Применение микросервисной архитектуры в образовательном процессе

Родичкин П.А.[1], Соловьев В.М.[2]
[1]*MrNyanKat@yandex.ru*, [2]*SVM@sgu.ru*,
Саратовский государственный университет имени Н.Г. Чернышевского

Аннотация. Несмотря на то, что само понятие «микросервисы» появилось примерно десять лет назад, должное внимание это направление получило сравнительно недавно. В данный момент эта область IT активно развивается, расширяясь за пределы только лишь разработки программного обеспечения.

Ключевые слова: микросервисы, микросервисная архитектура, разработка, монолит, образование.

Найти упоминания о микросервисах можно примерно с 2000х, но само появление концепции принято относить к 2012-му году, когда в Кракове проходила конференция 33d Degree, на которой с докладом «Micro services – Java, the Unix Way» выступил James Lewis. С этого момента микросервисы обретают все большую популярность в сфере информационных технологий, постоянно развиваясь и совершенствуясь.

Классический подход к программированию и его проблемы

Что из себя представляет микросервисная архитектура? Чтобы ответить на этот вопрос, следует обратить внимание на классический подход «Монолит». Это подход к программированию, в рамках которого созданное программное обеспечение представляет собой единый (монолитный) программный продукт, не его отдельные компоненты.

Этот подход обладает рядом фундаментальных недостатков, основным из которых является экспоненциальный рост количества возникающих проблем, по мере увеличения объёма разрабатываемого программного продукта. Имеет смысл обратить внимание на самые основные из них:

а) Высокие трудозатраты на создание современного программного продукта со сложной бизнес-логикой, а отсюда и высокая стоимость. Поэтому современный сложный программный продукт в настоящее время создаёт коллектив разработчиков (программистов), где каждый решает свою отдельную задачу бизнес-логики. Кроме того, возникают сложности объединения (стыковки) этих задач в единый программный продукт.

б) Монолитный громоздкий код становится все труднее обслуживать – выражается это в том, что чем больше объем программного продукта, тем труднее его тестировать, а также все сложнее становится исправлять возникающие ошибки в работе продукта.

в) Ограниченность в выборе языков и средств программирования. Единожды выбранный для монолита язык остаётся таковым до конца разработки. А ведь нередка ситуация, когда в процессе поддержки уже выпущенного продукта вдруг выясняется, что столь необходимый в данный момент модуль в выбранной реализации просто отсутствует.

г) Запутанная структура – если поначалу проект имеет простую и понятную структуру, то по мере его роста понять, где какой модуль находится и за что отвечает становиться весьма проблематично.

д) Сильная зависимость одного модуля от другого - это ведёт, во-первых, к низкой отказоустойчивости. Если один из модулей прекращает функционировать, то останавливается и все приложение. Сюда же относится проблема обновления продукта – во-первых, это сложно по причине нагромождения кода, а во-вторых, ведёт к остановке всего приложения на неопределённый срок.

Микросервисный (контейнерный) подход к программированию

Микросервисы решают многие из вышеназванных проблем – проблемы ограничиваются рамками модуля, в котором они возникли (чем-то напоминает архитектуру Linux). Это же упрощает обслуживание кода и разработку его новых частей, а также не нарушает работу «остальной» части программы.

Помимо этого, возможность выбрать любую реализацию. Можно создать сервисы (контейнеры) на абсолютно разных языках программирования и все это будет работать, достаточно лишь правильно прописать их взаимодействие.

Но, с другой стороны, следует очень чётко понимать, когда и в какой момент перевести проект на микросервисную архитектуру. Это складывается из того факта, что создание самих микросервисов тоже весьма трудоёмкий и затратный процесс, а также он требует высокой квалификации специалистов. В настоящее время эта проблема решается использованием микросервисов с открытым исходным кодом (open source). Таких микросервисов по разным литературным источникам насчитывается более 700.

В связи с этим, вполне здраво выглядит следующая последовательность действий – после фазы планирования реализуется минимально жизнеспособный монолит, а потом, по мере развития, постепенно переписывается с использованием микросервисов. Так поступают большие IT компании при обновлении своих программных продуктов.

Говоря об остальных недостатках микросервисной архитектуры, стоит обозначить тот факт, что наибольшим из них выступает уже упомянутая выше коммуникация между частями разрабатываемой программы. Это необходимо тщательно планировать и ещё более тщательно реализовывать, иначе в противном случае список возникающих проблем и ошибок с лёгкостью перекроет все названные ранее плюсы. К другим недостаткам можно отнести тот факт, что микросервисная архитектура может превысить по объёму

изначальный монолит, что приведёт к дополнительным затратам на серверное оборудование, а также к сложностям мигрирования и последующего обслуживания.

Микросервисы в образовании

В связи с тем, что на данный момент микросервисы являются перспективной областью информационных технологий, логично предположить, что помимо разработки непосредственно коммерческих продуктов найти применение им можно и в областях, связанных с образованием.

