УДК 004.057.4 DOI: 10.26565/2519-2310-2019-2-05

ВИМОГИ ДО СТВОРЕННЯ ІНСТРУМЕНТАЛЬНИХ ЗАСОБІВ ДЛЯ ЕМУЛЯЦІЇ ПРОЦЕСІВ ІНТЕГРАЦІЇ ПРОГРАМНИХ СИСТЕМ ІЗ ВИКОРИСТАННЯМ HTTP/HTTPS ПРОТОКОЛІВ

Костянтин Д'яченко, Дмитро Зінов'єв

Харківський національний університет імені В.Н. Каразіна, майдан Свободи 4, Харків, 61022, Україна
konstantyn.diachenko@gmail.com, zinoviev@karazin.ua

Рецензент: Ткачук Микола Вячеславович, доктор технічних наук, професор, Харківський національний університет імені В.Н. Каразіна, майдан Свободи, 4, Харків, 61022, Україна.
tka.mobile@gmail.com

Надійшло: Листопад 2019.

Анотація: *Емуляція процесів інтеграції програмних систем із використанням HTTP/HTTPS протоколів на теперішній час широко використовується при розробці програмних систем з розподіленою архітектурою. У роботі розглянуто сучасний стан проблеми, проаналізовано існуючі інструментальні засоби для її вирішення, виявлені недоліки та запропонований концептуальний підхід та вимоги до програми, яка дозволить автоматизувати і налагодити процеси розробки та тестування програмних систем із використанням HTTP/HTTPS протоколів.*

Ключові слова: *комп'ютерні науки; програмні системи; прикладний програмний інтерфейс; REST; емуляція веб-сервісів.*

Вступ

На даний час багато сучасних програмних систем мають розподілену архітектуру, тобто складаються з двох або більше компонентів, які узгоджено взаємодіють між собою.

Для забезпечення якості під час розробки програмного продукту обов'язково виконується його тестування. Тести виконують перевірки відповідності заявлених до програмної системи або її компонентів вимог і реально реалізованої функціональності, здійснювані шляхом спостереження за її або їх роботою в штучно створених ситуаціях і на наборі тестів, створених на основі вимог [1].

В залежності від масштабу та бюджету проекту, розробники відповідних програмних систем (ПС), використовують як мінімум, два середовища для їх розгортання. Це середовище для розробки (DEV) та середовище для готової версії системи (Production). Також, іноді додається третє середовище – для тестування (QA).

Середовище для тестування містить версію продукту, яка планується до запуску на Production середовищі і яка містить зміни відносно до поточної версії, що запущена на Production. На цьому середовищі може проводитися E2E (*end two end*) тестування, тестування прикладного програмного інтерфейсу (ППІ), інтеграційне тестування або мануальне тестування та регресія. Якщо ПС, що розробляється, під час проходження тест-кейсів взаємодіє зі сторонніми системами або веб-сервісами (*компонентами*), які є платні, – плата може стягуватися за кожне звернення, що значно збільшує кошти на розробку. Окрім того, не всі сторонні системи та сервіси надають DEV та Production середовища.

На сьогодні зазначені труднощі вирішують двома способами. Перший, це відключення взаємодії ПС, що розробляється, зі сторонньою службою. Це призводить до зниження якості продукту із-за неможливості тестування процесів взаємодії зі сторонніми службами. Другий спосіб, це емуляція поведінки, сценаріїв або проходження потоків даних сторонньої системи або сервісів [2]. Такий спосіб потребує створення нових системи-емуляторів, розробка яких може займати багато часу.

Тести виконують перевірки, в яких використовуються попередньо накопичені вхідні дані. Наприклад, коли відомо, що на «Запит А» система повинна повернути «Відповідь А». Пара «Запит А» та «Відповідь А» - є вхідними даними для тесту. Для забезпечення достовірності

проходження тест-кейсів необхідно ізолювати усі компоненти, взаємодія яких тестується, бо будь-яка паралельна взаємодія з компонентами ПС може вплинути на вихідні дані. Наприклад, для тестування резервації конкретного продукту треба забезпечити такі умови для тесту, щоб більше ніхто не зміг зарезервувати цей продукт в один і той же час. Цю особливість вирішують відповідні системи-емулятори [3].

