УДК 004.42:004.75:004.8
DOI https://doi.org/10.32782/tnv-tech.2025.1.14

РОЗРОБКА ІНФРАСТРУКТУРИ ДЛЯ ІНТЕГРАЦІЇ ШТУЧНОГО ІНТЕЛЕКТУ В ТЕЛЕМЕДИЧНІ СИСТЕМИ ЗА ДОПОМОГОЮ МІКРОСЕРВІСНОЇ АРХІТЕКТУРИ

Петренко А. І. *– доктор технічних наук, професор кафедри системного проектування Інституту прикладного системного аналізу Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського»*
ORCID ID: 0000-0001-6712-7792

Болобан О. А. *– аспірант кафедри системного проектування Інституту прикладного системного аналізу Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського»*
ORCID ID: 0009-0004-9074-4077

У цій роботі представлена мікросервісна архітектура для інтеграції моделей штучного інтелекту в телемедичні системи, яка забезпечує масштабованість, гнучкість та автоматизацію процесів навчання та використання моделей ШІ. Запропоноване рішення орієнтоване на ефективну взаємодію між сервісами через Message Broker (RabbitMQ/ Kafka), що дозволяє усунути залежності між мікросервісами, підвищити надійність та забезпечити безперервну обробку великих обсягів медичних даних. Використання HL7 Conversion Service дозволяє трансформувати вихідні результати прогнозів у стандартний медичний формат FHIR HL7, що забезпечує їхню сумісність із клінічними інформаційними системами. Крім того, Model Training Service дозволяє автоматизувати процес навчання, версіонування та оновлення моделей, що значно спрощує їхню інтеграцію та адаптацію до нових даних.

Розроблена архітектура може бути застосована для моніторингу пацієнтів у реальному часі, аналізу медичних зображень, прогнозування стану пацієнтів та персоналізованої медицини. Її використання дозволяє скоротити час обробки запитів, покращити точність діагностики та підвищити ефективність лікування завдяки аналізу великих масивів медичних даних. Використання розподілених сервісів та автоматизованих пайплайнів навчання моделей забезпечує гнучкість системи, що дозволяє легко масштабувати її та адаптувати до зростаючих обсягів даних та нових досліджень у сфері медицини. Основними викликами є складність налаштування інфраструктури, високі вимоги до обчислювальних ресурсів, а також необхідність дотримання стандартів безпеки даних відповідно до вимог HIPAA та GDPR.

Таким чином, запропонована архітектура демонструє ефективний підхід до розгортання інтелектуальних медичних платформ, що поєднують потужні алгоритми машинного навчання з високонадійною мікросервісною структурою, сприяючи розвитку інноваційних рішень у сфері охорони здоров'я.

Ключові слова: *мікросервісна архітектура, штучний інтелект (ai), телемедицина, message broker, rabbitmq, fhir hl7, інтеграція ai в медичні системи.*

Petrenko A. I., Boloban O. A. Development of infrastructure for integration of artificial intelligence into telemedicine systems using microservice architecture

This paper presents a microservice architecture for integrating artificial intelligence (AI) models into telemedicine systems, providing scalability, flexibility, and automation of AI model training and deployment. The proposed solution focuses on efficient service interaction through a Message Broker (RabbitMQ/Kafka), eliminating direct dependencies between microservices, improving reliability, and ensuring continuous processing of large volumes of medical data. The HL7 Conversion Service transforms AI predictions into the FHIR HL7 standard, ensuring seamless compatibility with clinical information systems. Additionally, the Model Training

Service automates training, versioning, and updating AI models, significantly simplifying their integration and adaptation to new datasets.

The designed architecture can be applied for real-time patient monitoring, medical image analysis, patient condition prediction, and personalized medicine. Its implementation reduces query processing time, enhances diagnostic accuracy, and improves treatment efficiency by analyzing vast amounts of medical data. The use of distributed services and automated AI training pipelines ensures system flexibility, allowing it to be easily scaled and adapted to growing data volumes and new research in the healthcare field. The main challenges include complex infrastructure setup, high computational resource requirements, and the need to comply with data security standards, such as HIPAA and GDPR.

Thus, the proposed architecture represents an effective approach to deploying intelligent medical platforms, combining powerful machine learning algorithms with a highly reliable microservice structure, fostering the development of innovative healthcare solutions.

