Integrarea aplicatiilor

Cursul 10

Agenda

1. Enterprise Integration Patterns
2. Niveluri de servicii in cloud
3. Modelul de maturitate al SaaS
4. Arhitectura de integrare aplicatii SaaS

Enterprise Integration Patterns

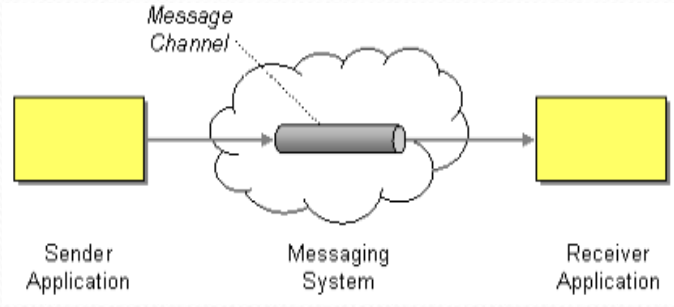
- EIP sunt design patterns cunoscute care își propun să rezolve diferite probleme de integrare prin oferirea de soluții la incidentele care pot apărea în timpul procesului de integrare ale aplicațiilor. Printre problemele care pot interveni se regăsesc:
  - eterogenitatea aplicațiilor, acestea fiind dezvoltate utilizându-se limbaje de programare diferite, rulând pe sisteme de operare diferite și lucrând cu diferite formate de date;
  - multitudinea de schimbări ale aplicațiilor, acestea fiind des supuse unor îmbunătățiri astfel încât să poată fi adaptate cerințelor de business curente;
  - necesitatea schimbului de date prin intermediul unei rețele într-un mod fiabil și sigur, securitatea fiind un aspect foarte important.

Framework-uri de integrare

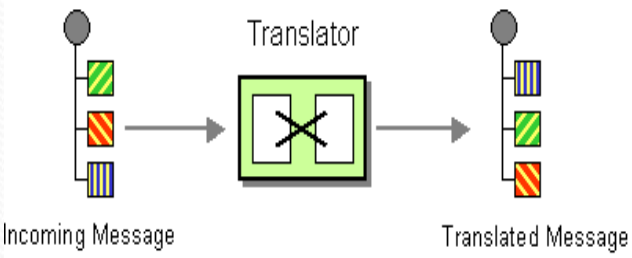
- Majoritatea framework-urilor de integrare sunt bazate și implementează un set de Enterprise design patterns.
- Framework-urile de integrare se pot utiliza atunci când se dorește integrarea cu o aplicație existentă, conectarea la un serviciu web third party sau contruirea unei platforme de procesare a tranzacțiilor complexe.
- Pe lângă implementarea celor mai importante pattern-uri de integrare, framework-urile furnizează și următoarele funcționalități principale:
  - **Orchestrație**: organizarea și aranjarea componentelor în diferite fluxuri logice de procesare;
  - **Manipulare și transformare**: transformarea datelor de la un format la altul sau îmbogățirea conținutului datelor în formatul existent;
  - **Transport**: furnizare de comunicare într-o multitudine de formate precum HTTP, FTP sau JDBC;
  - **Mediere**: decuplarea obiectelor prin furnizarea unui nivel de comunicare prin care se realizează interacțiunea. [4]

Clasificare: 1) Sisteme de mesagerie

- **Canal de mesaje**
- Un canal de mesaj este un canal logic utilizat pentru a conecta aplicațiile, după cum se observă și în imaginea de mai sus. Una dintre aplicații scrie mesaje pentru canalul de mesaj, în timp ce cealaltă aplicație sau celelalte, în cazul în care există mai multe aplicații interconectate, citește sau citesc mesajul recepționat de la canal. Exemplu: coada de mesaje.

