http://www.moderntechno.de/index.php/meit/article/view/meit38-01-026
DOI: 10.30890/2567-5273.2025-38-01-026

УДК 004.2

INTEGRATION OF MEDICAL INFORMATION SYSTEMS: ARCHITECTURAL ANALYSIS AND ONTOLOGY-BASED APPROACH TO COMPATIBILITY TESTING

ІНТЕГРАЦІЯ МЕДИЧНИХ ІНФОРМАЦІЙНИХ СИСТЕМ: АРХІТЕКТУРНИЙ АНАЛІЗ І ОНТОЛОГІЧНИЙ ПІДХІД ДО ТЕСТУВАННЯ СУМІСНОСТІ

Krupa D.V / Крупа Д.В
аспірант
ORCID: 0009-0002-6087-9988
Pavliv I.I. / Павлів І.І.
аспірант
ORCID: 0009-0003-0957-0843
Lviv Polytechnic National University,
12 Stepana Bandery Street, Lviv, 79000, Ukraine
Національний університет «Львівська політехніка»,
м. Львів, Степана Бандери 12, 79000

Анотація. *У роботі представлено архітектурний підхід до побудови модуля запису на прийом у медичних інформаційних системах із акцентом на уніфікацію користувацької моделі, стабільність інтеграції та автоматизацію перевірки сумісності між компонентами. У рамках запропонованої моделі користувач може виступати як пацієнтом, так і лікарем, із розширенням через відповідні профілі, що містять рольові атрибути. Медичні записи реалізовані через модель прийому з перевіркою узгодженості із розкладом лікаря. Інтеграція із зовнішніми платформами — мобільними додатками, телемедичними сервісами, порталами страхових компаній та системами EHR — здійснюється за допомогою REST API, який підтримує OpenAPI-специфікацію, аутентифікацію через JWT і принципи семантичної версіонності.*

Окрема увага приділена автоматизованому тестуванню сумісності між системами. Запропоновано механізм оцінювання структурної, обмежувальної та логічної відповідності між компонентами на основі онтологічного представлення даних і методу нечіткого логічного виводу Мамдані. Для агрегування результатів використано метод зважених сум, що дозволяє сформувати узагальнену оцінку якості сумісності з урахуванням вагових коефіцієнтів критеріїв. Наведено приклад реалізації онтології у форматі RDF, побудову тестових класів системи-споживача та експеримент з оцінкою часу виконання перевірки, що засвідчує ефективність підходу порівняно з традиційними інтеграційними тестами. Запропонований підхід дозволяє підвищити прозорість процесу прийняття рішень про випуск нових версій продукту та може бути використаний як частина DevOps-пайплайну в системах, де важливе дотримання онтологічної відповідності та стабільної міжсистемної взаємодії.

Ключові слова*: медичні інформаційні системи, запис на прийом, REST API, онтологія, сумісність компонентів, контрактне тестування, нечітка логіка, профілі користувача, інтеграція з EHR, автоматизація тестування, прийняття рішень, онтологічний підхід*

Вступ

У сучасних медичних інформаційних системах модуль запису на прийом до лікаря відіграє ключову роль у забезпеченні ефективної та безперебійної

взаємодії між пацієнтами та медичними закладами. Зважаючи на високі вимоги до точності, безпеки та доступності медичних даних, архітектура цього модуля повинна підтримувати масштабованість, гнучкість та надійність при інтеграції з іншими системами.

У запропонованому модулі управління медичними записами використовується уніфікована модель User, яка репрезентує як пацієнтів, так і лікарів. Залежно від ролі, кожен користувач може мати один з відповідних профілів — PatientProfile або DoctorProfile, що містять специфічну для ролі інформацію. Такий підхід дозволяє централізовано керувати автентифікацією, авторизацією та іншими спільними механізмами для всіх типів користувачів. Водночас профілі забезпечують розділення логіки та зберігання тільки релевантних даних.

Для організації медичних записів застосовується модель Appointment, яка слугує зв'язком між пацієнтом і лікарем, відображаючи факт медичної зустрічі. Кожен лікар має пов'язану модель Schedule, що визначає доступні часові слоти для запису. Appointment перевіряє узгодженість з цим розкладом, забезпечуючи коректність і унеможливлення конфліктів при бронюванні. Разом ці моделі формують цілісну архітектуру, яка підтримує автоматизований запис, гнучке управління доступністю лікарів і стабільну інтеграцію з іншими медичними системами.

У багатьох медичних інформаційних системах пацієнти та лікарі представляють окремі сутності. Проте уніфікація їх у спільну модель User спрощує систему автентифікації, дозволяє централізовано керувати правами доступу, спрощує реалізацію спільних функцій (наприклад, вхід до системи, редагування профілю) та підвищує узгодженість логіки взаємодії з REST API. Це особливо важливо при міжсистемній інтеграції, коли зовнішні сервіси (наприклад, мобільні додатки або сторонні платформи телемедицини) мають взаємодіяти з уніфікованими інтерфейсами.

Згідно із дослідженнями компанії Postman, Inc у 2024 році (Postman 2024 State of the API Report), найбільшою складністю для 39% респондентів була

недостатня кількість документації і 43% учасників опитування відповіли що надіються на допомогу колег у розумінні API [7]. Зазначмо, що у даному звіті велика увага приділяється проблемам інтеграції API із AI засобами, а у звіті за 2023 рік (Postman 2023 State of the API Report) більше була акцентована увага саме на складність розуміння API розробниками. У звіті 2023 року були згадані такі проблеми як складнощі дослідження API систем (32%) та нестача часу (27%) [8]. Це означає, що коли API системи-постачальника змінюється, відстежити ці зміни може бути непросто — особливо якщо вони стосуються внутрішньої логіки або прихованих механізмів роботи.

Основний текст τ SЃI ′ SI Ґѡ‴ Ë· ′˜′“

У модулі запису на прийом використовується п’ять ключових моделей, кожна з яких виконує чітко визначену роль. Архітектура побудована навколо уніфікованої моделі користувача, що дозволяє централізовано керувати доступом, ролями та взаємодією з іншими компонентами системи.