Key words: *microservice architecture, artificial intelligence (ai), telemedicine, message broker, rabbitmq, fhir hl7, ai integration into medical systems.*

Вступ. Стрімкий розвиток штучного інтелекту (ШІ) та використання його у сферу охорони здоров'я відкривають нові можливості для автоматизації діагностики та моніторингу пацієнтів у режимі реального часу [1]. Високий рівень варіативності медичних даних, необхідність їх швидкої обробки та інтеграція з існуючими інформаційними системами вимагають побудови ефективних архітектурних рішень [2]. Використання мікросервісної архітектури дозволяє створювати масштабовані та стійкі до навантажень телемедичні системи, що забезпечують швидкий доступ до медичних записів та аналітичних прогнозів на основі ШІ [3].

Основною проблемою в сучасних медичних системах управління даними є фрагментація та низька інтероперабельність між різними інформаційними платформами. Медичні установи використовують різні системи для збору, обробки та збереження клінічних даних, що ускладнює їх інтеграцію, обмін і використання для прийняття рішень. Це призводить до таких викликів:

1. Несумісність форматів даних – різні медичні системи зберігають інформацію в різних структурах, що ускладнює обмін та аналіз даних.
2. Розподіленість даних між відділами – відсутність централізованої моделі обробки інформації призводить до дублювання записів і втрати важливих медичних показників.
3. Ручне введення та обробка даних – високий рівень ручної роботи збільшує ризик помилок і затримує отримання результатів аналізів.
4. Недостатня аналітика для підтримки рішень – традиційні інформаційні системи не надають інструментів для швидкого аналізу та прогнозування на основі накопичених даних.
5. Складність інтеграції нових технологій – сучасні рішення ШІ та машинного навчання важко вбудувати в існуючі монолітні системи через їхню жорстку архітектуру.

Використання мікросервісної архітектури дозволяє децентралізувати обробку медичних даних, створюючи модульну систему, де кожен сервіс відповідає за окрему функцію – збереження, обробку, аналіз або обмін даними [4]. Це значно підвищує гнучкість, масштабованість і спрощує інтеграцію нових сервісів.

Одним із ключових аспектів впровадження ШІ в телемедицину є його інтеграція з базами даних та сервісами обробки медичних показників. Це вимагає стандартизованої комунікації між сервісами, дотримання медичних регламентів (наприклад, HL7) та надійних механізмів передачі даних. Використання подійно-орієнтованої архітектури (EDA) на базі брокерів повідомлень, таких як RabbitMQ, забезпечує гарантовану доставку медичних даних і ефективну взаємодію між сервісами [4].

У цій статті розглядається мікросервісний підхід до побудови телемедичних систем та особливості інтеграції ШІ та організація хмарної системи шляхом використання брокерів повідомлень та алгоритмів забезпечення безпеки медичних даних.

Матеріали та методи. Монолітна архітектура передбачає централізовану структуру додатка, де всі його компоненти функціонують як єдине ціле, що означає високу зв'язаність між усіма модулями системи. У такій архітектурі всі частини – інтерфейс користувача, бізнес-логіка, обробка даних та база даних – розташовані в єдиному кодовому просторі, що унеможливлює їх незалежний розвиток. У випадку оновлення або внесення змін, необхідно оновлювати весь додаток, що може призводити до тривалих затримок та втрати даних, у випадку виникнення помилок під час розгортання нових оновлень.

Така архітектура була широко розповсюдженою в системах управління медичними даними, оскільки забезпечувала простоту розробки та управління, однак у сучасних умовах її недоліки стають очевидними та складними для виправлення. Наприклад, у великих медичних установах, де обробляється велика кількість даних пацієнтів, будь-які зміни в одному модулі можуть порушити роботу всієї системи. Крім того, масштабування таких систем потребує значних ресурсів, оскільки доводиться збільшувати обчислювальні потужності для всієї системи, а не окремих її частин.

Ще одним важливим обмеженням є низька адаптивність до нових технологій. Інтеграція штучного інтелекту, аналітичних модулів або хмарних сервісів у монолітну архітектуру є складною через жорстку структуру та відсутність гнучкості в обміні даними. Це ускладнює впровадження інновацій у сфері телемедицини, що потребує швидкого обміну інформацією та розподіленої обробки великих обсягів медичних даних.

Мікросервісна архітектура передбачає модульний підхід до проєктування медичних інформаційних систем, що дозволяє розділити систему на незалежні сервіси, кожен з яких виконує окрему функцію. Кожен сервіс працює автономно та взаємодіє з іншими за допомогою стандартизованих API, що забезпечує високу гнучкість та можливість швидкого оновлення окремих компонентів без впливу на загальну роботу системи [5]. Наприклад, один мікросервіс може відповідати за обробку та зберігання електронних медичних записів (EMR), інший – аналізувати діагностичні дані за допомогою штучного інтелекту, а ще один – виконувати функції автентифікації користувачів та управління доступом до інформації.

