Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра інформатики факультету інформатики

Розробка архітектури високонавантажених сервісів у сфері HEALTHCARE

Текстова частина магістерської роботи
за спеціальністю „Інженерія програмного забезпечення” 121

Керівник магістерської роботи
д.т.н. А.М. Глибовець

_____
(підпис)

“____” _____ 2021 р.

Виконав студент Д. О. Мирошник
“____” _____ 2021 р.

Київ 2021

Тема:Розробка архітектури високонавантажених сервісів у сфері HEALTHCARE

Календарний план виконання роботи:

№ п/п	Назва етапу дипломного проекту (роботи)	Термін виконання етапу	Примітка
1.	Отримання завдання на дипломну роботу	01.11.2021	
2.	Огляд технічної літератури за темою роботи	15.11.2021	
3.	Виконання аналізу сучасних рішень	22.02.2022	
4.	Розробка архітектури високонавантажених сервісів у сфері HEALTHCARE	27.04.2022	
5.	Написання пояснювальної записки	08.06.2022	

Студент _____

Керівник _____

“____” _____

Керівник курсової роботи

ЗМІСТ

Анотація 4
Вступ 5
Розділ 1 Архітектура високонавантажених сервісів та порівняння існуючих аналогів 7
1.1 Класифікація високого навантаження 7
1.2 Огляд проблем з високим навантаженням 8
1.3 Огляд існуючих рішень 11
Розділ 2 Розробка архітектури системи 15
2.1 Вимоги до системи 15
2.2 Вимоги до Архітектури 15
2.3 Аналіз та порівняння інструментів 16
2.4 Розробка Архітектури 19
2.5 Альтернативна архітектура 21
Розділ 3 Обробка та аналіз медичних даних 23
3.1 Критерії аналізу даних 23
3.2. Огляд метаданих 24
Висновки по роботі 27
Список літератури 28

Анотація

У даній роботі було проведено огляд та аналіз найпоширеніших застосунків, що допомагають користувачам відслідковувати персональні медичні дані, да дані про здоров'я. Було розглянуто основні проблеми, визначення терміну високого навантаження, та що потребується для реалізації високонавантаженої системи.

Було сформульовано вимоги до нашої системи та архітектури. Також були спроектовані основна та альтернативна архітектура для реалізації застосунку. Були обрані та порівняні інструменти для реалізації цієї архітектури.

Розглянуто формування аналітики на базі отриманих даних, та приведено приклад реалізації для обробки отриманих даних.

Ключові слова: Azure, Cosmos DB, Python, трекер, Apple Health, архітектура високонавантажених сервісів у сфері Healthcare, візуалізація.

Вступ

Більшість з нас завжди мають при собі різноманітні пристрої для відстеження та запису різноманітної інформації. Смартфони, якими ми користуємось, це мультифункціональні пристрої які можуть працювати як трекери. Одне з найнадійніших сховищ про здоров'я людини є на наших смартфонах, носимих пристроях і трекерах активності. Використовуючи кілька датчиків, наші телефони та носимі пристрої можуть інтерпретувати наші моделі руху та повідомляти нам, скільки кроків протягом дня ми зробили, скільки сходів ми піднялися, навіть нерівномірність у ходьбі. Якщо ви використовуєте пристрій для носіння з датчиком частоти серцевих скорочень, ви також можете фіксувати частоту серцевих скорочень у стані спокою, активності та сну та навіть знати, скільки часу ви спали.

Існують різні способи та причини, чому люди відстежують свої життєві показники: через проблеми зі здоров'ям, або ж коли людина постійно займається спортом. Коли справа доходить до запису їх щоденних рухів, найпоширенішим методом відстеження життєвих показників, є фітнес-трекер активності або розумний годинник. Згідно з інфографікою Statista , сьогодні найбільш використовуваними пристроями для носіння є Fitbit, Apple Watch, Garmin, Mi-Band від XiaoMi та Samsung [1]. Цікаво, що існують десятки інших пристроїв із значно меншою часткою ринку, але які пропонують додатковий набір датчиків для відстеження інших точок даних, таких як кров'яний тиск та рівень кисню у крові.

Смартфон користувачів компанії Apple, відстежує ваші кроки та безліч інших показників здоров'я. Деякі записуються безпосередньо телефоном. Інші реєструються через інші програми охорони здоров'я, які зберігають їхні дані в сховищі Apple Health. Якщо користувач регулярно носить фітнес трекер вдень, під час тренувань і вночі, то у вас буде ще більше даних, як-от частота серцевих скорочень і можливість трекінгу сну.

У цій курсовій роботі я поставив перед собою задачу розробки забезпечення для роботи з медичними даними які надходять у стандартний застосунок Apple Health, та подальша його обробка, для демонстрації важливих показників вашому лікарю, що може допомогти у якнайшвидшому вирішенні проблеми при зверненні до лікаря. Також система буде оснащена можливістю авторизації, контролю документів та запису до лікаря.

Розділ 1 Архітектура високонавантажених сервісів та порівняння існуючих аналогів

Високонавантажені сервіси включають в себе різноманітну кількість архітектурних рішень, які задовольняють вимогам користувачів, для створення конкурентоздатного застосунку. Саме тому, необхідно розглянути що саме включає в себе поняття високонавантажений сервіс.

1.1 Класифікація високого навантаження

Складно надати чітке визначення високому навантаженню. Її не можна виміряти кількістю запитів, які надходять на сервер, або швидкістю веб-сайту. Адже не існує «середньої» кількості запитів, як і «середнього» сайту. Один веб-ресурс зможе обробляти тисячу запитів в секунду без особливих труднощів, а інший вийде з падінням на сотому з'єднанні. Тож справа тут зовсім не в кількісних показниках.

Високонавантажені системи, це системи які масштабуються, тобто при додаванні ресурсів, система збільшувала свою продуктивність.

