Пригода Андрій Ярославович, аспірант

Державний торговельно-економічний університет

0000-0003-3774-4583

**СТРАТЕГІЇ ТЕСТУВАННЯ CRM-СИСТЕМИ НА ОСНОВІ МІКРОСЕРВІСНОЇ АРХІТЕКТУРИ**

Впровадження CRM-систем на основі мікросервісної архітектури представляє собою складний технологічний процес, що вимагає ретельного планування та виконання стратегій тестування для забезпечення високої якості та надійності функціонування системи. Мікросервісна архітектура передбачає розподіл додатку на невеликі, автономні та взаємодіючі сервіси, кожен з яких відповідає за виконання конкретної функціональності. Така архітектура дозволяє розробникам ефективно масштабувати та підтримувати систему, а також робить її більш гнучкою та модульною.

Проте, при впровадженні CRM-систем на основі мікросервісної архітектури виникають нові виклики у сфері тестування. Перш за все, необхідно враховувати велику кількість сервісів, що входять до складу системи, та їх потенційну взаємодію. Таким чином, стратегії тестування повинні бути спрямовані на виявлення і аналіз критичних шляхів взаємодії між сервісами, а також на перевірку їхньої узгодженості з функціональними вимогами 3].

Валідація інтеграції між сервісами є важливою складовою стратегій тестування CRM-систем на базі мікросервісної архітектури, оскільки успішна взаємодія між компонентами системи є ключовою для забезпечення її правильного функціонування. Техніки, що застосовуються для валідації інтеграції між сервісами, включають:

* Тестування API. API (Application Programming Interface) визначає спосіб взаємодії між різними сервісами. Тестування API дозволяє перевіряти, чи відповідають API кожного сервісу вимогам та чи забезпечують вони необхідний функціонал.
* Тестування взаємодії між сервісами. Ця техніка включає в себе тестування функціональності, яка відбувається при взаємодії різних сервісів. Це може включати передачу даних між сервісами, обробку повідомлень та подій, а також синхронізацію стану.
* Автоматизоване тестування міжкомпонентних інтерфейсів. Дана техніка передбачає автоматизоване створення та виконання тестових сценаріїв для перевірки інтеграції між різними компонентами системи. Вона дозволяє швидко та ефективно перевіряти, чи працюють сервіси разом, і виявляти проблеми в їх взаємодії [2].

Крім того, тестування відновлення після збоїв є ще одним важливим елементом стратегій тестування CRM-систем. Ця техніка передбачає перевірку реакції системи на відмову окремих сервісів та забезпечення безперебійної роботи інших компонентів. Вона включає в себе такі етапи, як відтворення сценаріїв збоїв, перевірку механізмів автоматичного відновлення, тестування реплікації даних та інші техніки, спрямовані на забезпечення надійності системи в умовах незапланованих ситуацій.

Начало формы

Для забезпечення ефективності тестування в контексті мікросервісної архітектури також важливо використовувати автоматизацію тестування та контейнеризацію сервісів. Це дозволяє швидко та ефективно проводити тести на великій кількості конфігурацій та забезпечувати стабільність системи при її розгортанні та масштабуванні.

Автоматизація тестування передбачає створення автоматизованих скриптів та програмних засобів для виконання тестових сценаріїв без участі людини. Це дозволяє автоматично виконувати тести на різних конфігураціях та в умовах масштабування, що сприяє покращенню ефективності та швидкості процесу тестування. Для CRM-систем на основі мікросервісної архітектури автоматизація тестування може включати тестування окремих сервісів, API, взаємодії між сервісами та інші аспекти функціональності.

Контейнеризація полягає у запуску та управлінні додатками та їх залежностями у стандартизованих та ізольованих середовищах, відомих як контейнери. Найпопулярнішим інструментом для контейнеризації є Docker. В контексті стратегій тестування CRM-систем контейнеризація дозволяє швидко розгортати тестові середовища з необхідними компонентами та налаштуваннями, забезпечуючи однакові умови для виконання тестів на різних стадіях розробки та між різними командами. Вона також дозволяє забезпечити ізоляцію тестових середовищ від реальних даних та інших компонентів системи, що збільшує безпеку тестування та унеможливлює вплив тестів на живу систему [1, с. 379–380].

Важливо зазначити, що моніторинг продуктивності та масштабованість грають важливу роль у стратегіях тестування CRM-систем на мікросервісній архітектурі. Перше вимагає вимірювання та аналізу різних показників продуктивності тестового середовища, таких як час відгуку та завантаження ресурсів. Це допомагає виявити проблеми та ризики, пов’язані з продуктивністю, та приймати відповідні заходи.

Друге полягає в аналізі впливу масштабування на стратегії тестування. Із зростанням масштабу можуть виникати проблеми, такі як збільшення часу виконання тестів та складніше управління ресурсами. Дослідження цього впливу допомагає розробити ефективніші підходи до організації та виконання тестів у великому масштабі.

Нарешті, на основі аналізу можуть бути розроблені методи для забезпечення стабільності тестового середовища. Це може включати оптимізацію ресурсів, автоматичне масштабування інфраструктури, використання хмарних сервісів та інші стратегії розподілу тестових завдань [3].

Отже, стратегії тестування CRM-систем на основі мікросервісної архітектури повинні бути орієнтовані на виявлення та усунення недоліків у взаємодії між сервісами, на перевірку відновлення після збоїв та забезпечення стабільності та надійності системи. Автоматизація та контейнеризація можуть стати суттєвими факторами у забезпеченні ефективності тестування в умовах мікросервісної архітектури.

# Література

1. Seemann M., Steven van D. Dependency Injection Principles, Practices, and Patterns: Manning, 1st edition, 2019, 552 р.
2. A guide to testing microservices architecture: веб-сайт. URL: https: // reintech. io / blog / a-guide-to-testing-microservices-architecture (дата звернення: 06.05.2024).
3. Mamoojee T. Refining your test automation approach in a microservice architecture world: веб-сайт. URL: https: // applitools. com / blog / refining-test-automation-microservice-architecture/ (дата звернення: 05.05.2024).