Модель User є базовою сутністю системи, яка репрезентує будь-якого користувача — як пацієнта, так і лікаря (рисунок 1). Вона об’єднує спільні для обох ролей характеристики, забезпечує механізми автентифікації та авторизації, а також слугує центральною точкою для побудови зв’язків із рештою моделей. Завдяки єдиній моделі користувача забезпечується консистентність у роботі API та спрощується інтеграція з зовнішніми сервісами.

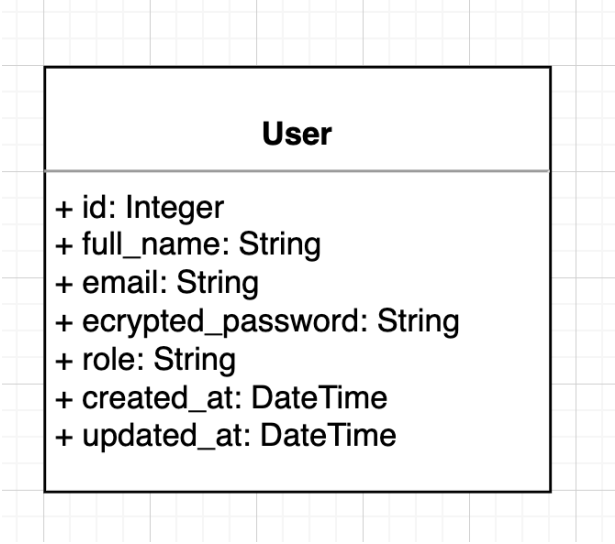

Рисунок 1 - *UML діаграма Користувача*

Авторська розробка

PatientProfile є розширенням моделі User для випадку, коли користувач виконує роль пацієнта (рисунок 2). У цьому профілі зберігається інформація, специфічна для обслуговування пацієнта, така як медичні ідентифікатори, дані про контактні особи тощо. Відокремлення таких атрибутів у профіль дозволяє підтримувати модульність і чітке розділення логіки між ролями.

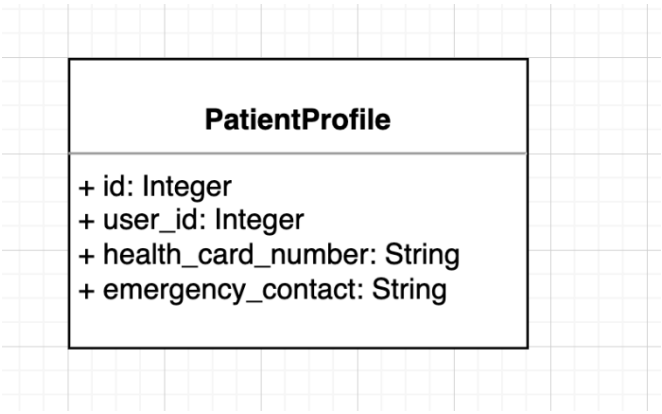

Рисунок 2 - *UML діаграма Пацієнта*

Авторська розробка

Аналогічно до пацієнтського профілю, DoctorProfile розширює модель User, коли користувач є лікарем (рисунок 3). Тут зберігається інформація про спеціалізацію, відділення, ліцензії та інші атрибути, важливі для професійної діяльності. Це дозволяє системі динамічно визначати роль користувача, не дублюючи загальні дані.

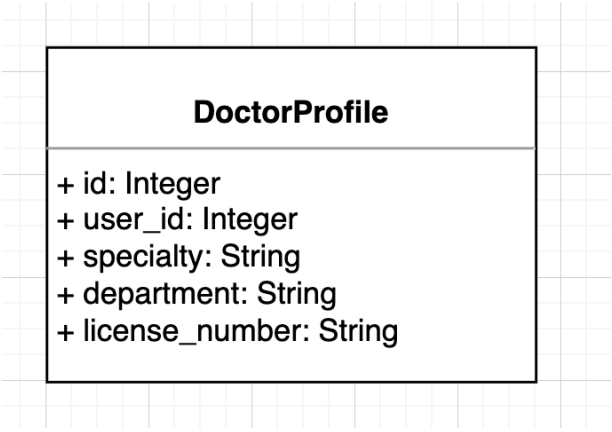

Рисунок 3 - *UML діаграма Лікаря*

Авторська розробка

Модель Schedule (рисунок 4) прив'язана до лікаря і визначає доступні часові слоти для запису пацієнтів. Розклад може формуватися автоматично або вручну

через адміністративну панель, а також бути синхронізованим із зовнішніми системами (наприклад, EHR або календарями). У контексті автоматизації процесів у сфері охорони здоров'я подібні підходи застосовуються і в інших дослідженнях. Так, у статті The Development of a Medical Equipment Inventory Information System описано створення програмного забезпечення для підтримки електромедичного персоналу в управлінні та моніторингу медичного обладнання в лікарнях, що також передбачає формування графіків обслуговування. Коректне функціонування цієї моделі є критичним для уникнення конфліктів при бронюванні та для забезпечення високої якості обслуговування. [2]

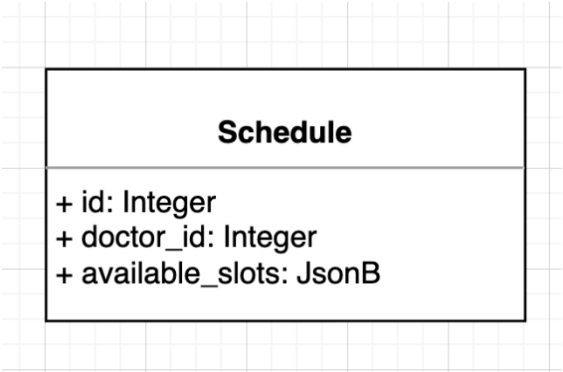

Рисунок 4 - *UML діаграма Розкладу*

Авторська розробка

Модель Appointment (рисунок 5) є ключовим елементом, що об'єднує пацієнта, лікаря та визначений часовий слот. Вона відображає запланований або завершений візит і містить інформацію про дату, час та статус запису. Кожен запис перевіряється на відповідність до графіка лікаря (Schedule), що гарантує узгодженість і точність в роботі системи. Модель також служить джерелом для статистичного аналізу та звітності.

