роботи, такі як стандартний інтерфейс до різноманітних драйверів датчиків, а також налаштування та розширення платформи для підтримки нових фізичних датчиків та сервісів обробки сигналів. Рівень шлюзу призначений для координатора; він управляє вузлами датчиків (наприклад, координатор може повідомити про відкриття нових датчиків), збирає, зберігає та аналізує дані, отримані від датчиків, і діє як шлюз для підключення BSN до віддаленого клінічного сервера. Ці два рівні називаються стороною тіла. Прикладний рівень підтримує розподілену систему реального часу для моніторингу та аналізу потоків даних BSN, а також систему візуалізації результатів аналізу даних за допомогою вдосконаленої графічної звітності. Перший розгортається у хмарі через модель SaaS і називається стороною хмари, тоді як другий розгортається як клієнтська програма HTML / Android і називається стороною перегляду. Нарешті, можна розглянути новий рівень - рівень абстракції програмування, який надає розробникам абстракції програмування для швидкого прототипування додатків BSN і називається стороною аналітика. З цією метою автори визначили набір програмних абстракцій таким чином: група для формалізації джерела даних BSN; робочі процеси для визначення дій, які будуть реалізовані над вхідними даними - наприклад, зберігання, обробка, аналіз; перегляди для формалізації результатів для глядачів; і спосіб формалізації взаємодії між вхідними даними, діями та вихідними даними.

KASO Проміжне програмне забезпечення [22] - це також трирівнева архітектура, хоча рівні описуються як шари. Проміжне програмне забезпечення KASO розроблено на основі парадигми агента, яка називається Perceptual Reasoning Agents (PRA). Найнижчий рівень відповідає багатофункціональному вбудованому шару (де розташована апаратна платформа), операційній системі в режимі реального часу, яка приховує неоднорідність апаратних засобів від вищих рівнів, і рівню мережевого протоколу, який передає пакети в мульти-хоп-маршрутизації. Проміжний рівень - це рівень, який складається з трьох підсистем: фреймворк-служби, що складаються із сервісів запитів, за допомогою яких можна отримати інформацію про конкретні параметри, сервіс диспетчера виконання, що відповідає за завантаження / розвантаження та запуск / зупинку компоненти та PRA, служба безпеки для управління дозволами на доступ та шифрування інформації та служба конфігурації для встановлення параметрів на низькому рівні; послуги зв'язку, сформовані з трьох модулів, що дозволяють виявляти ресурси низького рівня, виявляти та впливати на сервіс, а також Інтернет зв'язок; та сервіси управління знаннями, реалізовані брокером та організатором сервісів, що реалізують API, запропонований розробникам PRA. Верхній шар - шар агента, де визначені PRA.

У роботі [23] пропонується розробка кіберфізичного інтерфейсу для автоматизації зчитування біосигналів в системах EMR, описуючи трирівневу архітектуру. Необхідно підкреслити, що автори відносять рівні до шарів. Отже, вони описують наступну декомпозицію архітектури. Інтерфейсний рівень включає станцію зчитування життєвих ознак (додаток, що зчитує біосенсори та зберігає інформацію у внутрішній пам’яті пристрою, де вона розгорнута), та спеціальну форму життєвих ознак (яка підтримує автоматичне читання біосигналів із читальної станції). Логічний рівень програми відповідає за збір даних із станції зчитування, їх відображення у формі та надсилання їх на рівень даних. Рівень даних позначає існуючу систему EMR.

PRIME [44] пропонує дворівневу архітектуру, що складається з комунікаційного рівня, який використовує накладення, використовуючи пристрої в накладеній мережі, побудованій поверх низькорівневих технологій бездротового зв'язку, та рівень програмування API, який забезпечує абстракції та операції реалізувати програми P-RESTful (Pervasive REST).

E-SURAKSHAK [26] складається із сервісно-орієнтованої архітектури (SOA) з трьома рівнями. Екологічний рівень (еквівалентний рівню сприйняття попередньої роботи) складається з WEID і маршрутизаторів і визначає життєво важливі параметри здоров'я пацієнтів. Рівень обслуговування складається з основних та складних функцій аналізу. Нарешті, рівень управління зберігає та обробляє зібрану інформацію та забезпечує веб-інтерфейс.

Система моніторингу пацієнтів (PMS) [36] також представляє SOA, яка реалізована за допомогою середовища OSGi. OSGi - це ініціатива, орієнтована на взаємодію програм та служб на основі платформи інтеграції компонентів (платформи обслуговування), що забезпечує сервісно-орієнтоване середовище на основі компонентів. Рівень послуг динамічно пов'язує набори - компоненти OSGi - за допомогою моделі публікації-пошуку-прив'язки для простих старих об'єктів Java. Рівень життєвого циклу реалізує API для встановлення, запуску, зупинки, оновлення та видалення пакетів. Рівень модулів визначає, як пакет імпортує та експортує код. Рівень безпеки гарантує безпечний доступ. Нарешті, середовище виконання визначає, які методи та класи доступні в конкретній структурі. PMS визначає набір пакетів для безпечного входу в систему, потокового передавання відео, медичного спостереження за пацієнтами, пристроїв для пацієнтів, аудіозв'язку та GPS-навігації.

Додаток ЕКГ для Android [35] реалізує наскрізну архітектуру системи. Необхідно підкреслити, що автори описують трирівневу архітектуру, але посилаються на рівні як шари. Отже, вони описують таку декомпозицію архітектури: апаратний рівень включає мікроконтролери та датчики; прикладний рівень реалізує модель / вигляд / контролер програми Android для взаємодії з

користувачами, отримує дані біосигналу з апаратного рівня, зберігає дані в буфері та двійковому файлі та передає файл із пристрою Android на FTP-сервер на хмарному рівні.

Система iSenior [31] реалізує такі модулі: портал: цей модуль використовує інтерфейс, що дозволяє автентифікацію та доступ до iSenior. Портал відображає інформацію про життєво важливі ознаки від приладу для спостереження (Elderly Monitoring Device) за літніми людьми через записи та попередження, забезпечує аналіз історії та відстежує пацієнтів за допомогою Карт Google, конфігурації тривоги та функцій моніторингу; ядро: цей модуль керує зв'язком між віддаленим медичним сервером та EMD; обробка сигналів: цей модуль відповідає за активацію попереджень шляхом виявлення ненормальних умов, надсилання SMS-сповіщень та реалізації алгоритмів обробки необроблених даних

[41] також пропонує трирівневу архітектуру, яка складається з модульного прототипу портативного пристрою (рівень сприйняття), який включає модуль обробки (відповідальний за підготовку та обробку сигналу), модуль збору даних, який вимірює параметри , і модуль передачі даних, який відповідає за передачу отриманих даних на шлюз або точку доступу. Шлюз управляє зв'язком; ця пропозиція розробила два протоколи зв'язку - пристрій до шлюзу та шлюз до сервера. Модулі для розпізнавання активності та складного моніторингу пацієнтів впроваджують послуги охорони здоров’я (прикладний рівень). Перший визначає діяльність, яку реалізують пацієнти (наприклад, ходьба або біг), а другий контролює набір з восьми параметрів (наприклад, ЕКГ, артеріальний тиск, потік повітря та рівень кисню).

VIRTUS [28] пропонує багатошарову архітектуру (layered architecture), на базі SOA, яка складається з: носимих пристроїв, які збирають дані, пов’язані з діяльністю пацієнта; проміжне програмне забезпечення, що дозволяє здійснювати зв’язок між модулями системи; інтерфейс між користувачем, системою та віддаленим центром; а також програмне забезпечення центрального

підрозділу або менеджера проектів, яке відповідає за класифікацію даних та формування щоденних звітів та відгуків для пацієнта.

Медична КФС [55] пропонує подібну архітектуру до VIRTUS [28] з наступними рівнями: рівень збору даних, що складається з вузлів даних, адаптерів та інтерфейсу для різнорідних даних з різних джерел; рівень управління даними, який містить модуль розподіленого зберігання файлів (DFS) та модуль розподілених паралельних обчислень (DPC); і сервісний прикладний рівень, який забезпечує візуальну презентацію результатів та відкритий уніфікований API для розробників.

μWoTOP [34] реалізує багатошарову архітектуру (layered architecture): шар екосистеми IoT, який відповідає за адаптери, що визначають потенційні ресурси (датчики та виконавчі механізми), класифікують їх на основі їх топології та повідомляють про них верхнім шарам; рівень проміжного програмного забезпечення Web of Things, метою якого є надання функціональних можливостей, щоб дизайнери та розробники сервісів могли проектувати, розробляти та розгортати розумні простори; і склад ресурсів та рівень організації, який відповідає за розгортання необхідних компонентів, щоб можна було пропонувати сервіси та функціональні можливості.

[47] пропонує децентралізовану хмарну архітектуру з трьома рівнями: датчики та виконавчі механізми (рівень сприйняття), рівень шлюзу, що відповідає за зв'язок між управлінням ресурсами та віддаленими одиницями телеметрії (RTU), а також послуги презентації та застосування (рівень додатків).

[51] також пропонує багатошарову архітектуру: рівень збору даних відповідає за отримання та обробку даних; рівень зв'язку забезпечує наскрізне підключення, агрегування виміряних даних з інших пристроїв та перетворення у формат; обробний шар виконує статистичні розрахунки; рівень управління складається з медичної експертної системи, яка пропонує дії на основі даних, що генеруються процесом обробки; а рівень обслуговування забезпечує доступ до кінцевого

споживача (лікарні, служби екстреної допомоги, швидкої допомоги та відділення міліції), а також забезпечує аналіз на основі історії хвороби пацієнта.

Деталі реалізації

Платформа EcoHealth [40] реалізована на мові програмування Java і розгортається на прикладному сервері JBoss. Дані зберігаються в реляційній базі даних MySQL за допомогою специфікацій Java Persistence API (JPA), реалізованих в рамках Hibernate. Дані, якими слід керувати, описуються наступним чином: медичні записи, історичні дані від датчиків тіла, повідомлення, інформація, пов'язана з пацієнтами, та інформація про екстрені служби. Наданий веб-інтерфейс реалізований за допомогою технології JavaServer Faces (JSF). Відповідно до специфікацій авторів, драйвери повинні реалізовувати інтерфейс RESTful для клієнтів (людей чи додатків), зокрема реалізацію RESTEasy для архітектурного стилю REST та API Java для служб RESTfulWeb (JAX-RS). Нарешті, автори використовують специфікації служби автентифікації та авторизації Java (JAAS), реалізовані на сервері JBoss, для забезпечення автентичності, конфіденційності та цілісності користувачів.

В архітектурі, описаній у [46], є нечіткий логічний алгоритм та система нечітких висновків - реалізована за допомогою програми для смартфона - для визначення епізоду замерзлої ходи (FoG) на основі набору параметрів ходи: довжина кроку, крок частота, індекс замерзання та енергія. Додаток для смартфонів був розроблений для операційних систем iOS та Android. Дані зберігаються в реляційній базі даних MySQL на віддаленому клінічному сервері. Інформація, яку слід зберігати, така: особисті дані користувачів веб-додатків, дані моніторингу пацієнтів із програми для смартфона та фармакологічні та реабілітаційні методи лікування, призначені клініцистами пацієнтам.

Особливість [38] полягає у використанні розподіленої нереляційної бази даних для зберігання дуже великих обсягів даних наступним чином. Розумний мобільний пристрій реалізує безперервне опитування датчиків. Дані

зберігаються з використанням файлів, записи яких мають формат кількох елементів (ідентифікатор_сенсора, часовий_знак, значення), і кожен запис займає 10 байт пам'яті. Коли розмір файлу досягає 500 Кб, його закривають та створюють інший файл для збереження найсвіжіших даних. Файли даних, збережені на інтелектуальному пристрої, синхронізуються на сервері в Інтернеті, тобто дані передаються на сервер через захищений розподілений механізм синхронізації файлів P2P (протокол Bit-Torrent Sync). Через велику кількість даних, що підлягають обробці на стороні сервера, [38] використовує базу даних RIAK, яка є розподіленою, масштабованою нереляційною базою даних ключ / значення з відкритим кодом. У RIAK єдиною одиницею зберігання даних є об’єкти, які можна ізолювати за допомогою віртуальних просторів ключів (сегментів). Отже, дані організовані таким чином: сегмент для користувачів, де ключовим значенням є ідентифікатор, а значенням є об'єкт, що містить дані користувача та список ідентифікаторів датчиків, пов'язаних з цим користувачем, і (II) сегмент для датчиків, де ключове значення формується ідентифікаторами користувача та пристрою, а значення є об’єктом, що містить необхідні дані для ідентифікації останніх даних, переданих одним датчиком.

[39] інтегрує хмарні обчислення та BSN. Хмарна сторона розроблена за допомогою Google App Engine (GAE). Зокрема, BodyCloud використовує API схожих даних GAE для збереження даних потоків, API чергових даних GAE для складеного поділу, що перетворює реалізацію додаткових програм для аналізу даних та інтеграції робочих процесів видобутки сторонніх даних (наприклад, WEKA [51]) та (III) OAuth 2.0 [27] для систем авторизації. Взаємодія між компонентами цього рішення - тілом та хмарою (збір даних), хмарою та засобом перегляду (візуалізація даних) та хмарою та аналітиком (визначення нових служб аналізу даних) - підтримується через веб-службу RESTful, реалізовану за допомогою Restlet Framework.

KASO Middleware [22] полягає у можливості динамічної реконфігурації ресурсів (наприклад, датчиків) залежно від умов навколишнього середовища. З

цією метою цей підхід визначає набір послуг управління знаннями (Knowledge Management), які залежать від інформаційної моделі системи, що описує ресурси низького та високого рівня. Дані зберігаються в базах знань (Knowledge Bases), контекстній базі знань (Contextual Knowledge Base) та системній базі знань (System Knowledge Base), які розміщуються на вузлах датчиків та виконавчих механізмів, контекстних вузлах та шлюзах відповідно. Бази знань структуровані через онтологію над RDF / OWL 2, а його послуги наносяться на стандартизовані веб-служби REST.

[23] реалізував кіберфізичний інтерфейс для автоматизації зчитування біосигналів у системах EMR. Реалізація цього інтерфейсу була зосереджена на власній та ліцензованій ERM: Centricity Practice Solution®, яка розгортається на сервері Citrix. Ця робота реалізувала обгортку (або адаптер) для розширення функціональності ERM, тобто для автоматизації генерування EMR від датчиків та зберігання даних у базах даних.

Основними особливостями Системи моніторингу пацієнтів (Patient Monitoring System) [36] є орієнтація на сервіси та агрегація сервісів із використанням платформи Eclipse OSGi та архітектури завантажувача класу Java для розгортання служб під час виконання. PMS включає GPS-навігаційну систему, доступ до якої здійснюється через Інтернет через TCP/IP за допомогою спеціальної мережі. Він підтримує потокове передавання відео через IP-камери з фільтрацією та співвідношенням передачі 1:1 (автори не використовують будь-яке власне програмне забезпечення), моніторинг стану здоров'я пацієнта шляхом попередньої конфігурації OSGi, щоб будь-який пристрій медичного моніторингу надсилало дані через спеціальну мережу та відображає результати у вигляді графіку та аудіо зв’язку через VoIP (voice over IP).

Додаток ECG для Android [35] використовує технології Filestream та Filetable для зберігання неструктурованих даних. Він реалізує мікроконтролер, який отримує біосигнали пацієнта і передає їх на мобільний пристрій по бездротовій мережі за допомогою технології Bluetooth. Дані зберігаються у двійковому

файлі, зашифрованому на захищеній цифровій (SD) карті пристрою, після чого пацієнт може завантажувати двійкові файли до централізованої приватної хмари за допомогою захищеного сервера FTPES (через FTP).

[41] представляє нейронну мережу для визначення виду діяльності, яку слід реалізувати пацієнтам, наприклад, сидячи, стоячи, ходьба, біг або невизначена діяльність.

Відмінними рисами проміжного програмного забезпечення VIRTUS [28] є використання мови програмування Java, використання миттєвих повідомлень XMPP (eXtensible Messaging and Presence Protocol) та використання фреймворку OSGi.

Нарешті, [51] використовує обробку в режимі реального часу та технології MapReduce (Apache Spark та Hadoop) для обробки даних та аналізу та обчислення статистичних параметрів.

Сумісність

У [40] автори визначають три рівні взаємодії: "на нижньому рівні необхідно безперешкодно інтегрувати безліч різнорідних фізичних пристроїв між собою", "на проміжному рівні, дані, надані пристроями повинні бути легко доступними в Інтернеті», «на більш високому рівні стандартизована модель програмування може забезпечити прозорість збору та перетворення інформації з пристроїв вимірювання, не вимагаючи якихось конкретних знань щодо особливостей цих фізичних пристроїв та мережного середовища."

Платформи EcoHealth [40], BodyCloud [39], KASO [22] та Activity-as-aService [53] мають справу з трьома рівнями, інкапсулюючи неоднорідність пристроїв у драйвери для зв'язку з API EcoHealth , зробити дані (канали), що надаються пристроями, доступними для Інтернету через протокол HTTP та RESTful API, та структурувати канали за допомогою XML або JSON. Платформа EcoHealth використовує розширену мову розмітки середовищ (EEML) для вирішення питань взаємодії даних між датчиками тіла від різних виробників. EEML - це

мова, заснована на XML, для опису даних датчиків, зібраних з пристрою, побудови, системи або простору, у структурованому вигляді (у цьому випадку життєві показники людського тіла, зібрані з датчиків тіла). Однак жодна з цих платформ не стосується стандартизації медичних записів, а також реляційні моделі історичних даних детально не описані.

[46] та [26] передбачають лише взаємодію на нижчому та середньому рівнях. У [46] інтеграція датчиків інкапсульована смартфонами - оскільки цей підхід заснований на датчиках вбудованих в смартфи, - і дані датчиків можуть бути доступні через HTTP. У [26] автори визначають взаємодію на фізичному та сервісному рівнях. У першому вони визначають Edge Routers та WEID, пов’язані з пацієнтами, тоді як у другому вони встановлюють послуги, незалежні від контексту або стану інших сервісів.

[38] та [35], навпаки, реалізують смарт-мобільний пристрій на основі мікроконтролера з модулем Bluetooth, і, отже, не зосереджуються на підтримці неоднорідності датчиків.

Запропоноване рішення в [23] базується на станції зчитування Spot Vital Signs LXi®, яка дозволяє зчитувати життєві показники із обмеженого набору сумісних датчиків.

Для збереження взаємодії з іншими рішеннями розроблено механізми, що дозволяють VIRTUS обмінюватися даними з модулями платформ, що використовують інші протоколи зв'язку, ніж XMPP (eXtensible Messaging Presence Protocol), такі як веб-сервіси для SOA. Система [47] інтегрує домашній шлюз у хмарне проміжне програмне забезпечення, дозволяючи тим самим різнорідним пристроям інтегрувати дані з інших платформ охорони здоров’я.

Система [16] пропонує інтеграцію із застосуванням стандарту CDA (Архітектура клінічних документів) HL7, оскільки це провідний світовий медичний стандарт ІКТ, який передбачається для семантичної сумісності медичних даних.

Основні нефункціональні особливості та атрибути якості

Платформа EcoHealth, описана в [40], об’єднує слабо пов'язані модулі, надійність, доступність і масштабованість в залежності від інфраструктур, що базуються на хмарі, та безпеку з точки зору автентичності користувачів, а також цілісності та конфіденційності шляхом шифрування даних, переданих датчиками через Інтернет (наприклад, лише призначений лікар може отримати доступ до запису пацієнта та даних, наданих його датчиком тіла). [46] головним чином зосереджується на забезпеченні набору нефункціональних особливостей для моніторингу хворих на паркінсонізм, до яких раніше не звертались. Автори розглядали такі фактори, як зниження вартості (за допомогою датчиків на борту смартфона) та забезпечення гнучкості (за рахунок того, що це робить пацієнта ненав’язливим). Нарешті, це рішення також стосується питання автентифікації.

Система [38] пропонує обробляти великі обсяги інформації за допомогою розподіленої масштабованої нереляційної бази даних з відкритим кодом, що називається RIAK. Система, яку описують автори, може збирати близько 100 зразків в секунду з датчиків. Підхід BodyCloud [39] включає швидке прототипування через набір веб-абстракцій програмування, які дозволяють формалізувати програми BSN, розширюваність та настроюваність, вносячи зміни з мінімальними перешкодами для користувачів, які вже використовують систему, гнучкість, що дозволяє відкрити нові датчики, та масштабованість, яка покладається на платформи як сервіси.

Однією з нефункціональних особливостей, яку пропонує e-SURAKSHAK [26], є підвищення надійності системи за допомогою надлишкових фронтальних маршрутизаторів. Система моніторингу пацієнтів (Patient Monitoring System PMS) [36] захищає передачу конфіденційної інформації та перешкоджає доступу неавторизованого персоналу через функції безпеки в рамках OSGi. Структура OSGi також забезпечує доступ та повторне використання для незалежності між компонентами та підтримує динамізм та взаємодію для сприяння масштабованості системи.

Пропонована програма ЕКГ для Android [35] стосується таких нефункціональних особливостей, як продуктивність, конфіденційність/безпека, портативність, масштабованість, гнучкість та вартість. Запропоноване рішення здатне контролювати ЕКГ за низькою вартістю. Зберігання даних на SD-карті мобільного телефону дозволяє покращити продуктивність та масштабованість. Крім того, формат двійкового файлу оптимізований для швидкого та компактного аналізу. Забір ЕКГ проводиться через рівні проміжки часу, що забезпечує енергоефективність та продовжує термін служби датчиків та мобільного пристрою.

У роботі [53] висвітлюється надійність розробленого алгоритму моніторингу фізичної активності. Автори виділяють мінімальне обслуговування акумулятора та швидкість роботи смартфона Android та вузол датчика акселерометра. Модуль обробки для розумних візків заснований на Arduino; це забезпечує низьке споживання енергії, високу частоту дискретизації та високі можливості обробки.

Платформа eHealth [54] забезпечує безпечний, точний та безперебійний обмін даними. Крім того, автори згадують, що вони продемонстрували здатність центру боротьби з хворобами годувати пацієнтів, коли цілі в галузі охорони здоров’я не виконуються. Іншим фактором, який слід згадати, є емоційна підтримка, що надається пацієнтам через діалоги між пацієнтом і роботом. Крім того, платформа включає оцінку якості даних за наступними вимірами: точність, повнота, послідовність, часові рамки та зручність використання.

Відмінною особливістю VIRTUS [28] є надійний, масштабований та безпечний канал зв'язку, який дозволяє уникнути втрати даних та скористатися характеристиками парадигми публікації / підписки, коли абонент знаходиться поза мережею.

МФКС [55] включає перетворення та шифрування формату даних. Попередньо оброблені дані шифруються, тобто розшифрувати дані можуть лише авторизовані пристрої.

1.2. Архітектурні та функціональні вимоги до медичних кіберфізичних систем

Програмна платформа та проміжне програмне забезпечення медичної кіберфізичної системи повинні відповідати архітектурним вимогам:

- Неоднорідність та динамічне підключення датчика. Медична кіберфізична система вимагає можливості використовувати велику кількість датчиків тіла за допомогою платформ, що підтримують високорівневі інтерфейси для прозорого доступу до датчиків, приховуючи специфікації та протоколи для користувачів та програм, які їх використовують [22] [39] [40] [41] [44] [53]. На додаток до підтримки неоднорідності, виявлення датчиків також важливо для динамічної інтеграції нових пристроїв у медичних КФС [22] [31] [39] [41].
- Масштабованість. Величезний обсяг даних, який, як очікується, буде сформовано датчиками медичної КФС, вимагає масштабованих можливостей зберігання та обробки [24] [35] [38] [39] [47] [51] [53] [55]. Потоки даних у режимі реального часу з мереж датчиків тіла (BSN) можуть викликати контекстуальні та ситуативні події. Управління та обробка цих подій (так звана аналітика великих даних) може бути корисним для прийняття рішень [46] [47] [51] [53] [55] і може, в свою чергу, викликати сервіси у відповідь на ці події.
- Відповідність стандартам. Передача даних до хмарних служб зберігання та обробки часто вимагає поліпшення якості даних і передбачає перетворення даних у стандартні формати (наприклад, HL7) для вирішення питань взаємодії, очищення даних, зменшення надмірності та агрегування [53] [55]. (Стандарт EHR, HL7 etc.)
- Безпека та конфіденційність. Автентичність та цілісність є ключовими з огляду на необхідність конфіденційності даних [28] [55]. [22] [35] [39] [53] включають у свої рішення конкретні рівні безпеки (конфіденційності та автентичності), цілісності даних та шифрування даних.

Також, варто виділити функціональні вимоги до медичних кіберфізичних систем. Отже, медична КФС повинна виконувати:

- Збір додаткових медичних даних з сторонніх сервісів.
- Функцію сповіщення про події.
- Виявляти критичні значення життєвих показників стану людини.
- Аналітику користувачів та їх життєвих показників, для генерації рекомендацій та виявлення подібностей.
- Обробляти та агрегувати дані для зберігання та подальшої візуалізації в стандартизованому форматі для семантичної сумісності медичних даних у разі потреби їх надсилання в зовнішні сервіси.
- Обробляти, зберігати та агрегувати історичні (архівні) медичні дані користувача.

1.3. Принципи зберігання даних в медичних кіберфізичних системах

Кіберфізичні системи потребують цілого ряду рішень для управління даними в залежності від їх бортової обчислювальної потужності та енергетичних обмежень. Існує декілька проблем, пов’язаних з даними в кіберфізичних системах: вибір системи управління даними, яка тісно відповідає функціональним та нефункціональним вимогам КФС, робота з якістю даних, питання конфіденційності [58] [59] та безпеки даних. Економія в масштабуванні, яку пропонують хмарні платформи, та їх розповсюдженість є вагомим аргументом для їх використання для управління даними кіберфізичних систем.

Якщо розглядати потреби управління базами даних кіберфізичних систем, то вони вони здатні генерувати величезні масиви даних у реальному часу. Такі бази даних дуже часто можуть розширюватись, тобто змінювати свою структуру. У багатьох КФС повна схема бази даних не існує заздалегідь і розвивається з часом. Крім того, деякі системи вимагають простої моделі даних і швидкий пошук по них має вирішальне значення. Вони вимагають операцій вставки та пошуку у дуже великих масштабах. Крім того, є потреба у декількох мовах

запитів, починаючи від простого інтерфейсу програмування Representational State Transfer (REST) (API) через складні та спеціальні запити та інтерактивній обробці спеціальних запитів масових паралельних обчислень із часовими даними, використовуючи такі фреймворки, як Hadoop та Spark.

В останні роки з’явилася велика кількість нових систем управління даними для задоволення вищезазначених потреб [50]. Ці нові системи згадуються в літературі під різними назвами, включаючи NoSQL, NewSQL та Not only SQL. Загальний термін NoSQL використовується для позначення тих систем, які не використовують SQL як основну мову запитів. Системи NewSQL - це СУБД з новими та інноваційними функціями. Деякі системи NoSQL можуть використовувати SQL і називаються Not only SQL.

Типи систем зберігання даних:

- **Реляційні системи управління базами даних** (СУБД) були основою майже всіх програмних програм до недавнього часу. В основі СУБД лежить реляційна модель даних для структурування даних та стандарт ISO / ANSI SQL [75] для обробки даних та запитів. Реляційна модель даних має визначену структуру і забезпечує декларативний метод для визначення запитів у базі даних.
- **Розподілені файлові системи**: Файлові системи, такі як файлова система Hadoop (HDFS), надають можливість надійно зберігати великі обсяги неструктурованих даних. Хоча існують файлові системи з кращою продуктивністю, HDFS є невід'ємною частиною фреймворку Hadoop і вже досяг рівня фактичного стандарту. Він розроблений для великих файлів даних і добре підходить для швидкого зберігання даних та масової обробки.
- **Бази даних NoSQL**: Можливо, найважливішим сімейством технологій зберігання великих даних є системи управління базами даних NoSQL. Бази даних NoSQL використовують моделі даних поза реляційним світом, які не обов'язково дотримуються транзакційних властивостей атомності, послідовності, ізоляції та довговічності (ACID).

