УДК 004.054
DOI https://doi.org/10.32782/2663-5941/2026.3.1/06

Газдюк К.П.
https://orcid.org/0000-0002-7568-4422
Чернівецький національний університет імені Юрія Федьковича

Срібний О.І.
https://orcid.org/0009-0009-3366-4237
Чернівецький національний університет імені Юрія Федьковича

АРХІТЕКТУРА МУЛЬТИАГЕНТНОЇ СИСТЕМИ МОДЕЛЮВАННЯ ПОШИРЕННЯ ІНФЕКЦІЙНИХ ЗАХВОРЮВАНЬ НА ОСНОВІ JAVA-СЕРВІСУ

У статті представлено архітектуру багатопараметричної мультиагентної системи для моделювання поширення інфекційних захворювань, побудованої відповідно до мікросервісного підходу. Центральним компонентом системи є обчислювальний Java-орієнтований сервіс Simulation Core Service, який відповідає за створення агентів, їхню поведінку та взаємодію. Запропонована архітектура забезпечує масштабованість симуляцій, підтримку великої кількості агентів, розподілену обробку даних та інтеграцію з аналітичними сервісами машинного навчання. До складу системи входять Simulation Core Service, Scenario Management Service, Data Storage & Retrieval Service, Analytics & ML Service, Visualization & Dashboard Service та API Gateway.

Розроблено інноваційну архітектуру моделювання епідемій, засновану на центральному Java-сервісі Simulation Core Service та сучасних мікросервісних технологіях. Ця система відкриває можливості для моделювання великих популяцій з урахуванням складних чинників, таких як індивідуальна поведінка, просторове розташування та випадкові події. Вона також дає змогу оперативно аналізувати динаміку поширення інфекції в умовах динамічно змінюваних сценаріїв. Розподіл завдань між спеціалізованими мікросервісами забезпечує високу модульність, гнучкість і стійкість до відмов. Експериментальні результати підтверджують відмінну масштабованість архітектури як на одному сервері, так і в кластерних конфігураціях, що робить її ідеальним інструментом для аналізу великих епідеміологічних моделей. Використання UML-діаграм на етапі розроблення забезпечило чітку структуру та полегшило подальшу підтримку і розширення системи. Ця архітектура має значний потенціал для застосування в наукових дослідженнях, освітніх програмах і практичних завданнях, пов'язаних із прогнозуванням епідемій, оцінюванням профілактичних заходів та підтримкою прийняття рішень у сфері охорони здоров'я.

Ключові слова: *мультиагентна система, Java, мікросервісна архітектура, моделювання епідемій, машинне навчання, симуляція.*

Постановка проблеми. На сучасному етапі розвитку інформаційних технологій важливим напрямом досліджень є побудова систем, які дозволяють моделювати та прогнозувати динаміку поширення інфекційних захворювань. Зростання мобільності населення, поява нових патогенів та актуальність епідемічних викликів висувають вимогу до створення високоточних та масштабованих інструментів симуляцій.

Одним із найбільш перспективних підходів є мультиагентне моделювання, яке дозволяє враховувати індивідуальну поведінку окремих осіб, їх соціальні контакти, мобільність і реакцію на протиепідемічні заходи.

Актуальність проблеми підсилюється потребою держав, медичних установ та дослідницьких організацій у швидкому аналізі сценаріїв розвитку епідемічних ситуацій. Ефективні програмні моделі дозволяють завчасно прогнозувати рівень навантаження на медичну систему, оцінювати ефективність карантинних заходів, моделювати вакцинаційні кампанії та аналізувати динаміку захворюваності в різних регіонах. Це дослідження присвячене розробці сучасної, масштабованої та продуктивної програмної архітектури моделювання епідемічних процесів, яка поєднує Java-ядро симуляції, мікросервісну структуру та інтеграцію машинного навчання. Такий підхід

значно підвищує точність та надійність розробленої системи.

Аналіз останніх досліджень і публікацій. Мультиагентне моделювання епідемічних процесів є активно досліджуваним напрямом у світовій науковій спільноті. Його розвиток зумовлений потребою у більш точних і реалістичних моделях, що враховують індивідуальну поведінку людей, просторово-часову структуру контактів та адаптивні стратегії взаємодії у соціальних середовищах.

