МИКРОСЕРВИСЫ

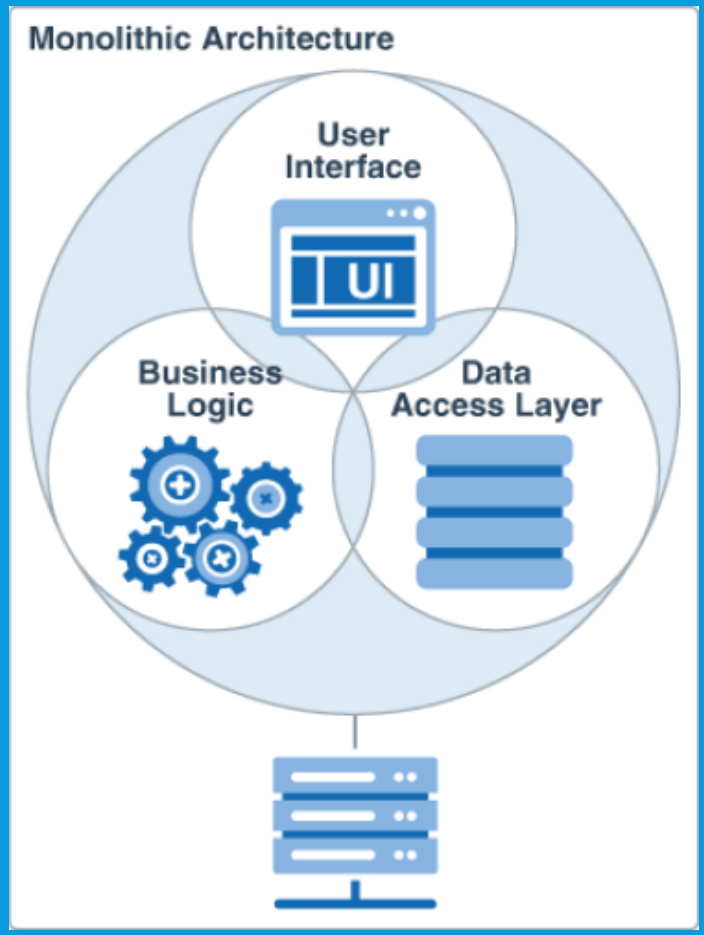
Гурулев Дмитрий

ЧТО ТАКОЕ МОНОЛИТНАЯ АРХИТЕКТУРА?

Монолитное приложение (монолит) представляет собой приложение, доставляемое через единое развертывание.

Давайте представим классический интернет-магазин. Стандартные модули: UI, бизнес-логика и дата-слой. Возможны способы взаимодействия с сервисом: API REST и веб-интерфейс. При построении монолита все эти вещи будут управляться внутри одного и того же модуля.

На картинке приведен пример, как все части находятся в одном и том же модуле развертывания.

