УДК 004.5

# APPLICATION PROGRAMMING INTERFACE (API): ОСНОВИ ТА СУЧАСНІ ПІДХОДИ

**Мандрика Юрій**

**Науковий керівник: канд. фіз.-мат. наук, доц. Лисенко І.М.**

*Ніжинський державний університет імені Миколи Гоголя*

*м. Ніжин, Україна*

*API (Application Programming Interface) – це набір правил та протоколів, що дозволяють різним програмним додаткам взаємодіяти між собою. Концепція API виникла ще в 1960-х роках, коли з'явилася потреба стандартизувати спосіб обміну інформацією між програмами. Перші API були дуже обмеженими, однак з розвитком технологій їх можливості значно розширились, і сьогодні API є основою для створення гнучких та масштабованих додатків, особливо в контексті веб-сервісів [1].*

*Ключові слова: API, REST, JSON-RPC, SOAP, мікросервісна архітектура, DevOps, CI/CD, автоматизація, токени, JWT, автентифікація, авторизація, безпека даних, інтеграція, хмарні сервіси, обробка даних, конфіденційність.*

# APPLICATION PROGRAMMING INTERFACE (API): FUNDAMENTALS AND MODERN APPROACHES

**Y. Mandryka**

**Scientific supervisor: Candidate of Physics and Mathematics Sciences, Associate Professor Lysenko I.M.**

*Nizhyn Mykola Gogol State University, Nizhyn, Ukraine*

*API (Application Programming Interface) is a set of rules and protocols that enable different software applications to interact with each other. The concept of API originated in the 1960s when the need arose to standardize the exchange of information between programs. The first APIs were very limited, but with technological advancements, their capabilities have significantly expanded. Today, APIs are fundamental for creating flexible and scalable applications, especially in the context of web services [1].*

*Key words: API, REST, JSON-RPC, SOAP, microservice architecture, DevOps, CI/CD, automation, tokens, JWT, authentication, authorization, data security, integration, cloud services, data processing, confidentiality.*

**Постановка проблеми.** API стали ключовими у програмному забезпеченні, дозволяючи створювати сервіси, які інтегруються з іншими додатками, обмінюються даними та автоматизують процеси. Вони забезпечують зв’язок між веб-додатками, мобільними додатками, серверами та базами даних. Наприклад, мобільний додаток для прогнозу погоди використовує API для отримання даних з сервера.

**Аналіз досліджень і публікацій.** API бувають відкритими (публічними) та закритими (внутрішніми). Публічні API, як у Facebook або Twitter, дозволяють стороннім розробникам інтегрувати свої сервіси з платформами. Внутрішні API використовуються в компаніях для взаємодії між компонентами додатків, забезпечуючи конфіденційність та безпеку даних [1].

API широко використовуються для мікросервісної архітектури, що є ключовою у великих корпоративних системах. Вони дозволяють розбивати додаток на незалежні компоненти (мікросервіси), які взаємодіють через API, що забезпечує гнучкість та масштабованість. API також важливі у DevOps, автоматизуючи розгортання, тестування та моніторинг додатків. Інструменти CI/CD за допомогою API можуть автоматично запускати деплой, тести і збирати логи, підвищуючи ефективність розробки.

API також використовуються для обробки великих обсягів даних, інтеграції з базами даних, хмарними сервісами або платформами машинного навчання. Наприклад, фінансові платформи через API підключаються до банківських систем для отримання актуальної інформації про курси валют чи рахунки.

Таким чином, API є не лише засобом комунікації між додатками, але й важливим інструментом, що забезпечує інтеграцію, автоматизацію, захист та ефективність у розробці сучасного програмного забезпечення.

**Мета статті.** В статті розглянемо типи API та їхнє застосуванням у різних сферах, зокрема для інтеграції між додатками, автоматизації процесів, обробки даних та підвищення безпеки, звертаючи увагу на сучасні практики та

інструменти, зокема мікросервісну архітектуру, DevOps, CI/CD, використання токенів для автентифікації та авторизації.

**Виклад основного матеріалу (результатів) дослідження.** На сьогодні існує кілька типів API, кожен з яких має свою специфіку та сферу застосування. Розглянемо найпопулярніші з них:

– REST (Representational State Transfer): Найпопулярніший підхід для створення API, що використовує HTTP-запити (GET, POST, PUT, DELETE) для роботи з ресурсами. REST забезпечує простоту інтеграції з різними платформами, підтримуючи формати JSON та XML, що робить його зручним для масштабованих веб-додатків [2].

– JSON-RPC (Remote Procedure Call): Протокол, що використовує JSON для виклику процедур на віддаленому сервері. JSON-RPC підходить для виклику конкретних функцій, де важлива швидкість і простота, особливо для автоматизації та обробки транзакцій [4].

– SOAP (Simple Object Access Protocol): Використовує XML для передачі даних, підтримуючи надійність та безпеку (WS-Security). SOAP підходить для корпоративних додатків, де критично важливо захистити конфіденційні дані [5].

Для тестування та роботи з API широко використовується cURL – це команда для виконання HTTP-запитів з командного рядка, яка дозволяє швидко тестувати API. За допомогою cURL можна виконувати запити (GET, POST, PUT, DELETE), перевіряти автентифікацію за токенами та передавати файли. Це зручний інструмент для діагностики API та інтеграції у розробку.