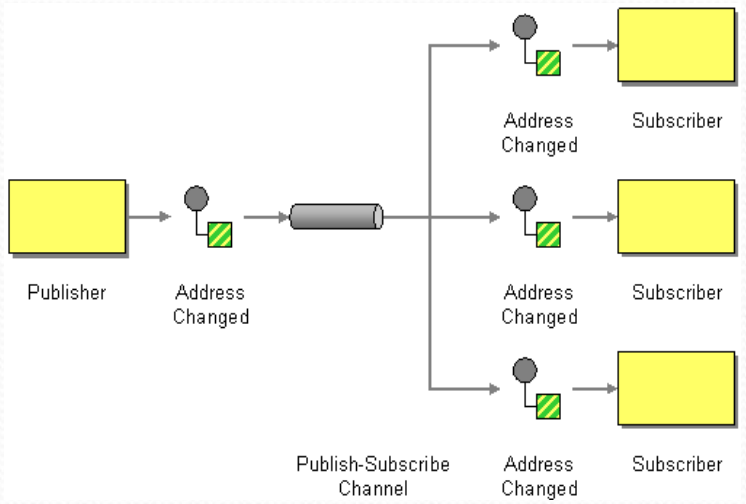


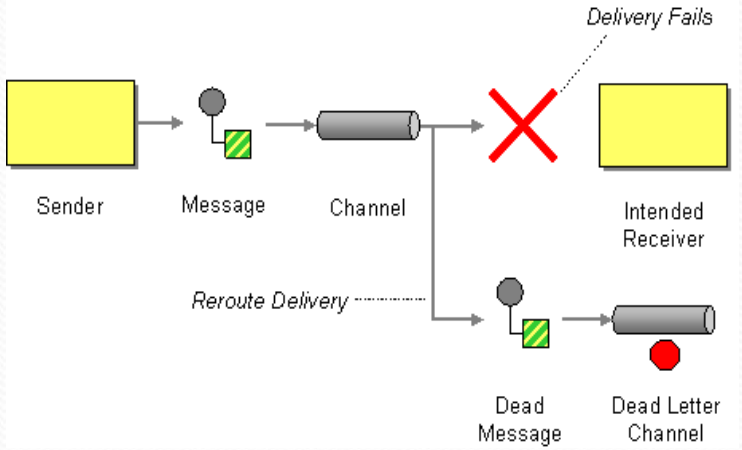
- **Translator de mesaje**
- Un translator de mesaje are rolul de a transforma mesajele dintr-un format în altul. De exemplu, o aplicație trimite un mesaj în format XML, dar cealaltă aplicație acceptă doar mesaje JSON astfel încât una dintre părți trebuie să se ocupe de transformarea sau convertirea datelor din XML în JSON.



Clasificare: 2) Canale de transmitere a mesajelor

- **Canalul de publicare-abonare -** Acest tip de canal are rolul de a transmite un eveniment sau o notificare către toți receptorii care sunt abonați. Fiecare abonat primește mesajul o singură dată și următoarea copie a acestui mesaj este ștearsă din canal.
- **Canalul Dead Letter -**Acest tip de canal descrie următorul scenariu: ce să se întâmple în cazul în care sistemul de mesagerie stabilește că acesta nu poate livra un mesaj la destinatarul specificat. Acest lucru poate fi cauzat de probleme ale conexiunii sau alte inconveniente precum memoria sau spațiul de pe disc care sunt prea pline. De cele mai multe ori, în acest proces de trimitere a mesajului către acest tip de canal, au loc multiple încercări de livrare a mesajului.