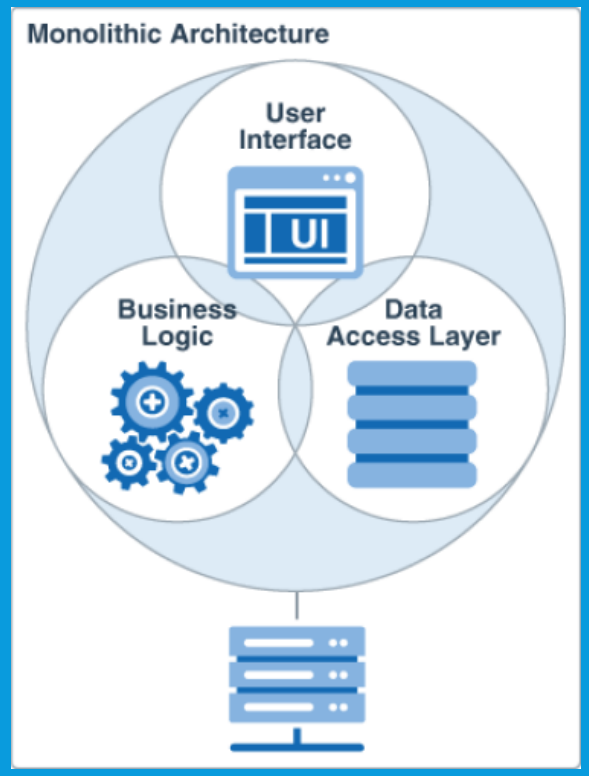


ПРОБЛЕМАТИКА БОЛЬШИХ МОНОЛИТНЫХ СИСТЕМ

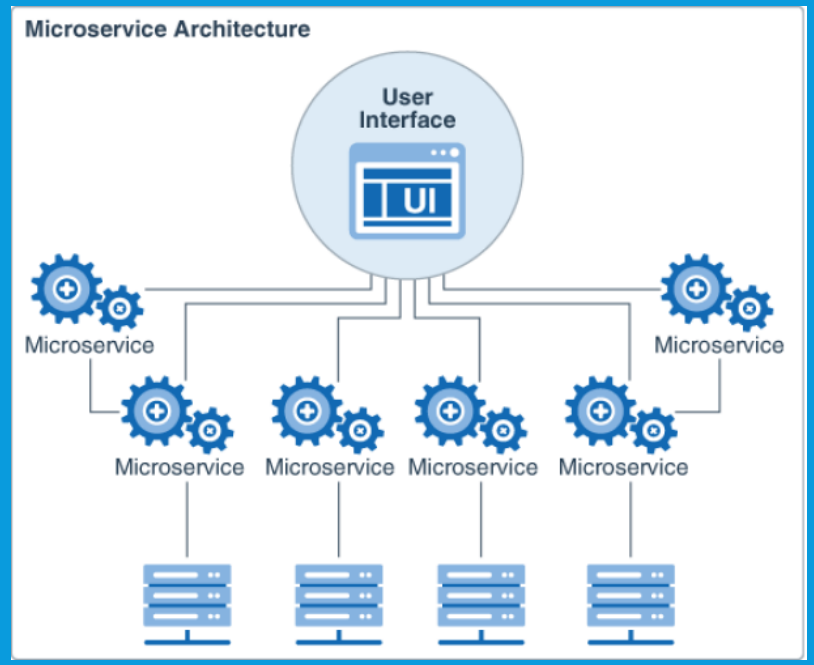
- Плохое горизонтальное масштабирование
- Плохая отказоустойчивость
- Сложность внедрения новых технологий
- Сложность рефакторинга legacy-кода

СУТЬ МИКРОСЕРВИСНОГО ПОДХОДА

Монолит



Микросервисы



Микросервисы — это небольшие, автономные, совместно работающие сервисы.

ПРИНЦИПЫ МИКРОСЕРВИСОВ

ПАРА ПРИМЕРОВ УСПЕШНОГО ПРИМЕНЕНИЯ МИКРОСЕРВИСНОЙ АРХИТЕКТУРЫ

Стремление к созданию небольших команд, владеющих полным жизненным циклом своих сервисов, и стало основной причиной того, что в Amazon была разработана платформа Amazon Web Services.

Этот пример был взят на вооружение компанией Netflix и с самого начала определил формирование ее структуры вокруг небольших независимых команд, образуемых с прицелом на то, что создаваемые ими сервисы также будут независимы друг от друга.

ПЛЮСЫ И МИНУСЫ

Плюсы

- Горизонтальное масштабирование только нужных частей
- Отказоустойчивость
- Масштабирование команд
- Гибкость стека
- Переиспользование

Минусы

- Дополнительная сложность в тестировании и развертывании
- Высокая начальная стоимость
- Бывает трудно определить границу между сервисами

ВАМ НЕ НУЖНЫ МИКРОСЕРВИСЫ, ЕСЛИ

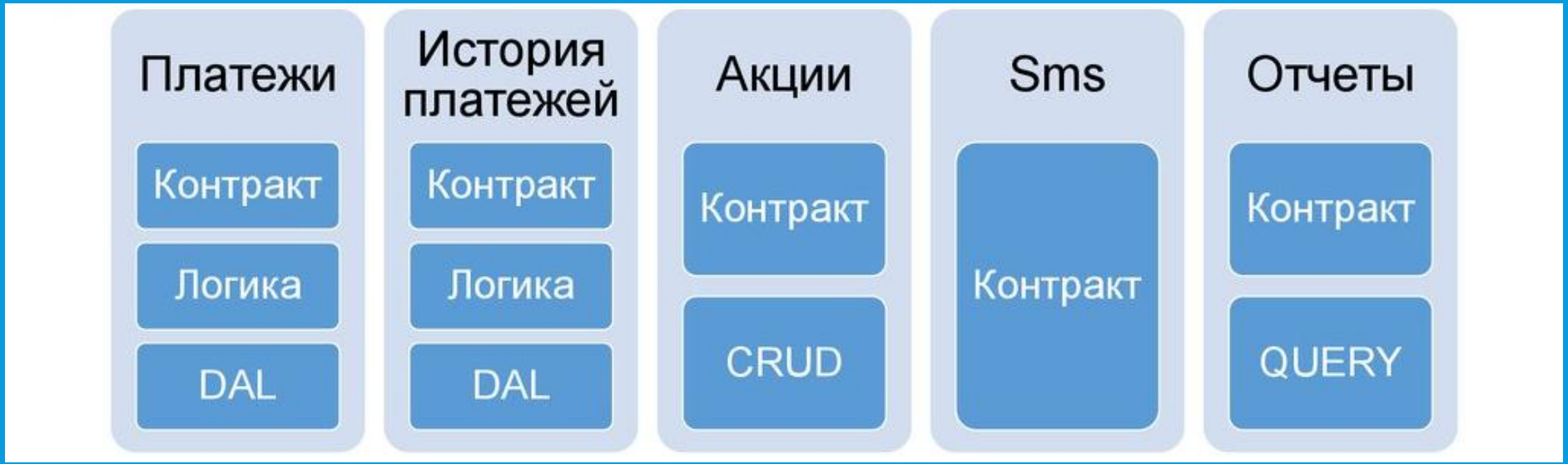
- Вы делаете стартап (MVP)
- Нет или не предполагается рост нагрузки
- Нет плана, как порезать сервисы на слабосвязанные

ВАМ НУЖНЫ МИКРОСЕРВИСЫ, ЕСЛИ

- Предполагается высокая нагрузка
- Система растет
- Команда растет
- Нужна отказоустойчивость
- Необходимо сократить время между релизами (TTM)

РАЗРЕЗ НА СЕРВИСЫ

Главное правило: не разрезайте по слоям, разрезайте по бизнес-контекстам



DAL – слой доступа к данным.

