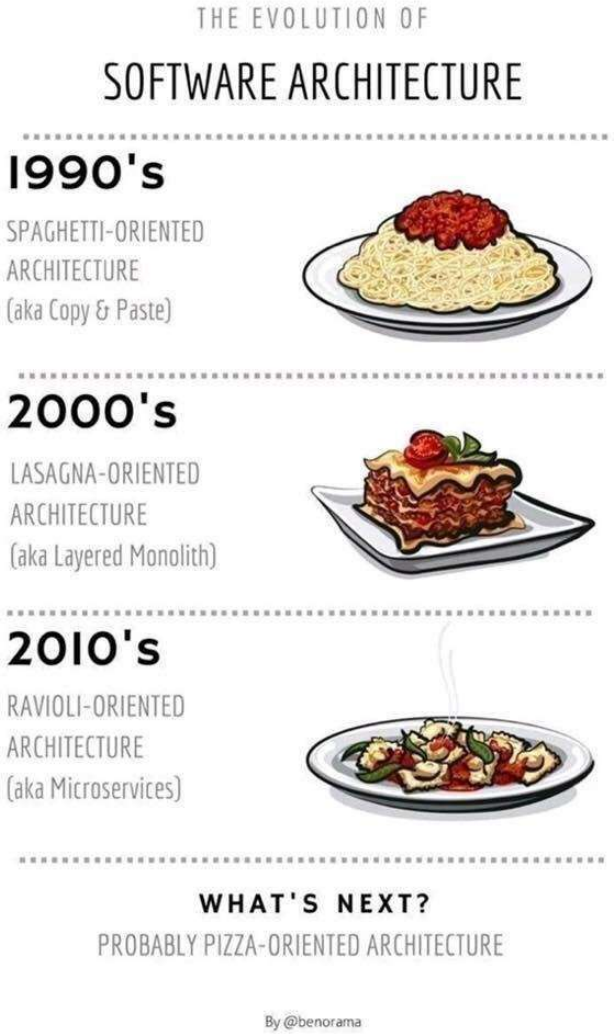
INFOSÜSTEEMIDE ARENDAMINE III - HAJUSRAKENDUSED

Loeng 8 – Mikroteenused

Tarvo Treier
Tarkvarateaduse instituut

21.10.2024

ARHITEKTUURIDE AJALUGU



Viide: https://media.licdn.com/media-proxy/ext?w=650&h=960&f=pj&hash=Pr1qPG2NUkaQyBLurB8OQ%2FCorWU%3D&ora=1%2CaFBCTXdkRmpGL2lvQUFBPQ%2CxAVta5g-0R6jnhodx1Ey9KGTqAGj6E5DQJHUA3L0CHH05IbfPWi6cZSOfbf390BAKi1SjQBhLua1EjOzEY6_f4nsdIkjjsO2I5H5aRUPbhU4hGUB5sE-Pg&shareType=image

JEFF BEZOS-I (CEO) KIRI/KORRALDUS AMAZONI TÖÖTAJATELE (~2002)

- Kõik meeskonnad avaldavad edaspidi oma andmed ja funktsionaalsuse teenusliideste kaudu
- Meeskonnad peavad nende liideste kaudu omavahel suhtlema
- Muud protsessidevahelise suhtluse vormid ei ole lubatud: otsene sidumine, teise meeskonna andmebaasi lugemine, ühismälu mudel, ühtegi tagaust ei ole. Ainus suhtlus, mida lubatakse, on võrguteenuste kaudu kasutatavate teenuseliidese väljakutsed
- See ei ole oluline, millist tehnoloogiat nad kasutavad
- Meeskond peab planeerima ja kujundama nõnda, et oleks võimalik välise maailma arendajatele liides avada. Eranditeta.

- **Igaüks, kes seda ei tee, vallandatakse. Aitäh; head päeva!**

TAL TECH

Viide: https://apievangelist.com/2012/01/12/the-secret-to-amazons-success-internal-apis/