Високонавантажений сервіс – це сервіс який може впоратись з великою кількістю запитів, запису у бази даних, та можливість працювати з великим обсягом даних. Це налаштування архітектури сайту: робота з базами даних, сервером, використанням сучасних технологій та мов програмування.

Важливо розуміти, що у сфері LifeCare, система має бути масштабованою, адже основною проблемою клінічних даних є не тільки його обсяг що постійно збільшується та зростає, а і те що ці данні зберігаються у різних форматах та відокремлених базах даних, та які можуть бути не у відповідних стандартах зберігання даних, та не структуровані. Взагалі у світі, відносно проблеми клінічних даних, була виділена проблема "V"(від назви на англійській мові): Об'єм "volume", різноманітність "variety" швидкість "velocity" та достовірність "veracity". [3]

- Об’єм: обсяг даних відносно кожного пацієнта зростає щодня. Дані відносно кожної людини накопичуються і з’являється проблема зберігання цього обсягу даних.
- Різноманітність: Дані можуть надходити з різних джерел та можуть зберігатись у різних форматах.
- Швидкість: більшість даних необхідно аналізувати та обробляти в реальному часі . Саме тому архітектура для запису великих потоків даних повинна підтримуватись у реальному часі. Другою ж проблемою є актуальність цих даних. Застарілі данні мають видалятись або є замінятись новою або ж біль актуальною інформацією.
- Достовірність: Великі дані через свою складність можуть містити невідповідності або ж деякі дані можуть бути відсутніми.

Важливо розуміти, що масштабування будь-якої програми – складається з трьох кроків:

- Аналіз навантаження, задля визначення зон найбільшого впливу на систему
- Перенесення зон найбільшого впливу на окремі вузли та їх оптимізація системи
- Повторний аналіз навантаження

1.2 Огляд проблем з високим навантаженням

Інфраструктура з високим навантаженням обробляє великі обсяги даних і, таким чином, створює велику цінність для бізнесу. Основним джерелом проблем в інфраструктурі з високим навантаженням є обсяг даних, складність і швидкість змін. Тому важливо, щоб загальна архітектура великого завантаження великої програми була розроблена як з точки зору програмних компонентів, так і апаратного забезпечення, на якому вони працюють. Більше того, розробляючи індивідуальну систему високого навантаження, необхідно

чітко розуміти, які терміни доставки встановлені, які законодавчі обмеження, який досвід мають фахівці, які займаються проектуванням

При створенні високонавантажених систем важливо враховувати ряд принципів.

- Доступність - час безперебійної роботи безпосередньо пов’язаний з репутацією та продуктивністю багатьох компаній.
- Продуктивність - швидкість веб-ресурсу впливає на задоволеність користувачів сервісом, а також на рейтинг у результатах пошуку (що відбивається на трафіку).
- Надійність - запит завжди повинен повертати користувачам одні й ті ж дані, щоб користувачі були впевнені, що якщо якісь дані будуть записані/введені в систему, під час подальшого вилучення ви можете розраховувати на їх незмінність і безпеку.
- Масштабованість - ми можемо говорити про різні параметри системи: скільки додаткового трафіку вона може обробляти, наскільки легко збільшити ємність сховища, скільки транзакцій можна обробити понад поточні можливості.
- Керованість - Розробка спеціальної системи високого навантаження, яка проста в експлуатації, надзвичайно важлива на пізніх етапах розробки проекту (проста діагностика та розуміння суті проблем, коли вони виникають, легке оновлення або модифікації).
- Вартість - Включає витрати на апаратне та програмне забезпечення. Важливо враховувати й інші аспекти, необхідні для розгортання та обслуговування системи: кількість часу, витраченого розробниками на збірку системи, кількість зусиль, необхідних для запуску системи, навчання, навчання персоналу тощо.

Основні проблеми при розробці користувацьких систем високого навантаження виникають у таких сегментах: обсяги даних, поширення даних,

корекція даних, використання програмного забезпечення з відкритим кодом, пошук, обробка й аналіз даних та моделювання інформації.

Важливо також розуміти що розробники повинні приділяти більше уваги надійності систем з високим навантаженням. Надійність означає здатність системи продовжувати нормально працювати навіть у разі виникнення проблеми.

Говорячи про надійність високонавантажених систем, необхідно згадати документацію з управління несправністю. Добре написана документація з управління збоями повинна включати простий покроковий посібник із відновлення системи практично від будь-якого можливого збою.

Коли справа доходить до великих центрів обробки даних, відомо, що збої обладнання (будь то відключення електроенергії, збій жорстких дисків або оперативної пам’яті) трапляються постійно. Одним із способів вирішення проблеми є створення неспільної архітектури високого навантаження. Завдяки такій архітектурі відсутній центральний сервер, який контролює та координує дії інших вузлів, і, відповідно, кожен вузол системи може працювати незалежно один від одного. Ці системи не мають єдиної точки відмови, тому вони набагато стійкіші до збоїв. Іншим методом запобігання збоям є збільшення резервування окремих компонентів системи для зниження частоти збоїв (резервний блок живлення, RAID — резервний масив дисків тощо). Коли один із компонентів виходить з ладу, його функціональність переходить на запасний компонент. Таким чином, неможливо повністю уникнути збою, Якщо говорити про глобальну надлишковість, то всі правила взаємодії між серверами поширюються і на дата-центри - ми повинні мати запас міцності (потужність, обчислювальна потужність і т. д.), щоб продовжити роботу з втратою одного дата-центру. без істотної шкоди для якісних послуг, що надаються. Для забезпечення надійності системи рекомендується використовувати такі підходи:

- Відокремлення частин системи, які впливають на продуктивність системи, від частин, найбільш схильних до людських помилок.
- Впровадження всіх форм тестування, будь то модульне тестування, комплексне системне тестування або ручні тести.
- Забезпечення інструментами для якнайшвидшого відновлення системи у разі збою, щоб мінімізувати вплив.
- Впровадження системи метрик, моніторингу та ведення журналів як інструментів для діагностики помилок та причин збоїв.