Часто буває так, що UI/UX команда змушена розробляти свою частину ПС спираючись на специфікації, контракт або інтерфейс до сервісу, з якими їх компоненти системи будуть взаємодіяти. Із-за відсутності належних інструментів емуляції процесів інтеграції ПС виникають затримки розробки. Також затримки розробки можуть виникати, коли під час розробки та тестування ПС один із сервісів, що необхідний для коректної роботи системи, не працює.

Тому команди розробників і тестувальників змушені витрачати ресурси для створення специфічних інструментів для вирішення проблем, що перераховані вище. Тому, імітація реального ППІ веб-сервісу зараз широко використовується у сфері розробки програмних продуктів. Під імітацією ППІ ми розуміємо керовану емуляцію поведінки ППІ веб-сервісу. Загалом можна зазначити, що імітація реальних сервісів та створення керованих заглушок активно використовується у наступних областях:

- організація E2E, інтеграційного, ППІ, мануального, регресійного, компонентного тестування програмного продукту;
- організація паралельної роботи фронтенд і бекенд команд над однією задачею без затримок;
- ізолювання єдиного сервісу для проведення налагодження програмної системи.

Поширення ідей створення та використання інструментів імітації реальних сервісів обумовлене наступними причинами:

- неможливістю забезпечити ізоляцію середовища від зовнішнього середовища під час перевірки тест-кейсами;
- відсутністю підтримки декількох середовищ стороннім сервісом, з яким взаємодіє система;
- наявністю проблем з проведенням тестування розподілених програмних систем, де система має зв`язки з однієї та більше сторонніми системами або сервісами;
- наявністю проблем з організацією паралельної розробки одного продукту;
- тенденцією на оптимізацію розробки та тестування програмного продукту;
- вимогами до покращення якості кінцевого програмного продукту.

Для більшості розробників та тестувальників, особливий інтерес представляють можливості створення «заглушок» або емуляторів сторонніх систем для використання останніх у процесі інтеграції з програмною системою, що розробляється, та її налагодженням за час, у декілька разів менший у порівнянні зі створенням специфічних інструментів, а також для проведення повного тестування системи із самостійно-керованими системами-емуляторами.

Наявність такого інструменту дозволить:

- заощадити кошти у разі, коли імітована система є платною і не має спеціальних тестових режимів;
- заощадити час на розробку і налагодження певної частини програмного забезпечення;
- ізолювати ПС від стану сторонніх сервісів для тестування бізнес-логіки;
- розгортати інструмент як окремий компонент інфраструктури продукту.

У межах даної роботи розглядається питання стосовно концептуального підходу до розробки інструментальних засобів для емуляції процесів інтеграції програмних систем із використанням HTTP/HTTPS протоколів.

1 Процес емуляції реального прикладного програмного інтерфейсу сервісів або служб

Веб-сервіс (або веб-служба) – це програмний сервіс із стандартизованими інтерфейсами, що ідентифікується веб-адресою. Веб-служби можуть взаємодіяти одна з одною і зі сторонніми додатками за допомогою повідомлень, заснованих на певних протоколах. На сьогодні

для взаємодії компонентів розподіленого додатка в мережі широко використовується архітектурний стиль REST (Representational State Transfer) [4].

Реалізація REST стилю виконується за допомогою протоколу передачі даних – HTTP, який використовується для управління та передачі інформації ресурсу RESTful сервісу.

Імітація реального ППІ дозволяє емулювати будь-який RESTful сервіс.

Це корисно в сценаріях тестування, паралелізації роботи та ізолювання однієї служби.

