УДК 004.6

Д.Ю. Пилипенко

ОСОБЛИВОСТІ ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ РОЗПОДІЛЕНИХ ІНФОРМАЦІЙНИХ СИСТЕМ

Вінницький національний технічний університет;

Анотація *Проаналізовано особливості розробки та тестування програмного забезпечення розподілених інформаційних систем. Визначені основні переваги та недоліки. Розглянуті різні технології та інструменти тестування. Наведено приклад тестування онлайн-магазину як інформаційної розподіленої системи.*

Ключові слова: розподілені інформаційні системи; програмне забезпечення; тестування; якість програмного забезпечення, програмна інженерія.

Abstract *The features of software development and testing of distributed information systems are analyzed. The main advantages and disadvantages are identified. Various technologies and testing tools are considered. An example of testing an online store as an information distributed system is given.*

Keywords: distributed information systems; software; testing; software quality, software engineering.

Вступ

Сучасний бізнес функціонує в епоху великих обсягів інформації, яка обробляється та активно використовується. Компанії, незалежно від їхнього розміру та сфери діяльності, постійно збирають, обробляють та аналізують великі обсяги інформації з різноманітних джерел. Це можуть бути дані про клієнтів, постачальників, внутрішні процеси, ринкові тенденції тощо.

Для ефективної роботи організаціям потрібні системи, які дозволяють об'єднати всю цю інформацію в єдину платформу. Це дозволяє приймати обґрунтовані рішення, оптимізувати бізнес-процеси та підвищити конкурентоспроможність. Однак, зі збільшенням обсягів даних та їх розподіленості за різними системами виникають нові виклики.

Інформація, необхідна для прийняття рішень, може знаходитися в різних місцях і форматах. Це можуть бути реляційні бази даних, NoSQL-бази даних, хмарні сховища, файли на локальних дисках тощо. Крім того, дані можуть бути структурованими (наприклад, дані в таблицях) та неструктурованими (наприклад, тексти, зображення, відео) [1].

Для вирішення цієї задачі використовують розподілені інформаційні системи. Вони дозволяють об'єднати дані з різних джерел, забезпечити доступ до них з будь-якої точки світу та масштабувати систему в міру зростання обсягів даних.

Задачі перевірки роботи розподілених інформаційних систем та їх програмного забезпечення передбачають підвищення надійності роботи системи, збільшити швидкість доступу до даних, забезпечити масштабованість системи, знизити витрати на обслуговування.

Для досягнення цих цілей використовуються різноманітні методи та технології, серед яких вибір ефективних ефективних алгоритмів для обробки даних; кешування; паралельна обробка даних; зменшення обсягу даних, що передаються; використання гнучких технологій управління процесами створення та використання програмного забезпечення розподілених інформаційних систем.

Розподілені інформаційні системи пройшли довгий шлях від простих мереж терміналів до сучасних хмарних платформ. Розуміння еволюції їх розвитку допомагає краще оцінити можливості та виклики, які стоять перед розробниками та тестувальниками програмного забезпечення. сьогодні.

Концепція термінальних систем полягала в тому, що користувачі підключалися до центрального комп'ютера (мейнфрейму) через прості термінали. Весь обчислювальний процес відбувався на мейнфреймі, а термінали слугували лише для введення даних та відображення результатів. Перевагою цього є ефективне використання потужностей, а недоліком – обмежена інтерактивність і залежність від центрального комп`ютера [1]. Сучасні розподілені інформаційні системи використовують мікросервісну архітектуру [2], яка замінила монолітну.

Розподілені системи пройшли довгий шлях від простих термінальних систем до складних хмарних платформ. Сучасні тенденції свідчать про те, що розподілені системи продовжуватимуть

розвиватися, стаючи все більш масштабними, гнучкими та інтегрованими. Але підтримка програмного забезпечення розподілених інформаційних систем вимагає використання різних технологій, високої кваліфікації фахівців.

Ключові переваги розподілених інформаційних систем – масштабованість, надійність, гнучкість, доступність.

Базові недоліки для розробників та адміністраторів розподілених інформаційних систем - складна розробка та підтримка, забезпечення безпеки, управління розподіленими ресурсами.

Створення програмного забезпечення для розподілених інформаційних систем та його тестування мають свої специфічні особливості, які відрізняють його від розробки монолітних застосунків [3].

Мета досліджень – визначення особливостей створення й тестування програмного забезпечення розподілених інформаційних систем.

