УДК 004.4`2 **doi: 10.26906/SUNZ.2020.4.051**

М. В. Ліпчанський, О. О. Ільяшенко

Національний технічний університет «ХПІ», Харків, Україна

ПОРІВНЯННЯ ПІДХОДІВ CODE FIRST ТА DESIGN FIRST В РОЗРОБЦІ API

Анотація. У статті розглянуті питання щодо дизайну та моделювання API під час розробки програмних продуктів. Останнім часом виявлено, що API можна використовувати в якості повноцінних продуктів та інтерфейсів для бізнесу, що значно дозволяє розширити власну ціннісну пропозицію за допомогою можливостей партнерів, та з'єднуватися з клієнтами за допомогою різноманітних каналів. Метою статті є огляд, аналіз, та порівняння методу Design First на базі OpenAPI Specification з підходом Code First для створення типового RESTful API. На основі підходу Design First можна виконувати розробку API базуючись на згенерованому boilerplate-коді з опису (контракту) OpenAPI або інших форматів опису, тестових сценаріях та заглушках, що дозволяє розпаралелити виконання задач між виконавцями та підвищити швидкість розробки. Наведені результати порівняння продуктивності розробки на базі API, які створені із використанням двох підходів, встановлено, що реалізація API підходом Design First надає ряд переваг для різних суб'єктів розробки у порівнянні з методом Code First. За допомогою Burndown діаграми та розрахунку Velocity (швидкості) роботи команди зроблено висновок, що підхід Design First дозволяє отримувати більш швидке виконання поставлених завдань.

Ключові слова: Design First; Code First; API; RESTful API; OpenAPI Specification.

Вступ

Постановка проблеми. API (Application Programming Interface) або прикладні програмні інтерфейси являються набором програмного коду, що забезпечує передачу даних між одним програмним продуктом та іншим. Вони також містять умови цього обміну даними та слугують інтерфейсами, що дозволяють програмам спілкуватися одна з одною протягом десятиліть. Але роль API різко змінилася за останні кілька років. Інноваційні компанії виявили, що API можна використовувати в якості інтерфейсів для бізнесу, що дозволяє їм монетизувати цифрові активи, розширити свою ціннісну пропозицію за допомогою можливостей, що надають партнери, та підключитися до клієнтів через різноманітні канали.

Аналіз літератури. Проте для отримання якісного продукту у вигляді API на початкових етапах розробки завжди постає питання дизайну API. Послідовність у сфері дизайну API – це тема, яка широко обговорюється. Стандартизований дизайн API – це важливе питання, яке організаціям потрібно вирішувати на своєму шляху під час створення API, яке легко підтримувати, застосовувати та використовувати. Тим не менш, організації не витрачають достатньо часу на стандартизацію способу розробки API, частково тому, що вони не усвідомлюють цінність цього питання [1-11].

Виклад основного матеріалу

API складаються з двох компонентів:

1. Технічна специфікація, що описує варіанти обміну даними між рішеннями, специфікація виконана у формі запиту на обробку та протоколи доставки даних;

2. Інтерфейс програмного забезпечення, написаний за специфікацією, яка його представляє.

Кожен API містить і реалізується за допомогою функціональних викликів – мовних операторів, які вимагають програмного забезпечення для виконання певних дій та послуг. Виклики функцій – це фрази, що складаються з дієслів та іменників. Формати опису API виступають як контракт, який кінцеві користувачі можуть використовувати, щоб зрозуміти, як найкраще працювати з API. Цей контракт є мовно агностичним і інтерпретується як людьми, так і машинами, що допомагає спростити засвоєння та покращити взаємодію між програмами. Що стосується використання форматів опису API, то з'явилися дві важливі школи: підходи Design First та Code First до розробки API. Підхід Code First – це більш традиційний підхід до побудови API, причому розробка коду відбувається після викладання бізнес-вимог, з часом генеруючи документацію з коду. Підхід Design First виступає за початкову розробку контракту API перед написанням будь-якого коду. Це відносно новий підхід, але він швидко набирає популярність, особливо із використанням форматів опису API. Щоб краще зрозуміти два підходи, треба побачити їх місце у життєвому циклі створення API (рис. 1).