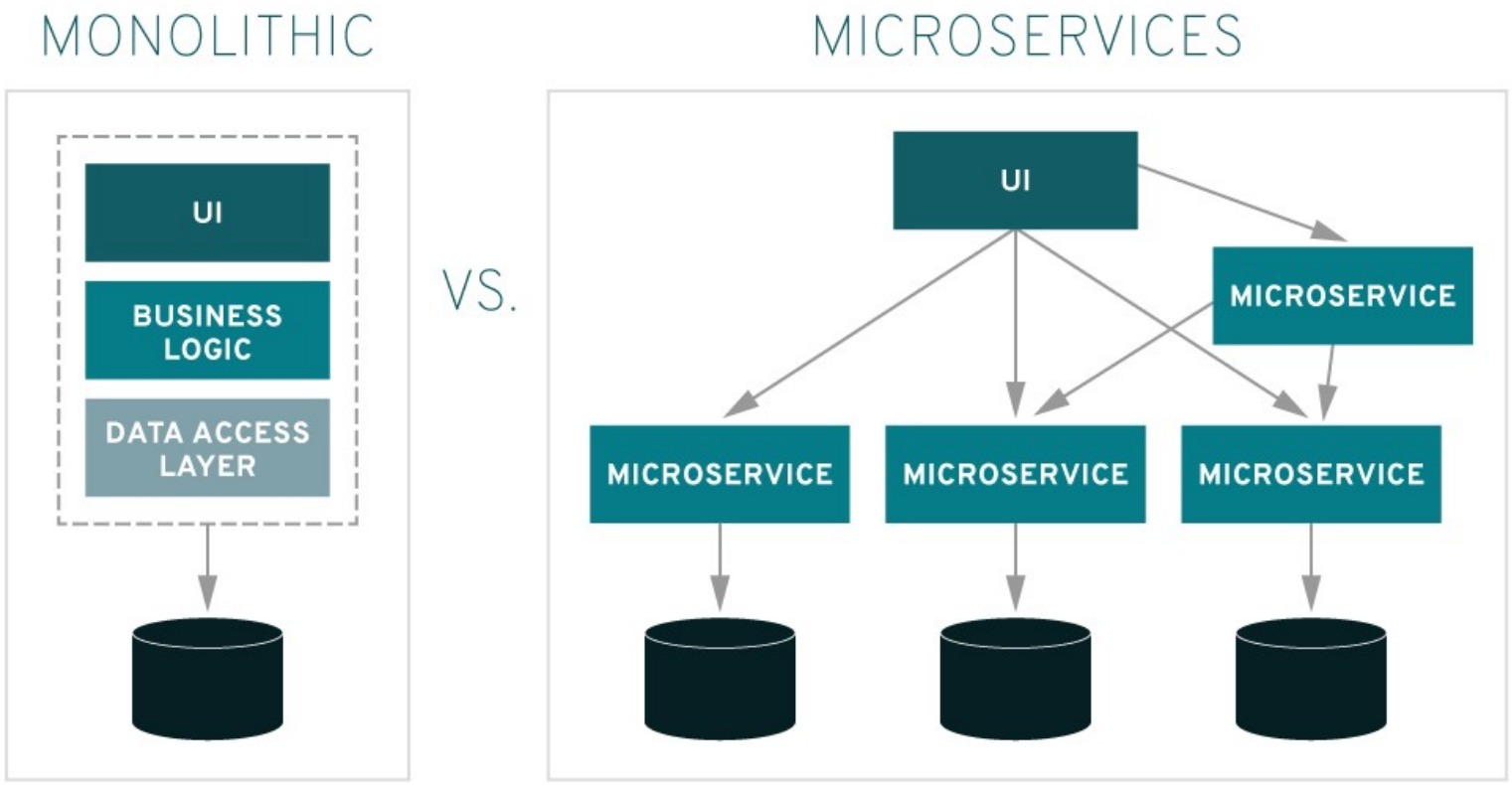
MIKROTEENUSED

- On arhitektuur/muster rakenduste ehitamiseks
- On omavahel nõrgalt seotud ja hajutatud (hajusrakendused) nii, et ühe meeskonna muudatused ei rikuks kogu rakendust
- Eeliseks, et arendusmeeskonnad suudavad kiiresti muutuvate rakenduste vajadustele vastavalt rakenduse olemasolevaid komponente muuta ja uusi luua

