https://doi.org/10.31891/2219-9365-2025-84-35
УДК 004.42:004.738.5:004.77

ТЕЛЕЖЕНКО Денис
Приватний вищий навчальний заклад «Європейський університет»
https://orcid.org/0000-0002-8377-8517
ПОЗНЯК Анатолій
Приватний вищий навчальний заклад «Європейський університет»
https://orcid.org/0009-0006-8071-0556

ООП ЯК КОНТРАКТ МІЖ БЕКЕНДОМ І МОБІЛЬНИМ ЗАСТОСУНКОМ

У статті розглядається об'єктно-орієнтоване програмування (ООП) як концептуальна та практична основа контрактної взаємодії між серверною частиною (backend) і мобільним застосунком. Особливу увагу приділено ролі моделей даних, інтерфейсів, DTO, принципів інкапсуляції та наслідування у забезпеченні стабільності API, зменшенні залежностей між компонентами та спрощенні супроводу програмних систем. Представлено аналіз архітектурних підходів, що базуються на ООП, приклади типових помилок при порушенні контрактів, а також рекомендації щодо проєктування узгоджених моделей взаємодії між клієнтом і сервером. Результати дослідження можуть бути корисними для розробників мобільних застосунків, backend-інженерів та архітекторів програмних систем.

Ключові слова: об'єктно-орієнтоване програмування, backend, мобільний застосунок, API, контракт, DTO, архітектура, клієнт-серверна взаємодія.

TELEZHENKO Denys, POZNIAK Anatolii
Private Higher Educational Institution "European University"

OBJECT-ORIENTED PROGRAMMING AS A CONTRACT BETWEEN BACKEND AND MOBILE APPLICATION

The article explores object-oriented programming (OOP) as a conceptual and practical mechanism for establishing a stable contract between backend systems and mobile applications in modern client–server architectures. In contemporary software development, mobile applications operate in close integration with server-side components through APIs, where inconsistencies in data models and interaction rules often lead to runtime errors, increased maintenance costs, and technical debt. The study emphasizes that OOP should be considered not only as an internal implementation paradigm, but also as a formal tool for defining inter-system agreements.

Special attention is given to the role of object-oriented data models, interfaces, and Data Transfer Objects (DTOs) in ensuring structural and behavioral consistency between backend and mobile clients. The principles of encapsulation, inheritance, and abstraction are analyzed in the context of API stability, reduction of coupling, and support for parallel development processes. The article examines typical architectural approaches based on OOP and highlights common integration issues caused by contract violations, such as unversioned model changes, type mismatches, and the use of weakly typed or dynamic data structures.

An experimental client–server system was designed to evaluate the impact of object-oriented contract violations on application stability. Several scenarios involving changes to backend models were analyzed, demonstrating a significant increase in critical failures when object-oriented contracts were not preserved. The results confirm that the use of DTOs and clearly defined interfaces substantially reduces integration errors and improves system reliability.

Based on the findings, the article provides practical recommendations for designing consistent interaction models between backend services and mobile applications using object-oriented principles. The proposed approach contributes to improved predictability, maintainability, and scalability of software systems and can be applied by mobile developers, backend engineers, and software architects working in Agile and DevOps environments.

Keywords: object-oriented programming, backend, mobile application, API, contract, DTO, architecture, client-server interaction.

Стаття надійшла до редакції / Received 02.11.2025
Прийнята до друку / Accepted 04.12.2025

ПОСТАНОВКА ПРОБЛЕМИ ТА ЇЇ ЗВ'ЯЗОК ІЗ ВАЖЛИВИМИ НАУКОВИМИ ЧИ ПРАКТИЧНИМИ ЗАВДАННЯМИ

Сучасні мобільні застосунки функціонують у тісній взаємодії з серверною частиною, яка відповідає за бізнес-логіку, зберігання даних, автентифікацію та інтеграцію із зовнішніми сервісами. У більшості випадків саме API виступає ключовою точкою контакту між бекендом і клієнтським застосунком. Від стабільності, узгодженості та передбачуваності цієї взаємодії безпосередньо залежить якість, масштабованість і супровід програмного продукту.

На практиці поширеною проблемою є неузгоджені зміни: оновлення моделей на бекенді без відповідних змін у мобільному застосунку, різне трактування полів даних, відсутність чітко визначених контрактів відповідальності між компонентами. Це призводить до помилок під час виконання, збільшення кількості hotfix-релізів, ускладнення тестування та зростання технічного боргу.