Як і будь-який продукт, концепція API починається з того, що команда бізнесу визначає ідею. Ідея аналізується, і план, щоб скористатися нею, створюється в текстовому документі стратегами, аналітиками та іншими представниками бізнесу. Потім цей документ передається команді розробників, де план набуває певної реальної форми.

В цей момент є дві можливості розробити API:

1. Design First: план перетворюється на зручний для читання людиною чи машиною контракт, з якого у подальшому створюється кодова база.

2. Code First: на основі бізнес-плану API кодується безпосередньо. З цього коду може бути сформований зручний для читання людиною чи машиною документ.

Слід зазначити різницю в тому, що Design First дозволяє швидко розпочати паралельну роботу між командами дослідження та розробки, в той час як з використанням Code First потрібно очікувати на вихід першого випуску API, перш ніж почати його тестування та інтеграцію в код на стороні клієнта (рис. 2).

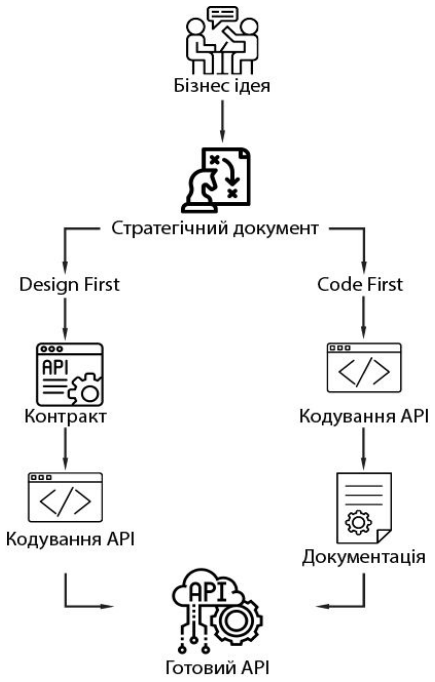

Рис. 1. Загальна схема підготовки та реалізації API

Специфікація OpenAPI, початково відома як Swagger – це специфікація машиночитабельних файлів з інтерфейсами, для опису, створення, використання і візуалізації REST веб-сервісів. Web APIs які відповідають архітектурним обмеженням REST, називаються RESTful API. Ці API використовують HTTP-запити, які також називають методами або дієсловами, для роботи з ресурсами: GET, PUT, HEAD, POST, PATCH, CONNECT, TRACE, OPTIONS та DELETE.

Ця специфікація є частиною моделювання API, тобто створення дизайн-документу, яким можна поділитися з іншими командами розробників, клієнтами або керівниками. Ця схема – це контракт між розробником API та організаціями, клієнтами, які будуть її використовувати. Модель схеми по суті є контрактом описуючи, що таке API, як він працює, і які саме кінцеві точки плануються бути розробленими. Тобто схема є зручним для читання описом кожної кінцевої точки, яка може використовуватися для обговорення API перед написанням будь-якого коду. Специфікація API є ключовим елементом у моделюванні API, який дозволяє виконувати генерацію API-документації, вихідного коду сервера та клієнта (рис. 3).

Специфікація OpenAPI не залежить від мови. Також її можна поширювати на нові технології та протоколи передачі даних. З декларативною специфікацією ресурсу OpenAPI, клієнти можуть розуміти і використовувати сервіси без знання деталей реалізації сервера [8]. Прикладом опису операцій у форматі OpenAPI Specification 3.0.3 YAML, що пов'язані зі створенням, видаленням, оновленням та отриманням за унікальним ідентифікатором ресурсу типу «Абонент» наведено на рис. 4.

Серед інструментів для розробки API підходом Design First слід виділити OpenAPI Generator, що призначений для створення клієнтських бібліотек API, заглушок сервера, конфігурацій та документації з OpenAPI 2.0 та 3.x документів. Він має широкий спектр функцій і використовується широким колом користувачів, деякі з яких також є супровідниками. OpenAPI Generator зосереджений на простоті використання; він позиціонує себе як інструмент зменшення навантаження на нові розробки та технології за рахунок інтеграції та використання документів OpenAPI (рис. 5).