- **Бази даних NewSQL**: Сучасна форма реляційних баз даних, яка спрямована на порівнянну масштабованість, як бази даних NoSQL, зберігаючи гарантії транзакцій, що надаються традиційними системами баз даних.
- **Платформи запитів Big Data**: технології, що забезпечують фасади запитів перед великими сховищами даних, такими як розподілені файлові системи або бази даних NoSQL. Головною проблемою є забезпечення інтерфейсу високого рівня, наприклад через SQL3, як мови запитів та досягнення низьких затримок запитів.

За задумом, системи NoSQL не забезпечують всіх функцій СУБД і в основному зосереджуються на забезпеченні читання та запису майже в реальному часі для додатків з мільйонами одночасних користувачів. Термін "ефективність у масштабі" використовується для опису основних характеристик таких систем. Системи NoSQL задовольняють потреби нових програм різним ступенем. Крім того, їх функціональність значно відрізняється одна від одної. Наприклад, Redis відомий своїми структурами даних та елегантними механізмами запитів, Riak дуже масштабований та доступний, MongoDB ефективно управляє глибоко вкладеними структурованими документами та обчислює сукупності в документах.

КФС є повсюдними в інтелектуальних мережах, транспорті, промисловому контролі та інших важливих інфраструктурах. Ці системи повинні працювати надійно у разі зловмисних кібератак та непередбачених порушень. У порівнянні із реляційними СУБД, безпека даних є головною проблемою для систем NoSQL. Ризики безпеки варіюються від припущення, що системи NoSQL працюють у надійному середовищі (отже, не потрібна автентифікація), до очищення передачі тексту та відсутності шифрування даних на диску. Авторизація та остаточний контроль доступу є ще одним ризиком для безпеки. Оскільки системи NoSQL зберігають дублікати копій даних для оптимізації обробки запитів, важко виділити конфіденційні дані та вказати елементи керування доступом. Хоча рішення безпеки RDBMS застосовуються в принципі до систем NoSQL,

відмінності в моделі даних, мовах запитів та методах клієнтського доступу NoSQL вимагають нових рішень.

Проблеми кібербезпеки в системах NoSQL посилюються швидкістю передачі даних, обсягом та неоднорідністю. Крім того, різноманітність джерел даних, потокових даних, розміщення в хмарі та великі обсяги міжхмарних переміщень даних ще більше посилюють питання безпеки. Традиційні механізми безпеки, призначені для захисту дрібних статичних даних, недостатні для систем NoSQL. В якості основних загроз безпеки бази даних можна виділити: надмірні, недоречні та невикористані привілеї, зловживання приватною службою, недостатня безпека веб-додатків [62] / ін'єкція SQL, слабке логування дій та незахищені носії інформації [70]. У більшості випадків ці загрози можна запобігти, застосувавши прості кроки та дотримуючись найкращих практик та внутрішнього контролю.

Як зазначалося раніше, реляційні СУБД не є ідеальними для КФС, великих даних, мобільних обчислень чи інших подібних систем. СУБД в основному розроблені для роботи на одновузлових комп'ютерах. Вони дозволяють збільшити робоче навантаження та обсяг даних, використовуючи швидші процесори, більше пам'яті та більші та швидші диски. Цей підхід до збільшення обчислювальних потужностей називають вертикальним. Рішення постачальників СУБД для вертикального масштабування, як правило, вимагають дорогого обладнання та часто залучають власне програмне забезпечення.

Горизонтальне масштабування, навпаки, стосується реалізації підвищеної обчислювальної потужності за допомогою обчислювального кластера, побудованого з декількох недорогих комп'ютерів. Останні, як правило, будуються з компонентів, які відносно недорогі, широко доступні та легко взаємозамінні. У порівнянні з горизонтальним масштабуванням, вертикальне масштабування є більш дорогим та обмежувальним.

Розподілені обчислення - це парадигма, при якій мережеві комп’ютери вирішують обчислювальні проблеми за допомогою спілкування та координації шляхом передачі повідомлень. Поточне виконання, відсутність глобального

годинника та незалежний збій компонентів - характеризують розподілені системи. Розподілені алгоритми необхідні для вирішення розподілених системних проблем, таких як атомна фіксація, консенсус, розподілений пошук, вибірку, взаємне виключення та розподіл ресурсів [32]. У контексті систем NoSQL розподілені обчислення є засобом забезпечення масштабної продуктивності та досягнення високої доступності та надійності [56].

Архітектури NoSQL складаються з декількох компонентів (наприклад, дисків зберігання, процесорів, серверів додатків), що перебувають на мережевих комп'ютерах. Компоненти спілкуються та координують свої дії для досягнення спільної мети за допомогою таких механізмів, як поділена пам'ять та передача повідомлень. У контексті систем NoSQL ми визначаємо вузол обробки як автономний комп'ютер, що складається з центрального процесора, оперативної пам'яті та дискового сховища. Логічна сукупність вузлів називається кластером. Деякі системи NoSQL працюють на кластерах, вузли яких знаходяться в географічно відокремлених центрах обробки даних. У всіх випадках вузли взаємопов’язані за допомогою високошвидкісних комп’ютерних мереж.

Сервер NoSQL зазвичай працює на кластері у виробничих середовищах. Повторна відповідальність за обробку запитів клієнтів та розподіл та координацію робочого навантаження між різними вузлами може бути централізованою або розподіленою.

Деякі системи NoSQL дозволяють додавати нові вузли або видаляти існуючі (навмисно або через відмову вузла) без переривання обслуговування. Системи NoSQL, засновані на архітектурі, що не має спільного доступу, забезпечують збільшення робочих навантажень за допомогою горизонтального масштабування.

Деякі системи NoSQL доступні як сервіс, розміщений у хмарі, з назвою база даних як сервіс (DBaaS). Це дозволяє організації користуватися послугами баз даних без необхідності місцевої технічної експертизи або локальної інсталяції систем NoSQL. Хмарний оператор надає ресурси на вимогу, щоб задовольнити

робочі навантаження програми пружною масштабованістю. Наприклад, Amazon Simple Storage Service та DynamoDB - це два приклади DBaaS.

Оперативна база даних - це база даних, яка використовується для управління та зберігання даних у режимі реального часу. Ці бази даних можуть бути на основі SQL або NoSQL, де остання спрямована на операції в режимі реального часу. Ключовою характеристикою оперативних баз даних є їх орієнтація на операції в режимі реального часу, порівняно зі звичайними базами даних, які спираються на пакетну обробку. За допомогою оперативних баз даних записи можна додавати, видаляти та змінювати в режимі реального часу. Оперативні системи управління базами даних можуть базуватися на SQL, але все більша кількість використовує NoSQL та неструктуровані дані.

Використання якогось єдиного рішення не задовольняє виконання всіх функцій МКФС, оскільки дані будуть використовуватись різними сервісами, для різних задач в різних форматах, з різним рівнем доступу.

1.4. Постановка задачі оцінювання стану людини в медичних кіберфізичних системах

Сьогодні світ стикається зі збільшенням кількості захворювань та пацієнтів. Крім того, війни, забруднення навколишнього середовища, хвороби, пов’язані з їжею, та стосунки між людиною та тваринами спричиняють появу та поширення нових видів хвороб та вірусів. Згодом, з появою невідомої хвороби, уряди стикаються з двома основними проблемами: по-перше, медичні центри потребують дедалі більше кваліфікованого персоналу, щоб періодично контролювати кожного пацієнта та швидко діяти при виявленні надзвичайного стану. По-друге, правильний діагноз стану пацієнта та прогрес його ситуації є критично важливими для медичного персоналу. Крім того, інтеграція нових технологій в охорону здоров'я стала важливим завданням для того, щоб зробити лікарні більш ефективними з точки зору моніторингу та аналізу даних пацієнтів, таким чином, покращуючи медичні процеси.

Нещодавна поява сенсорних пристроїв, особливо бездротової сенсорної мережі (WSN) та Інтернету речей (IoT), а також технологій великих даних, таких як екосистема Hadoop, призводить до нової революції в галузі охорони здоров'я. В основному, медична кіберфізична система, складається з набору біомедичних датчиків, що дозволяє постійно контролювати життєві показники (наприклад, частоту серцевих скорочень, частоту дихання, рівень кисню в крові, температуру, кров’яний тиск тощо) пацієнта та періодично передавати зібрані дані відповідальній особі для подальшої перевірки. Розумні технології відкрили цілий світ застосувань у діагностиці та лікуванні захворювань, таких як рак, моніторинг глюкози, депресія, хвороба Паркінсона, підключені контактні лінзи тощо. Крім того, це стало серйозною мотивацією для медичних організацій до значних вкладень у аналіз великих даних.

Однак, медичні кіберфізичні системи, як і раніше, викликають труднощі у наукової спільноти через:

- Збір і зберігання великих даних: біосенсори постійно реєструють життєві показники пацієнтів, як правило щосекунди, а потім вони надсилають свої записи до центру зберігання даних. Життєві показники пацієнтів мають три характеристики: масовий збір даних; високошвидкісна генерація; та неоднорідна природа. На жаль, реляційні системи управління базами даних і традиційні технології зберігання даних не можуть обробляти такий тип і обсяг даних.
- Швидке виявлення надзвичайних ситуацій: Зазвичай для кожного життєво важливого показника визначається нормальний діапазон здоров'я. Записи за межами цього діапазону призводять до ненормальної ситуації, яку слід швидко виявити та повідомити про це медичному персоналу, щоб вжити відповідних заходів. Отже, швидке виявлення та реагування на надзвичайні ситуації є вирішальним, тому що може загрожувати безпеці або навіть життю пацієнта.

- Діагностика захворювань та класифікація пацієнтів: У сферах охорони здоров’я кілька захворювань можуть викликати подібні симптоми. Отже, діагностика хвороби сприйнятлива до людських помилок, які можуть спровокувати інвалідність або смерть. Тому розробка нових методів та алгоритмів з метою вивчення, з одного боку, кореляції між симптомами, а з іншого боку, класифікації пацієнтів відповідно до їх ситуації може допомогти прийняти правильне рішення.

Щоб подолати вищезазначені проблеми, пропонується розробити ефективну та надійну медичну кіберфізичну систему оцінювання стану людини, яка базується на запропонованій моделі обробки життєвих показників людини. Ця модель покриває 2,3 та 4 рівні (див Рис. 1.1) медичної КФС.

1.5. Висновки до розділу

1. Проведено аналіз існуючих підходів до побудови медичних кіберфізичних систем, а саме їх структури та архітектурних рішень серверного програмного забезпечення. Аналіз показав, що при проектуванні та реалізації медичних КФС, потрібно враховувати, що медичні дані є великими даними, а їх обробка повинна відбуватись паралельно.

2. Згідно проведеному аналізу визначено архітектурні та функціональні вимоги до медичних КФС. Основними архітектурними вимогами є: неоднорідність та динамічне підключення сенсорів; масштабованість; сумісність; безпека та конфіденційність.

3. Проведено аналіз принципів зберігання даних в медичних КФС. Аналіз показав, що використання якогось єдиного рішення не задовольняє виконання всіх функцій МКФС, оскільки дані будуть використовуватись різними сервісами, для різних задач в різних форматах, з різним рівнем доступу.

РОЗДІЛ 2

МЕТОДИ, ЗАСОБИ ТА МОДЕЛЬ ОЦІНЮВАННЯ СТАНУ ЛЮДИНИ В МЕДИЧНІЙ КІБЕРФІЗИЧНІЙ СИСТЕМІ

2.1. Засоби обробки інформації в медичних кіберфізичних системах

Результат виконання своїх функцій засобами обробки інформації в медичних кіберфізичних системах є оцінкою стану людини. Такі засоби повинні обробляти необроблені дані життєвих показників, отримані з вимірювальних пристроїв для того, щоб надати цим даним цінності в процесі моніторингу та оцінювання стану людини.

Засоби обробки інформації повинні задовольняти встановлені раніше вимоги до медичних кіберфізичних систем та вирішувати поставлені задачі. Кожен засіб повинен виконувати свою роботу незалежно від інших засобів, що дозволяє розпаралелити процес обробки інформації. Це являється важливим аспектом в медичній КФС, оскільки деякі засоби повинні давати результат якомога швидше, від чого напряму може залежати стан здоров'я людини. У випадку, коли певний засіб залежний від результату роботи іншого, потрібно продумати приорітети цих засобів, тим самим визначити порядок виконання обробки даних цими засобами.

Одним з найважливіших і пріоритетних засобів обробки інформації в медичній КФС є засіб виявлення критичних показників. При поступленні даних про життєві показники людини, у першу чергу потрібно визначити наскільки критичними ці показники є. Щоб цей засіб міг виконувати оцінювання показників стану людини потрібно використати певний метод оцінювання життєвих показників людини, за допомогою якого буде прийняте рішення про виявлення події критичності показників. Варто зазначити, що засіб виявлення критичних показників повинен виконувати свої функції якомога швидше. Не менш важливою функцією цього засобу є миттєве реагування на цю подію, оскільки це може підвищити шанси людини на вчасне регулювання цих

показників без шкоди для здоров'я. Отже, засіб виявлення критичних життєвих показників людини є одним з ключових в плані оцінювання стану людини в медичних КФС. Оскільки реагування на події є ключовим моментом засобу виявлення критичних показників стану людини, постає потреба в засобі сповіщень про події.

Засіб сповіщень про події повинен виконувати функцію інформування відповідальної особи чи групи осіб, які ведуть моніторинг стану людини. Події можуть бути різного походження та рівня серйозності. Наприклад, може бути подія, викликана на вимогу самим пацієнтом, метою якої є сповіщення відповідальної особи про цю подію. Також події можуть створюватись іншими засобами обробки інформації в медичній КФС, такими як засіб виявлення критичних показників та ін. Тобто це події, які створюються в результаті певної обробки даних автоматично. Потрібно враховувати, що ці події можуть бути критичними або інформаційними, тому засіб сповіщень про події повинен враховувати рівень серйозності цих подій і використовувати відповідні методи сповіщень про них. Отже, засіб сповіщень про події може мати декілька способів сповіщень, і навіть різні кінцеві точки цих сповіщень. А саме, певні сповіщення про події повинні бути негайними і адресовані відповідальній особі за допомогою розроблених способів негайних сповіщень, а ціль таких способів є досягнути сповіщення кінцевої точки в найбільш ефективний і зручний спосіб . Якщо подія не є критичного рівня, то сповіщення буде відправлено в іншій формі. Таким чином, реагування на події різного рівня серйозності теж повинне бути пріоритезованим. Кінцевою точкою сповіщення про подію не обов'язково повинна бути відповідальна особа чи група осіб, це також може бути інший засіб обробки даних людини, який повинен виконуватись лише при певних подіях. Загалом, засіб сповіщень про події частково пов'язаний з іншими засобами обробки інформації в МКФС, оскільки він повинен реагувати на події, які можуть створюватись в цих засобах. Отже, основним завдання засобу сповіщення про події це інформування інших засобів чи відповідальних осіб про ту чи іншу подію. Також, крім інформації про умови створення події, ці

сповіщення можуть містити додаткове повідомлення для кінцевої точки згідно визначених методів сповіщень про події, які використовуються для різних типів подій.

Не менш важливим засобом обробки інформації в МКФС є засіб класифікації пацієнтів та діагностики захворювань. Порівнюючи з засобом виявлення критичних показників, він не є настільки пріоритетним, і не має обмежень в тривалості виконання своїх функцій. Основними завданнями цього засобу є класифікація людей за певними схожими параметрами, визначені в методах класифікації цього засобу, а також діагностика захворювань на основі методів діагностики симптомів та захворювань. Цей засіб повинен спростити завдання лікаря як відповідальної особи в правильно встановленому діагнозі. Також, цей засіб може виконувати функцію прогнозування показників стану людини, а також пропонувати поради на основі методів прийняття рішень, які базуються на цьому прогнозуванні. Отже, цей засіб також є одним з ключових в плані оцінювання стану людини.

Невід'ємною складовою медичної КФС є візуалізація даних життєвих показників стану людини. Правильне і зручне візуальне подання цих даних може суттєво покращити та пришвидшити роботу відповідальної особи, яка веде моніторинг та оцінювання стану людини. Для забезпечення швидкої відповіді на запит клієнта відповідальної особи, а також швидше візуальне відображення цих даних на клієнті, пропонується розробити засіб обробки даних для зберігання і подальшої візуалізації. Цей засіб має 2 основних завдання: обробка необроблених даних в плані їх структуризації; забезпечення зручної структурованої відповіді на запит клієнта. Оскільки дані про стан життєвих показників людини поступають в медичну КФС у вигляді необроблених даних, постає потреба в обробці цих даних та їх структуризації. Збереження даних в правильному уніфікованому форматі забезпечить сумісність з різними типами клієнтів, а також швидку відповідь на запит клієнтів, оскільки при запиті на отримання цих даних не доведеться їх обробляти додатково.

Оскільки історія показників стану людини може містити корисну інформацію для відповідальної особи, постає завдання в архівуванні даних життєвих показників стану людини. Це завдання виконується засобами збереження та доступу до архівних даних. Архівування даних про пацієнтів є ключовою операцією в процесі лікування людини. З одного боку, це дозволяє відповідальній особі помічати прогрес у стані пацієнта з плином часу, а з іншого - допомагає професіоналам краще зрозуміти захворювання та покращити якість медичної допомоги. Потрібно врахувати, що архівні дані можна вважати як великі дані, тому ці засоби повинні бути спроектовані для роботи з такими даними в питанні обробки, агрегації та збереження архівних даних. Варто зазначити, що архівні дані являють собою не надто актуальні дані, і питання швидкої відповіді на запит отримання архівних даних не актуальне.

Всі ці засоби обробки інформації в медичній КФС безумовно є важливими для оцінювання стану людини. Проте, варто зазначити що вони можуть мати різний пріоритет в плані негайності виконання своїх функцій. Постає питання розмежуванні цих засобів на окремі незалежні логічні елементи. Для вирішення цього питання потрібно побудувати правильну модель обробки інформації в МКФС, в яку можна інтегрувати вище описані засоби. Всі засоби повинні працювати паралельно, щоб своєчасно реагувати на зміну стану людини, що є ще одною вимогою до моделі обробки інформації медичної КФС.

2.2. Модель обробки інформації в медичних кіберфізичних системах

Модель обробки інформації в медичних кіберфізичних системах повинна задовольняти вимоги для об’єднання та реалізації засобів обробки життєвих показників стану людини.

Оскільки вище наведені засоби виконують одночасні і незалежні один від одного задачі, доцільно їх виконувати паралельно. Для розмежування засобів обробки інформації на окремі незалежні логічні елементи доцільно використати модель обробки інформації на основі мікросервісної архітектури. Я пропоную

застосувати мікросервісну архітектуру, де кожен засіб буде у вигляді мікросервісу.

Мікросервіси - або архітектура мікросервісів - це програми, які упорядковані або структуровані як сукупність вільно пов’язаних сервісів. Загалом, мікросервіси мають такі характеристики:

- Підвищена продуктивність.
- Кожен мікросервіс має власну модель даних та керує власними даними.
- Дані переміщуються між мікросервісами за допомогою "тупих каналів", таких як брокер подій та/або легкий протокол типу REST.
- Невеликий обсяг, який охоплює єдину функціональність сервісу.
- Внутрішні операції - це «чорна скринька», доступна для зовнішніх програм лише через API.

Підвищена продуктивність:

- Розбиття програми на менші автономні фрагменти полегшує її створення та обслуговування. Кожен сервіс може бути розроблений, розгорнутий та управлятися незалежно, а також може використовувати різні мови програмування, різні технології та різні середовища програмного забезпечення залежно від потреб кожної з них.
- Скорочена база кодів кожного модульного елемента програми робить реліз, масштабування, розгортання та тестування різних служб більш керованими.

Мікросервіси добре працюють з гнучкими процесами розробки та задовольняють зростаючу потребу у більш плавному потоці інформації.

Мікросервіси розгортаються незалежно один від одного і забезпечують більшу автономію команди розробників серверної частини медичної кіберфізницної системи:

- Кожен мікросервіс може бути розгорнутий незалежно, за необхідності, що забезпечує постійне вдосконалення та швидше оновлення програм.
- Конкретні мікросервіси можуть бути призначені певним командам розробників, що дозволяє їм зосередитися виключно на одному сервісі чи функції. Це означає, що команди можуть працювати автономно, не турбуючись про те, що відбувається з рештою програми.

Мікросервіси незалежно масштабовані. Оскільки кількість користувачів та даних в системі зростає, її легше масштабувати за допомогою мікросервісів. Ви можете збільшити ресурси для найнеобхідніших мікросервісів, а не масштабувати всю систему. Це також означає, що масштабування є більш швидким і часто більш економнішим. Нові мікросервіси можуть додаватись до медичної КФС не вимагаючи простоїв системи.

Мікросервіси скорочують простої системи за рахунок ізоляції несправностей. Якщо певний мікросервіс виходить з ладу, ви можете ізолювати цей збій у цьому єдиному сервісі та запобігти каскадним збоям, які спричинили б збій системи. Ця ізоляція несправностей означає, що ваша критична система може залишатись справною і працювати, навіть якщо один з її модулів виходить з ладу.

Запропонована модель виглядає наступним чином (див Рис. 2.1) та складається з таких ключових елементів:

- сервіс прийняття рішень роботи мікросервісів;
- оперативна БД;
- БД для організації черги процесів;
- N-кількість мікросервісів з власною БД, за потреби.

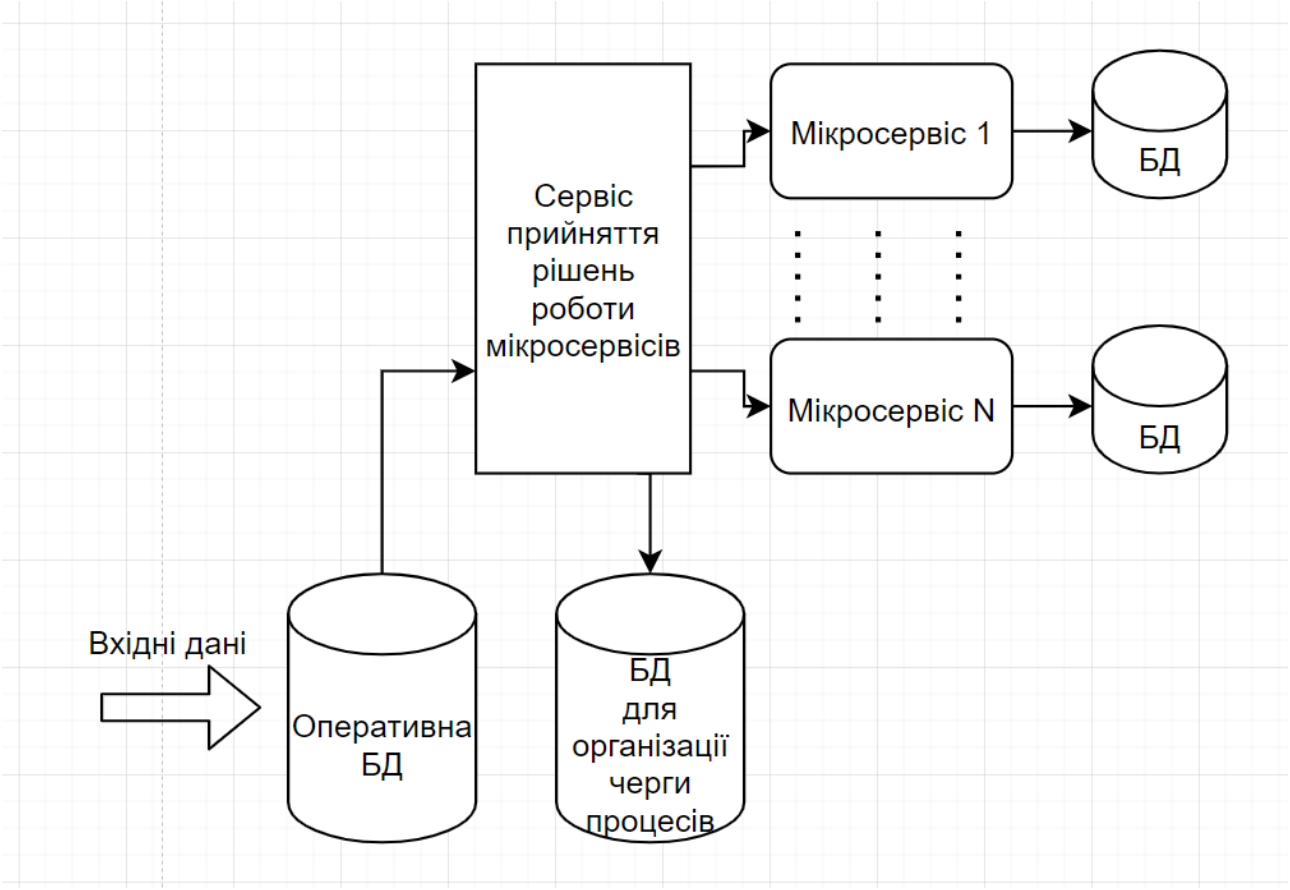

Рисунок 2.1. Запропонована модель обробки інформації в медичних кіберфізичних системах

Джерелом даних для запропонованої моделі являється оперативна база даних. Як зазначалося раніше, модель приймає сирі, тобто не агреговані та необроблені дані. Це означає, що цих даних багато, і зберігання їх в єдиному місці сховища призводить до ряду проблем. Для уникнення цих проблем пропонується саме оперативна БД, суть якої полягає в тимчасовому зберіганні даних. Дані зберігаються в ній рівно до того моменту часу, допоки усі засоби обробки інформації - мікросервіси - виконають своє завдання, тобто загальний процес обробки необроблених даних закінчиться. Зберігати дані в такому вигляді надалі не актуально, тому вони видаляються. Це дозволяє не переповнювати оперативну базу даних, що зберігає швидкодію читання-запису даних. Це також вирішує питання очікування відповіді на запит запису даних, адже механізм передачі даних з біосенсорів у систему не повинен очікувати обробки інформації, а отримує негайну відповідь про прийняття інформації до асинхронної обробки.

Контроль виконання засобами обробки інформації їх завдань відбувається у одному з найважливіших елементів запропонованої моделі - сервісі прийняття рішень роботи мікросервісів. Саме цей сервіс визначає, коли дані повинні бути видалені з оперативної бази даних. Сервіс прийняття рішень роботи мікросервісів є мікросервісом, який формує чергу з завдань різних засобів обробки інформації, розставляє пріоритети цих засобів, стежить за їх виконанням. Також, за потреби, він дозволяє мікросервісам спілкуватись між собою. Ще одним завданням цього сервісу є агрегація даних для відповіді на запит клієнта, тобто для різних запитів він спілкується з різними мікросервісами, які відповідають за ті чи інші засоби обробки інформації.

Для управління і формування черги завдань різних засобів обробки інформації, запропоновано використати спеціальну базу даних. Сервіс прийняття рішень створює пріоритетні черги, які зберігаються в цій БД. Після виконання мікросервісом свого завдання, запис з черги видаляється. Тобто, ця БД є списком завдань, які заплановані виконуватись мікросервісами обробки інформації. Саме на основі цього списку і приймається рішення про видалення необроблених даних з оперативної бази даних, при умові що всі сервіси виконали своє завдання для даного набору життєвих показників стану людини. За допомогою цієї БД, різні процеси обробки інформації мікросервісами можуть виконуватись асинхронно і паралельно.

В запропонованій моделі, кожен вище запропонований засіб обробки даних МКФС є мікросервісом, який вирішує чітко поставлену перед ним задачу, що є основною концепцією мікросервісної архітектури. В залежності від кількості засобів, які пропонується використати в медичній КФС, кількість мікросервісів може бути різна. Це дозволяє будувати гнучку архітектуру медичної кіберфізичної системи, в якій можна додавати чи вилучати певні засоби обробки інформації навіть після стадії проектування системи.