Об'єктно-орієнтоване програмування, яке традиційно розглядається як парадигма організації коду, у контексті клієнт-серверної архітектури набуває ширшого значення. ООП дозволяє формалізувати контракт

між бекендом і мобільним застосунком через чітко визначені моделі, інтерфейси та правила взаємодії. Саме ці контракти визначають, які дані передаються, у якому форматі, які обмеження накладаються та які гарантії надаються кожною стороною.

У контексті Agile- та DevOps-підходів, де частота змін є високою, потреба в стабільному та формалізованому контракті стає критичною. Відсутність такого контракту ускладнює паралельну розробку бекенду й мобільного клієнта, знижує ефективність командної взаємодії та негативно впливає на швидкість виходу продукту на ринок.

Дослідження ролі ООП як інструмента побудови контрактної взаємодії між бекендом і мобільним застосунком є важливим як з наукової, так і з практичної точки зору.

АНАЛІЗ ДОСЛІДЖЕНЬ ТА ПУБЛІКАЦІЙ

Аналіз сучасних наукових публікацій і прикладних досліджень свідчить, що переважна більшість робіт у галузі клієнт-серверних систем зосереджена на питаннях проєктування API, зокрема на використанні REST- та GraphQL-підходів, оптимізації HTTP-взаємодії, а також на проблемах версіонування сервісів і забезпечення зворотної сумісності. Значна увага приділяється формальним аспектам контрактів, таким як специфікації OpenAPI, документація endpoint-ів та механізми контролю змін інтерфейсів.

Водночас роль об’єктно-орієнтованого моделювання як фундаментальної основи контрактної взаємодії між бекендом і клієнтськими застосунками, зокрема мобільними, у більшості досліджень розглядається поверхнево або фрагментарно. ООП часто згадується лише як внутрішній механізм реалізації серверної логіки або клієнтських компонентів, без чіткого акценту на його значенні для узгодження моделей даних між різними частинами системи.

У працях, присвячених підходу domain-driven design (DDD), об’єктно-орієнтоване програмування розглядається насамперед як засіб точного відображення предметної області, формалізації бізнес-правил та побудови доменних моделей. Хоча DDD передбачає чітке моделювання сутностей, агрегатів і value objects, питання узгодження цих моделей між бекендом і клієнтськими застосунками, а також їх використання як стабільного контракту, часто залишаються поза межами аналізу. У результаті доменні моделі нерідко трансформуються у клієнтські представлення без чітко визначених правил відповідності.

Дослідження, пов’язані з архітектурними підходами mobile-first або backend-for-frontend (BFF), акцентують увагу на необхідності адаптації серверних API до потреб конкретних клієнтів. У таких роботах підкреслюється важливість використання DTO, view-моделей та спеціалізованих контрактів для різних типів клієнтів. Проте навіть у цьому контексті DTO часто розглядаються як технічний інструмент оптимізації передачі даних, а не як формалізований елемент об’єктно-орієнтованого контракту, що визначає стабільні правила взаємодії між системами.

Окрему групу становлять практичні публікації, технічні блоги та інженерні звіти, у яких описуються проблеми, що виникають під час використання «гнучких» структур даних, таких як Map, динамічні JSON-об’єкти або слабко типізовані відповіді API. Автори таких матеріалів відзначають зростання кількості помилок інтеграції, складність тестування та збільшення технічного боргу. Однак ці проблеми зазвичай розглядаються з позиції інструментів або фреймворків, без глибокого аналізу їхнього зв’язку з порушенням принципів об’єктно-орієнтованого програмування та відсутністю чітко визначеного контракту.

Отже, огляд літератури свідчить про наявність наукової та практичної прогалини у дослідженні ООП як засобу міжсистемної домовленості між бекендом і мобільним застосунком. Існує потреба у комплексному аналізі об’єктно-орієнтованого моделювання не лише як стилю програмування або внутрішньої архітектурної техніки, а як ключового механізму формування контрактів, що забезпечують узгодженість, стабільність і передбачуваність клієнт-серверної взаємодії.