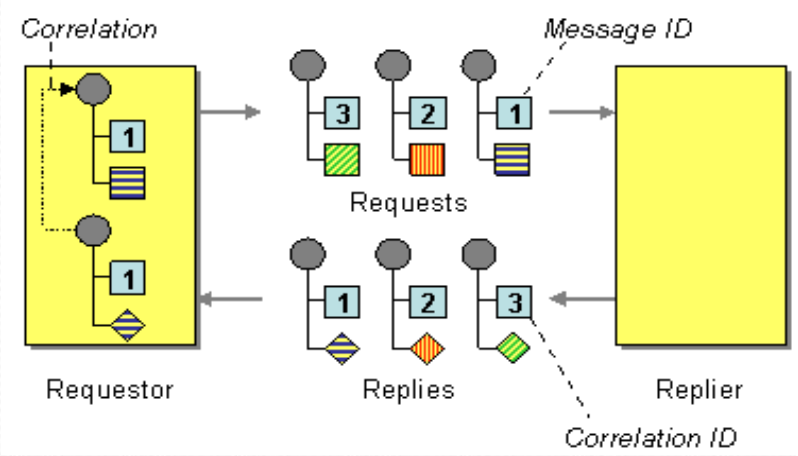




Clasificare: 3) Construcții de mesaje

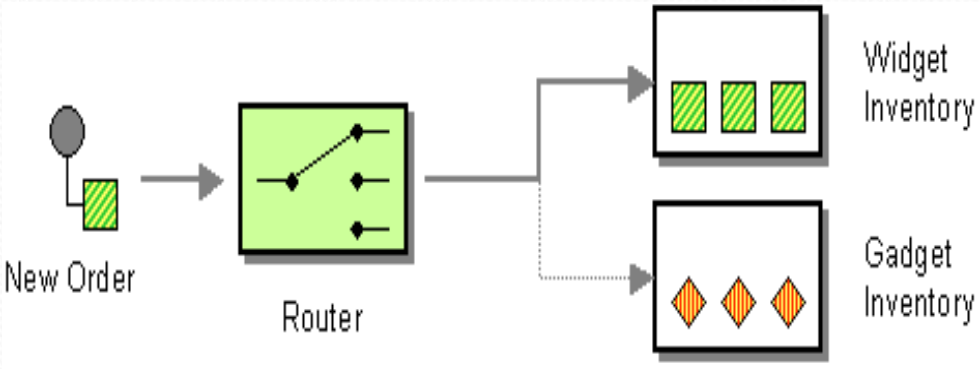
Identificatorul de corespondență

- Acest identificator oferă posibilitatea de verificare a cererii și de a genera un mesaj de răspuns ori de câte ori este folosit un sistem de mesagerie asincron. De obicei, procesul are loc în următorul mod:
  - furnizorul generează un identificator unic de corespondență;
  - tot furnizorul trimite mesajul cu identificatorul unic de corespondență atașat;
  - consumatorul procesează mesajul și trimite răspuns cu identificatorul de corespondență dat în cerere atașat;
  - furnizorul pune în corelație cererea și trimite mesaj bazat pe identificatorul de corespondență.



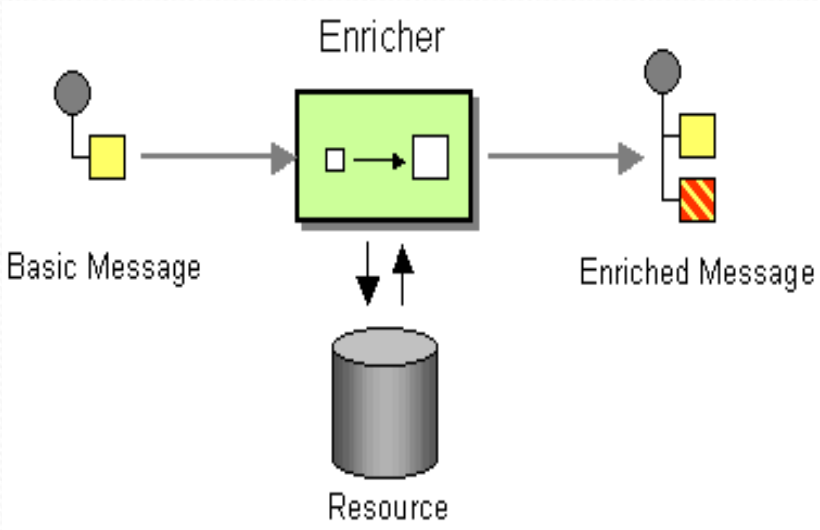
Clasificare: 4) Mesaje de rutare

- **Router bazat pe conținut**
- Acest router examinează conținutul mesajului și mesajele din rută pe baza datelor conținute în mesaj.