Програма масштабується так само ефективно, як і її найслабший компонент, тому ми повинні пам’ятати про визначення таких місць.

1.3 Огляд існуючих рішень

Наразі всі технологічні гіганти, активно розробляють та впроваджують все нові функції у своїх застосунках для здоров’я. Кожен смартфон оснащений мінімальним набором функцій які він відслідковує автоматично, та записує у відповідну до операційної системи програму. Основною ціллю цих програм, та великі інвестиції у цю сферу, викликані власним бажанням користувачів знати свої показники, трекінг яких не вимагає ніяких зусиль з боку користувача.

Всі інтегровані у операційну систему застосунки здоров’я, автоматично записуються свої, та заповнюються даними зі сторонніх застосунків, якщо ви користуєтесь додатково фітнес трекером або будь яким девайсом для контролю вашого здоров’я.

Рис.1.1 Вигляд даних у інтегрованому застосунку Apple Health

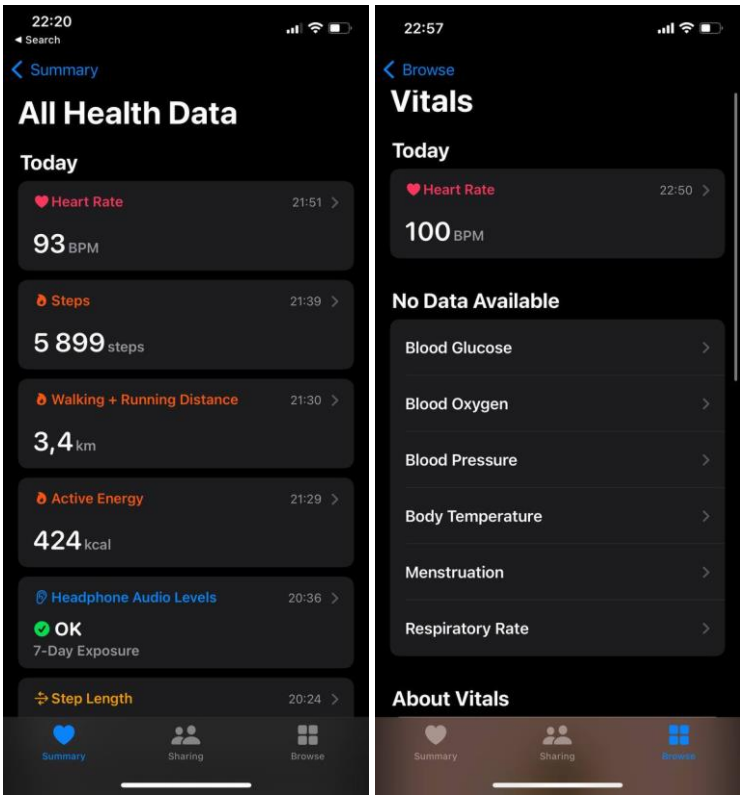


Рис.1.2 Загальний вигляд даних та додаткові функції відслідковування Apple Health

Як можна побачити на рис.1.1 Дані збережені у Apple Health, містять у собі дані як власні, в данному випадку це кількість кроків та сон, так і завантаженні - сердцевий ритм. Ці дані автоматично були взяті з завантаженого мною застосунку Zepp Life який візуалізає всі дані отримані зі смарт годинника Mi band, вызуалызацію даних якого буде розглянуто нижче. На рис 1.2 Можна побачити узагальнену інформацію по користувачу та подивитись окремо кожен з параметрів, та додатково показано що застосунок може зберігати дані з додаткових пристроїв для відстеження здоров'я користувача таких як: кров'яний тиск, кисень у крові, рівень глюкози та багато інших. Це означає що наша система може бути оснащена додатковими параметрами для відстеження стану здоров'я пацієнта.