Для тестування можливо:

- легко відтворювати всі типи відповідей для компонентів складних програмних систем, таких як веб-служби REST;
- ізолювати перевірку системи, щоб переконатися, що тести надійно виконуються і не виконуються, коли існує справжня помилка. Важливим є лише перевірка системи, що тестується, а не її залежностей, щоб уникнути невдач тестів через невідповідні зовнішні зміни, такі як відмова мережі або перезавантаження сервера, де розташована веб-служба;
- легко встановлювати шаблонні відповіді незалежно для кожного тесту, щоб гарантувати наявність тестових даних відповідно до контракту веб-сервісу в кожному тесті;
- створювати тестові твердження, які мають перевіряти запити, які надсилала система, що тестується [5].

При паралелізації роботи можливо наступне:

- почати роботу з ППІ служб до того, як веб-служба буде реалізована. Якщо ППІ або послуга ще не повністю розроблені, емуляція може імітувати ППІ, що дозволяє будь-якій команді, яка використовує цю службу, розпочати роботу без затримки;
- ізолювати групи розробників під час початкових етапів розробки, коли ППІ служби можуть бути надзвичайно нестабільними і мінливими. Використання імітації реальних ППІ дозволяє продовжувати роботу з розробки, навіть коли зовнішня служба виходить з ладу.

Під час розгортання та налагодження ПС буває корисно запускати окрему програму або одну службу, або обробляти підмножину запитів на локальному комп'ютері в режимі налагодження. Використовуючи імітації реального ППІ, легко вибірково пересилати запити до локальних процесів, які виконуються в режимі налагодження.

2 Аналіз інструментів для емуляції прикладного програмного інтерфейсу

На теперішній час існує клас інструментальних засобів, які дозволяють вирішувати задачі імітації процесів інтеграції ПС із використанням HTTP/HTTPS протоколів. До них відносяться, наприклад: MockServer, Mockable.io, Prism, Restmock, та Okhttpmockwebserver.

Розглянемо більш детально можливості та недоліки деяких з них.

MockServer – безкоштовний інструмент для створення заглушок будь-якої системи, яка інтегрується через HTTP або HTTPS [6].

Перш за все цей інструмент дає можливість:

- запускати спеціальний сервер для імітування сервісів і служб для інтеграційного тестування;
- створювати імітацію ППІ реального сервісу і налаштувати проксі до реального сервісу;
- створювати очікувані відповіді за кількома різними характеристиками запиту до сервісу;
- запам'ятовувати відповіді від сервісу;
- підтримки двох мов програмування.

До недоліків MockServer можна віднести наступне:

- необхідність досвіду та знань з програмування для створення емуляторів;
- відсутність графічного інтерфейсу для налаштувань;
- імітація повністю потребує написання програмного коду;
- відсутність можливості збереження накопичених відповідей від реального сервісу;

- відсутність можливості по препроцесингу і постпроцесингу запиту та відповіді.

Програма Mockable.io – безкоштовний інструмент для створення «заглушок» для ППІ. Mockable.io дає можливості створювати очікувані відповіді за явно вказаними характеристиками запиту, динамічно генерувати відповіді від сервісу та має графічний інтерфейс для налаштувань.

До її недоліків можна віднести неможливість зберігання накопичених відповідей від реального сервісу, відсутність можливості щодо препроцесинга і постпроцесинга запиту та відповіді, неможливість створювати очікувані відповіді за кількома різними характеристиками запиту.

Програма Prism – це безкоштовний набір пакетів для створення макетів API з OpenAPI v2 (*раніше відомий як Специфікація Swagger*) та OpenAPI v3.

Переваги Prism:

- можливість створювати очікувані відповіді за явно вказаними характеристиками запиту;
- повне логування;
- створення шаблонів запитів за заданими характеристиками запиту.

Недоліки Prism:

- неможливо зберегти накопичені відповіді від реального сервісу;
- відсутні можливості по препроцесингу і постпроцесингу запиту та відповіді;
- відсутність можливості створювати очікувані відповіді за кількома різними характеристиками запиту;
- відсутність графічного інтерфейсу для налаштувань.