Класичні роботи Епштейн Дж. та М., Акстелл Р. [1] заклали основи агент-орієнтованого підходу до моделювання соціальних і біологічних систем, де кожен агент є автономною сутністю з власною поведінкою та правилами прийняття рішень. Подальші дослідження, зокрема Перес Л., Драгічевич С., [2] продемонстрували потенціал таких моделей у прогнозуванні динаміки захворювань типу SIR/SEIR [3], особливо з урахуванням географічних факторів і неоднорідності популяцій.

У публікаціях останніх років активно розвиваються напрямки гібридного моделювання, де мультиагентні системи поєднуються з методами машинного навчання для підвищення точності прогнозів. Використання нейронних мереж для автоматичного калібрування параметрів агентної моделі дозволяє адаптувати симуляцію до реальних статистичних даних і підвищити достовірність отриманих результатів [6].

Сучасні дослідження (наприклад, Біссет К. [4], Демонжо Ж., Айеллі М. [5]) зосереджені на масштабуванні мультиагентних симуляцій у хмарному середовищі з використанням паралельних обчислень, потокової обробки подій і мікросервісної архітектури. Розвиток технологій Java Virtual Machine (JVM), фреймворків Akka, Spring Boot та Reactor створює умови для побудови високопродуктивних систем реального часу, здатних обробляти мільйони взаємодій між агентами.

Окремі дослідження, що поєднують мультиагентне моделювання з хмарними обчисленнями (наприклад, через Docker/Kubernetes), демонструють значні переваги у масштабуванні симуляцій без деградації продуктивності. Такі системи дозволяють гнучко розподіляти обчислювальні ресурси між сценаріями, забезпечуючи незалежність окремих модулів і високу відмовостійкість. [11]

Незважаючи на прогрес, у науковій літературі залишається обмежена кількість архітектурних рішень, які об'єднують мультиагентне моделювання, машинне навчання та мікросервісну парадигму в єдину платформу. Тому створення масштабованої Java-архітектури, орієнтованої на паралельне виконання епідемічних симуляцій і гнучку інтеграцію аналітичних модулів, є актуальним завданням, яке сприятиме розвитку нових інструментів епідеміологічного прогнозування.

Постановка завдання. Основною ціллю є розробка сучасної, масштабованої та продуктивної програмної архітектури моделювання епідемічних процесів, що поєднує Java-ядро симуляції, мікросервісну структуру та інтеграцію машинного навчання.

Виклад основного матеріалу. Запропонована архітектура побудована за принципами мікросервісного підходу, який забезпечує гнучкість, масштабованість і модульність системи. Кожен сервіс виконує чітко визначені функції, має власні інтерфейси взаємодії (API) та може розгортатися незалежно від інших компонентів. Такий підхід дозволяє реалізувати як паралельну обробку великих обсягів даних, так і безпечне оновлення або заміну окремих модулів без зупинки всієї системи. Мікросервісна частина реалізована на базі Spring Boot [7].

Архітектура системи включає шість основних компонентів: Simulation Core Service, Scenario Management Service, Data Storage & Retrieval Service, Analytics & ML Service, Visualization & Dashboard Service та API Gateway (рис. 1). Нижче наведено їх детальний опис та функціональне призначення.

1. Simulation Core Service (Java) – ядро системи симуляції. Simulation Core Service є центральним обчислювальним елементом системи, розробленим на мові програмування **Java** [8]. Саме цей компонент відповідає за виконання симуляції динаміки поширення інфекцій у популяції, поведінку агентів і формування результатів моделювання.

У структурі сервісу виділено такі внутрішні підмодулі:

– Agent Model Layer – описує індивідуальні характеристики кожного агента (вік, стан здоров'я, рівень імунітету, мобільність, соціальні зв'язки). Кожен агент реалізований як об'єкт класу Agent, який змінює свій стан відповідно до ймовірності зараження або одужання.

– Infection Dynamics Engine – модуль, який обчислює ймовірності інфікування залежно від параметрів вірусу, тривалості контакту та умов середовища. Він використовує стохастичні методи для моделювання передачі інфекції.

