ИНФОРМАТИКА

Вестник Сыктывкарского университета.
Серия 1: Математика. Механика. Информатика.
Выпуск 2 (35). 2020

УДК 004.03

АКТУАЛЬНОСТЬ ИСПОЛЬЗОВАНИЯ МИКРОСЕРВИСОВ ПРИ РАЗРАБОТКЕ ИНФОРМАЦИОННЫХ СИСТЕМ

Ю. В. Гольчевский, А. В. Ермоленко

Использование микросервисной архитектуры при разработке приложений является сегодня популярным подходом к созданию современных приложений. Он позволяет органично проводить горизонтальное масштабирование приложений и возлагает на команду разработчиков полную ответственность за разработанный сервис. По мнению авторов статьи, на этапе обучения будущих ИТ-специалистов требуется уделять особое внимание вопросам использования различных архитектур разработки, необходимо объяснять достоинства и недостатки различных архитектур, отрабатывать навыки разработки микросервисов с использованием различных подходов.

Ключевые слова: микросервисы, монолитная архитектура, сервис-ориентированная архитектура, flask.

Введение

Стремительный рост технологий в последнее время приводит к тому, что то, что еще недавно было экзотикой, фантастикой, сейчас внедряется в обыденную жизнь и становится привычным инструментом работы. В качестве примера можно привести технологии, связанные с экспертными системами, базами знаний, интерфейсами на основе управления голосом и жестами, нейронными сетями, см., например, [1; 2]. Возрастающие требования к программному обеспечению приводят к необходимости пересмотра и развития архитектур и подходов к проектированию и разработке информационных систем.

Достаточно долгое время преобладающим при разработке информационных систем был способ создания «монолитных» приложений, при котором изначально не выделялись ни пользовательский уровень, ни

© Гольчевский Ю. В., Ермоленко А. В., 2020.

уровень бизнес-логики. Далее появились подходы, в которых стали выделять несколько уровней (соответствует монолитной клиент-серверной архитектуре):

1) пользовательские интерфейсы;

2) уровень бизнес-логики системы;

3) уровень баз данных.

Первый уровень называют фронт-эндом (Front-End), второй и третий – бек-эндом (Back-End). Соответственно разработчики условно разбивались на две группы – Front-End-разработчики и разработчики Back-End. Главным недостатком монолитных приложений является необходимость сборки всего приложения после внесения изменений, что приводит к следующим последствиям [3]:

- существенно увеличивается время на развертывание системы и внесение изменений;
- сильная связь кода делает разработчиков зависимыми друг от друга, сбой в работе одного компонента приводит к отказу всей системы;
- большая кодовая часть не позволяет легко перейти на новые технологии, внесение изменений также затруднено, в результате приложения морально устаревают;
- затруднено масштабирование приложений;
- при внесении изменений количество ошибок значительно возрастает, что приводит к частым откатам системы.

Риски, связанные с существованием указанных недостатков, могут быть устранены путем деления приложений на более мелкие программные составляющие, что привело к возникновению способа проектирования и организации информационной архитектуры и бизнес-функциональности с использованием сервис-ориентированной архитектуры (СОА) [4; 5] и микросервисной архитектуры (МСА) в частности[1].

Цель данной работы состоит в том, чтобы дать представление о возможностях микросервисов при разработке программного обеспечения и

[1] Термин «микросервис» впервые обсуждался в 2011 г. [6].

обосновать необходимость развития умений и навыков работы с микросервисами в рамках обучения по направлениям, связанным с разработкой и эксплуатацией информационных систем.

Переход к микросервисной архитектуре

СОА – это парадигма организации и использования распределенных возможностей. Ключевым понятием является сервис – независимая от прочих сервисов компонента информационной системы. В качестве сервиса может выступать как целое приложение, так и отдельные его функциональные модули. Сервисы могут реализовывать как бизнес-логику, так и функции более низкого уровня, в том числе некоторые системные функции (см. рис. 1).

