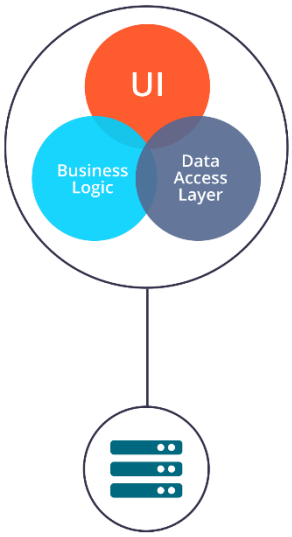
Защо микросървиси и защо Амазон ги използва ?

Започвам с един фундаментален въпрос: „Как започва всичко (с Амазон) ?”

Всичко започва от раждането на Джеф Безос... Само „няколко“ години по-късно се ражда монолитното приложение на амазон – онлайн магазин. Този магазин обаче се разраства с невиждана скорост.



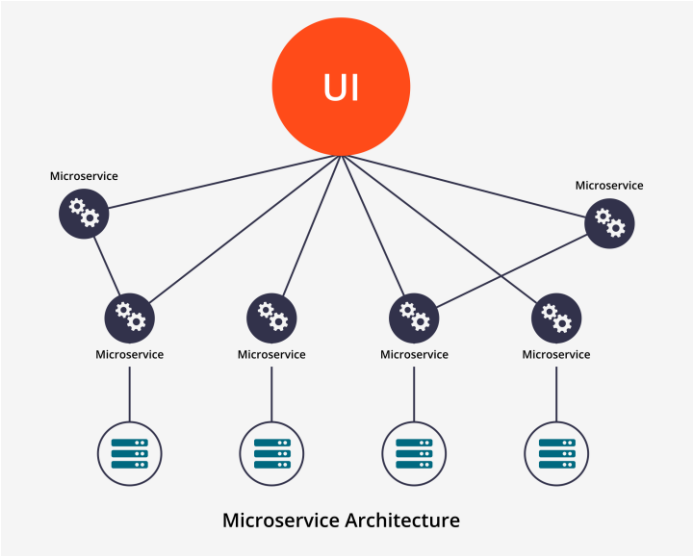
Monolithic Architecture

Добавят се все повече функционалности. С това се увеличава и броят програмисти, които го разработват и поддържат. Увеличава се и кода на това вече огромно приложение. Амазон обаче не спира да иска нови и нови функционалности, които разработчиците трябва да напишат, тестват, другите да го одобрят и то да влезе в употреба.

Тук идва един проблем. Колкото по-голямо става приложението, толкова по-трудно се пише нещо ново към него и още по-трудно то се вкарва в употреба. Този процес започва да отнема все повече и повече време, докато стига време от един месец за нова функционалност. Архитектите се усещат че нещо не е наред и не би трябвало да се разбработва толкова бавно. Тогава откриват, че на програмистите им е много трудно да се съчетаят с още 100 и дори повече колеги и да работят по един и същи проект с милиони редове код. На никой не му е изгодно да има такива забавяния. Какво следва тогава ?

Появата на микросървиси и предимствата им, които грабват Амазон...

Това е било началото на 21ви век. Тепърва света е започнал да научава за микросървисната архитектура. Какво е това ?



Microservice Architecture

Много просто. Разбиваме голямото монолитно приложение на няколко малки, които си комуникират. Така спестяваме лутането на програмистите в безкрайния код. Намаляме времето за тестване. Времето за стартиране.

Имаме възможност да скалираме само отделни (критични) компоненти от системата, вместо

цялото приложение. Позволява да подобрим устойчивостта на софтуера, както и да улесним и ускорим имплементирането и добавянето на нови функционалности.

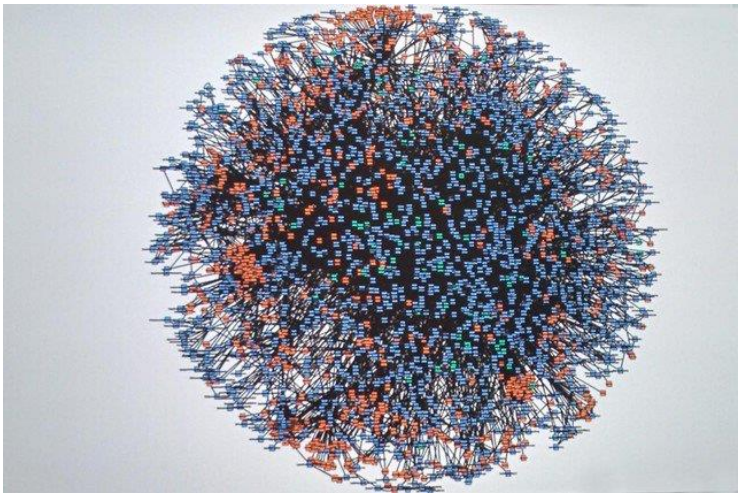
Кратки срещи на екипа, понеже той е малък и така е лесно всеки да знае какво правят другите. Има една поговорка – „Ако не можеш да нахраниш екипа с две големи пици, значи е твърде голям!“

Малките промени в монолитното приложение могат да „счупят“ цялата система и откриването на грешките отнема цяла вечност и могат да доведат до още много промени. Също ако трябва да се мигрира към по-висока версия на някой важен компонент, е много бавен и труден процес. При микросървисите ако един компонент се „счупи“, другите продължават да работят, макар и частично.

При използването на микросървиси, всеки сървис може да е на различен език или stack, в зависимост кое е най-подходящо за конкретната задача, която трябва да изпълнява сървиса.

Всичко това помага на Амазон да развива своя софтуер, възможно най-продуктивно.

Така изглежда архитектура на Амазон:



Всичко това звучи много логично и много лесно за имплементиране, но когато се разровим малко по-дълбоко, намираме някои минуси...

Continuous integration или преведено - постоянната интеграция(добавяне) на нови функционалности, може да бъде голямо предизвикателство.

Комуникацията между разработчиците в различните екипи, работещи по различните микросървиси е по-трудна от тази на един екип работещ по монолит. Когато приложението е монолит, всички от екипа знаят за всичко, но когато има много екипи, всеки екип разбира неговата част и не е много запознат с останалите. Тогава проблема се появява, когато единия

модул на приложение, поддържан от един екип, трябва да използва друг модул, поддържан от друг екип.

Интеграцията на нови разработчици по екипите се осъществява малко по-трудно когато екипите са разбити. Тогава на новият колега в екипа му е трудно да разбере цялостната идея и целия процес по разработката.

Ще заключа с отговора на въпроса: Кога е подходящо да използваме микросървисна архитектура?

Когато искаме нашето монолитно приложение да бъде лесно скалируемо, гъвкаво и да се поддържа лесно.

Ако има части от монолит, които трябва да се преизползват от други системи или приложения, е добра идея да използваме микросървиси. Така без да показваме всичко от монолита, показваме само частта, която трябва да се използва от другата система.

Ако приложението ни трябва да доставя нови функционалности много бързо.

Със сигурност не е добра идея да започнем да разработваме нашето приложение с микросървиси, само за да сме в крак с модата. Това отнема повече време и повече усилия и знания. Особено ако приложението ни е малко и няма какво да разбием.

По темата за микросървиси може да се говори много, но ви препоръчвам да се запознаете по-дълбоко с технологията и сами ще откриете защо да я използвате или не.

Източници:

Лични наблюдения за микросървисната архитектура.

https://thenewstack.io/led-amazon-microservices-architecture/

https://www.hys-enterprise.com/blog/why-and-how-netflix-amazon-and-uber-migrated-to-microservices-learn-from-their-experience/