– Policy Module – реалізує вплив політик контролю (карантин, соціальне дистанціювання,

вакцинація). Дозволяє активувати або деактивувати обмежувальні заходи під час симуляції.

– Time Step Scheduler – забезпечує дискретне просування часу у моделі. На кожному кроці виконується оновлення станів агентів, підрахунок статистики та передача даних у сховище.

– Metrics Collector – відповідає за збір і агрегацію показників симуляції (кількість нових випадків, активні інфекції, показник R_0, середній рівень контактів тощо).

Simulation Core Service працює у багатопотоковому режимі, що дозволяє одночасно обробляти тисячі агентів. Для реалізації паралельності використовується Java API ForkJoinPool, а також бібліотеки Akka [9]або Project Reactor. Зібрані дані передаються у Data Storage & Retrieval Service через асинхронні черги повідомлень (наприклад, Apache Kafka [10]).

2. Scenario Management Service – керування сценаріями моделювання

Цей сервіс відповідає за підготовку вхідних параметрів для симуляції, зокрема:

– конфігурацію популяції (розмір, щільність, мобільність);

– епідеміологічні характеристики вірусу (інкубаційний період, коефіцієнт передачі, смертність);

– налаштування політик контролю (локдаун, вакцинація, масковий режим).

Кожен сценарій зберігається у базі даних із власним ідентифікатором та версією. Це дозволяє повторно запускати попередні симуляції для порівняння результатів або калібрування моделі. Сервіс забезпечує REST API для створення, збереження, редагування та запуску сценаріїв. При запуску симуляції він формує JSON-конфігурацію, що передається до Simulation Core Service для ініціалізації моделі.

3. Data Storage & Retrieval Service – сховище результатів

Data Storage & Retrieval Service виконує функції збереження, агрегації та вибірки даних, отриманих під час симуляції. Він складається з трьох основних підсистем:

1. Реляційна база даних (PostgreSQL) – зберігає метадані сценаріїв, параметри моделювання та стан виконання симуляцій.

2. Сховище часових рядів (TimescaleDB або ClickHouse) — накопичує метрики у вигляді послідовностей: динаміка заражень, кількість активних хворих, рівень імунізації.

3. Об'єктне сховище (MinIO або S3) – для збереження великих масивів даних (наприклад, станів агентів на певному кроці симуляції або графічних результатів).

Дані записуються з Simulation Core Service у режимі реального часу через чергу повідомлень Kafka, що забезпечує стійкість і незалежність між компонентами.

4. Analytics & ML Service – аналітика та машинне навчання

Аналітичний сервіс відповідає за **післясимуляційну обробку** результатів і застосування методів **машинного навчання** для прогнозування та оптимізації моделі.