Аналіз можливостей існуючих інструментів для імітації ППІ дозволяє зробити висновок, що існуючі рішення мають або вузький спектр можливостей, або не зручний інтерфейс, або є платними. У реальності все це призводить до того, що частіше за все, задачі, які потребують імітацію зовнішніх сервісів, розробники ПС вирішують за допомогою дрібних самостійно створених програмних рішень. Тому було б корисно створити інструментальний засіб для емуляції процесів інтеграції ПС, який би об'єднав усі корисні функції та був вільний від перелічених недоліків.

3 Концептуальне проектування програми для емуляції прикладного програмного інтерфейсу

По-перше, у функціях програми для імітації прикладного програмного інтерфейсу необхідно передбачити можливість створення декількох режимів емуляції програмної системи. А саме:

- “збір даних” – отримання варіантів відповідей від реального сервісу та їх збереження;
- “проксінг” – звернення безпосередньо до даних реального сервісу;
- “заглушка” – використання збережених або завантажених даних;
- змішаний режим зі сценаріями – препроцесінг, постпроцесінг запитів з використання DataFlow шаблонів на об'єктно-орієнтованій мові Groovy.

Діаграма варіантів використання, що представлена на рис.1, описує систему на концептуальному рівні. Вона відображає відношення між актором “Тестувальник” та прецедентами .

Це дасть можливість розробникам ПЗ створювати гнучкі емуляції веб-служб і використовувати їх як у якості примітивних заглушок так і у якості емуляторів з поведінкою, встановленою Groovy скриптами. Тобто для тестування можна використовувати режим “заглушки”, який дає можливість імітувати веб-сервіси із попередньо завантаженими даними.

Режими “збір даних” та “проксінг” доповнюють один одного, та відрізняються лише особливістю збереження даних і можуть бути використані для підготовки тестування.

Змішаний режим надає можливість задати звичайній заглушці різну поведінку в залежності від зовнішніх параметрів стану системи. Наприклад, коли на один і той же запит ППІ сервіс повинен відповідати по різному в залежності від часу. Для змішаного режиму необхідно мати досвід у написанні скриптів на мові програмування Groovy.

По-друге, необхідно передбачити можливість узгодження запитів за параметрами, тілом або заголовками запиту і запам'ятовувати схожі відповіді для однакових по заданим характеристикам запитів. Іншими словами, для запитів, з однаково визначеними характеристиками, має повертатися однакова відповідь.