Таким чином, мікросервісна архітектура значно покращує ефективність медичних інформаційних систем, забезпечуючи їхню гнучкість, масштабованість та інтеграцію новітніх технологій. Це робить її оптимальним вибором для розгортання сучасних телемедичних платформ, які вимагають високої продуктивності, надійності та можливості адаптації до змінних умов роботи.

Приклади успішного використання мікросервісів у медицині

За результатами досліджень [6], використання мікросервісної архітектури в телемедичних системах забезпечує значне підвищення продуктивності, ефективності обробки даних та оптимізацію процесів управління медичною інформацією. Це дозволяє зменшити затримки у передачі та обробці даних, покращити доступність систем та забезпечити більш ефективну інтеграцію із зовнішніми медичними платформами та базами даних.

Одним із найбільш успішних прикладів реалізації мікросервісної архітектури є VITASENIOR-MT [7] – розподілена хмарна система, яка була розроблена для моніторингу пацієнтів у реальному часі, використовуючи набір незалежних

мікросервісів. Ця система дозволяє обробляти дані про стан здоров’я користувачів, надсилати оповіщення у випадку критичних змін у життєво важливих показниках та забезпечувати лікарів необхідною аналітичною інформацією.

Ще одним вагомим прикладом є Framework for Public Health Monitoring [8], який застосовується у великомасштабних медичних дослідженнях. Ця система побудована на базі мікросервісної архітектури, що забезпечує гнучке масштабування, можливість адаптації під нові вимоги та швидке впровадження змін. Вона використовується для збору, аналізу та візуалізації епідеміологічних даних, що дозволяє прогнозувати розвиток захворювань та оперативно реагувати на спалахи інфекційних хвороб.

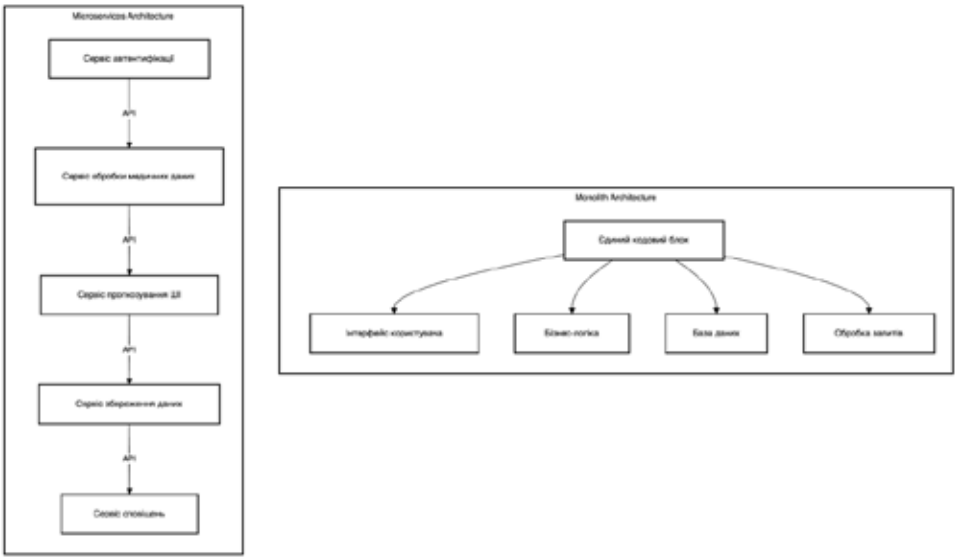

Рис. 1. Відмінність моноліту від мікросервісної архітектури

Як видно з діаграми (рис. 1), монолітна архітектура об’єднує всі модулі в єдине ціле, що ускладнює їх масштабування та оновлення. Натомість мікросервіси розділяють функціонал на окремі незалежні компоненти, які можуть взаємодіяти через API, що забезпечує високу гнучкість і масштабованість системи.

Перехід від монолітної архітектури до мікросервісної у медичних системах є необхідним для підвищення продуктивності, безпеки та інтероперабельності. Використання мікросервісів у поєднанні з ШІ дозволяє забезпечити ефективний аналіз медичних даних, швидку адаптацію до нових вимог та підтримку високих стандартів медичної інформації.

Взаємодія сервісів в мікросервісній архітектурі

У мікросервісній архітектурі взаємодія між сервісами може здійснюватися кількома способами, кожен з яких має свої переваги та обмеження. REST API (HTTP/HTTPS) є найбільш поширеним методом комунікації, який використовує стандартні HTTP-запити для передачі даних у форматі JSON або XML. gRPC (Google Remote Procedure Call) – це більш ефективний підхід, що застосовує HTTP/2 і Protobuf для швидкої та оптимізованої передачі інформації, забезпечуючи підтримку стрімінгових запитів. Event-Driven Architecture (EDA) базується

на асинхронній передачі подій між сервісами через брокери повідомлень, такі як RabbitMQ, Kafka або NATS, що дозволяє масштабувати систему без прямої залежності між сервісами.