2.3. Метод виявлення критичних показників в медичній кіберфізичній системі.

Сповіщення про події можна розділити на автоматичні та на вимогу. Автоматичні сповіщення створюються про події, які були визначені в процесі певної обробки показників чи інших даних людини. В свою чергу сповіщення на вимогу створюються лише тоді, коли, наприклад, пацієнт вручну хоче викликати до себе допомогу. Генерація такого типу сповіщень забезпечується попередньо створеним механізмом на сенсорі чи мобільному терміналі людини, який робить запит до сервісу сповіщень.

Прикладом автоматичних сповіщень про події може бути аналіз актуальних показників біосенсорів людини. Отримавши періодичні дані, надіслані від усіх біосенсорів, сервіс виявлення критичних подій повинен проаналізувати їх у реальному часі та попередити медичний персонал у разі виявлення надзвичайної ситуації.

У медичній КФС використано алгоритм виявлення надзвичайних ситуацій, який дозволяє сповіщати про будь-яку ненормальну ситуацію пацієнта, а також визначає відповідну реакцію, яку повинен приймати медичний персонал. Щоб перевірити ненормальні ситуації, пропонується використати посібник з оцінки раннього попередження (EWS).

EWS - це посібник, заснований на життєвих показниках, наприклад, V, і він використовується медичним персоналом у лікарні для відстеження рівня критичності пацієнта. Для кожного життєвого показника $v \in V$ зібраний запис $r_i \in {}_tR_v^p$ порівнюється з нормальним діапазоном для обчислення оцінки s_i між 0 і 3; 0 означає нормальний запис, де інші значення вказують на ненормальні ситуації зі збільшенням тяжкості відносно збільшенням балу. Отже, для ${}_tR_v^p$ розраховується набір балів записів ${}_tS_v^p = [s_1, s_2, \cdots, s_\tau]$. На таблиці 1 зображено шкалу оцінювання життєвих показників стану людини, яка базується на одному з найбільш використовуваних посібників EWS, який розроблено у

Великобританії та розповсюджується по всьому світу, і називається National EWS (NEWS) [71].

Таблиця 2.1. Шкала оцінювання життєвих показників стану людини.

Життєвий показник	Оцінка						
	3	2	1	0	1	2	3
частота дихання (кст/хв)	≤8		9-11	12-20		21-24	≥25
$\boldsymbol{SpO_2}$, шкала 1 (%)	≤91	92-93	94-95	≥96			
$\boldsymbol{SpO_2}$, шкала 2 (%)	≤83	84-85	86-87	88-92 ≥93 при повітрі	93-94 при кисні	95-96 при кисні	≥97 при кисні
повітря чи кисень		Кисень		Повітря			
систолічний кров'яний тиск (мм рт. ст.)	≤90	91-100	101-110	111-219			≥ 220
Пульс (за хвилину)	≤40		41-50	51-90	91-110	111-130	≥131
Температура тіла (°C)	≤35.0		35.1-36.0	36.1-37.0	37.1-38.0	38.1-39.0	≥ 39.1

Після підрахунку балів, встановлених на кожному періоді, пропонується безпосередньо проаналізувати зібрані записи та попередити, відповідальну особу шляхом відповідної реакції. Спеціальний сервіс медичної КФС обчислює агрегований бал записів, отриманих за кожний період, а потім надсилає відповідний сигнал – сповіщення про подію – відповідальній особі згідно з таблицею 2.2. Наприклад, якщо встановлений бал, розрахований протягом періоду, становить ${}_tS_v^p = [0,2,1,2]$, тоді загальний бал дорівнює 5 (тобто 0 + 2 + 1 + 2), і сервіс надсилає попередження відповідальній особі, повідомляючи, що показники перетинають встановлені допустимі межі з певним рівнем критичності.

Таблиця 2.2.Залежність рівня критичності та рекомендацій від сукупної оцінки життєвих показників стану людини.

Сукупна оцінка	**Рівень критичності**	**Рекомендація**
0	1	Відсутні
1-4	2	Моніторити життєві показники з певною періодичністю, допоки вони не прийдуть в норму. Вияснити причину погіршення показників.
5+ або хоча б один з показників 3	3	Критична ситуація. Вимагає постійного моніторингу та дій для відновлення стабільного стану людини.

Алгоритм описує процес виявлення критичної ситуації та реагування, який застосовується у сервісі сповіщень. Алгоритм бере в якості входу вектор записів, зібраних усіма біосенсорами за період часу t. Потім він обчислює оцінку для кожного запису, а потім сукупний бал (рядки 2–6). Нарешті, сервіс надсилає попередження відповідальній особі із зазначенням рівня критичності, та рекомендації, яку слід прийняти відповідно до ситуації користувача (рядки 7 та 8).

Алгоритм виявлення екстреної ситуації та клінічного реагування – сповіщення про подію.

Вимоги: Пацієнт – P, Набір біосенсорів – B_v^p, Період часу – t, Записи, зібрані протягом проміжку часу $t - {}_tR_v^p = [r_1, r_2, \cdots, r_\tau]$.

Забезпечує: Сповіщення про подію.

1: Початок

2: $Sum = 0$

3: **for** кожного запису $r_i \in {}_tR_v^p$ **do**

4: обчислити оцінку s_i запису r_i згідно EWS

5: $Sum += s_i$

6: **end for**

7: Визначити відповідні рівень критичності та рекомендацію базуючись на Sum

8: Надіслати сповіщення про подію відповідальній особі

9: Кінець

2.4. Алгоритм класифікації пацієнтів

Першим кроком аналізу даних є класифікація пацієнтів у кластери, де пацієнти в одному кластері мають загальні характеристики (симптоми та ситуації). Класифікація даних є добре відомим підходом для більшості областей. Можна знайти величезну кількість алгоритмів класифікації даних, таких як K-means, дерево рішень тощо [31]. Однак кластеризація K-means все ще є найбільш використовуваним алгоритмом класифікації, оскільки його можна ефективно адаптувати до величезної кількості додатків. На жаль, K-means має значний негативний вплив на затримку даних, особливо в додатках, які виробляють велику кількість даних, таких як медична кіберфізична система. Щоб подолати цю проблему, я пропоную адаптовану версію K-means, яка конкретно присвячена медичним кіберфізичним системам.

Щоб класифікувати подібні ситуації пацієнтів за допомогою адаптованого алгоритму K-means, спочатку потрібно визначити рівень стабільності кожного користувача під час його моніторингу (згідно з його архівом), а потім призначити користувача до найближчої групи стабільності користувачів.

Отримано з ${}_tR_v^p$, визначаємо підмножину ${}_t^sR_v^p$, яка містить лише записи життєвих показників $v \in V$ з оцінкою s у ${}_tR_v^p$, де $s \in [0,3]$. Відповідно, ${}_t^sR_v^p$ - це підмножина з оцінками ${}_t^sS_v^p$ для набору записів ${}_tR_v^p$. Таким чином, ${}_tR_v^p = \cup_{i=0}^3 {}_t^iR_v^p$, тоді як ${}_tS_v^p = \cup_{i=0}^3 {}_t^iS_v^p$. Нехай $|X|$ - нульова норма, тобто кількість ненульових елементів у наборі X. Для простоти припустимо порцію записів, зібраних для двох користувачів p і q протягом періоду t, де два вектора запису ${}_tR_v^p$ і ${}_tR_v^q$ зберігаються відповідно для p та q. Тоді рівень стабільності, відзначений як $stab$, життєвих ознак v двох користувачів, можна обчислити як перекриття між кількістю подібних записів (з однаковими оцінками) у ${}_tR_v^p$ та ${}_tR_v^q$, як показано в наступному рівнянні:

$$stab\left({}_tR_v^p, {}_tR_v^q\right) = \frac{\sum_{k=0}^3 \min(|{}_t^kR_v^p|,|{}_t^kR_v^q|)}{\min(|{}_tR_v^p|,|{}_tR_v^q|)} \times 100 \qquad (2.1)$$

де $\left|{}_t^kR_v^p\right|$ означає кількість записів з оцінкою k у ${}_tR_v^p$. Тому діапазон $stab$ коливається від 0 до 100, де 0 означає, що обидва пацієнти мають однакову стабільну ситуацію, а 100 вказує на дуже нестабільну ситуацію для обох пацієнтів.

В основному, адаптований алгоритм K-means дозволяє групувати пацієнтів із схожою ситуацією, щоб знайти кореляцію між пацієнтами та зрозуміти поведінку певних захворювань. Пацієнтів можна класифікувати за одним або всіма життєвими ознаками залежно від медичних потреб. Подібно до традиційних K-means, ідея адаптованого K-means полягає в тому, щоб спочатку визначити позитивну кількість кластерів (K), а потім набір K наборів даних випадковим чином вибирається як початкові центроїди кластерів. Після цього для кожного набору даних обчислюються відстані до всіх центроїдів, де набір даних призначається до найближчого. На цьому етапі новий центроїд перераховується для кожного кластера, і процес повторюється до збіжності алгоритму. Очевидно, що оптимальну кількість кластерів (K) можна вибрати,

наприклад, залежно від кількості пацієнтів у лікарні та їхньої ситуації. Однак, порівняно з традиційним K-means, адаптований має дві основні відмінності:

- По-перше, користувачі групуються за результатами записів, а не за самими записами.
- По-друге, в кожній ітерації пацієнта призначають до найближчого центроїду на основі рівня стабільності, обчисленого за допомогою формули 2.1, а не евклідової відстані по всьому вектору запису, як у традиційному K-means. Це призводить до значного зниження витрат на розрахунок.

Алгоритм: Адаптований K-means алгоритм.

Вимоги: Набір пацієнтів: $P = [P_1, P_2, \cdots, P_\tau]$, Період часу t, Життєві показники: $v \in V$, Набір збережених записів: ${}_tR_v = [{}_tR_v^1, {}_tR_v^2, \cdots, {}_tR_v^\tau]$, Номер кластера: K.

Забезпечує: набір кластерів пацієнтів.

1: ${}_tS_v \leftarrow \emptyset$ // $\emptyset$ порожній набір

2: **for** $i = 1\ to\ \gamma$ **do**

3: обчислити ${}_tS_v^i$

4: ${}_tS_v \leftarrow {}_tS_v \cup \{{}_tS_v^i\}$

5: **end for**

6: **for** $i = 1\ to\ K$ **do**

7: випадково обрати центроїд c_i серед ${}_tS_v$ який належить C_j

8: **end for**

9: $C \leftarrow \emptyset$

10: **repeat**

11: **for** кожного набору оцінок ${}_tS_v^i \in {}_tS_v$ **do**

12: // обчислити стабільність між ${}_tS_v^i$ та всіма центроїдами

13: **for** $k = 1\ to\ K$ **do**

14: $st_k = stab({}_tS_v^i, c_k)$

15: **end for**

16: Призначити ${}_tS_v^i$ до кластеру C_k з найближчим цетроїдом c_k

(напр., $|st_k - c_{j*}| \leq |st_k - c_j|; j \in \{1, \dots, K\}$)

17: **end for**

18: Оновити центроїди всіх кластерів

19: **until** не буде досягнуто збіжність в кластері

20: повернути C

Алгоритм описує засоби K-means адаптовані до рівня стабільності, щоб класифікувати користувачів відповідно до ситуації життєвих ознак. По-перше, він обчислює вектор оцінки набору записів для користувача (рядки 1–5). Потім він випадковим чином вибирає набір векторів оцінок K як початкові центроїди кластерів (рядки 6-8). Для кожної ітерації він обчислює стабільність між кожною оцінкою вектора та усіма кластерними центроїдами та призначає пацієнта до найближчого стабільного центроїда (рядки 10–17). Нарешті, алгоритм обчислює нові центроїди кластерів і повторює їх до тих пір, поки не буде досягнуто збіжність в кластері.

2.5. Алгоритм діагностики захворювань

Вивчення кореляції між життєвими ознаками пацієнтів в одному кластері може допомогти аналітиці медичних даних зрозуміти причини захворювання та поведінку, а отже, уникнути помилкової діагностики захворювання та знайти відповідне лікування.

Постановка правильного діагнозу - ключ до пацієнта. В іншому випадку наслідки неправильної діагностики є найбільш руйнівними, що може спричинити обструкцію, неправильне призначення ліків або навіть смерть. Отже, діагностика захворювання з кожним днем потребує все більшої уваги як клініцистів, так і дослідників. Дійсно, виявлення кореляції між варіаціями показників життєдіяльності одного і того ж пацієнта є одним із найефективніших способів постановки правильного діагнозу. Ідея такого підходу полягає у встановленні зв'язку між варіаціями життєво-важливих ознак та симптомами, що з'явилися у пацієнта, таким чином, клініцистам стає легше визначити захворювання. Більш формально, якщо дві або більше життєво-важливих ознаки значною мірою корелюють у величезної кількості пацієнтів, а також схожі симптоми з'явилися у цих пацієнтів, то точність діагностики захворювання підвищується. Пропонується методика діагностики захворювання, що базується на адаптованій версії алгоритму Association rule learning, яка застосовується в кожному кластері після класифікації пацієнтів.

Як і в традиційному алгоритмі, адаптована версія алгоритму Association rule learning складається з двоетапного процесу: знайти часті оцінки та створити міцну асоціацію. Однак, адаптована версія присвячена медичним кіберфізичним системам та враховує рекордні оцінки при знаходженні кореляції між життєвими показниками. Припустимо, що кластер містить θ пацієнтів, де $\theta < \gamma$, з відповідними збереженими векторами запису ${}_tS_v = \left[{}_tS_v^1, {}_tS_v^2, \cdots, {}_tS_v^\gamma\right]$. Обидва етапи, використані в адаптованому алгоритмі, можна описати наступним чином:

- Знайти часті оцінки: Цей крок має на меті знайти найбільш часто повторювані оцінки кожного життєвого ознаки пацієнта для досягнення певного порогу. Поріг можна назвати мінімальною силою балів, вираженою як μ, яка приймає значення від 0 до 100. Потім ми обчислюємо силу, str, кожної оцінки ${}_t^sS_v^p \in S_v^p$, де $s \in [0,3]$, за формулою 2.2.

$$str\left({}_t^sS_v^p\right) = \frac{\left|{}_t^sS_v^p\right|}{\left|{}_tS_v^p\right|} \times 100 \qquad (2.2)$$

Отже, якщо $str\left({}_t^s S_v^p\right) \geq \mu$, то оцінка вважається сильною. Дійсно, якщо ознака життєдіяльності містить більше одного бала, то за життєвою ознакою призначається унікальне позначення відповідно до таблиці 2.3.

Таблиця 2.3. Призначення унікального позначення в залежності від оцінки життєвих показників.

Сильні оцінки	**Позначення**
${}_t^0\boldsymbol{S}_v^p$	I_v^0
${}_t^1\boldsymbol{S}_v^p$	I_v^1
${}_t^2\boldsymbol{S}_v^p$	I_v^2
${}_t^3\boldsymbol{S}_v^p$	I_v^3
${}_t^0\boldsymbol{S}_v^p$ та ${}_t^1\boldsymbol{S}_v^p$	I_v^4
${}_t^0\boldsymbol{S}_v^p$ та ${}_t^2\boldsymbol{S}_v^p$	I_v^5
${}_t^0\boldsymbol{S}_v^p$ та ${}_t^3\boldsymbol{S}_v^p$	I_v^6
${}_t^1\boldsymbol{S}_v^p$ та ${}_t^2\boldsymbol{S}_v^p$	I_v^7
${}_t^1\boldsymbol{S}_v^p$ та ${}_t^3\boldsymbol{S}_v^p$	I_v^8
${}_t^2\boldsymbol{S}_v^p$ та ${}_t^3\boldsymbol{S}_v^p$	I_v^9
${}_t^0\boldsymbol{S}_v^p$ та ${}_t^1\boldsymbol{S}_v^p$ та ${}_t^2\boldsymbol{S}_v^p$	I_v^{10}
${}_t^0\boldsymbol{S}_v^p$ та ${}_t^1\boldsymbol{S}_v^p$ та ${}_t^3\boldsymbol{S}_v^p$	I_v^{11}
${}_t^0\boldsymbol{S}_v^p$ та ${}_t^2\boldsymbol{S}_v^p$ та ${}_t^3\boldsymbol{S}_v^p$	I_v^{12}
${}_t^1\boldsymbol{S}_v^p$ та ${}_t^2\boldsymbol{S}_v^p$ та ${}_t^3\boldsymbol{S}_v^p$	I_v^{13}
${}_t^0\boldsymbol{S}_v^p$ та ${}_t^1\boldsymbol{S}_v^p$ та ${}_t^2\boldsymbol{S}_v^p$ та ${}_t^3\boldsymbol{S}_v^p$	I_v^{14}

Наприкінці цього кроку пацієнти з нотаціями балів у кожному кластері визначаються у вигляді такої матриці ($i \in [0,14]$) (див. Рис. 2.2).

$$\begin{matrix} patient\#i = \\ \vdots \\ patient\#N = \end{matrix} \begin{pmatrix} I_{HR}^i & I_{SBP}^i & I_{RR}^i & I_{OS}^i & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ I_{HR}^i & I_{SBP}^i & I_{RR}^i & I_{OS}^i & \cdots \end{pmatrix}$$

Рисунок 2.2. Матриця пацієнтів з нотаціями балів у кластері.

Де HR - частота пульсу, SBP - систолічний артеріальний тиск, RR – рівень респірації (частота дихання) , OS – рівень насичення крові киснем.

- Створити міцну асоціацію: На цьому етапі необхідно знайти чіткі правила кореляції між життєвими показниками всіх пацієнтів. За визначенням, правила повинні бути надані на основі частого оцінювання пацієнтів і повинні задовольняти, крім мінімальної сили балів μ, мінімальну впевненість, виражену як ρ. Припустимо, що множина позначень визначена як $I_v = \{I_v^0, I_v^1, \cdots, I_v^{14}\}$. Тоді мета - знайти список правил майнінгу, $M = \{M_1, M_2, \dots, M_\delta\}$, де кожен $M_i \in M$ має вигляд $L => N$; L і N можуть бути одним або перетином нотацій з I_v. Правило M_i має сильну асоціацію майнінгу, якщо його підтримка, $sup(M_i)$, більша за мінімальну підтримку μ (див формулу 2.2), а її впевненість, $conf(M_i)$, перевищує мінімальний поріг впевненості ρ наступним чином (див формулу 2.3).

$$sup(M_i) = \frac{|L \cap N|}{\theta} \times 100 \geq \mu \qquad (2.2)$$

де $|L \cap N|$ - це кількість пацієнтів у наведеній вище матриці, яка містить комбінації нотацій у L та N.

$$conf(M_i) = \frac{|N|}{|L|} \times 100 \geq \rho \qquad (2.3)$$

Алгоритм правил асоціативного майнінгу.

Вимоги: Набір позначень: $I_v = \{I_v^0, I_v^1, \cdots, I_v^{14}\}$, Мінімальна сила балів: μ, Мінімальний поріг впевненості: ρ.

Забезпечує: Набір сильних правил асоціативного майнінгу: M.

1: Знайти всі комбінації позначень, T, з I_v

2: $T \leftarrow \{ I_v^i, \text{де } i \in [0,14]\} \cup \{I_v^i \cap I_v^j, \text{ де } i,j \in [0,14] \text{ та } i \neq j\} \cup \dots \cup I_v$

3: Знайти всі комбінації правил майнінгу, U

4: $U = \{L => N, \text{де } L \text{ та } N \in T\}$

5: $M \leftarrow \emptyset$

6: **for** кожного правила $U_i \in U$ **do**

7: обчислити $sup(U_i)$ та $conf(U_i)$

8: **if** $sup(U_i) > \mu$ та $conf(U_i) > \rho$ **then**

9: $M \leftarrow M \cup \{U_i\}$

10: **end if**

11: **end for**

12: повернути M

Алгоритм описує процес пошуку набору сильних правил асоціації між I_v. Процес починається з пошуку всіх можливих комбінацій позначень із I_v, а потім пошуку списку всіх правил асоціації (рядки 1–4). Після цього ми обчислюємо підтримку та впевненість для кожного правила, і правило додається до списку сильних правил, лише якщо його підтримка та впевненість перевищують визначені пороги μ та ρ відповідно (рядки 5–11). Дійсно, отримані правила дозволять відповідальним особам знайти кореляцію між варіаціями показників життєдіяльності користувачів. Отже, на додаток до симптомів, що з'являються у відповідних користувачів, вони можуть краще зрозуміти хворобу та її діагноз, одночасно даючи потрібні пацієнтам ліки.

2.6. Висновки до розділу

1. Запропоновано засоби обробки інформації в медичних кіберфізичних системах. Результат виконання своїх функцій засобами обробки інформації в медичних КФС є оцінкою стану людини.
2. Запроповано модель обробки інформації в медичних КФС. Для розмежування засобів обробки інформації на окремі незалежні логічні елементи запропонована модель обробки інформації базується на основі мікросервісної архітектури, де кожен засіб є мікросервісом.

3. Запропоновано метод виявлення критичних подій в медичній кіберфізичній системі, який базується на одному з найбільш використовуваних посібників EWS та агрегованій оцінці життєвих показників стану людини.
4. Запропоновано алгоритм класифікації пацієнтів та алгоритм діагностики захворювань. Алгоритм класифікації пацієнтів - це адаптований алгоритми класифікації даних K-means. В свою чергу алгоритм діагностики захворювань - це адаптована версію алгоритму Association rule learning.

РОЗДІЛ 3

АРХІТЕКТУРНО-ІНФОРМАЦІЙНА МОДЕЛЬ ОЦІНЮВАННЯ СТАНУ ЛЮДИНИ В МЕДИЧНІЙ КІБЕРФІЗИЧНІЙ СИСТЕМІ

3.1. Архітектурно-інформаційна модель обробки інформації

Згідно запропонованим моделі та засобам обробки інформації, пропонується наступна структура медичної кіберфізичної системи. Вона має такий вигляд (див Рис. 3.1).

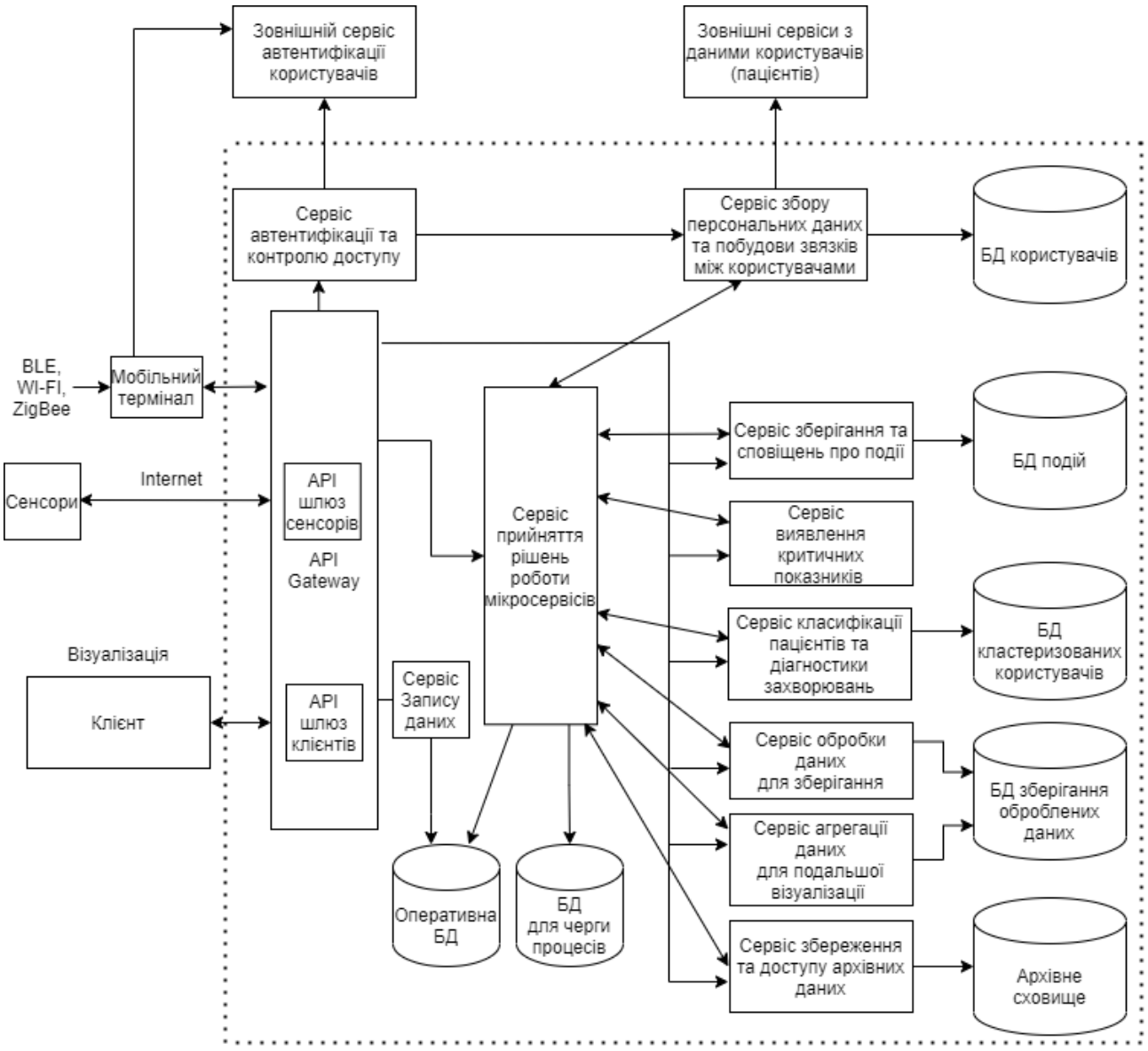

Рисунок 3.1 Запропонована структура медичної кіберфізичної системи.

З даної структури можна виділити архітектурно-інформаційну модель обробки інформації, яка складається з наступних елементів:

- Шаблон спілкування - API Gateway (шлюз);
- Сервіс запису даних та оперативної БД;
- Сервіс автентифікації та контролю доступу;
- Сервіс збору персональних даних та побудови зв’язків між користувачами та БД користувачів;
- Сервіс прийняття рішень роботи мікросервісів та БД для черги процесів;
- Сервіс зберігання та сповіщень про події та БД подій;
- Сервіс виявлення критичних показників;
- Сервіс класифікації пацієнтів та діагностики захворювань та БД кластеризованих юзерів;
- Сервіс обробки даних для зберігання та БД зберігання оброблених актуальних даних;
- Сервіс агрегації даних для подальшої візуалізації;
- Сервіс збереження та доступу архівних даних та архівного сховища.

Принцип роботи архітектурно-інформаційної моделі обробки інформації виглядає наступним чином. Життєві показники людини поступають на API шлюз. Після успішної автентифікації та авторизації цього запиту, дані записуються в оперативну базу даних через сервіс запису даних, а також інформується сервіс прийняття рішень роботи мікросервісів про надходження нових необроблених даних життєвих показників певної людини. Таким чином, джерело надходження даних отримує відповідь, що дані прийнято до обробки. Після цього, сервіс прийняття рішень роботи мікросервісів формує чергу процесів, які повинні виконати сервіси обробки інформації. Ця черга створюється в спеціальній базі даних для черги процесів.

Згідно вищезгаданої черги, мікросервіси виконують свою функцію обробки необроблених даних для оцінювання стану людини:

- виявлення критичних показників;
- класифікацію пацієнтів та діагностику захворювань;
- обробку даних для зберігання;
- збереження архівних даних.

Коли всі ці процеси виконаються, необроблені дані видаляться з оперативної бази даних.

