УДК 004.4:339 DOI: https://doi.org/10.30837/IYF.IIS.2024.828

РОЗРОБКА КОМПОНЕНТІВ ІНФОРМАЦІЙНОЇ СИСТЕМИ БІРЖОВОЇ ТОРГІВЛІ

Ткаченко І. Є.
Науковий керівник – к.т.н., ст. викл. Яцик М. В.
Харківський національний університет радіоелектроніки, каф. СТ
м. Харків, Україна
e-mail: ihor.tkachenko1@nure.ua

This architecture for stock trading, where speed and reliability are critical, microservices allow you to optimize the reaction time to changes from stock trading and ensure a high level of availability. Moving to a microservices architecture can increase flexibility of stock trading platforms and reduce the time needed to develop new features. With high benefits to productivity and security in the financial sector, microservices can help maintain a high level of efficiency and security of systems.

У світі сучасних фінансових технологій, де швидкість, надійність та здатність масштабуватися є ключовими факторами успіху. Мікросервісна архітектура стає все більш популярним підходом для вирішення вимог цих даних типів задач.

Мікросервісна архітектура полягає в тому, що розбиває додаток на невеликі незалежні компоненти, які звуться мікросервісами. У такій архітектурі кожен мікросервіс повинен відповідати за конкретний функціонал і може бути розроблений, впроваджений та масштабованим незалежно від інших сервісів [1].

У системах біржової торгівлі мікросервісна архітектура дозволяє швидко реагувати на зміни на ринку, швидко впроваджувати новий функціонал та легко масштабувати систему для забезпечення високої пропускної здатності та надійності. Наприклад, можливість розгортання окремих мікросервісів на різних серверах дозволяє ефективно розподіляти навантаження та запобігати виникненню одномоментних відмов або розгорнути декілька екземплярів мікросервісів, які будуть слугувати, як запасний варіант, якщо головний мікросервіс перестане працювати, зробити так би мовити, кластер реплік [2].

Монолітна архітектура використовується для розробки програмного забезпечення. Основна проблема монолітних систем полягає в тому, що вони складаються з одного великого блоку коду, в якому всі функції та компоненти збираються разом. Це означає, що будь-які зміни, навіть невеликі, потребують перебудови та розгортання всього додатку, це може призвести до складнішої розробки, а зміни логіки одного модуля програми можуть впливати на код іншого модуля. Крім того, монолітні програми важко масштабувати, коли різні модулі мають вимоги, які конфліктують між собою.

Мікросервісна архітектура виявляється не такою ідеальною, як здається на перший погляд. Декілька з найбільш вагомих недоліків мікросервісної архітектури:

1. Збільшення витрат на інфраструктуру [3]. З поступовим збільшенням кількості компонентів з'являється необхідність у більшій кількості серверів та їх потужностях. Це може призвести до значного збільшення витрат на обладнання та утримання серверів.

2. Підвищена складність моніторингу. У монолітному додатку треба налагодити моніторинг тільки за одним компонентом, але у мікросервісній архітектурі може бути дуже багато компонентів, для кожного з яких треба налагоджувати моніторинг.

Мікросервісна архітектура стає все більш популярним підходом для розробки програмного забезпечення, особливо в середовищі великих технологічних компаній. Історія доводить, що такі гіганти, як Netflix, Twitter, Spotify та інші, вирішили перейти на мікросервісну архітектуру для побудови багатоповерхових монолітних продуктів. За даними, відомими з початку 2010-х років [4], багато з цих компаній розпочали перехід на мікросервісну архітектуру, що свідчить про її ефективність та надійність. У цей список також включаються такі компанії, як eBay, PayPal, Stripe, Amazon, хоча інформація про час їхнього переходу є конфіденційною.

Вибір між монолітною та мікросервісною архітектурою для будь-якого проекту зазвичай залежить від його розміру, складності та масштабів. Монолітна архітектура має сенс для невеликих додатків. Однак, коли проект зростає та стає більш складним, а також вимагає більшої відмовостійкості та масштабування, мікросервісна архітектура стає більш підходящим варіантом.

Список використаних джерел:

1. Microservices architecture. Mehmet Ozkaya: стаття про мікросервісну архітектуру. URL: https://medium.com/design-microservices-architecture-with-patterns/microservices-architecture-2bec9da7d42a (дата звернення: 06.09.2021).

2. Microservices replication. AppMaster, стаття про реплікації мікросервісів. URL: https://appmaster.io/glossary/microservices-replication (дата звернення 27.08.2023).

3. Безугла Г. Є., Хряпкін О. В. Розробка інфраструктури web-компонентів інформаційної системи // International Science Conference on Multidisciplinary Research. 2021/01. P. 996-1002.

4. Microservices use cases and real-world examples. Yuriy Rybkin: стаття про приклади використання мікросервісної архітектури. URL: https://codeit.us/blog/microservices-use-cases (дата звернення 16.03.2023).