Rest API

REST API є найбільш поширеним методом комунікації між мікросервісами завдяки простоті впровадження та широкій підтримці в різних мовах програмування. Даний метод комунікації базується на стандартних методах HTTP (GET, POST, PUT, DELETE) і використовує JSON або XML як формати передачі даних (рис. 2), що робить його сумісним із веб-додатками та мобільними сервісами. Однак, цей підхід має деякі обмеження, зокрема високу затримку через використання текстових форматів і необхідність обробки кожного запиту окремо, що може знижувати продуктивність у високонавантажених системах [9]. Незважаючи на це, REST API залишається ефективним для CRUD-операцій і добре підходить для веб-сервісів, які потребують надійної, але не обов'язково низькозатримкової взаємодії між мікросервісами.

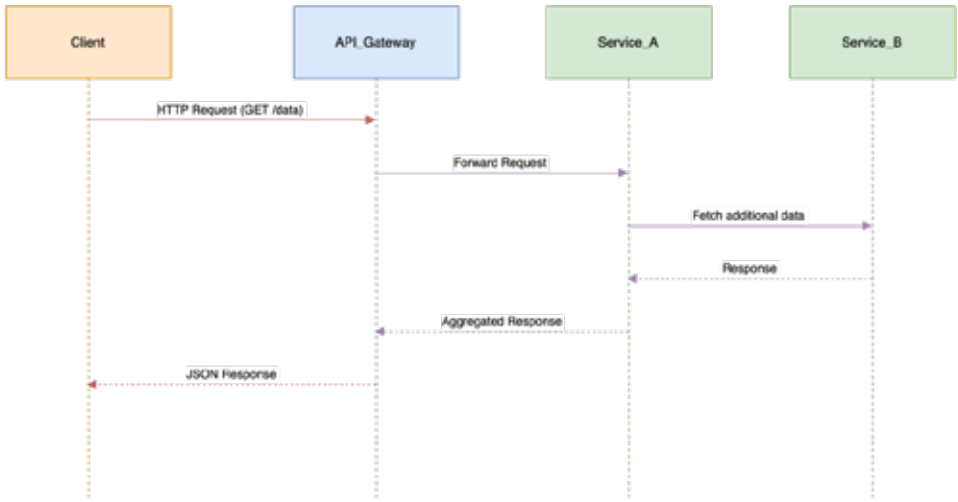

Рис. 2. Використання Rest API для взаємодії сервісів

gRPC

Одним із сучасних підходів до забезпечення ефективної взаємодії між мікросервісами є використання gRPC (Google Remote Procedure Call), який був розроблений Google і базується на протоколі HTTP/2 та форматі серіалізації Protobuf. На відміну від REST API, який використовує текстові формати JSON або XML, gRPC застосовує бінарний формат передачі даних, що забезпечує значно вищу продуктивність та менші мережеві витрати [10].

Згідно з дослідженням [11], gRPC демонструє кращу швидкодію та ефективність у порівнянні з REST API та GraphQL у мікросервісних архітектурах, особливо у високонавантажених системах, де важлива швидкість обробки запитів і мінімізація затримок. Цей підхід підтримує як синхронну, так і асинхронну комунікацію, а також стрімінгові API, що дозволяє реалізовувати складні сценарії інтеграції мікросервісів.