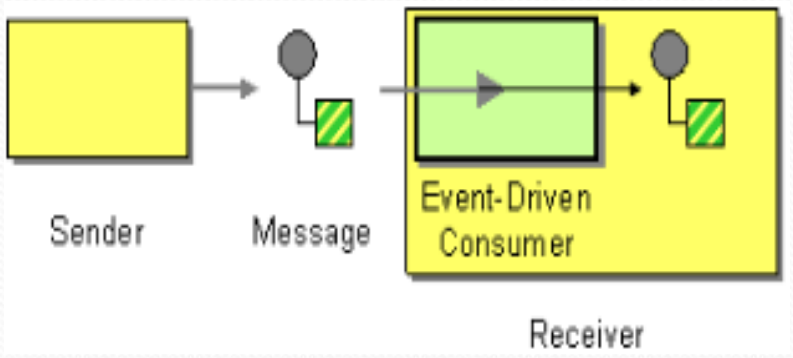
Clasificare:5) Transformarea mesajului

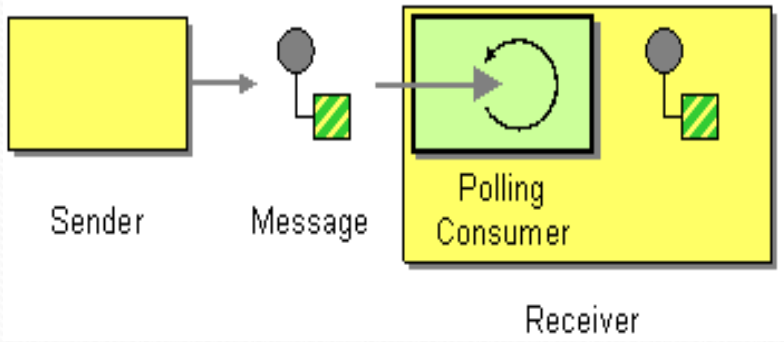
- **Îmbogățirea conținutului**: După cum reiese și din denumire, acest tip de pattern se ocupă cu îmbogățirea mesajului cu informațiile care lipsesc, folosindu-se de cele mai multe ori o sursă de date externă cum ar fi o bază de date sau un serviciu web.



Clasificare: 6) Puncte finale de mesagerie

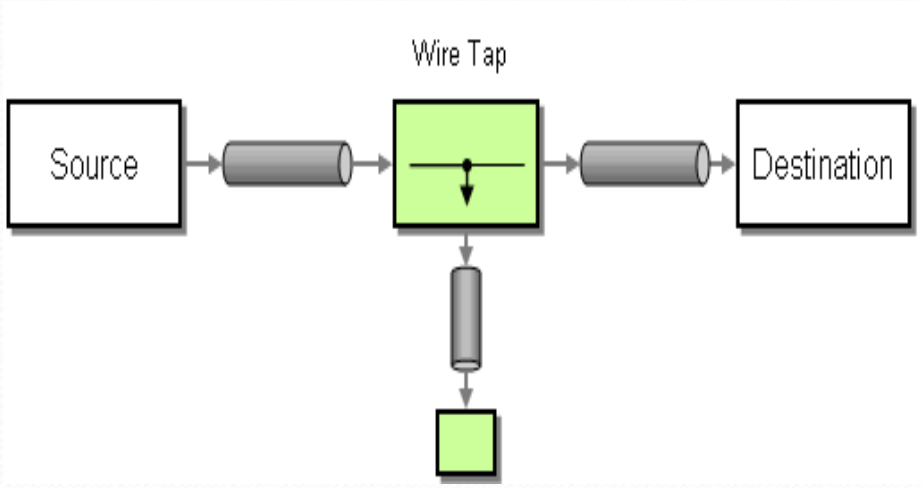
- **Event-Driven Consumer**
- Acest pattern permite furnizarea unei acțiuni care este apelată în mod automat de către canalul de mesaje sau nivelul de transport.
- **-Polling Consumer**
- Ceea ce este important în acest pattern este faptul că este sincron, deoarece blochează firul de execuție până când mesajul este primit.





Clasificare: 7) Gestionarea sistemului

- **Wire Tap :** Acest pattern copiază un mesaj pe care îl direcționează către un canal separat, în timp ce mesajul original este transmis la canalul de destinație.



SPRING INTEGRATION

VS.

APACHE CAMEL

APACHE CAMEL

- Apache Camel este un motor de rutare și de mediere bazat pe reguli, care oferă o implementare bazată pe obiecte a Enterprise Integration Patterns (EIP).
- Foloseste o interfață de programare a aplicațiilor (API) pentru a configura reguli de rutare și de mediere.
- Limbajul specific domeniului înseamnă că Apache Camel poate suporta completarea de tip type-safe a regulilor de rutare într-un mediu de dezvoltare integrat, folosind cod Java obisnuit, fără multe fișiere de configurare XML.

SPRING INTEGRATION

- Spring Integration este o platforma open-source pentru integrarea aplicatiilor enterprise. Aceasta se bazeaza pe codul de baza al platformei Spring, facand chiar parte din portofoliul Spring.
- A fost conceput pentru a permite dezvoltarea solutiilor de integrare a arhitecturilor bazate pe evenimente si a celor bazate pe mesaje.
- Scopul principal este de a oferi un model simplu pentru construirea de soluții de integrare a aplicatiilor enterprise, menținând în același timp separarea preocupărilor - esențială pentru producerea de cod mentenabil si testabil.

