Elemente de arhitectura software

...

George Popa
Gemini Solutions

Cuprins

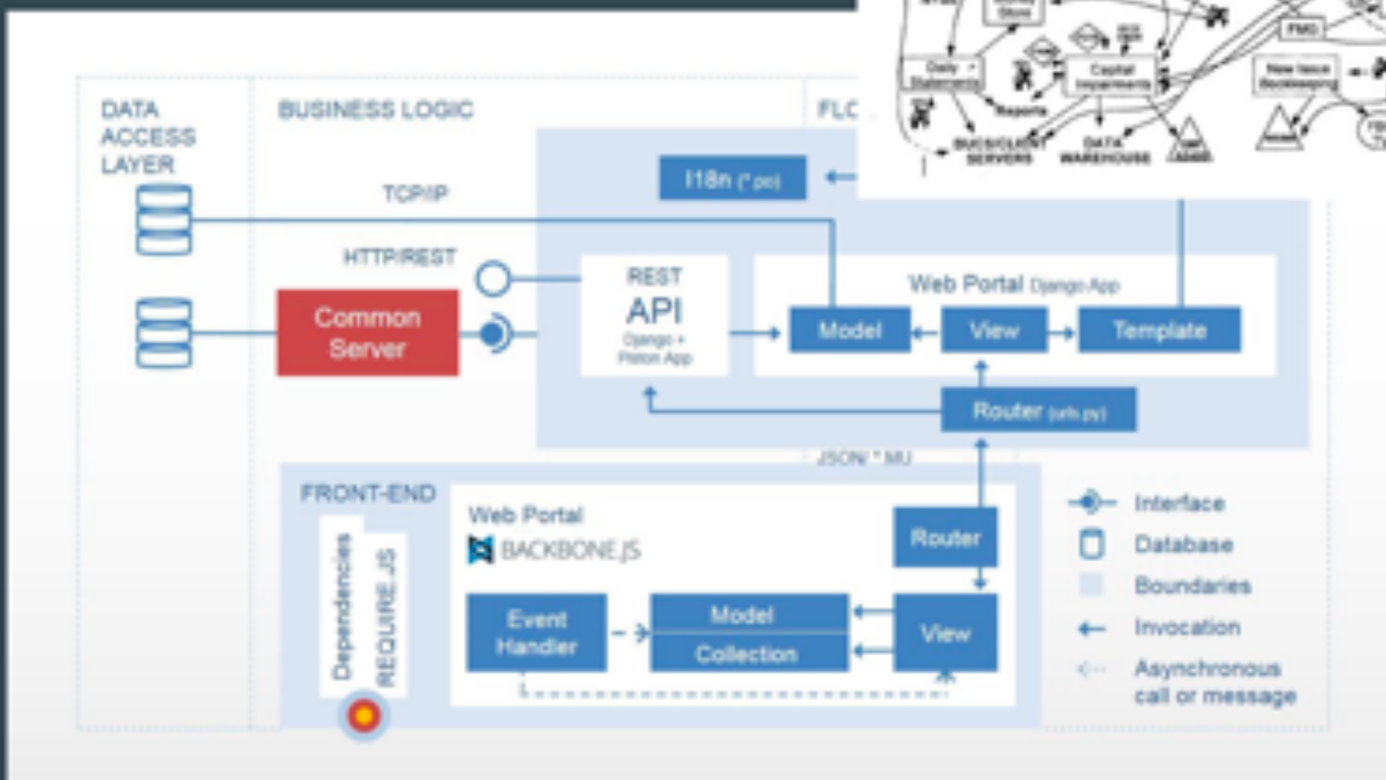
Introducere

- Definitie

Arhitectura aplicatiilor software este un concept ce descrie structura fundamentala a sistemelor software, din etapa de proiectare, dezvoltare si pana in productie, la un nivel inalt, pentru a raspunde diverselor cerinte, precum functionalitate, performanta, securitate, ...etc.