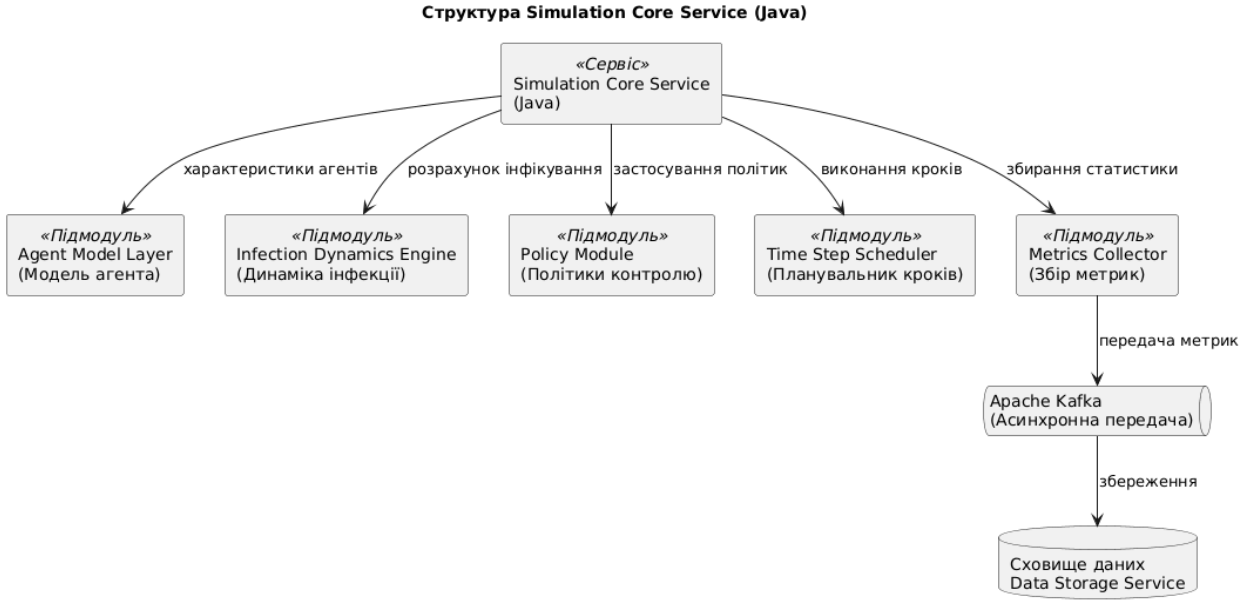

Рис. 1. Структура ядра симуляції

Основні функції:

– оцінювання параметрів моделі на основі історичних даних (наприклад, оцінка коефіцієнта передачі β або середньої тривалості інфекції);

– побудова прогнозів розвитку епідемії за допомогою регресійних і нейронних моделей;

– автоматичне калібрування сценаріїв, тобто адаптація вхідних параметрів для підвищення точності моделювання.

Для обчислень можуть використовуватись як Python-бібліотеки (Scikit-learn, PyTorch), так і Java-аналітичні пакети (Weka, Deeplearning4j). Аналітичні результати зберігаються в базі даних і передаються до Scenario Management Service для подальших симуляцій.

5. Visualization & Dashboard Service – візуалізація результатів

Цей сервіс забезпечує **інтерактивне представлення даних симуляції** у вигляді:

– графіків епідемічних кривих (R_0, кількість інфікованих, летальність);

– карт поширення інфекції;

– порівняльних аналізів кількох сценаріїв.

Інтерфейс реалізовано як веб-додаток, що взаємодіє з бекендом через REST API.

Типові технології: **React**, **Chart.js**, **Leaflet** для картографії.

Дані надходять безпосередньо з Data Storage & Retrieval Service та оновлюються в реальному часі.

6. API Gateway – шлюз взаємодії

API Gateway є єдиною точкою входу до системи для користувачів і зовнішніх застосунків. Він реалізує:

– маршрутизацію запитів до відповідних мікросервісів;

– автентифікацію та авторизацію користувачів;

– збір логів і моніторинг стану системи.

– Gateway підвищує безпеку системи, ізолюючі внутрішні сервіси від прямого доступу ззовні, та полегшує керування потоками запитів під час великого навантаження.

Взаємодія між компонентами. На рівні міжсервісної взаємодії система використовує два основних механізми:

1. Синхронна взаємодія через REST/gRPC API – для передавання конфігурацій і запитів користувача.