CRUD – акроним, обозначающий четыре базовые функции, используемые при работе с базами данных.

QUERY – запросы к базе данных.

РАЗМЕР СЕРВИСА

Для того, чтобы при разрезе на сервисы получились именно микросервисы, а не еще один монолит, размер сервиса выбирается:

- на основе бизнес-контекста
- на основе сетевых запросов (чем меньше, тем лучше)
- на основе транзакций (транзакция внутри одного сервиса)

VIRTUAL MACHINE ИЛИ CONTAINER

VITRUAL MACHINE

Преимущества

- Образ ВМ инкапсулирует стек технологий.
- Экземпляры сервиса изолированы.
- Используется зрелая облачная инфраструктура.

Недостатки

- Менее эффективно используются ресурсы.
- Развертывание протекает довольно медленно.
- Требуются дополнительные расходы на системное администрирование.

VIRTUAL MACHINE ИЛИ CONTAINER

CONTAINER

Преимущества

- Инкапсуляция стека технологий, благодаря которой API для управления сервисом превращается в API контейнера.
- Экземпляры сервиса изолированы.
- Ресурсы экземпляров сервиса ограничены.

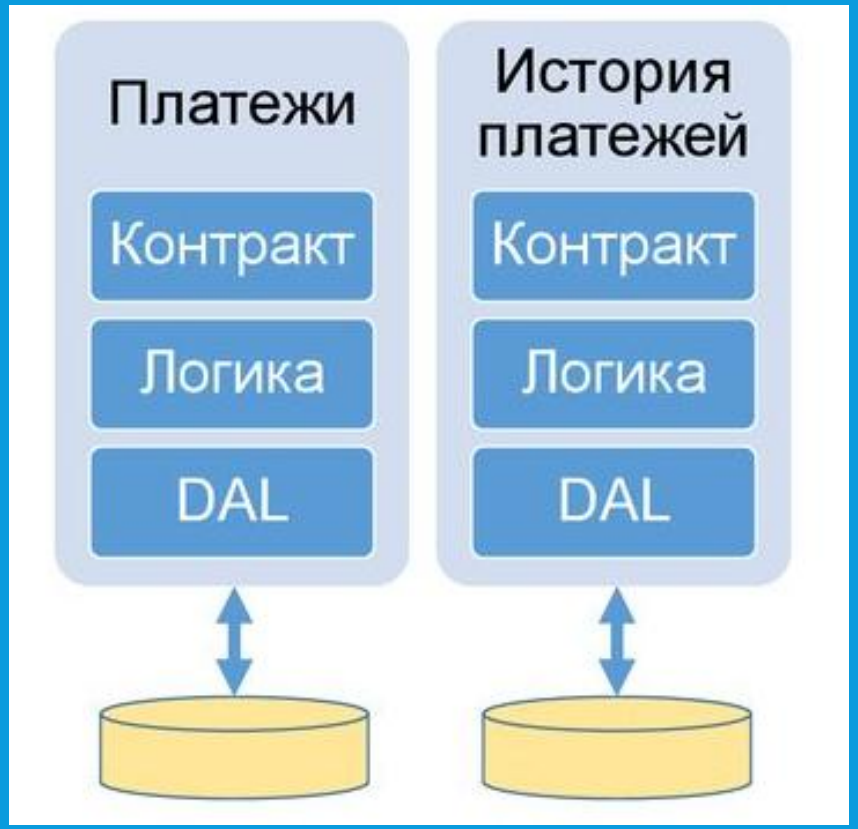
Недостатки

- Существенный недостаток контейнеров состоит в том, что вы должны постоянно заниматься управлением образами контейнеров и обновлением операционной системы вместе со средой выполнения.