Cazuri de folosire Spring Integration

- Cand proiectul este deja implementat cu Spring si scopul este optarea pentru tehnologii mai omogene.
- Cand arhitectura proiectului este bine definita, si nu este nevoie de adaptori customizabili, ci doar de integrari de baza (File, FTP, JMS, TCP, HTTP, servicii Web).
- Cand numarul tehnologiile ce urmeaza a fi integrate este relative mic.

Cazuri de folosire Apache Camel

- Cand proiectele folosesc limbaje de programare diferite, protocoale diferite, tehnologii diferite, DSL diferit.
- Cand testele, unitare, de integrare sau automate, sunt o prioritate.
- Cand managementul erorilor este o functionalitate importanta a proiectului/aplicatiei enterprise (Camel ofera functionalitati importante de suport pentru gestiunea erorilor).

Asemanari

- Ambele platforme sunt licentiate de acelasi proiect – Apache 2.0, ambele sunt implementate in limbajul de programare Java, si ambele proiecte au o activitate moderata spre inalta chiar si in momentul de fata, la mai mult de 9 ani de la lansare.
- Atat Apache Camel cat si Spring Integration impartasesc acelasi scop: sa ofere un mecanism usor de folosit pentru a implementa sarcini de integrare tipice (mediere, rutare, adaptarea protocoalelor), intr-un mod incorporabil in structura aplicatiilor deja existente.
- Tehnologii pe care cele 2 platforme le au in comun: servicii Web (SOAP, REST), JMS sau mesagerie asincrona, socketuri TCP, sisteme batch bazate pe fisiere, RDBMS, FTP/SFTP etc.

Deosebiri

- O diferenta importanta intre cele doua platform consta in testare. Initial, Spring Integration nu a avut niste practici prea bune pentru testarea rutelor in mod specific. Acesta functioneaza destul de bine, toti dezvoltatorii fiind deja obisnuiti sa scrie teste unitare sau de integrare cu Spring, si simuland colaboratorii cu EasyMock sau Mockito. Insa cu Apache Camel, testarea imbogatita, cu multiple functionalitati, vine implicit. Platforma de testare Camel este creata pe baza functionalitatilor Spring Test, deci are ce e mai bun din ambele platforme de testare, atat din Spring cat si din Camel.
- Diferențe semantice: in timp ce puterea de expresie a codului este aproximativ la fel, DSL-ul din Spring Integration expune cel mai de jos nivel EIP (canale, porți logice etc.), Camel DSL pare să se concentreze mai mult pe intenția de integrare.

Concluzii

- Spring Integration si Apache Camel ofera o abordare simpla si curata asupra problemelor de integrare ale aplicatiilor enterprise. In timp ce Sprin Integration adauga functionalitati peste portofoliul Spring si extinde modelul de programare familiar pentru domeniul de integrare, Apache Camel ofera un DSL bazat pe Java, se integreaza foarte bine cu Spring Core, avand o curba de invatare destul de blanda. Oricare dintre ele ar fi o alegere excelentă .

