Микросервисы для экономных, или Искусство разумной роскоши

В настоящее время микросервисный подход является абсолютным хайпом. О микросервисах говорят везде. Микросервисы ругают, микросервисами восторгаются. Единственное, что можно сказать точно: микросервисный подход интересен для всех

Текст
ЛАРИСА СТАНКЕВИЧ,
ТЕХНИЧЕСКИЙ ДИРЕКТОР
«АРТ-ФИНТЕХ», Д.Т.Н.



Все технологические лидеры IT-индустрии на данный момент либо уже перешли на микросервисные рельсы, либо уверенно двигаются в этом направлении. Это не случайность, поскольку такой подход дает очень много плюсов для бизнеса. Из самого очевидного — это неимоверно короткий срок time to market, неограниченные возможности гибкой автоматизации процессов любой сложности, отсутствие ограничений на выбор лучших средств разработки и многое-многое другое.

Микросервисы — прекрасная технология, но в настоящее время бытует мнение, что этот подход слишком сложный, дорогой и по силам только самым крупным игрокам рынка. Этот тезис вызван тем, что микросервисный подход наиболее активно используют лидеры и гиганты рынка IT-индустрии с огромным штатом разработчиков, например СберБанк и ВТБ.

На самом деле микросервисы не должны пугать своей недоступностью — это не так страшно и не так дорого. Об этом и хочется рассказать.

Наша компания специализируется именно в области микросервисных решений. Результатом нашей работы является микросервисная АБС «АРТ-Финтех», и не только она.

Поскольку у нас нет ресурсов, сопоставимых с гигантами, нам пришлось искать пути сокращения издержек для работы с микросервисами. И надо сказать, что нам вполне удалось достичь поставленной цели. Сама разработка решения была начата пять лет назад, и уже четыре года ряд продуктов находится в промышленной эксплуатации. Так что микросервисная АБС — уже свершившаяся реальность, вполне опробованная промышленной эксплуатацией, что немаловажно.

Теперь о проблемах и заблуждениях, связанных с микросервиснной разработкой.

Первое заблуждение — для разработки микросервисов требуется огромная армия программистов. Считается, что микросервисы нужно программировать «руками», а сама по себе микросервисная система очень дорога, потому что программистов нужно много и каждый из них стоит «очень много денег». Мысль понятна: в первом случае программирование

в рамках одной базы данных, одного приложения, во втором (и это пугает) нужно разработать много мелких программ, которые должны как-то взаимодействовать между собой.

Сейчас попробуем рассказать о том, почему это может быть не так, если пользоваться современными технологиями и современными подходами.

Решение проблемы — генерация кода. Наша компания широко пользуется собственными разработками в области автоматической генерации кода. По сути, речь идет о том, что программисты нашей компании разрабатывают только шаблоны для генерации кода. Далее в дело вступают аналитики, которые прекрасно знают предметную область и настраивают логику работы микросервисов уже в low-code-стиле из этих шаблонов. Например, условие для формирования тарифа — это маленький шаблончик, содержащий код на Java, и делает его программист. А вот сказать, для какого тарифа в тарифном сборнике использовать именно это условие и сколько процентов от суммы транзакции это составит, удобнее именно аналитикам. Общая экономия на генерации кода — два-три дня вместо двух-трех месяцев, как при написании кода «руками».

Сгенерированный код ничем не будет отличаться от написанного программистом, и его при желании можно править. Однако мы приняли административное решение, которое запрещает сотрудникам нашей компании править код после генерации. Это привело к тому, что наши средства настройки позволяют генерировать код даже для самых сложных случаев автоматизации. Дополнительный плюс генерации кода состоит в том, что изменяется процесс разработки, если в классическом написании кода «руками» сам процесс включает в себя code review. В случае генерации кода code review выполняется только для шаблонов кода, то есть однократно для каждого типа конструкторов. То же относится и к проверкам по безопасности кода.

Второе заблуждение — интеграция микросервисов это неимоверно сложно и требует неимоверно больших ресурсов. В недавнем прошлом так и было — микросервисы интегрировались при помощи хореографии, то есть каждый микросервис должен иметь информацию о других, участвующих в процессе и связанных с ним. Соответственно с ростом количества микросервисов задача становилась все более сложной.

Решение проблемы — использование оркестровки вместо хореографии. В настоящее время фактически все компании, которые работают с микросервисами, переходят на стиль оркестровки, то есть отдельных оркестровочных микросервисов, которые управляют всей работой.

В наших решениях мы используем генерацию кода оркестовочных микросервисов на основании описаний BPMN-роутов. Сгенерированный код является полнофункциональным, и нет необходимости править его руками. Наше решение Smart-ESB кроме генерации кода интеграционных микросервисов ведет реестры микросервисов, включая их API.

Дополнительный плюс такого подхода заключается в том, что автоматизировать с помощью Smart-ESB можно не только интеграции, но и любые сложные бизнес-процессы, связанные с продуктовым и финансовым учетом. И если интеграция выполняется чаще всего программистами, то бизнесовые задачи чаще достаются системным аналитикам, которые хорошо понимают процессы и их логику.

Третье заблуждение — микросервисные системы очень сложно разворачивать и поддерживать. И для этого нужна армия дефицитных и дорогих DevSecOps. В общем случае так и есть. Гораздо проще запустить один монолит, чем целую кучу микросервисов. Но и здесь у нас есть свое решение.

Решение проблемы — автоматический DevSecOps. И это не шутка. В рамках нашего решения разработан комплекс средств для генерации настроек безопасности, разворачивания, ведения версионности, переменных окружения и прочего. Поскольку в нашей компании производятся разработки примерно в 150 стеках, их настройка уже давно ведется с использованием генерации кода. Новые стеки в основном настраиваются системными аналитиками при минимальном привлечении DevSecOps. Ну, и наши DevSecOps традиционно разрабатывают шаблоны, на основании которых ведутся остальные настройки.

Четвертое заблуждение — микросервисные системы требуют огромных процессорных мощностей. Но это зависит от стиля работы с микросервисами. Если переписывать и дублировать функционал многократно, не следить за качеством написания кода и выбирать чересчур мелкие или, напротив, крупные блоки для создания микросервисов, а также не придерживаться принципа четко разграниченной задачи для каждого микросервиса, то «железа» потребуется много.

Решение задачи — аккуратное проектирование микросервисной архитектуры. В нашем случае микросервисы генерируются на основе шаблонов. Функции каждого микросервиса строго определены. Переиспользование микросервисов является обязательным. Это приводит к экономному расходованию процессора, что с блеском подтверждают тесты по быстродействию.

Ну и, конечно, здесь вступают в силу традиционные плюсы микросервисной архитектуры. Монолит можно запустить только на одном сервере — он лишен возможностей горизонтального масштабирования. Микросервисы можно запускать на наборе небольших по мощности серверов, что существенно снижает общую стоимость владения системой.

Резюме

В качестве заключения можно сказать, что правильно спроектированная микросервисная система, которая разрабатывается при плотном архитектурном надзоре, является вполне доступной роскошью как для разработки, так и для эксплуатации. При этом количество выгод, которые сулит переход на микросервисы, поистине огромно. Выгоды работы с микросервисными системами уже очевидны, поскольку все больше крупных игроков рынка встают на путь перехода к микросервисной архитектуре. А проблемы вполне преодолимы. Б.О

реклама