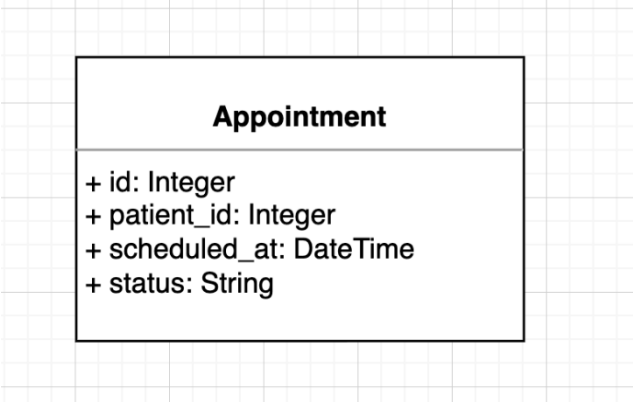

Рисунок 5 - *UML діаграма Зустрічі*

Авторська розробка

Цей набір моделей утворює модульну та гнучку основу для створення розширюваної медичної системи, яка легко інтегрується з іншими сервісами через REST API (рисунок 6).

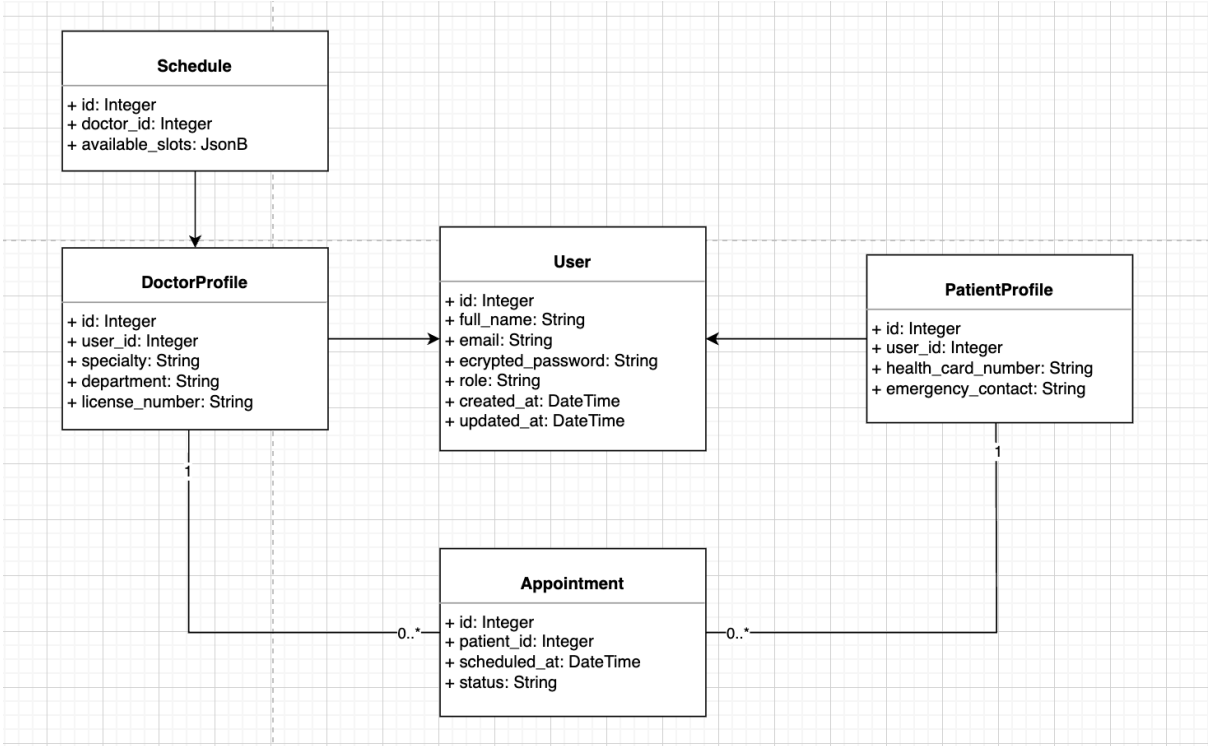

Рисунок 6 - *UML діаграма зв'язків*

Авторська розробка

REST API інтерфейс модуля

Модуль запису на прийом реалізує REST API, який забезпечує взаємодію між пацієнтами, лікарями та адміністративними системами. API орієнтований на сучасні принципи REST-архітектури: чітка структура URI, використання стандартних HTTP-методів (GET, POST, PATCH, DELETE) та передбачувані статус-коди. Це дозволяє легко інтегрувати модуль із внутрішніми або зовнішніми медичними платформами, зокрема електронними медичними записами (EHR), системами онлайн-запису, страхуванням та мобільними застосунками (таблиця 1).

Ендпоінт GET /doctors/:id/schedule дозволяє пацієнту або зовнішній системі отримати доступні часові слоти певного лікаря. На підставі цих даних фронтенд або партнерська система формує інтерфейс вибору дати та часу для запису.

Таблиця 1 – Опис основних ендпоінтів

Метод	Шлях	Призначення
GET	/doctrors/:id/schedule	Отримання актуального розкладу лікаря
GET	/users/:id/appointments	Отримання всіх записів користувачів
POST	/appointments	Створення нового запису на прийом
PATCH	/appointments/:id	Зміна параметрів запису
DELETE	/appointments/:id	Видалення запису

Авторська розробка


```
{
 "doctor_id":45,
 "timezone":"America/Toronto",
 "available_slots":[
  "2025-04-20T09:00:00Z",
  "2025-04-20T10:00:00Z"
 ]
}
```


Запит GET /users/:id/appointments дозволяє отримати всі візити, пов’язані з користувачем, незалежно від його ролі. Якщо користувач — пацієнт, API поверне список записів до лікарів. Якщо лікар — список майбутніх і минулих прийомів.

Це дозволяє реалізувати єдиний фронтенд для обох типів користувачів і спрощує обробку на стороні клієнта.