Descrierea schematica cuprinde atat informatii despre fiecare element principal al sistemului, cat si relatia dintre aceste componente.

A nu se confunda cu arhitectura calculatoarelor sau sistemelor de calcul, sau arhitectura intreprinderilor.


DATA ACCESS LAYER
BUSINESS LOGIC
TCP/IP
HTTP/REST
Common Server
REST
API
Web Portal Django App
Model
View
Template
I18n
Router
FRONT-END
Web Portal
BACKBONE.JS
Dependencies
REQUIRE.JS
Router
Event Handler
Model
Collection
View
Interface
Database
Boundaries
Invocation
Asynchronous call or message

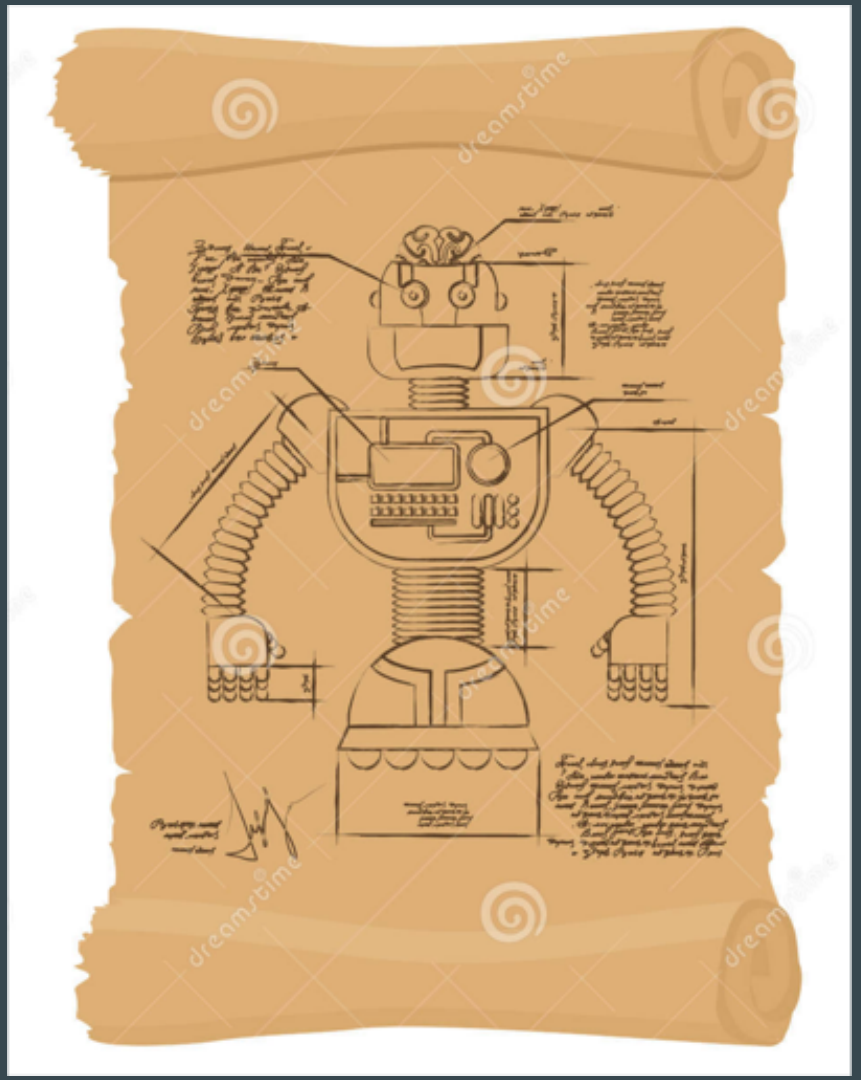

NIGHT CYCLE
Clearance
Communication
Receipt & Delivery
Reference Databases
General Ledger Interface
Global Exposure
Capital Requirements
DATA WAREHOUSE
General Expense
Position & Balances
Agency Inventory