Процес отримання даних для візуалізації виглядає наступним чином. Клієнт виконує запит на конкретну кінцеву точку API шлюзу, де запит та користувач проходять процес автентифікації та авторизації. За умов, що користувач автентифікований та має право на отримання запитуваної інформації, API шлюз доступається до потрібного мікросервісу, який відповідає за запитувані дані, через сервіс прийняття рішень роботи мікросервісів. Дані надходять у відповідь запиту клієнта, де відбувається подальша візуалізація.

3.2. Функціональні елементи архітектурно-інформаційної моделі обробки інформації

API Gateway (або шлюз). Для того, щоб пояснити необхідність використання API шлюзу, спочатку потрібно оглянути мікросервісний підхід без його використання. Отже, можливий підхід полягає у використанні архітектури прямого зв’язку клієнт-мікросервіс. При цьому підході клієнтська програма може надсилати запити безпосередньо до деяких мікросервісів, як показано на рисунку 3.2.

При такому підході кожен мікросервіс має загальнодоступну кінцеву точку, іноді з різним портом TCP для кожного мікросервісу.

Архітектура прямого спілкування від клієнта до мікросервісу може бути достатньо хорошою для невеликого додатку на основі мікросервісу, особливо якщо клієнтська програма є веб-програмою на стороні сервера. Однак, коли ви створюєте великі та складні системи, як медичні кіберфізичні системи, на базі

мікросервісів (наприклад, при обробці десятків типів мікросервісів), і особливо коли клієнтські програми - це віддалені мобільні програми або веб-програми SPA, такий підхід стикається з кількома проблемами.

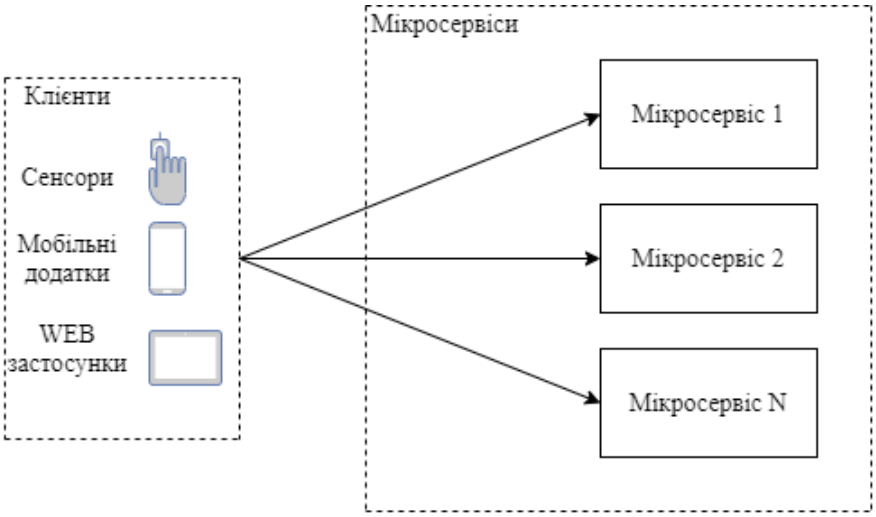

Рисунок 3.2. Мікросервісна архітектура з прямим зв’язком між клієнтом та мікросервісами

Розглянемо наступні питання при розробці великої системи на основі мікросервісів:

- Як клієнтські програми можуть мінімізувати кількість запитів на кінцеву точку і зменшити балакуче спілкування до декількох мікросервісів?

  Взаємодія з кількома мікросервісами для створення єдиного екрану інтерфейсу збільшує кількість зворотних запитів через Інтернет. Цей підхід збільшує затримку та складність на стороні інтерфейсу. В ідеалі, відповіді повинні бути ефективно агреговані на стороні сервера. Цей підхід зменшує затримку, оскільки кілька фрагментів даних повертаються паралельно, і деякий інтерфейс може відображати дані, як тільки вони будуть готові.

- Як ви можете вирішити наскрізні проблеми, такі як авторизація, перетворення даних та динамічне відправлення запитів?

Реалізація питань безпеки та наскрізних питань, таких як безпека та авторизація, на кожній мікрослужбі може вимагати значних зусиль з розробки. Можливим підходом є наявність цих служб у хості Docker або внутрішньому кластері, щоб обмежити прямий доступ до них ззовні, та реалізувати ці наскрізні проблеми в централізованому місці, як шлюз API.

- Як клієнтські програми можуть взаємодіяти зі службами, що використовують протоколи, що не відповідають Інтернету?

  Протоколи, що використовуються на стороні сервера (наприклад, AMQP або двійкові протоколи), не підтримуються в клієнтських програмах. Отже, запити повинні виконуватися за допомогою таких протоколів, як HTTP / HTTPS, а потім перекладатись на інші протоколи. У цій ситуації може допомогти підхід «людина посередині».

- Як можна сформувати фасад, спеціально розроблений для мобільних додатків?

  API декількох мікросервісів може бути погано розроблений для потреб різних клієнтських програм. Наприклад, потреби мобільного додатка можуть відрізнятися від потреб веб-додатку. Для мобільних додатків вам може знадобитися ще більше оптимізувати, щоб відповіді на дані могли бути ефективнішими. Ви можете зробити цю функцію, об’єднавши дані з декількох мікросервісів і повернувши один набір даних, а іноді і усунувши будь-які дані у відповіді, які не потрібні мобільному додатку. І, звичайно, ви можете стиснути ці дані. Знову ж таки, фасад або API між мобільним додатком та мікросервісами можуть бути зручними для цього сценарію.

Навіщо розглядати шлюзи API замість прямого зв'язку клієнт-мікросервіс. В архітектурі мікросервісів клієнтським програмам, як правило, потрібно використовувати функціональність декількох мікросервісів. Якщо це споживання виконується безпосередньо, клієнт повинен обробляти кілька викликів до кінцевих точок мікросервісу. Що відбувається, коли додаток

розвивається та впроваджуються нові мікросервіси або оновлюються існуючі мікросервіси? Якщо у вашому додатку багато мікросервісів, обробка такої кількості кінцевих точок з клієнтських програм може бути проблемою. Оскільки клієнтська програма буде поєднана з цими внутрішніми кінцевими точками, розвиток мікросервісів у майбутньому може спричинити великий вплив на клієнтські програми.

Отже, наявність середнього рівня або рівня опосередкованості (Шлюз) може бути зручним для програм, що базуються на мікросервісах. Якщо у вас немає шлюзів API, клієнтські програми повинні надсилати запити безпосередньо до мікросервісів, що створює такі проблеми, як:

Зв'язок: Без шаблону API шлюзу клієнтські програми пов'язані з внутрішніми мікросервісами. Клієнтські програми повинні знати, як різні мікропрограми розкладаються в мікросервісах. Під час еволюції та рефакторингу внутрішніх мікросервісів ці дії впливають на обслуговування, оскільки вони спричиняють порушення змін у клієнтських програмах через пряме посилання на внутрішні мікросервіси з клієнтських програм. Клієнтські програми потрібно часто оновлювати, ускладнюючи розвиток системи.

Забагато взаємних запитів: для однієї сторінки / екрана в клієнтському додатку може знадобитися кілька запитів до кількох сервісів. Цей підхід може призвести до кількох мережевих поїздок між клієнтом та сервером, додаючи значну затримку. Агрегація, що обробляється на API шлюзі, може покращити продуктивність та зручність роботи клієнтської програми.

Проблеми безпеки: Без шлюзу всі мікросервіси повинні бути піддані впливу «зовнішнього світу», що робить поверхню атаки більшою, ніж якщо ви приховуєте внутрішні мікросервіси, які безпосередньо не використовуються клієнтськими програмами. Чим менша поверхня атаки, тим безпечнішою може бути ваша програма.

Наскрізні проблеми: Кожна публічно опублікована мікросервісна служба повинна вирішувати такі проблеми, як авторизація та SSL. У багатьох ситуаціях ці проблеми можуть бути вирішені на одному рівні, щоб спростити внутрішні мікросервіси.

Коли ви проектуєте та створюєте великі або складні додатки на основі мікросервісу з кількома клієнтськими програмами, хорошим підходом може бути API-шлюз. Цей шаблон є послугою, яка забезпечує єдину точку входу для певних груп мікросервісів. Це схоже на візерунок "Фасад" з об'єктно-орієнтованого дизайну, але в цьому випадку це частина розподіленої системи. Шаблон шлюзу API також іноді називають "серверним інтерфейсом" (BFF), оскільки ви створюєте його, думаючи про потреби клієнтської програми.

Тому шлюз API знаходиться між клієнтськими програмами та мікросервісами. Він діє як зворотний проксі-сервер, що спрямовує запити від клієнтів до служб. Він також може надавати інші наскрізні функції, такі як аутентифікація, кеш.

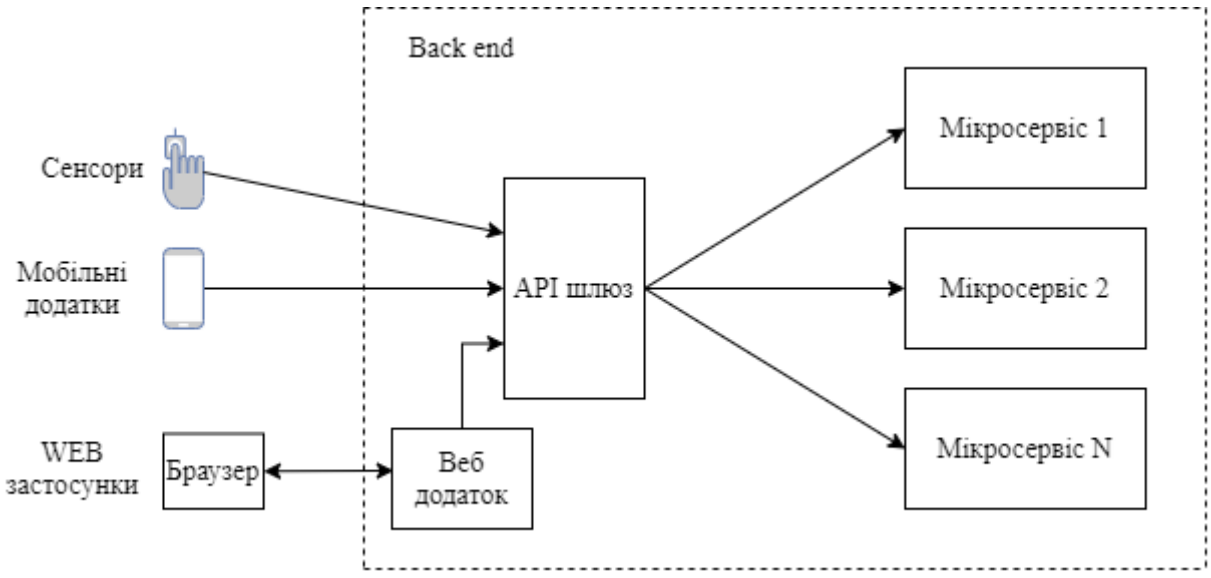

Рисунок 3.3. Мікросервісна архітектура з зв’язком між клієнтом та мікросервісами через API шлюз.

На рисунку 3.3 показано, як користувальницький шлюз API може вписатись у спрощену архітектуру, засновану на мікросервісах, лише за допомогою декількох мікросервісів.

Програми підключаються до однієї кінцевої точки, шлюзу API, яка налаштована на пересилання запитів до окремих мікросервісів.

Важливо підкреслити, що на цій діаграмі ви використовуєте одну спеціальну службу шлюзу API, яка стоїть перед різними клієнтськими програмами. Цей факт може бути важливим ризиком, оскільки ваш сервіс шлюзу API буде рости та розвиватися на основі багатьох різних вимог, що пред'являються до клієнтських програм. Врешті-решт, він буде роздутий через ці різні потреби, і фактично він може бути схожий на монолітну програму чи монолітний сервіс. Ось чому дуже рекомендується розділити шлюз API на кілька служб або кілька менших шлюзів API, наприклад, по одному на тип форм-фактора клієнтської програми.

Потрібно бути обережним при реалізації шаблону API шлюзу. Зазвичай не є гарною ідеєю мати єдиний шлюз API, який об'єднує всі внутрішні мікросервіси вашої програми. Якщо це так, він діє як монолітний агрегатор або оркестратор і порушує автономію мікросервісу, поєднуючи всі мікросервіси.

Тому шлюзи API повинні бути розділені на основі ділових кордонів та клієнтських програм, а не діяти як єдиний агрегатор для всіх внутрішніх мікросервісів.

При розподілі рівня шлюзу API на кілька шлюзів API, якщо у системі є кілька клієнтських програм, це може бути основним стрижнем при ідентифікації декількох типів шлюзів API, так що ви можете мати різний фасад для потреб кожного клієнтського додатка. Цей випадок є шаблоном під назвою "Backend for Frontend" (BFF), де кожен шлюз API може надавати різний API, розроблений для кожного типу клієнтської програми, можливо, навіть на основі клієнтського

форм-фактора шляхом реалізації конкретного коду адаптера, який внизу викликає кілька внутрішніх мікросервісів, як показано на рисунку 3.4.

На рисунку 3.4 показані шлюзи API, які розділені за типом клієнта; один для сенсорів, один для мобільних клієнтів та один для веб-клієнтів. Приклад зображує спрощену архітектуру з безліччю дрібнозернистих шлюзів API. У цьому випадку межі, визначені для кожного шлюзу API, базуються виключно на шаблоні BFF, отже, лише на основі API, необхідного для клієнтської програми.

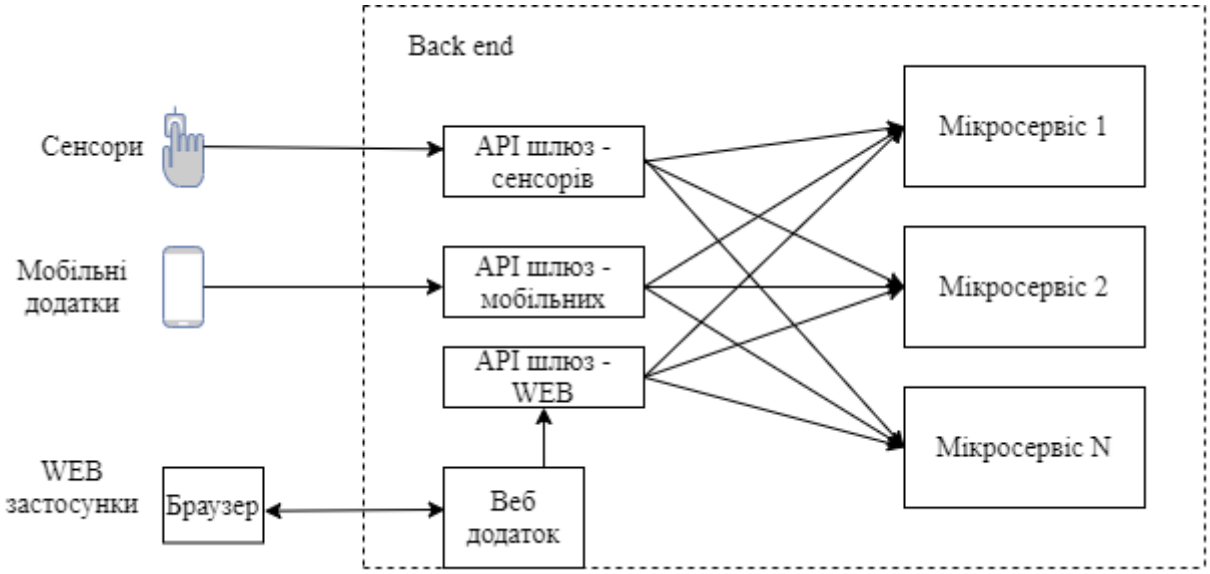

Рисунок 3.4. Мікросервісна архітектура з зв’язком між клієнтом та мікросервісами через кілька API шлюзів.

Шлюз API може запропонувати кілька функцій. В даній моделі використані такі основні функції шлюзу API:

- Зворотній маршрутизатор проксі або шлюзу. Шлюз API пропонує зворотний проксі-сервер для перенаправлення або маршрутизації запитів до кінцевих точок внутрішніх мікросервісів. Шлюз забезпечує єдину кінцеву точку або URL-адресу для клієнтських програм, а потім внутрішньо відображає запити до групи внутрішніх мікросервісів. Ця функція маршрутизації допомагає відокремити клієнтські програми від

мікросервісів, але це також зручно при модернізації монолітного API, розміщуючи шлюз API між монолітним API та клієнтськими програмами, тоді ви можете додавати нові API як нові мікросервіси, використовуючи при цьому застарілу версію монолітний API, доки він не буде розділений на багато мікросервісів у майбутньому. Через шлюз API клієнтські програми не помітять, чи використовуються API як внутрішні мікросервіси або монолітний API, і що більш важливо, при розробці та рефакторингу монолітного API в мікросервіси завдяки маршрутизації шлюзу API, клієнтські програми не зазнають будь-яких змін URI.

- Агрегація запитів. Як частина шаблону шлюзу ви можете об'єднати кілька клієнтських запитів, націлених на кілька внутрішніх мікросервісів, в один клієнтський запит. Цей шаблон особливо зручний, коли клієнтській сторінці / екрану потрібна інформація від декількох мікросервісів. При такому підході клієнтська програма надсилає один запит шлюзу API, який відправляє кілька запитів до внутрішніх мікросервісів, а потім агрегує результати та відправляє все назад до клієнтської програми. Основною перевагою та метою цього шаблону дизайну є зменшення кількості взаємних запитів між клієнтськими програмами та серверним API, що особливо важливо для віддалених додатків поза центром обробки даних, де живуть мікросервіси, як-от мобільні програми. Для звичайних веб-додатків, що виконують запити в серверному середовищі, цей шаблон не настільки важливий, оскільки затримка значно менша, ніж для віддалених клієнтських програм.
- Наскрізні проблеми або розвантаження шлюзу. Залежно від функцій, пропонованих кожним продуктом API Gateway, ви можете завантажити функціональність від окремих мікросервісів до шлюзу, що спрощує реалізацію кожної мікросервіси, консолідуючи наскрізні проблеми в один рівень. Цей підхід особливо зручний для спеціалізованих функцій, які можуть бути складними для належного впровадження у кожній внутрішній мікросервісі, наприклад, для таких функцій: автентифікація та авторизація;

інтеграція служби виявлення; кешування відповідей; обмеження та регулювання швидкості; балансування навантаження; протоколювання, трасування, кореляція; заголовки, рядки запитів та перетворення претензій; список дозволених IP.

Сервіс запису даних та оперативна БД. Даний сервіс виконує запис отриманих необроблених даних без попередної обробки в оперативну базу даних. Метою створення цього сервісу є знизити час очікування клієнта, яка прислав дані життєвих показників людини. Оперативна база даних в свою чергу служить централізованим сховищем необроблених даних, яке використовується мікросервісами обробки інформації асинхронно до запиту на запис цих даних. Дані в цій базі зберігаються тимчасово, що дозволяє уникнути проблем з перенавантаженням бази даних, та зберігає її швидкодію.

Сервіс автентифікації та контролю доступу. У цьому сервісі відбуваються процеси автентифікації та авторизації запитів та користувачів.

При реалізації автентифікації та авторизації в мікросервісах процес стає набагато складнішим, ніж у традиційній монолітній архітектурі. Автентифікація та авторизація програм у архітектурі мікросервісів зазвичай реалізуються у централізованій службі, яка відповідає за це. Реалізація перевірок автентифікації повинна припинятися всередині шлюзу API. Реалізація авторизації може бути здійснена або в шлюзі API, або в мікросервісах. Щоб мати можливість проводити масштабні перевірки авторизації для конкретних програм, авторизацію слід обробляти у певних мікросервісах.

Одним із способів вирішення проблеми авторизації системи є реалізація цього в шлюзі API. Таким чином стає можливим обмеження запитів на певні кінцеві точки. Недоліком реалізації авторизації в шлюзі API є те, що це може бути лише доступ на основі ролей до кінцевих точок. Реалізація додаткових перевірок доступу до певних об’єктів домену потребує створення певної логіки домену всередині шлюзу API, а отже, призведе до витоку логіки домену. Крім того, при

впровадженні декількох бекендів для інтерфейсів/шлюзів API керування авторизацією стає все важчим і складнішим.

Кращим рішенням було б покласти на мікросервіси відповідальність за обробку авторизації. Таким чином, певну програму потрібно реалізувати лише в одному місці. Недоліком цього є те, що авторизація буде більш розповсюджена в різних службах. Якщо у вас багато ролей, які дуже часто змінюються, керувати цим стає більш важче.

Це означає, що при надходженні даних життєвих показників людини з сенсорів, шлюз API комунікує з сервісом автентифікації та контролю доступу, де запит автентифіковується, і ці дані будуть належати певному користувачу. Також буде перевірено, чи автентифікований користувач має право на запис цих даних в систему. У випадку запиту на зчитування даних, так само автентифікований користувач буде перевірений на рівень доступу до запитуваних даних. Отже, даний сервіс забезпечує дотримання конфіденційності та приватності персональних та технічних даних користувачів, а також захищеності даних від несанкціонованого доступу.

Сервіс збору персональних даних та побудови зв’язків між користувачами та БД користувачів. Основною функцією даного мікросервісу є обробка та збереження персональних даних користувачів. В першу чергу, він відповідає за реєстрацію нових користувачів в системі, створюючи запис в базі даних користувачів, включаючи обов’язкові поля персональних даних людини, необхідні для подальшої роботи в системі та взаємодії з іншими користувачами.

Не менш важливою функцією цього сервісу є побудова зв’язків між користувачами, яка є основою в подальшому контролі доступу до даних іншого користувача згідно бізнес логіці системи.

Додатковою функцією є взаємодія з зовнішніми ресурсами для отримання додаткових персональних та технічних даних. Оскільки, окрім даних про життєві показники стану людини, є також інші дані, які могли б бути корисними для

відповідальної особи в процесі оцінювання стану людини в медичній кіберфізичній системі, отримання таких даних з сторонніх ресурсів можуть полегшити процес оцінювання стану людини та внести додаткову інформацію про користувача, яка в даній медичній кіберфізичній системі не вимірюється та не обробляється.

Отже, даний сервіс є централізованим сховищем персональних даних користувачів та зв’язків між ними, для подальшого контролю доступу запису/читання даних згідно бізнес логіки.

Сервіс прийняття рішень роботи мікросервісів та БД для черги процесів. Для організації процесів обробки даних доцільно створити окремий мікросервіс, який б міг контролювати процес та пріоритетність виконання функцій інших мікросервісів, які відповідають за засоби обробки інформації.

При отриманні необроблених даних, сервіс прийняття рішень роботи мікросервісів інформує інші сервіси про потребу обробки цих даних, згідно пріоритетної черги створеної в спеціальній базі даних. Він також контролює виконання цих процесів, і відповідальний за видалення необроблених даних після їх повної обробки.

Також цей мікросервіс є певного типу оркестратором, який дозволяє іншим мікросервісам спілкуватись між собою. Наприклад, якщо при виконанні процесу певної обробки даних була визначена та створена певна подія, то через нього інформація про цю подію буде передана в відповідальний за події сервіс.

Якщо говорити про запит на читання даних, то цьому сервісі відбувається агрегація даних для відповіді на запит. Це означає, що в залежності від запиту, сервіс прийняття рішень роботи мікросервісів спілкується з необхідними сервісами, відповідальними за той чи інший тип запитуваних даних та агрегує їх в єдиний набір даних.

Варто відмітити, що даний мікросервіс являється центром роботи архітектури медичної кіберфізичної системи. Це є проблемою, оскільки відмова від роботи

даного мікросервісу ставить під загрозу роботу всієї системи. Для вирішення цієї проблеми, в системі також залишається можливість мікросервісів спілкуватись між собою, в разі недоступності сервісу прийняття рішень роботи мікросервісів. Такий ж підхід і з агрегацією запитуваних даних. При відмові даного мікросервісу, це завдання бере на себе API шлюз, який також має можливість спілкуватись з кожним мікросервісом напряму.

Сервіс зберігання та сповіщень про події та БД подій. Як повідомлялось раніше, події слід розділяти на автоматичні та події, створені на вимогу. А також, на критичні та інформаційні. Якщо реакція на критичні події повинна бути негайною, то з інформаційними подіями ситуація дещо інша. Реагування на інформаційні події не несе негайного характеру, тому повинна бути можливість дізнатись про цю подію з певним проміжком часу після її створення. Постає питання в збереженні подій та даних про них для подальшого їх отримання та візуалізації на клієнті. Саме тому сервіс зберігання та сповіщень про події має власну базу даних, де зберігаються всі події та супутня їх інформація. Сповіщення про інформаційні події відбувається на стороні клієнта за допомогою графічного інтерфейсу.

Якщо інформаційні події достатньо зберегти у базі, і очікувати допоки відповідальна особа зробити запит на отримання такого типу подій, то принцип сповіщення про критичні події зовсім інший. Як зазначалось вище, реакція на критичні події повинна бути негайна, оскільки вона може мати прямий вплив на стан та здоров’я людини. Для забезпечення негайної реакції на такого типу події постає питання в створенні засобу сповіщень, що також являється функцією цього мікросервісу. Це означає, що за допомогою розроблених механізмів надсилання сповіщень, при створенні такого типу події сервіс відповідає за негайне інформування відповідальної особи, чи групи осіб.

Якщо говорити про події, створені на вимогу, вони можуть мати також різний рівень серйозності, згідно бізнес логіці розробленої медичної кіберфізичної системи. Тим не менш, створення такого типу події працює за тим же

принципом, що і автоматичних. Запис про їх створення та супутню інформацію вноситься с базу даних подій, а сповіщення про критичні події відбувається в негайному порядку.

При реалізації системи сповіщень про критичні події повинні бути виконані такі цілі:

- Можливість інформувати відповідальну особу.
- Можливість інформувати групу осіб.
- Можливість інформувати використовуючи різні канали зв’язку.

Отже, функцією даного сервісу є зберігання та інформування про події, пов’язані з функціональним станом організму людини.

Сервіс виявлення критичних показників є одним з найважливіших сервісів в плані оцінювання стану людини в медичній кіберфізичній системі. Виявлення критичних показників та своєчасне реагування на цю подію несе дуже відповідальний характер, оскільки це може напряму впливати на здоров’я та навіть життя людини.

Процес виконання функції цього сервісу має найвищий пріоритет, отже виявлення критичних життєвих показників стану людини відбувається в першу чергу. Згідно вище описаного методу та елементів бізнес логіки системи, сервіс виявлення критичних показників обробляє отриману з сенсорів інформацію. У разі виявлення критичних показників, створюється критична подія, про яку негайно інформується відповідальна особа за допомогою сервісу зберігання та сповіщень про події.

Сервіс класифікації пацієнтів та діагностики захворювань та БД кластеризованих користувачів. Ще одним важливим етапом оцінювання стану людини в медичній кіберфізичній системі є процес аналізу даних, для зрозуміння захворювання та спробувати мінімізувати їх майбутні наслідки. Першим кроком аналізу даних є класифікація пацієнтів у кластери, де пацієнти в одному кластері мають загальні характеристики (симптоми та ситуації). Другий крок - вивчення

кореляції між життєвими ознаками пацієнтів в одному кластері. Це може допомогти відповідальній особі зрозуміти причини захворювання та поведінку, а отже, уникнути помилкової діагностики захворювання та знайти відповідне лікування. Метою цього мікросервісу є виконання двох запропонованих алгоритмів аналізу даних, один для класифікації пацієнтів, а інший для діагностики захворювання.

Потрібно розуміти, що дані, які аналізуються є великими даними. Великі дані - це термін, який позначає дані, які виходять за межі обсягу зберігання та можливостей обробки класичного комп'ютера, і отримати певну інформацію від великої кількості даних - дуже велика проблема. Саме тому, квантові обчислення приходять на допомогу, пропонуючи багато обіцянок у системах обробки інформації, особливо в аналітиці великих даних.