POST /appointments — основний ендпоінт, який створює запис на прийом. Запит містить patient_id, doctor_id та scheduled_at. Система перевіряє, чи є слот доступним у моделі Schedule, і лише тоді зберігає запис.


```
{
 "patient_id":123,
 "doctor_id":45,
 "scheduled_at":"2025-04-20T10:00:00Z"
}
```

PATCH /appointments/:id використовується для зміни дати або скасування запису. Це дозволяє реалізувати зручну зміну планів, наприклад через мобільний застосунок пацієнта. При зміні часу API знову проводить перевірку на доступність нового слоту. При скасуванні змінюється статус status = cancelled, що дозволяє зберігати історію змін без фізичного видалення

DELETE /appointments/:id застосовується обмежено (наприклад, тільки адміністраторами або в тестових середовищах). У продуктивному середовищі рекомендовано не видаляти записи, а позначати їх як cancelled або archived, з міркувань безпеки та відстеження медичної історії.

Усі ендпоінти підтримують:

- автентифікацію через JWT-токени
- CORS для інтеграції з фронтендом
- OpenAPI-специфікації для автоматичної генерації клієнтів (Swagger)
- чітку систему помилок (422 для валідації, 403 для відмови в доступі, 404 для неіснуючих ресурсів)

Актуальність безпечної автентифікації особливо зростає в умовах інтеграції з хмарними платформами медичних послуг. Наприклад, у статті Smart Mutual Authentication Protocol for Cloud Based Medical Healthcare Systems Using Internet of Medical Things запропоновано смарт-протокол автентифікації для забезпечення конфіденційності пацієнтів, захисту медичних звітів і перевірки доступу в телемедичних хмарних середовищах. [3]

Стандартизований API забезпечує стабільну взаємодію із зовнішніми платформами. Наприклад:

- Мобільні застосунки можуть здійснювати запис у реальному часі.
- Портали медичних закладів можуть синхронізувати графіки лікарів.
- Телемедичні платформи можуть перевіряти доступність лікарів і бронювати онлайн-консультації.
- Страхові або аналітичні системи можуть формувати звіти на основі статусів записів.

Особливості інтеграції з іншими медичними системами

Сучасна медична екосистема складається з багатьох взаємопов'язаних сервісів: електронних медичних записів (EHR), лабораторних систем, телемедичних платформ, мобільних додатків для пацієнтів, а також зовнішніх систем страхування чи державного регулювання. Ефективне функціонування модуля запису на прийом можливе лише за умови стабільної, безпечної та стандартизованої інтеграції з цими компонентами. У цьому контексті REST API інтерфейс модуля розроблений з урахуванням відкритості до зовнішньої взаємодії, захищеності даних та відповідності вимогам інтероперабельності.

Інтеграція з EHR-системами (наприклад, Epic, Cerner, Meditech) дозволяє автоматично оновлювати статус візитів без дублювання інформації. Після завершення прийому або його скасування статус у Appointment передається до EHR-системи, яка відповідно оновлює клінічну історію пацієнта. Навпаки, EHR може передати інформацію про перенесення прийому, яка синхронізується з локальним розкладом у моделі Schedule. Важливо дотримуватись стандартів, таких як HL7 FHIR, які визначають формат обміну медичними подіями, записами й учасниками процесу.

Сторонні системи, такі як портали страхових компаній або мережі партнерських клінік, можуть використовувати API модуля для перевірки доступності лікаря, планування візитів, а також формування попередніх розрахунків вартості послуг. Це дозволяє пацієнтам здійснювати запис безпосередньо з порталу страхової компанії, отримуючи гарантію того, що вибраний лікар доступний у зазначений час. При цьому важливо забезпечити обмеження доступу до даних: такі портали можуть бачити лише необхідну інформацію — доступні слоти, ПІБ лікаря, тип консультації — без розкриття деталей інших пацієнтів чи внутрішньої логіки системи.

Мобільні додатки — один із ключових каналів доступу до системи для пацієнтів. Вони використовують API модуля для реєстрації в системі, перегляду доступних лікарів, бронювання слотів у реальному часі, а також для управління власними записами (перенесення, скасування, отримання нагадувань). Аспект

зручності та інтуїтивності інтерфейсів користувача в медичних додатках також підкреслюється в дослідженні Designing User-friendly Medical AI Applications. У ньому наголошується на важливості user experience, юзабіліті та “радості використання” як критично важливих складових при розробці медичних AI-систем, а також запропоновано універсальні принципи дизайну для таких застосунків. Особливістю мобільної інтеграції є необхідність високої швидкодії API, оптимізації запитів (через кешування або попередню агрегацію даних), а також підтримки push-нотифікацій і реального часу (через WebSocket або polling). [4]

Телемедичні сервіси все частіше інтегруються з традиційними медичними системами, створюючи гібридні формати прийомів. API модуля дозволяє телемедичним платформам резервувати слоти у розкладі лікаря для онлайн-консультацій, автоматично створювати записи (Appointment) з типом “відео”, і при необхідності — ініціювати з’єднання через сторонні сервіси (Zoom, Jitsi, proprietary WebRTC). Також можлива двостороння інтеграція: після завершення візиту телемедична платформа передає зведення до EHR або створює відповідний лог у системі.

Усі зовнішні інтеграції повинні відповідати таким критеріям:

- Аутентифікація через OAuth2 або JWT.
- Обмеження доступу через рольові політики та scope-права.
- Валідація запитів і відповідей через OpenAPI-специфікацію.
- Контроль логів та аудиту — для забезпечення юридичної відповідальності.

Забезпечення якості інтеграції

Якість інтеграції є критичним аспектом у проєктуванні будь-якої медичної інформаційної системи, особливо коли йдеться про компоненти, що взаємодіють із зовнішніми системами — мобільними застосунками, клінічними інформаційними платформами, лабораторіями, системами телемедицини та страхування. У запропонованому модулі запису на прийом ключовими засобами забезпечення стабільності інтеграції є контрактне тестування, валідація API,

семантична версіонність, автоматизація в DevOps-пайплайні, а також використання онтології для формалізації поведінки компонентів.