Результати дослідження

Аналіз відомомих технологій та інструментів. Програмне забезпечення розподілених інформаційних систем передбачає дотримання спеціальних правил для мережевої взаємодії; незалежної роботи кожного модуля, безпеки роботи системи та компонентів; масштабованості та надійності.

Для цього необхідно сформувати загальні композиції та декомпозиції системи, визначити всі залежності та розподілені транзакції сформувати спеціальні сценарії для тестування.

Розподілені інформаційні системи використовують спеціальні технології та інструменти. Серед них такі мови програмування як Java, Python, Go, Node.js. Фреймворки – Spring, Django, Ruby on Rails, Node.js. Інструменти контейнеризації Docker, оркестрації – Kubernetes, бази даних - MySQL, PostgreSQL, MongoDB. Також використовуються спеціальні хмарні платформи – AWS, Azure, GCP. Мікросервісна архітектура базується на методах інтеграції SOA, RESTful API [4-10].

Принципи розробки програмного забезпечення розподілених інформаційних систем базуються на правилах розподіленості – модульності, абстракції (створення абстрактних інтерфейсів для взаємодії між компонентами; капсулювання (інкапсуляція даних і логіки всередині компонентів). Крім того, використовується розподілені методи управління станом системи та окремих застосунків, асинхронна комунікація між компонентами.

Тестування розподілених інформаційних систем має свої специфічні виклики, які відрізняють його від тестування монолітних додатків. Складність полягає в тому, що взаємодія між різними компонентами системи, які можуть бути розподілені по різних серверах або навіть географічних локаціях, створює додаткові фактори, які необхідно враховувати при тестуванні [11].

Розподілені системи зазвичай працюють у складних і динамічних середовищах, що ускладнює ізоляцію проблем і відтворення помилок.

Якість мережевого з'єднання може суттєво впливати на роботу системи, тому необхідно тестувати систему в різних мережевих умовах. Затримки в мережі можуть призводити до несподіваних результатів, тому тести повинні враховувати асинхронний характер взаємодії компонентів.

Дані можуть бути розподілені по різних базах даних або сховищах, що ускладнює їх синхронізацію і верифікацію.

Тести повинні перевіряти, як система поводиться при збільшенні навантаження.

Розглянемо основні тести для програмного забезпечення розподілених інформаційних систем.

Функціональне тестування – перевірка відповідності системи вимогам.

Навантажувальне тестування – оцінка продуктивності системи під великим навантаженням.

Стрес-тестування – перевірка стійкості системи до екстремальних навантажень.

Тестування на відмову – перевірка поведінки системи при виході з ладу окремих компонентів.

Інтеграційне тестування – перевірка взаємодії різних компонентів системи.

Тестування безпеки – перевірка системи на вразливості.

Серед інструментів для тестування розподілених інформаційних систем можна виділити такі [10-13]:

для автоматизації тестування: Selenium, JUnit, TestNG.

для моніторингу: Prometheus, Grafana.

для симуляції навантаження: JMeter, Gatling.

для тестування безпеки: OWASP ZAP, Burp Suite.

Дослідження особливостей тестування прозгрампного забезпечення на прикладі онлайн-магазину. Припустимо, що маємо онлайн-магазин, побудований на мікросервісній архітектурі. Кожен сервіс відповідає за свою функціональність – каталог товарів, кошик, оплата, доставка тощо.

Мета тестування – перевірити, як ці сервіси взаємодіють між собою, чи правильно обробляються замовлення і чи є система стійкою до навантажень. Розглянемо приклади реалізації сценаріїв тестування онлайн-магазину.

Функціональне тестування передбачає такі види перевірки:

коректності роботи кожного сервісу окремо (наприклад, чи правильно додається товар в кошик, чи коректно обчислюється вартість замовлення).

взаємодії між сервісами (наприклад, чи передаються правильні дані між сервісом каталогу і сервісом кошика).

сценаріїв користувача (наприклад, оформлення замовлення від початку до кінця).

Навантажувальне тестування реалізується за допомогою симуляції роботи великої кількості користувачів, які одночасно здійснюють покупки; перевірки, як система поводиться під великим навантаженням (чи не сповільнюється, чи не виникають помилки); визначення максимальної пропускної здатності системи.

Стрес-тестування передбачає такі етапи:

Створення екстремальних умов для перевірки стійкості системи (наприклад, відключення частини сервісів, перевантаження бази даних).

Визначення "точки зламу" системи.

Тестування на відмову – це симуляція відмов окремих компонентів системи (наприклад, відключення сервісу платежів) та перевірка, як система відновлюється після відмови.

