DOI 10.37539/2949-1991.2023.7.7.006
УДК 004.051

Егоров Юрий Сергеевич,
старший преподаватель кафедры “Электроника и сети ЭВМ”
Нижегородский государственный технический университет им. Р.Е. Алексеева, Нижний Новгород, Россия

Рындов Сергей Николаевич, студент
Нижегородский государственный технический университет им. Р.Е. Алексеева, Нижний Новгород, Россия

Вайнбаум Денис Алексеевич, студент
Нижегородский государственный технический университет им. Р.Е. Алексеева, Нижний Новгород, Россия

КОНЦЕПТУАЛЬНОЕ ПРОЕКТИРОВАНИЕ И ТЕХНИЧЕСКАЯ РЕАЛИЗАЦИЯ МИКРОСЕРВИСНОЙ АРХИТЕКТУРЫ ИННОВАЦИОННЫХ ИНФОРМАЦИОННЫХ СИСТЕМ

Аннотация. В статье описаны аспекты формирования микросервисной архитектуры инновационных информационных систем и приложений, ее преимущества и недостатки для бизнес-процессов. Рассмотрены возможности создания гибкого, безопасного и масштабируемого приложения. Описаны важные аспекты проектирования, реализации и развертывания микросервисного приложения для предотвращения критических ситуаций.

Ключевые слова: концептуальное проектирование, информационная система, микросервис, сервис-ориентированная архитектура, программное обеспечение.

Введение

Использование информационных технологий привело к стремительному развитию ИТ отрасли во всем мире, став определяющим фактором инновационного развития многих государств. Одной из основных проблем современных информационных систем является увеличение скорости работы, гибкости, легкости замены и обслуживания компонентов без остановки работы всей системы. Решением этой проблемы является микросервисная архитектура (МСА), которая сочетает в себе преимущества сервисной архитектуры и способно удовлетворить потребности рынка.

Концептуальное проектирование информационных систем

Для эффективной реализации информационных систем, в т.ч. инновационных (не имеющих аналогов) приложений, необходимо провести концептуальное проектирование микросервисной архитектуры, включающее анализ предметной области, определение процессов и построение модели системы. Микросервисы являются небольшими, автономными компонентами, которые обеспечивают максимальную эффективность процесса и взаимодействуют через API. Использование контейнеров для развертывания и виртуализации процесса также является важным требованием. Конечная модель должна учитывать интеграцию с другими системами и окружением, чтобы создать гибкую и эффективную информационную систему, соответствующую потребностям пользователя и требованиям бизнеса. На рисунке 1 представлена

концептуальная модель абстрактной инновационной информационной системы, построенной на базе микросервисной архитектуры.

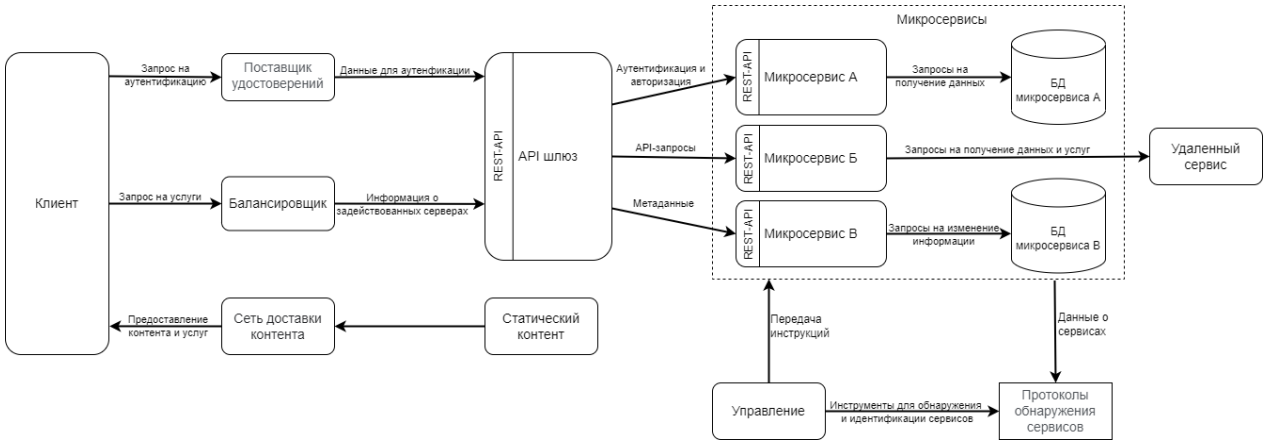

Рис. 1. Концептуальная модель абстрактной инновационной информационной системы, построенной на базе микросервисной архитектуры

Техническая реализация микросервсиной архитектуры

Для реализации информационной системы на базе микросервисной архитектуры чаще всего применяются технологии виртуализации и контейнеризации. Технология виртуализация использует виртуальные машины, представляющие из себя программный слой, который обеспечивает полную эмуляцию низкоуровневых аппаратных устройств. Главной проблемой виртуальных машин является большое потребление ресурсов вычислительной машины. Технологии контейнеризации используют контейнеры – изолированную виртуальную среду, которая создается поверх операционной системы (ОС). Виртуальной среде не нужно эмулировать низкоуровневые аппаратные устройства и за счёт этого среда потребляет значительно меньше ресурсов, в сравнении с виртуальной машиной.