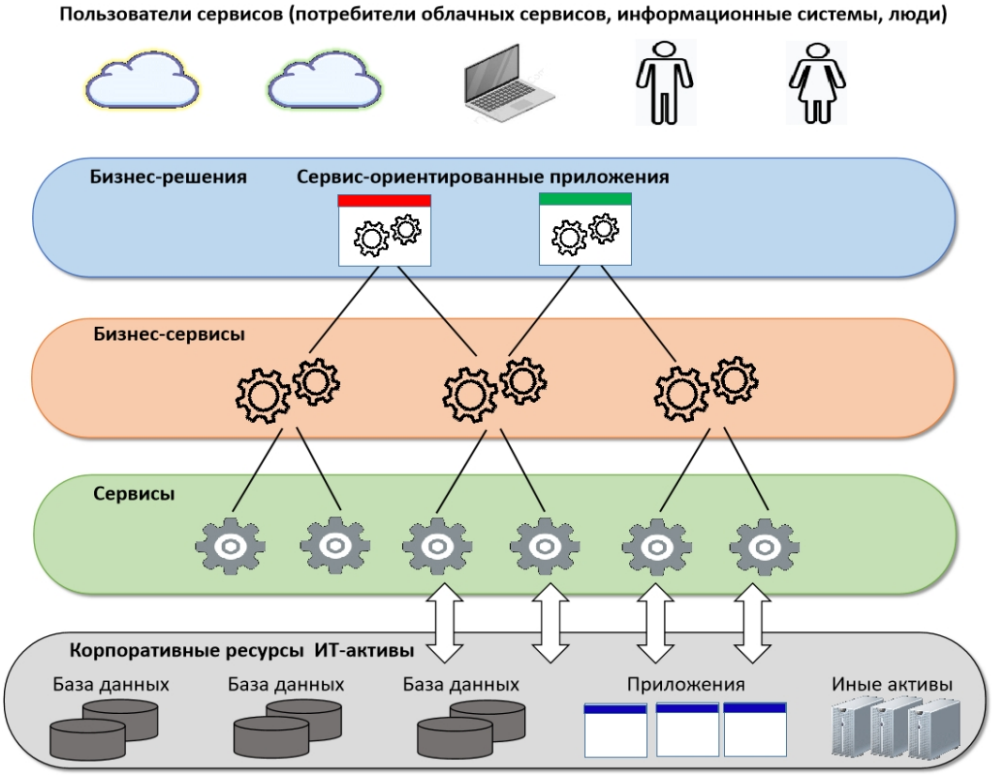

Рис. 1. Схема взаимодействия элементов СОА

Дальнейшим развитием СОА стала МСА, которая, по сути, является одним из способов реализации СОА. В данном случае приложение представляет собой набор небольших независимых сервисов, взаимодействующих через обмен сообщениями, например, по протоколу HTTP [7]. При этом микросервисы могут располагаться на разных серверах, быть

написанными на разных языках программирования с использованием различных технологий.

Важным элементом МСА и СОА является децентрализованное хранение данных, при котором каждый сервис может иметь свою базу данных, которая не взаимодействует с другими базами данных. Все взаимодействие идет через сообщения сервисов.

Развитию микросервисной архитектуры способствовал активный переход от настольных приложений к браузерным решениям, при котором необходимо ориентироваться только на используемые браузеры с применением библиотек веб-графики, языка JavaScript [8]. Существенное развитие МСА получили при появлении технологии WebSocket. Кроме этого, значимым преимуществом МСА является то, что микросервисы используют технологии с малой ресурсоемкостью. Они полагаются на REST, HTTP и другие форматы, предъявляющие незначительные требования к ресурсам и которые принято считать «нативными» для веб-разработки.

На рис. 2 представлена схема постепенного перехода от монолитной схемы к полной микросервисной архитектуре при разработке приложений.