Відомо, що квантові комп'ютери виконують певні складні алгоритми експоненціально швидше, ніж класичні комп'ютери. Квантова версія алгоритму K-means забезпечує експоненціальне прискорення для дуже високих розмірних вхідних векторів. Прискорення відбувається через те, що для завантаження N-мірних вхідних векторів за допомогою амплітудного кодування потрібні лише $\log N$ кубітів. Одна з версій квантового алгоритму K-means описана в [57]. Класифікації пацієнтів та діагностика захворювань відбувається за схожим принципом, використовуючи адаптований алгоритм K-means.

Сервіс обробки даних для зберігання та БД зберігання оброблених актуальних даних. Правильне і зручне представлення даних на клієнті є запорукою швидкого та ефективного оцінювання стану людини відповідальною особою. Процеси додаткової обробки отриманих даних на клієнті викликатиме зайві затрати часу для візуалізації життєвих показників та стану організму людини в цілому. Про обробці необроблених даних на сервері в момент запиту клієнта, це так само викликатиме затримку. Тому було прийнято рішення виконувати обробку необроблених даних відразу після отримання, не очікуючи

запиту клієнта на читання цих даних, що зменшить в майбутньому час отримання і візуалізації запитуваних даних.

Цей процес відбувається паралельно з іншими процесами обробки необроблених даних. В сервісі обробки даних для зберігання, необроблені дані обробляються і формуються в певну структуру (визначену бізнес логікою системи), яка буде сумісною з різного типу клієнтами. Структуровані та оброблені дані зберігаються в базі даних оброблених актуальних даних. В цій БД зберігаються лише актуальні дані (за певний невеликий проміжок часу, визначений бізнес логікою системи), які готові до відправки клієнту для подальшої візуалізації, або в інші медичні інформаційні системи. Згідно бізнес логіці системи, в певний період часу дані стають неактуальними, що є потребою їх видалення з бази зберігання оброблених актуальних даних.

Сервіс агрегації даних для подальшої візуалізації. Для розпаралелення процесів запису і читання актуальних даних про життєві показники стану людини, прийнято рішення розділити ці процеси на окремі мікросервіси. При запиті клієнта на отримання даних про життєві показники організму людини, сервіс агрегації даних для подальшої візуалізації доступається до бази оброблених актуальних даних та витягує необхідні дані.

Запитувані дані агрегуються згідно запиту клієнта та бізнес логіці системи. Наприклад, сервіс може повертати біжучі актуальні дані або агреговані дані за певний період актуального часу (визначеного бізнес логікою системи). Також сервіс може агрегувати дані всіх життєвих показників стану людини, декількох, або тільки одного з них в залежності від запиту клієнта.

Агреговані сервісом дані готові до візуалізації на клієнті і не потребують додаткової обробки.

Сервіс збереження та доступу архівних даних та архівного сховища. Як випливає з сервісу обробки даних для зберігання, я пропоную розділити дані на актуальні та архівні. Відмінністю між цими даними є їх актуальність в процесі

моніторингу та оцінювання стану людини, а також частота запитів на отримання цих даних.

Зрозуміло, що актуальні дані будуть часто запитуваними, тому цілком логічно їх відділити в окрему базу даних (БД зберігання оброблених актуальних даних). В свою чергу архівні дані повинні мати своє власне сховище. На відміну від актуальних даних, архівні дані являються великими даними, що потребує відповідного рішення зберігання таких даних. Але, варто врахувати, що архівні дані не є часто запитуваними та не несуть негайного характеру, тому швидкодія обробки запиту на отримання архівних даних про життєві показники людини не має різких обмежень в часі обробки запиту.

Архівні дані обробляються і зберігаються паралельно та за іншими принципами, ніж актуальні дані. Тому є очевидним створення окремого мікросервісу для таких потреб.

Сервіс збереження та доступу архівних даних виконує функції обробки та зберігання обробленої інформації в архівному сховищі. Також, він відповідальний за агрегацію цих даних відповідно до запиту клієнта на отримання цих даних. Агрегація даних також відбувається згідно бізнес логіці системи.

3.3. Бізнес логіка в роботі архітектурно-інформаційної моделі обробки інформації медичної кіберфізичної системи.

Бізнес логіка або доменна логіка — система зв'язків та залежностей елементів бізнес-даних та правил обробки цих даних відповідно до особливостей ведення окремої діяльності (бізнес правил), яка встановлюється при розробці програмного забезпечення, призначеного для автоматизації цієї діяльності. Бізнес логіка описує бізнес-правила реального світу, які визначають способи створення, представлення та зміни даних. Бізнес логіка контрастує з іншими частинами програми, які мають відношення до низького рівня: управління базою даних, відображення інтерфейсу користувача, інфраструктура і т.д. Бізнес логіка

визначається робочими процесами, відображається у впорядковані у часі задачі, структури інформаційних моделей та потоків даних від одного учасника робочих процесів (людини або програмного забезпечення) до іншого. У відносно простих системах бізнес логіка визначається алгоритмами обробки даних відповідно до вимог предметної області, у більш складних системах бізнес логіка переноситься на рівень даних (опис бізнес логіки), які у той чи інший спосіб визначають правила обробки даних [72].

Тому, бізнес логіка є важливим інструментом в процесі оцінювання стану організму людини та роботи медичної кіберфізичної системи в цілому.

3.3.1. Зацікавлені сторони та логіка побудови зв'язків між користувачами

Важливим елементом бізнес логіки системи є зацікавлені сторони. На відміну від існуючих систем, описаних в першому розділі роботи, запропонована модель медичної кіберфізичної системи не має строгих обмежень в цьому питанні. За рахунок гнучкої бізнес логіки та реалізації системи зв'язків між користувачами, зацікавленими сторонами можуть бути різні особи. Наприклад, відповідальної особою, чи навіть групою осіб, за моніторинг та оцінювання стану людини, можуть бути як лікарі, так і родичі чи близькі люди користувача, за яким відбувається моніторинг життєвих показників стану організму. Особа, за якою ведуть спостереження може бути пацієнтом в лікарні, на амбулаторному лікуванні, чи просто близькою людиною відповідальної особи. Більш того, користувач може вести моніторинг своїх власних життєвих показників стану організму.

Як випливає з опису зацікавлених сторін, сфера застосування медичної кіберфізичної системи теж не є різко обмежена. Таку систему оцінювання стану людини можна використовувати в медичних закладах, закладах догляду за літніми людьми, залах для тренувань, на змаганнях, а також і для персонального домашнього використання.

Принцип побудови зв’язків між користувачами та їх авторизації теж пов'язаний з бізнес логікою системи. А саме: користувач може мати різні ролі. Тобто, користувач може бути людиною, над якою ведуть спостереження, і в той же час наглядати за кимось іншим, або за самим собою. Кількість зв’язків між користувачами є необмежена, організовані за принципом багато до багатьох, як показано на рисунку 3.5.

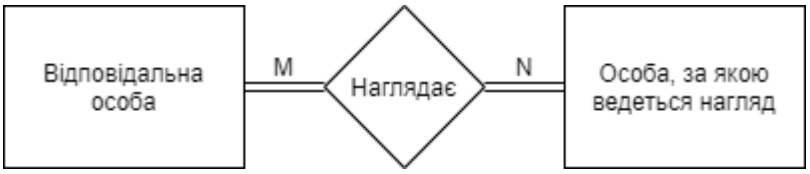

Рисунок 3.5. Організація зв’язків між відповідальною особою та особою, що під наглядом.

Це означає що один користувач може наглядати за декількома людьми одночасно. Аналогічно, одна людина може бути під наглядом не одної, а цілої групи осіб. Зі сторони користувача, який веде моніторинг життєвих показників стану організму людини, він може створити групу осіб, або декілька різних груп, за власними критеріями. Зі сторони користувача, за яким ведеться нагляд, він може спостерігатись необмеженою кількістю осіб (див Рис. 3.6).

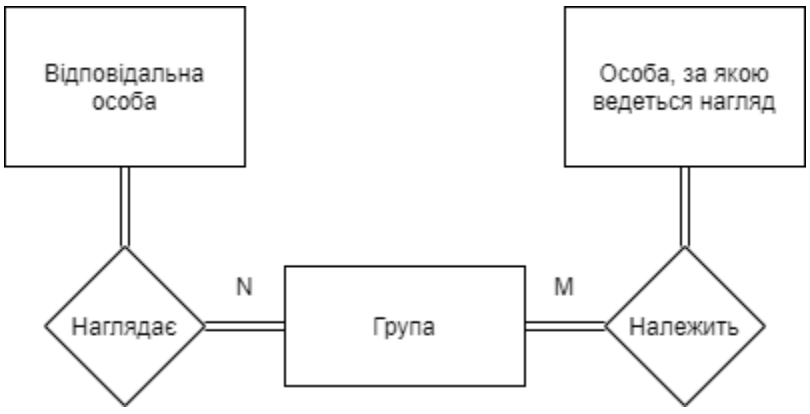

Рисунок 3.6. Організація зв’язків між відповідальною особою та особою, що під наглядом за допомогою сутності штучного типу.

Варто зазначити, що моніторинг і побудова зв’язків між користувачами (відповідальною особою та особою, за якою ведеться спостереження) відбувається за рахунок взаємного погодження

Відповідальна особа надсилає запрошення користувачеві, за яким повинен вестись нагляд. Запрошення приймається, і відправляється запит у відповідь на підтвердження. Запит-відповідь підтверджується відповідальною особою. Зв'язок між користувачами побудовано.

Процес побудови зв’язків між користувачами згідно бізнес логіки системи виглядає наступним чином (див Рис. 3.7).

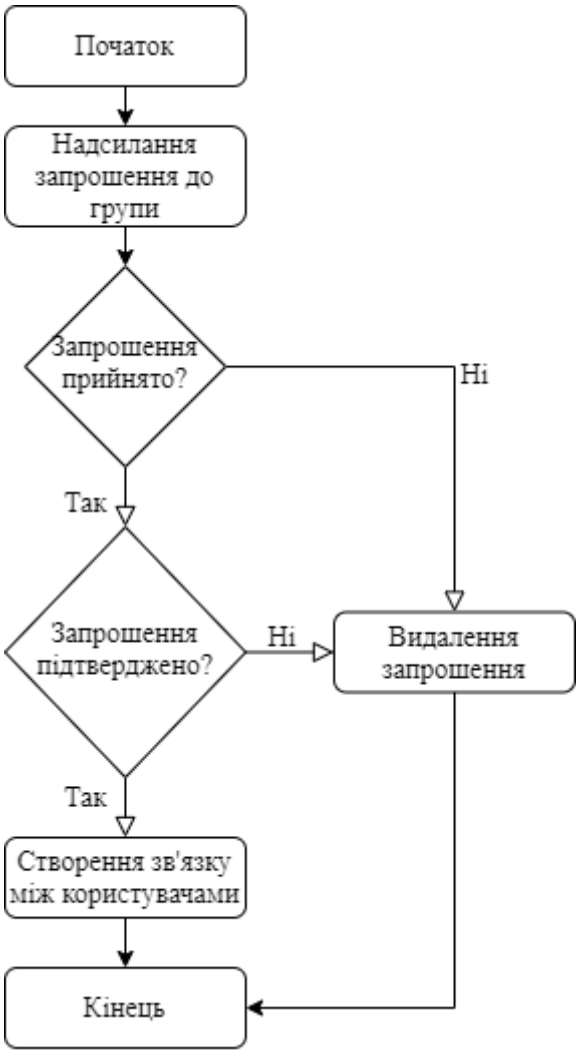

Рисунок 3.7. Алгоритм побудови зв’язків між користувачами.

Після цих дій, відповідальна особа має право на запит персональних та технічних даних з системи про особу, за якою ведеться моніторинг, і надала дозвіл на обробку своїх даних. Без цього дозволу, доступ до даних користувачів відсутній.

Отже, на основі таких зв’язків між користувачами забезпечується контроль доступу, а також конфіденційність та приватність даних користувачів. Якщо говорити про право на запис даних в систему, то воно обмежене лише автентифікацією користувача.

3.3.2. Бізнес логіка зберігання необроблених даних та їх обробки.

Бізнес логіка зберігання необроблених даних вже неодноразово зачіпалась в даній роботі. Запропонований підхід полягає в зберіганні необроблених даних тимчасово, допоки сенс зберігання таких даних зникне. Сенс зберігання таких даних полягає в тому, що вони є джерелом даних для подальшої їх обробки різними мікросервісами. Тому логічно випливає, що ці дані стануть неактуальними в той момент часу, коли всі процеси обробки цих даних успішно виконаються. Завдяки цьому підходу, необроблені дані доступні для паралельної обробки різними мікросервісами, а регулярне очищення бази від неактуальних даних дозволяє не переповнювати її.

Процес обробки необроблених даних визначається сервісом прийняття рішень роботи мікросервісів, який створює список задач, які повинні виконатись для оцінювання стану людини. З точки зору бізнес логіки, ці задачі виконуються паралельно і незалежні одна від одної. А також, при завершенні виконання останньої задачі з цього списку, оперативна база даних видаляє необроблені дані.

3.3.3. Бізнес логіка сервісів обробки інформації.

Сервіс зберігання та сповіщень про події. Як згадувалось раніше, події можна розділити за двома принципами. А саме, за природою їх створення: автоматичні та на вимогу, за рівнем серйозності: критичні та інформаційні.

Згідно бізнес логіки, автоматичні події виявляються всередині медичної кіберфізичної системи внаслідок певної обробки інформації та результатів цієї обробки. В результаті цієї обробки, мікросервіс, який виконував ту чи іншу функцію обробки інформації спілкується з сервісом зберігання та сповіщень про події, який в свою чергу записує дані про подію в базу даних та приймає подальше рішення на рахунок сповіщення про цю подію.

Виявлення події, створеної на вимогу, відбувається на іншому рівні медичної кіберфізичної системи, а саме на клієнті або на сенсорах. Це означає, що є створений механізм для передачі інформації про виявлену подію з клієнта чи сенсора до мікросервісу, який так само записує дані про подію в базу даних та приймає подальше рішення на рахунок сповіщення про цю подію.

Отже, незалежно від природи створення, всі події зберігаються в базі даних і доступні для читання.

По тому ж принципу, незалежно від рівня серйозності події, всі події зберігаються в базі даних. Проте, незалежно від природи створення, але в залежності від рівня серйозності події, приймаються різні рішення щодо сповіщення про подію. А саме, якщо подія критичного рівня, то сповіщенню про цю подію є негайним. Для цього реалізовуються спеціальні способи сповіщень про подію відповідальних осіб. При умові, що подія інформаційного рівня, негайне сповіщення про неї не має особливого змісту, оскільки не є критичним для стану людини. Тому, згідно бізнес логіці системи, ознайомлення з такими подіями відбувається на клієнті, який запитує в мікросервісу дані про всі події, пов’язані з людиною, за якою спостерігає відповідальна особа.

Запис події в базі даних має таку структуру:

- назва події;
- інформація про подію;
- повідомлення з рекомендованими діями;
- час виявлення події; підтвердження, що подію переглянуто;

- час перегляду події відповідальною особою.

В запропонованій медичній кіберфізичній системі можна виділити такі події:

- Подія, створена на вимогу користувачем, щоб відповідальна особа звернула увагу на його показники та зв’язалась з ним. Ця подія є критичною.
- Подія, створена автоматично сенсором, про низький заряд акумулятора. Ця подія є критичною.
- Подія, створена автоматично, про виявлення критичних життєвих показників людини. Ця подія є критичною.
- Подія, створена автоматично, про приведення життєвих показників в норму. Ця подія є інформаційною.
- Подія, створена автоматично, про класифікацію користувача та діагностику ймовірного захворювання. Ця подія є інформаційною.
- Подія, створена автоматично, про прогнозування імовірних показників. Ця подія є інформаційною.
- Подія, створена автоматично, про перегляд відповідальною особою показників користувача. Ця подія є інформаційною.

Сервіс виявлення критичних показників. Згідно бізнес логіці системи, виявлення критичних показників відбувається за вище запропонованим методом. Проте, виявлення критичності показників може бути як загальноприйнятим, так і індивідуальним.

Загальноприйнятим виявленням критичних показників полягає в тому, що для тих чи інших життєвих показників використовуються загальноприйняті норми, по-іншому – межі. Тобто, якщо показники в певній мірі перетинає ці межі, вони вважаються критичними. Але, в даній кіберфізичній системі я пропоную дозволити відповідальній особі встановлювати індивідуальні межі для кожного користувача, над яким ведеться нагляд. Це означає, що за замовчуванням межі критичності показників є однаковими для всіх. Але, враховуючи різні фактори,

такі вік, стать, хронічні хвороби та інші фактори, які можуть впливати на життєві показники людини, ці межі повинні бути індивідуальними для кожної людини. Так, в більшості випадків вони можуть співпадати з загальноприйнятими межами, але сам факт можливості встановлення індивідуальних меж робить систему більш гнучкою, і дозволяє відповідальний особам вести більш контрольовані моніторинг та оцінювання стану людини.

Отже, виявлення критичних показників працює згідно запропонованого методу, в загальноприйнятих чи індивідуальних межах життєвих показників. Варто зазначити, що критичність показників може бути різною, і описана в запропонованому методі. Проте, згідно бізнес логіці запропонованої медичної кіберфізичної системи, при виявленні різного рівня критичних показників, створюється відповідна критична подія, про яку надсилається сповіщення відповідальній особі, чи групі осіб. Це сповіщення містить не тільки інформацію, про походження події (виявлення критичних показників), але і про рівень критичності цих показників та можливі рекомендації прийняття рішень відповідальною особою. Ці рекомендації можуть бути різними, в залежності від сфери застосування медичної кіберфізичної системи: в лікарні, чи в тренажерному залі, тощо.

Варто зазначити, що виявлення критичних показників є одним з основних елементів оцінювання стану людини, тому реалізації виявлення та сповіщення про ці події є дуже важливими в медичній кіберфізичній системі.

Сервіс класифікації пацієнтів та діагностики захворювань та БД кластеризованих користувачів. Класифікації пацієнтів (користувачів) відбувається згідно запропонованого вище методу, а саме на основі оцінки життєвих показників. Згідно бізнес логіки, цей сервіс допомагає відповідальній особі знайти схожих користувачів за показниками, симптомами і т.д., що дозволить спростити процес оцінювання стану людини.

Варто зазначити, що класифікація і подальша діагностика відбувається з усіма користувачами, які існують в базі даних. Цілком ймовірно, що деякі з таких користувачів не надавали доступ до своїх персональних даних конкретній відповідальній особі. Тому, згідно бізнес логіці, відповідальна особа може отримати лише технічні дані таких користувачів. Оскільки ці дані анонімізовані, питання конфіденційності в даному випадку не порушуються, а технічні дані при відсутності персональних, не дають можливості ідентифікувати особу.

Сервіс обробки даних для зберігання та БД зберігання оброблених актуальних даних. Бізнес логіка обробки даних для зберігання та подальшої візуалізації визначає принцип структуризації даних життєвих показників. Дані повинні бути пов’язані з зареєстрованим користувачем в системі для подальшого контролю доступу до них. Принцип обробки інформації полягає в приведенні необроблених даних до певної структури та зберігання цих даних в відповідних таблицях бази даних.

Згідно бізнес логіці, кожен тип життєвих показників буде записуватись в окрему таблицю бази даних. Запис в такій таблиці повинен мати, як мінімум, інформацію про значення показника, час вимірювання, час запису в систему, ідентифікатор користувача. Всі ці дані повинні бути отримані внаслідок обробки необроблених даних.

Як зазначалось раніше, цей сервіс зберігає лише актуальні дані. Згідно бізнес логіці, актуальними дані в запропонованій медичній кіберфізичній системі є ті дані про життєві показники, час вимірювання яких є не пізнішим ніж задана кількість днів тому. Саме дані за останній час мають найбільшу інформативність для відповідальної особи в плані оцінювання стану людини. Всі старіші дані, згідно бізнес логіці, є архівними, і зберігаються та видобуваються за іншим принципом з іншого сховища даних.

Для зберігання лише актуальних даних в цій базі даних, бізнес логіка повинна визначати принцип видалення неактуальних даних. Це питання

вирішується за допомогою служби планування задач, яка виконує заплановану задачу автоматично в заданий час. Отже, кожного дня в заданий час виконується процес, який запитує з бази даних ті показники, які були виміряні більше заданої кількості днів тому та видаляє їх. Це також допомагає утримувати розмір бази даних, що впливає на продуктивність виконання процесів читання та запису в цю базу даних.

Варто зазначити, що для медичних даних про життєві показники важливим є питання цілісності виміряних даних. Тому в процесі обробки інформації не використовуються фільтри даних, які можуть спотворити виміряні дані. Таким чином, дані зберігаються без модифікацій, і залишають та значення, яку було виміряно медичним сенсором.

Також, окрім зберігання кожного значення показника, в базі даних створюється окремий запис для кожного користувача з біжучими даними. По своїй суті, біжучі дані це ті дані, які були виміряні останнім вимірюванням сенсорів. Цей запис має значення кожного життєвого показника, час його вимірювання та час запису в систему. Також, при обробці вимірюваних даних життєвих показників, повинне обраховуватись середнє значення кожного показника за останній день, тиждень. Такий підхід дозволяє витягувати біжучі дані для подальшої візуалізації не роблячи запит по всіх записах життєвих показників в базі даних, що пришвидшує час виконання витягування інформації з бази даних.

Результатом обробки необроблених даних і зберігання їх в вищезгаданому форматі дозволяє не обробляти інформацію додатково при запиті клієнта на отримання життєвих показників для подальшої візуалізації.

Сервіс агрегації даних для подальшої візуалізації. Як згадувалось вище, при запиті клієнта на отримання актуальних даних, додаткова обробка інформації не потрібна. Проте, агрегація таких даних повинна відбуватись.

Агрегація запитуваних даних може відбуватись за декількома критеріями, заданими запитом клієнта. Запити клієнта можуть варіюватись згідно принципу часу вимірювання показників, а також списком життєвих показників, значення яких потрібно повернути, або одночасно за двома цими критеріями.

Отже, згідно бізнес логіці, сервіс агрегації актуальних даних для подальшої візуалізації повинен вміти агрегувати дані так, щоб задовільнити запит клієнта. А саме: мати змогу повернути значення всіх життєвих показників, конкретного показника, або лише декількох з усіх існуючих у системі та базі даних; мати змогу повертати лише останні значення виміряних життєвих показників (так звані біжучі дані); мати змогу повертати середні значення показників за останній день, тиждень; мати змогу повертати один чи декілька життєвих показників за певний актуальний період часу, наприклад значення всіх чи одного показника за останній день.

Всі ці правила побудовані на основі запитів клієнта, враховуючи принципи подальшої візуалізації.

Сервіс збереження та доступу архівних даних та архівного сховища. Окрім актуальних даних, архівні дані теж можуть мати корисний вплив на оцінювання стану людини. Тому, архівні дані теж повинні оброблятись та зберігатись. Обробка даних та їх зберігання відбувається за схожим принципом, як і актуальні дані. Проте, обчислення середнього значення показників за певний період часу відбувається не під час обробки та запису архівних даних в сховище, а при запиті клієнта, оскільки передбачається що клієнт може вимагати середні значення за певний специфічний період часу, не визначений строго бізнес логікою. Ці значення повинні обчислюватись динамічно. Це збільшує час виконання агрегації даних на запит клієнта, але як зазначалось раніше, час відповіді на запит архівних даних не є критичним.

Також, на відміну від актуальних даних, в архівному сховищі не існує запису з біжучими показниками. Це обгрунтовано тим, що біжучі дані клієнт

повинен отримувати з бази актуальних даних, оскільки такий тип запит відбувається доволі часто і повинен виконуватись швидко, згідно бізнес логіці роботи системи та візуалізації.

Отже, архівне сховище містить лише значення життєвих показників всіх вимірювань та час вимірювання кожного показника. Агрегація цих даних відбувається динамічно, згідно запиту клієнта.

3.4. Висновки до розділу

1. Згідно запропонованим моделі та засобам обробки інформації, побудована структура медичної кіберфізичної системи. З даної структури можна виділити запропоновану архітектурно-інформаційну модель обробки інформації, яка базується на моделі обробки даних, в основі якої лежить мікросервісна архітектура з використанням API шлюзу.
2. Проведено опис функціональних елементів архітектурно-інформаційної моделі обробки інформації. Чітка взаємодія функціональних елементів в архітектурно-інформаційній моделі обробки інформації є засобом оцінювання стану людини в медичній кіберфізичній системі.
3. Описано бізнес логіку роботи медичної КФС, а саме систему зв’язків та залежностей елементів бізнес-даних та правил обробки цих даних.

РОЗДІЛ 4

РЕАЛІЗАЦІЯ СЕРВІСІВ МЕДИЧНОЇ КІБЕРФІЗИЧНОЇ СИСТЕМИ ДЛЯ ОЦІНЮВАННЯ СТАНУ ЛЮДИНИ

4.1. Архітектура та функціонування програмного забезпечення медичної кіберфізичної системи.

Згідно запропонованій архітектурно-інформаційній моделі, реалізація серверного програмного забезпечення медичної КФС відбувається на основі мікросервісної архітектури, де кожен засіб обробки інформації є окремим мікросервісом.

Реалізації мікросервісної архітектури. Для реалізації синхронних повідомлень (клієнт очікує своєчасної відповіді від служби та чекає, поки її отримає) в архітектурі мікросервісів, REST є доцільним вибором, оскільки він забезпечує простий стиль обміну повідомленнями, реалізований за допомогою HTTP-запиту-відповіді, на основі стилю API ресурсів. Тому реалізація мікросервісів використовує HTTP разом зі стилями на основі API ресурсів (кожна функціональність представлена ресурсом та операціями, що виконуються поверх цих ресурсів).

Визначення найкращого формату повідомлення для мікросервісів - ще один ключовий фактор. Традиційні монолітні програми використовують складні двійкові формати. У реалізованій архітектурі використовуються прості формати текстових повідомлень, такі як JSON та XML.

Оскільки в системі є бізнес засоби, реалізовані як сервіси, потрібно визначити та опублікувати договір на обслуговування (документацію). Оскільки мікросервіси створені поверх архітектурного стилю REST, доцільно використовувати ті ж методи визначення REST API для визначення документації мікросервісів. Тому використано стандартну мову визначення API REST, а саме Swagger [76] для визначення контрактів на обслуговування.

В архітектурі мікросервісів програмні засоби будуються як набір незалежних сервісів. Отже, для того, щоб реалізувати бізнес сценарій, потрібно мати комунікаційні структури між різними мікросервісами/процесами. Ось чому міжсервісна/технологічна комунікація між мікросервісами є таким життєво важливим аспектом. Реалізованим підходом у спілкуванні з мікросервісами є використання шлюзу з мінімальними можливостями маршрутизації, який діє за принципом «тупа труба» без бізнес логіки, реалізованої на ньому. Виходячи з цього стилю, в архітектурі мікросервісів доцільно використати відповідний шаблон спілкування.

Ключова ідея стилю API Gateway полягає в тому, щоб використати шлюз повідомлень як основну точку входу для всіх клієнтів/споживачів та реалізувати загальні нефункціональні вимоги на рівні шлюзу. Загалом, шлюз API дозволяє використовувати керований API через REST/HTTP. Тому тут можна представити бізнес засоби, які реалізовані як мікросервіси, через API Gateway, як керовані API. По суті, це поєднання архітектури мікросервісів та API-менеджменту, що дає найкраще з обох світів в даній реалізації.

Для кожного типу клієнтів реалізовано спеціальний API шлюз. Системою передбачено клієнт типу сенсор, та клієнт типу мобільний додаток/браузер.

Реалізація автентифікації та авторизації. Захист мікросервісів є досить поширеною вимогою, коли ви використовуєте мікросервіси в реальних сценаріях. Перш ніж перейти до захисту мікросервісів, давайте коротко подивимося, як зазвичай реалізується безпека на рівні монолітного додатка.