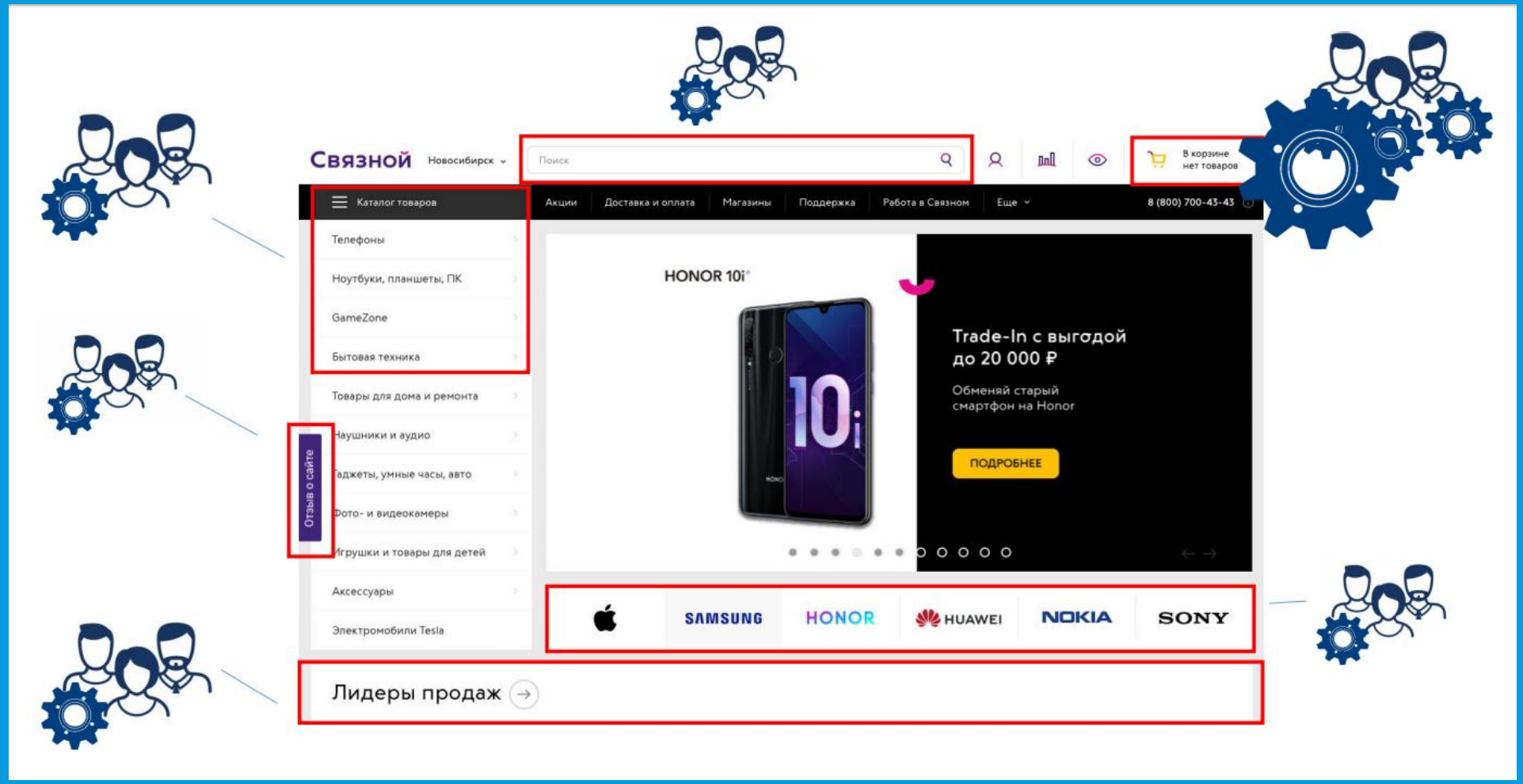
БАЗА ДАННЫХ

Один сервис – одна база

Выбор типа базы данных зависит от задачи

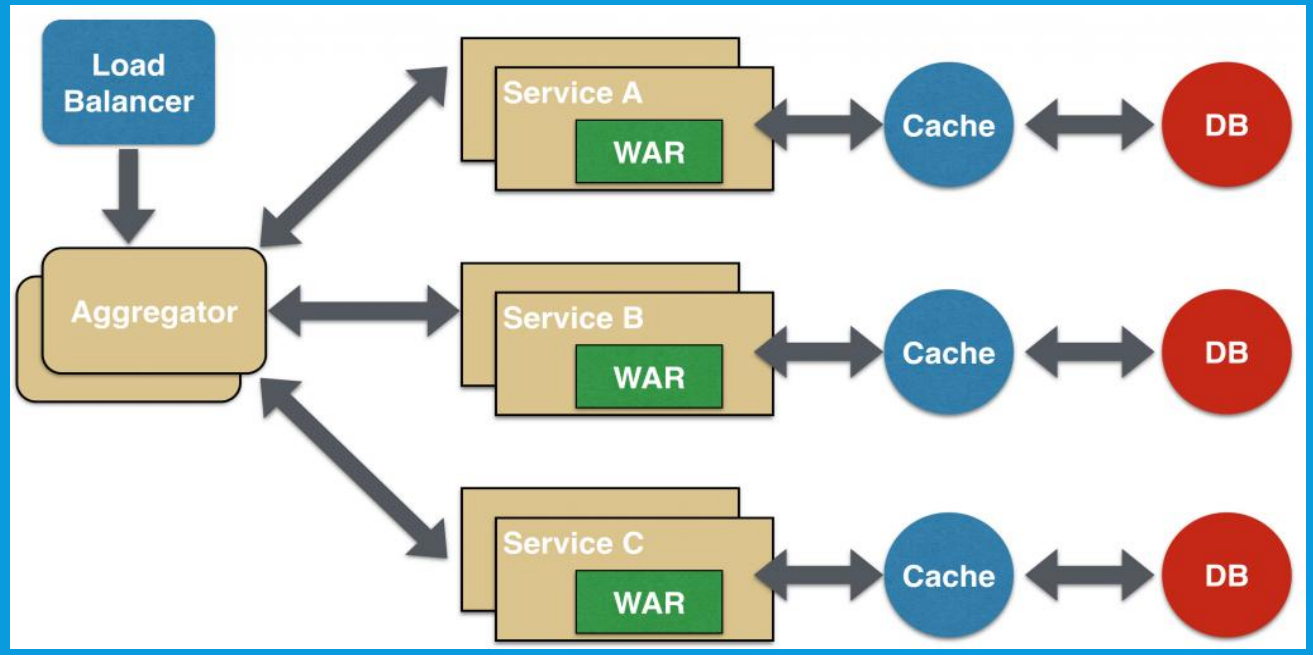


ВИЗУАЛЬНЫЙ ПРИМЕР РАЗБИЕНИЯ НА МИКРОСЕРВИСЫ



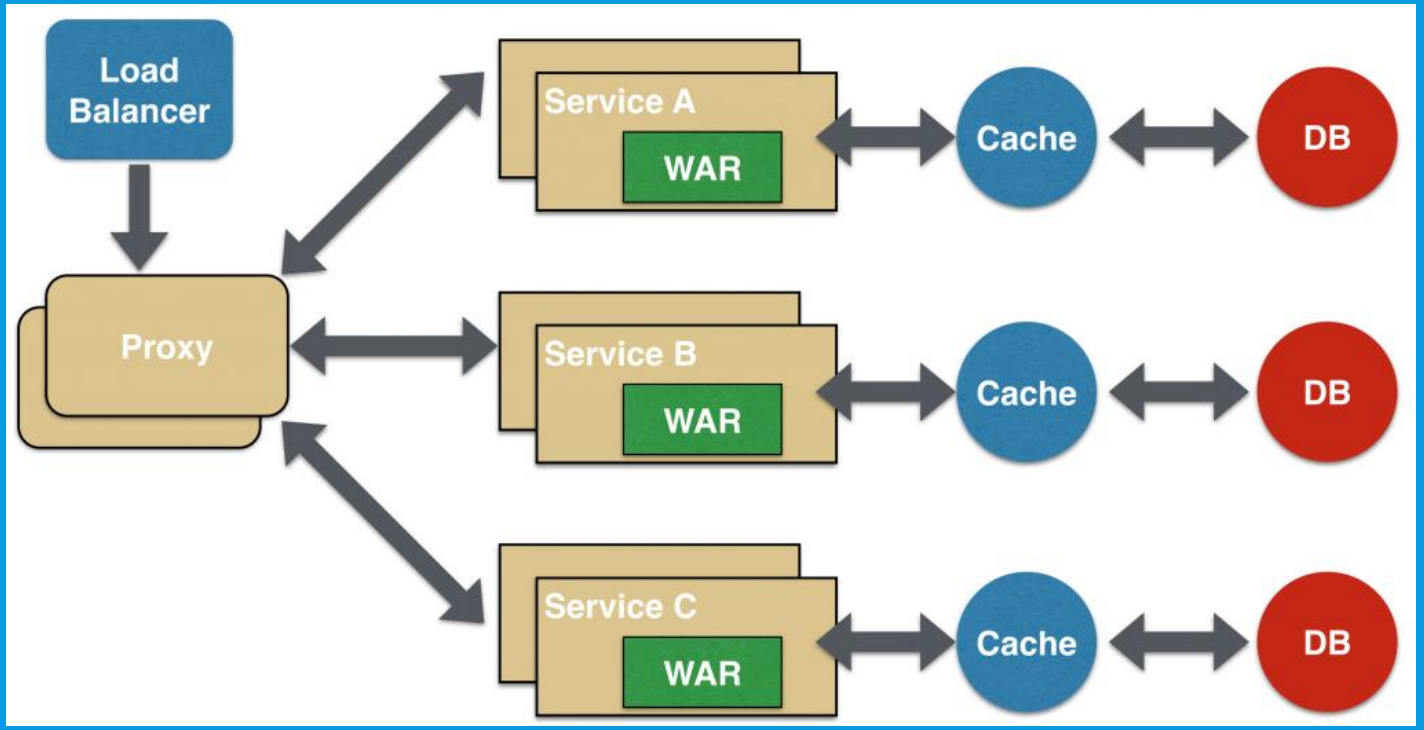
ПАТТЕРН АГРЕГАТОР (AGGREGATOR)

В простейшей форме агрегатор представляет собой обычную веб-страницу, вызывающую множество сервисов для реализации функционала, требуемого в приложении. Агрегатор может использоваться и в тех случаях, когда не требуется ничего отображать, а нужен лишь более высокоуровневый составной микросервис, который могут потреблять другие сервисы.