Utilizare

- Intelegerea cerintelor aplicatiei afecteaza structura finala a acesteia;
- Nevoia de cunoastere in ansamblu a sistemului, de catre membrii echipei;
- Arhitectura aplicatiei software trebuie sa raspunda cerintelor initiale, curente si potential viitoare;
- Eventuale schimbari ale arhitecturii implica schimbari radicale in aplicatie. Acestea pot fi normale pe parcursul evolutiei unei aplicatii in productie, insa pot semnala si erori de proiectare, daca se schimba foarte des.

Evolutia arhitecturilor software

- 1970 Edsger W. Dijkstra “Structura sistemelor software este importanta, gasirea structurii corecte este critica”;
- 1990: apar primele definiri conceptuale, *patterns*, limbaje de descriere a arhitecturii software;
- Astazi avem mai multe standarde IEEE ce definesc arhitecturi software; arhitecturile software au devenit o disciplina in industria software, existand si rolul de arhitect software.



Importanta arhitecturii in proiectarea aplicatiilor

- Document ce se adreseaza atat clientului / beneficiarul aplicatiei, cat si echipei de dezvoltare;
- Se creaza o legatura intre cerintele logice ale aplicatiei si posibilitatile tehnologice;
- Se definesc exemple de utilizare curenta a aplicatiei si se gasesc metode practice de implementare ale acestora;
- Arhitectura descrie structura sistemului, fara a expune informatii detaliate despre implementare;
- Raspunde cerintelor de functionalitate, prin enumerearea acestora, precum si metrici de calitate, privind performanta, securitate, sau mentenanta.

Clasificare

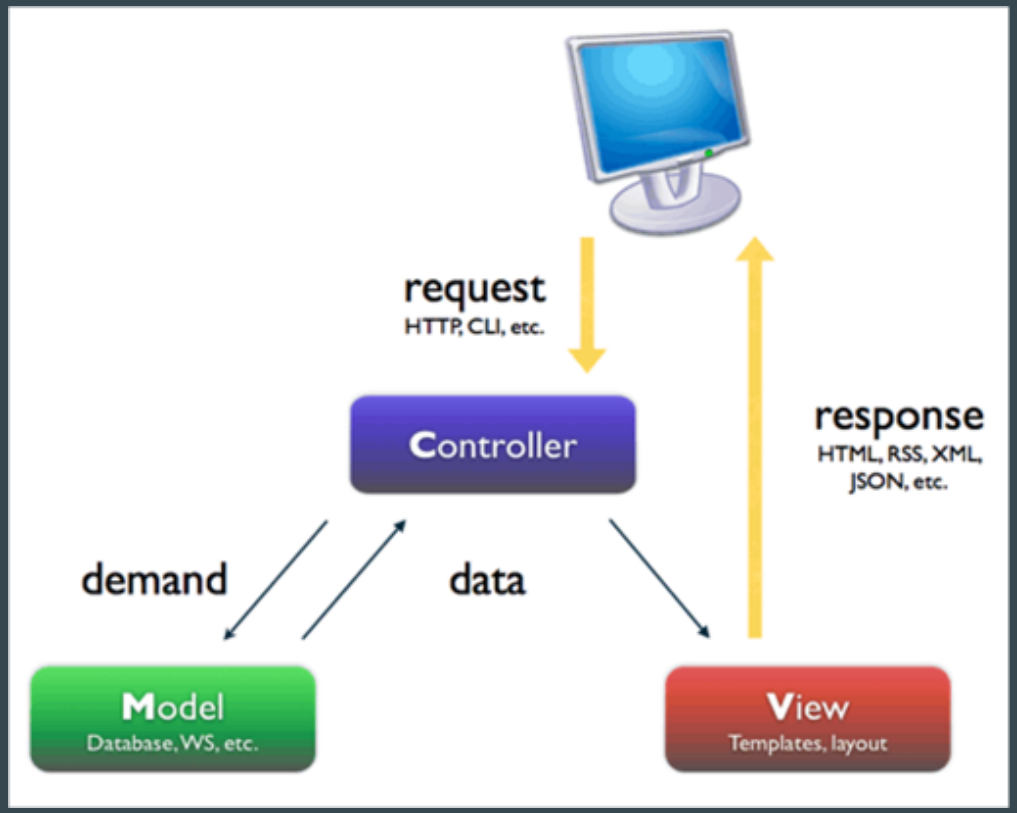
Modele (*styles / patterns*) de arhitecturi software:

- Structurale:
  - Client / server
  - Arhitecturi bazate pe componente
  - Arhitecturi structurate pe nivele
- Functionale:
  - Mesagerie (*message bus*)
  - N - nivele / 3 - nivele (segregarea functionalitatii pe nivele)
  - Arhitecturi orientate obiect
  - Arhitecturi orientate servicii
    - Web services
    - Microservices

Model View Controller (MVC)

Software design pattern utilizat in implementarea aplicatiilor cu interfata catre utilizatori, de regula aplicatii web.

Limbaje de programare precum Java, C#, Ruby sau PHP ofera *frameworks* MVC complete pentru dezvoltarea acestor aplicatii.



Model View Controller

- **Model** - Componenta software ce reprezinta resursele conceptuale puse la dispozitie de catre aplicatie, de regula instantiate prin tehnici ORM din medii de stocare persistente.

- **View** - Construieste interfata prezentata utilizatorilor, pe baza cerintelor transmise de catre controller sau interogand direct modelele, obtinand ca rezultat un cod HTML, XML, JSON, text, ...etc.

- **Controller** - Componenta logica a aplicatiei, ce intercepteaza cererile clientilor, interogheaza baza de date prin intermediul modelelor, construieste un raspuns utilizand *view-urile*, pe care apoi il intoarce catre client.