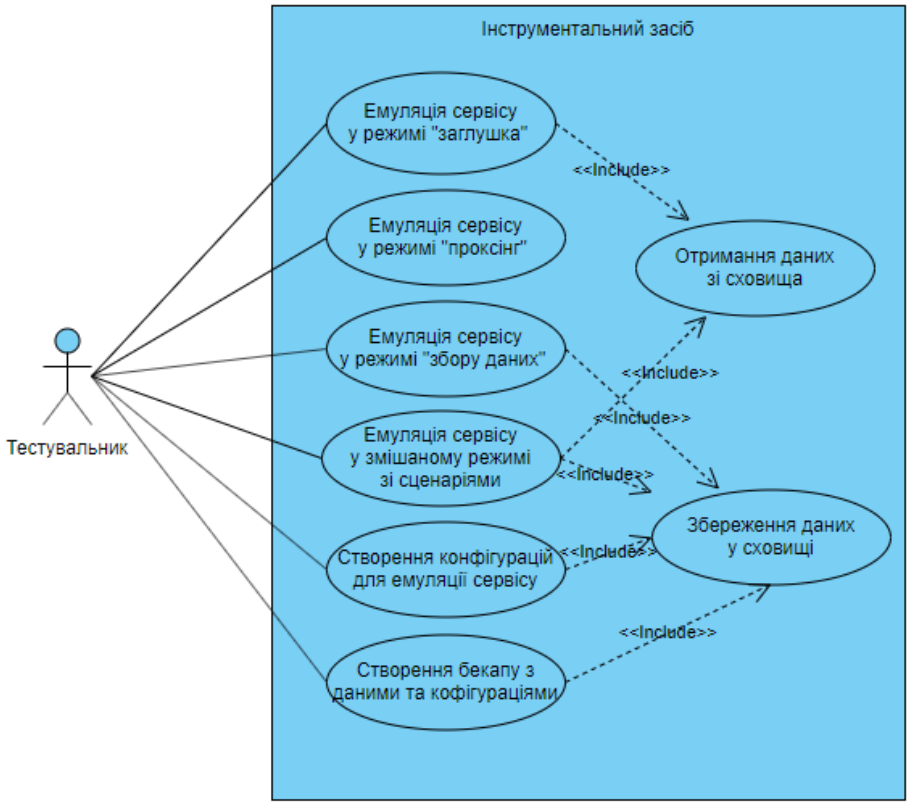

Рис. 1 – Use-case діаграма для інструментального засобу

Втілення такої пропозиції дасть можливість розробнику створювати детальні емуляції веб-служб з урахуванням не тільки URI адреси до ресурсу, а й інших елементів HTTP запиту: заголовків та тіла запиту. Чим більше характеристик може бути залучено під час емуляції веб-сервісу, тим детальніше буде описаний ППІ веб-сервісу для подальшого тестування.

Третя пропозиція, яка має бути реалізована при розробці програми для імітації ППІ – це здатність створювати та задавати сценарії для запитів за допомогою мови XML або за допомогою предметно-орієнтованої мови. Сценарій в контексті даного питання – це опис кроків, які повинні виконуватися під час обробки запиту та відповіді (див. Рис. 2). Можливі варіанти: виклики підзапитів, модифікація будь-яких властивостей запиту, препроцесінг запиту, постпроцесінг запиту, модифікація відповіді.

```
def headerName = 'x-app'
def headerValue = 'mock-service'
request?.headers.find { mockServiceHeader -> mockServiceHeader.name == headerName }.value = [headerValue]
if (request.query) {
    request.query = "${request.query}&lang=en"
}
```

Рис. 2 – Groovy сценарій для модифікації вхідного запиту

Ця пропозиція може бути реалізована за допомогою використання паттерну DataFlow та існуючих реалізацій від компанії Apache. Це надасть можливість розробнику задати поведінку веб-сервісу при його емуляції у випадках, коли неможливо заздалегідь створити дані для

тестування. Наприклад, коли веб-сервісу у запиті передається електронна пошта будь-якого користувача системи та номер картки лояльності. Так як сервіс відповідає на запит, шукаючи за необхідними характеристиками дані для відповіді у заздалегідь підготовлених даних, то для кожного тесту необхідно підготувати дані з новими користувачем. Це дасть можливість створювати моделі для препроцесінгу запиту та постпроцесингу відповіді за допомогою написання Groovy скрипту [7].

Четверта пропозиція полягає у створенні можливості експорту та імпорту даних до інструменту імітації за визначеними шаблонами, а також можливість експорту усієї конфігурації інструменту та можливість імпорту його повної конфігурації без втрат. Це дозволить переносити інструментальний засіб в інші середовища без втрати даних і конфігурацій. У такому разі усі накопичені дані мають зберігатися примусово, або у «бекапі» за розкладом.

4 Висновки

В межах даної роботи були запропоновані концептуальний підхід та відповідні вимоги до інструментального засобу, який дозволить автоматизувати та налагодити процеси розробки та тестування програмних систем із використанням HTTP/HTTPS протоколів.

В цілому, така програма має являти собою систему, яку можна запустити на окремому сервері або локальному комп'ютері (*як окремий компонент з графічним інтерфейсом та REST ППП*) для емуляції програмних систем.

Загальні вимоги до інструментального засобу можна сформулювати наступним чином:

- інструментальний засіб повинен мати декілька режимів емуляції програмної системи;
- інструментальний засіб повинен надавати можливості орієнтуватися на різні характеристики запиту в залежності від природи запитів до веб-служби;
- інструментальний засіб повинен надавати можливість запам'ятовувати відповіді від сервісу та конфігурації сервісів-емуляторів;
- інструментальний засіб повинен надавати можливість препроцесінгу і постпроцесінгу запитів та відповідей за допомогою сценаріїв, які можуть бути створені на основі предметно-орієнтованої мови Groovy;
- повинен надавати можливість експортувати та імпортувати дані до інструменту емуляції.