ФОРМУЛЮВАННЯ ЦІЛЕЙ СТАТТІ

Метою статті є аналіз об’єктно-орієнтованого програмування як механізму формування контракту між бекендом і мобільним застосунком. Для досягнення поставленої мети визначено такі завдання:

- проаналізувати роль ООП-моделей у клієнт-серверній взаємодії;
- дослідити значення DTO, інтерфейсів та абстракцій як елементів контракту;
- визначити типові помилки, що виникають при порушенні об’єктних контрактів;
- сформулювати практичні рекомендації щодо проєктування узгоджених моделей між бекендом і мобільним клієнтом.

ВИКЛАД ОСНОВНОГО МАТЕРІАЛУ

У межах дослідження об’єктно-орієнтоване програмування розглядається не лише як парадигма організації програмного коду, а як формальний механізм побудови контрактної взаємодії між бекендом і мобільним застосунком. Під контрактом у даному контексті розуміється сукупність структурних, типових і поведінкових домовленостей, які визначають формат, семантику та правила обміну даними між клієнтською та серверною частинами системи.

Для дослідження було змодельовано клієнт-серверну систему, що імітує типовий мобільний застосунок для керування користувацькими профілями. Архітектура системи включала:

- backend-сервіс із REST API;
- мобільний клієнт із асинхронною обробкою відповідей сервера;
- набір ООП-моделей, DTO та інтерфейсів;
- систему логування та збору помилок.

Загальну схему досліджуваної архітектури подано на рис. 1

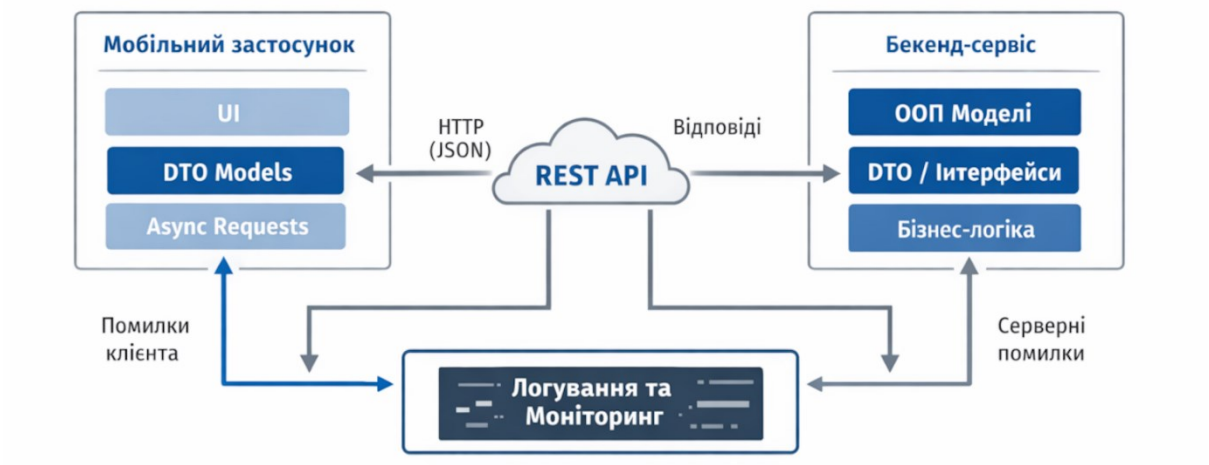


Рис. 1. Архітектура клієнт-серверної системи з ООП-контрактом

У клієнт-серверних системах модель даних виступає базовим елементом узгодження між бекендом і мобільним застосунком. З позиції ООП модель є класом, що визначає чітко формалізовану структуру об'єкта, його властивості та допустимі типи значень. У контексті міжсистемної взаємодії така модель фактично виконує роль структурного контракту.

Коли бекенд і мобільний застосунок використовують однакові або синхронізовані ООП-моделі, вони дотримуються єдиної домовленості щодо формату даних. Наприклад, клас User, який містить поля id, name, email та role, визначає очікувану структуру об'єкта користувача на обох сторонах. Будь-яка зміна цієї структури без відповідного оновлення клієнта або без версіонування API порушує контракт.

На рис. 2 зображено приклад відповідності та невідповідності ООП-моделей між бекендом і мобільним застосунком.