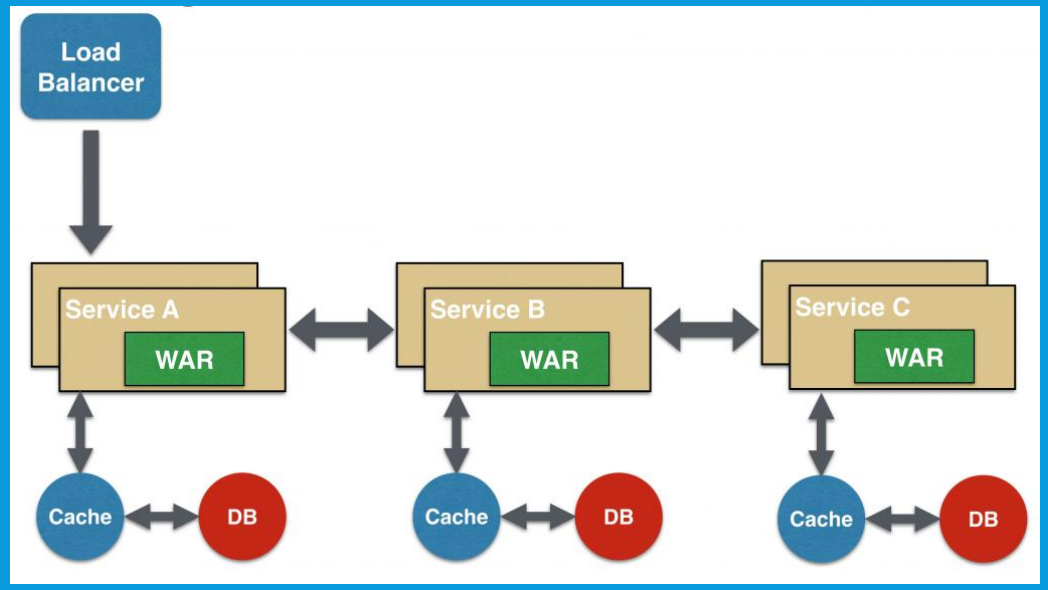
ПАТТЕРН ПОСРЕДНИК (PROXY)

Паттерн «посредник» при работе с микросервисами – это вариант агрегатора. В таком случае агрегация должна происходить на клиенте, но в зависимости от бизнес-требований при этом может вызываться дополнительный микросервис.



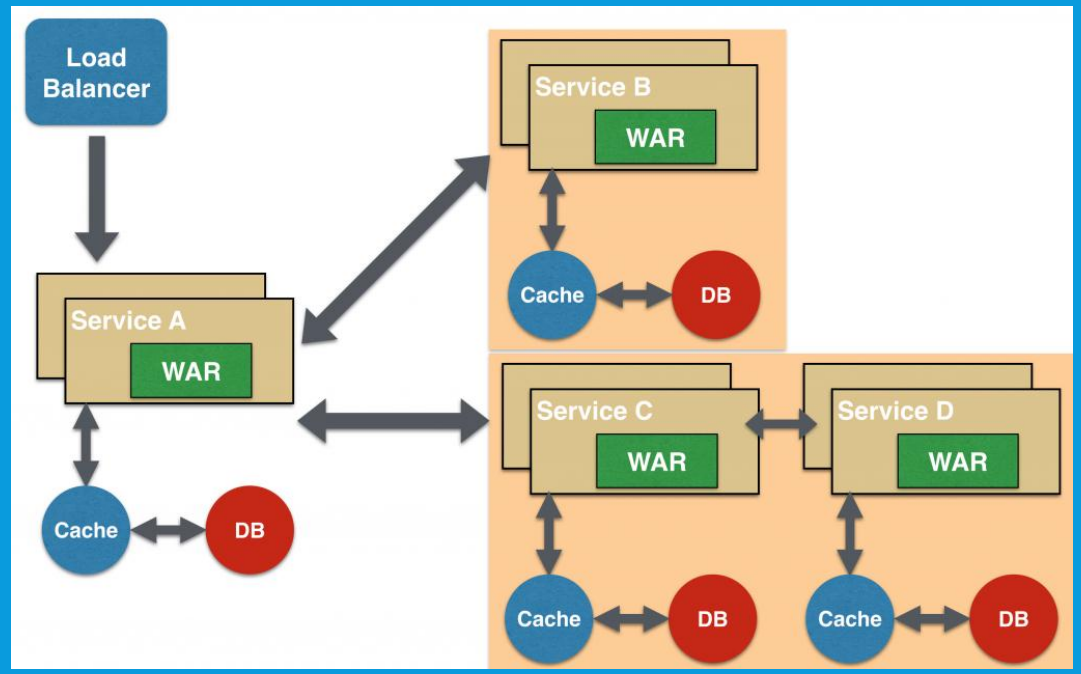
ПАТТЕРН ПРОЕКТИРОВАНИЯ «ЦЕПОЧКА» (CHAINED)

Микросервисный паттерн проектирования «Цепочка» выдает единый консолидированный ответ на запрос. В данном случае сервис A получает запрос от клиента, связывается с сервисом B, который, в свою очередь, может связаться с сервисом C. Все эти сервисы, скорее всего, будут обмениваться синхронными сообщениями «запрос/отклик» по протоколу HTTP. Нельзя делать цепочку слишком длинной!