Viide: https://www.redhat.com/en/topics/microservices/what-are-microservices

TAL TECH

MONOLIITNE VS MIKROTEENUSTE ARHITEKTUUR



Viide: https://www.redhat.com/en/topics/microservices/what-are-microservices

CAP TEOREEM

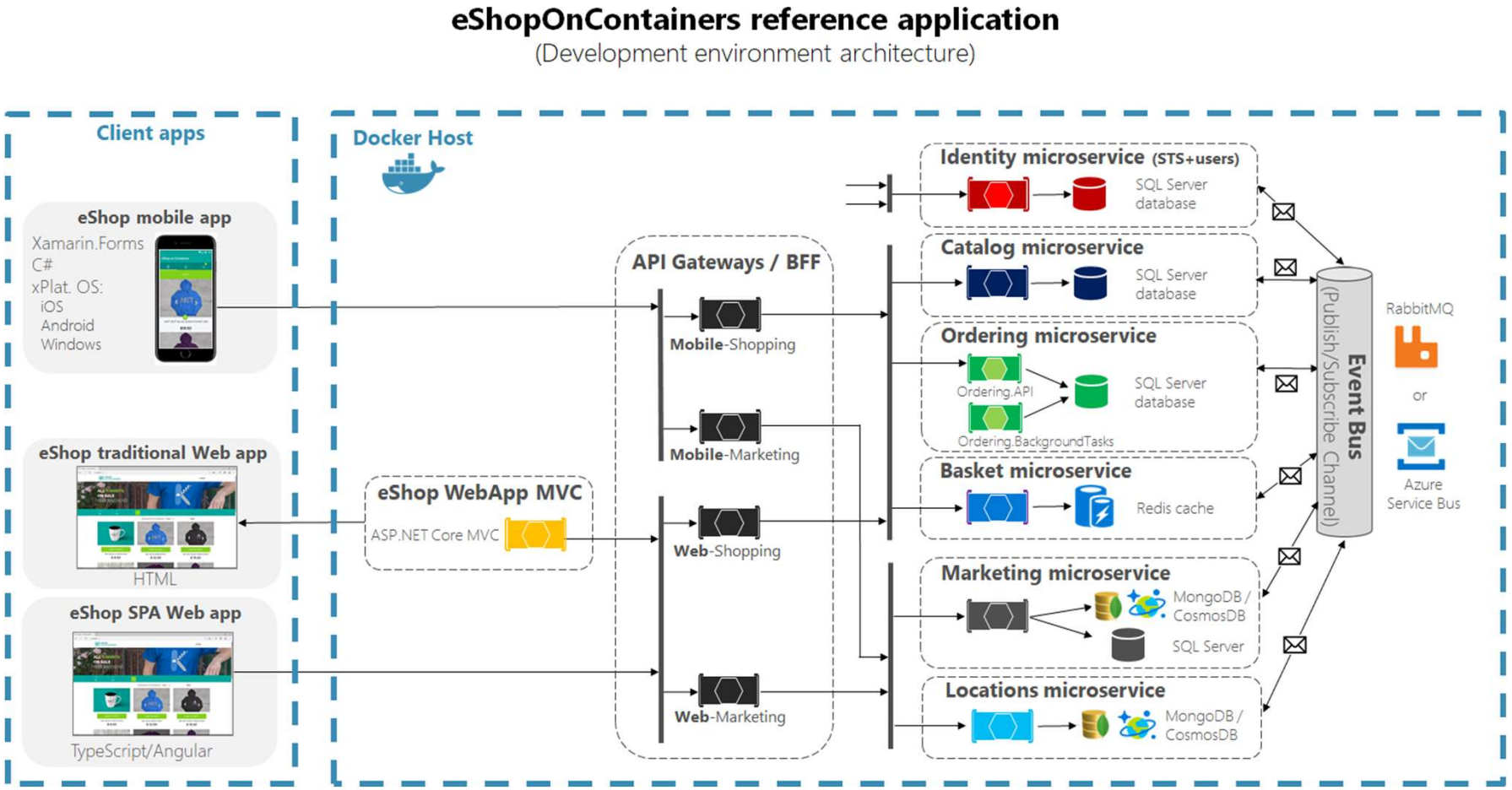
- **C**onsistency
- **A**vailability
- **P**artioner tolerance

- Teoreem ütleb, et korraga saab garanteerida ainult 2-te kolmest.

- Mikroteenuste puhul P peab olema alati garanteeritud. Me peame leidma kompromissi C ja A vahel.