Паттерны и принципы

При внедрении микросервисов можно столкнуться с проблемами. Общие закономерности их решения легли в основу паттернов микросервисной разработки. Они включают паттерны декомпозиции, рефакторинга, управления данными, коммуникации, пользовательского интерфейса, обнаружения, развертывания, повышения отказоустойчивости и мониторинга. Примеры паттернов включают разбиение по бизнес-возможностям и разбиение по поддоменам для избежания божественных классов. Паттерн API Gateway обеспечивает единую точку входа для клиента, но может стать узким местом системы. Шаблон цепочки позволяет микросервисам вызывать другие микросервисы, но не следует делать цепочку слишком длинной.

Варианты применения микросервисной архитектуры

Применение микросервисной архитектуры в проекте может быть полезно как на начальном этапе разработки, так и на этапе декомпозиции уже готового приложения. На начальных этапах микроссервисная архитектура позволяет задать правильный вектор развития всего проекта и исправить недостатки уже существующего приложения, на этапе переноса или рефакторинга этого приложения.

Для задачи переноса приложения с монолитной архитектуры на микросервисную, в первую очередь, следует ограничить зону ответственности каждого отдельного функционала. Микросервис не должен выполнять множество различных задач. При необходимости микросервисы объединяются в подсистемы, что упростит мониторинг за работой будущего приложения. На этом этапе необходимо обеспечить безопасность разрабатываемого приложения. При проектировании микросервисов, важным аспектом является факт устойчивости разрабатываемого приложения к сбоям. Сущность микросервисной архитектуры способствует созданию устойчивого приложения, но не гарантирует этого. В случае сбоя микросервиса или некорректной

обработки запроса в другой системе, возможен выход из строя целой подсистемы.

После реализации логики обеспечения внутренней безопасности микросервисов, наступает этап создания локальной сети микросервисов. Если к внутренней структуре приложения можно будет получить доступ, то можно вывести из строя работу целого приложения, получить права суперпользователя, заполучить личные данные пользователей и т.п. Поэтому внутренняя система должна быть полностью скрыта от внешнего мира в локальной сети, доступ к которой должен быть старого через фронтенд.

Заключение

Микросервисная архитектура применяется, как правило, в крупных проектах для обеспечения гибкости и масштабирования. Паттерны помогают минимизировать ее недостатки. Для автоматизации разверывания и управления микросервисами используются технологии на базе контейнеров и виртуальных машин, выбор зависит от масштаба проекта. Настройка серверной части важна для упрощения мониторинга и обеспечения безопасности работы приложения.

Список литературы:

1. Швакин В.Ю., Куликова Л.Л. О некоторых аспектах инвестиционно-инновационных процессов в сфере информационных технологий [Электронный ресурс]/ Швакин В.Ю., Куликова Л.Л. // Вестник ИрГТУ»: электрон. науч. журн. – 2013. - №5. - Режим доступа: https://vestnik.utmn.ru/upload/iblock/2fb/2_Tokareva _No11_ekonomika_2012-2.pdf (дата обращения 18.10.2020).

2. Ньюмен С. Создание микросервисов = Building Microservices. — СПб.: Питер, 2016. — 304 с. — ISBN 978-5-496-02011-4

3. Лобанов Е.В. Инновационная деятельность компаний в сфере информационных технологий [Электронный ресурс] / Лобанов Е.В. //

Ползуновский альманах. 2010. №3. - Режим доступа: http://masters.donntu.org/2012/iem/bailo/library /ar1.pdf (дата обращения 08.10.2020).

4. Храмушина В.А. Особенности инноваций в сфере ИТ-услуг // Материалы VII Международной научно-практической конференции «Новые импульсы развития: вопросы научных исследований». – Москва: КДУ, 2021. С. 171-179

5. Эшби У.Р. Введение в кибернетику.– М.: Иностранная литература, 1959. – 432 с.

6. Experiences from Failing with Microservices [Электронный ресурс].- Режим доступа https://www.infoq.com/news/2014/08/failingmicroservices/

7. Виртуализация [Электронный ресурс].- Режим доступа https://www.sim-networks.com/ru/blog/hypervisors-vmware-kvm-xen-openvz

8. Микросервисные паттерны проектирования [Электронный ресурс].- Режим доступа https://habr.com/ru/company/piter/blog/275633/

9. 26 основных паттернов микросервисной разработки [Электронный ресурс].- Режим доступа https://mcs.mail.ru/blog/26-osnovnyh-patternov-mikroservisnoj-razrabotki