SaaS: Software as a Service

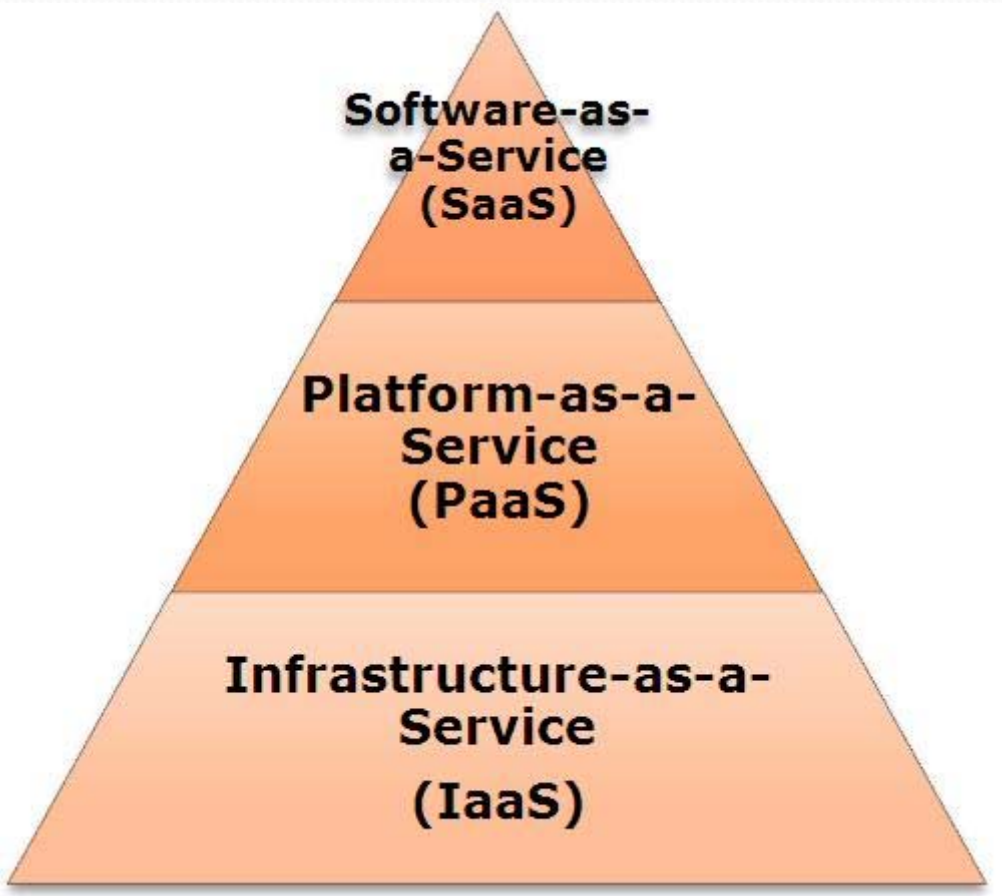
SaaS: Avantaje…

- *Cloud computing* - utilizatorii vor avea toate informațiile necesare, sisteme de operare, medii de dezvoltare de aplicații informatice, platforme de lucru pentru companii, baze de date imense, aplicații grafice, resurse în general la dispoziție, cu minimul de costuri pentru infrastructură, organizare și acces.
- un nou stil de a folosi resursele IT, aplicațiile informatice devenind servicii disponibile pe internet.
- stația de lucru - doar un punct de acces la informații, informația fiind stocată și redată utilizatorului indiferent de punctul de acces de pe care se conectează.
- *Cloud computing* - înglobează mecanismele de securitate în soluția de la distanță (antivirus, firewall). Deci protecția datelor și securitatea sunt asigurate cu costuri/eforturi minime, într-o manieră unitară, la distanță

Dezavantaje

- Virtualizarea are ca principale dezavantaje problemele legate de administrare și securitate.
- Administarea mediilor virtuale trebuie să permită integrarea acestora cu produsele informatice generale de management al centrelor de date.
- Alte arii problematice pentru virtualizare sunt reprezentate de securitate și de îmbunătățirea modului de licențiere.
- *Cloud computing* are nevoie de îmbunătățirea securității și interoperabilității platformelor, de portabilitate și interoperabilitate a aplicațiilor și a datelor, administrare și management, măsurare și monitorizare.

Niveluri in cloud



a.Infrastructure as a service – IaaS

- livrarea de resurse, cum ar fi servere, stocare și componente de rețea ca un serviciu
- avansul realizat de companii cum ar fi Amazon, Google sau Rackspace duc la o **culegere si stocare** a datelor mult mai ieftina.
- este nevoie de ea pentru a gestiona cantitatile uriase de date care au inundat firme care nu isi permit sa achizitioneze ferme de servere.
- acum, aceste servicii se pot plati pe masura ce sunt furnizate, costurile fiind mult mai usor de suportat.

b.Nivelul Saas

- Clientul poate utiliza aplicațiile software puse la dispoziție de furnizor pe o infrastructură de tip “cloud” – (servicii de găzduire web, servicii email, etc
- Clientul **nu poate configura** parametrii infrastructurii utilizate (bandă de transfer, servere, sisteme de operare, spațiu de stocare).
- **aplicatia raspunde tututror cerintelor functionale?**

Acesta cerinta poate fi un CRM sau un ERP

- Un alt aspect important este **arhitectura multi-tenant** a SaaS, care permite agregarea datelor dintr-o comunitate de utilizatori pentru a produce statistici benchmark, KPI foarte utili =>nivel de analiza superior clientilor
- ce se intampla in spatele dashboard-lui? **confidentialitatea datelor**,
- Exemple: Facebook, SalesForce, BaseCamp, etc.

c.Platform as a service (PaaS)

- PaaS= inchirierea hardware, sistem de operare, capacitati de retea si stocare in Internet.
- Clientul poate instala și configura pe infrastructura “cloud” furnizată aplicațiile software proprii, folosind instrumente și / sau bibliotecile de la furnizorul de servicii.
- Exemple: Google App Engine, Force.com, Microsoft Azure, WOLF, etc.
- dezvoltarea de noi aplicatii cu capacitate de analiza suplimentare in mediul de dezvoltare Paas poate fi o modalitate de a exploata mai bine datele, decat cu Saas.