Model View Controller

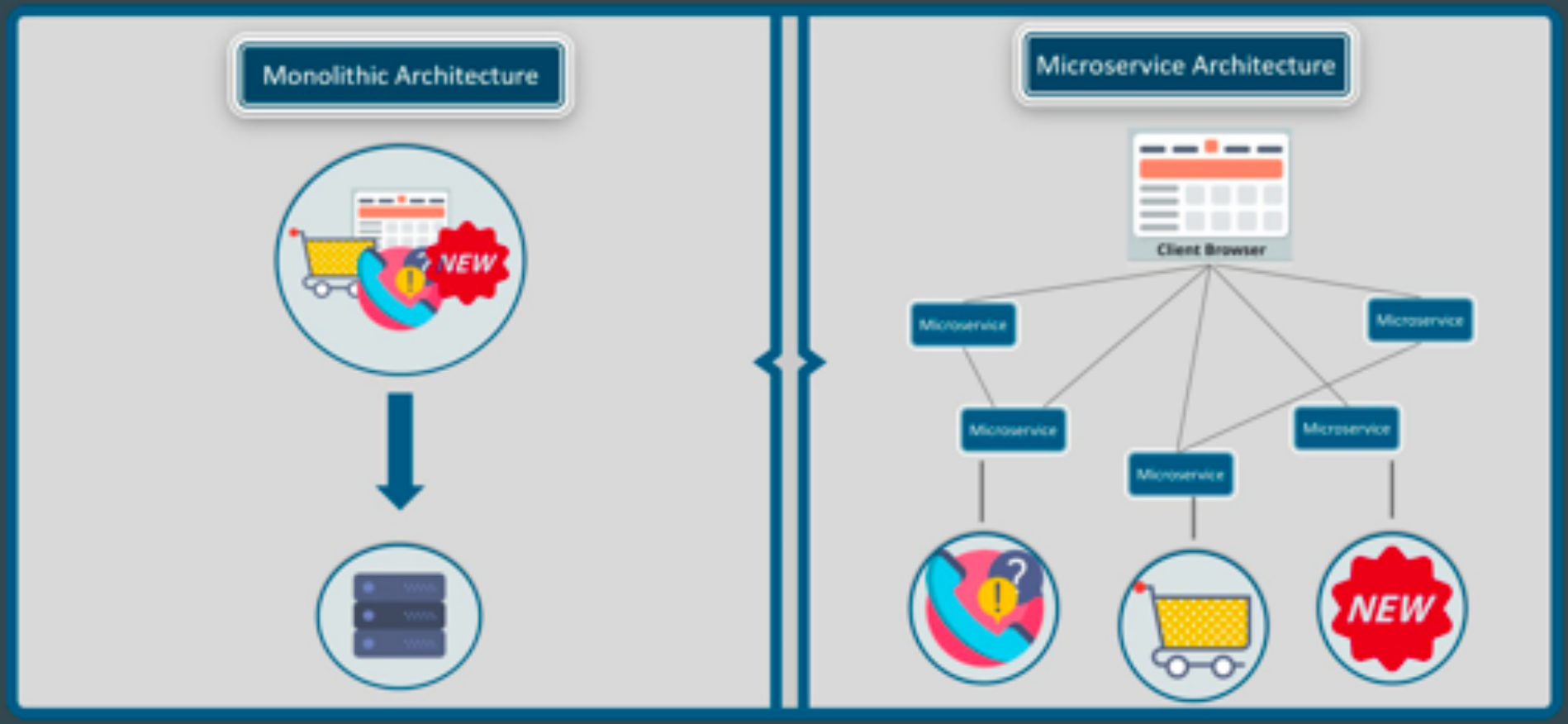
- Avantaje:
  - Decupleaza componentele aplicatiei, ce pot fi ulterior mai usor de realizat, modificat sau inlocuit;
  - Modularizeaza logic aplicatia, previne duplicarea codului sursa;
  - Permite crearea de interfete multiple pentru utilizatori, ale acelorasi date sau logica a aplicatiei (in general interfetele unei aplicatii se modifica mai des decat baza de date sau logica aplicatiei).
  - Datorita tehnologiilor diferite si utilizarea specifica a componentelor MVC, acestea pot fi dezvoltate si intretinute de o echipa specializata pe fiecare component.
- Dezavantaje:
  - Creste complexitatea aplicatiei, pe nivele;
  - Separa fiecare functionalitate pe mai multe nivele;
  - Necesita o perioada de adaptare mai mare a dezvoltatorilor, precum si aptitudini de baza pe toate cele trei nivele.

Model View Controller

Exemple de limbaje de programare si frameworks ce pun la dispozitie instrumente pentru creare de aplicatii MVC:

- PHP frameworks (Zend, Symfony, CodeIgniter...);
- Microsoft ASP.NET;
- Java EE (Servlets, Hibernate ORM, JSP, JSF);
- Ruby on Rails (ActiveRecords, ERBs).