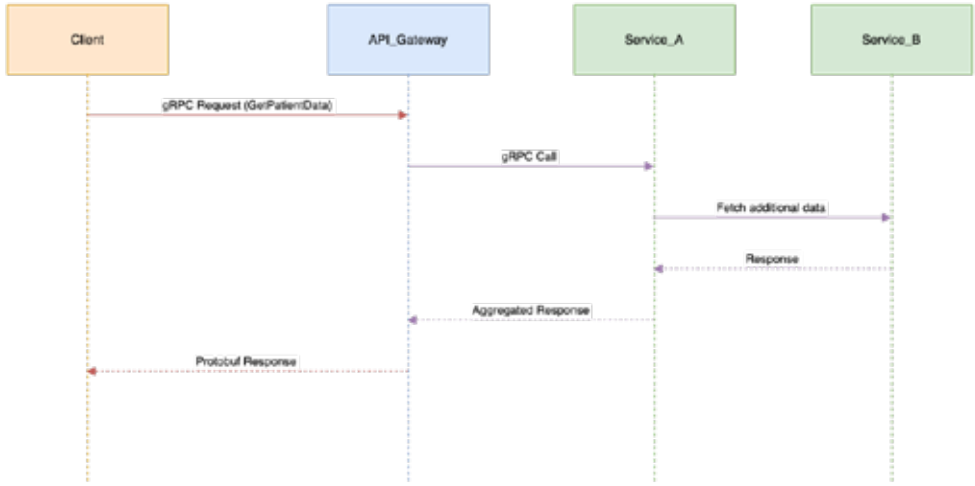

Рис. 3. Використатння gRPC для коммунікації сервісів

На діаграмі (рис. 3) представлено взаємодії сервісів між собою та з API Gateway за допомогою gRPC-запитів. У разі необхідності мікросервіс може взаємодіяти з іншими сервісами через gRPC, отримуючи необхідні дані та повертаючи відповідь у бінарному форматі Protobuf.

Використання брокерів повідомлень для взаємодії сервісів

Окрім синхронних методів комунікації, таких як REST API та gRPC, мікросервіси можуть ефективно взаємодіяти за допомогою брокерів повідомлень у подійно-орієнтованій архітектурі (Event-Driven Architecture, EDA). Цей підхід дозволяє сервісам обмінюватися подіями в асинхронному режимі, зменшуючи їхню залежність один від одного, що підвищує масштабованість і стійкість системи [12]. Подібна архітектура є особливо ефективною для телемедичних систем, де необхідно обробляти великий потік даних у реальному часі, наприклад, події про зміни в життєво важливих показниках пацієнтів.

RabbitMQ – це класичний брокер повідомлень, який працює за моделлю черг повідомлень (message queueing). Він забезпечує гарантовану доставку повідомлень, підтримує складні маршрутизаційні схеми та є зручним для використання в мікросервісній архітектурі. RabbitMQ використовує протокол AMQP (Advanced Message Queuing Protocol), що дозволяє створювати надійні механізми комунікації між сервісами.

Переваги RabbitMQ:

- Висока надійність та підтримка підтверджень доставки повідомлень.
- Гнучка маршрутизація повідомлень між мікросервісами.
- Підтримка затриманих та пріоритетних черг.
- Простота інтеграції з різними мовами програмування та інструментами DevOps.

Таким чином RabbitMQ є найбільш оптимальним вибором для телемедичних систем, які використовують подійно-орієнтовану архітектуру (EDA). Він забезпечує надійну доставку повідомлень, гнучку маршрутизацію та підтримує підтвердження доставки, що критично важливо у сфері охорони здоров’я, де втрата даних може мати серйозні наслідки [13]. Завдяки гнучкому управлінню чергами

RabbitMQ дозволяє створювати складні механізми обробки повідомлень між сервісами, що значно покращує стабільність системи та її ефективність.

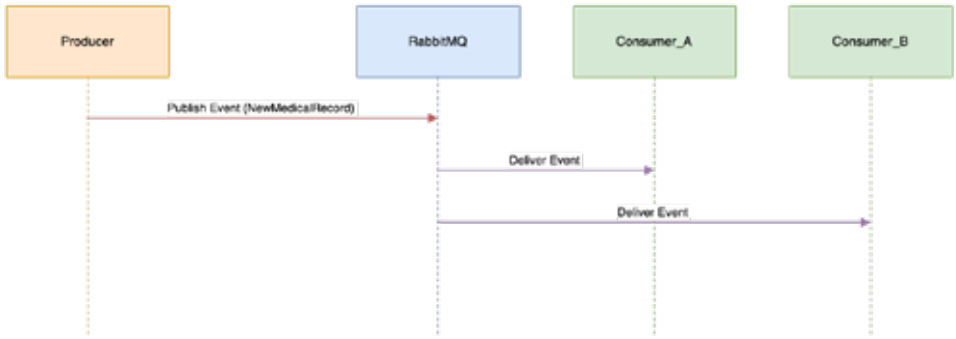

Рис. 4. Процес комунікації між продюсером та сервісами

Висока гнучкість RabbitMQ, його простота інтеграції та надійність роблять його чудовим рішенням для медичних мікросервісних архітектур. Використання RabbitMQ дозволяє телемедичним платформам ефективно обробляти події, покращувати відмовостійкість системи та забезпечувати високу продуктивність навіть за умов значного навантаження.