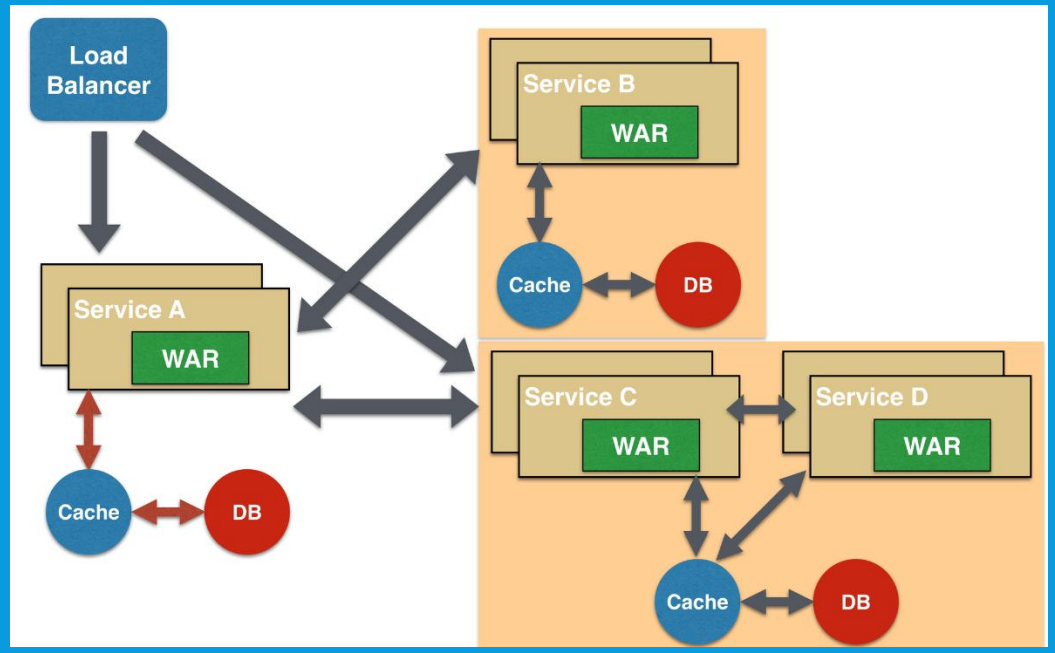
ПАТТЕРН ПРОЕКТИРОВАНИЯ «ВЕТКА» (BRANCH)

Микросервисный паттерн проектирования «Ветка» расширяет паттерн «Агрегатор» и обеспечивает одновременную обработку откликов от двух цепочек микросервисов, которые могут быть взаимоисключающими. Этот паттерн также может применяться для вызова различных цепочек, либо одной и той же цепочки.



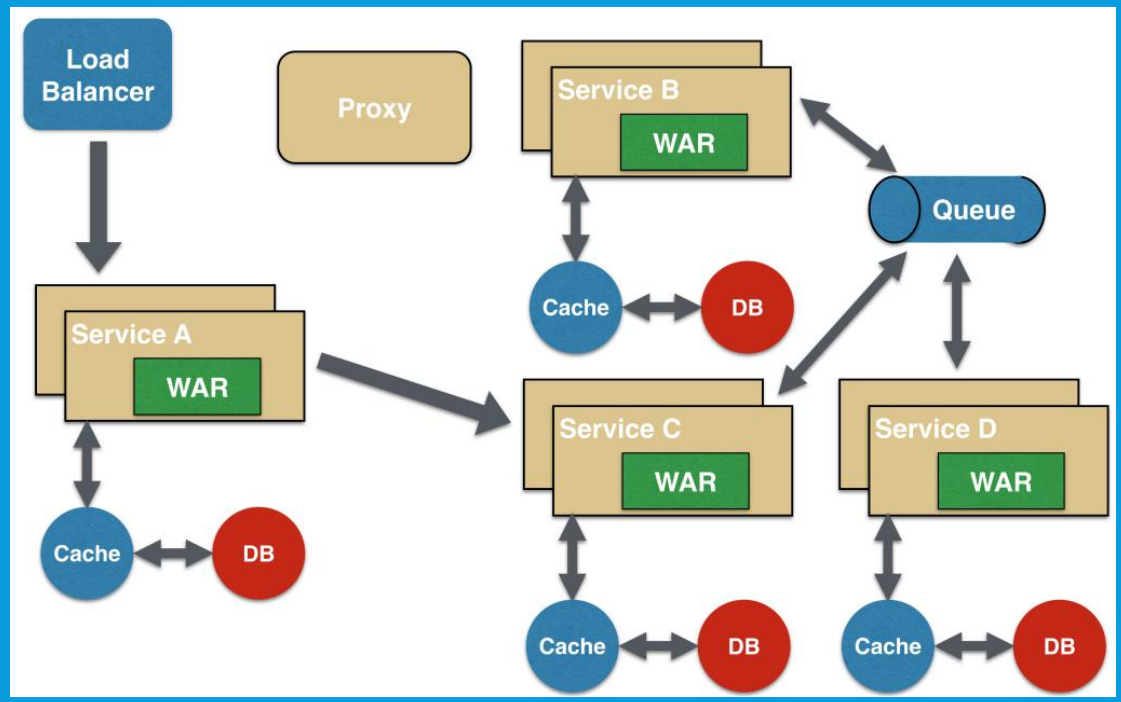
ПАТТЕРН «РАЗДЕЛЯЕМЫЕ ДАННЫЕ» (SHARED DATA)