Microservices

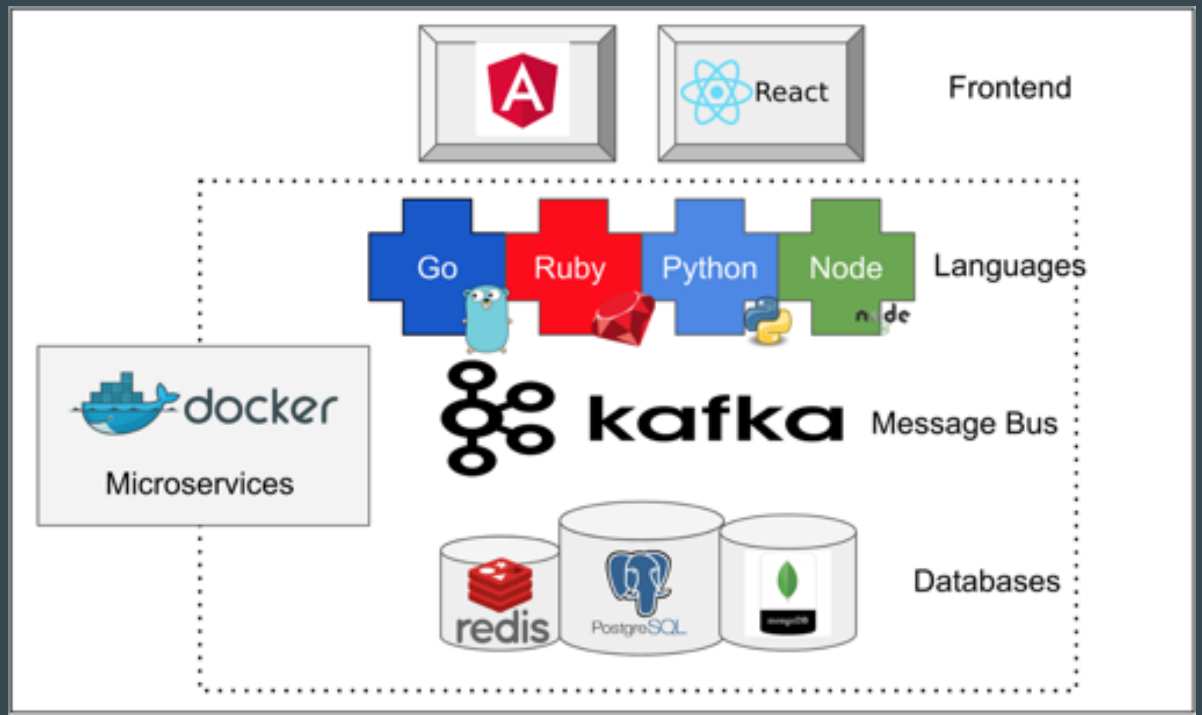


Microservices - particularitati

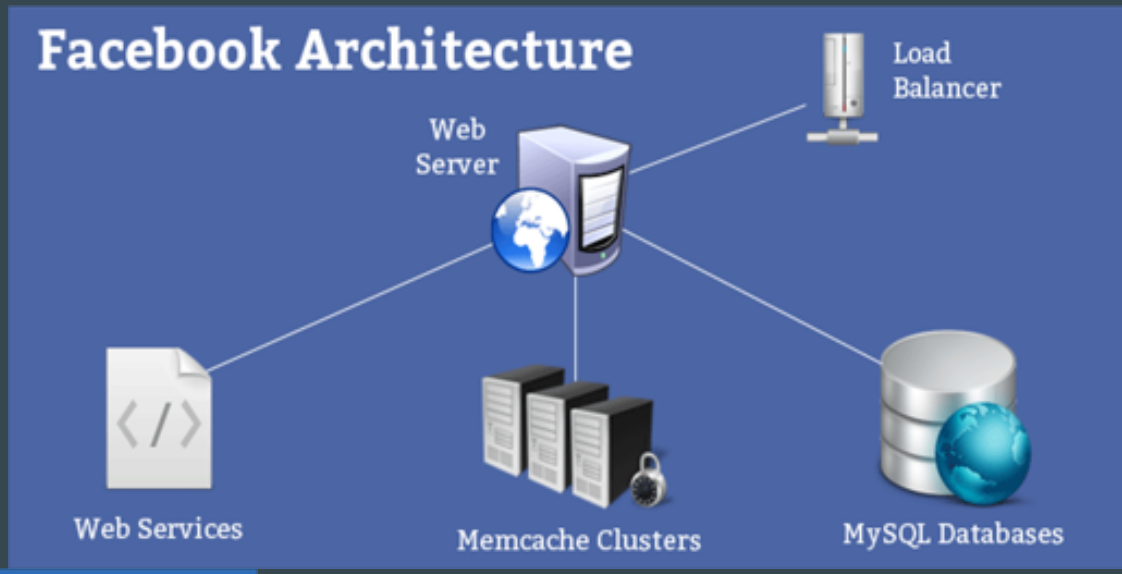
- Separarea functionalitatilor in servicii adiacente independente
- Dimensiune redusa a componentelor, deployment individual
- Permite izolarea dezvoltarii, testarii, *Continuous delivery*
- Comunicarea / colaborarea serviciilor devine element cheie al sistemului
- Complexitatea initiala a aplicatiei monolit este transferata catre complexitatea operationala