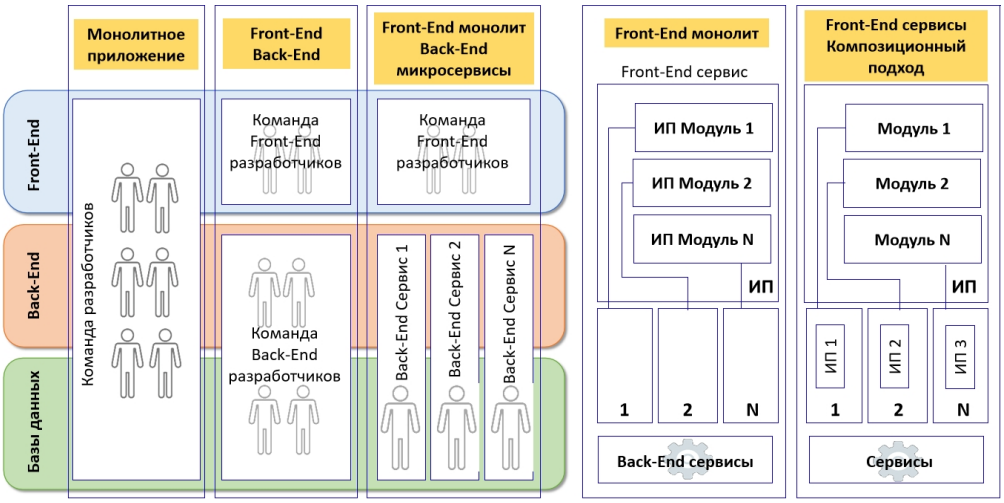

Рис. 2. Схема разработки приложений с использованием различных подходов к использованию микросервисов

Первоначально переход к МСА осуществлялся только в серверной части, оставляя интерфейс пользователя (ИП) монолитным. Дальнейшее развитие микросервисов привело к появлению микроинтерфейсов (вариант справа на рис. 2). При этом интерфейс пользователя заменя-

ется композицией микросервисов, каждый из которых посредством своего кода соединяется со своей базой данных. Таким образом у каждого микросервиса появляется своя база данных, интерфейсная и программная часть, реализующие свою бизнес-логику. Каждый микросервис разворачивается независимо, что обеспечивает автономность разработки и большую гибкость для всей системы. Также меняется и ответственность разработчиков. В последнем варианте, показанном на рис. 2, ответственность из «горизонтальной» (Front-End, Back-End и уровень баз данных) становится «вертикальной», так как разработчик несет полную ответственность за разработанный им сервис.

Микросервисы позволяют проводить горизонтальное масштабирование приложения, при этом сервисы могут взаимодействовать между собой синхронно с использованием HTTP/ REST или асинхронно с использованием брокеров сообщений типа Apache Kafka.

MCA не стоит использовать как единственный способ создания приложений, так как при использовании этой архитектуры может возникнуть целый ряд трудностей [9]. Отметим некоторые из них.

1. Необходимо решать вопросы, связанные с обеспечением безопасности данных. Например, может потребоваться шифрование при передаче данных, должны быть продуманы и качественно реализованы решения, связанные с аутентификацией сторон.

2. Необходимость шифрования, а также возникновение задержки передачи данных в сети, необходимость виртуализации приводят к замедлению работы системы в целом.

3. Оборотная сторона простоты в сопровождении заключается в том, что микросервисы менее надежны, чем монолитные системы. А, кроме того, при существенном увеличении сервисов возникают трудности конфигурирования системы в целом.

4. Трудно оценить стоимость перехода от монолитного приложения к микросервисному.

Пример реализации микросервиса

Современные языки программирования вводят поддержку разработки микросервисов путем использования специализированных библиотек. Рассмотрим в качестве примера реализацию работы с записями о пользователях методами HTTP/ REST на языке Python, широко

используемом практически по всех отраслях экономики [10; 11]. Для реализации сервиса используем библиотеку *flask*.