REST та JSON-RPC є популярними підходами до створення API, однак вони мають свої відмінності та особливості застосування:

– REST: Орієнтований на роботу з ресурсами через HTTP методи (GET, POST, PUT, DELETE). REST-API забезпечує гнучкість та масштабованість, що робить його зручним для інтеграції з клієнтами та системами, наприклад, для інтернет-магазинів чи систем керування контентом.

– JSON-RPC: Використовує JSON для виклику методів без складної структури ресурсів. Підходить для швидких викликів у спеціалізованих

системах, таких як автоматизація чи обробка транзакцій. Однак JSON-RPC менш гнучкий і обмежений у масштабованості порівняно з REST.

Розглянемо приклади запитів на основі Kanban-системи. REST API надає можливість взаємодії з ресурсами Kanban-дошки, такими як задачі, проєкти та користувачі. Для виконання запитів можна використовувати різноманітні інструменти, наприклад Postman [3]. Наприклад, запит на отримання всіх задач на дошці:GET запит: GET /api/tasks.

Цей запит повертає список усіх задач у форматі JSON:

```
 [
    {
        "id": 1,
        "title": "Design UI for login page",
        "description": "Create wireframe and initial design",
        "status": "In Progress",
        "priority": "High"
    },
    {
        "id": 2,
        "title": "Set up database schema",
        "description": "Define tables for users, tasks, and projects",
        "status": "To Do",
        "priority": "Medium"
    }
]
```

У випадку з JSON-RPC, API працює не з ресурсами напряму, а з функціями, які виконуються на сервері. Наприклад, додавання нової задачі на дошку:

```
{
    "jsonrpc": "2.0",
    "method": "createTask",
    "params": {
        "title": "Implement drag and drop for tasks",
        "description": "Allow users to move tasks between columns",
        "status": "To Do",
        "priority": "High"
    },
    "id": 1
}
```

Отримаємо відповідь, схожу з попередньою:

```
{
    "jsonrpc": "2.0",
    "result": {
        "id": 3,
        "title": "Implement drag and drop for tasks",
        "status": "To Do",
        "priority": "High"
    },
    "id": 1
}
```

JSON-RPC дозволяє викликати методи напряму, що зручно для викликів конкретних функцій або команд. У цьому прикладі ми створюємо нову задачу на Kanban-дошці, не маніпулюючи ресурсами напряму, як у REST, а викликаючи метод createTask.

Виконаємо запит для створення нової задачі на Kanban-дошці через REST API та с застосуванням cURL:

```
curl -X POST http://example.com/api/tasks \
-H "Content-Type: application/json" \
-d '{
    "title": "Review code for bugs",
    "description": "Check all modules for potential bugs",
    "status": "In Progress",
    "priority": "Medium"
}'
```

У цій команді ми надсилаємо POST-запит для створення нової задачі з вказаними полями, такими як title, description, status та priority. cURL дозволяє швидко і зручно тестувати роботу API без необхідності створення повного інтерфейсу або написання коду для клієнта.

Слід зазначити, що REST API підтримує передачу файлів за допомогою методу POST з використанням multipart/form-data, що є стандартом у веб-розробці та широко підтримується. JSON-RPC не підтримує передачу файлів напряму, тому для цього використовують обхідні методи, наприклад, кодування файлів у Base64.

У сучасних API для безпеки та контролю доступу використовують токени, найпоширеніший формат – JWT (JSON Web Token). JWT забезпечує авторизацію, дозволяючи API перевіряти запити та надавати доступ до ресурсів лише автентифікованим клієнтам. Запити до захищеного API включають заголовок Authorization: Bearer <token>, що дозволяє визначити рівень доступу клієнта.

**Висновки та перспективи подальших пошуків у напрямі дослідження.** API стали невід'ємною складовою сучасних програмних екосистем, забезпечуючи інтеграцію, автоматизацію та масштабованість застосунків. Завдяки таким підходам, як REST, JSON-RPC і SOAP, API адаптуються до потреб бізнесу та дозволяють створювати гнучкі й безпечні рішення, які легко інтегруються з різними системами. Перспективи подальших досліджень у цьому напрямі включають розвиток нових засобів для підвищення безпеки та конфіденційності, оптимізацію обміну даними для роботи з великими обсягами інформації, а також удосконалення інструментів DevOps для більш ефективної автоматизації процесів розробки й розгортання API.

**Список літератури**

1. Fielding M. REST: Architectural Styles and the Design of Network-based Software Architectures. Doctoral dissertation. University of California. Irvine, 2000. URL: https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm (Дата звернення 25.10.2024).

2. Richardson A. API Testing Tips: REST API Testing Strategy. DZone, 2024. URL: https://dzone.com/articles/api-testing-tips-rest-api-testing-strategy (Дата звернення 25.10.2024).

3. Postman, API Testing – Introduction to Postman. URL: https://www.postman.com/api-testing/ (Дата звернення 25.10.2024).

4. What is JSON-RPC? JSON-RPC Official Website. URL: https://www.jsonrpc.org/ (Дата звернення 25.10.2024).

5. SOAP Web Services. W3Schools. URL: https://www.w3schools.com/xml/xml_soap.asp (Дата звернення 25.10.2024).

**Відомості про автора:**

**Мандрика Юрій Валерійович** – студент магістратури ІІ курсу, спеціальність 122 Комп'ютерні науки, ННІ природничо-математичних, медико-біологічних наук та інформаційних технологій Ніжинського державного університету імені Миколи Гоголя, +380671550600, yuriy.mandryka@gmail.com