УДК 004.01

ВЫБОР АРХИТЕКТУРЫ ДЛЯ ИНТЕГРАЦИИ СИСТЕМ

Шмелев П.Ю.

Пермский национальный исследовательский политехнический университет, г. Пермь, Россия

Научный руководитель: Прозорова Л.Ю. – канд. эконом. наук, доцент кафедры ИТАС

Аннотация. Интеграция приложений – неотъемлемая часть для успешного функционирования любого IT проекта. При выборе архитектуры для реализации интеграции необходимо обращать внимания на возможности компании. В данной статье будут сравниваться две архитектуры – это ESB и микросервисы. ESB содержит шину, через которую происходит все взаимодействие с приложениями, в то время как микросервисы не имеют общую шину и общаются напрямую с приложением. Именно эта ключевая особенность влияет на выбор архитектуры, потому как не всегда есть возможность менять код в интегрируемых приложениях.

Ключевые слова: Интеграция, Сервисная шина предприятия, ESB, микросервисы, преимущества, недостатки.

Введение. Практически любое предприятие работает сегодня с распределенной ИТ-инфраструктурой, которая объединяет разнородные платформы и прикладные решения, включая и унаследованные приложения. Кроме того, для современного предприятия весьма актуальна задача поддержки взаимосвязей с партнерами в рамках корпоративных информационных систем, а также обеспечения на уровне инфраструктуры адекватной реакции на процессы слияния или приобретения компаний. Перед выбором приложения, которое будет поддерживать интеграцию, встает вопрос о выборе архитектуры. Рассмотрим две похожие архитектуры, это *ESB* и микропроцессоры, которые подходят для интеграции.

ESB. Сервисная шина предприятия (*enterprise service bus, ESB*) - связующее программное обеспечение, обеспечивающее централизованный и унифицированный событийно-ориентированный обмен сообщениями между различными информационными системами на принципах сервис-ориентированной архитектуры [1]. Главной отличительной чертой использования данной архитектуры является повторное использование сервисов и корпоративная шина.

Ниже на рисунке 1 представлена схема работы архитектуры.

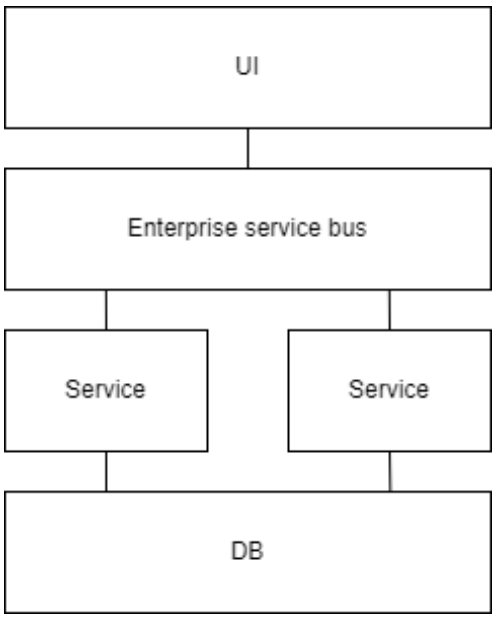

Рисунок 1 – Архитектура ESB

Все сообщения, полученные шиной, обрабатываются и в зависимости от контента отправляются в определенный сервис. Отсюда и идет первый недостаток данной архитектуры – «бутылочное горлышко». В случае, когда одновременно на шину придет большое количество сообщений – шина будет загружена обработкой, что приведет к задержке обработки сообщений. Следующий недостаток – сложность разработки. Данную систему придется разрабатывать довольно продолжительное время. Необходимо проработать логику взаимодействия сервисов. Из преимуществ можно отметить несколько. Во-первых, повторное использование сервисов – информация, полученная в результате работы одного сервиса, может повторно несколько раз использоваться, как например вариант с авторизацией. Пройдя авторизацию один раз пользователю не нужно будет выполнять авторизацию повторно при каждой введенной команде. Во-вторых, простота подключения сервисов. В приложении достаточно указать, что нужно отправлять данные в шину. Остальная же логика передачи данных настраивается непосредственно в шине. В-третьих, конвертация данных. Когда шина будет готова, она может быть связующим элементом между сервисами, которые используют разные архитектурные подходы: *REST* и *SOAP* [2].

Микросервисы. Микросервисная архитектура, наоборот, избегает повторного использования, применяя методологию «предпочтительнее дублирование», а не независимость от других сервисов. Повторное использование предполагает связанность, а данная архитектура старается ее избегать. Это достигается путем разбиения системы на сервисы с конечным контекстом (законченной бизнес-задачей). Принцип работы архитектуры представлен на рисунке 2 ниже.