В листинге приведен код микросервиса, который выдает информацию о списке пользователей, конкретном пользователе, а также принимает информацию о новом пользователе.

```
from flask import Flask, jsonify, abort, request

app = Flask(__name__)

users = []  # Здесь формируем список пользователей.
# В данном примере он пуст

@app.route('/users', methods = ['GET'])
def get_users():
    return jsonify({'users': users})

@app.route('/users/<int:user_id>', methods = ['GET'])
def get_user(user_id):
    user = list(filter(lambda x: x['id'] == user_id, users))
    if len(user) == 0:
        abort(404)
    return jsonify({'user': user[0]})

@app.route('/users', methods=['POST'])
def create_user():
    if not request.json or not 'name' in request.json:
        abort(400)
    user = {
        'id': users[-1]['id'] + 1 if users else 1,
        'name': request.json['name']
    }
    users.append(user)
    return jsonify({'user': user}), 201

if __name__ == '__main__':
    app.run()
```

Листинг. Пример реализации микросервиса

Здесь приведен код целиком, можно запустить эту программу. Обращаем внимание, что исходный список user пуст, но он может быть

записан непосредственно или получен из базы данных.

При построении REST веб-сервисов необходимо просить клиентов отправлять свои регистрационные данные при каждом запросе. Аутентификация для микросервисов может быть реализована с использованием библиотеки *flask_ httpauth*.

После запуска представленного сервиса, набрав в браузере адрес

```
http://localhost:5000/users/
```

получаем ответ

```
{"user":{"id":2,"name":"Alex"}}
```

Если хотим получить информацию о конкретном пользователе, пишем адрес

```
http://localhost:5000/users/2
```

Однако в приведенном коде нет обработчика ошибок, поэтому в случае отсутствия пользователя с нужным идентификатором будет выдана ошибка 404.

Для проверки REST-запросов можно воспользоваться утилитой *curl*. Например, чтобы добавить пользователя с именем *Olga*, в командной строке набираем команду

```
curl -i -H "Content-Type: application/json" -X POST -d
"{"""name""":"""Olga"""}" http://localhost:5000/users
```

Заключение

МСА предлагает большие возможности по масштабированию и обслуживанию приложений, позволяет использовать разные технологии для разных сервисов. Однако это не означает, что необходимо забыть о других архитектурах. Один из общепризнанных подходов состоит в том, что необходимо начинать с монолитных приложений, а переходить на микросервисы только при возникновении трудностей с обслуживанием и масштабированием.

Современная практика показывает, что на фоне популярности микросервисов разработчики в ряде случаев забывают о других архитектурах и используют МСА в случаях, когда проект предпочтительнее было бы разрабатывать как монолитный. Например, микросервисная архитектура могла бы успешно применяться для решения ряда проблем, описанных в работах [12–14]. Это приводит к необходимости знакомить

будущих разработчиков информационных систем с различными архитектурами еще на этапе обучения, показывать достоинства и недостатки в рамках лекционного и лабораторного материала.

В статье как пример показан синхронный обмен данными между приложениями, однако в случае большого количества сообщений более востребованными оказываются асинхронные методы.

Авторы статьи убеждены, что в рамках обучения по направлениям, связанным с информационными технологиями, в частности по направлениям «Прикладная информатика» и «Прикладная математика и информатика», необходимо развивать у обучающихся навыки создания информационных систем с использованием различных подходов к созданию приложений. Данный материал рекомендуется рассматривать в рамках дисциплин «Проектирование информационных систем» и «Web-интеграция».