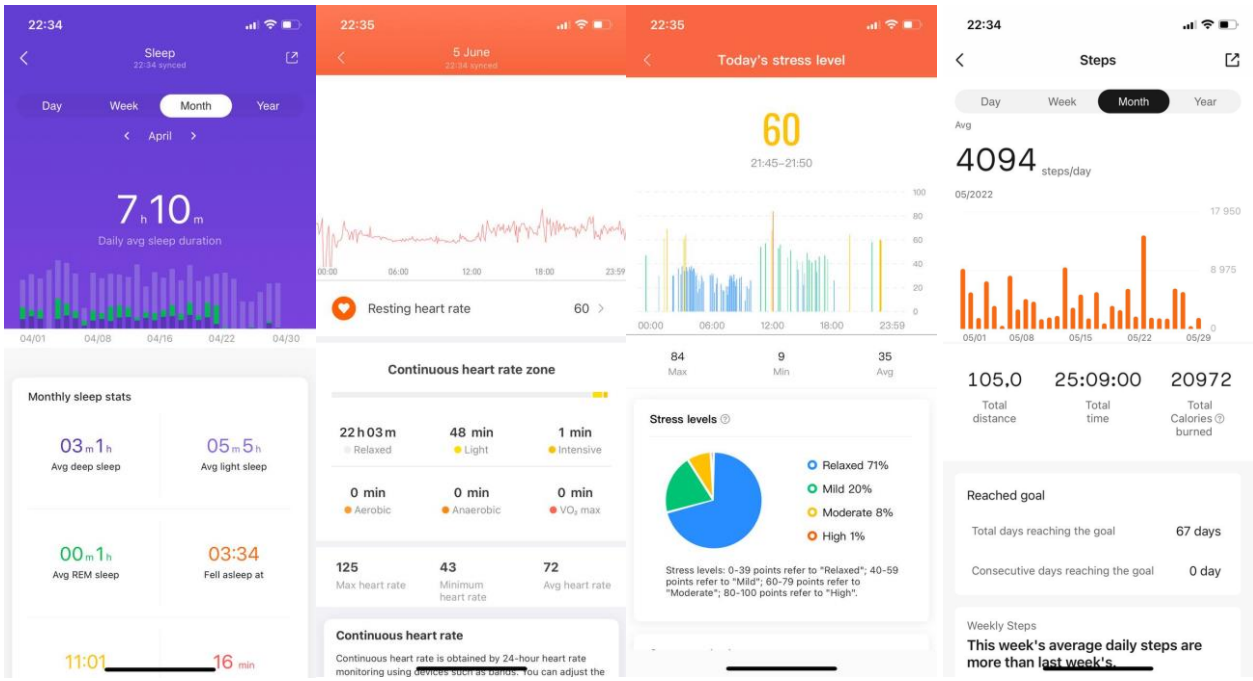


Рис. 1.3 проаналізовані дані зі смарт годинника

Як можна побачити на рис 1.3 дані які записував та надсилав годинник до встановленого застосунку. Ці дані мають більш ґрунтовну інформацію як

наприклад про фази сну, коли людина прокинулась, встала з ліжка. Але неймовірною перевагою є те що ці дані досить точні. З похибкою до кількох хвилин воно зберігає та візуалізує вашу активність протягом дня та сну. На рис.1.3 зображені дані які було зібрано моїм годинником про сон, серцевий ритм, стрес, та активність протягом дня.

Розділ 2 Розробка архітектури системи

Необхідно визначити вимоги до системи які будуть враховуватись при розробці архітектури.

2.1 Вимоги до системи

Необхідно визначити вимоги до системи які будуть враховуватись при розробці архітектури:

- Система має збирати дані з інтегрованого застосунку Apple Health, обробляти зібрані дані, та зберігати їх у сховищі
- Аналіз та формування графіків з критичними значеннями, які надають вичерпну інформацію про стан здоров’я пацієнта, що може допомоги в розумінні лікарем причини хвороби, або ж для відстеження стану здоров’я пацієнта
- Надання актуальної інформації про лікарів які необхідні користувачу, графік роботи лікарів та можливість запису та оновлення графіку роботи лікарів.

2.2 Вимоги до Архітектури

- Періодичне видалення застарілих або ж неактуальних даних зі сховища, задля забезпечення швидкості роботи системи. Як було розглянуто у першому розділі об’єм медичних даних є великою проблемою, дані про кожного користувача будуть збільшуватись щоденно. Тому необхідно робити періодичну фільтрацію збережених даних у сховищі.
- Розширюваність системи, завдяки якій можна буде додавати та аналізувати нові показники
- Доступність системи у будь-який час
- Дані які надають користувачі мають бути надійно захищені від передачі їх третім особам, або ж в інші застосунки. Ця вимога є

надзвичайно важливою, при роботі з персональними даними користувача, так як медичні дані є досить непублічною інформацією.

2.3 Аналіз та порівняння інструментів

Хмарне середовище. Найкращим рішенням буде побудувати систему на базі хмарного провайдера Microsoft Azure, так як наразі послуги цього провайдера постійно вдосконалюються та оновлюються. Також, це додатково забезпечить цю систему додатковою безпекою, так як більшість сервісів містять засоби безпеки за замовчуванням, та додатково містять послуги які дозволяють посилити безпеку рішень.

Мобільний застосунок. Потреба в постійному оновленні даних про запис до лікаря або ж медичних даних, створює необхідність мобільного застосунку. Єдиним рішенням є обрання мови програмування SWIFT, так як застосунок буде розроблюватись на операційній системі IOS.

Front-end . В рамках цієї системи необхідно розробити достатньо простий web застосунок який зможе відтворювати результати роботи наших сервісів. Тому я пропоную React, який є бібліотекою JavaScript із відкритим вихідним кодом для створення інтерфейсів користувача на основі компонентів UI.

Бази даних. Система буде складатись з двох баз даних: реляційної та нереляційної бази даних. Необхідність використання двох різних баз даних виникає через необхідність масштабування системи, та простоту збереження медичних даних. Нереляційні бази даних легше масштабуються аніж реляційні, та потребують менших затрат як людських, так і апаратних.

Реляційна база даних зберігатиме у собі дані записів до лікаря, персональні дані користувачів, дані про лікарів та їх розклад. В якості реляційної бази даних була обрана PostgreSQL через те що вона є досить гнучкою базою даних яка може працювати з великим об’ємом даних.

Нереляційне сховище даних необхідне для збереження медичних даних, та результатів відвідування лікаря. Необхідно щоб база даних могла обробляти великий обсяг даних і могла масштабуватися як по вертикалі, так і по горизонталі. Для порівняння було обрано 2 популярні нереляційні бази даних : Microsoft Azure Cosmos DB та MongoDB.

Таблиця 1.1. Порівняння баз даних

Порівняння	Microsoft Azure Cosmos DB	MongoDB
Опис	Горизонтально масштабований та багатомодульний сервіс бази даних	Доступний як повністю керований хмарний сервіс, так і для розгортанні в інфраструктурі що самостійно керує
Ліцензія	Комерційний	Відкритий код
Шардинг	Так	Так
На основі хмари	Так	Ні
Концепції користувача	Права доступу можуть бути визначені аж до рівня елемента	Права доступу для користувачів і ролей
API	DocumentDB API Graph API (Gremlin) MongoDB API RESTful HTTP API Table API	Власний протокол із використанням JSON

Також нижче приведене порівняння

Для 1 КБ корисного навантаження: Azure Cosmos DB працює краще для операцій створення, оновлення та видалення, тоді як MongoDB працює добре для операцій читання.

Для 10 КБ корисного навантаження: Azure Cosmos DB працює краще для операцій створення, оновлення та видалення, тоді як MongoDB добре справляється з операціями читання.

Для 100 КБ корисного навантаження: тут Azure Cosmos DB працює краще для операцій створення та читання, тоді як MongoDB — для операцій оновлення та видалення.