Контрактне тестування (contract testing) використовується для перевірки того, що API відповідає очікуванням зовнішніх клієнтів. Це особливо важливо, коли REST API використовується іншими командами або зовнішніми організаціями. Для цього використовується специфікація OpenAPI (Swagger), а також інструменти на кшталт Pact або Dredd. Вони дозволяють створювати "контракти" між сервером і клієнтом — очікувану структуру запитів і відповідей, яка автоматично перевіряється під час змін. Це знижує ризик того, що зміна на сервері порушить роботу вже існуючих інтеграцій.

Усі API відповіді проходять валідацію за схемами, які фіксують структуру об'єктів, допустимі типи полів, перелік обов'язкових значень, допустимі статуси та формати дат. Застосування суворої схемної валідації знижує ймовірність помилок, пов'язаних з неконсистентними або неповними даними при обміні. Крім цього, дотримується семантична версіонність API (наприклад, /v1/appointments, /v2/appointments), що дозволяє одночасно підтримувати декілька версій API без порушення існуючих клієнтів.

Інтеграція контрактного тестування та API-валідацій у DevOps-процеси дозволяє ще до релізу переконатися, що зміни в коді не порушують інтеграційні контракти. В пайплайні використовуються етапи для лінтингу API-специфікацій, запуску тестів з мок-серверами, а також емуляції клієнтських запитів. Це значно скорочує час на ручне тестування інтеграцій і знижує ризики регресії.

Для формалізації взаємодії між компонентами системи використовується онтологічна модель користувачів, де ролі (patient, doctor) та їхні допустимі дії описані у вигляді концептів і зв'язків. Подібні підходи до онтологічного моделювання використовуються і в сучасних дослідженнях у галузі медичної освіти. Наприклад, у статті A Medical Ontology Informed User Experience Taxonomy to Support Co-creative Workflows for Authoring Mixed Reality Medical Education Spaces описано реалізацію UX-таксономії для XR-ресурсів, яка базується на SKOS-онтологіях і дозволяє структурувати цифрові активи

медичного призначення для подальшого повторного використання. На її основі реалізуються додаткові перевірки в API — наприклад, користувач із роллю patient не може доступати до Schedule або змінювати Appointment інших пацієнтів. Таким чином, навіть при інтеграції з неконтрольованими зовнішніми системами дотримуються базові обмеження доступу на концептуальному рівні [1]

Автоматизований підхід до перевірки інтеграції між системами

Для перевірки сумісності компонент окремих веб-серверів запропонуймо CI/CD конвеєр для виконання тестів паралельно. У даній статті зосередимося саме на тестуванні сумісності компонент. За своєю суттю, це виконання модуля із тестами, що базуються на методах підтримки прийняття рішень, та надсилання (виведення) результатів для розробників у зручній формі.

Методи підтримки прийняття рішень — це підходи, що сприяють формуванню обґрунтованих рішень шляхом збору, оброблення та аналізу отриманих даних. Залежно від характеру задачі та якості доступної інформації, ці методи можна поділити на на п’ять основних груп [11].

До першої групи належать методи, для яких відома функціональна залежність загальної корисності альтернатив від оцінок за окремими критеріями. Друга група охоплює компенсаційні підходи, що дозволяють урівноважити значення критеріїв задля визначення переваг між альтернативами. Існують також групи, що включають методи порогів порівняння, аксіоматичні методи та діалогові методи, але усі вони є складнішими у реалізації, адже потребують формальних правил порівнянь критеріїв, або взаємодії із експертом. Методи зважених сум та добутків належать до прямих підходів у багатокритеріальному аналізі та використовуються для оцінювання і порівняння альтернатив на основі їхніх критеріїв. Ці методи передбачають агрегування оцінок за окремими критеріями в комплексну оцінку, яка дозволяє ранжувати або вибирати найкращу альтернативу.

У нашому прикладі використаємо найконсервативніший варіант, а саме метод зважених сум. Це зроблено із метою надавати зрозумілі оцінки команді та для зменшення впливу одних полів на загальний результат. Вважатимемо що усі

вони рівнозначні між собою.

Основою методу зважених сум є лінійне агрегування оцінок альтернатив за критеріями з урахуванням їхніх вагових коефіцієнтів. Формула методу зважених сум:

$$u_i = \sum m w_j \cdot q_{ij}, \quad (1)$$

Джерело: [15]

де, u_i – комплексна оцінка корисності альтернативи x_i; w_j – вага критерію Q_j, яка відображає його відносну важливість; q_{ij} – оцінка альтернативи xi за критерієм Q_j ; m – кількість критеріїв.

Особливістю даного методу є те, що переваги альтернатив визначаються шляхом підсумовування їх оцінок, з урахуванням вагових коефіцієнтів критеріїв. Ваги можуть бути задані експертним шляхом або визначені на основі попереднього аналізу даних. Значення кожного критерію розраховується незалежно від інших, що спрощує реалізацію методу. Метод зважених сум є інтуїтивно зрозумілим і добре підходить для задач, у яких усі критерії підлягають нормалізації або можуть бути приведені до єдиної шкали оцінювання. Разом з тим, його суттєвим обмеженням є відсутність урахування взаємного впливу між критеріями, що в разі наявності залежностей між ними може призводити до викривлення загальної оцінки альтернатив.

Метод зважених добутків відрізняється від методу зважених сум тим, що ваги критеріїв у ньому виступають як степені, а не просто коефіцієнти. Такий підхід дозволяє краще враховувати взаємодію між критеріями: якщо хоча б один з них має низьке значення, це помітно знижує загальну оцінку альтернативи. Метод особливо корисний у тих випадках, коли критерії не є незалежними, і взаємний вплив між ними має значення. Крім того, він дозволяє підсилити вплив важливіших показників, надаючи їм більшу вагу. Водночас, значна різниця між оцінками окремих критеріїв може суттєво вплинути на підсумковий результат, що іноді спотворює загальну картину. Формула методу зважених добутків:

$$u_i = \prod m(q_{ij}) w_j, \quad (2)$$

Джерело: [15]

де, u_i – комплексна оцінка корисності альтернативи x_i; w_j – вага критерію Q_j, яка визначає його відносну важливість; q_{ij} – оцінка альтернативи xi за критерієм Q_j; m – кількість критеріїв.