Список литературы

1. **Koshy R., Padalkar A., Nikam N., Jain V.** Easy verdict : Digital assistant to resolve criminal litigation // *10th International Conference on Advances in Computing, Control and Telecommunication Technologies, ACT 2019. Pp. 17–23.*

2. **Golchevskiy Yu., Yermolenko A.** Knowledge Base as an Integral Attribute of a Modern Company // *International Conference on Economics, Management and Technologies (ICEMT 2020). Advances in Economics, Business and Management Research, Atlantis Press. Vol. 139. Pp. 548–552.*

3. **Холодок Д. А., Пресняцкий В. Ю., Лецук Р. А.** Микросервисы как архитектурный стиль // *Образование и наука в России и за рубежом. 2019. 14 (62). С. 214–218.*

4. **Платонов Ю. Г.** Анализ перспектив перехода информационных систем на сервисно-ориентированную архитектуру // *Проблемы информатики. 2011. 4 (12). С. 56–65.*

5. **Костров И. А., Ковшов Е. Е.** Сервисно-ориентированная архитектура приложений как средство организации распределенных систем в среде слабоструктурированных данных // *Вестник МГТУ Станкин. 2012. 3 (22). С. 140–144.*

6. **Джамшиди П., Льюис Д., Паль К., Мендонча Н., Тилков С.** Микросервисы: пройденный путь и дальнейшие цели // *Открытые системы. СУБД. 2018. 3. С. 17–21.*

7. **Календарев А.** Современная веб-архитектура. От монолита к микросервисам // *Системный администратор. 2017. 1–2 (170–171). С. 80–83.*

8. **Schäffer E., Mayr A., Fuchs J., Sjarov M., Vorndran J., Franke J.** Microservice-based architecture for engineering tools enabling a collaborative multi-user configuration of robot-based automation solutions // *Procedia CIRP 86, 2020. Pp. 86–91.*

9. **Ларрусеа Х., Сантамариа И., Коломо-Паласьос Р., Эберт К.** Микросервисы // *Открытые системы. СУБД. 2018. 3. С. 10–12.*

10. **Шелудько В. М.** Основы программирования на языке высокого уровня Python : учебное пособие. Ростов н/Д; Таганрог: Южный федеральный университет, 2017. 147 с.

11. **Ермоленко А. В., Осипов К. С.** О применении библиотек Python для расчета пластин // *Вестник Сыктывкарского университета. Сер. 1: Математика. Механика. Информатика. 2019. Вып. 4 (33). С. 86–95.*

12. **Гольчевский Ю. В., Северин П. А.** Безопасное Web-программирование: безопасность CMS : учебное пособие. Сыктывкар: Сыктывкарский государственный университет, 2013. 68 с.

13. **Гольчевский Ю. В., Сургуладзе А. М.** Опыт разработки и эксплуатации системы онлайн-приема электронных заявлений от абитуриентов // *Открытое образование. 2012. 6. С. 57–63.*

14. **Бабенко В. В., Гольчевский Ю. В.** Концепция информационного пространства кафедры вуза на основе веб-портала // *Открытое образование. 2014. 2 (103). С. 46–50.*

Summary

Golchevskiy Yu. V., Yermolenko A. V. The relevance of using microservices in the development of information systems

The use of microservice architecture in application development is a popular approach to creating modern applications. It allows you to seamlessly carry out horizontal scaling of applications and assigns the development team full responsibility for the developed service. According to the authors of the article, at the stage of training future IT specialists, it is necessary to pay special attention to the use of various development architectures, it is necessary to explain the advantages and disadvantages of various architectures, and to develop microservice development skills using various approaches.

Keywords: service-oriented architecture, monolithic architecture, microservices, flask.

References

1. **Koshy, R., Padalkar, A., Nikam, N., Jain, V.** Easy verdict: Digital assistant to resolve criminal litigation, *10th International Conference on Advances in Computing, Control and Telecommunication Technologies*, ACT 2019, pp. 17–23.

2. **Golchevskiy Yu., Yermolenko A.** Knowledge Base as an Integral Attribute of a Modern Company, *International Conference on Economics, Management and Technologies (ICEMT 2020). Advances in Economics, Business and Management Research*, Atlantis Press, Vol. 139, pp. 548–552.