Результати. У ході дослідження було спроектовано систему, що базується на паттернах проектування мікросервісної архітектури, яка дозволяє інтегрувати та ефективно використовувати моделі штучного інтелекту в телемедичних проектах. Особливістю цієї архітектури є її масштабованість та адаптивність, що дає змогу інтегрувати різні моделі штучного інтелекту в медичні системи без значних змін у кодовій базі. Запропонована архітектура передбачає розділення компонентів на сервіси збору даних, їхньої обробки, навчання моделей, використання результатів прогнозування та збереження моделей. Всі компоненти працюють ізольовано, обмінюючись повідомленнями через Message Broker (рис. 5), що дозволяє забезпечити високу продуктивність, асинхронність та незалежне масштабування окремих сервісів. Таким чином, ця система надає можливість для користувачів взаємодіяти з моделями ШІ для аналізу медичних показників, отримуючи результати у зручному для сприйняття та аналізу форматі без необхідності заглиблюватися у технічні аспекти роботи алгоритмів.

Необхідні компоненти системи

API Gateway єдина точка входу для всіх зовнішніх запитів, що забезпечує маршрутизацію, автентифікацію та контроль доступу. Він підтримує REST та gRPC-запити, виконує балансування навантаження між сервісами та відповідає за конвертацію запитів у потрібний формат. Його основна мета – мінімізувати навантаження на внутрішні сервіси, забезпечуючи ефективну маршрутизацію трафіку та захист від несанкціонованого доступу.

Message Broker (RabbitMQ/Kafka) виступає в ролі централізованої шини обміну повідомленнями між сервісами, дозволяючи передавати запити асинхронно. Він гарантує доставку повідомлень і підтримує обробку запитів у реальному часі, що особливо важливо для систем, які працюють з великим потоком даних, зокрема телемедицини. Використання брокера повідомлень дозволяє сервісам працювати незалежно, не створюючи прямої залежності між мікросервісами.

Рис. 5. Архітектура сервісів для розгортання інфраструктури навколо сервісу з моделлю ШІ

AI Service є основним мікросервісом, що містить модель ШІ та відповідає за обробку вхідних запитів. Даний сервіс отримує дані через Message Broker у форматі JSON або Protobuf, після чого конвертує їх у необхідний формат для моделі. Після обчислення результат повертається у форматі JSON або у стандарті FHIR HL7 для інтеграції з медичними системами.

Model Storage (S3 / MinIO) використовується для збереження моделей у форматі ONNX, TensorFlow SavedModel або PyTorch TorchScript. Це забезпечує гнучке оновлення моделей без необхідності зупинки сервісу та дозволяє підтримувати кілька версій одночасно. Таке рішення сприяє швидкому деплойменту оновлених моделей та їх поверненню до попередніх версій у разі помилки.

Result Storage – база даних, у якій зберігаються результати роботи моделі. Вона використовується для подальшого аналізу та інтеграції з іншими медичними системами. Як основне сховище можуть бути використані PostgreSQL або MongoDB для структурованих даних, а неструктуровані результати можуть зберігатися в об’єктному сховищі.

Data Collector Service відповідає за збір нових даних з телемедичних пристроїв та їх збереження у Data Warehouse. Він підтримує взаємодію з сенсорами через MQTT або HTTP API, що дозволяє отримувати дані у реальному часі та забезпечує їхню коректну агрегацію для подальшого аналізу.

HL7 Conversion Service – сервіс, який перетворює результати моделі у стандарт HL7 FHIR для їхнього подальшого використання в медичних системах. Він отримує прогноз у форматі JSON, конвертує його у FHIR HL7 та передає через API Gateway для подальшої взаємодії з електронними медичними записами та лікарняними інформаційними системами.

Model Training Service є модулем, що відповідає за навчання моделей на GPU/TPU. Він отримує оновлені набори даних, проводить навчання, оцінює якість

моделей та деплоїть їх у Model Storage. Використання автоматизованих пайплайнів навчання забезпечує безперервне вдосконалення точності прогнозів.

Model Registry виконує роль централізованого сховища версій моделей. Цей сервіс дозволяє вести контроль якості моделей, відстежувати їхню продуктивність та гарантувати стабільне оновлення моделей у продуктивному середовищі. Завдяки системі версіонування можна швидко повертатися до попередніх версій у разі виявлення відхилень у роботі моделей.

Monitoring & Logging – система моніторингу продуктивності сервісів, що дозволяє оцінювати ефективність моделей та ініціювати їхнє перенавчання при необхідності. Вона включає збір логів, метрик продуктивності та механізми виявлення аномалій у прогнозах, що дозволяє підтримувати високу якість та надійність роботи моделі.

Запропонована архітектура демонструє ефективний підхід до інтеграції моделей ШІ у мікросервісну телемедичну систему, забезпечуючи гнучкість, масштабованість та безперервне вдосконалення моделей. Використання Message Broker забезпечує асинхронність комунікації, а HL7 Conversion Service гарантує відповідність медичним стандартам. Дана архітектура дозволяє швидке оновлення моделей, підтримку автоматизованого навчання та високу надійність прогнозів у реальному часі.