При оцінці сумісності використаємо три оцінки

- Структурна – оцінює відповідність між будовою тіла запиту/відповіді системи-постачальника та класом системи-споживача, а саме наявність в них полів із такими ж або схожими типами. Для цієї перевірки використовуємо основні типи JSON: string (текстовий рядок), number (число), array (масив). Даний підхід дозволяє мінімізувати вплив мов програмування та технологій на оцінку сумісності
- Обмежень (допустимих значень) – оцінка відповідності обмежень що накладені на відповідні поля. Наприклад, поле має бути не пустим та мати кількість символів більше або рівне за 50
- Логічна - оцінює відповідність структури концепту онтології та класу. Дана оцінка базується на структурних методах вирівнювання онтологій. [14, 5]

На сьогодні не існує спеціалізованого інструменту, призначеного саме для перевірки сумісності онтологій між двома різними системами або для оцінки відповідності онтології структурі конкретної бізнес-системи. Наявні засоби, такі як Protégé, дозволяють виконувати валідацію онтології окремо від прикладної системи за допомогою вбудованих reasoner-ів. Також використовується SHACL, який дає змогу перевіряти відповідність онтології певним структурним вимогам [9, 10].

Кожна оцінка являє собою числове значення в межах [0…1], де 0 – повна несумісність, а 1 – повна сумісність. Будь які значення між 0 та 1 вважають частковою сумісністю.

Для того щоб допомогти команді приймати рішення необхідно перетворити числові оцінки в одну загальну на основі правил що моделюють мисленєву діяльність людини. Для цього було використано нечіткий логічний висновок Мамдані [6]. Для роботи цього методу були сформульовані наступні правила:

- ЯКЩО структурна оцінка дуже добра і оцінка обмежень дуже добра і

логічна оцінка дуже добра, ТО сумісність є дуже хорошою

- ЯКЩО структурна оцінка добра і оцінка обмежень дуже добра і логічна оцінка погана, ТО сумісність є поганою
- ЯКЩО структурна оцінка дуже добра і оцінка обмежень дуже добра і логічна оцінка погана, ТО сумісність потребує додаткової роботи

Іншою перевагою даного підходу є те, що він підтримує нечітку оцінку як вхідний параметр. Наприклад, можна налаштувати даний алгоритм щоб він використовував наступні оцінки:

а) Структурна оцінка = 0.65

б) Оцінка допустимих значень = 0.8

в) Логічна оцінка = потребує додаткової сумісності

Цей підхід є виправданим тоді, коли написання коду перевірки є трудоємним, але доступна оцінка експерта.

Експеримент

Для експерименту була розроблена онтологія що відповідає структурі класів (рисунок 7). [13]

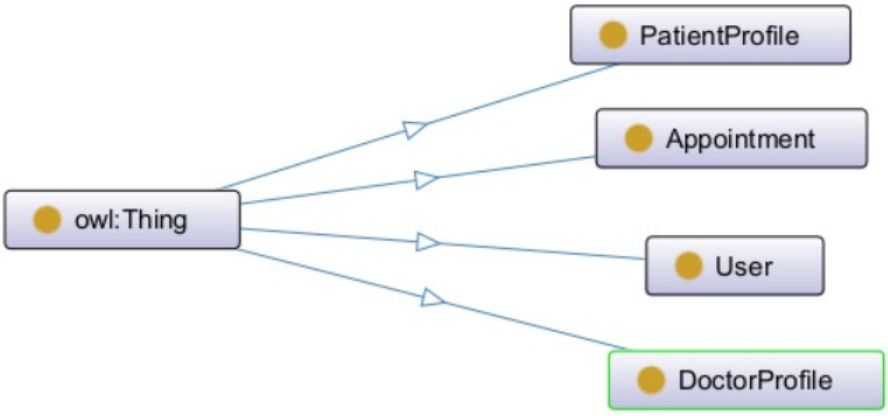

Рисунок 7 – Представлення створеної онтології у Protege

Авторська розробка

А також класи, що представляють систему-споживача: UserEntity та PatientData. Були сформовані два правила сумісності: “User is UserEntity” та “PatientProfile is PatientData”. Для демонстрації роботи були сформовані усі необхідні файли (контракти та онтологія), а також використаний StopWatch

таймер, що надається фреймворком Spring. Результат експерименту зображений на рисунку 8. Зауважмо, що у реально робочій системі консольне логування використовуватися не буде, а результати будуть формуватися як частина звіту для команди розробників.

```
================================================
Processing compatibility check: User is UserEntity
Total structure assessment: 1.0
Total value assessment: 1.0
Total logical assessment: 0.6666666666666666

Fuzzy logic result = good

================================================
Processing compatibility check: PatientProfile is PatientData
Total structure assessment: 1.0
Total value assessment: 1.0
Total logical assessment: 1.0

Fuzzy logic result = good

StopWatch 'Compatibility Check': 0.2676869 seconds
-------------------------------------------------
Seconds       %       Task name
-------------------------------------------------
0.2458958     92%     Processing compatibility check: User is UserEntity
0.0217911     08%     Processing compatibility check: PatientProfile is PatientData
```

Рисунок 8 – *Логування результатів виконання програми*

Авторська розробка

На основі вищенаведеної оцінки розробники можуть зробити висновок, що загальний час перевірки двох контрактів на відповідність структурі запиту та класу, а також на відповідність класу та концепту склав 0.26 секунди, що швидше ніж інтеграційні тести, що перевіряють кінцеві точки (endpoints) послідовно. Важливою перевагою розроблюваних тестів є невикористання бази даних, доступ до якої та виконання запитів можуть бути найповільнішою частиною тестів. Крім того, перевірка сумісності з використанням онтологічного підходу дозволяє переконатися, що система-споживач відповідає бізнес-структурі системи-постачальника з точки зору експертів у області, а не розробників. Дані тести, надають оцінку сумісності, яка виступає допоміжною у прийнятті рішення про розгортання нової версії продукту. Наприклад, із результатів вище можна зробити висновок, що клас UserEntity системи-споживача не повністю відповідає концепту і варто додатково переглянути їхню сумісність, але повна структурна

сумісність між запитом та класом надають інформацію, що критичних змін не виявлено і сумісність є повною. З чого можна зробити висновок, що не варто відкладати випуск нової версії продукту, адже інтеграція між системами працюватиме без змін, проте необхідно запланувати перегляд відповідності концепту та класу. Використання нечіткої логіки надає короткий результат, який можна інтерпретувати наступним чином: якщо всі оцінки добрі (good), то немає причини зупиняти випуск продукту.