3. **Kholodok D. A., Presnyatsky V. Yu., Letsuk R. A.** *Mikroservisy kak arkhitekturnyy stil'* (Microservices as architectural style), Education and science in Russia and abroad, no 14 (62), 2019, pp. 214–218.

4. **Platonov Yu. G.** Analiz perspektiv perekhoda informatsionnykh sistem na servisno-oriyentirovannuyu arkhitekturu (Analysis of the prospects for the transition of information systems to a service-oriented architecture), *Problems of Informatics*, 2011. no 4 (12), pp. 56–65.

5. **Kostrov I. A., Kovshov E. E.** Servisno-oriyentirovannaya arkhitektura prilozheniy kak sredstvo organizatsii raspredelennykh sistem v srede slabostrukturirovannykh dannykh (Service-oriented architecture

asan approachto building distributed systems in the environment of semi-structured data), *Vestnik MSTU «STANKIN»*, 2012, no 3, pp. 140–144.

6. **Jamshidi P., Lewis D., Pal K., Mendoncha N., Tilkov S.** Mikroservisy: proydennyy put' i dal'neyshiye tseli (Microservices: the path traveled and future goals), *Open Systems. DBMS*, 2018, no 3, pp. 17–21.

7. **Kalendarev A.** Sovremennaya veb-arkhitektura. Ot monolita k mikroservisam (Modern web architecture. From monolith to microservices), *System Administrator*, 2017. no 1–2(170–171), pp. 80–83.

8. **Schäffer E., Mayr A., Fuchs J., Sjarov M., Vorndran J., Franke J.** Microservice-based architecture for engineering tools enabling a collaborative multi-user configuration of robot-based automation solutions, Procedia CIRP 86, 2020, pp. 86–91.

9. **Larrucea X., Santamaria I., Colomo-Palacio R., Ebert Ch.** Mikroservisy (Microservices), *Open Systems. DBMS*, 2018, no 3, pp. 10–12.

10. **Sheludko V. M.** *Osnovy programmirovaniya na yazyke vysokogo urovnya Python : uchebnoye posobiye* (Fundamentals of programming in the high-level Python language : textbook), Rostov-on-Don, Taganrog: Southern Federal University, 2017. 147 p.

11. **Yermolenko A. V., Osipov K. S.** O primenenii bibliotek Python dlya rascheta plastin (On using Python libraries to calculate plates), *Bulletin of Syktyvkar University. Series 1: Mathematics. Mechanics. Informatics*, 2019, 4 (33), pp. 86–95.

12. **Golchevskiy Yu. V. Severin P. A.** *Bezopasnoye Web-programmirovaniye: bezopasnost' CMS : uchebnoye posobiye* (Secure Web Programming: CMS Security : textbook), Syktyvkar, Syktyvkar State Univ., 2013.

13. **Golchevskiy Yu. V., Surguladze A. M.** Opyt razrabotki i ekspluatacii sistemy onlajn priema elektronnyh zayavlenij ot abiturientov (Developing and operating experience in online admission system for prospective university students), *Open Education*, 2012, no 6(95), pp. 57–63.

14. **Babenko V. V., Golchevskiy Yu. V.** Kontseptsiya informacionnogo prostranstva kafedry VUZa na osnove Web-portala (Concept of web-based university department information space), *Open Education*, 2014, no 2(103), pp. 46–50.

Для цитирования: Гольчевский Ю. В., Ермоленко А. В. Актуальность использования микросервисов при разработке информационных систем // *Вестник Сыктывкарского университета. Сер. 1: Математика. Механика. Информатика. 2020. Вып. 2 (35). С. 25–36.*

For citation: Golchevskiy Yu. V., Yermolenko A. V. The relevance of using microservices in the development of information systems, *Bulletin of Syktyvkar University. Series 1: Mathematics. Mechanics. Informatics*, 2020, 2 (35), pp. 25–36.

СГУ им. Питирима Сорокина *Поступила 15.05.2020*