Для 1 МБ корисного навантаження: у міру збільшення розміру корисного навантаження Azure Cosmos DB працює краще для операцій читання, тоді як MongoDB перевищує кількість операцій створення, оновлення та видалення.

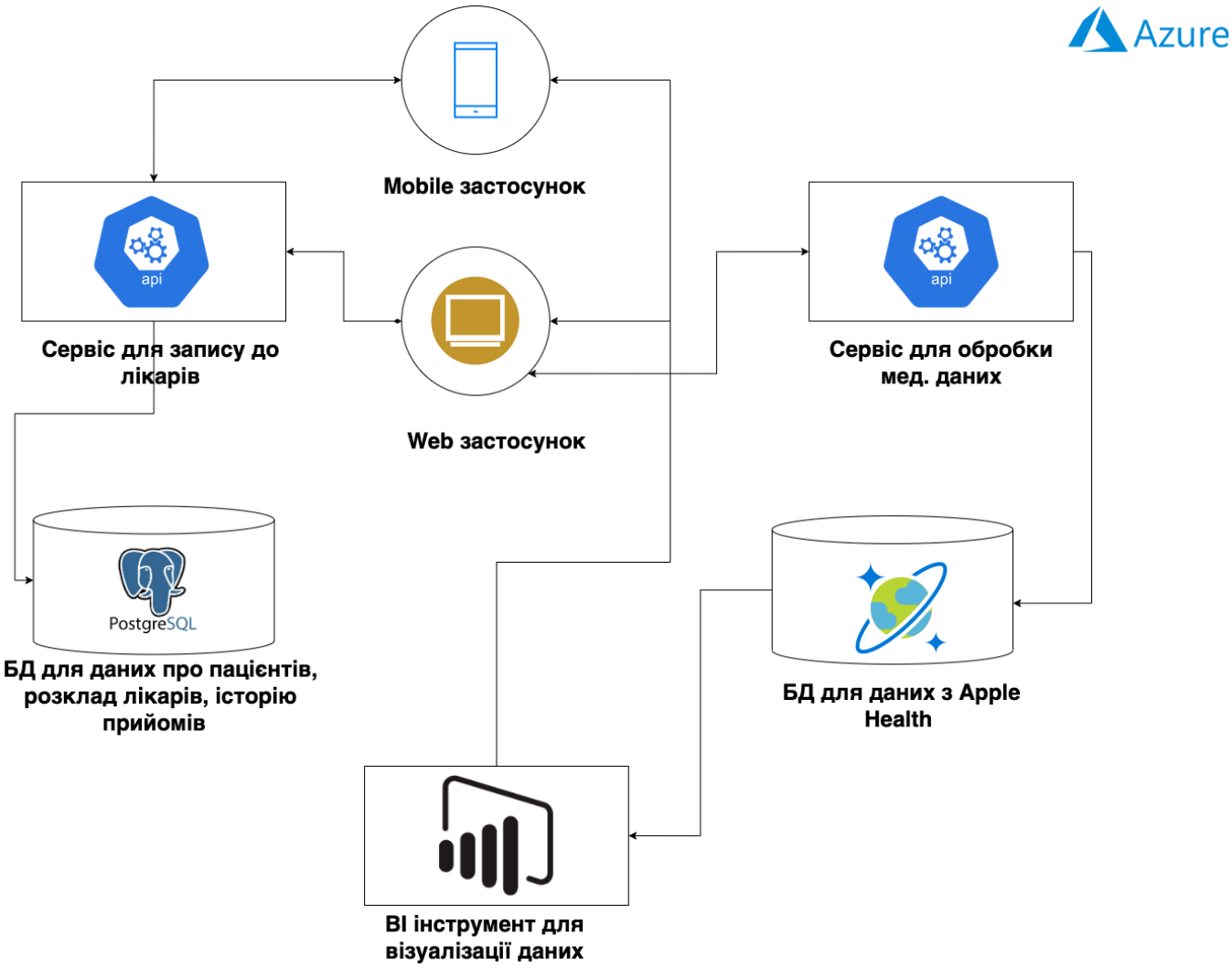
Для корисного навантаження 2 МБ: MongoDB повністю перевершує Azure Cosmos DB у цьому сценарії.

Виходячи з того що ми використовуємо хмарний провайдер Microsoft Azure, варто обрати саме Microsoft Azure Cosmos DB, але за вподоби, можна обрати MongoDB так як було виявлено, що при роботі з обсягом даних 2 мб та вище MongoDB впорається краще. [9]

Візуалізація. Візуалізація даних важливим показником системи. Оброблені та збережені дані необхідно візуалізувати для створення щоденної аналітики відносно пацієнта. Для візуалізації пропонується використати Azure data Explorer який дає змогу візуалізувати великі об’єми даних.

API. Python є відмінним варіантом, якщо вашому API потрібно інтегруватися з кількома базами даних, обробляти складні та численні міграції схем, користуватися можливостями адміністратора або надавати кінцеві точки виявлення.

2.4 Розробка Архітектури



Система складається з 7 компонент:

- Мобільний застосунок, за допомогою якого користувач матиме змогу записатись до лікаря, переглянути інформацію про свій стан здоров’я, переглянути висновок до попереднього відвідування лікаря. Також мобільний застосунок необхідний для витягування, та надсилання даних до наших баз даних.
- WEB застосунок, за допомогою якого лікар зможе отримати актуальну інформацію про стан здоров’я пацієнта, відслідковувати динаміку показників, переглядати записи до нього.

- Сервіс запису який необхідний для того щоб надсилати користувачам актуальні дані про наявність вільного часу у лікаря, та подальшого запису до нього.
- Сервіс для запису медичних даних, обробляє дані отримані від мобільного застосунку користувача з програми health на телефоні користувача
- реляційна база даних необхідна для збереження персональних даних користувачів та лікарів, а також, зберігатиме у собі дані про відвідування та розклад роботи лікаря.
- нереляційна база даних виконує роль збереження оброблених даних за допомогою сервісу запису медичних даних. Також, додатково у цій базі даних будуть зберігатися дані про відвідування пацієнтом лікаря, час та діагноз
- візуалізація даних – це служба яка працює для створення аналітичних рішень, відносно отриманих медичних даних, збережених у нереляційній базі даних.