Результати виконання тестів можуть бути використані для перевірки контрольних точок якості, що відповідає загальноприйнятим практикам у межах стандартного DevOps-процесу або ітеративної моделі, такої як Agile/Scrum [12]. Відповідно до цього, подальшими діями можуть стати або випуск нової версії програмного продукту, або усунення виявлених дефектів. Якщо ж тести були ініційовані з боку системи-постачальника і в ході їх виконання виявлено помилки, доцільно здійснити перепланування поточної ітерації з урахуванням необхідності їх усунення, а також із врахуванням узгодженого графіка релізу постачальника.

Висновок

У даній статті було детально розглянуто архітектурний підхід до побудови модуля запису на прийом для медичних інформаційних системах. Запис до лікаря реалізовано через модель прийому з перевіркою узгодженості із розкладом лікаря. Також була розглянуто сучасні підходи інтеграція із зовнішніми платформами — мобільними додатками, телемедичними сервісами, порталами страхових компаній та системами EHR з використанням REST API, який підтримує OpenAPI-специфікацію, аутентифікацію через JWT і принципи семантичної версіонності. Були розроблені діаграми класів та приклади REST інтегрфейсів модулів.

Онтологічний підхід, представлений у даному дослідженні, сприяє побудові тестів, які є організованими та зручними в підтримці. Він також покращує процес оцінювання сумісності між компонентами системи з позиції експертів, що залучені до проєктування онтології системи-постачальника. У статті також

розглянуто перевірку сумісності компонент систем із використанням механізмів нечіткого логічного виведення методом Мамдані, що дає змогу формувати рекомендації для команди інженерів і керівників проєктів.

Був проведений експеримент із використанням онтології (у форматі RDF файлу) та класів системи-споживача, який демонструє перспективність використання даного підходу у тестуванні та надає короткий результат виконання перевірки команді розробників.

Перспективним напрямом подальших досліджень є створення гнучких програмних і тестових модулів, а також розробка програмних інтерфейсів, орієнтованих на нетехнічних користувачів, зокрема у сфері медичних інформаційних систем.

Література.

1. Antoniou P.E., Chondrokostas E., Bratsas Ch., Filippidis P.-M., Bamidis P.D. A Medical Ontology Informed User Experience Taxonomy to Support Co-creative Workflows for Authoring Mixed Reality Medical Education Spaces // 7th International Conference of the Immersive Learning Research Network (iLRN). – 2021. – DOI: 10.23919/iLRN52045.2021.9459388.

2. Fajrin H.R., Pramudya T., Supriyadi K. The Development of a Medical Equipment Inventory Information System // 2024 IEEE International Conference on Artificial Intelligence and Mechatronics Systems (AIMS). – 2024. – DOI: 10.1109/AIMS61812.2024.10512877.

3. Deebak B.D., Al-Turjman F. Smart Mutual Authentication Protocol for Cloud Based Medical Healthcare Systems Using Internet of Medical Things // IEEE Journal on Selected Areas in Communications. – 2021. – T. 39. – № 2. – DOI: 10.1109/JSAC.2020.3020599.

4. Wiebelitz L., Schmid P., Maier T., Volkwein M. Designing User-friendly Medical AI Applications – Methodical Development of User-centered Design Guidelines // 2022 IEEE International Conference on Digital Health (ICDH). – 2022. – DOI: 10.1109/ICDH55609.2022.00011.

5. Romero M.M., Vázquez Naya J.M., Pereira Loureiro J., Ezquerra N. Ontology Alignment Techniques // University of A Coruña; Georgia Institute of Technology. – [Електронний ресурс]. – Режим доступу: https://www.researchgate.net/profile/Marcos-Martinez-Romero/publication/236145047_Ontology_alignment_techniques/links/02e7e52fa85b85bba2000000/Ontology-alignment-techniques.pdf, вільний.

6. Желдак Т.А., Коряшкіна Л.С., Ус С.А. Нечіткі множини в системах управління та прийняття рішень: навч. посібник. – Дніпро: НТУ «Дніпровська політехніка», 2020. – УДК 510.644.4:004.8:519.816.

7. Postman. 2024 State of the API Report [Електронний ресурс]. – Режим доступу: https://voyager.postman.com/doc/postman-state-of-the-api-report-2024.pdf, вільний. – Дата звернення: [17.04.2025].

8. Postman. 2023 State of the API Report [Електронний ресурс]. – Режим доступу: https://www.postman.com/state-of-api/api-global-growth/#api-global-growth, вільний. – Дата звернення: [17.04.2025].

9. Protégé Project. Reasoner Preferences. Protégé 5 Documentation [Електронний ресурс]. – Режим доступу: https://protegeproject.github.io/protege/preferences/reasoner/, вільний.

10. World Wide Web Consortium (W3C). Shapes Constraint Language (SHACL) [Електронний ресурс] / W3C Recommendation, 20 July 2017. – Режим доступу: https://www.w3.org/TR/shacl/, вільний. – Дата звернення: [10.04.2025].

11. Демиденко М.А. Системи підтримки прийняття рішень: навч. посіб. – Дніпро: Нац. гірн. ун-т, 2016. – [Електронний ресурс]. – Режим доступу: http://kist.ntu.edu.ua/textPhD/sppr2.pdf, вільний. – Дата звернення: [15.04.2025].

12. Metridev. Quality Gates: Everything You Need to Know [Електронний ресурс]. – Режим доступу: https://www.metridev.com/metrics/quality-gates-everything-you-need-to-know/, вільний. – Дата звернення: [12.04.2025].

13. Басюк Т.М., Досин Д.Г., Литвин В.В. Онтологічний інжиніринг: навч. посіб. – Львів: Національний університет «Львівська політехніка», 2016..