У типовому монолітному застосуванні безпека полягає в тому, щоб знайти "хто є абонентом", "що може зробити абонент" і "як ми поширюємо цю інформацію". Зазвичай це реалізується у загальному компоненті безпеки, який знаходиться на початку ланцюжка обробки запитів, і цей компонент заповнює необхідну інформацію за допомогою базового репозиторію користувачів (або сховища користувачів).

Для безпосереднього переведення цього шаблону у архітектуру мікросервісів, потрібен компонент безпеки, реалізований на кожному рівні мікросервісів, який спілкується з централізованим/спільним сховищем користувачів і отримує необхідну інформацію. Натомість запропоновано використати широко використовувані засоби автентифікації Single Sign-on (SSO), такі як OAuth2 та OpenID Connect. SSO - це служба автентифікації сеансу та користувача, яка дозволяє користувачеві використовувати один набір облікових даних для доступу до кількох програм.

На рисунку 4.1 показано схему автентифікації та авторизації користувачів.

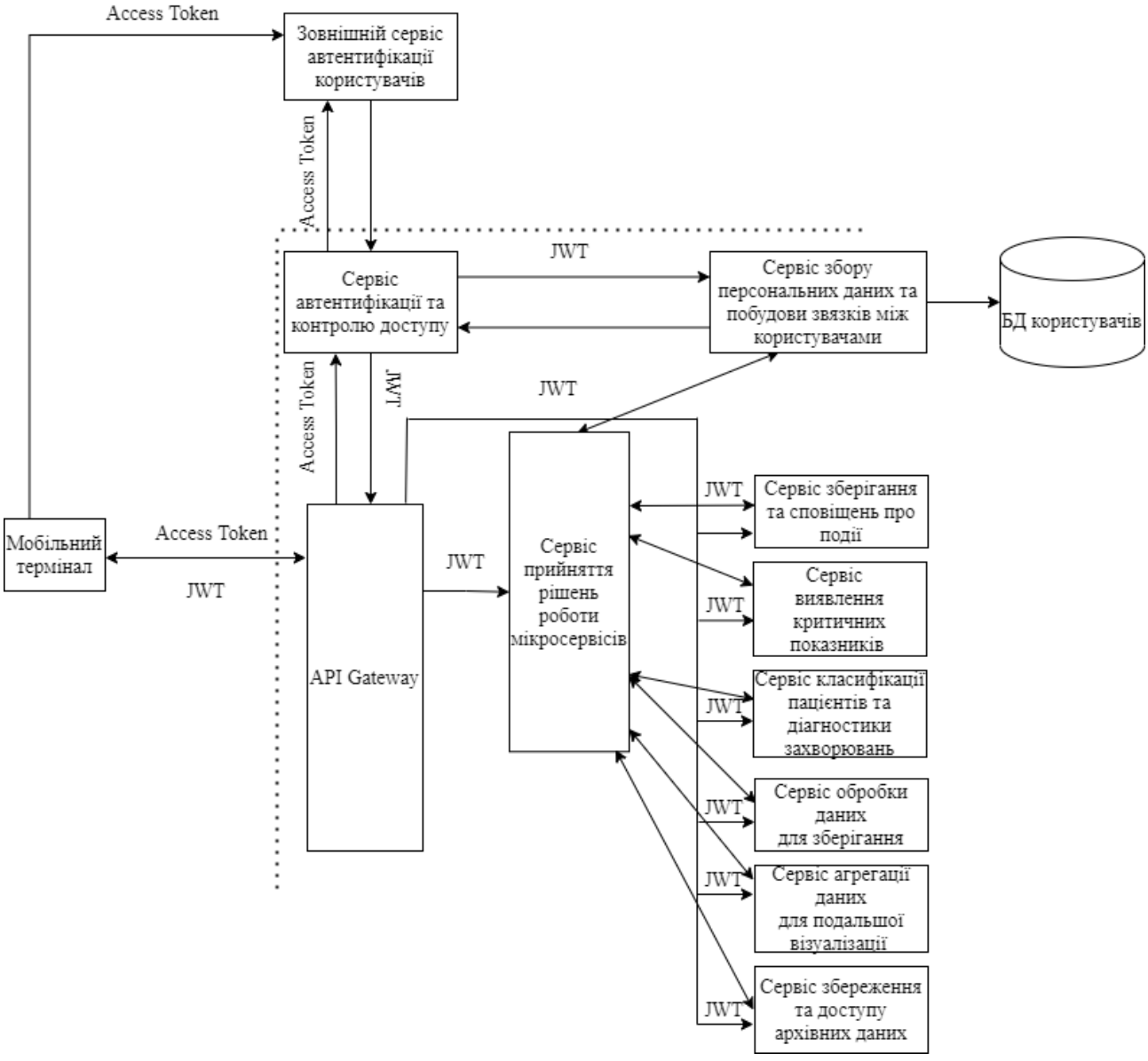

Рисунок 4.1. Схема реалізації автентифікації та авторизації.

Отже, процес автентифікації та авторизації реалізовано наступним чином. Клієнт автентифікується на зовнішньому сервісі автентифікації та отримує Access Token, відомий як «маркер доступу». Токен доступу містить нульову інформацію про користувача/клієнта. Він містить лише посилання на інформацію користувача, яку може отримати лише зовнішній сервісі автентифікації.

Отриманий Access Token відправляється разом з запитом клієнта на автентифікацію до системи, а саме до кінцевої точки API шлюзу. В свою чергу API шлюз повинен автентифікувати користувача, який зробив запит, а також авторизувати його дії. Для цього використовується сервіс автентифікації та контролю доступу, в якому реалізовано зв'язок з зовнішнім сервісом автентифікації користувачів. Сервіс автентифікації та контролю доступу відправляє отриманий від клієнта Access Token до зовнішнього сервісу автентифікації користувачів, і при умові успішної автентифікації зовнішнім сервісом користувача, за допомогою цього токена, сервіс автентифікації та контролю доступу отримує у відповідь дані про користувача.

З отриманими даними, сервіс автентифікації та контролю доступу звертається до сервісу збору персональних даних та побудови зв’язків між користувачами, який в свою чергу запитує дані про автентифікованого користувача в базі даних користувачів на основі зовнішнього ідентифікатора користувача, а саме внутрішній ідентифікатор користувача, який йде у відповідь до сервісу автентифікації та контролю доступу.

Для подальшої авторизації та ідентифікації дій користувача, створюється JWT (Json Web Token). JWT - це відкритий стандарт (RFC 7519), який визначає формат токена та його вміст [73]. JWT токен відправляється користувачеві у відповідь на запит автентифікації. Всі наступні запити клієнт повинен супроводжувати цим JWT, інакше його запити будуть неавторизованими. Токен має обмежений час дії - Expiration Time - час закінчення терміну дії токена в форматі Unix (кількість секунд що пройшли від 1970-01-01T00:00:00Z). Після

закінчення терміну дії токена, користувач повинен пройти процес автентифікації повторно.

Варто зазначити, що створений JWT використовується і при внутрішньому спілкуванні між мікросервісами, для ідентифікації користувача, якому належить запит чи оброблені дані, а також для авторизації запиту користувача, тобто чи має право даний користувач на ту чи іншу запитувану дію.

Реалізації прийняття рішень роботи мікросервісів. Сервіс прийняття рішень роботи мікросервісів відповідає за виконання процесів обробки необроблених даних іншими мікросервісами. Для виконання цих процесів потрібно реалізувати рішення для побудови і опрацювання списку та черг фонових процесів асинхронно, які повинні виконатись для оцінювання стану людини.

Варто зазначити, що разом з занесенням процесів обробки інформації в список виконання, також в список додається процес, який виконається після успішного виконання всіх інших процесів для даного набору даних. Цей процес виконує очищення необроблених даних з оперативної бази даних, оскільки необроблені дані були оброблені і більше не потрібні в системі.

Даний мікросервіс реалізований за допомогою фреймфорка Ruby on Rails (ROR). Він має вбудований фреймворк для декларування відкладених задач (Active Job) та їх виконання у різних чергах. Я використав Sidekiq як адаптер черги для реалізованої системи (див Рис. 4.2). Sidekiq використовує потоки для одночасної роботи з багатьма завданнями в одному процесі. Він надає функції для асинхронного виконання будь-яких завдань або для планування їх подальшого виконання шляхом розміщення їх у черзі фонових завдань. Він також пропонує явні стратегії повторення спроб у разі невдалого виконання. Отже, він чудово відповідає принципу реалізації процесів обробки інформації різними мікросервісами. Для зберігання всіх оперативних даних використовується сховище даних Redis.

Також Sidekiq має вбудований графічний інтерфейс (див. Рис. 4.2), на якому можна відслідковувати різні фонові задачі. Наприклад: опрацьовані задачі; невдалі; ті, що в процесі виконання; в черзі;, заплановані; та невдалі задачі.

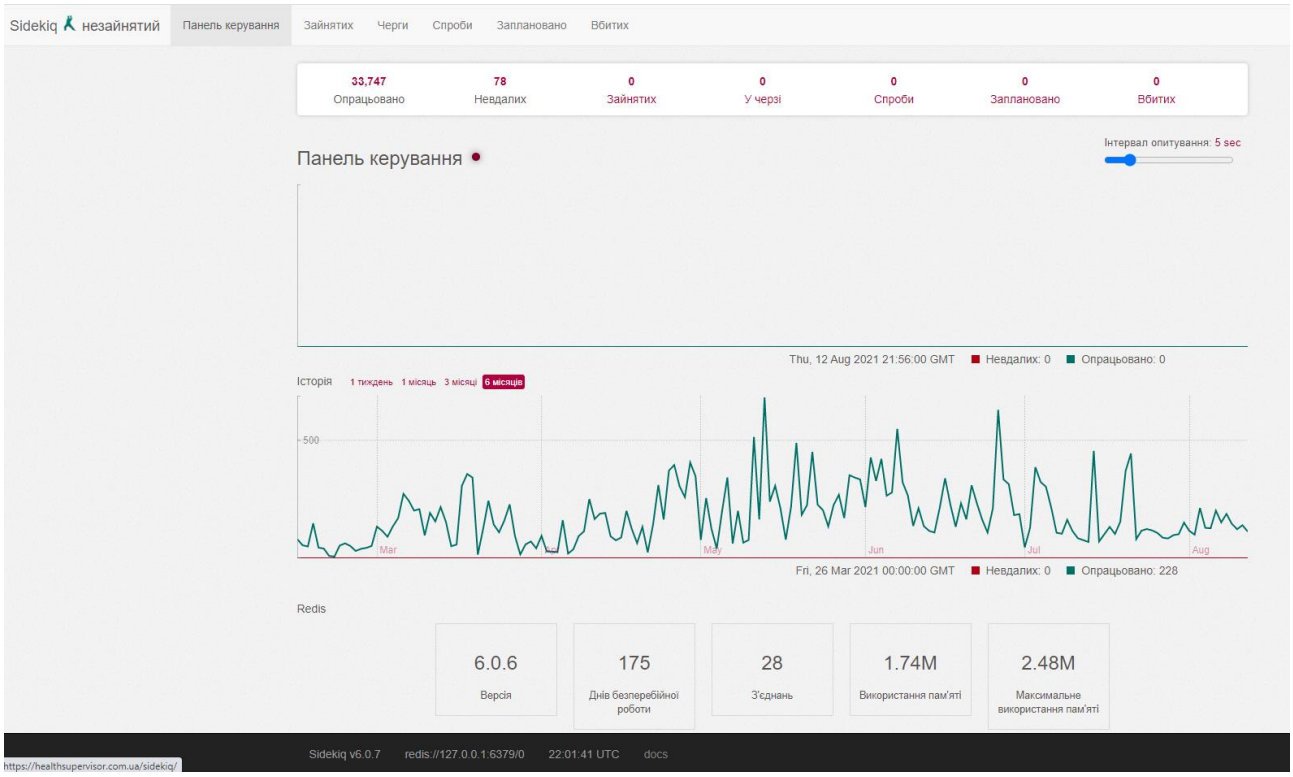

Рисунок 4.2. Кількість виконаних обробок інформації в системі HealthyLungs у графічному інтерфейсі Sidekiq.

Реалізація системи сповіщень про події. Реалізація сповіщень про події поділена на два підходи, в залежності від рівня критичності події.

Для інформаційних подій, сповіщення про події відбуваються в момент виконання запиту на сервер користувачем. У випадку, якщо для даного користувача в базі даних подій знаходяться інформаційні події, про які він ще не був сповіщений, він отримає інформацію про існування таких подій разом з запитуваною інформацією. Деталі цих подій користувач може отримати зробивши відповідний запит.

Реалізація сповіщень про критичні події дещо складніша. Оскільки рівень критичності події вимагає негайного реагування, обмежитись підходом очікування запиту користувача, як з інформаційними подіями, не доцільний. Тому, в запропонованій медичній КФС реалізовано негайний спосіб сповіщення про критичні події.

Отже, в запропонованій медичній КФС реалізовано два канали зв'язку для сповіщення про критичні події: лист на електронну пошту та Push-сповіщення.

При виявленні критичної події, асинхронно відбуваються процеси сповіщення через різні канали зв'язку. А саме, відправляються листи та Push-сповіщення на мобільний додаток. Варто зазначити, що дані сповіщення відправляються не лише одному користувачеві, а групі користувачів, при умові що вони ведуть спостереження за особою, стан якої оцінений як критичний і викликав критичну подію.

Для реалізації відправлення електронних листів, використано сервіс SendGrid. Імплементація використання цього сервісу передбачає генерування API ключа та його використання при подальших запитах системи до сервісу для відправлення листа на електронну адресу відповідальної особи чи групи осіб.

Для реалізації відправлення Push-сповіщень на мобільний додаток відповідальної особи, чи групи осіб, використовується бібліотека Rpush, розроблена для сервісів побудованих на основі ROR. Вона підтримує платформи push-сповіщень Google, Apple, Microsoft та Amazon.

Реалізацію відправлення Push-сповіщень на мобільний додаток Android можна описати наступним чином. Перш ніж мати змогу відправляти сповіщення на додаток, потрібно згенерувати облікові дані для мобільного додатку на стороні сервісу Google. Використання створених облікових даних мікросервісом виглядає наступним чином (див Рис. 4.3).

```
app = Rpush::Gcm::App.new
app.name = “medical_cps_client_app”
app.auth_key = “API key”
app.connections = 1
app.save!
```

Рисунок 4.3. Конфігурація Rpush бібліотеки для виконання сповіщень на мобільний додаток Andorid.

Як показано на Рис.4.4, створюється Push-сповіщення, яке надсилається згідно назви мобільного додатку. Отримувачі, тобто відповідальна особа або група осіб, визначаються за допомогою поля registration_ids. Registration_id присвоюється клієнту, тобто мобільному додатку при його встановленні на мобільний термінал.

```
notification = Rpush::Gcm::Notification.new
notification.app = Rpush::Gcm::App.find_by(name: “medical_cps_client_app”)
notification.registration_ids = […]
notification.data = { message: “Alert!” }
notification.save!
Rpush.push
```

Рисунок 4.4. Створення та відправка Push-сповіщення.

Було проведено тестування сповіщення про критичну подію. Згідно проведеному тестуванні, після виявлення критичної події на її створення та сповіщення про неї в середньому потрібно 2225мс, тобто 2,2 секунди.

Реалізація виявлення критичних показників. Система отримує набір життєвих показників стану людини за певний період часу. В ідеальних умовах, сенсори надсилають дані системі в режимі близького до реального часу.

Отже, сервіс виявлення критичних показників згідно логіці та порядку виконання процесів обробки інформації, бере необроблені дані з оперативної бази даних.

Тому кожен з життєвих показників має лише одне значення для одиниці часу, потрібної для проведення вимірювання показника.

Згідно запропонованого методу, обчислюється оцінка для кожного показника (від 0 до 3). При умові, що хоча б один з показників має оцінку 3, створюється критична подія і сервіс сповіщення про події інформується відповідно, приймаючи подальші рішення на рахунок сповіщення про цю подію.

Варто зазначити, що при виявленні значення оцінки будь-якого життєвого показника, яке рівне 3, створення критичної події відбувається в функції обчислення суми оцінок (calculate_aggregated_score). Це означає, що критична подія створиться негайно, не очікуючи закінчення обчислення оцінки всіх показників та їх суми.

Сума оцінок показників, яка перевищує значення 0 стає причиною створення події про те, що показники стану людини не є в нормі. При виявленні критичних показників, з необроблених даних до уваги беруться найновіші дані, тобто значення показників останнього вимірювання. Тобто, незалежно від розміру вхідних даних, час, необхідний для виявлення актуальних критичних показників є однаковим. Тим не менш, виявлення критичних показників відбувається для всіх наборів необроблених даних, починаючи з найновіших, проте створення критичної події відбувається лише для останнього вимірювання, оскільки саме останні дані потребують негайного виявлення рівня критичності. Події про критичність старіших вимірювання, створюються в базі як інформаційні події. Відповідно сповіщення про них не відбувається. Приклад, як відбувається виявлення критичних показників наведено на рисунках 4.5 та 4.6.

```
def inspect_indicators(indicators_data)
  last_indicator_time = DataAggregationService.get_running_data(user_id: current_user.id).created_at
  measuring_time_offset =  + indicators_ data[:durationInSeconds]
  indicators_data[:values].each do |indicator_values|
```

```
    index = o
    one_measurement_values = []
    measurement_offset = indicators_data[:durationInSeconds]
    indicator_values.each do |indicator_value|
      measuring_time = DateTime.strptime((indicators_data[:startTimeInSeconds] +
indicator_value.key).to_s, '%s')
      return if measuring_time < last_indicator_time
      one_measurement_values << indicator_value[:measurement_offset]
      aggregated_score = calculate_aggregated_score(one_measurement_values)
      if aggregated_score = 0
        measurement_offset -= 60
        next
      else
        event_type = measurement_offset == indicators_data[:durationInSeconds] ?
"informating" : "critical"
        EventService.call(
          name: “critical_aggregated_score”,
          type: event_type,
          measured_at:  measuring_time_offset,
          user_id: current_user.id,
          aggregated_score: aggregated_score
        )
        measurement_offset -= 60
      end
    end
  end
end
```

Рисунок 4.5. Виявлення критичних показників методом обчислення суми оцінок показників та надсилання запиту створення відповідної події.

```
def calculate_aggregated_score(one_measurement_values)
  score = 0
  one_measurement_values.each do |indicator_type, indicator_value|
    index = 0
    indicator_score = Indicator.calculate_score(indicator_value, indicator_type)
    if indicator_score == 3 and index == 0
      EventService.call(
        name: “critical_indicator_value”,
        type: “critical”,
        measured_at:  measuring_time,
        user_id: current_user.id,
        indicator_type: indicator_type,
        indicator_value: value
      )
    end
    index += 1
    score += indicator_score
  end
  return score
end
```

Рисунок 4.6. Обчислення суми оцінок показників та надсилання запиту створення критичної події, при виявлення оцінки рівної 3.

Проведено тестування виявлення критичних показників. Згідно проведеному тестуванні, на виявлення критичних показників останнього вимірювання та надсилання запиту створення критичної події в середньому потрібно 1236,69мс, тобто 1,24 секунди.

Реалізація класифікації пацієнтів та діагностики захворювань полягає в виконанні квантової версії адаптованого алгоритму K-means. Цей алгоритм виконується на цифровому квантовому співпроцесорі на ПЛІС. Цифровий

квантовий співпроцесор призначений для реалізації алгоритмів виконання на аналогових квантових комп'ютерах. Цифровий квантовий співпроцесор працює під управлінням класичного комп’ютера, і разом вони є цифровим квантовим комп’ютером [60] [61].

Для реалізацій адаптованого квантового алгоритму K-means, потрібно використати три різні квантові підпрограми: SwapTest, DistCalc та Optimization Grovers.

Підпрограма SwaptTest, яка вперше була використана в [11], вимірює перекриття двох квантових станів $\langle\alpha|\beta\rangle$ на основі ймовірності вимірювання керуючого кубіта у стані $|0\rangle$. Перекриття є мірою подібності між двома станами. Якщо ймовірність керуючого кубіта у стані $|0\rangle$ дорівнює 0,5, це означає, що стани $|\alpha\rangle$ та $|\beta\rangle$ ортогональні, тоді як якщо ймовірність дорівнює 1, то стани ідентичні. Стани $|\alpha\rangle$ та $|\beta\rangle$ можуть бути невідомими перед запуском процедури, і потрібно лише вимірювання на контрольному кубіті. Наведена ймовірність перебування керованого кубіту в стані $|0\rangle$ (див. формулу 4.1).

$$P(|0\rangle) = \frac{1}{2} + \frac{1}{2}|\langle\alpha|\beta\rangle|^2 \qquad (4.1)$$

Схема SwaptTest показана на рисунку 4.7.

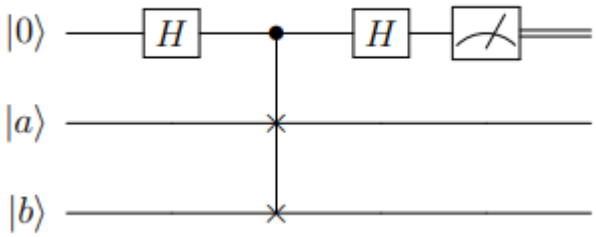

Рисунок 4.7. Схема підпрограми SwaptTest.

Стани $|\alpha\rangle$ та $|\beta\rangle$ складаються з n кубітів кожен. Вони або готуються з використанням амплітудного кодування, або їх можна завантажити безпосередньо з пам’яті квантового випадкового доступу (QRAM). Потім перекриття обчислюється шляхом вимірювання на контрольному кубіті.

Перекриття з підпрограми SwapTest використовується для обчислення відстані в підпрограмі DistCal. Алгоритм обчислення відстані за допомогою SwapTest описано в [30]. Евклідова відстань $|\alpha - \beta|^2$ між двома векторами $|\alpha\rangle$ та $|\beta\rangle$ обчислюється шляхом виконання наступних трьох кроків:

1. Підготовка станів: Підготувати два квантових стани,

$$|\vartheta\rangle = \frac{1}{\sqrt{2}}(|0, \alpha\rangle + |1, \beta\rangle)$$

$$|\varphi\rangle = \frac{1}{\sqrt{Z}}(|\alpha||0\rangle + |\beta||1\rangle)$$

де $Z = |\alpha|^2 + |\beta|^2$.

2. Знаходження перекриття: Обчислити перекриття $\langle\vartheta|\varphi\rangle$ використовуючи SwapTest.

3. Обчислення відстані: Використовуючи формулу 4.2 обчислити Евклідову відстань.

$$Distance = 2Z|\langle\vartheta|\varphi\rangle|^2 \qquad (4.2)$$

За допомогою цього методу К відстані обчислюються до кожного центроїда кластера. Щоб знайти найближчий кластерний центроїд, викликається підпрограма Grovers Optimization [4], яка базується на алгоритмі Гровера [5]. Точка вхідних даних потім призначається кластеру цього найближчого центроїда. Це робиться для всіх точок вхідних даних. Після призначення кластерів кожній точці даних центроїди перераховуються шляхом обчислення середнього значення всіх точок даних у відповідних кластерах. Весь процес повторюється, поки призначення кластерів не перестане змінюватись.

Вузьким місцем у класичній версії алгоритму K-means є обчислення відстані між N-мірними векторами. Завдяки ефективному способу обчислення відстаней за допомогою квантового паралелізму адаптований квантовий алгоритм K-means досягає експоненціального прискорення. Алгоритм К- means

Ллойда [1] має часову складність $O(NMK)$, тоді як квантова версія має часову складність $O(\log(N)\,MK)$. Це можна пояснити, враховуючи, що N-розмірна класична інформація кодується за допомогою $\log N$ кубітів.

Реалізація обробки актуальних даних та їх агрегації. Реалізація сервісу обробки актуальних даних теж пов'язана з обробкою необроблених даних.

Наприклад, необроблені дані вимірювань такого показника як рівень насиченості кисню в крові, виміряний вимірювальним пристроєм Garmin [63] виглядають так (див Рис.4.8).

"garmin"=>{"pulseox"=>[{"userId"=>"293eb46d-5238-4ad6-9503-48736d293015", "userAccessToken"=>"303aad3b-cc82-4ff5-b1f1-3b155c849943", "summaryId"=>"x3a72c14-61154580", "calendarDate"=>"2021-08-13", "startTimeInSeconds"=>1628784000, "durationInSeconds"=>14280, "startTimeOffsetInSeconds"=>28800, "timeOffsetSpo2Values"=>{"540"=>91, "780"=>93, "840"=>93, "1080"=>95, "1140"=>96, "1380"=>96, "2040"=>97, "2880"=>97, "3420"=>98, "3600"=>96, "3900"=>90, "3960"=>91, "4020"=>95, "4320"=>99, "5040"=>99, "5280"=>96, "5340"=>94, "5460"=>94, "5580"=>100, "8340"=>100, "8460"=>100, "8580"=>100, "8700"=>98, "8760"=>97, "8880"=>96, "9060"=>94, "9180"=>94, "9420"=>94, "9480"=>94, "9540"=>94, "9660"=>93, "9780"=>93, "9900"=>93, "10020"=>93, "10200"=>92, "10260"=>89, "10320"=>90, "10380"=>91, "10440"=>91, "10500"=>90, "10560"=>93, "10620"=>93, "10680"=>91, "10740"=>92, "10800"=>91, "10860"=>92, "10920"=>90, "10980"=>91, "11040"=>88, "11100"=>88, "11160"=>91, "11220"=>91, "11280"=>91, "11340"=>92, "11400"=>92, "11460"=>92, "11520"=>92, "11580"=>92, "11640"=>92, "11700"=>91, "11760"=>91, "11820"=>91, "11940"=>90, "12360"=>89, "12660"=>89, "12720"=>89, "12780"=>87, "12840"=>87, "12960"=>87, "13020"=>88, "13080"=>88, "13140"=>90, "13200"=>100, "13260"=>100, "13320"=>100, "13380"=>100, "13440"=>100, "13500"=>100, "13560"=>100, "13620"=>100, "13680"=>99, "13740"=>98, "14220"=>96}, "onDemand"=>false}]}}

Рисунок 4.8. Приклад необроблених вхідних даних вимірювань показника рівня кисню в крові.

Отже, приходять не тільки останні дані вимірювань, але всі вимірювання за останній день. Це вимагає такої обробки необроблених даних, яка забезпечить уникнення дублювання показників в базі актуальних даних. Час вимірювання в

необроблених даних містить інформацію в форматі часу Unix, що формує потребу конвертування в звичну форму.

```
start_time = correct_hash[:startTimeInSeconds]
timeOffsetSpo2Values = raw_hash[:timeOffsetSpo2Values]
last_measurement_time = DataAggregationService.get_running_data(user_id: current_user.id).created_at
timeOffsetSpo2Values.reverse.each do |k, v|
  index = 0
  time = DateTime.strptime((start_time.to_i + k.to_i).to_s, '%s')
  if time > last_measurement_time
    OxygenIndicators.create(
      user_id: current_user.id,
      oxygen: v.to_f,
      updated_at: time,
      source: 'garmin'
    )
  end
  if index == 0
    RunningIndicators.where(user_id: current_user.id).update(
      oxygen: v.to_f,
      oxygen_updated_at: time,
      oxygen_avg_daily: calculcate_average_daily_value(oxygen),
      oxygen_avg_weekly: calculcate_average_weekily_value(oxygen),
    )
  end
    index += 1
end
```

Рисунок 4.9. Приклад обробки даних показника рівня насиченості крові виміряним пристроєм Garmin.

Реалізації обробки необроблених даних показана на прикладі обробки необроблених даних показника рівня насиченості киснем крові використовуючи ROR на рисунку 4.9. Першим кроком є визначення потрібних змінних, а саме: час початку вимірювань; хеш даних, де ключ - це кількість секунд, які пройшли від часу початку вимірювань, а значення - це показник. Також визначається час вимірювання останнього запису показника існуючого в базі даних.