Мобільний застосунок буде напряму пов’язаний з обома сервісами, кожен з яких відповідає за свою частину обробки інформації. Мобільний застосунок дозволяє користувачу переглянути поточні дані (медичні або персональні), які він вносив у застосунок. В мобільному застосунку користувач може: зареєструватись або увійти у існуючий обліковий запис, обрати лікаря який йому необхідний, записатись на прийом до лікаря та переглянути розклад роботи закладу і лікаря, ці дані зберігатимуться на реляційні базі даних.

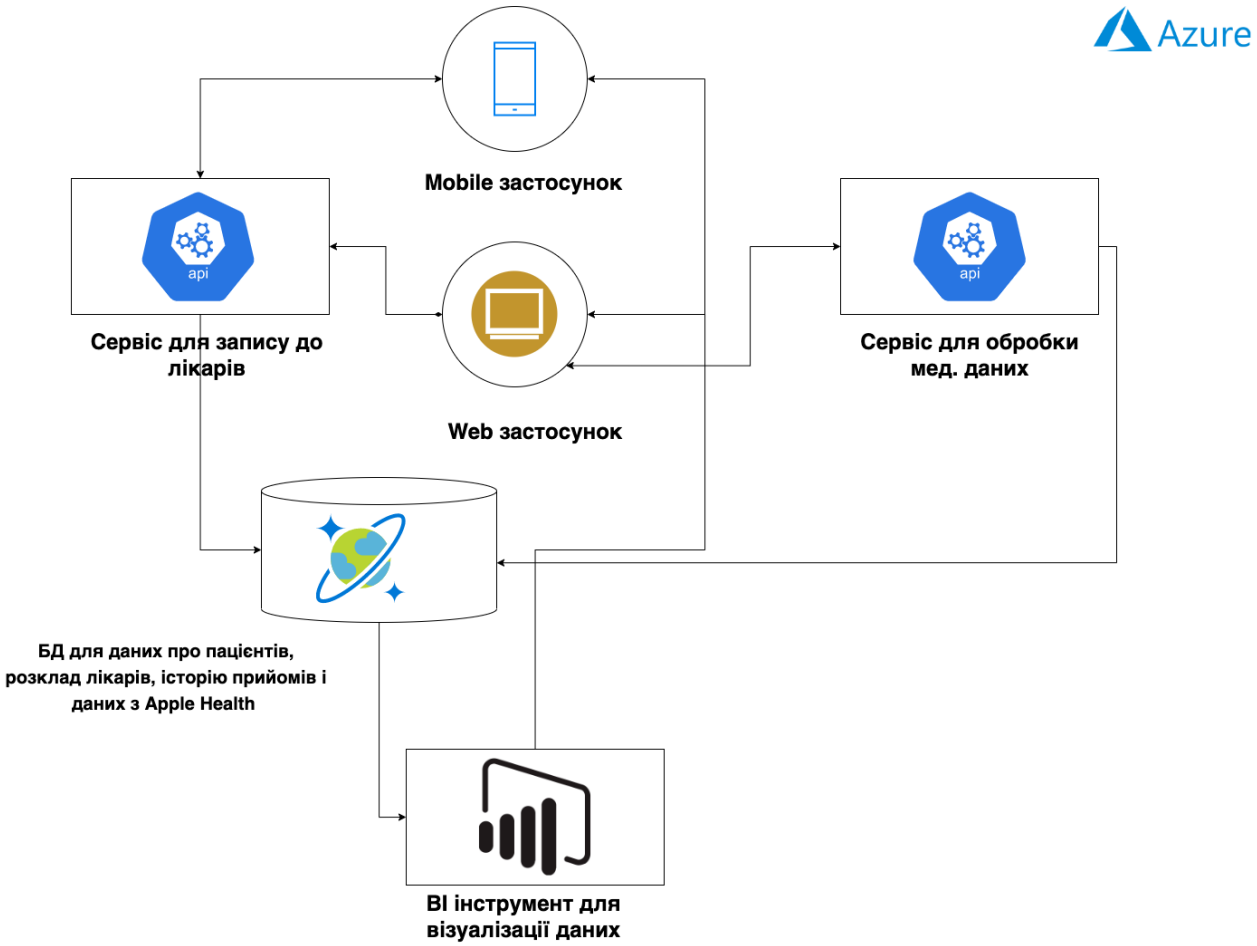
Якщо користувач хоче переглянути аналітику свого здоров’я, або ж переглянути дані про останній візит до лікаря, то ці дані зберігаються на нереляційній базі даних. Важливо зазначити, що система передбачає побажання користувача не ділитись медичними даними.

Завдяки сервісу запису, користувач має змогу записатись до лікаря на доступний для нього та лікаря час. Результатом виконання цього сервісу є надсилання запиту до лікаря, та збереження останнього у базі.

Якщо користувач погодився ділитись медичними даними, застосунок автоматично збиратиме дані з Health та надсилати їх сервісу запису та обробки медичних даних. Сервіс оброблятиме дані, та зберігатиме результат у нереляційній базі даних.

Для візуалізації даних використовується служба Azure яка збиратиме оброблені дані з бази даних, візуалізуватиме їх, та надсилатиме результат, у мобільний застосунок для користувача, web застосунок для лікаря та зберігатиме результат у базі даних.

2.5 Альтернативна архітектура



Відмінністю цієї архітектури буде наявність лише однієї бази даних, в нашому випадку нереляційної, в якій буду зберігатися як дані отримані з Apple Health так і дані про відвідування лікаря, виписки, та розкладу роботи лікарів.

Ця альтернатива необхідна для того, щоб не поєднувати в межах однієї системи дві різні бази даних, що може викликати проблеми при масштабуванні системи.

