**Uber**

Как и многие другие стартапы, Uber начал свой путь с монолитной архитектуры, построенной для использования в одном городе. Такая архитектура решала их основные проблемы бизнеса на тот момент, которые включали подключения водителей с клиентами, выставление счетов и платежей. Это было разумно тогда, чтобы вся бизнес-логика Uber была в одном месте. С тем как они быстро расширялись в большем количестве городов, это быстро изменилось. Всего за последние 4 года Uber вырос в 38 раз.

С тем как модели ядра домена росли, вводились новые функции, компоненты стали тесно связаны, и обеспечение инкапсуляции сделало разделение задач трудным. Непрерывная интеграция превратилась в обязательство, поскольку развертывание кодовой базы означало развертывание всей системы сразу. Добавление новых функций, исправление ошибок, и решения технической проблем все в одном месте стало крайне сложно.

Они решили последовать примеру других быстроразвивающихся компаний, таких как Amazon, Netflix, SoundCloud, Twitter и другие, и разбить монолит на несколько баз кода для формирования сервисо-ориентированной архитектуры (SOA).

Се́рвис-ориенти́рованная архитекту́ра (SOA, [англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) service-oriented architecture) — [модульный](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D1%83%D0%BB%D1%8C%D0%BD%D0%BE%D1%81%D1%82%D1%8C_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) подход к разработке [программного обеспечения](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5), основанный на использовании [распределённых](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D1%81%D0%BF%D1%80%D0%B5%D0%B4%D0%B5%D0%BB%D1%91%D0%BD%D0%BD%D1%8B%D0%B5_%D0%B2%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F), [слабо связанных](https://ru.wikipedia.org/w/index.php?title=%D0%A1%D0%BB%D0%B0%D0%B1%D0%BE%D0%B5_%D1%81%D0%B2%D1%8F%D0%B7%D1%8B%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5&action=edit&redlink=1) ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) [loose coupling](https://en.wikipedia.org/wiki/loose_coupling)) заменяемых компонентов, оснащённых стандартизированными [интерфейсами](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F_%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B9) для взаимодействия по стандартизированным [протоколам](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%D0%BE%D0%B9_%D0%BF%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB). В частности, так как термин SOA может означать множество различных вещей, они приняли архитектуру microservice. Этот шаблон дизайна усиливает развитие малых сервисов, посвященных конкретным, хорошо инкапсулированным доменным зонам. Каждый сервис может быть написан на своем собственном языке или фремверке, и может иметь свою собственную базу данных или не иметь ее вовсе.

[Таким образом, системы, основанные на SOA, могут быть независимы от технологий разработки и\_платформ(таких как .NET и т.д).  К примеру, сервисы, написанные на](http://dic.academic.ru/dic.nsf/ruwiki/90411)[C#](http://dic.academic.ru/dic.nsf/ruwiki/1112), работающие на платформах .Net и сервисы на\_Java, работающие на платформах [Java EE](http://dic.academic.ru/dic.nsf/ruwiki/12124), могут быть с одинаковым успехом вызваны общим составным\_приложением. Приложения, работающие на одних платформах, могут вызывать сервисы, работающие надругих платформах, что облегчает повторное использование компонентов. (Средняя стоимость СОА-проекта оценивается AMR Research примерно в 500 тыс. долл.)

**ОБЗОР АРХИТЕКТУРЫ UBER:**

- диспетчерская часть по-большей части написан на node.js.

- Системы для работы с картами и оценки времени прибытия автомобиля на Python, C++ и Java. Для геопозиционирования, Uber использует S2 Geometry Library, которая способен разделить землю на клетки, и таким образом S2 может представлять каждый квадратный сантиметр с помощью Int64 идентификатора.

- Сервисы бизнес логики написаны на Python.

- Используется большое количество различных баз данных. Используется postgres, redis, Twemproxy, MySQL, Riak.

Как вы знаете, Uber все еще развивается, поэтому они не придерживаться одного вида структуры. Многие из архитектурных выборов Uber являются прямым следствием быстрого роста. Вот почему так много технологий было использованы на бэкэнде Uber, потому что их главная цель состоит в том, чтобы получить скорость развития настолько большой, насколько это возможно.

**Platform**

* Node.js
* Python
* Java
* Go
* Native applications on iOS and Android
* Microservices
* Redis
* Postgres
* MySQL
* [Riak](http://basho.com/)
* Twitter’s [Twemproxy](https://github.com/twitter/twemproxy) for Redis
* Google's [S2 Geometry Library](https://code.google.com/p/s2-geometry-library/)
* [ringpop](https://github.com/uber/ringpop-node) - consistent hash ring
* [TChannel](https://github.com/uber/tchannel) - network multiplexing and framing protocol for RPC
* Thrift