При обробці необроблених даних, час вимірювання показника конвертується з формату часу Unix в звичний формат. Конвертований час вимірювання порівнюється з часом вимірювання останнього запису, і при умові що вхідні вимірювання новіші, значення показників записуються в базу даних.

При записі в базу даних, дані структуруються. При записі біжучих даних (останніх вимірювань), також обчислюється середнє значення всіх показників за останній день та тиждень.

В реалізованій медичній КФС, вимірювання біосенсорами відбуваються кожну хвилину. Відповідно, після проведення кожного вимірювання, дані відправляються на сервер. Вхідні дані складаються з набору значень, виміряних для кожного показника за весь період часу, починаючи з 00:00 поточного дня. При обробці даних, першим чином перевіряється час вимірювання останніх збережених показників в системі, тому дані показників, які вже були оброблені в збережені в системі, додатково не обробляються. Таким чином, якщо дані для обробки поступають регулярно, обробляються лише дані останніх вимірювань.

Проте, за певних умов (наприклад, за відсутності з’єднання з мережею Інтернет), дані кожного вимірювання не відправлялись і накопичувались на клієнті. В такому випадку відбувається обробки даних не лише одного вимірювання, а багатьох, в залежності від тривалості відсутності з’єднання. Максимальний об’єм даних, який може поступити для обробки на сервер, може включати в себе вимірювання показників за весь день (запит на обробку даних прийшов вперше в 23:59). В такому випадку, вхідні дані містять інформацію про

показники за 1439 вимірювань, тобто складаються з 1439 наборів даних з значенням кожного показника.

Згідно проведених тестувань, було визначено середній час обробки вхідних даних в залежності від їх об’єму. А саме, при регулярному надсиланні даних кожного вимірювання, та при надсиланні максимального об’єму даних - набору вимірювань за весь день (див. Табл. 4.1).

Таблиця 4.1. Залежність часу виконання обробки вхідних даних в залежності від їх об’єму.

Об’єм вхідних даних	**Час виконання обробки даних**
Набір даних 1 вимірювання	149мс, тобто 0,15 секунди
Набір даних 1439 вимірювань	4200мс, тобто 4,2 секунди

Для зберігання лише актуальних даних реалізовано процес видалення неактуальних даних. Використано службу планування задач UNIX Cron. Список та час виконання запланованих задач створюється в спеціальному файлі crontab. В цей список заноситься вбудований в Ruby механізм виконання зовнішніх задач, який називається Rake Task. Отже, crontab файл виглядає наступним чином (див Рис.4.10).

```
0 0 * * * rake Cleanup:IrrelevantData
```

Рисунок 4.10. Вміст файлу crontab враховуючи заплановану задачу очищення актуальної бази даних

Такий запис означає, що кожного дня в 00:00 буде виконуватись запланована Rake задача *Cleanup:IrrelevantData*, яка запитує всі записи показників, час вимірювання яких більший, ніж заданий в системі час актуальності даних та видаляє їх, оскільки такі дані більше не актуальні.

Також, проведено тестування часу виконання агрегації даних, в залежності від параметрів запиту даних. Тестування проводилось згідно запитів, які формує

клієнтська програма. Середній час виконання агрегації даних показано на таблиці 4.2.

Таблиця 4.2. Залежність часу виконання агрегації збережених даних в залежності від типу запиту клієнта.

Параметри запиту	**Середній час виконання агрегації даних**
Отримання біжучих значень усіх показників	19мс, тобто 0,02 секунди
Отримання середніх значень всіх показників за день	15мс, тобто 0,015 секунди
Отримання значень конкретного показника за день	89мс, тобто 0,09 секунди
Отримання значень конкретного показника за тиждень	284мс, тобто 0,29 секунди
Отримання значень конкретного показника за місяць	678мс, тобто 0,68 секунди

Реалізація архівування та обробки історичних даних. Історичні (архівні) дані являють собою великі дані та потребують рішення, яке здатне такі дані обробляти та зберігати. Я пропоную використати готове рішення – Hadoop [77].

Hadoop - це розподілена системна інфраструктура, розроблена Фондом Apache. Користувачі можуть розробляти розподілені програми, не розуміючи основних деталей архітектури, щоб вони могли повною мірою використовувати величезні можливості зберігання даних платформи та швидкі обчислювальні можливості. Hadoop реалізує розподілену файлову систему, яка називається HDFS [12].

Основними функціональними елементами фреймворка Hadoop - це HDFS та Map Reduce. HDFS в основному забезпечує масове зберігання даних, а Map Reduce надає розподілені обчислювальні послуги для даних.

Hive [74] - це програмне забезпечення, що використовується для запитування даних, реалізоване поверх HDFS Hadoop для забезпечення запитів та аналізу даних. Крім того, Hive дозволяє створювати сховище метаданих у вигляді таблиць у системі реляційних баз даних. Це робить реалізовану систему більш ефективною з точки зору скорочення часу доступу до архіву пацієнтів. В реалізованій системі встановлено Hive на головному вузлі кластера і створено зовнішню таблицю бази даних, розташовану в головному каталозі HDFS, де система може запитувати дані, імпортовані до HDFS за допомогою консолі HiveQL.

4.2. Організація зберігання даних в медичній кіберфізичній системі.

Організація зберігання персональних даних та зв'язків між користувачами реалізована за допомогою об'єктно-реляційної СУБД PostgreSQL. Структура таблиць в базі даних та зв'язків зображена на рисунку 4.11.

Таким чином, в даній БД зберігаються необхідні персональні дані та ідентифікатори користувачів, а також створюються зв'язки багато-до-багатьох між користувачами через сутність групи.

Таблиця *users* містить записи користувачів, включаючи такі дані: повне ім'я; електронна адреса; номер телефону; зовнішній ідентифікатор SSO; час реєстрації; дата народження; вага та ріст; ідентифікатор клієнтської програми.

Таблиця *group_users* містить записи зв'язків між користувачами та групами, а саме: ідентифікатори користувача та групи; час створення звязку; статус та час його зміни.

Таблиця *groups* містить записи груп, включаючи такі дані: ідентифікатор відповідальної особи (власника групи); назву; ідентифікатор, який використовується для запрошень в групу.

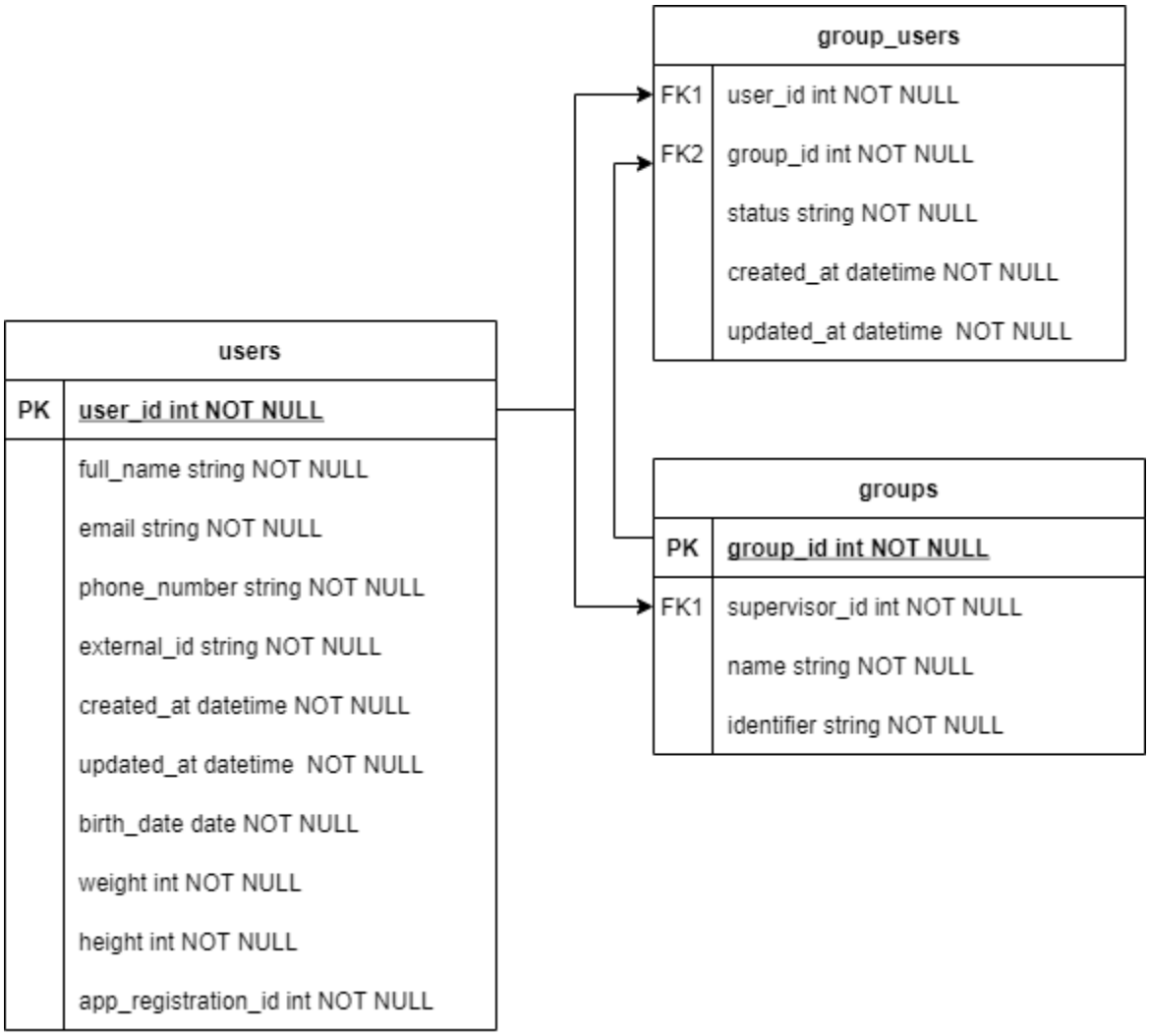

Рисунок 4.11. Структура таблиць користувачів та їх зв’язків.

Організація зберігання необроблених даних в оперативній базі даних реалізована за допомогою MongoDB. Це документо-орієнтована NoSQL СУБД, яка не потребує опису схеми таблиць. Ця СУБД підтримує зберігання документів в JSON-подібному форматі, що задовольняє потреби швидкого запису необроблених неструктурованих даних. Запис необроблених даних в оперативну БД зображений на рисунку 4.12.

При чому, raw_data може мати довільну структуру. Реалізація оперативної БД як MongoDB вирішує проблему неоднорідності вимірювальних пристроїв, структура необроблених даних яких є гетерогенною. Це означає, що незалежно від типу вимірювального пристрою, логіка зберігання необроблених даних в оперативну БД є однаковою.

```
{
  identifier: {
    user_id: user_id,
    indicators_raw_data: raw_data
  }
}
```

Рисунок .4.12. Структура зберігання необроблених даних в оперативну БД.

Організація зберігання фонових задач, які повинні виконатись для обробки необроблених даних реалізована за допомогою Redis DB. Redis - це відкрите сховище структури даних у оперативній пам’яті, яке використовується як база даних. Redis є резидентною СУБД класу NoSQL, яка працює з структурами даних типу «ключ-значення».

Варто зазначити, що для зберігання даних в Redis використовується оперативна пам'ять. Тому, доцільно зберігати якомога менше інформації, пов’язаної з тою чи іншою фоновою задачею. В реалізованій системі це лише 2 значення.

В реалізації збереження фонових задач, ключем є унікальний ідентифікатор задачі, а значенням є вхідні дані, для виконання процесу обробки необроблених даних, як, назва сервісу, який повинен виконати свою функцію, ідентифікатор необроблених даних в оперативній БД.

Організація зберігання подій. При виявленні події, сервіс сповіщення про події обробляє отримані дані про виявлену подію, та окрім сповіщення про події виконує зберігання даних про цю подію в базу даних. База даних подій реалізована за допомогою реляційної СУБД PostgreSQL. Структура таблиці події показана на рисунку 4.13.

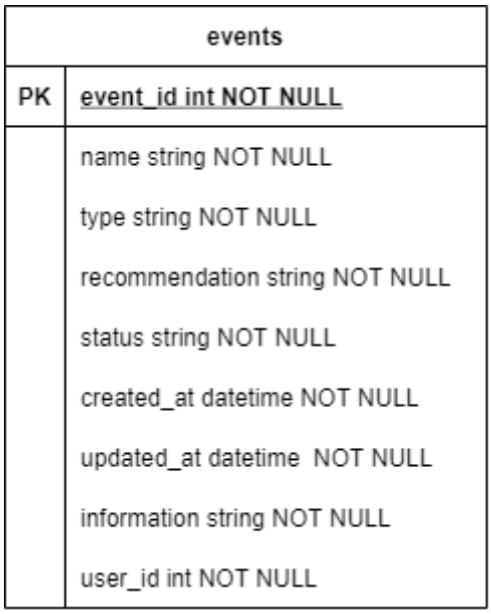

Рисунок 4.13. Структура таблиці подій.

Отже, згідно рисунку 4.13 має певний набір параметрів, необхідних для інформативного сповіщення про подію та моніторингу статусу ознайомлення з подією. Назва події несе інформативний характер, пояснюючи причини створення події. Типу події визначається рівнем критичності (критична або інформативна). Поле рекомендації може містити для деяких подій інформацію про рекомендацію прийняття певних дій, в залежності від типу та назви події. Поле статус відповідає за індикацію сповіщення про подію, чи вона була переглянута. Часові поля визначають час створення події, та час внесення змін в подію, таких як зміна її статусу. Поле інформації містить більш детальну інформацію про причину створення події, коли назви події недостатньо. Наприклад, коли назва події може бути «виявлення критичних показників», то поле інформації містить детальний опис які показники є критичними, значення цих показників. Поле user_id необхідне для зв’язку між подією і користувачем, якого дана подія стосується.

Організація зберігання оброблених актуальних даних. Оскільки база актуальних даних постійно очищується, доцільно використати реляційну СУБД PostreSQL для зберігання біжучих та актуальних даних про життєві показники стану людини.

Структура біжучих та актуальних даних зображена на рисунку 4.14.

x_indicators	
PK	x_indicator_id int NOT NULL
	value float NOT NULL
	created_at datetime NOT NULL
	measured_at datetime NOT NULL
	user_id int NOT NULL

running_indicators	
PK	running_indicator_id int NOT NULL
	user_id int NOT NULL
	x_indicator_value float NOT NULL
	x_indicator_avg_daily_value float NOT NULL
	x_indicator_avg_weekly_value float NOT NULL

Рисунок 4.14. Структура таблиць зберігання актуальних даних.

Як зображено на рисунку 4.13, кожен показник має власну таблицю записів, відповідно назва таблиці залежить від назви показника. Структура запису в такій таблиці є однаковою для кожного показника, а саме: значеня показника; час його вимірювання; час запису в систему; ідентифікатор користувача, якому належать показники.

Також існує окрема таблиця для біжучих актуальних даних. Структура запису в цій таблиці наступна: ідентифікатор користувача, якому належать біжучі показники; значення останнього вимірювання кожного показника, який використовується в системі; середнє значення кожного показника за поточний день; середнє значення кожного показника за поточний тиждень. Цей підхід реалізовано для спрощення агрегації та формування запитів для отримання різного типу інформації.

Організація зберігання архівних даних реалізована за допомогою розподіленої файлової системи Hadoop HDFS [77], яка зберігає дані на вузлах кластера. Hadoop HDFS розбиває отримані дані на блоки, а потім розподіляє їх по вузлах. Загалом, характеризується двома аспектами: по-перше, забезпечує високу надійність даних завдяки процесу реплікації даних у вузли (принаймні три). Цей аспект є дуже корисним у медичних кіберфізичних системах, оскільки архіви пацієнтів ніколи не будуть втрачені, якщо станеться апаратний збій. По-

друге, дані в HDFS Hadoop обробляються паралельно, тому забезпечується високий процес пошуку даних.

В даній системі реалізовано три вузли кластера, впровадивши Hadoop HDFS на кожному з них для зберігання записів пацієнтів у файлах HDFS.

4.3. Медична кіберфізична система «HealthyLungs».

Медична кіберфізична система «HealthyLungs» [64][65][66] розроблена для цілодобового віддаленого моніторингу функцій легень у пацієнтів з COVID-19.

Основні можливості, які надає біомедична кіберфізична система цілодобового моніторингу функцій легень у пацієнтів із COVID-19:

- Закріплення клієнтів за лікарями або тренерами, інструкторами тощо;
- Автоматичне збирання даних про стан легень клієнтів без їх участі та передавання цих даних через мережу Інтернет;
- Дистанційний контроль лікарями рівня кисню в крові пацієнтів з COVID-19;
- Дистанційний контроль лікарями частоти пульсу пацієнтів із COVID-19;
- Дистанційний контроль лікарями респірації в пацієнтів із COVID-19;
- Можливість додавання функцій контролю інших параметрів;
- Накопичення результатів контролю та архівування історії показів кожного пацієнта, а також їх знеособлення для подальшої статистичної обробки;
- Проведення моніторингу 24 години на добу та без контакту між клієнтом та лікарем;
- Наявність функції попередження лікаря про небезпечно низький рівень показників;
- Забезпечення можливості проведення самоконтролю клієнтами функціонування легень.

Переваги використання створених засобів:

1. Постійне клінічне спостереження за клієнтами підвищує шанси на те, що їх стан не стане критичним.

2. Використання віддаленого моніторингу клієнтів зменшує потребу у фізичному контакті та дозволяє лікарям дистанційно стежити за станом клієнтів 24 години на день, 7 днів на тиждень у лікарні, спортзалі, на роботі або вдома.

3. Зменшується кількість хворих у лікарнях та перевантаженість персоналу.

4. За допомогою засобів Біомедичної кіберфізичної системи цілодобового моніторингу функцій легень у пацієнтів із COVID-19 лікар більше не обмежується визначенням діагнозу на основі необроблених, негайних даних, отриманих із пояснень клієнта.

5. Засоби Біомедичної кіберфізичної системи цілодобового моніторингу функцій легень у пацієнтів із COVID-19 дають можливість клієнту відстежувати стан свого організму.

6. Засоби Біомедичної кіберфізичної системи цілодобового моніторингу функцій легень у пацієнтів із COVID-19 забезпечують організоване зберігання отриманих даних про стан пацієнта за час спостереження. Дані знеособлюються і обробляються для отримання загальної картини в регіоні, професії, занятті.

Цілодобовий метод дистанційного моніторингу. Були розроблені інструменти, які дозволяють керівникам одночасно віддалено контролювати функціональний стан багатьох клієнтів у лікарняних палатах (тих, що знаходяться під спостереженням і тих, що перебувають у критичному стані) без прямого контакту з ними, а також інтегрувати інтелектуальні функції, щоб попередити керівників про небезпечно низький рівень показників, а також тих, хто лікуються вдома (рис. 4.15). Відповідальна особа може встановити межі індикаторів для кожного клієнта особисто і буде поінформований, коли ці межі будуть перетнуті. Це дозволяє керівнику встановлювати обмеження відповідно до стану клієнта, особливостей тощо [64][65][66].

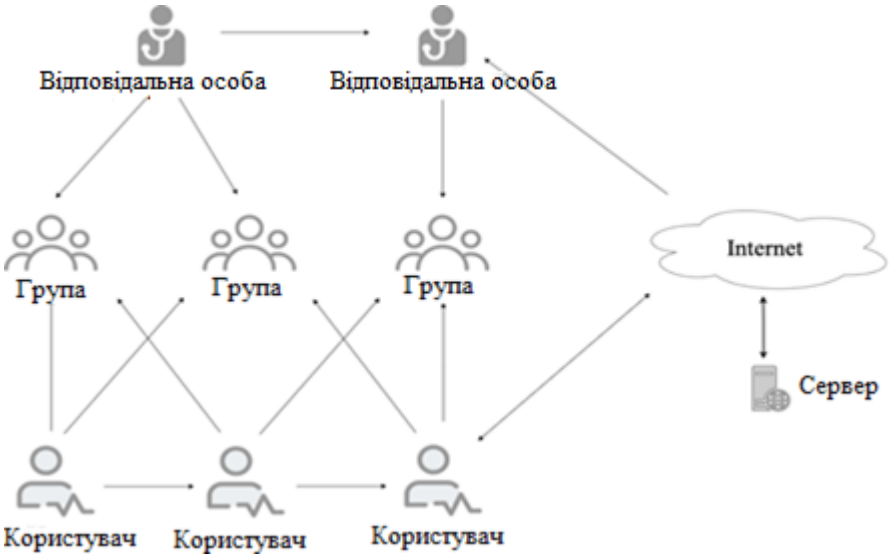

Рисунок 4.15. Цілодобовий моніторинг стану клієнтів відповідальною особою.

Клієнти оснащені персональними вимірювальними пристроями, підключеними до їх смартфонів. Вибір вимірювальних пристроїв, які підтримується системою описано в [63]. Дані, отримані зі смартфонів клієнтів, автоматично передаються через Інтернет на сервер. На стороні сервера дані обробляються та зберігаються. Супервайзори мають доступ до клієнтських даних на сервері з мобільних додатків на своїх смартфонах, які представлені у вигляді різних графіків та таблиць. Кожен супервайзор має можливість створювати власні групи клієнтів, що дозволяє їм відрізняти клієнтів за назвою групи.

Ці ж інструменти надають клієнтам можливість самостійно контролювати рівень кисню в крові, частоту пульсу та частоту дихання, дозволяють отримати результати тестів та рекомендації щодо їх покращення.

Елементи системи моніторингу та організації мережі передачі даних. Система моніторингу включає: мережу бездротових датчиків та пов’язаних смартфонів із встановленим мобільним додатком HealthyLungs, які використовуються для взаємодії з бездротовими датчиками та для обробки результатів моніторингу стану клієнтів; Серверне програмне забезпечення для збору рівня кисню в крові в реальному часі, частоти пульсу та частоти дихання

24 години на добу від спостережуваних та важкохворих клієнтів, забезпечуючи захист персональних даних; Мобільні програми супервізора HealthyLungs для моніторингу функціонального стану клієнтів та попередження керівника про небезпечно низький рівень показників [64][65][66].

Структура цілодобової системи дистанційного моніторингу наведена на рисунку 4.16.

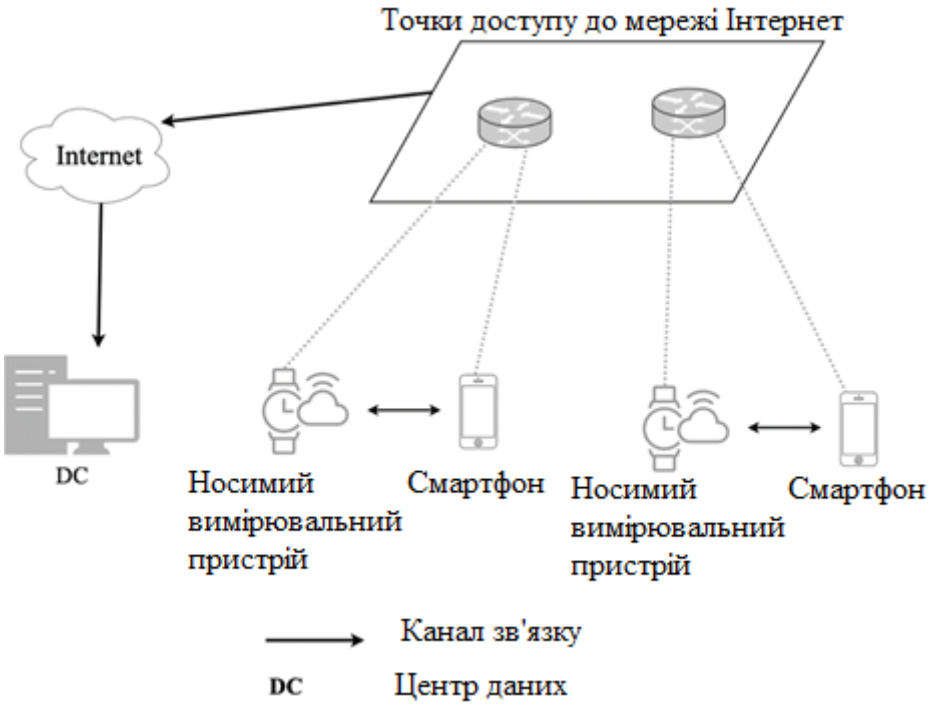

Рисунок 4.16. Структура системи цілодобового дистанційного моніторингу

Організація дистанційного моніторингу відповідальною особою функціонального стану клієнта за допомогою мобільного додатку HealthyLungs.

Клієнт отримує мобільний пристрій з пульсоксиметром, який має бездротове з'єднання, і інтегрує його з GoogleFit на Android або Apple Health на iOS. Потім клієнт підключає браслет і мобільний додаток за допомогою технології Bluetooth. Після реєстрації мобільний додаток HealthyLungs починає передавати параметри клієнта на сервер для зберігання та обробки.

Додаток HealthyLungs реєструється на сервері та отримує доступ до даних клієнта. Диспетчер дистанційно бачить поточні дані та історію параметрів своїх клієнтів, а також повідомлення про критичний стан клієнта. Таку саму інформацію клієнт також може побачити у своїй заявці - це означає, що він може контролювати себе.

Основний функціонал медичної кіберфізичної системи HealthyLungs наведений на рисунках з зображенням графічного інтерфейсу мобільного додатку.

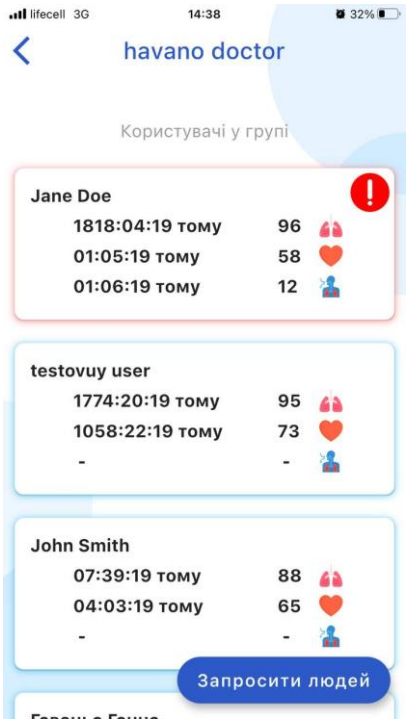

Рисунок 4.17. Список користувачів в групі. Відображення біжучих даних показників.

На рисунку 4.17 зображено список користувачів, які належать одній групі, тобто список осіб, за якими веде нагляд відповідальна особа. Також, в цьому списку зображено біжучі показники стану людини та час проведення їх вимірювання.

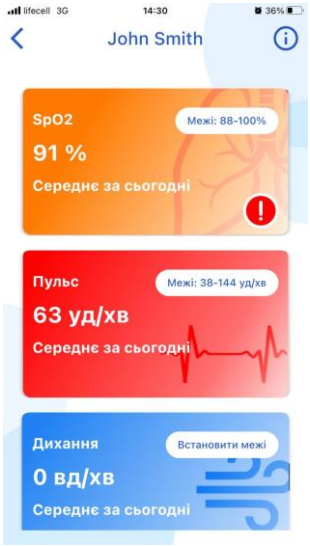

Рисунок 4.18. Відображення середніх значень показників за день.

На рисунку 4.18 зображено список всіх показників, а також середня значення кожного з них за поточний день. Символ «!» зображений в правому нижньому куті плитки показника символізує про події, які відповідальна особа ще не переглянула. Також показано задані межі для кожного показника, з можливістю їх редагування.

Рисунок 4.19. Відображення актуальних даних показника SpO2 за тиждень.

На рисунку 4.19 зображено сторінку з актуальними даними показника рівня кисню в крові. Дані візуалізовано у вигляді графіку, і можна переглядати дані за поточний день, тиждень, місяць.

В свою чергу на рисунку 4.20 зображено можливість перегляду архівних даних, а саме даних за попередні місяці, у вигляді графіку чи таблиці значень. Вибірку даних можна фільтрувати за часом та значенням.