При этом паттерне несколько микросервисов могут работать о цепочке и совместно использовать хранилища кэша и базы данных. Это целесообразно лишь в случае, если между двумя сервисами существует сильная связь. Некоторые могут усматривать в этом антипаттерн, но в некоторых бизнес-ситуациях такой шаблон действительно уместен.



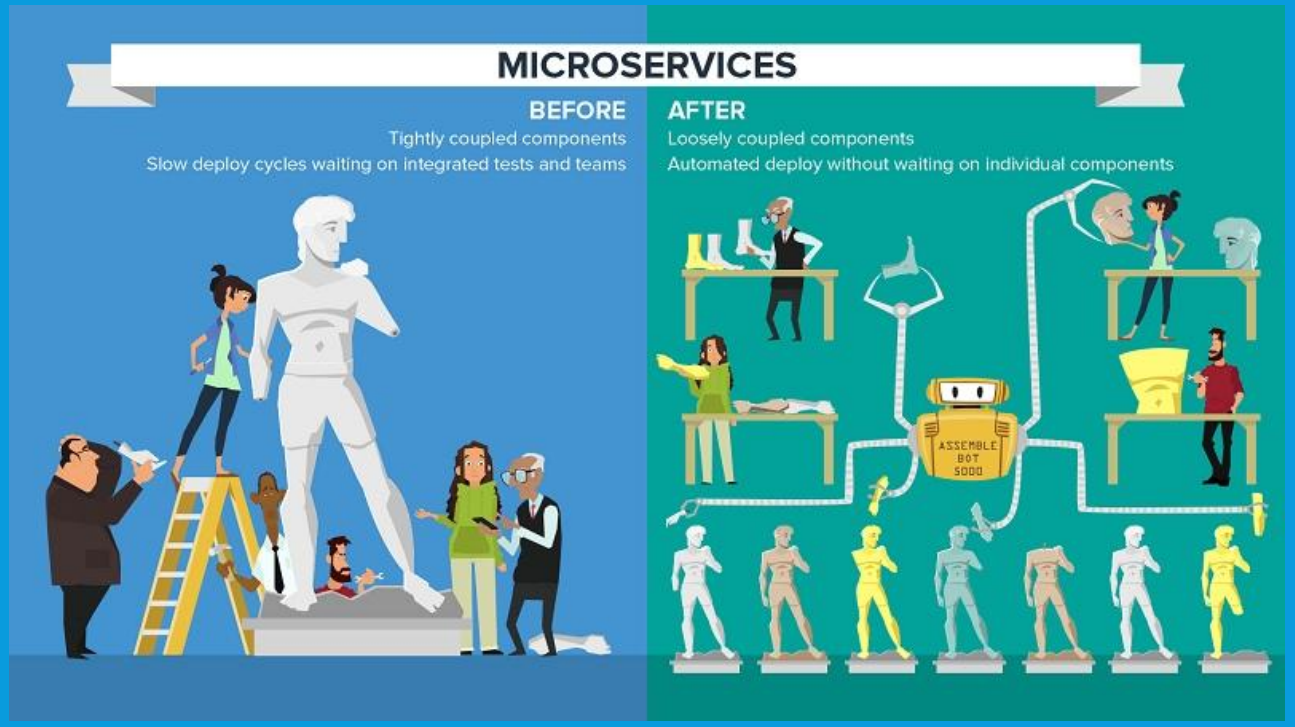
ПАТТЕРН «АСИНХРОННЫЕ СООБЩЕНИЯ» (ASYNCHRONOUS MESSAGING)

При всей распространенности и понятности паттерна REST, у него есть важное ограничение, а именно: он синхронный и, следовательно, блокирующий. Поэтому в некоторых микросервисных архитектурах могут использоваться очереди сообщений, а не модель REST запрос/отклик.



ВЫВОДЫ

Грамотное разбиение на микросервисы даёт преимущества в виде увеличения скорости разработки. Но для этого нужны специалисты, способные не только писать код, но и развёртывать то, что написано.



СПИСОК ЛИТЕРАТУРЫ

1. Кристиан Хорсдал Микросервисы на платформе. NET / Хорсдал Кристиан. - М.: Питер, 2018. - **442** с.
2. Сэм, Ньюмен Создание микросервисов. Руководство / Ньюмен Сэм. - М.: Питер, 2016. - **145** с.
3. Ньюмен, Сэм Создание микросервисов / Сэм Ньюмен. - М.: Питер, 2015. - **497** с.