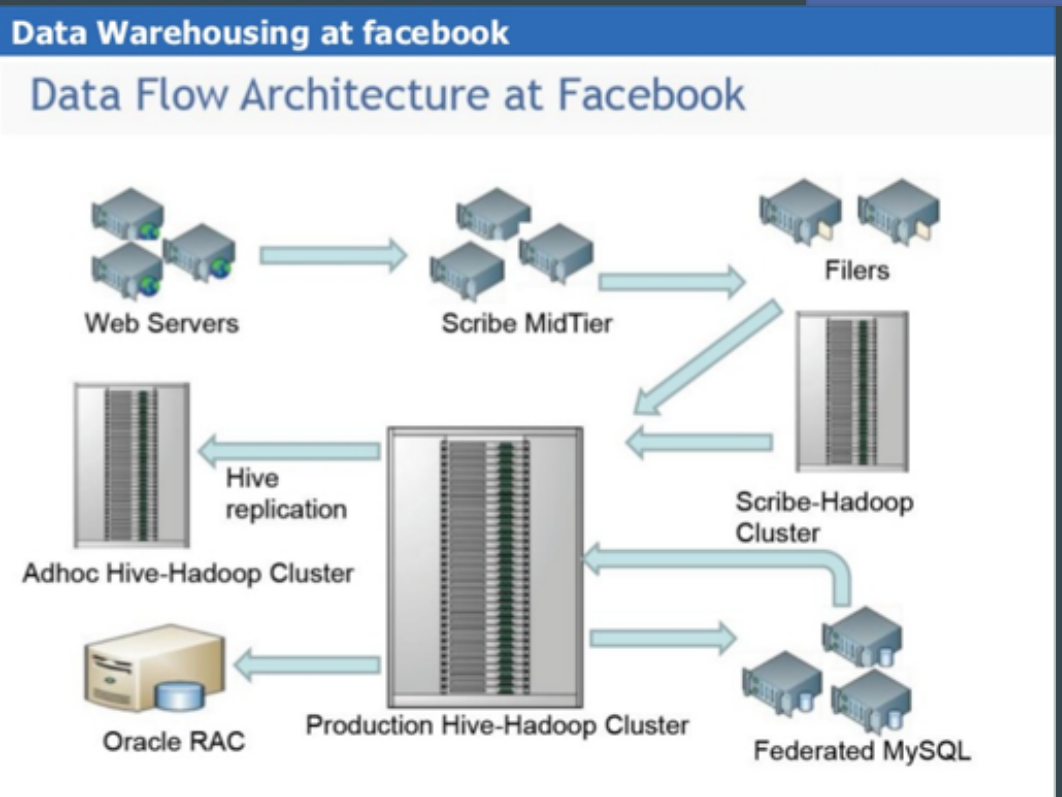
Microservices - technology stack

- Sisteme de operare: micro-Linux
- Containers: Docker
- Container management: Kubernetes
- Monitoring: Prometheus



Arhitectura Facebook





Arhitectura Netflix

- 93 M active users
- 500 B events / day
- 800 M events / sec (peak)
- 1.3 PB data / day

- 100 data engineers / analysts
- Apache Kafka, Spark, Hadoop
- Amazon ES, S3, EMR