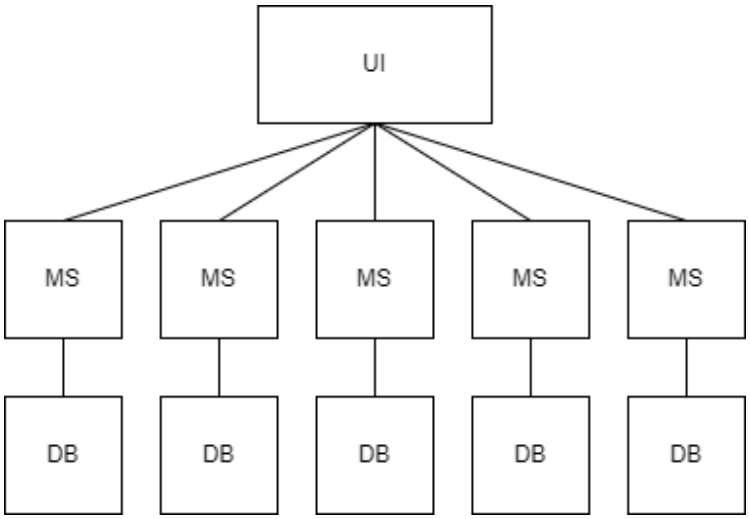

Рисунок 2 – Архитектура MSA

В данном случае приложение само определяет, к какому сервису необходимо подключиться. Преимуществом данной архитектуры является надежность. Если выйдет из строя один микросервис, но на работе остальных это никак не отобразится. Следующим преимуществом архитектуры является простота разработки. Так как микросервис – это самостоятельная единица, которая никак не связана с остальными сервисами – нет необходимости писать универсальный код, который бы подходил под разные сервисы. Так же немало важным плюсом является «полная картина бизнес-логики». Каждый сервис представляет как самостоятельную единицу. Бизнес-процесс, который представляет данный процесс, предоставлен от начала до конца. Это упрощает процесс понимания кода сотрудников и сокращает количество меж сервисных вызовов. Из недостатков можно выделить следующие: сложность – разделение приложения на независимые микросервисы влечет за собой больше артефактов управления. Этот тип архитектуры требует тщательного планирования, огромных усилий, командных ресурсов и навыков. Следующим недостатком является проблема безопасности. В микросервисном приложении каждый функционал, который взаимодействует через API извне, увеличивает вероятность атак.

Выбор архитектуры. В случае с микросервисной архитектурой для интеграции пришлось бы значительно дорабатывать код приложений для отправки данных под каждый микросервис. Это не всегда удобно, особенно если приложение принадлежит сторонней компании и, если этих приложений большое количество. К тому же, в отличие от *ESB*, микросервисы необходимо разрабатывать под свою систему самостоятельно с нуля, в то время как *ESB* решения уже существуют.

В случае с *ESB* – дорабатывать код придется меньше, так как отправка происходит в одну точку – в шину. Это позволяет настраивать интеграцию непосредственно в самой шине без вмешательства в код интегрируемых приложений. Так же присутствует такой инструмент, как конвертация данных. В микросервисной архитектуре это тоже возможно сделать, но в данном случае придется разрабатывать логику конвертации для каждого приложения отдельно.

Заключение. Таким образом можно сделать следующие выводы. Если необходима надежность, простота разработки, но нет времени на длительную разработку *ESB* (либо средств на приобретение готового решения) и доступны для изменения код приложений, то можно выбрать микросервисную архитектуру. В случае, если интегрироваться будут сторонние проекты, то лучше выбрать архитектуру *ESB*.

Список литературы

1. *Сервисная шина предприятия // Википедия. [Электронный ресурс]. – 2020. Режим доступа: https://ru.wikipedia.org/wiki/Сервисная_шина_предприятия. Дата обращения: 22.02.2022;*

2. *SOA vs MSA // Курс «Микросервисная архитектура». [Электронный ресурс]. 2019. Режим доступа https://microarch.ru/blog/soa-vs-msa. Дата обращения: 23.02.2022.*

UDC 004.1

CHOOSING AN ARCHITECTURE FOR SYSTEM INTEGRATION

Shmelev P.Y.

Perm National Research Polytechnic University, Russia, Perm

Prozorova L.Y. – Candidate of Economic Sciences, Associate Professor of the Department of Information Technologies and Automated Systems (ITAS)

Annotation. Application integration is an integral part for the successful functioning of any IT project. When choosing an architecture for implementing integration, it is necessary to pay attention to the company's capabilities. This article will compare two architectures - ESB and microservices. The ESB contains a bus through which all interaction with applications takes place, while microservices do not have a common bus and communicate directly with the application. It is this key feature that influences the choice of architecture, because it is not always possible to change the code in integrated applications.

Keywords: Integration, Enterprise service bus, ESB, microservices, advantages, disadvantages.