Висновки. Спроектована мікросервісна архітектура демонструє ефективний підхід до інтеграції моделей штучного інтелекту в телемедичні системи, забезпечуючи високу гнучкість, масштабованість та можливість безперервного вдосконалення моделей. Основною перевагою цієї системи є її модульна структура, що дозволяє незалежний розвиток кожного сервісу, а використання Message Broker усуває жорсткі залежності між мікросервісами, покращуючи їхню стійкість до збоїв. Система підтримує автоматизоване навчання та оновлення моделей ШІ, використовуючи Model Training Service та Model Registry, що дозволяє швидке впровадження нових алгоритмів та контроль якості прогнозування. Використання HL7 Conversion Service забезпечує відповідність даних медичним стандартам, що є критично важливим для інтеграції у клінічні інформаційні системи.

Сильними сторонами запропонованої архітектури є її масштабованість, що дозволяє збільшувати продуктивність шляхом додавання нових екземплярів сервісів без значних змін у кодовій базі. Асинхронна взаємодія між сервісами через RabbitMQ/Kafka значно знижує навантаження на систему, дозволяючи ефективно обробляти великий обсяг медичних даних у реальному часі. Крім того, система підтримує версіонування моделей, що дозволяє тестувати нові версії ШІ перед їх впровадженням у продуктивне середовище, що є критично важливим для медичних додатків.

Водночас, є кілька викликів, які слід врахувати при використанні такої архітектури в реальних системах. Насамперед, складність налаштування всіх компонентів, що вимагає глибокого розуміння роботи мікросервісів, брокерів повідомлень та розгортання моделей ШІ. Також необхідно забезпечити високу безпеку даних, оскільки система обробляє чутливу медичну інформацію, що регулюється стандартами HIPAA та GDPR. Іншим потенційним обмеженням є потреба у високопродуктивному обладнанні, особливо при навчанні моделей, що може вимагати використання GPU або TPU-кластерів.

Таким чином, ця розробка демонструє інноваційний підхід до побудови інтелектуальних медичних систем, який може бути легко масштабований та адаптований до різних медичних застосувань. Вона дозволяє створювати гнучкі, інтегровані рішення на основі штучного інтелекту, що забезпечують високу якість аналізу даних та полегшують ухвалення рішень у медицині.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ:

1. Timilehin, Oladoja. (2024). Continuous Health Monitoring with AI and Wearables: The Future of Preventive Medicine.

2. Walter, Holmes. (2024). Continuous Health Monitoring with AI and Wearables: The Future of Preventive Medicine., Calderón-Gómez et al., 2022

3. Petrenko, A., & Boloban, O. (2023). Generalized information with examples on the possibility of using a service-oriented approach and artificial intelligence technologies in the industry of e-Health. Technology Audit and Production Reserves, 4(2(72), 10–17. https://doi.org/10.15587/2706-5448.2023.285935

4. Boloban, O., Pysmennyi, I., Kyslyi, R., & Kyriusha, B. (2024). Development of a patient health monitoring system based on a service-oriented architecture using artificial intelligence. Technology Audit and Production Reserves, 3(2(77), 23–29. https://doi.org /10.15587/2706-5448.2024.306622

5. Ianculescu, Marilena. (2020). Microservices, a dynamic response for accelerating the assessment of patient-centric eHealth solutions.

6. Calderón-Gómez, Huriviades & Mendoza-Pitti, Luis & Vargas-Lombardo, Miguel & Gómez, José & Rodríguez-Puyol, Diego & Sencion-Martinez, Gloria & Polo-Luque, María-Luz. (2021). Evaluating Service-Oriented and Microservice Architecture Patterns to Deploy eHealth Applications in Cloud Computing Environment. Applied Sciences. 11. 4350. 10.3390/app11104350.

7. Mendes, D.; Jorge, D.; Pires, G.; Panda, R.; Antonio, R.; Dias, P.; Oliveira, L. VITASENIOR-MT: A distributed and scalable cloud-based telehealth solution. In Proceedings of the 2019 IEEE 5th World Forum on Internet of Things (WF-IoT), Limerick, Ireland, 15–18 April 2019; pp. 767–772, doi:10.1109/wf-iot.2019.8767184

8. Khalique, F.; Khan, S.A.; Nosheen, I. A Framework for Public Health Monitoring, Analytics and Research. IEEE Access 2019, 7, 101309–101326, doi:10.1109/ access.2019.2930730.

9. Owen, Antony. (2025). Microservices Architecture and API Management: A Comprehensive Study of Integration, Scalability, and Best Practices.