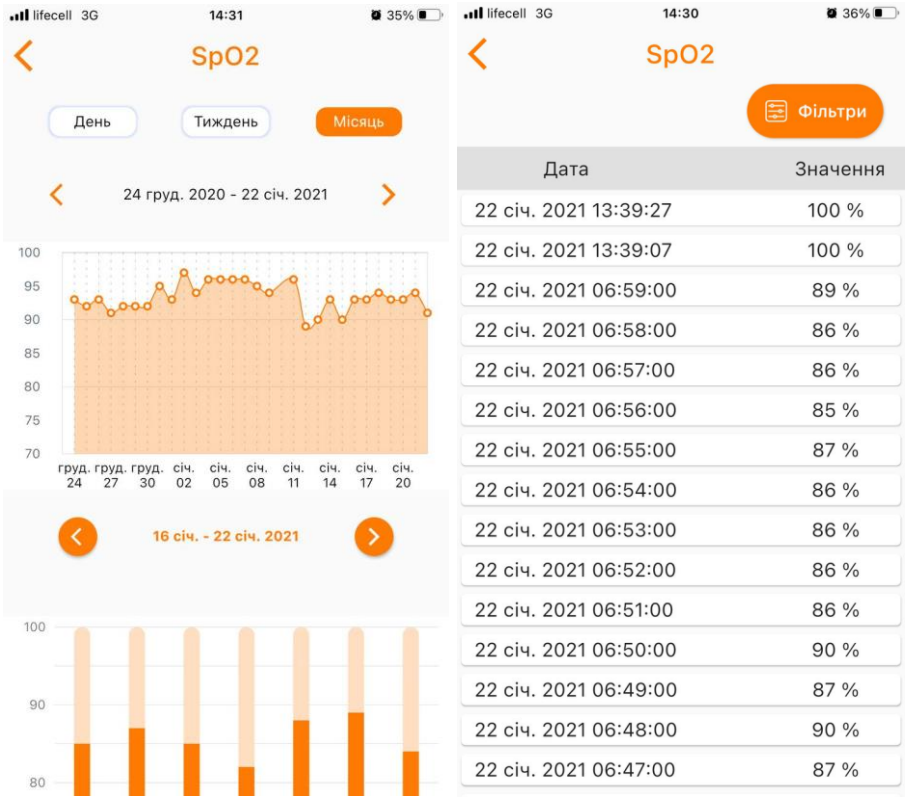

Рисунок 4.20. Відображення архівних даних показника SpO2 за місяць.

4.4. Висновки до розділу

1. Реалізовано серверне програмне забезпечення медичної кіберфізичної системи, яке базується на запропонованій архітектурно-функціональній моделі обробки інформації.
2. Реалізовано засіб автентифікації та авторизації користувачів за допомогою служби автентифікації SSO, використовуючи oAuth2.0 та OpenID, та стандарту JWT.

3. Реалізовано сервіс декларування фонових задач, за допомогою адаптеру черги Sidekiq.
4. Реалізовано засіб сповіщення про події, використовуючи 2 канали зв’язку: електронна пошта та Push-повідомлення. Згідно проведеному тестуванні, процес створення та сповіщення про подію в середньому виконується 2225мс, тобто 2,2 секунди.
5. Реалізовано засіб виявлення критичних показників, в основі якого лежить запропонований метод виявлення критичних показників. Згідно проведеному тестуванні, на виявлення критичних показників в середньому потрібно 1236,69мс, тобто 1,24 секунди.
6. Реалізовано засіб, який включає адаптовані алгоритми класифікації пацієнтів та діагностики захворювань. Реалізація полягає в виконанні квантової версії адаптованого алгоритму K-means. Цей алгоритм виконується на цифровому квантовому комп’ютері.
7. Реалізовано засоби обробки та агрегації виміряних показників. Проведено тестування, згідно яким було визначено середній час обробки вхідних даних, значення якого коливається від 149мс, тобто 0,15 секунди до 4200мс, тобто 4,2 секунди в залежності від їх об’єму. Також, проведено тестування часу виконання агрегації даних. Середній час виконання агрегації даних варіюється від 15мс, тобто 0,015 секунди до 678мс, тобто 0,68 секунди, в залежності від параметрів запиту даних.
8. Описано принципи організації зберігання даних в реалізованій медичній кіберфізичній системі, з використанням сховищ різного типу, в залежності від потреби засобів обробки інформації.
9. Описано принцип роботи медичної кіберфізичної системи «HealthyLungs», в якій впроваджувались запропоновані методи, засоби та модель обробки інформації.

ВИСНОВКИ

У дисертаційній роботі проведено теоретичне обґрунтування та нове вирішення актуальної наукової задачі розробки та дослідження методів та засобів оцінювання стану людини в медичних кіберфізичних системах. Ці методи та засоби передбачають паралельну обробку даних життєвих показників стану людини, організовуючи засоби обробки інформації в окремі незалежні логічні елементи в порівнянні з іншими існуючими медичними КФС. При цьому було отримано такі наукові та практичні результати:

1. Проведений аналіз відомих методів та засобів оцінювання стану людини в медичних кіберфізичних системах. Аналіз показав, що при проектуванні та реалізації медичних КФС, потрібно враховувати, що медичні дані є великими даними, а їх обробка повинна відбуватись паралельно.
2. Запропонувана архітектурно-інформаційна модель обробки інформації в медичних кіберфізичних системах для оцінювання стану людини. Для розмежування засобів обробки інформації на окремі незалежні логічні елементи запропонована модель обробки інформації в основі якої лежить мікросервісна архітектура з використанням API шлюзу.
3. Вдосконалений метод виявлення критичних показників в медичній кіберфізичній системі, який базується на одному з найбільш використовуваних посібників EWS та агрегованій оцінці життєвих показників стану людини.
4. Розроблені алгоритми діагностики захворювань в медичній кіберфізичній системі. Алгоритм класифікації пацієнтів - це адаптований алгоритми класифікації даних K-means. В свою чергу алгоритм діагностики захворювань - це адаптована версію алгоритму Association rule learning.
5. Розроблені нові засоби оцінювання стану організму людини в медичній кіберфізичній системі. Середній час виконання цих засобів обробки інформації наступний:

- створення та сповіщення про подію в середньому виконується 2225мс, тобто 2,2 секунди;
- виявлення критичних показників виконується 1236,69мс, тобто 1,24 секунди;
- процес обробки вхідних даних, значення якого коливається від 149мс, тобто 0,15 секунди до 4200мс, тобто 4,2 секунди в залежності від їх об’єму;
- виконання агрегації даних варіюється від 15мс, тобто 0,015 секунди до 678мс, тобто 0,68 секунди, в залежності від параметрів запиту даних.

6. Практично реалізована медична кіберфізична система оцінювання стану організму людини «HealthyLungs», в якій впроваджувались запропоновані методи, засоби та модель обробки інформації.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Stuard P. Lloyd. “Least squares quantization in PCM”. In: Information Theory, IEEE Transactions (1982), pp. 129–137.

2. Perry DE, Wolf AL. Foundations for the study of software architecture. ACM SIGSOFT Software engineering notes 1992; 17(4):40–52.

3. Kruchten PB. The 4+1 view model of architecture. IEEE Software 1995; 12(6):42-50.

4. Christoph Durr and Peter Hoyer. “A Quantum Algorithm for Finding the Minimum”. In: (1996). doi: arXiv:quant-ph/96070148.

5. L. K. Grover. “A fast quantum mechanical algorithm for database search”. In: ACM symposium on Theory of computing, ACM 28 (1996), pp. 212– 219.

6. Garlan D. Software architecture. Wiley Encyclopedia of Computer Science and Engineering 2001.

7. Bowers J, May J, Melander E, Baarman M, Ayoob A. Tailoring XP for large system mission critical software development. Extreme Programming and Agile Methods XP/Agile Universe 2002, Springer; 100–111.

8. Kitchenham B. Procedures for performing systematic reviews. Keele, UK, Keele University 2004; 33(2004):1-26.

9. WHO global report. Preventing chronic diseases: a vital investment, World Health Organization (2005)

10. Kruchten P, Obbink H,Stafford J. The past, present, and future for software architecture. IEEE Software 2006; 23(2):22-30.

11. E. Aımeur, G. Brassard, and S. Gambs. “Machine learning in a quantum world”. In: Advances in Artificial Intelligence, Springer (2006), pp. 431–442.

12. Apache Software Foundation; 2006 Режим доступу: https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html.

13. Brown A, McDermid J. The art and science of software architecture. Proceedings of First European Conference on Software Architecture, ECSA'07, Springer-Verlag Berlin Heidelberg 2007; volume 4758 of Lecture Notes in Computer Science: 237-256.

14. World Health Organization, Global age-friendly cities: a guide, 2007.
15. IEEE Std 1471-2000 (2007). ISO/IEC Standard for Systems and Software Engineering - Recommended Practice for Architectural Description of Software-Intensive Systems. ISO/IEC 42010 IEEE Std 1471-2000 First edition 2007-07-15, (pp. c1-24).
16. Blazona B, Koncar M. HL7 and DICOM based integration of radiology departments with healthcare enterprise information systems. International journal of medical informatics 2007; volume 76:S425-S432. DOI:https://doi.org/10.1016/j.ijmedinf.2007.05.001
17. Lee EA. Cyber Physical Systems: Design Challenges. 11th IEEE International Symposium on Object Oriented Real-Time Distributed Computing, isorc 2008, IEEE, 2008; 363-369.
18. Yick J, Mukherjee B, Ghosal D. Wireless sensor network survey. Computer Networks 2008; 52(12):2292-2330.
19. Dybå T, Dingsøyr T. Empirical studies of agile software development: A systematic review. Information and software technology 2008; 50(9):833–859
20. Feng X, Shen J, Fan Y. REST: An alternative to RPC for web services architecture. First International Conference on Future Information Networks, ICFIN Oct 2009; 7-10.
21. Erl T. SOA Design Patterns (1st ed.), Prentice Hall PTR, Upper Saddle River, NJ, USA, 2009
22. Corredor I, Martínez JF, Familiar MS. Bringing pervasive embedded networks to the service cloud: A lightweight middle1012 ware approach. Journal of Systems Architecture 2011; 57(10):916-933, ISSN 1383-7621. DOI:http://dx.doi.org/10.1016/j.sysarc.2011.04.005.
23. Méndez EO, Ren S. Design of cyber-physical interface for automated vital signs reading in electronic medical records systems. 2012 IEEE International Conference on Electro/Information Technology, Indianapolis, IN, 2012; 1-10. DOI:10.1109/EIT.2012.6220696.

24.Hu X, Wang L, Gao P, Dong X, Ji Z, You F, et al. Study on Disease Screening and Monitoring System Based on Wireless Communication and IOT, 2012 Spring Congress on Engineering and Technology, Xian, 2012; 1-6. DOI:10.1109/SCET.2012.6341961

25.Miorandi D, Sicari S, De Pellegrini F, Chlamtac I. Internet of things: Vision, applications and research challenges. Ad Hoc Networks 2012; 10(7):1497-1516.

26.Rao IH, Amir NA, Dagale H, Kuri J. e-SURAKSHAK: A Cyber-Physical Healthcare System with Service Oriented Archi1020 tecture. 2012 International Symposium on Electronic System Design, ISED 2012, IEEE; 177–182. DOI:10.1109/ISED.2012.66

27.Hardt D. The OAuth 2.0 Authorization Framework, IETF RFC 6749, October 2012

28.Bazzani M, Conzon D, Scalera A, Spirito MA, Trainito CI. Enabling the IoT paradigm in e-health solutions through the VIRTUS middleware. 2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications, TrustCom 2012; 1954-1959. DOI:10.1109/TrustCom.2012.144

29.Kramp T, van Kranenburg R, Lange S. Introduction to the Internet of Things. Enabling Things to Talk 2013, Springer Berlin Heidelberg; 1-10. DOI:10.1007/978-3-642-40403-0_1.

30.S. Lloyd, M. Mohseni, and P. Rebentrost. “Quantum algorithms for supervised and unsupervised machine learning”. In: arXiv preprint 1307.0411 (2013).

31.Rodrigues A, Silva JS, Boavida F. iSenior—A Support System for Elderly Citizens. IEEE Transactions on Emerging Topics in Computing 2013; 1(2):207-217. DOI:10.1109/TETC.2013.2294917

32.W. Fokkink, Distributed Algorithms: An Intuitive Approach, The MIT Press, Cambridge,Massachusetts, 2013.

33.Haque SA, Aziz, SM, Rahman M. Review of cyber-physical system in healthcare. International journal of distributed sensor networks 2014; 10(4):217415. DOI:10.1155/2014/217415

34. Corredor I, Metola E, Bernardos AM, Tarrío P, Casar JR. A lightweight Web of Things open platform to facilitate context data management and personalized healthcare services creation. International journal of environmental research and public health 2014; 11(5):4676-4713.

35. Mohammed J, Lung CH, Ocneanu A, Thakral A, Jones C, Adler A. Internet of Things: Remote Patient Monitoring Using Web Services and Cloud Computing. 2014 IEEE International Conference on Internet of Things (iThings), and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom), Taipei, 2014; 256-263. DOI:10.1109/iThings.2014.45

36. Ragavan SV, Haider K, Shanmugavel M. Healthcare Telematics Service Implementation Using OSGi Framework. Procedia Computer Science 2014; volume 42:168-174, ISSN 1877-0509. DOI:http://dx.doi.org/10.1016/j.procs.2014.11.048.

37. Cai H, Cui L., Shi, Y, Kong L, Yan, Z. Multi-tenant service composition based on granularity computing. 2014 IEEE International Conference on Services Computing, SCC 2014; 669-676.

38. Páez DG, Aparicio F, de Buenaga F, Ascanio J. Big Data and IoT for Chronic Patients Monitoring. International Conference on Ubiquitous Computing and Ambient Intelligence 2014; volume 8867:416–423. DOI:10.1007/978-3-319-13102-3_68

39. Fortino G, Parisi D, Pirrone V, Di Fatta G. BodyCloud: A SaaS approach for community Body Sensor Networks. Future Generation Computer Systems 2014; volume 35:62-79. DOI:http://dx.doi.org/10.1016/j.future.2013.12.015.

40. Maia P, Batista T, Cavalcante E, Baffa A, Delicato FC, Pire PF, Zomaya A. A Web Platform for Interconnecting Body Sensors and Improving Health Care. Procedia Computer Science 2014; volume 40:135-142. DOI:http://dx.doi.org/10.1016/j.procs.2014.10.041.

41. Sebestyen G, Hangan, A, Oniga S, Gal Z. eHealth solutions in the context of Internet of Things. 2014 IEEE International Conference on Automation, Quality and Testing, Robotics 2014; 1-6. DOI:10.1109/AQTR.2014.6857876
42. Lewis J, Fowler M.. Microservices, 2014 http://martinfowler.com/articles/microservices.html
43. Richardson I. Connected health: people, technology and processes, Lero-TR-2015-03. Lero Technical Report Series, University of Limerick, 2015.
44. Caporuscio M, Ghezzi C. Engineering future internet applications: The Prime approach. Journal of Systems and Software 2015; volume 106:9-27, ISSN 0164-1212. DOI:http://dx.doi.org/10.1016/j.jss.2015.03.102.
45. Lee EA. The past, present and future of cyber-physical systems: A focus on models. Sensors 2015; 15(3):4837-4869
46. Pepa L, Capecci M, Verdini F, Ceravolo MG, Spalazzi L. An architecture to manage motor disorders in Parkinson's disease. 2nd World Forum on Internet of Things (WF-IoT), Milan, 2015 IEEE; 615-620. DOI: 10.1109/WF-IoT.2015.7389124
47. Suciu G, Suciu V, Martian A, Craciunescu R, Vulpe A, Marcu I, et al. Big data, internet of things and cloud convergence–an architecture for secure e-health applications. Journal of medical systems 2015; 39(11):141.
48. Russinovich M. Microservices: An application revolution powered by the cloud, 2016. Режим доступу: https://azure.microsoft.com/en-us/blog/microservices-an-application-revolution-powered-by-the-cloud/
49. Carroll N. Key success factors for smart and connected health software solutions. Computer 2016; 49(11):22-28. DOI:10.1109/MC.2016.340.
50. Gudivada, D. Rao, V. Raghavan, Renaissance in database management: navigating thelandscape of candidate systems, IEEE Computer 49 (4) (April. 2016) 31-42.
51. Rathore MM, Ahmad A, Paul A, Wan J, Zhang D. Real-time Medical Emergency Response System: Exploiting IoT and Big Data for Public Health. Journal of medical systems 2016; 40(12):283.

52.Botta A, De Donato W, Persico V, Pescapé A. Integration of cloud computing and internet of things: a survey. Future Generation Computer Systems 2016; volume 56:684–700. DOI=http://dx.doi.org/10.1016/j.future.2015.09.021

53.Gravina R, Ma C, Pace P, Aloi G, Russo W, Li W, Fortino G. Cloud-based Activity-aaService cyber–physical framework for human activity monitoring in mobility. Future Generation Computer Systems 2017; volume 75. DOI:http://dx.doi.org/10.1016/j.future.2016.09.006.

54.Al-Taee Ma, Al-Nuaimy W, Muhsin ZJ, Al-Ataby A. Robot Assistant in Management of Diabetes in Children Based on the Internet of Things. IEEE Internet of Things Journal 2017; 4(2):437-445. DOI:10.1109/JIOT.2016.2623767

55.Zhang Y, Qiu M, Tsai CW, Hassan MM., Alamri A. Health-CPS: Healthcare cyber-physical system assisted by cloud and big data. IEEE Systems Journal 2017, 11(1):88-95. DOI:10.1109/JSYST.2015.2460747

56.M. Kleppmann, Designing Data-Intensive Applications: The Big Ideas Behind Reliable,Scalable, and Maintainable Systems, O'Reilly Media, Sebastopol, California, 2017.

57.Dawid Kopczyk. "Quantum machine learning for data scientists". In: (2018). doi: arXiv:1804.10068.

58.Havano B. Problems of privacy and security in cyber physical systems of intellectual houses / Bohdan Havano // Litteris et Artibus : proceedings, 23–25 November 2018 (10th International academic conference "Computer science & engineering 2018"), Lviv. — Lviv : Lviv Politechnic Publishing House, 2018. — P. 58–59.

59.Гаваньо Б. І. Проблеми конфіденційності та безпеки в кіберфізичних системах інтелектуальних будинків // Вісник Національного університету "Львівська політехніка". Серія: Комп'ютерні системи та мережі. 2018. № 905. С. 49–55.

60.Valerii Hlukhov, Bohdan Havano. FPGA-based Digital Quantum Coprocessor. Advances in Cyber-Physical Systems. Volume 3. Number 2. Lviv Polytechnic National University. 2018. p. 12–31.

61. Valerii Hlukhov, Bohdan Havano. Principles of Digital Quantum Coprocessor Based on a FPGA, which Operates under the Control of a Classical Computer // 2019 9th International Conference on Advanced Computer Information Technologies (ACIT). IEEE, 2019, p. 191-194
62. Havano Bohdan, Kytsun Hennadiy, Tkachyk Oleksandr. Web-server cross-site request forgery protection // Perspectives of science and education : proceedings of the 7th International youth conference, 10th May, 2020 New York, USA. – 2020. – С. 9–16
63. Melnyk Anatoliy, Morozov Yuriy, Havano Bohdan, Hupalo Petro. Investigation of wireless pulse oximeters for smartphone-based remote monitoring of lung health // Advances in Cyber-Physical Systems. – 2020. – Vol. 5, № 2. – p. 70–76.
64. А. О. Мельник, Ю. В. Морозов, Б. І. Гаваньо, П. А. Гупало. Біомедична кіберфізична система цілодобового моніторингу функцій легень у пацієнтів із COVID-19 // Комп'ютерні системи та мережі. — Львів : Видавництво Львівської політехніки, 2020. — Том 2. — № 1. — С. 1–5.
65. А. О. Мельник, Ю. В. Морозов, Б. І. Гаваньо, П. А. Гупало. Мобільні додатки для цілодобового віддаленого моніторингу лікарями функцій легень пацієнтів // Сучасні проблеми і досягнення в галузі радіотехніки, телекомунікацій та інформаційних технологій : тези доповідей X Міжнародної науково-практичної конференції, 07–09 жовтня 2020 р., м. Запоріжжя. – 2020. – С. 83–84
66. Anatoliy Melnyk, Yurii Morozov, Bohdan Havano, Petro Hupalo HealthSupervisor: Mobile Application for Round-the-Clock Remote Monitoring of the Human Functional State // The 2nd International Workshop on Intelligent Information Technologies & Systems of Information Security (IntelITSIS-2021). CEUR-WS, 2021, Vol-2853, p. 24-37
67. Penagos SP, Salazar LD, Vera F, Control de signos vitales.
68. Chapter 3: Architectural Patterns and Styles. Режим доступу: https://msdn.microsoft.com/en-us/library/ee658117.aspx

69. Мельник А. О. Багаторівнева базова платформа кіберфізичних систем / А. О. Мельник // Кіберфізичні системи досягнення та виклики : матеріали I Наукового семінару, 25–26 червня 2015 року, Львів / Національний університет «Львівська політехніка». – Львів : НВФ «Українські технології», 2015. – С. 5–15

70. Imperva, Top 5 Database Security Threats, 2016. Режим доступу: https://www.imperva.com/docs/gated/WP_Top_5_Database_Security_Threats.pdf

71. National Early Warning Score (NEWS). Режим доступу: https://www.mdcalc.com/national-early-warning-score-news

72. Business logic. Режим доступу: https://en.wikipedia.org/wiki/Business_logic.

73. Стандарт rfc7519. Режим доступу: https://datatracker.ietf.org/doc/html/rfc7519.

74. APACHE HIVE TM. Режим доступу: https://hive.apache.org.

75. ANSI standarts. Режим доступу: https://docs.oracle.com/database/121/SQLRF/ap_standard_sql001.htm#SQLRF55514.

76. API Documentation & Design Tools for Teams | Swagger. Режим доступу: https://swagger.io/

77. The Apache Hadoop. Режим доступу: https://hadoop.apache.org/.

ДОДАТОК А. СПИСОК ПУБЛІКАЦІЙ ЗА ТЕМОЮ ДИСЕРТАЦІЇ

1. Melnyk Anatoliy, Morozov Yuriy, Havano Bohdan, Hupalo Petro. Investigation of wireless pulse oximeters for smartphone-based remote monitoring of lung health // Advances in Cyber-Physical Systems. – 2020. – Vol. 5, № 2. – p. 70–76.

2. А. О. Мельник, Ю. В. Морозов, Б. І. Гаваньо, П. А. Гупало. Біомедична кіберфізична система цілодобового моніторингу функцій легень у пацієнтів із COVID-19 // Комп'ютерні системи та мережі. — Львів : Видавництво Львівської політехніки, 2020. — Том 2. — № 1. — С. 1–5.

3. Anatoliy Melnyk, Yurii Morozov, Bohdan Havano, Petro Hupalo HealthSupervisor: Mobile Application for Round-the-Clock Remote Monitoring of the Human Functional State // The 2nd International Workshop on Intelligent Information Technologies & Systems of Information Security (IntelITSIS-2021). CEUR-WS, 2021, Vol-2853, p. 24-37

4. Гаваньо Б. І. Assessing the Human Condition in Medical Cyber-Physical System Based on Microservices Architecture // Досягнення у кіберфізичних системах. – 2021. – Vol. 6, № 2. – С. 112–120

Наукові праці, які засвідчують апробацію матеріалів дисертації:

5. А. О. Мельник, Ю. В. Морозов, Б. І. Гаваньо, П. А. Гупало. Мобільні додатки для цілодобового віддаленого моніторингу лікарями функцій легень пацієнтів // Сучасні проблеми і досягнення в галузі радіотехніки, телекомунікацій та інформаційних технологій : тези доповідей X Міжнародної науково-практичної конференції, 07–09 жовтня 2020 р., м. Запоріжжя. – 2020. – С. 83–84

6. Havano B. Problems of privacy and security in cyber physical systems of intellectual houses / Bohdan Havano // Litteris et Artibus : proceedings, 23–25 November 2018 (10th International academic conference "Computer science & engineering 2018"), Lviv. — Lviv : Lviv Politechnic Publishing House, 2018. — P. 58–59.

Наукові праці, які додатково відображають наукові результати дисертації:

7. Havano Bohdan, Kytsun Hennadiy, Tkachyk Oleksandr. Web-server cross-site request forgery protection // Perspectives of science and education : proceedings of the 7th International youth conference, 10th May, 2020 New York, USA. – 2020. – С. 9–16

8. Гаваньо Б. І. Проблеми конфіденційності та безпеки в кіберфізичних системах інтелектуальних будинків // Вісник Національного університету “Львівська політехніка”. Серія: Комп’ютерні системи та мережі. 2018. № 905. С. 49–55.

9. Valerii Hlukhov, Bohdan Havano. FPGA-based Digital Quantum Coprocessor. Advances in Cyber-Physical Systems. Volume 3. Number 2. Lviv Polytechnic National University. 2018. p. 12–31.

10. Valerii Hlukhov, Bohdan Havano. Principles of Digital Quantum Coprocessor Based on a FPGA, which Operates under the Control of a Classical Computer // 2019 9th International Conference on Advanced Computer Information Technologies (ACIT). IEEE, 2019, p. 191-194

11. Патент України на корисну модель №148157, «Біомедична система для цілодобового віддаленого моніторингу уповноваженою особою показників функціонального стану організму клієнта», заявка №u202008053 від 16.12.2020, А. О. Мельник, Ю. В. Морозов, Б. І. Гаваньо, П. А. Гупало.

ДОДАТОК Б. АКТ ВПРОВАДЖЕННЯ РЕЗУЛЬТАТІВ ДИСЕРТАЦІЙНОГО ДОСЛІДЖЕННЯ

«ЗАТВЕРДЖУЮ»
Проректор з наукової роботи
Національного університету
"Львівська політехніка"
_____ І.В. Демидов
_____ 2021 р.

АКТ

про впровадження результатів дисертаційного дослідження аспіранта кафедри ЕОМ Гаваньо Богдана Івановича на тему: «Методи та засоби оцінювання стану людини в медичних кіберфізичних системах», представленої на здобуття ступеня доктора філософії за спеціальністю 123 «Комп'ютерна інженерія» при виконанні наукового проєкту за грантом Агенції університетів Франкофонії «Мобільні додатки для цілодобового моніторингу функцій легень у пацієнтів із COVID-19»

Комісія у складі голови – начальника НДЧ д.т.н., ст.досл. Небесного Р.В. та членів: завідувача кафедри електронних обчислювальних машин д.т.н., професора Мельника А.О., доцента кафедри електронних обчислювальних машин, к.т.н. Морозова Ю.В. цим актом підтверджують, що результати дисертаційного дослідження Гаваньо Б.І. щодо методів та засобів оцінювання стану людини в медичних кіберфізичних системах використані при виконанні наукового проєкту за грантом Агенції університетів Франкофонії «Мобільні додатки для цілодобового моніторингу функцій легень у пацієнтів із COVID-19».

Зокрема, Гаваньо Б.І. апробував запропоновані ним методи побудови медичної кіберфізичної системи для цілодобового віддаленого моніторингу показників функціонального стану організму людини, методи реалізації засобів цілодобового віддаленого моніторингу показників функціонального стану організму людини, а також спосіб цілодобового віддаленого моніторингу уповноваженою особою показників функціонального стану організму людини.

Запропоновані Гаваньо Б.І. методи та способи можуть використовуватись при побудові медичних кіберфізичних систем.

Голова комісії:

Начальник НДЧ
д.т.н., ст.досл. Р.В. Небесний

Члени комісії:

Зав. кафедри ЕОМ, д.т.н., проф. А.О. Мельник

К.т.н., доцент, доцент кафедри ЕОМ Ю.В. Морозов