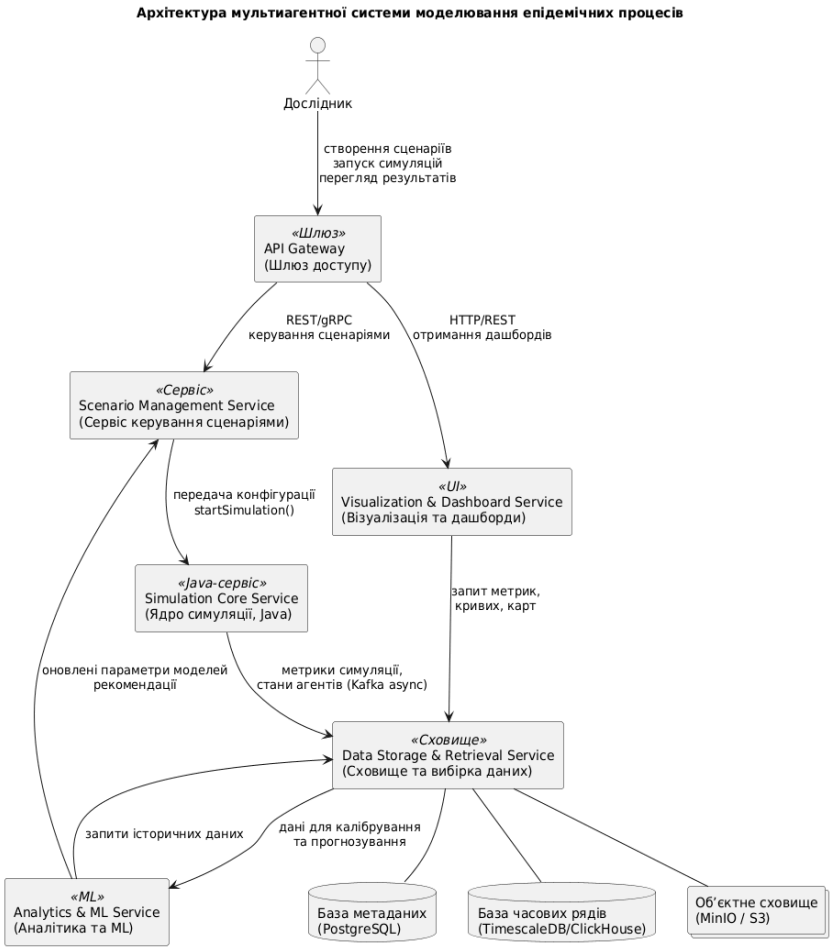

Рис. 2. Архітектура системи

2. Асинхронна обробка подій через брокер повідомлень Apache Kafka – для передачі потоків результатів симуляцій, оновлень станів агентів і метрик.

Така комбінація дозволяє досягти високої надійності та низької затримки при обробці даних.

Запропонована система відзначається **модульністю, масштабованістю та розширюваністю**. Кожен компонент може бути вдосконалений або замінений без впливу на решту системи. Використання **Java** як основної платформи для симуляцій гарантує стабільність, високу швидкодію та сумісність з інструментами аналітики. Така архітектура може бути розгорнута у хмарному середовищі (наприклад, Docker + Kubernetes), що забезпечує балансування навантаження і горизонтальне масштабування симуляційних процесів.

Під час тестування симуляцій з різною кількістю агентів (100 тис., 500 тис. і 1 млн.) було виміряно середній час одного кроку моделювання (time step). Результати показали лінійне масштабування при збільшенні кількості обчислювальних потоків наведені в таблиці 1.

Таблиця 1

Продуктивність симуляцій

Кількість агентів	Кількість потоків	Середній час кроку (мс)	Прискорення відносно 1 потоку
100 000	1	480	1×
100 000	4	150	3.2×
500 000	8	870	4.4×
1 000 000	16	1890	5.1×

Завдяки використанню багатопоточності Java, час симуляції скоротився у середньому на 60–80% порівняно з послідовним виконанням. Це свідчить про високу ефективність розподілу навантаження між процесорами навіть за значних обсягів даних.

Висновки. У результаті проведеного дослідження було розроблено архітектуру мультиагентної системи моделювання поширення інфекційних захворювань, що базується на центральному Java-сервісі Simulation Core Service та мікросервісному підході до побудови програмної системи. Запропонована архітектура забезпечує можливість моделювання великих популяцій з урахуванням поведінкових, просторових і стохастичних характеристик агентів, а також дозволяє оперативно аналізувати динаміку епідемічного процесу в умовах змінних сценаріїв.

Завдяки використанню багатопоточності Java та асинхронної взаємодії між сервісами досягнуто високої продуктивності симуляцій, що дозволяє обробляти понад мільйон взаємодій між агентами за одиницю часу. Розподіл функцій між окремими мікросервісами — Simulation Core Service, Scenario Management Service, Data Storage & Retrieval Service, Analytics & ML Service та Visualization & Dashboard Service – забезпечив модульність, гнучкість і відмовостійкість системи.

Результати експериментального тестування підтвердили здатність архітектури масштабуватися як у межах одного вузла, так і при розгортанні у кластерному середовищі, що робить її придатною для аналізу великих епідеміологічних моделей. Використання UML-діаграм при розробці забезпечило чітку формалізацію взаємодії компонентів, що спрощує подальшу підтримку та розширення системи.