Розділ 3 Обробка та аналіз медичних даних

В рамках проектування системи аналізу та обробки, необхідно розглянути критерії аналізу даних. Важливо розуміти, що візуалізувати всі дані не доцільно, тому що кожен застосунок має свою власну візуалізацію для персонального користування. Так як наша система передбачає використання аналізу і як працівниками медичної сфери, так і користувачами, візуалізувати необхідно дані які виходять за критичні межі, встановлені персонально.

3.1 Критерії аналізу даних

На етапі розробки, цією системою передбачено обробку наступних даних:

- пульс
- активність протягом дня (кількість кроків)
- Стрес
- Режим сну
- Тренування

Як було вказано вище, візуалізувати медичні дані будуть не всі, а лише ті які виходять за критичні значення. Тобто отримати аналіз можливо лише в тому випадку коли результат виходить за рамки дозволеного. Результат візуалізації буде включати в себе графіки по кожному з параметрів розміщені паралельно одне до одного, для повноти розуміння коли і за якої причини пацієнт відчував себе погано.

Критичні значення в нашій системі будуть відповідати нормам сну, активності та пульсу. Але важливо також врахувати вік та вагу пацієнта та зріст через те що в комбінації з цими категоріями змінюються і критичні значення медичних даних. Система передбачає розрахунок індексу маси тіла

що впливає на активність протягом дня та кількість серцевих скорочень для пацієнта. Він також впливає на пульс та збільшує критичні межі встановлені в цій системі.

Системою передбачено редагування критичних меж індивідуально лікарем для кожного пацієнта. Але система також матиме власні налаштування, які будуть базовими для нових користувачів. Якщо ж користувач має заповнені персональні дані, такі як вік, зріст та вага, система буде підлаштовуватись під його характеристики.

3.2. Огляд метаданих

Конфіденційність, є важливим параметром для пристроїв Apple. Наприклад, одним із цікавих варіантів дизайну Apple Health є те, що всі ваші дані зберігаються локально на пристрої.

Насправді Apple не має доступу до цих даних, якщо ви не надасте їх безпосередньо їм під час експорту. Це означає, що якщо ви втратите або зламаєте телефон, ви також втратите дані про здоров’я. Отже, якщо ваші дані важливі, вам слід інвестувати в регулярне резервне копіювання своїх повних даних в iCloud або принаймні регулярно експортувати дані про здоров’я.

Правда, наявність даних про здоров’я в хмарі значно спростить інтеграцію та доступ. Наприклад, Google і Fitbit синхронізують ваші кроки та інші дані в хмарі. Щоб отримати необроблений експорт, необхідно відкрити додаток Apple Health, та перейти у налаштування користувача, покроково зображено на рис.3.1.

Цей процес експорту може зайняти кілька хвилин, і після завершення ви отримаєте файл під назвою «export.zip». Ви можете поділитися файлом із собою за допомогою AirDrop, електронної пошти або будь-яким іншим способом.

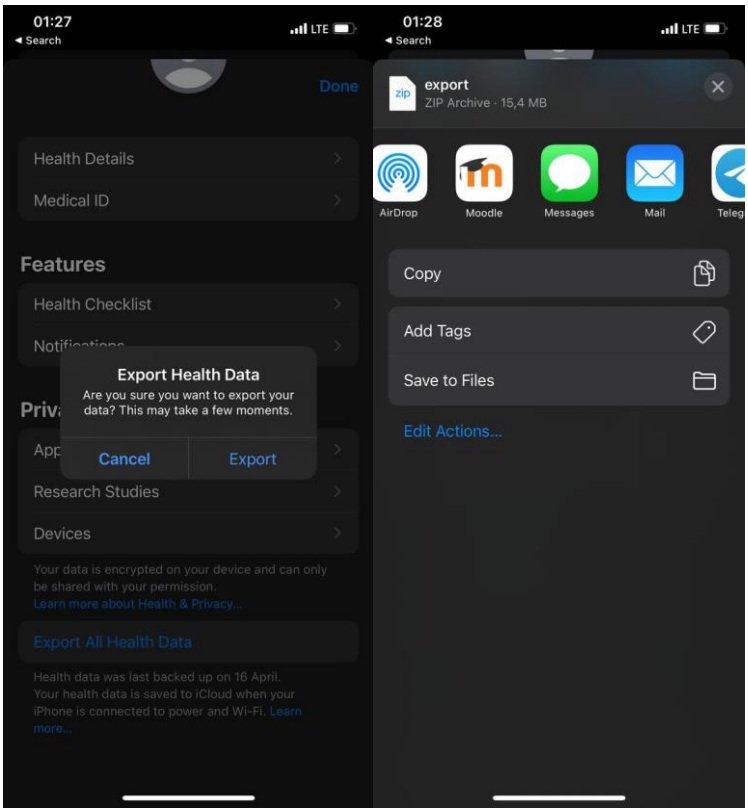


Рис. 3.1 Експорт даних за допомогою телефона

Отримавши архів, та відкривши його, всередині нього можна знайти файли, зображені на рис.3.2.

Имя	Дата изменения	Тип	Размер
export.xml	30.05.2022 1:27	Документ XML	186 952 КБ
export_cda.xml	30.05.2022 1:27	Документ XML	503 336 КБ

Рис. 3.2 Експортовані з zip архіву

Zip архів містить 2 файли: export_cda.xml та експорт.xml, та мають вони приблизно такий вигляд, як зображено на рис.3.3.