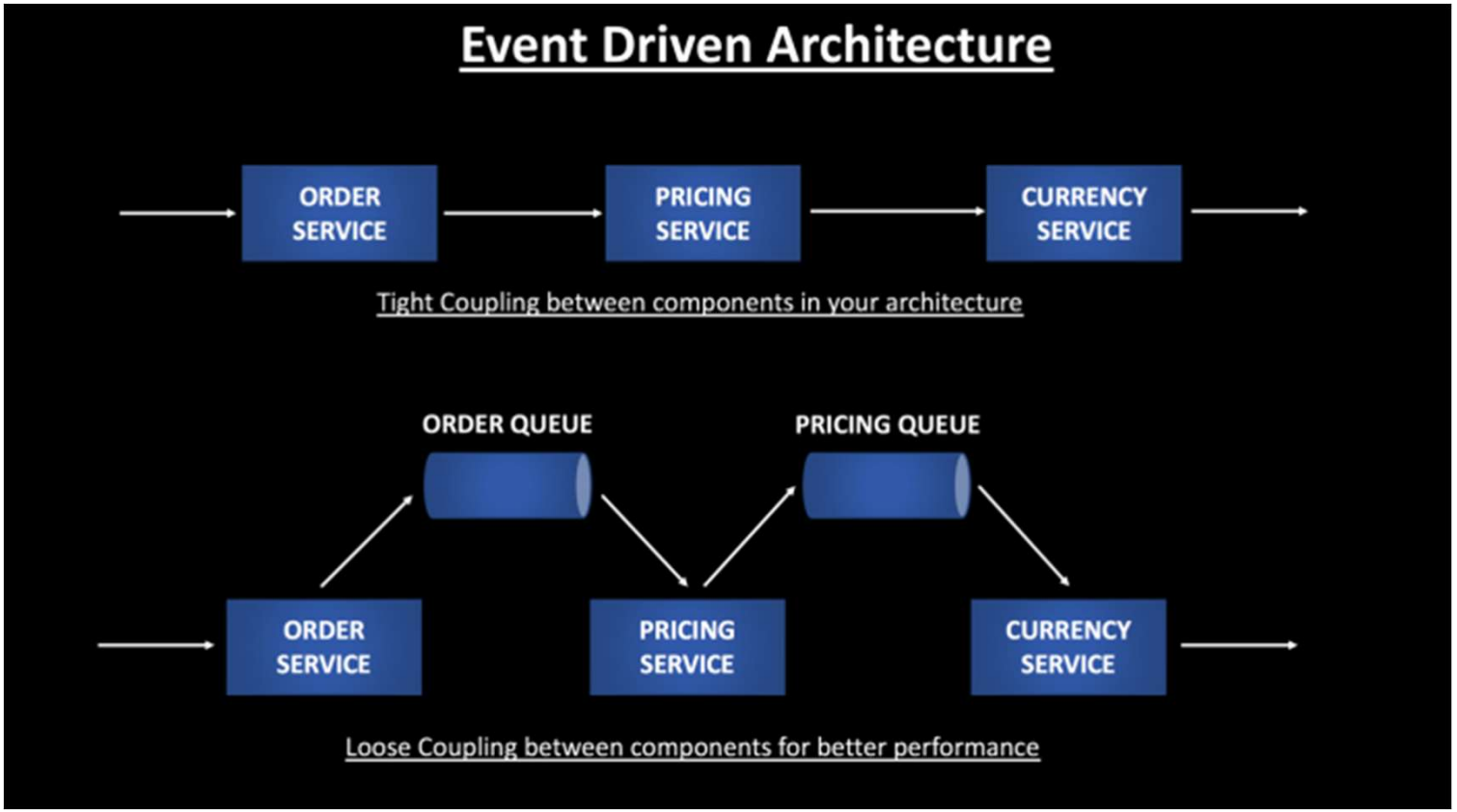
TAL TECH

API GATEWAY MUSTER



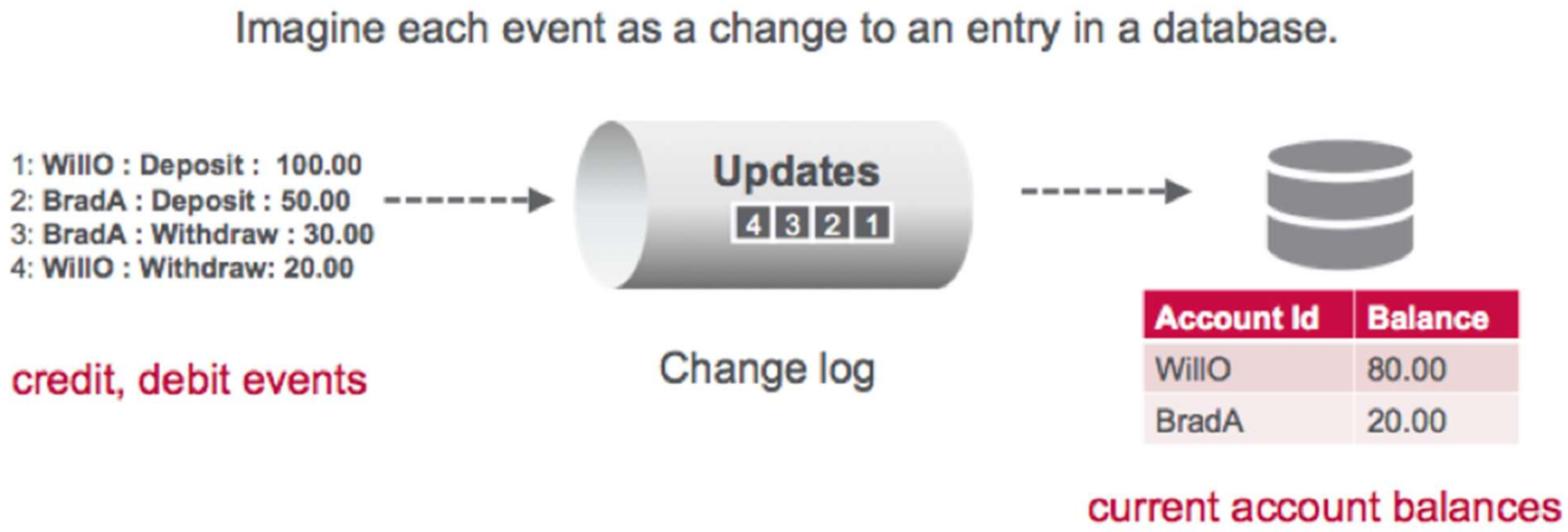
Viide: https://github.com/dotnet-architecture/eShopOnContainers

SÜNDMUSTEST JUHITUD ARHITEKTUUR/MUSTER



Viide: https://dzone.com/articles/breaking-the-monolithic-database-in-your-microserv

SÜNDMUSTEST JUHITUD ARHITEKTUUR/MUSTER



Viide: https://dzone.com/articles/event-driven-microservices-patterns

MIKROTEENUSED: UUE RAKENDUSE LOOMISEL 1. VALIK?

- Tavaliselt pole mikroteenused 1. valik uue rakenduse loomisel, kuna me tahame väikseima võimaliku pingutusega saada töötavat rakendust, millega saab kasutajatelt tagasiside, kas tehakse õiget asja
- Mikroteenused lisavad arendusele keerukust ja kui selgub, et kasutaja soovib midagi muud, siis on kogu lisandunud vaev olnud asjata
- Kui teeme monoliitse rakenduse ja selgub, et selle osi on vaja teha eraldi mikroteenusteks, kas siis monoliitse arendamine oli täiendav vaev?
- Kui meie kogemus ütleb, et mingi rakenduse osa vajab rohkem ressursse, siis tuleks see kohe mikroteenuseks teha

TAL TECH

KOKKUVÕTE: MIKS ME TAHAME KASUTADA MIKROTEENUSEID?

- There is a team of developers working on the application
- New team members must quickly become productive
- The application must be easy to understand and modify
- You want to practice continuous deployment of the application
- You must run multiple instances of the application on multiple machines in order to satisfy scalability and availability requirements
- You want to take advantage of emerging technologies (frameworks, programming languages, etc.)

TAL TECH

Viide: https://microservices.io/patterns/microservices.html

KOKKUVÕTE: MIKS ME EI TAHA KASUTADA MIKROTEENUSEID?

- Developers must deal with the additional complexity of creating a distributed system:
  - Developers must implement the inter-service communication mechanism and deal with partial failure
  - Implementing requests that span multiple services is more difficult
  - Testing the interactions between services is more difficult
  - Implementing requests that span multiple services requires careful coordination between the teams
  - Developer tools/IDEs are oriented on building monolithic applications and don’t provide explicit support for developing distributed applications.
- Deployment complexity. In production, there is also the operational complexity of deploying and managing a system comprised of many different services.
- Increased memory consumption. The microservice architecture replaces N monolithic application instances with NxM services instances. If each service runs in its own JVM (or equivalent), which is usually necessary to isolate the instances, then there is the overhead of M times as many JVM runtimes. Moreover, if each service runs on its own VM (e.g. EC2 instance), as is the case at Netflix, the overhead is even higher.

TAL TECH

Viide: https://microservices.io/patterns/microservices.html