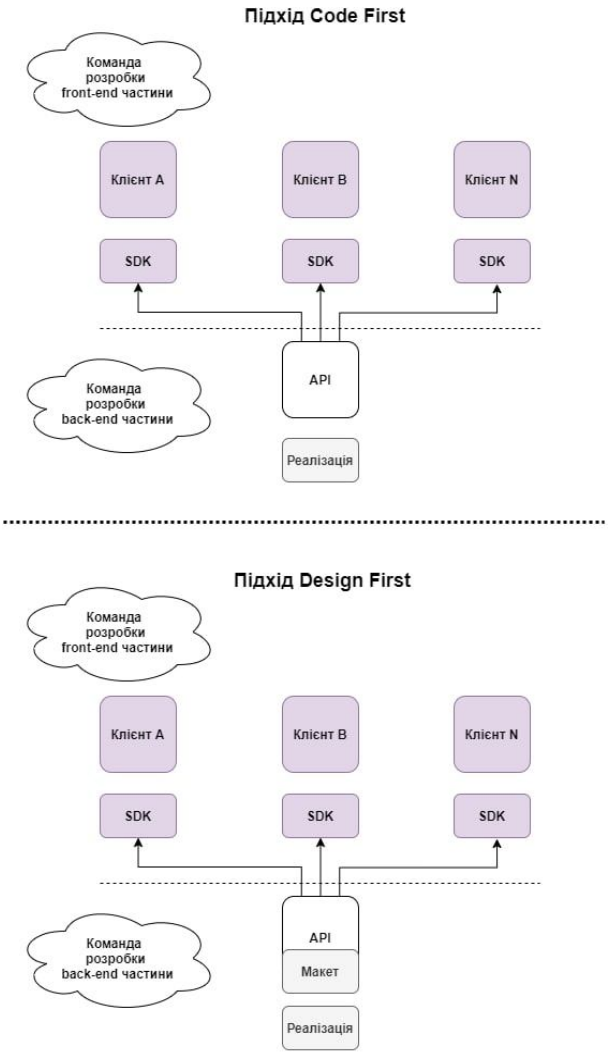

Рис. 2. Порівняння підходів Design First та Code First у площині паралельної роботи команд розробки

Рис. 3. Структура моделювання API, з визначеною специфікацією API

Рис. 4. Структура документу OpenAPI Specification YAML

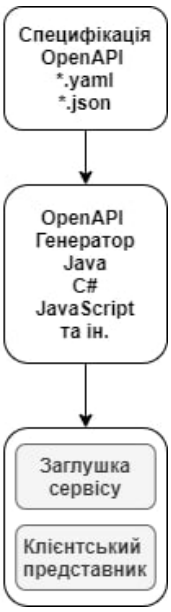

Рис. 5. Загальна схема роботи OpenAPI Generator

Результатом порівняння може слугувати burndown діаграма (діаграма згорання), яка є графічним представленням про те, як швидко команда працює з одними історіями (інструмент, який використовується для опису завдання з точки зору кінцевого користувача). Burndown діаграма показує загальний внесок у роботу в загальному обсязі роботи для кожної ітерації. За шкалою Y відзначають кількість запланованих балів (в даному випадку), годин або кількість. За шкалою X відзначають кількість днів до закінчення спринту. Спринт – це короткий часовий інтервал, протягом якого команда розробників виконує заданий обсяг роботи.

Початкові задачі тестового проекту та умови їх виконання були однаковими для обох реалізацій, підходом Code First та Design First. На рис. 6 зображена діаграма для тестового проекту, що відображає завершений спринт та показує невирішені завдання і трудовитрати, необхідні для їх завершення з розрахунку на 20 робочих днів.