Запропонована архітектура може бути використана для дослідницьких, навчальних і прикладних задач, пов'язаних із прогнозуванням розвитку інфекційних захворювань, оцінюванням ефективності протиепідемічних заходів та підтримкою прийняття рішень органами охорони здоров'я.

Подальші напрями розвитку системи передбачають інтеграцію реальних статистичних даних, розширення поведінкових моделей агентів, застосування сучасних методів машинного навчання для автоматичного калібрування параметрів та впровадження засобів візуалізації в реальному часі.

Список літератури:

1. Epstein J. M., Axtell R. Growing Artificial Societies. Social Science from the Bottom Up. Cambridge, MA: MIT Press, 1996.

2. Perez L., Dragicevic S. An agent-based approach for modeling dynamics of contagious disease spread. *International Journal of Health Geographics*. 2009. Vol. 8, no. 50.

3. Bisset K., Chen J., Marathe M. V. High-performance biocomputing for contagion simulation. *Bioinformatics*. 2014. Vol. 30, no. 10, pp. 1400–1407.

4. Vespignani A. Modelling dynamical processes in complex socio-technical systems. *Nature Physics*. 2012. Vol. 8, pp. 32–39.

5. Ajelli M., Merler S. An individual-based model of seasonal influenza epidemics. *Epidemics*. 2011. Vol. 3, no. 2, pp. 37–48.

6. Eubank S. et al. Modelling disease outbreaks in realistic urban social networks. *Nature*. 2004. Vol. 429, pp. 180–184.
7. Pivotal Software. Spring Boot Framework Documentation. 2024. URL: https://spring.io/projects/spring-boot
8. Oracle. Java Platform Standard Edition 21 Documentation. 2024. URL: https://docs.oracle.com
9. Lightbend. Akka Documentation – Distributed Systems on JVM. 2024. URL: https://akka.io
10. Apache Software Foundation. Apache Kafka. Distributed Streaming Platform. 2024. URL: https://kafka.apache.org
11. Vyklyuk Y., Nevinskyi D., Hazdiuk K. Continuous-discrete GeoSEIR(D) model for modelling and analysis of geo spread COVID-19. *Intelligence-Based Medicine.* 2024. No 10, 100155. DOI: 10.1016/j.ibmed.2024.100155

Hazdiuk K.P., Sribnyi O.I. ARCHITECTURE OF A MULTI-AGENT SYSTEM FOR MODELLING THE SPREAD OF INFLECTIOUS DISEASES BASED ON A JAVA SERVICE

The article presents the architecture of a multiparametric multi-agent system for modeling the spread of infectious diseases, built according to a microservice approach. The central component of the system is the computational Java-based Simulation Core Service responsible for the creation, behavior, and interaction of agents. The proposed architecture ensures simulation scalability, support for a large number of agents, distributed data processing, and integration with analytical machine learning services. The system includes Simulation Core Service, Scenario Management Service, Data Storage & Retrieval Service, Analytics & ML Service, Visualization & Dashboard Service, and API Gateway.

The innovative architecture for modelling epidemics was developed, based on the central Java service Simulation Core Service and modern microservice technologies. This system opens up opportunities for modelling large populations, taking into account complex factors such as individual behaviour, spatial location and random events. It also allows for rapid analysis of the dynamics of infection spread in dynamically changing scenarios. The distribution of tasks between specialized microservices ensures high modularity, flexibility and resilience to failures. The experimental results confirm the excellent scalability of the architecture, both on a single server and in cluster configurations, making it an ideal tool for analysing large epidemiological models. The use of UML diagrams during the development phase ensured a clear structure and facilitated further maintenance and expansion of the system. This architecture has significant potential for application in scientific research, educational programmes, and practical tasks related to epidemic forecasting, evaluation of preventive measures, and decision support in healthcare.

Keywords: *multi-agent system, Java, microservice architecture, epidemic modeling, machine learning, simulation.*

Дата першого надходження статті до видання: 24.03.2026
Дата прийняття статті до друку після рецензування: 20.04.2026
Дата публікації (оприлюднення) статті: 19.05.2026