Перевірка взаємодії між різними сервісами, включаючи мережеві протоколи та формати даних здійснюється під час інтеграційного тестування.

Інструменти для тестування онлайн-магазину також вибираються для кожного виду тестування.

Для автоматизації: Selenium, JUnit, TestNG, pytest.

Для навантажувального тестування: JMeter, Gatling.

Для оркестрації тестів: Kubernetes, Docker Compose.

Для моніторингу: Prometheus, Grafana.

Розглянемо сценарій тесту.

Користувач додає товар в кошик, оформляє замовлення і здійснює оплату.

Відкрити сторінку товару.

Додати товар в кошик.

Перейти до кошика і перевірити, що товар додався правильно.

Пройти процес оформлення замовлення.

Перевірити, чи було відправлено повідомлення про успішне оформлення замовлення.

Перевірити, чи зменшилася кількість товару на складі.

Перевірити, чи було сформовано рахунок на оплату.

Необхідно налаштувати тестове середовище, яке імітує реальну роботу системи.

Проаналізувати зміни в одному сервісі можуть вплинути на роботу інших.

Дані можуть бути розподілені по різних базах даних, що ускладнює їх верифікацію.

Необхідно враховувати можливі затримки в мережі.

Висновки

Тестування розподілених систем вимагає комплексного підходу та використання спеціалізованих інструментів. Створення й тестування програмного забезпечення повинно здійснюватись за допомогою гнучких технологій управління проєктом створення програмного продукту. Помилки в одній частині системи можуть вплинути на роботу всієї системи. Ретельне тестування дозволяє забезпечити високу якість і надійність розподілених інформаційних систем. Крім того, необхідно здійснювати всі види тестування для визначення метрик якості програмного продукту.

В планах подальших досліджень – сформувати стратегію автоматизованого та мануального тестування та методику формування сценаріїв, тестів, кейсів, аналітичних звітів тестування.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Talaver O.V. and Vakaliuk, T.A . Reliable distributed systems: review of modern approaches. *Journal of Edge Computing* [Online], 2(1),2023, pp.84–101. Available from: https://doi.org/10.55056/jec.586
2. Fowler M. Microservices URL: http://martinfowler.com/articles/microse rvices.html
3. Microservices URL: https://aws.amazon.com/ru/microservices/
4. Application Integration on AWS Integrate distributed systems and serverless applications with less code URL: https://aws.amazon.com/ru/products/applicationinte gration/?nc2=h_ql_prod_ap_ai
5. Introduction to microservices. URL: https://nginx.com/blog/introductiontomicroservices/
6. Using an API Gateway. URL:https://nginx.com/blog/buildingmicroservices - using-an-api-gateway/ 6. Service Discovery. URL: https://nginx.com/blog/servicediscoveryinamicroservices-architecture/
7. Docker. URL: https://docker.com/
8. Architecture using an Enterprise Service Bus. URL: http://www.ibm.com/d eveloperworks/library/ws-soaeda-esb/–
9. Kubernetes. URL: https://kubernetes.io/
10. Selenium WebDriver API. Available at: https://selenium-python.readthedocs.io/api.html
11. Spillner Andreas, Linz Tilo Software Testing Foundations: A Study Guide for the Certified Tester Exam- Foundation Level - ISTQB, 5th Edition, 2021. 390 p.
12. Pajankar Ashwin, Python Unit Test Automation: Automate, Organize, and Execute Unit Tests in Python, 2022. 209 p.
13. Full pytest Documentation. Available at: https://docs.pytest.org/en/7.1.x/contents.html

Пилипенко Дмитро Юрійович – здобувач вищої освіти третього рівня (phd), гр. 121-23а, факультет інформаційних технологій та комп'ютерної інженерії, Вінницький національний технічний університет, Вінниця, e-mail: dpilipenko@vntu.edu.ua

Науковий керівник – Коваленко Олена Олексіївна, к.т.н., доцент, доцент кафедри програмного забезпечення, Вінницький національний технічний університет, м. Вінниця, e-mail: ок@vntu.edu.ua.

Pylypenko Dmytro – Postgraduate Student (third-level higher education (PhD), student of group 121-23a, Department of Information Technologies and Computer Engineering, Vinnytsia National Technical University, Vinnytsia, e-mail: dpilipenko@vntu.edu.ua

Academic supervisor – Kovalenko Olena, Candidate of Technical Sciences, Associate Professor, Associate Professor of the Software Department, Vinnytsia National Technical University, Vinnytsia, e-mail: ok@vntu.edu.ua.