В первую очередь имеет смысл поговорить об организации процессов обучения. Эта тема совсем недалеко отстоит от процесса разработки, о котором шла речь до этого. Связано это с тем, что довольно часто учебному заведению требуется создать некую специализированную систему для собственных нужд.

Показательным будет некий сайт университета или школы. Все дело в том, что учебные планы постоянно меняются, образовательная программа регулярно преобразовывается, а сайт должен не только отображать актуальную информацию, но и, зачастую, поддерживать другой функционал разной сложности, к примеру, систему обработки и проверки задач учеников. А ведь и она должна регулярно обновляться, ведь малейшие изменения в структуре языка могут привести к неправильной работе контестера.

Объединяя это с предыдущим пунктом весьма, логичным будет выбор держать вышеописанный сайт на микросервисной архитектуре. В таком случае, обновить что-то, а то и, к примеру, добавить абсолютно новый модуль для только что созданного факультета будет, сравнительно с монолитом, не сложно.

С другой стороны, очевидно, имеется потребность в квалифицированных специалистах в области микросервисов. В связи с этим, было бы вполне логичным шагом начать их подготовку если не в школах\спецкурсах, то, как минимум, в университетах, чтобы к старшим курсам они уже имели представление и базовые знания в данной области, что так же упростит дальнейшее трудоустройство.

Пример реализации

Для понимания того, что из себя представляют микросервисы, хотелось бы обратиться к следующему примеру – он представляет собой систему, которая способная принимать в себя данные с помощью POST-запроса, и искать нужные записи посредством GET.

В рамках тестовой задачи предлагается хранить данные о студентах вуза. В качестве параметров передаётся имя и номер группы, в качестве первичного ключа выступает ID.

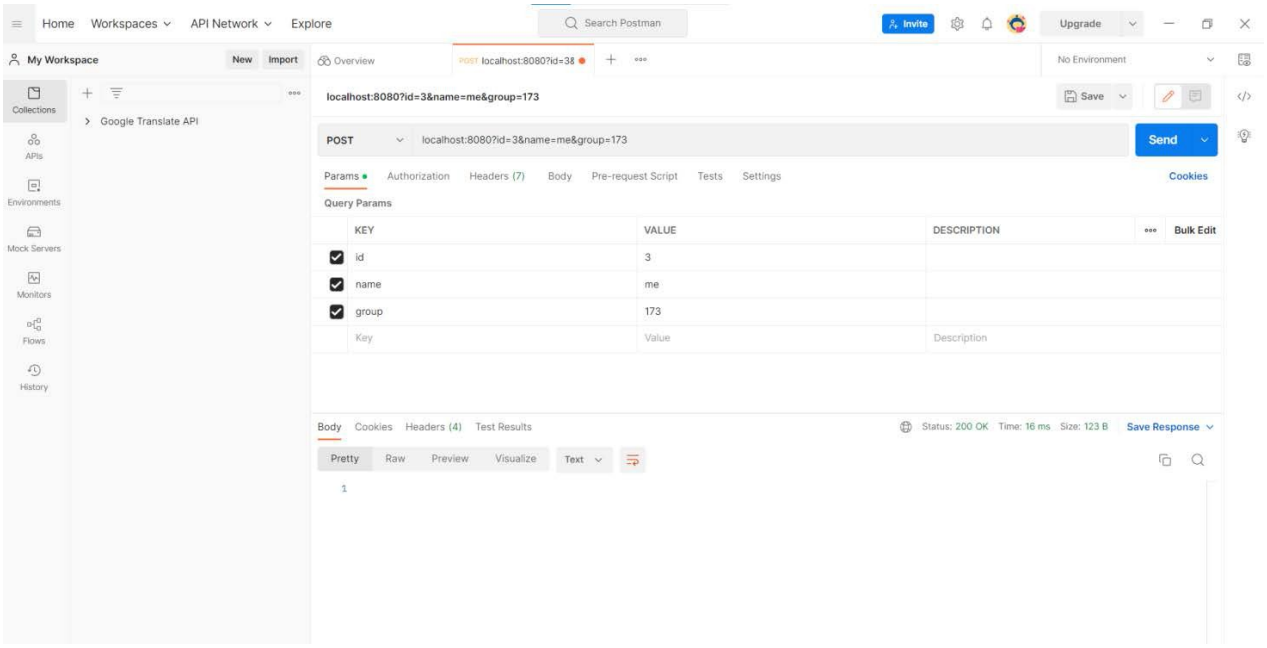

Рис. 1 Создание записи

На приведённом скриншоте можно увидеть создание записи, в качестве интерфейса был выбран Postman в связи с его простотой и удобством.