Tipuri de integrare

In funcţie de aria de cuprindere şi de gradul la care se realizează virtualizarea:

- *public cloud*: resursele sunt dinamic expuse în internet prin intermediul serviciilor/aplicaţiilor web;
- *private cloud*: virtualizarea se realizează la nivel privat, în reţele private;
- *hybrid cloud*: include furnizori de servicii din internet şi din domenii private – un amestec între primele două tipuri de cloud.

SaaS: scalabil, multi-tenant, eficient, configurabil

- **Level I: Ad Hoc/Client – Fiecare client are propria versiune de aplicatie gazduita si rulata pe serverul gazda.** Similar cu ASP (Application Service Provider). Tranzitia de la aplicatii clasice la SaaS I – effort minim
- **Level II: Configurabil**- Furnizorul gazduieste cate o instanta separata de aplicatie pentru fiecare client, dar toate instantele folosesc acelasi cod, cu optiuni de configurare pt a arata si a se comporta diferit.
- Instantele raman complet izolate intre ele. Trecerea la acest model implica schimbari de arhitectura, daca aplicatia initiala folosea particularizari individuale, fara metadate de configurare
- Exemple: Facebook, SalesForce, BaseCamp, etc.

Model de maturitate

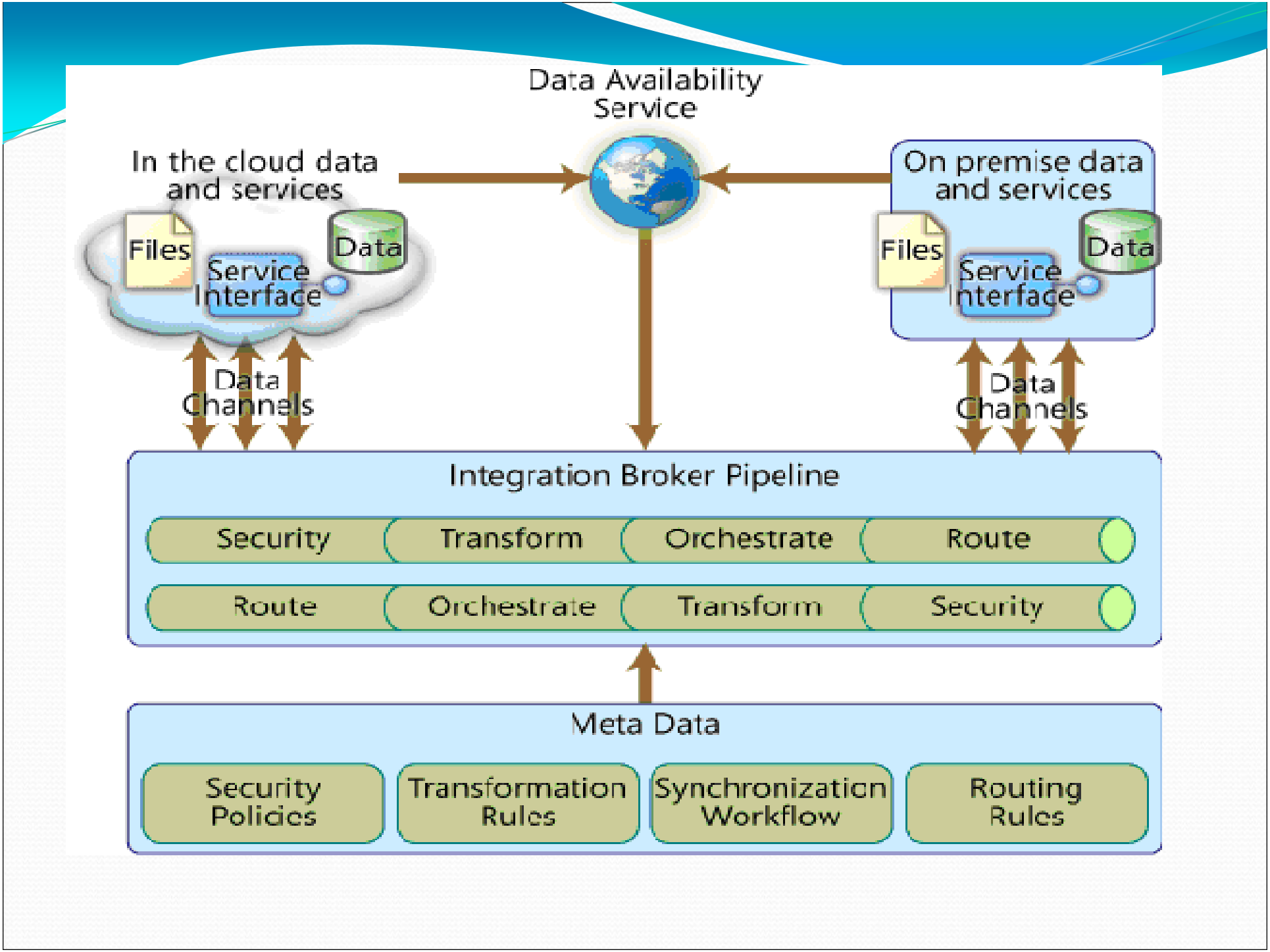
- **Level III: Configurabil, Eficient pentru chiriasi multipli**- Furnizorul ruleaza o singura instanta de aplicatie, folosind date de configurare pentru ca aceasta sa arate si sa se comporte diferit pentru fiecare.
- Politicile de securitate si autorizare –datele sunt in siguranta, separate de ale altor clienti. Modelul are scalabilitate redusa.
- **Level IV: Scalabil, Configurabil, Eficient pentru chiriasi multipli** – furnizorul gazduieste mai multi chiriasi pe o ferma de instante identice, fiecare client stocheaza date separat, se furnizeza metadate de configurare.
- Sistemul este scalabil