Посилання

[1] Кулямин В.В. Место тестирования среди методов оценки качества ПО / В.В. Кулямин, О.Л. Петренко. URL: https://software-testing.ru/library/5-testing/117-2008-10-13-19-25-13 (дата звернення 02.11. 2019)

[2] Aggarwal K. API Mocks and why should we care / Kapil Aggarwal. URL: https://medium.com/@kapil.aggarwal1/api-mocks-and-why-should-we-care-418b24c35c25 (дата звернення 08.11. 2019)

[3] Bulaty W. Stubbing, Mocking and Service Virtualization Differences for Test and Development Teams. URL: https://www.infoq.com/articles/stubbing-mocking-service-virtualization-differences/ (дата звернення 10.11. 2019)

[4] Артемий . Архитектура REST / Артемий. URL: https://habr.com/ru/post/38730/ (дата звернення 06.11. 2019)

[5] Fowler M. Mocks Aren't Stubs / Martin Fovler. URL: https://martinfowler.com/articles/mocksArentStubs.html (дата звернення 12.11. 2019)

[6] Bloom J. D. MockServer. What is Mock Server / James D Bloom. URL: http://www.mock-server.com/#what-is-mockserver (дата звернення 10.11. 2019)

[7] Integrating Groovy in a Java application. URL: http://docs.groovy-lang.org/latest/html/documentation/guide-integrating.html (дата звернення 06.11. 2019)

Reviewer: Tkachuk Mykola, Doctor of Science (Engineering), Full Prof., V.N. Karazin Kharkiv National University, Kharkiv, Ukraine.
E-mail: tka.mobile@gmail.com

Received on November 2019.

Authors:
Kostiantyn Diachenko, student, V. N. Karazin Kharkiv National University, Kharkiv, Ukraine.
E-mail: konstantyn.diachenko@gmail.com.

Dmytro Zinoviev, Senior Lecturer, V. N. Karazin Kharkiv National University, Kharkiv, Ukraine.
E-mail: zinoviev@karazin.ua.

Requirements for creating tools to emulate software integration processes using HTTP/HTTPS protocols.

Abstract. Emulation of software systems integration processes using HTTP/HTTPS protocols is now widely used in the development of distributed systems software systems. The paper discusses the current state of the problem, analyzes the existing tools to solve it, identifies shortcomings, and offers a conceptual approach and program requirements that will automate and debug software development and testing processes using HTTP/HTTPS protocols.

Keywords: Computer Science; Software Systems; Application Programming Interface; REST; Emulation of Web Services.

Рецензент: Ткачук Николай Вячеславович, д.т.н., проф., Харьковский национальный университет имени В. Н. Каразина, Харьков, Украина.
E-mail: tka.mobile@gmail.com.

Поступила: Ноябрь 2019.

Авторы:
Константин Дьяченко, студент 6 курса, Харьковский национальный университет имени В. Н. Каразина, Харьков, Украина.
E-mail: konstantyn.diachenko@gmail.com
Дмитрий Зиновьев, ст. преподаватель, Харьковский национальный университет имени В. Н. Каразина, Харьков, Украина.
E-mail: zinoviev@karazin.ua

Требования к созданию инструментальных средств для эмуляции процессов интеграции программных систем с использованием HTTP/HTTPS протоколов.

Аннотация. Эмуляция процессов интеграции программных систем с использованием HTTP/HTTPS протоколов в настоящее время широко используется при разработке программных систем с распределенной архитектурой. В работе рассмотрено современное состояние проблемы, проанализированы существующие инструментальные средства для ее решения, выявленные недостатки и предложен концептуальный подход и требования к программе, которая позволит автоматизировать и наладить процессы разработки и тестирования программных систем с использованием HTTP/HTTPS протоколов.

Ключевые слова: компьютерные науки; программные системы; прикладной программный интерфейс; REST; эмуляция веб-сервисов.