Далее можно получить только что выбранную запись, указав её ID:

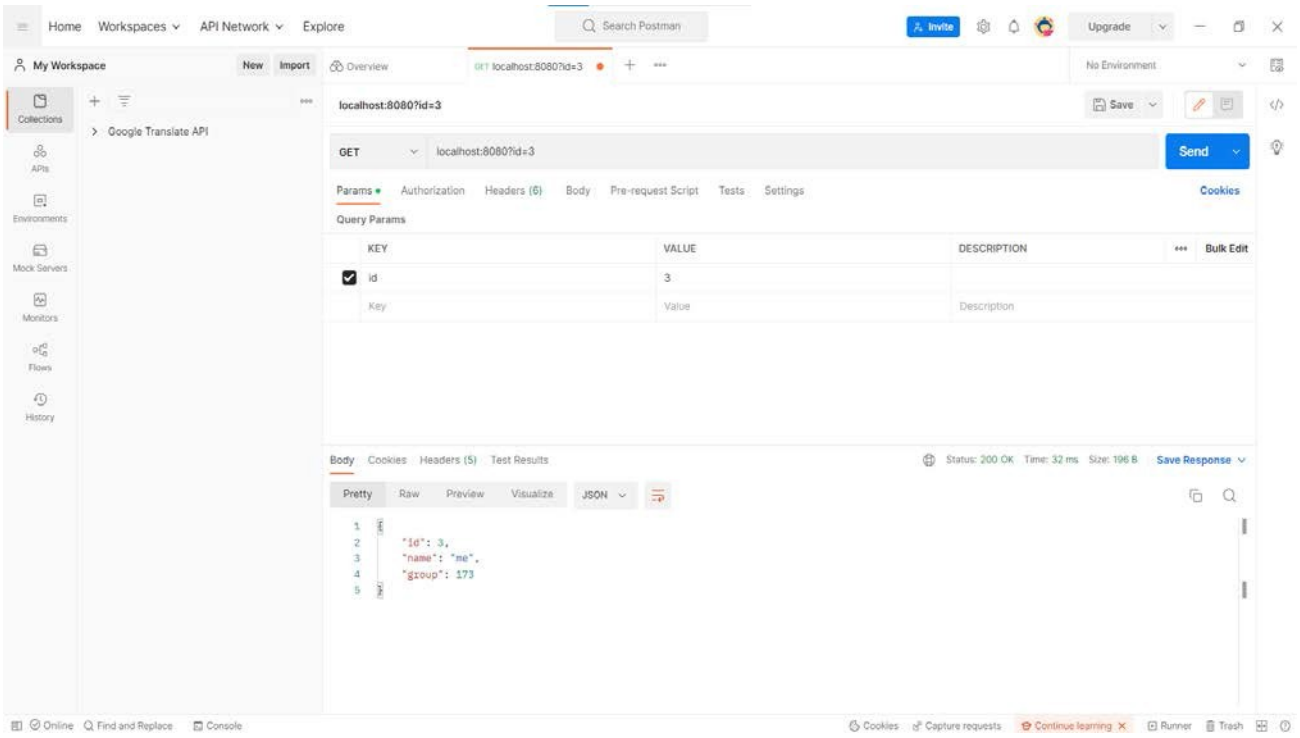

Рис. 2 Получение выбранной записи

Как можно заметить, все функционирует стабильно.

Данная разработка крайне проста и создана в первую очередь для демонстрации, но вместе с тем отвечает всем принципам микросервисов – сама она написана на Java Spring Boot, но дальнейшее взаимодействие можно настроить и с другими языками.

Другим примером может служить то, что на данный момент для хранения данных используется банальный массив, но заменить его на полноценную базу данных, например, от Oracle, тоже не составит особых проблем.

И это касается всех аспектов данной разработки – все они могут быть легко улучшены и переписаны под конкретные нужды, а так же весьма просто будет добавить нечто новое. А ведь это именно те свойства, которыми и должен обладать микросервис.

При изучении вопроса микросервисов были рассмотрены аспекты данного понятия, такие как положительные и отрицательные стороны, область применения, а так же был проведён сравнительный анализ микросервисной и монолитной архитектуры во время разработки приложения. Помимо этого была определена немаловажная потенциальная роль микросервисов в учебном процессе.

Все вышесказанное приводит к понимаю того факта, что микросервисная архитектура может применяться в том числе и в областях, сравнительно далёких от IT. Совместно с тем фактом, что эта область информационных технологий в данный момент весьма активно развивается, это создаёт почву для плодотворного сотрудничества – как и поддержку образования новыми современными технологиями, так и подготовку специалистов, имеющие нужные навыки для дальнейшего развития и применения микросервисной архитектуры.

Список литературы

[1] Микросервисная архитектура, ее паттерны проектирования и особенности [Электронный ресурс] URL: https://habr.com/ru/company/serverspace/blog/692916/ (дата обращения 12.10.2022).
[2] *Ньюмен Сэм* Создание микросервисов / С. Ньюмен. СПб.: Питер. 2016. – 304 с.
[3] *Джон Карнелл, Иллари Уайлупо Санчес* Микросервисы Spring в действии / пер. с англ. А. Н. Киселева. М.: ДМК Пресс. 2022 – 490 с.
[4] *Кочер П.С.* Микросервисы и контейнеры Docker / пер. с анг. А. Н. Киселева. М.: ДМК Пресс. 2019 – 240 с.: ил.
[5] «Микросервисы (Microservices)» [Электронный ресурс] URL: https://habr.com/ru/post/249183/ (дата обращения 29.09.2012)