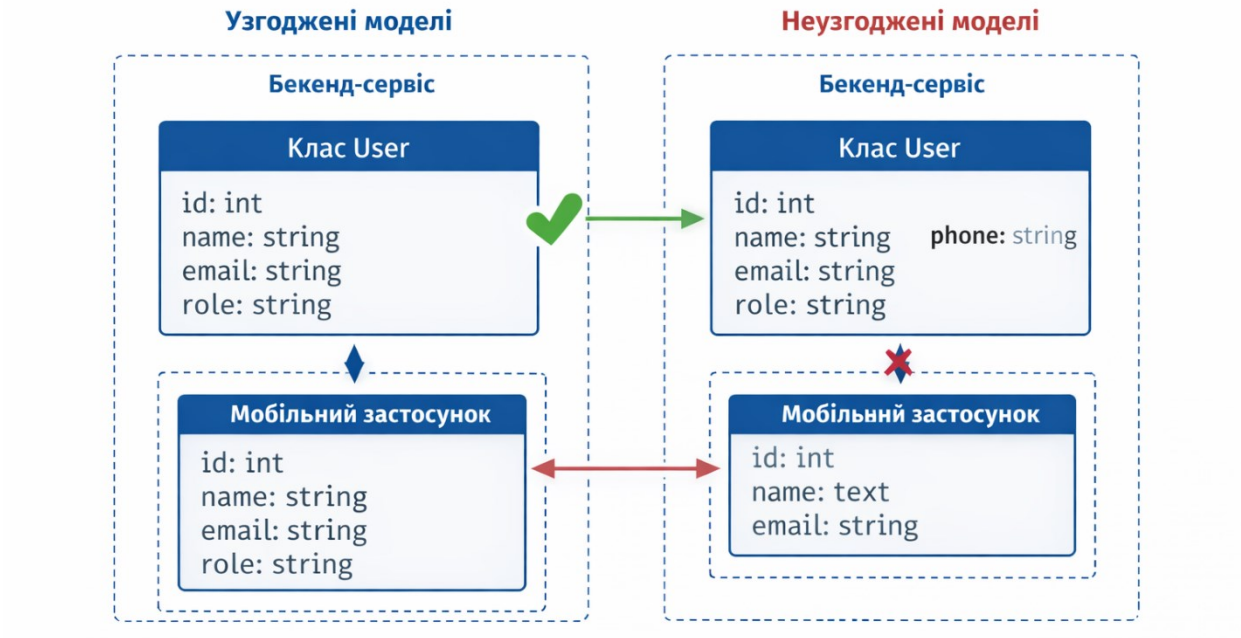
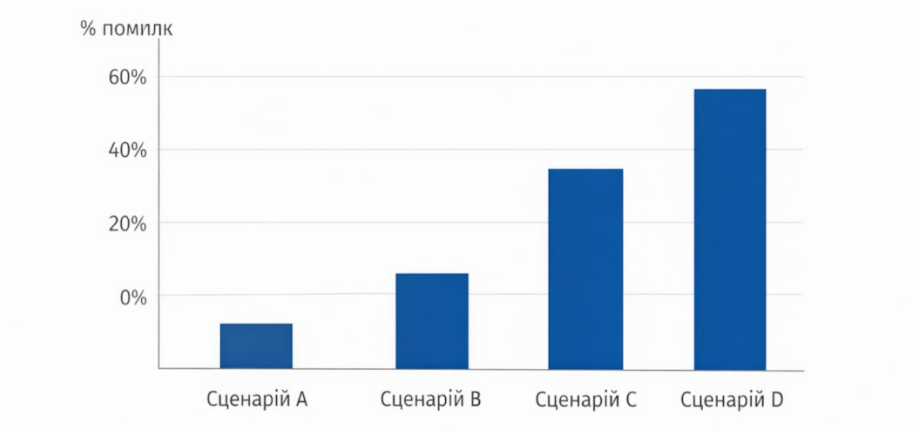


Рис. 2. Приклад узгоджених та неузгоджених ООП-моделей

Для кількісної оцінки впливу порушення ООП-контракту було проведено серію експериментів. У ході дослідження виконувалося 1000 послідовних запитів мобільного застосунку до бекенду за різних умов зміни моделей даних.