14. Ouali I., Ghozzi F., Taktak R., Hadj Sassi M.S. Ontology Alignment using

Stable Matching // Proceedings of the 23rd International Conference on Knowledge-Based and Intelligent Information & Engineering Systems. – Procedia Computer Science, 2019. – Vol. 159. – P. 746–755. – DOI: 10.1016/j.procs.2019.09.230.

15. E. Triantaphyllou, B. Shu, S. Nieto Sanchez, and T. Ray. Multi-criteria decision making: An operations research approach // Encyclopedia of Electrical and Electronics Engineering / ed. by J.G. Webster. – New York: John Wiley & Sons, 1998. – Vol. 15. – P. 175–186.

References

1. Antoniou P.E., Chondrokostas E., Bratsas Ch., Filippidis P.-M., Bamidis P.D. A Medical Ontology Informed User Experience Taxonomy to Support Co-creative Workflows for Authoring Mixed Reality Medical Education Spaces // 7th International Conference of the Immersive Learning Research Network (iLRN). – 2021. – DOI: 10.23919/iLRN52045.2021.9459388.

2. Fajrin H.R., Pramudya T., Supriyadi K. The Development of a Medical Equipment Inventory Information System // 2024 IEEE International Conference on Artificial Intelligence and Mechatronics Systems (AIMS). – 2024. – DOI: 10.1109/AIMS61812.2024.10512877.

3. Deebak B.D., Al-Turjman F. Smart Mutual Authentication Protocol for Cloud Based Medical Healthcare Systems Using Internet of Medical Things // IEEE Journal on Selected Areas in Communications. – 2021. – T. 39. – № 2. – DOI: 10.1109/JSAC.2020.3020599.

4. Wiebelitz L., Schmid P., Maier T., Volkwein M. Designing User-friendly Medical AI Applications – Methodical Development of User-centered Design Guidelines // 2022 IEEE International Conference on Digital Health (ICDH). – 2022. – DOI: 10.1109/ICDH55609.2022.00011.

5. Romero M.M., Vázquez Naya J.M., Pereira Loureiro J., Ezquerra N. Ontology Alignment Techniques // University of A Coruña; Georgia Institute of Technology. – [Electronic resource]. – Available at: https://www.researchgate.net/profile/Marcos-Martinez-Romero/publication/236145047_Ontology_alignment_techniques/links/02e7e52fa85b85bba2000000/Ontology-alignment-techniques.pdf, вільний.

6. Zheldak T.A., Koryashkina L.S., Us S.A. Fuzzy Sets in Control Systems and Decision Making: Textbook. – Dnipro: NTU "Dniprovska Politechnika", 2020. – УДК 510.644.4:004.8:519.816.

7. Postman. 2024 State of the API Report [Electronic resource]. – Available at: https://voyager.postman.com/doc/postman-state-of-the-api-report-2024.pdf, free access. – Accessed: [17.04.2025].

8. Postman. 2023 State of the API Report [Electronic resource]. – Available at: https://www.postman.com/state-of-api/api-global-growth/#api-global-growth, free access. – Accessed: [17.04.2025].

9. Protégé Project. Reasoner Preferences. Protégé 5 Documentation [Electronic resource]. – Available at: https://protegeproject.github.io/protege/preferences/reasoner/, free access.

10. World Wide Web Consortium (W3C). Shapes Constraint Language (SHACL) [Electronic resource] / W3C Recommendation, 20 July 2017. – Available at: https://www.w3.org/TR/shacl/, free access. – Accessed: [10.04.2025].

11. Demydenko M.A. Decision Support Systems: Textbook. – Dnipro: National Mining University, 2016. – [Electronic resource]. – Available at: http://kist.ntu.edu.ua/textPhD/sppr2.pdf, free access. – Accessed: [15.04.2025].

12. Metridev. Quality Gates: Everything You Need to Know [Electronic resource]. – Available at: https://www.metridev.com/metrics/quality-gates-everything-you-need-to-know/, free access. –

Accessed: [12.04.2025].

13. Basiuk T.M., Dosyn D.H., Lytvyn V.V. Ontological Engineering: Textbook. – Lviv: Lviv Polytechnic National University, 2016.

14. Ouali I., Ghozzi F., Taktak R., Hadj Sassi M.S. Ontology Alignment using Stable Matching // Proceedings of the 23rd International Conference on Knowledge-Based and Intelligent Information & Engineering Systems. – Procedia Computer Science, 2019. – Vol. 159. – P. 746–755. – DOI: 10.1016/j.procs.2019.09.230.

15. E. Triantaphyllou, B. Shu, S. Nieto Sanchez, and T. Ray. Multi-criteria decision making: An operations research approach // Encyclopedia of Electrical and Electronics Engineering / ed. by J.G. Webster. – New York: John Wiley & Sons, 1998. – Vol. 15. – P. 175–186.

Abstract. *This paper presents an architectural approach to designing a medical appointment module within healthcare information systems, with a focus on unified user modeling, integration stability, and automated component compatibility testing. The proposed model allows a user to act as either a patient or a doctor, extended via corresponding profiles containing role-specific attributes. Medical visits are managed through an appointment model that verifies consistency with the doctor's schedule. Integration with external platforms—such as mobile applications, telemedicine services, insurance portals, and EHR systems—is facilitated through a REST API that supports OpenAPI specifications, JWT authentication, and semantic versioning principles.*

Special attention is given to automated compatibility testing across systems. A mechanism is proposed to evaluate structural, constraint-based, and logical consistency between components, based on an ontological representation of data and the Mamdani fuzzy inference method. The weighted sum method is used to aggregate results into a single compatibility score, taking into account the relative importance of each criterion. The article provides an implementation example using RDF-based ontology, consumer-side data classes, and an experiment measuring execution time—demonstrating the efficiency of this approach compared to conventional endpoint-based integration tests. This method improves transparency in release decision-making and can be embedded into DevOps pipelines where ontological alignment and system interoperability are essential.

Key words*: healthcare information systems, appointment scheduling, REST API, ontology, component compatibility, contract testing, fuzzy logic, user profiles, EHR integration, test automation, decision-making, ontological approach*

Статтю надіслано: 19.04.2025 г.