Arhitectura de integrare

- SaaS presupune gazduirea datelor inafara retelei locale controlate, in cloud
- Fiecare componenta trebuies a aiba acces la datele de care are nevoie, indiferent de unde vin datele
- Se poate configura aplicatia **SaaS sa depinda de datele produse de aplicatii on-premise** ca parte a functionalitatii lor (de ex, o aplicatie CRM SaaS care refera date despre stocuri gestionate de o aplicatie de gestiune stocuri on-premise)
- Se poate configura aplicatia **on-premise application sa depinda de datele produse de o aplicatie SaaS ca parte a functionalitatii** (de ex, o aplicatie on-premise de salarizare care refera dare de HR gestionate de o aplicatie HR de tip SaaS)

Broker de integrare

- Folosit de companii pentru a expune functiile aplicatiilor, a gestiona procese de afaceri, ai a realiza integraea cu sistemele interne de back-end
- Datele pot proveni din surse diferite, pot utiliza protocoale diferite si pot avea o multime de formate incompatibile
- Are o arhitectura de tip **pipeline/ magistrala** careia i se pot adauga module care realizeza operatii de integrare specifice


Data Availability Service
In the cloud data and services
Files
Data
Service Interface
On premise data and services
Files
Data
Service Interface
Data Channels
Data Channels
Integration Broker Pipeline
Security
Transform
Orchestrate
Route
Route
Orchestrate
Transform
Security
Meta Data
Security Policies
Transformation Rules
Synchronization Workflow
Routing Rules

Broker de integrare

- Datele intra si ies din magistrala prin canale de date care definesc protocoalele folosite pentru a comunica cu sursele de date. De ex: un canal poate transmite date de la un serviciu Web la broker folosind SOAP
- Modulele conectate la magistrala determina modul de procesare a datelor, rutarea si integrarea lor cu datele la destinatie.
- Un serviciu de metadate ofera regulile de configurare pe care fiecare modul le foloseste pentru a-si realiza treaba

Operatii de integrare

- **Securitate**—Modulul de securitate realizeza operatii de autentificare a sursei de date sau semnaturii digitale, decriptare a datelor si examinarea lor pentru a examina risculride securitate aplicandu-se politicile de securitate existente pentru a controla accesul
- **Validare** - Modulul de validare poate compara datele cu niste scheme si sa respinga datele necorespunzatoare sau sa le transmita unei componente de transformare pentru a le trece intr-un format corespunzator.
- **Fluxul de sincronizare** – componenta de sincronizare foloseste fluxul de lucru si reguli pentru a determina modul de propagare al schimbarilor la destinatii si ordinea acestora.
- **Rutarea**—Regulile de rutare definesc destinatia pentru fiecare informatie.