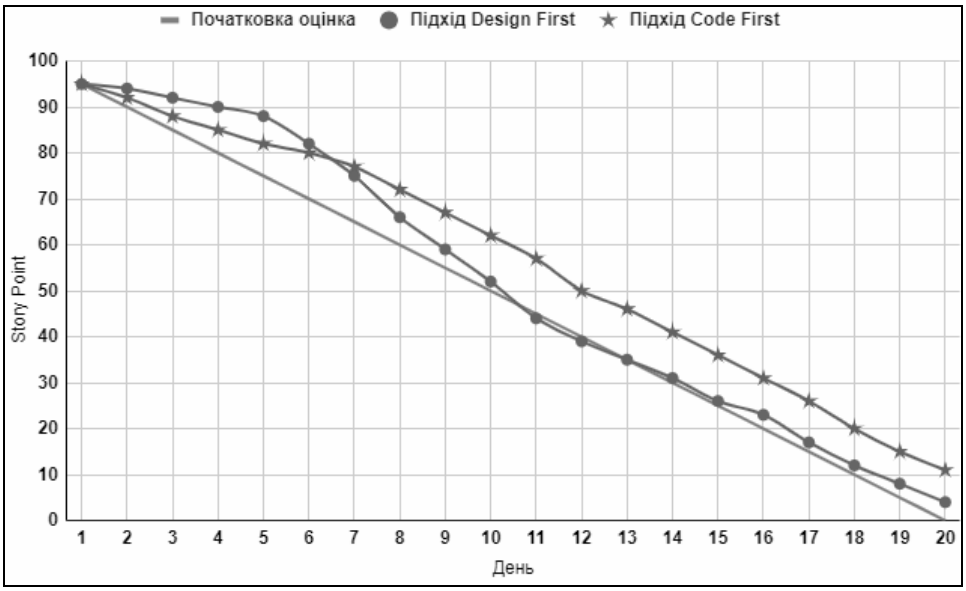

Рис. 6. Burndown діаграма розробки підходами Code First та Design First

Базуючись на burndown діаграмі проекту можна зробити висновки, що з підходом Code First кількість балів історій, які були завершені, є меншою в порівнянні з Design First. Використовуючи поняття Velocity (швидкості) роботи команди можна порівняти продуктивність розробки кожним із підходів:

$$v = \frac{\Delta x}{\Delta t}$$

де v – швидкість команди;
Δx – кількість завершених історій;
Δt – кількість ітерацій.

Для Code First це значення дорівнює 84 завершених історій на 1 ітерацію, а для Design First – 91. Тобто виконання поставлених завдань, використовуючи Code First, відбувалося з меншою швидкістю. Серед причин негативних результатів можна виділити відсутність паралельної роботи серед команд розробників, написання більшої кількості boilerplate-коду та переробки більшої частини функціональності після відгуку користувачів API. Взагалі одним з найважливіших аспектів підходу Design First є здатність отримувати зворотний зв'язок дуже швидко завдяки таким проектним аспектам, як:

– документація – спільне використання цієї документації з усіма можливими зацікавленими сторонами, такими як різні члени команди досліджень та розробок, може призвести до генерації корисних вказівок і думок, оскільки кожна зацікавлена сторона має різний погляд і, ймовірно, представляє різні аудиторії для API;

– тестування – оскільки визначення API є контрактом, його можна перевірити ще до того, як буде впроваджена бізнес-логіка;

– згенеровані заглушки коду серверної частини – розробники, які починають впроваджувати бізнес-логіку, можуть знаходити проблеми на етапі програмування та відповідно оновлювати визначення API.

Висновки

В результаті проведеного аналізу було встановлено те, що реалізація API підходом Design First надає ряд переваг для різних суб'єктів розробки у порівнянні з методом Code First. Для розробників API можна виділити можливість генерації коду та заглушок із визначення документу-опису API. Інші розробники, що є користувачами API, можуть використовувати ці заглушки, що зменшує час розробки продукту в цілому. Це надає їм додаткову швидкість і послідовність (оскільки можливо налаштувати шаблони відповідно до своїх потреб).