10. Purwanto, Alton & Darmanto, Darmanto & Trisno, Indra. (2024). Perancangan Protokol gRPC sebagai Komunikasi Data pada Arsitektur Microservices Aplikasi Manajemen Proyek. Jurnal Nasional Komputasi dan Teknologi Informasi (JNKTI). 7. 659-666. 10.32672/jnkti.v7i4.7697.

11. Niswar, Muhammad & Safruddin, Reza & Bustamin, Anugrayani & Aswad, Iqra. (2024). Performance evaluation of microservices communication with REST, GraphQL, and gRPC. International Journal of Electronics and Telecommunications. 429-436. 10.24425/ijet.2024.149562.

12. Kamau, Eunice & Mustapha, Sikirat & Babatunde, Gideon & Alabi, Abidemi & Pub, Anfo. (2025). Developing a Conceptual Model for Cross-Domain Microservices Using Event-Driven and Domain-Driven Design. International Journal of Multidisciplinary Research and Growth Evaluation. 04. 635-638.

13. (2024). SCALING MICROSERVICES ARCHITECTURES WITH CQRS AND EVENT-DRIVEN DESIGN.

REFERENCES:

1. Timilehin, Oladoja. (2024). Continuous Health Monitoring with AI and Wearables: The Future of Preventive Medicine.

2. Walter, Holmes. (2024). Continuous Health Monitoring with AI and Wearables: The Future of Preventive Medicine., Calderón-Gómez et al., 2022

3. Petrenko, A., & Boloban, O. (2023). Generalized information with examples on the possibility of using a service-oriented approach and artificial intelligence technologies in the industry of e-Health. Technology Audit and Production Reserves, 4(2(72), 10–17. https://doi.org/10.15587/2706-5448.2023.285935

4. Boloban, O., Pysmennyi, I., Kyslyi, R., & Kyriusha, B. (2024). Development of a patient health monitoring system based on a service-oriented architecture using artificial intelligence. Technology Audit and Production Reserves, 3(2(77), 23–29. https://doi.org /10.15587/2706-5448.2024.306622

5. Ianculescu, Marilena. (2020). Microservices, a dynamic response for accelerating the assessment of patient-centric eHealth solutions.

6. Calderón-Gómez, Huriviades & Mendoza-Pitti, Luis & Vargas-Lombardo, Miguel & Gómez, José & Rodríguez-Puyol, Diego & Sencion-Martinez, Gloria & Polo-Luque, María-Luz. (2021). Evaluating Service-Oriented and Microservice Architecture Patterns to Deploy eHealth Applications in Cloud Computing Environment. Applied Sciences. 11. 4350. 10.3390/app11104350.

7. Mendes, D.; Jorge, D.; Pires, G.; Panda, R.; Antonio, R.; Dias, P.; Oliveira, L. VITASENIOR-MT: A distributed and scalable cloud-based telehealth solution. In Proceedings of the 2019 IEEE 5th World Forum on Internet of Things (WF-IoT), Limerick, Ireland, 15–18 April 2019; pp. 767–772, doi:10.1109/wf-iot.2019.8767184

8. Khalique, F.; Khan, S.A.; Nosheen, I. A Framework for Public Health Monitoring, Analytics and Research. IEEE Access 2019, 7, 101309–101326, doi:10.1109/ access.2019.2930730.

9. Owen, Antony. (2025). Microservices Architecture and API Management: A Comprehensive Study of Integration, Scalability, and Best Practices.

10. Purwanto, Alton & Darmanto, Darmanto & Trisno, Indra. (2024). Perancangan Protokol gRPC sebagai Komunikasi Data pada Arsitektur Microservices Aplikasi Manajemen Proyek. Jurnal Nasional Komputasi dan Teknologi Informasi (JNKTI). 7. 659-666. 10.32672/jnkti.v7i4.7697.

11. Niswar, Muhammad & Safruddin, Reza & Bustamin, Anugrayani & Aswad, Iqra. (2024). Performance evaluation of microservices communication with REST, GraphQL, and gRPC. International Journal of Electronics and Telecommunications. 429-436. 10.24425/ijet.2024.149562.

12. Kamau, Eunice & Mustapha, Sikirat & Babatunde, Gideon & Alabi, Abidemi & Pub, Anfo. (2025). Developing a Conceptual Model for Cross-Domain Microservices Using Event-Driven and Domain-Driven Design. International Journal of Multidisciplinary Research and Growth Evaluation. 04. 635-638.

13. (2024). SCALING MICROSERVICES ARCHITECTURES WITH CQRS AND EVENT-DRIVEN DESIGN.