Було розглянуто такі експериментальні сценарії:

- Сценарій A: повна відповідність ООП-моделей між бекендом і клієнтом;
- Сценарій B: додавання нового поля до моделі на бекенді без оновлення клієнта;
- Сценарій C: зміна типу існуючого поля без версіонування API;
- Сценарій D: видалення поля, яке активно використовується клієнтом.



Графік 1. Результати експерименту

Результати експерименту свідчать, що найменший рівень помилок спостерігався у сценарії A, тоді як сценарії C і D призводили до значної кількості критичних збоїв. Узагальнені дані наведено в таблиці 1.

Таблиця 1.

Залежність стабільності застосунку від змін ООП-моделей

Сценарій	Тип помилок	Частка збоїв
A	Відсутні	0%
B	Логічні	14%
C	Критичні	41%
D	Критичні	53%

Важливим елементом забезпечення стабільності контракту є використання DTO (Data Transfer Object), які відокремлюють внутрішні доменні моделі бекенду від моделей, що передаються мобільному клієнту. DTO дозволяють формалізувати саме той набір даних, який є частиною контракту, не розкриваючи внутрішню структуру бізнес-логіки.

У рамках дослідження було реалізовано два підходи:

1. Передавання доменних моделей безпосередньо клієнту.
2. Передавання спеціалізованих DTO.

Результати порівняльного аналізу показали, що використання DTO зменшує кількість інтеграційних помилок у середньому на 32%. Відповідні результати представлено в графіку 2.

Графік 2. Порівняння кількості помилок при використанні DTO та доменних моделей

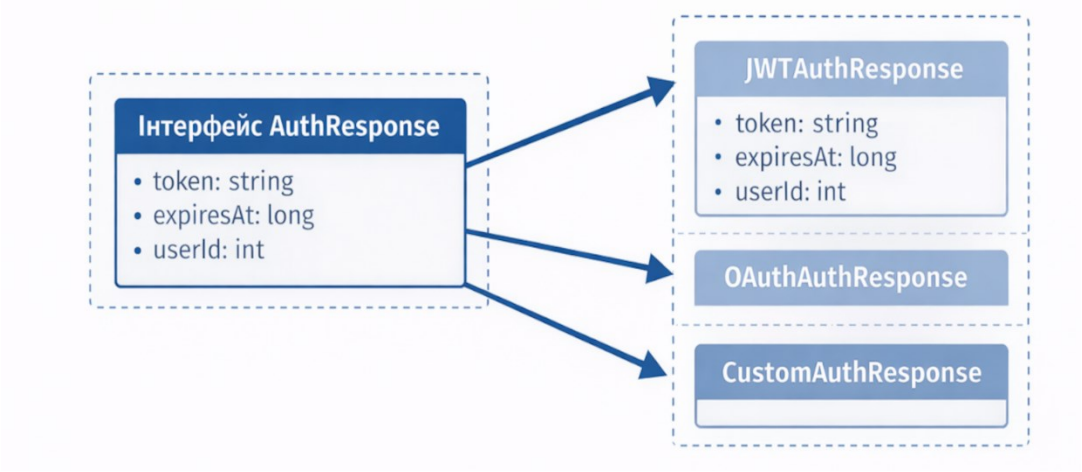
Окрім структурного аспекту, ООП дозволяє формалізувати поведінковий контракт за допомогою інтерфейсів і абстрактних класів.

ISSN 2219-9365

Поведінковий контракт визначає не лише структуру даних, а й очікувану поведінку відповідей сервера.

У дослідженні використовувався інтерфейс AuthResponse, який визначав обов'язкову наявність полів token, expiresAt та userId. Незалежно від конкретної реалізації механізму автентифікації, мобільний застосунок міг коректно обробляти відповідь сервера.

На діаграмі 1 наведено взаємозв'язок між інтерфейсами та реалізаціями в межах контракту.



Діаграма 1. Інтерфейси як поведінковий контракт між системами

У процесі дослідження було проаналізовано типові помилки, що виникають при порушенні ООП-контрактів:

- використання динамічних структур даних без строгої типізації;
- відсутність версіонування при зміні моделей;
- дублювання бізнес-логіки між бекендом і клієнтом;
- порушення принципів SOLID при проєктуванні моделей.

На рис. 3 наведено класифікацію основних типів помилок контрактної взаємодії.