Design First підхід має місце лише на етапі до або на ранній стадії розробки API, і початковим результатом цього підходу є зручне для читання людиною чи машиною визначення API. Опис API у документі OpenAPI, що є реалізацією підходу Design First, дозволяє використовувати інструменти для автоматизації багатьох процесів, пов'язаних з API, а саме: генерація boilerplate-коду, тестових модулів, заглушок майбутніх методів та базового каркасу додатку.

Список літератури

1. Patni, S. (2017), Pro RESTful APIs: Design, Build and Integrate with REST, JSON, XML and JAX-RS, 1st ed., Apress, Santa Clara, California, USA, 126 p.
2. Jin, B., Sahni, S. and Shevat, A. (2018), Designing Web APIs, 1st ed., O'Reilly Media, Inc., Sebastopol, CA, USA, 232 p.
3. "Understanding the API-First Approach to Building Products", [Електронний ресурс. – Режим доступу до матеріалу: https://swagger.io/resources/articles/adopting-an-api-first-approach/
4. "What is API: Definition, Types, Specifications, Documentation", [Електронний ресурс, 2019. – Режим доступу до матеріалу: https://www.altexsoft.com/blog/engineering/what-is-api-definition-types-specifications-documentation/
5. Gontovnikas, M. "The Business Value of API-First Design", [Електронний ресурс, 2020. – Режим доступу до матеріалу: https://auth0.com/blog/the-business-value-of-api-first-design/
6. Кучук Н.Г., Гавриленко С.Ю., Лукова-Чуйко Н.В., Собчук В.В. Перерозподіл інформаційних потоків у гіперконвергентній системі / С.Ю. Гавриленко. *Сучасні інформаційні системи*. 2019. Т. 3, № 2. С. 116-121. DOI: https://doi.org/10.20998/2522-9052.2019.2.20
7. Nechausov A., Mamusuĉ I., Kuchuk N. Synthesis of the air pollution level control system on the basis of hyperconvergent infrastructures. *Сучасні інформаційні системи*. 2017. Т. 1, № 2. С. 21-26. DOI: https://doi.org/10.20998/2522-9052.2017.2.04
8. Ponelat, J.S. and Rosenstock, L.L. (2021), Designing APIs with Swagger and OpenAPI, Manning, 400 p.
9. Зиков І. С., Кучук Н. Г., Шматков С. І. Синтез архітектури комп'ютерної системи управління транзакціями e-learning. *Сучасні інформаційні системи*. 2018. Т. 2, № 3. С. 60–66. DOI: https://doi.org/10.20998/2522-9052.2018.3.10
10. Rosenstock, L. "OpenAPI and Design-First Principles", [Електронний ресурс, 2018. – Режим доступу до матеріалу: https://stoplight.io/blog/openapi-and-design-first-principles-96e7c4b2aec1/
11. Takashi, N. "An API-First Approach For Designing Restful APIs", [Електронний ресурс, 2020. – Режим доступу до матеріалу: https://hackernoon.com/an-api-first-approach-for-designing-restful-apis-5tu3zru

Received (Надійшла) 22.10.2020
Accepted for publication (Прийнята до друку) 18.11.2020

Comparison of code first and design first approaches in api development

M.v. lipchanskyi, o.o. iliashenko

Abstract. This article deals with API design and modelling issues in software development. Recently it has been found that the API can be used as full-fledged products and interfaces for business, significantly allows to expand your own value proposition with the help of partners' capabilities, and to connect with clients through various channels. The purpose of the article is to review, analyze and compare the Openapi specification-based Design First method with the Code First approach to create a Restful API. Based on the Design First approach, you can develop an API which based on the generated boilerplate code from the Openapi description (contract) or other description formats, test scripts and stubs, which allows to parallelize execution of tasks between performers and increase speed of development. The results of the comparison of the performance of the API-based development, which are created using two approaches, are shown, that the implementation of the API approach Design First offers a number of advantages for different developers compared to the Code First method. Using Burndown diagram and calculating Velocity (the speed) of the team's work, it is concluded that the Design First approach allows faster execution of the tasks.

Keywords: Design First, Code First, API, RESTful API, OpenAPI Specification.