Категорія Рефакторинг Доступу

Струзік Владислав Анатолійович
ТОВ "ПЕРША УКРАЇНСЬКА ЛІЗИНГОВА КОМПАНІЯ"
Київ, Україна
struzik.vladislav@gmail.com

Access Refactorings Category

Vladislav Struzik
Persha Ukrayinska Lizynhova Kompaniya Ltd.
Kyiv, Ukraine
struzik.vladislav@gmail.com

***Анотація*—У даній роботі описані переваги та недоліки монолітного та мікросервісного шаблонів архітектури та умови, коли доцільно їх використовувати. На ряду з цим виділено окрему категорію операцій рефакторингу — рефакторинг доступу, а також наведений перелік операцій, що входять до даної категорії, можливість його використання при переходах між шаблонами архітектури та при виникненні подій, що пов'язані з політикою безпеки.**

***Abstract*—This paper describes the advantages and disadvantages of monolithic and microservice architecture patterns and when it is appropriate to use them. In addition, there is a new one category of refactoring operations — refactoring of access and its list of operations, the ability to use them in the migrations between architectural patterns and in the security-related events.**

Ключові слова—мікросервіси, сервіс-орієнтована архітектура, рефакторинг, рефакторинг доступа, моноліт, бази даних.

Keywords—microservices, service-oriented architecture, refactoring, refactoring of access, monolith, databases.

I. Вступ

Розвиток інформаційних технологій впливає на багато сфер життя сучасної людини. Процес створення програмних засобів є доволі складним і творчим і рутинним одночасно, але людський фактор не можливо оминути. Для покращення кінцевого програмного продукту виникла концепція рефакторингу, яка направлена на покращення елементів створюваного програмного продукту з метою зменшити витрати ресурсів на його вдосконалення. Рефакторинг є складовою частиною процесу розробки програмного коду, структури баз даних, а також має різні цілі застосування.

II. Основна частина

При проектуванні інформаційної системи одним з ключових моментів є вибір шаблону її архітектури. Найчастіше на початкових етапах розробки нового сервісу (бізнес-ідеї) розробники програмного забезпечення надають перевагу монолітному шаблону архітектури, оскільки основною з численних переваг є можливість дешевої розробки мінімально життєздатного продукту (англ. minimum viable product — MVP). При використанні монолітного шаблону архітектури ведеться розробка програмного продукту, в якому інтерфейс користувача та виконання прикладних задач обробки даних поєднанні у єдиний програмний модуль. Система, розроблена за таким шаблоном архітектури, автономна та працює в рамках однієї обчислювальної системи, без залежності від інших програмних додатків.

Основними перевагами використання монолітного шаблону архітектури є:

- простота розробки;
- простота розгортання;
- просте масштабування.

Варто звернути увагу на те, що при розширенні функцій системи та зміні команди розробників можуть виникнути наступні недоліки монолітного шаблону архітектури:

- складність підтримки та розширення великого програмного продукту;
- надмірне навантаження на засоби розробки;
- надмірне навантаження на середовище виконання;
- ускладненість безперервного розгортання;
- проблема масштабування — масштабування можливо тільки горизонтально, потенційне виникнення стану гонитви (англ. race condition, race hazard).

Проте вищеописані недоліки монолітного архітектурного шаблону вирішені в мікросервісному шаблоні, що є сучасним уявленням сервіс-орієнтованої архітектури програмного забезпечення.

Ресурси, створені з використанням мікросервісного шаблону архітектури, спроектовані як окремі сервіси в рамках однієї інформаційної системи, що взаємодіють між собою через стандартні засоби зв'язку (гіпертекстовий транспортний протокол, черги повідомлень тощо). Важливою особливістю мікросервісів є те, що кожен розробляється, тестується, розгортається і масштабується незалежно від інших.

Розробка інформаційної системи з використанням мікросервісного шаблону архітектури має ряд переваг:

- забезпечення безперервного розгортання системи через постійне оновлення інформаційної системи;
- усунення довгострокової відданості технологічному стеку;
- відносно невеликий розмір кожного мікросервісу;
- забезпечення ізоляції несправностей.

Також мікросервісний шаблон архітектури має наступні недоліки:

- складність розгортання;
- збільшене споживання ресурсів;
- виникнення потреби в інтеграційному тестуванні в рамках взаємодії між мікросервісами.

При проектуванні програмного продукту у відповідності до монолітного шаблону архітектури, використовують одну базу даних, що зберігає усі бізнес-важливі дані. В свою чергу, при мікросервісному шаблону архітектури мають місце два варіанти взаємодії з базою даних: “база даних на сервіс” (database per service) та “спільна база даних” (shared database). З назв підходів можно зрозуміти, що в першому випадку кожен сервіс працює виключно зі своєю базою даних, в другому випадку всі сервіси використовують одну базу даних.

Під час експлуатації та розширення монолітного програмного продукту розробники яскраво бачать вузькі місця, що потребують масштабування, і найчастіше таким вузьким місцем є запити до бази даних. Виникає необхідність виділення вузького місця в окремий сервіс, що є першим кроком до сервіс-орієнтованої архітектури. В загальному випадку масштабування баз даних здійснюється трьома варіантами: реплікація, шардинг та партиціонування. Партиціонування можливо застосовувати в рамках існуючої бази даних, проте при застосуванні реплікації та шардингу доречно вузьке місце виділити в окрему базу даних. Ще однією перевагою виділення в нову базу даних є відокремлення та ізоляція проблеми, що зменшить вплив на систему загалом. Виділення окремої бази даних з спільної при створенні окремого сервісу, виділення вузького місця в окрему базу даних, обмеження прав доступу до даних та інше — все це призвело до необхідності створення окремої категорії операцій рефакторингу — рефакторинг доступу.

Окремим випадком використання операцій рефакторингу доступу є події, що пов’язані з політикою безпеки підприємства, а саме:

- ротація паролів;
- компроментація паролів;
- підвищення/зменшення вимог до аутентифікації.

Категорія рефакторинг доступу акумулює в собі зміни в системі управління базою даних, що пов’язані з доступом до даних. До даної категорії належать наступні операції:

- зміна атрибутів аутентифікації;
- звуження області видимості об’єктів бази даних;
- розширення області видимості об’єктів бази даних;
- звуження привілеїв доступу;
- розширення привілеїв доступу;
- виділення схеми бази даних;
- злиття схем баз даних.

Висновок

Підводячи підсумки, важливо зазначити, що операції категорії рефакторингу доступу дають нам можливість контрольовано виконувати зміни у системі управління базою даних у частині обмеження доступу, створити регламентовані процеси реакції на події, що пов’язані з політикою безпеки..

References Література

[1] Microservice Architecture [electronic resource] - Access to the resource: https://microservices.io/patterns/microservices.html.

[2] Monolithic application [electronic resource] - Access to the resource: https://en.wikipedia.org/wiki/Monolithic_application.

[3] Database per service [electronic resource] - Access to the resource: https://microservices.io/patterns/data/database-per-service.html.

[4] Shared database [electronic resource] - Access to the resource: https://microservices.io/patterns/data/shared-database.html.

[5] Monolithic Architecture [electronic resource] - Access to the resource: https://microservices.io/patterns/monolithic.html.