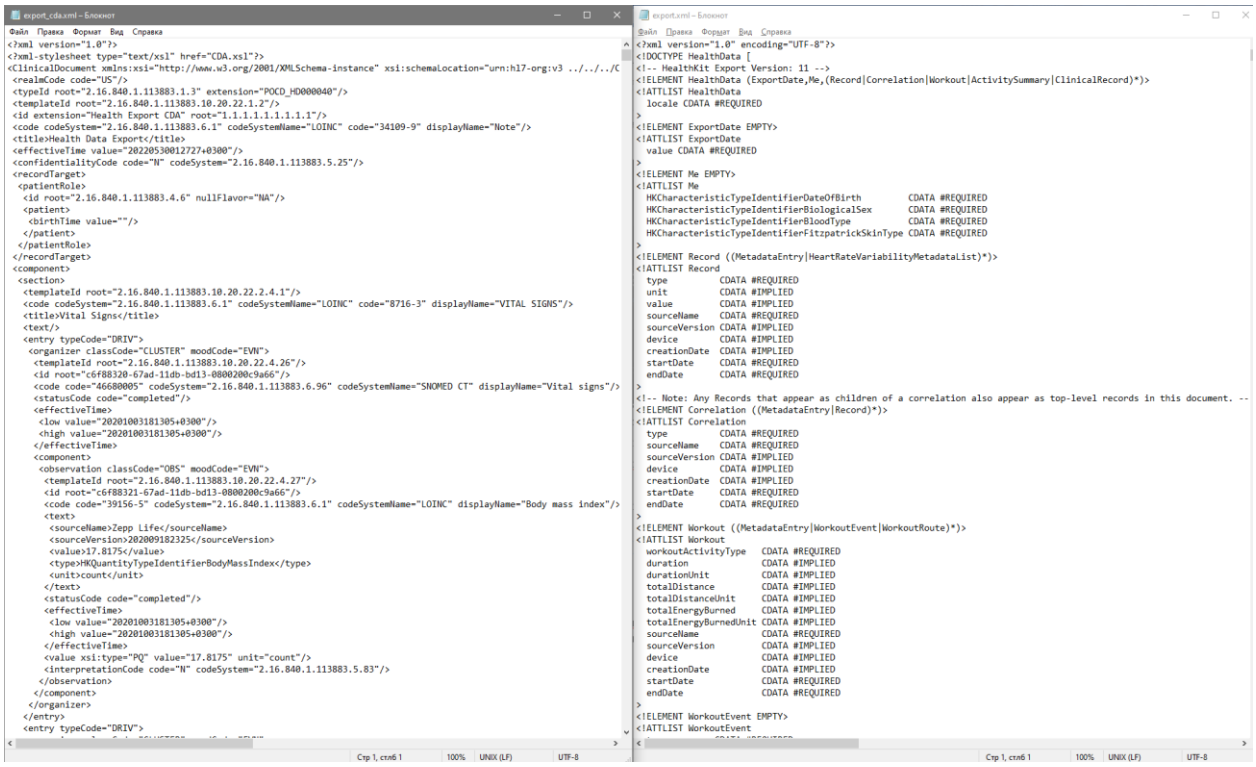


Рис. 3.3 приклад експортованих даних

Цей об’єм даних є достатньо великим, та необробленим. В моєму випадку обидва файли містять близько 700мб даних Ці дані збирались протягом декількох років, і нажаль після кожного експорту, ці дані будуть повторюватись, незалежно від того чи зберігались нові дані.

Висновки по роботі

У першому розділі була розглянута проблема високонавантажених систем, проблеми клінічних даних(V проблема), були розглянуті та порівняні сучасні рішення впроваджені для контролю та відстеження даних про здоров’я користувача. В результаті, як показало порівняння, дані які відображаються завдяки додатковим девайсам, більш змістовні і точні, аніж ті дані які автоматично збирає смартфон.

У другому розділі були сформовані вимоги до системи та архітектури. Проаналізували та порівняли інструменти які будуть використовуватись для реалізації. Було розроблено прототип архітектури, та запропоновано альтернативну архітектуру для реалізації, що може спростити подальшу роботу всієї системи в цілому.

Третій розділ присвячений обґрунтуванню налаштування параметрів для візуалізації, за якими буде налаштована наша система на аналіз та обробку даних. Також в третьому розділі було розглянуто приклади файлів xml формату.

Завдяки тому що Apple Health може зберігати різноманітні медичні дані, в подальшому система може бути модифікована, та в неї можуть бути додані нові функції для аналізу.

Окрім модифікації, система є досить гнучкої до масштабування, і розвиток може бути направлений на покращення інтерфейсу та додавання нових функцій як в мобільний застосунок так і на WEB застосунок.

Список літератури

1. https://www.statista.com/forecasts/997195/ehealth-tracker-smart-watch-usage-by-brand-in-the-us - статистика фітнес трекерів
2. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4758807/ - Sheridan W Miyamoto "Tracking Health Data Is Not Enough: A Qualitative Exploration of the Role of Healthcare Partnerships and mHealth Technology to Promote Physical Activity and to Sustain Behavior Change"
3. https://www.altexsoft.com/blog/big-data-healthcare/ - "Big Data in Healthcare: Sources and Real-World Applications"
4. https://link.springer.com/chapter/10.1007/978-3-030-16272-6_7 Salvatore Vitabile "Medical Data Processing and Analysis for Remote Health and Activities Monitoring"
5. Leijdekkers, Peter. "A Health Monitoring System Using Smart Phones and Wearable Sensors." (2007).
6. https://medium.com/mlearning-ai/lets-have-fun-with-those-apple-health-data-91a0f7f07447 - Let's mine those Apple Health data
7. Miyamoto SW, Henderson S, Young HM, Pande A, Han JJ. Tracking health data is not enough: a qualitative exploration of the role of healthcare partnerships and mHealth technology to promote physical activity and to sustain behavior change.
8. https://dev.to/smartym/how-to-build-a-high-load-architecture-for-your-web-project-4e8n - How to build a high load architecture for your web project?
9. https://cazton.com/blogs/technical/cosmosdb-vs-mongodb - Cosmos DB vs MongoDB.
10. https://mhealth.jmir.org/2017/1/e3/ - Mobile Health Physical Activity Intervention Preferences in Cancer Survivors: A Qualitative Study.