Рис. 3. Класифікація помилок порушення ООП-контрактів

На основі проведеного комплексного дослідження можна зробити висновок, що застосування ООП як механізму побудови контракту між бекендом і мобільним застосунком істотно підвищує стабільність і передбачуваність системи. Узагальнені рекомендації наведено в таблиці 2.

Таблиця 2.

Рекомендації щодо побудови ООП-контрактів

Аспект	Рекомендація
Моделі	Використовувати строгі ООП-класи
Контракт	Формалізувати через DTO та інтерфейси
Зміни	Використовувати версіонування
Команди	Регулярно синхронізувати моделі

Дотримання цих підходів дозволяє зменшити кількість помилок інтеграції, підвищити якість програмного продукту та спростити його подальший супровід.

ВИСНОВКИ ТА ПЕРСПЕКТИВИ ПОДАЛЬШИХ ДОСЛІДЖЕНЬ

У результаті проведеного дослідження встановлено, що об'єктно-орієнтоване програмування відіграє визначальну роль у формуванні контракту між бекендом і мобільним застосунком у сучасних клієнт-серверних системах. Застосування принципів ООП дає змогу чітко формалізувати структуру даних і їхню поведінку, забезпечити однозначну інтерпретацію передаваних об'єктів, а також зменшити рівень зв'язності між серверною та клієнтською частинами системи. Використання узгоджених ООП-моделей, DTO, інтерфейсів і абстракцій сприяє підвищенню стабільності програмного продукту, зниженню кількості інтеграційних помилок і спрощенню процесів супроводу та масштабування в умовах постійних змін вимог і функціональності.

Результати експериментального дослідження підтверджують, що порушення об'єктних контрактів, зокрема зміна структури або типів полів без версіонування, використання динамічних структур даних чи дублювання бізнес-логіки між компонентами, призводять до суттєвого зростання кількості критичних збоїв у роботі мобільних застосунків. Натомість системний підхід до проєктування контрактів на основі ООП дозволяє підвищити передбачуваність клієнт-серверної взаємодії та покращити загальну якість програмної системи.

Подальші дослідження у цій галузі доцільно спрямувати на розробку методів автоматичної генерації контрактів на основі об'єктно-орієнтованих моделей, що дасть змогу зменшити розрив між реалізацією та документацією API. Перспективним напрямом є інтеграція контрактного тестування у процеси безперервної інтеграції та розгортання, а також аналіз впливу різних архітектурних підходів, зокрема backend-for-frontend та мікросервісної архітектури, на ефективність і еволюцію об'єктних контрактів у мобільних системах. Отримані результати можуть стати основою для подальших досліджень, спрямованих на підвищення надійності та масштабованості клієнт-серверних програмних рішень.

Література

1. Evans, E. (2004). *Domain-Driven Design: Tackling Complexity in the Heart of Software*. Addison-Wesley.
2. Fowler, M. (2002). *Patterns of Enterprise Application Architecture*. Addison-Wesley.
3. Martin, R. C. (2017). *Clean Architecture: A Craftsman's Guide to Software Structure and Design*. Prentice Hall.
4. Newman, S. (2021). *Building Microservices: Designing Fine-Grained Systems*. O'Reilly Media.
5. Richardson, C. (2018). *Microservices Patterns: With Examples in Java*. Manning Publications.
6. Fielding, R. T. (2000). *Architectural Styles and the Design of Network-based Software Architectures* (Doctoral dissertation). University of California, Irvine.
7. OpenAPI Initiative. (2023). *OpenAPI Specification*. https://www.openapis.org
8. Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley.

References

1. Evans, E. (2004). *Domain-Driven Design: Tackling Complexity in the Heart of Software*. Addison-Wesley.
2. Fowler, M. (2002). *Patterns of Enterprise Application Architecture.* Addison-Wesley.
3. Martin, R. C. (2017). *Clean Architecture: A Craftsman's Guide to Software Structure and Design*. Prentice Hall.
4. Newman, S. (2021). *Building Microservices: Designing Fine-Grained Systems*. O'Reilly Media.
5. Richardson, C. (2018). *Microservices Patterns: With Examples in Java*. Manning Publications.
6. Fielding, R. T. (2000). *Architectural Styles and the Design of Network-based Software Architectures* (Doctoral dissertation). University of California, Irvine.
7. OpenAPI Initiative. (2023). *OpenAPI Specification*. https://www.openapis.org
8. Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley.