Integrarea sistemelor informatice

Suport curs nr. 5

Programator >> Arhitect

Aplicații compozite

2024-2025

C5 – Aplicații compozite

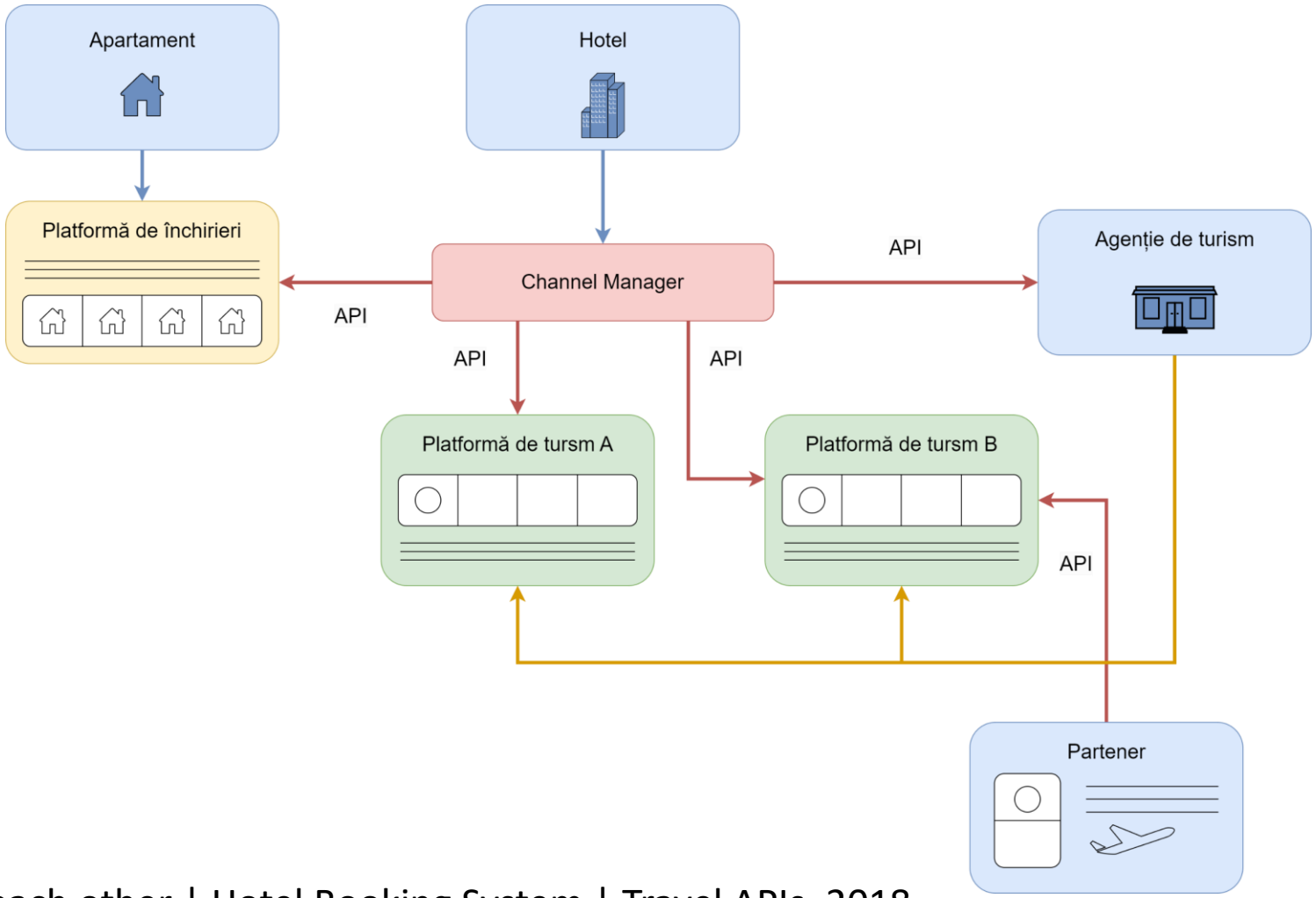
Obiective

- Înțelegerea conceptului de aplicații compozite
- Identificarea etapelor de inginerie a proiectării
  - Modelarea domeniului
  - Framework-uri
  - Tehnologii
  - Modele de proiectare

Recapitulare – Integrarea aplicațiilor

Problemă

- Existența mai multor aplicații și platforme necesită metode de integrare diverse
  - Baze de date
  - UI
  - SDK-uri
  - Framework-uri
  - API-uri



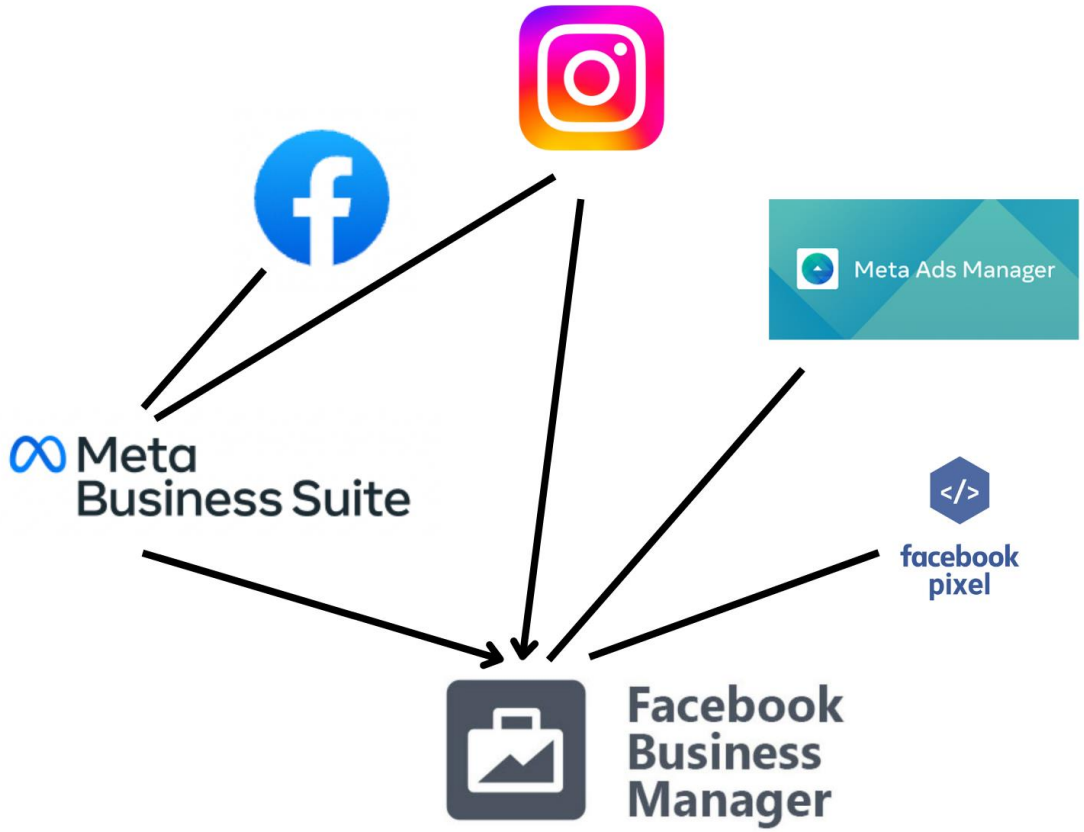
Sursa: How travel systems talk to each other | Hotel Booking System | Travel APIs, 2018

Studiu de caz: Meta Business Suite

Concept: Interfață centralizatoare “all-in-one” pentru gestionarea facilă a paginilor business pe Facebook și Instagram

Context: un ecosistem în continuă schimbare

Rezultat: o serie de interfețe imbricate, dificil de navigat – “așa nu”

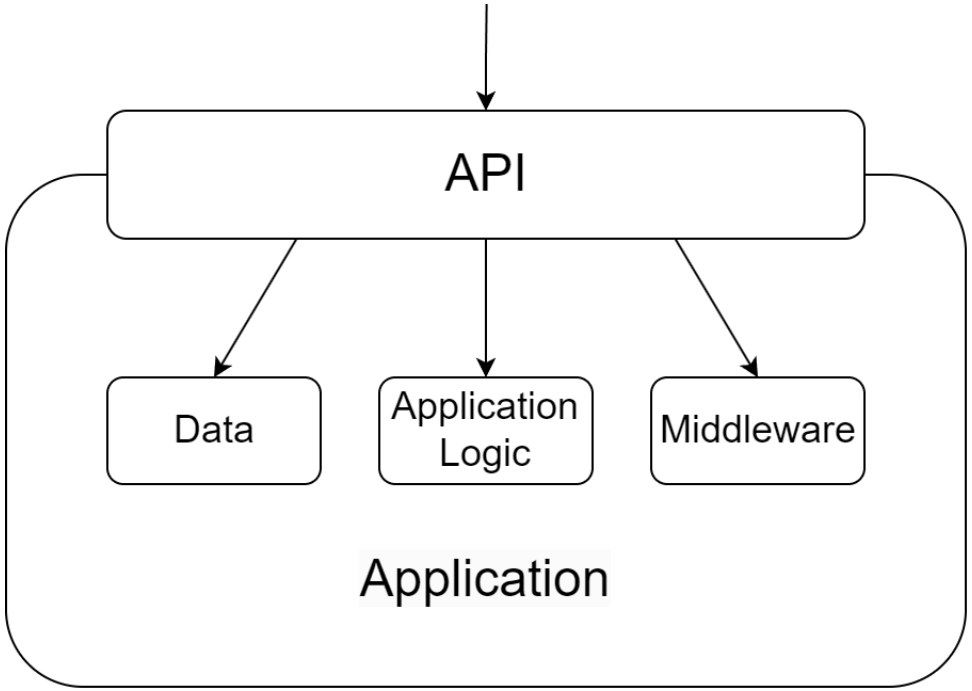


Sursa: Meta Business Suite vs Meta Business Manager, TwoSix Digital

Recapitulare – Arhitectura unui API

Modele arhitecturale de API

- **SOAP** – XML, aplicații enterprise
- **RESTful** – HTTP, servicii web
- **GraphQL** – cereri structurate
- **gRPC** – comunicație microservicii
- **WebSocket** – aplicații de timp real
- **Webhook** – HTTP, notificări /callback



Sursa: Alex Xu, SOAP vs REST vs GraphQL vs RPC, ByteByteGo, 2022

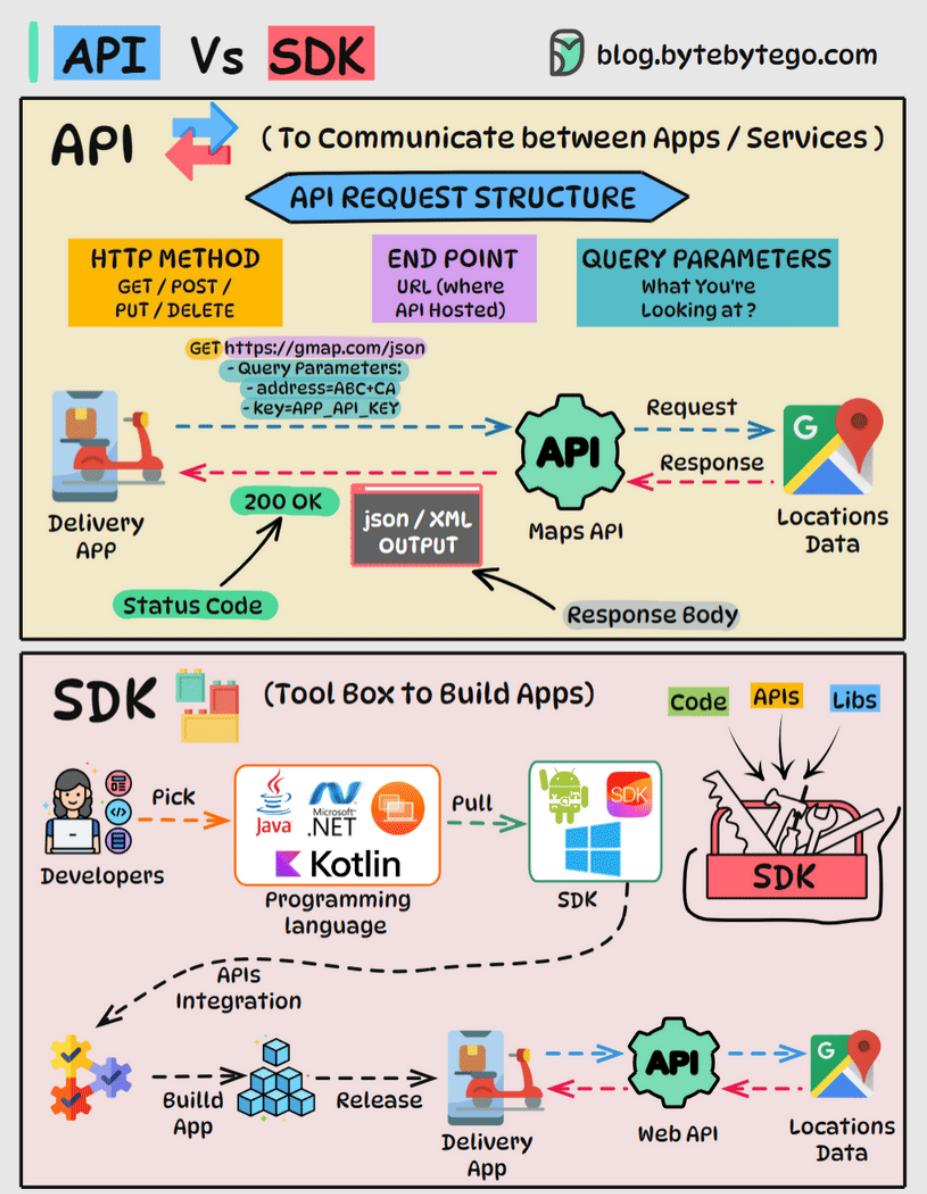
Recapitulare – API vs SDK

API

- Interfețe și protocoale de comunicație între aplicații software diferite

SDK

- Colecție de instrumente, biblioteci, secvențe de cod și documentație pentru realizarea aplicațiilor pentru o anumită platformă, framework, sau hardware



Sursa: ByteByteGo, EP83: Explaining 9 Types of API Testing, 2023

Recapitulare – API vs SDK

	API	**SDK**	**Bibliotecă**	**Framework**
Scop	Descrie interacțiunile dintre componente /aplicații diferite	Abstractizează implementările specifice	Încapsulează funcții și secvențe de cod reutilizabile	Modelează o structură complexă de biblioteci /servicii software
Utilizare	Aplicațiile interacționează prin API-uri	Aplicațiile includ SDK-uri	Aplicațiile includ biblioteci	Aplicațiile sunt construite pe baza unui Framework
Structură	Colecție de endpoint-uri de tip request – response	Colecție de instrumente și biblioteci specifice platformei	Colecție de funcții, clase, interfețe	Colecție de biblioteci și modele arhitecturale structurate
Exemple	• Google Maps API • ChatGPT API • Twitter API	• Android SDK • Java Development Kit	• Chart.js • Socket.io • Bootstrap	• .NET Framework • Spring Boot • Angular

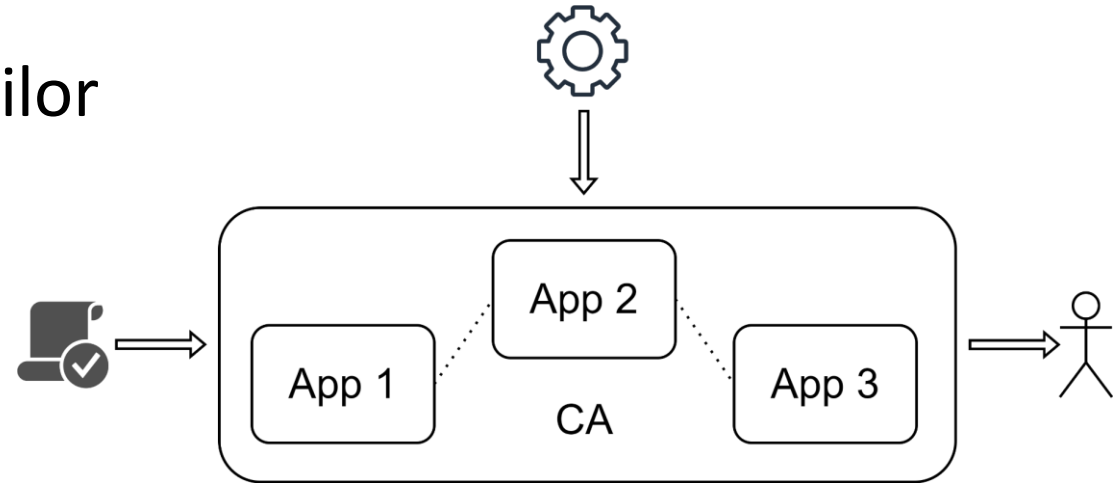
Proiectarea aplicațiilor compozite

Cum gândim problema?

- **Aplicații compozite / CA:** combinarea aplicațiilor / componentelor software existente într-o aplicație integrată

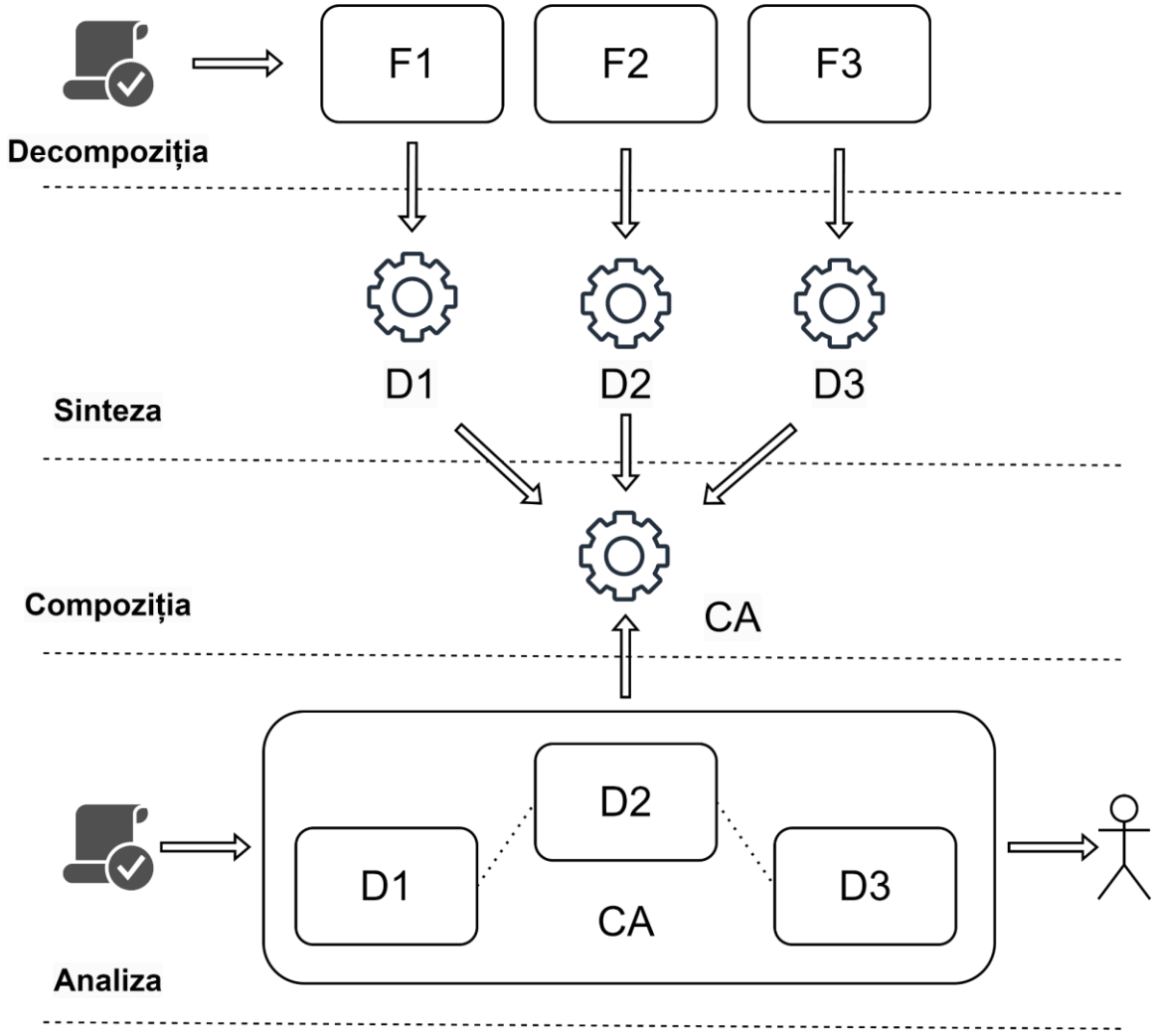
Cum abordăm proiectarea CA?

- **Ingineria proiectării:** aplicarea principiilor moderne de dezvoltare de software
- **Niveluri de abstractizare:** proiectarea arhitecturilor software integrate



Ingineria proiectării sistemelor compozite

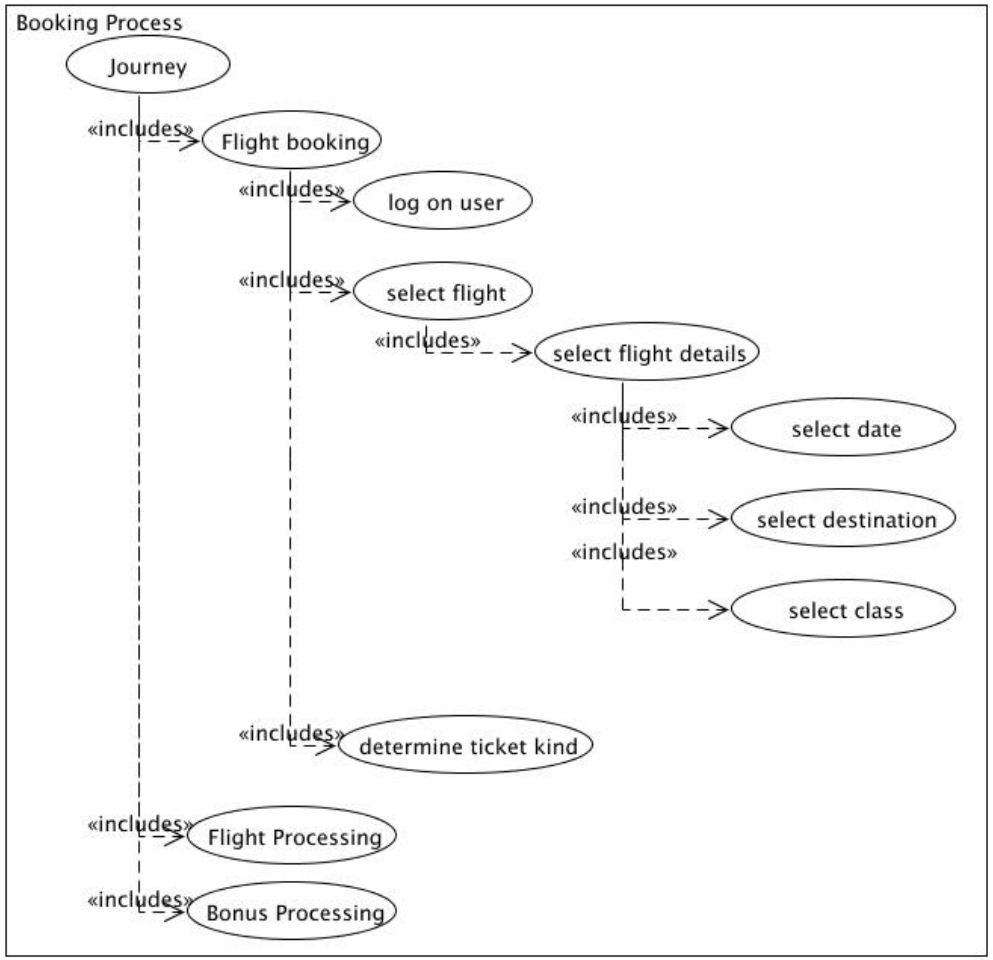
- Ingineria proiectării este o practică bine definită care are ca rezultat **specificația sistemului** ce satisface cerințele formulate
- Procesul ingineriei proiectării este o secvență recursivă care constă din patru activități de bază:
  - **Decompoziția** – func. / comport.
  - **Sinteza** – componente / dispozitive
  - **Compoziția** – CA
  - **Analiza** – testare / evaluare



Ingineria proiectării – Decompoziția

- O fază de decompoziție pleacă de la o cerință funcțională sau de comportament și o transformă într-un set de **sub-comportamente** care, împreună, vor realiza funcția / comportamentul cerut. Fiecare sub-comportament este atribuit unui anumit "dispozitiv".
  - Obiectivul acestei activități este de a identifica sub-comportamente care pot fi realizate printr-un dispozitiv cunoscut, sau sub-comportamente pentru care dispozitivul poate fi sintetizat.
  - Pot fi impuse cerințe suplimentare cu privire la dispozitiv, rezultate din nevoia de a lucra cu alte dispozitive, sau de a satisface seturi diferite de parametri ai sistemului.

Ingineria proiectării – Decompoziția

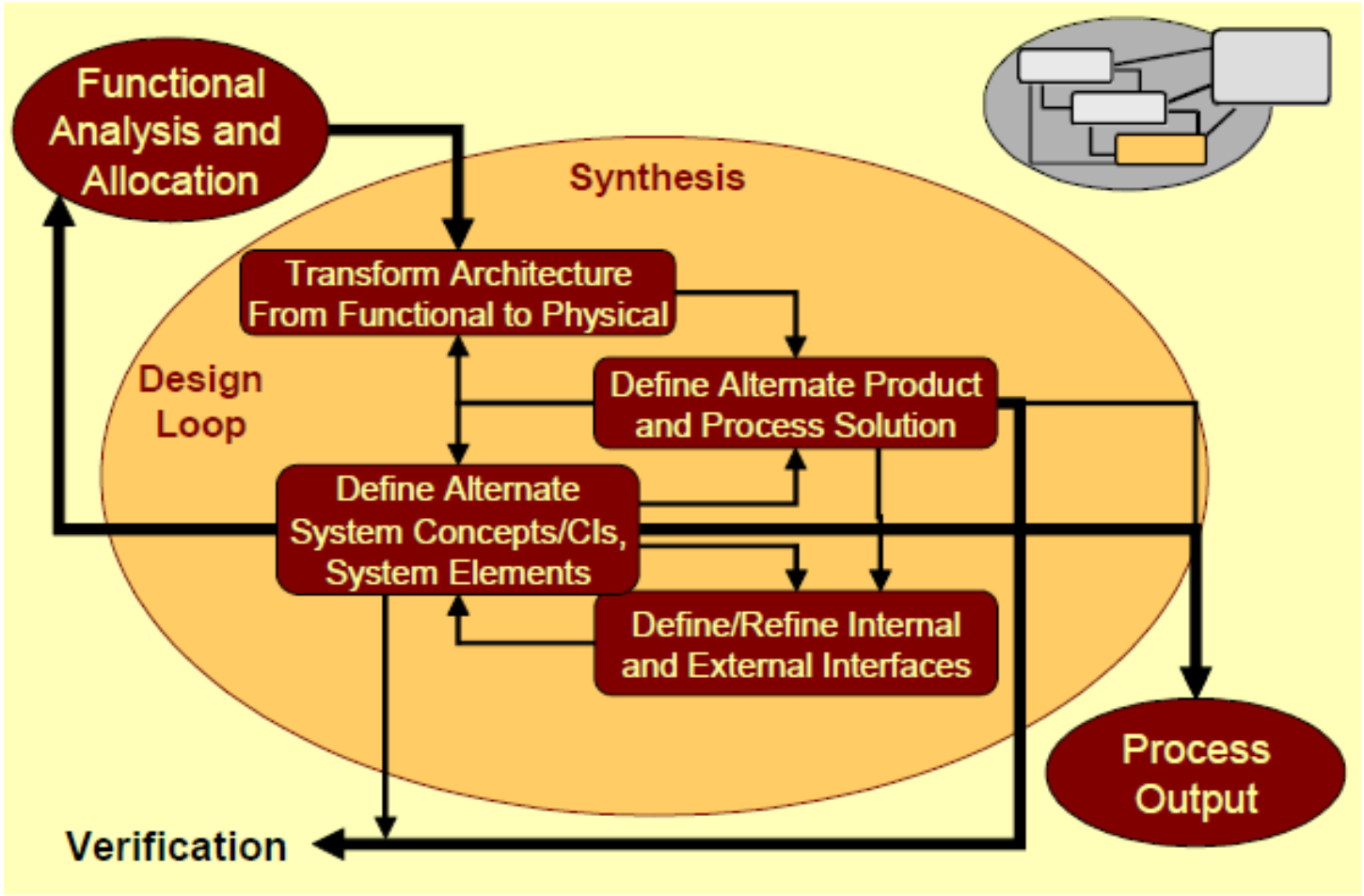


(exemplu)

Ingineria proiectării – Sinteza

- În faza de sinteză se realizează proiectul **setului de dispozitive** care produc comportamentul dorit. La nivelul de decompoziție cel mai detaliat, dispozitivul este o componentă elementară cu un proiect deja cunoscut.
  - În cazul software-ului, nivelul de decompoziție cel mai detaliat îl reprezintă specificația rutinei care îndeplinește funcția dorită
  - La toate celelalte niveluri / high-level, procesul de sinteză proiectează **interoperarea** dintre dispozitive pentru a produce comportamentul dorit

Ingineria proiectării – Sinteza

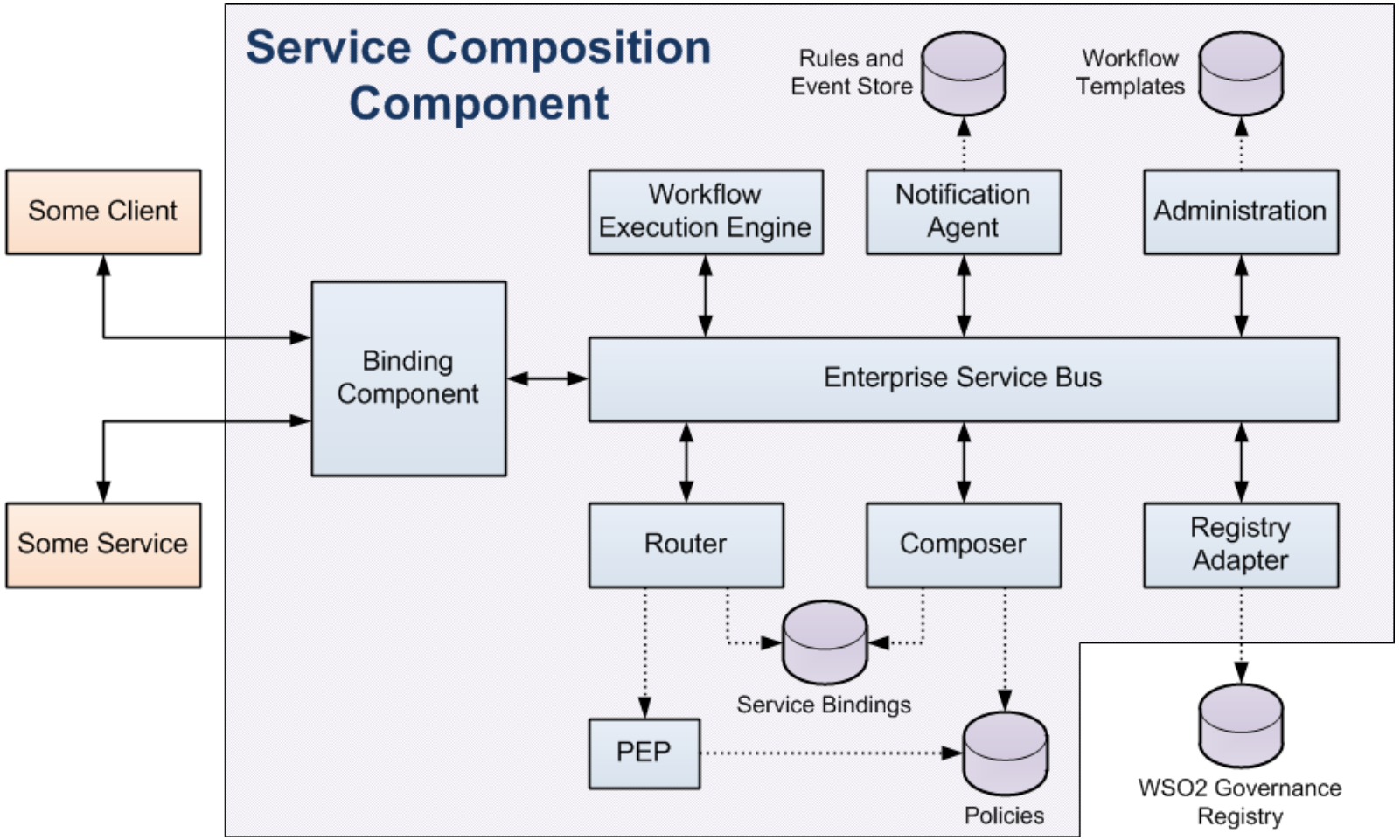


(exemplu)

Ingineria proiectării – Compoziția

- În faza de compoziție se identifică posibilitatea de a îngloba mai multe sub-comportamente într-un singur dispozitiv, de multe ori prin **combinarea elementelor** din două sau mai multe proiecte de dispozitive.
  - Se ajunge astfel la un proiect al **unui singur dispozitiv** care înglobează mai multe sub-comportamente. În multe cazuri, aceste sub-comportamente vor fi părți din diferite comportamente sau funcții, de nivel superior

Ingineria proiectării – Compoziția

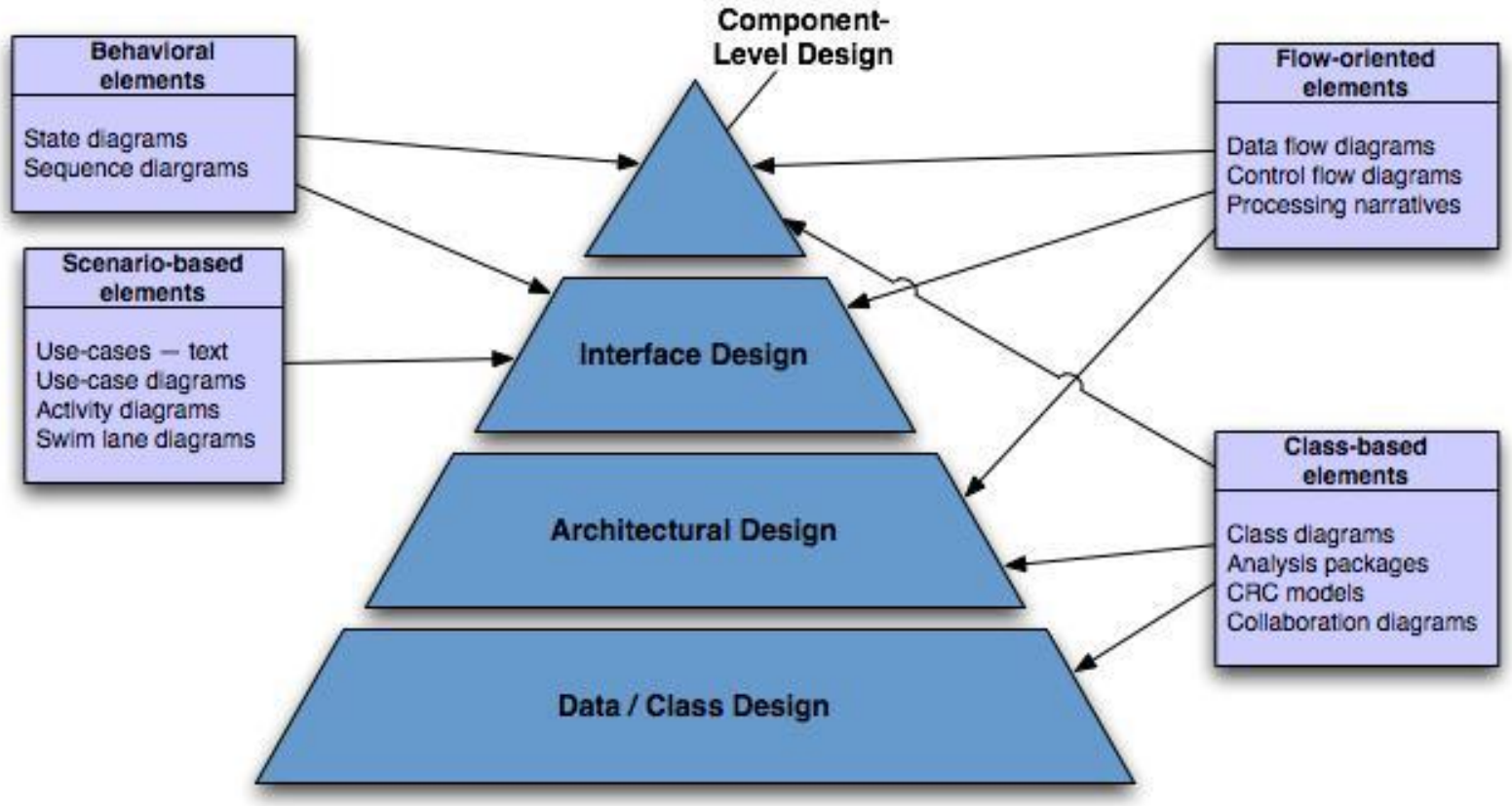


(exemplu)

Ingineria proiectării – Analiza

- În această fază **se testează** modul în care proiectul răspunde cerințelor de natură funcțională și non-funcțională.
  - Se elimină variantele de proiectare care nu îndeplinesc cerințele. Se analizează și situații speciale care se pot soluționa prin compromis.
  - Rezultatul nu este întotdeauna o soluție unică. Poate rezulta un set de proiecte admisibile care pot fi luate în considerare pentru a sintetiza un dispozitiv compus.
  - În unele cazuri, nu rezultă nicio soluție, fiind necesară rafinarea / corectarea cerințelor pentru unele dintre sub-comportamente, sau este nevoie de **reconsiderarea descompunerii comportamentului**.

Ingineria proiectării – Analiza



(exemplu)

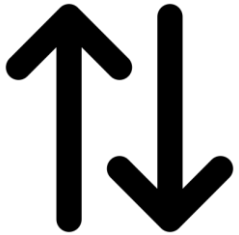
Ingineria proiectării

Sumar

- Decompoziția
  - descompunerea funcționalităților în sub-comportamente (proces – activități)
  - atribuirea activităților unor dispozitive (resurse)
  - implementarea comportamentelor (roluri)
- Sinteza: identificarea resurselor și adaptarea acestora la roluri
- Analiza: identificarea constrângerilor legate de interacțiune, conflicte și compromisuri (vezi curs – aspecte de integrare)
- Compoziția: proiectarea unui (singur) sistem care să înglobeze mai multe funcțiuni / comportamente ale unor subsisteme

Ingineria proiectării

- Ordinea acestor faze (**top-down / bottom-up**) precum și alte decizii de proiectare depind de domeniul specific produsului / comportamentelor și de expertiza inginerului
- Acestea pot fi impuse și de practicile curente ale organizației
- Reutilizarea componentelor poate fi privită atât ca
  - parte a sintezei
  - parte a compoziției
  - sau în analiză, ca opțiune de luat în calcul



Ingineria proiectării – concepte

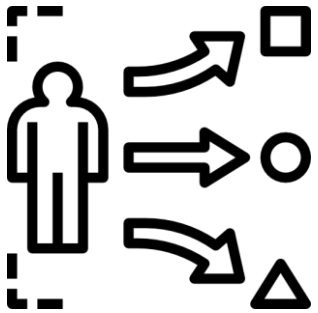
- **Abstractizare** – date, proceduri, control
- **Arhitectură** – structura generală a software-ului
- **Model (pattern)** – redarea esenței unei soluții de proiectare
- **Modularitate** – compartimentarea datelor si funcțiilor
- **Ascunderea de informații** – interfețe controlate
- **Independența funcțională**
- **Refactorizare** – îmbunătățirea proiectului fără afectarea comportamentului

Ingineria proiectării – principii

- **uniformitate** și integrare
- **modularizare**
- **structurare** care să permită modificări facile
- **implementarea** explicită a tuturor cerințelor
- **documentație** ușor de înțeles
- **imagine de ansamblu** completă asupra sistemului software

Ingineria proiectării – bune practici

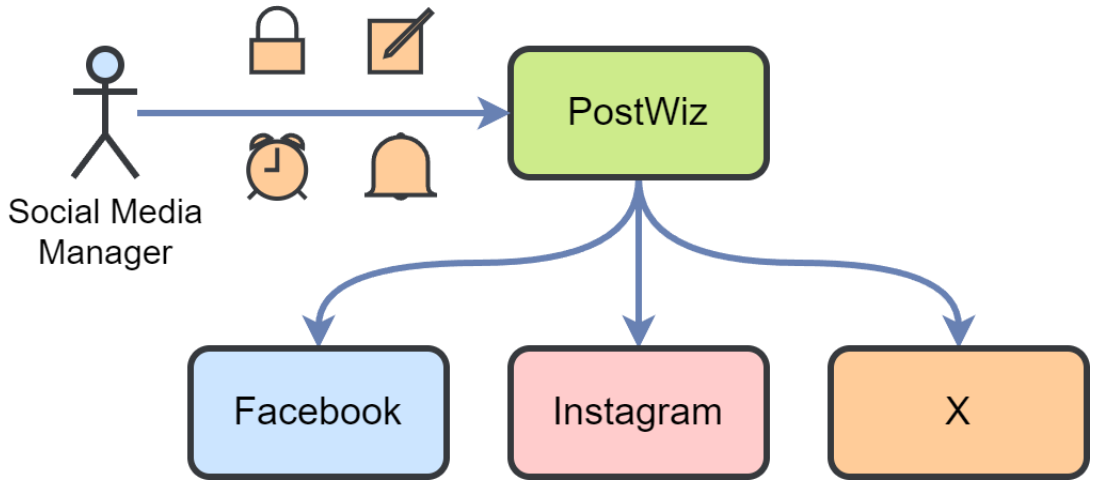
- Ar trebui să fie luate în considerare mai multe **alternative de proiectare**, inclusiv compromisuri – prima soluție viabilă nu este neapărat cea mai bună
- Proiectele trebuie testate **relativ la cerințe** (și alte considerente), iar testele se pot face la mai multe niveluri de decompoziție, dacă cerințele sunt rafinate în mod corespunzător



Studiu de caz

Concept: Platforma “PostWiz” își propune să integreze diverse sisteme de social media prin API-uri, să gestioneze eficient datele utilizatorilor și să ofere o experiență de utilizare unitară. Care ar fi nivelurile de integrare vizate de următoarele cazuri de utilizare:

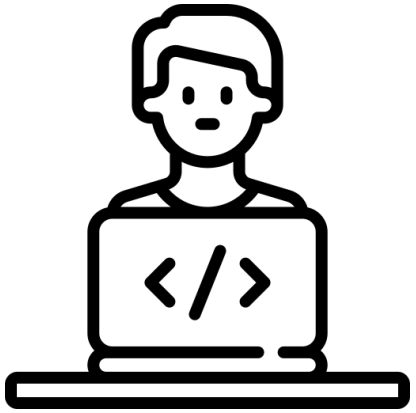
- Autentificare și gestionare conturi
- Editarea conținutului postărilor
- Programarea postărilor
- Centralizarea notificărilor



Sursa: M. Marica, Platformă de gestionare și planificare a postărilor din social media, lucrare de licență, 2024

Aplicații compozite (CA)

- Dezvoltare de software din perspectivă inginerească
  - Valorificarea tehnologiilor, instrumentelor metodelor și dispozitivelor într-un cadru organizat → framework (tehnic si organizatoric)
  - Principii de bază – deschidere, interoperabilitate, performanță și scalabilitate



TRUST ME I'M AN ENGINEER

Aplicații compozite – IEEE 1471-2000

Recomandări de bune practici pentru descrierea arhitecturilor sistemelor software

Contribuții

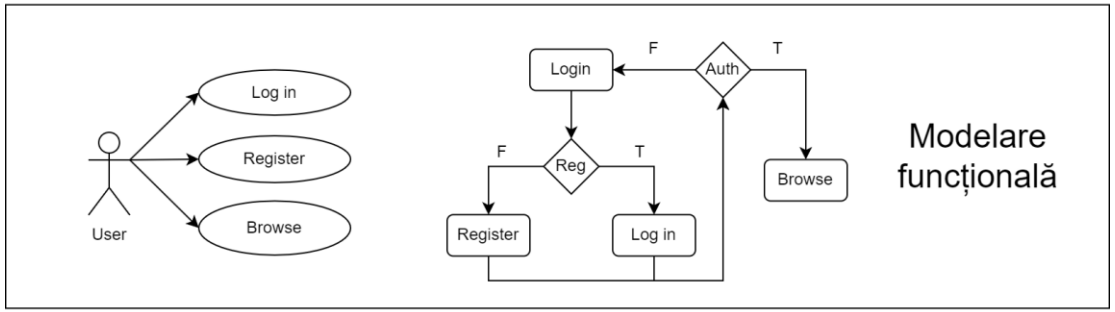
- Standarde pentru descrierea arhitecturii sistemelor
- Integrarea cerințelor definite de stakeholderi
- Utilizarea mai multor perspective pentru modelarea arhitecturii sistemelor

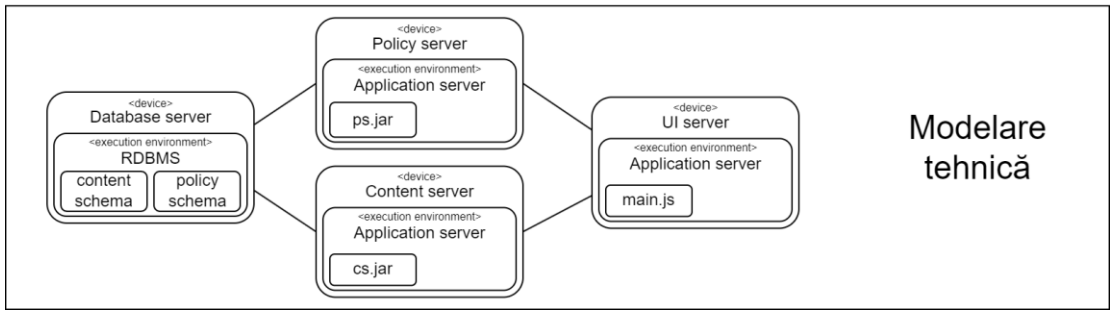
Aplicații compozite

Nivelul de modelare	'50	'60	'70	'80	'90	'2000	'2010
Modelare abstractă					ARIS UML	DSL	Aplicații compozite
Workflow					Workflow Reference Model	BPMN	
Orientare pe servicii						BPEL4WS Servicii Web	
Orientare pe componente					COM/DCOM	J2EE	
Orientare pe obiect		Simula	Smalltalk	C++, CORBA	Java		
Procedural	Fortran, Algol		Pascal, C				
Limbaj mașină	assembler						
	Timp						

Aplicații compozite (CA)

- CA = noțiune integratoare pentru toate principiile moderne de dezvoltare de software în medii distribuite
- Presupune introducerea diferitelor niveluri de abstractizare tehnică, respectiv a modelelor, prin prisma a 3 perspective de bază:
  - Nivelul **modelării logice** a sistemului
  - Nivelul **modelării funcționale**
  - Nivelul **modelării tehnice** a sistemului





Aplicații compozite – aspecte de modelare

GUI

- **Nivel Prezentare**
- Cuprinde toate interacțiunile cu utilizatorul
  - Conținutul paginilor
  - Succesiunea paginilor

Aplicații compozite – aspecte de modelare

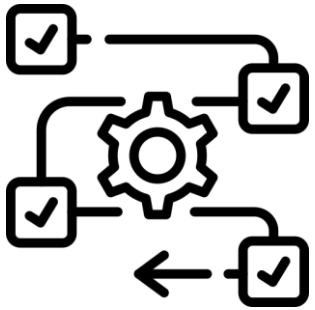
Procese

- **Nivel Aplicație**
- Au rol important la nivelul modelării – reprezintă descrierea funcționalităților viitorului sistem
- Metode de modelare
  - Descrieri liniare de proces
  - Descrierea evenimentelor

Aplicații compozite – aspecte de modelare

Workflow

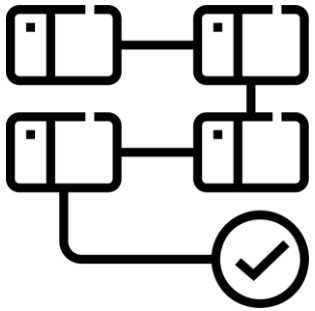
- **Nivel Aplicație**
- Constituie un ajutor în implementarea tehnică a proceselor de afaceri specifice
- Detaliază modelul tranzacțiilor
- Se pot reprezenta prin **diagrame de activități sau de secvență** – care descriu activități și condiții



Aplicații compozite – aspecte de modelare

Componente

- **Nivel Domeniu/Infrastructură**
- În aplicații de tip SOA, serviciile sunt componente
- O colecție de componente care depind funcțional una de alta determină un domeniu funcțional
- Fac parte din logica domeniului



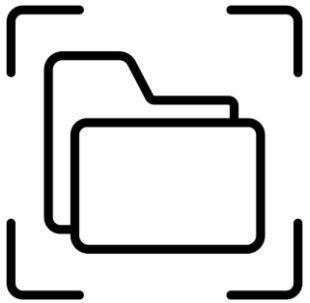
Aplicații compozite – aspecte de modelare

Reguli

- **Nivel Domeniu**
- Parte din logica activităților și a proceselor

Date

- **Nivel Domeniu**
- Modelarea datelor



Aplicații compozite – aspecte de modelare

Grupuri și roluri

- **Nivel Infrastructură**
- Managementul identității și accesului la sistem
- Gruparea utilizatorilor pe roluri permite accesul diferențiat la funcționalități

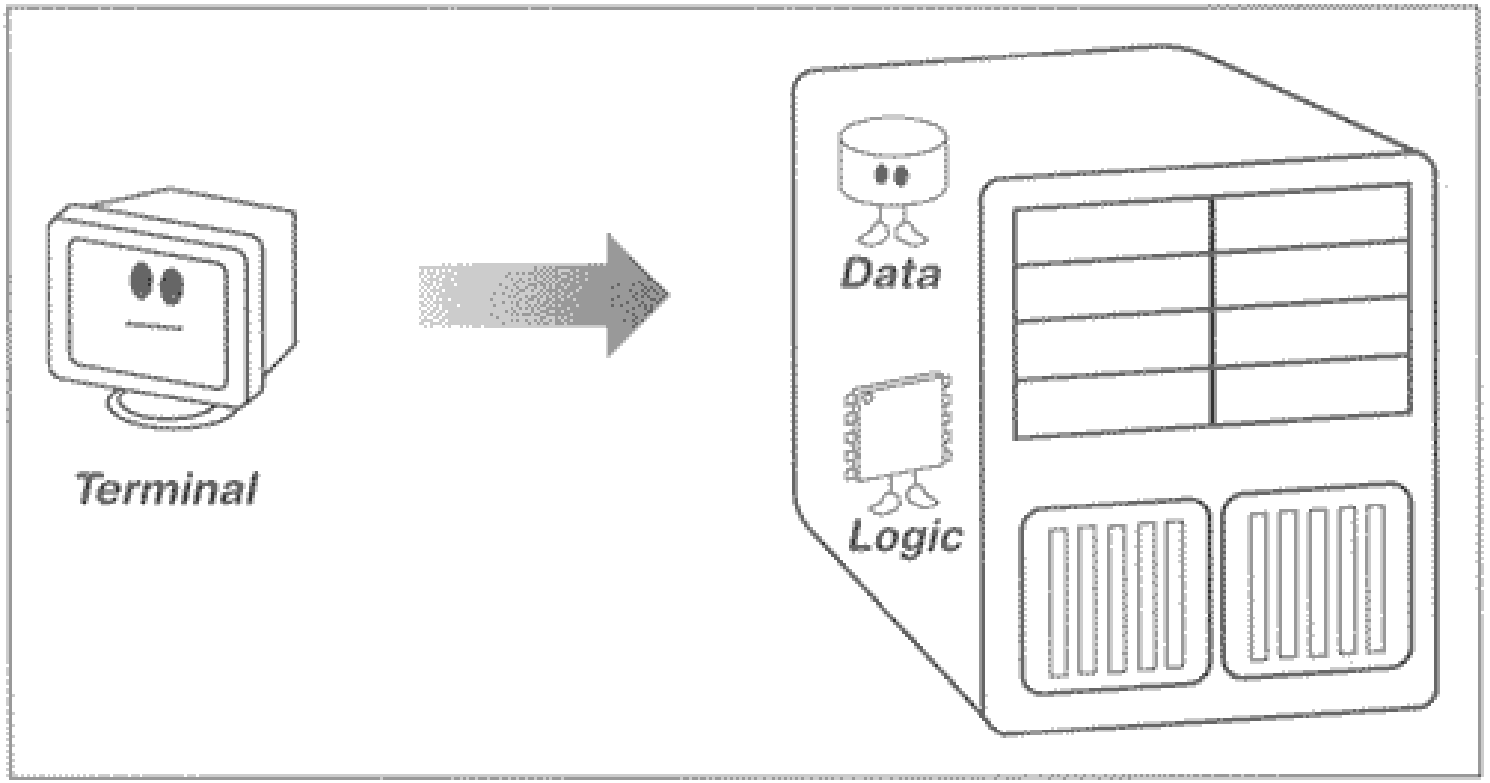
Modelarea aplicațiilor compozite – Arhitecturi

Tipuri de arhitecturi

- Arhitectura centralizată
- Arhitectura “two-tier”
- Arhitectura “three-tier”
- SOA (Service-Oriented Architecture)
- Arhitectura bazată pe microservicii
- Arhitecturi serverless

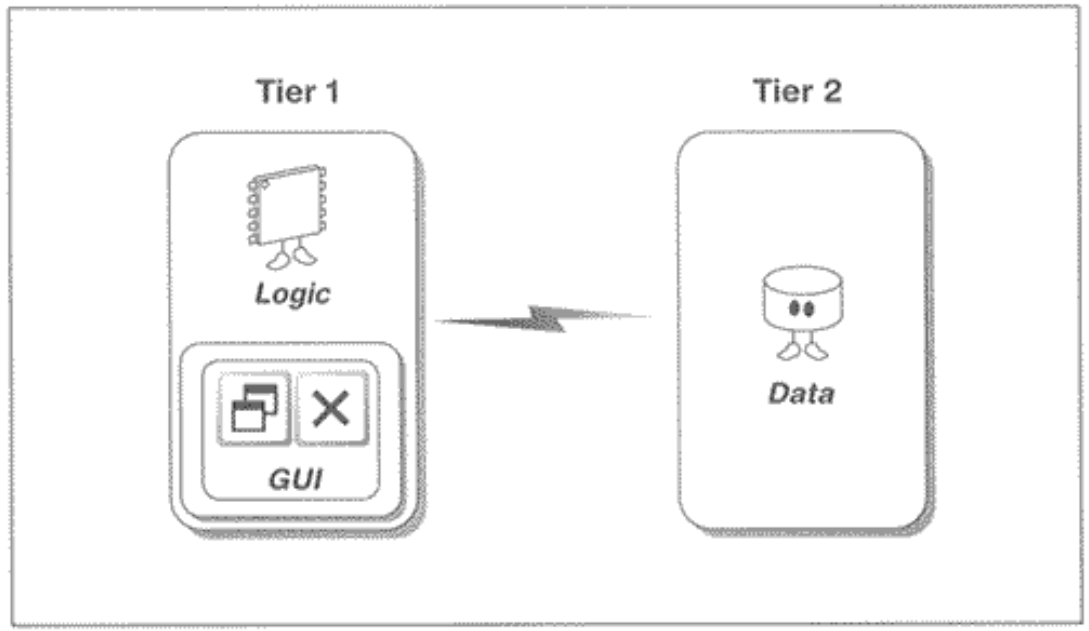
Image by Freepik

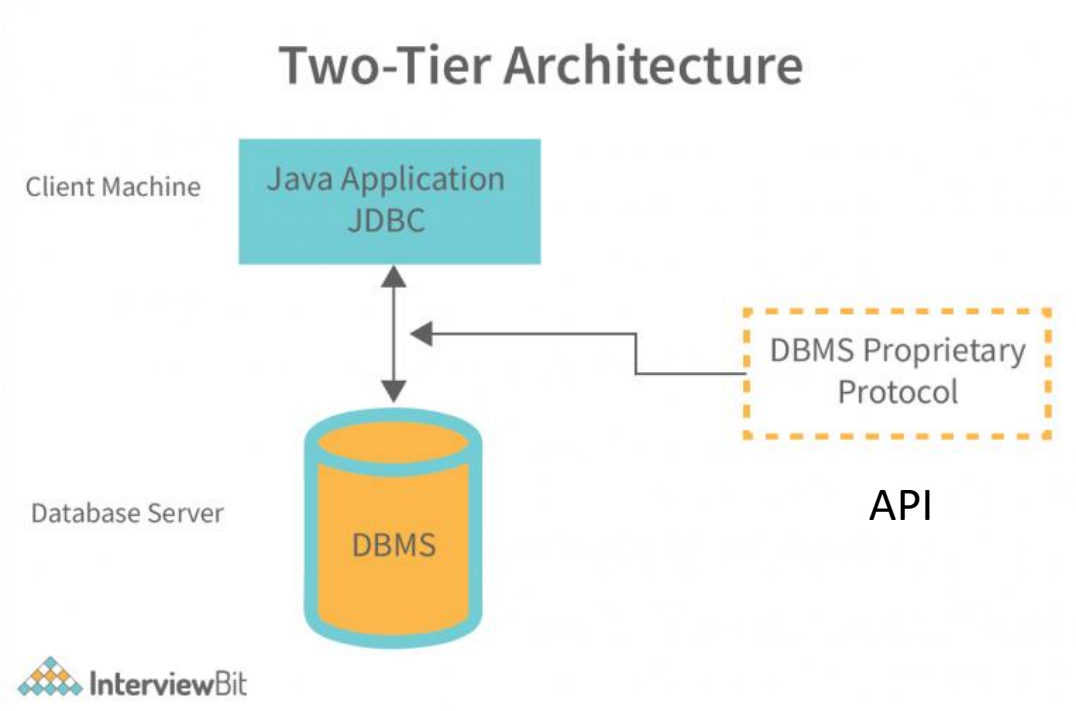
Arhitectura centralizată



Arhitectura “two-tier”

- Arhitectura pe 2 niveluri

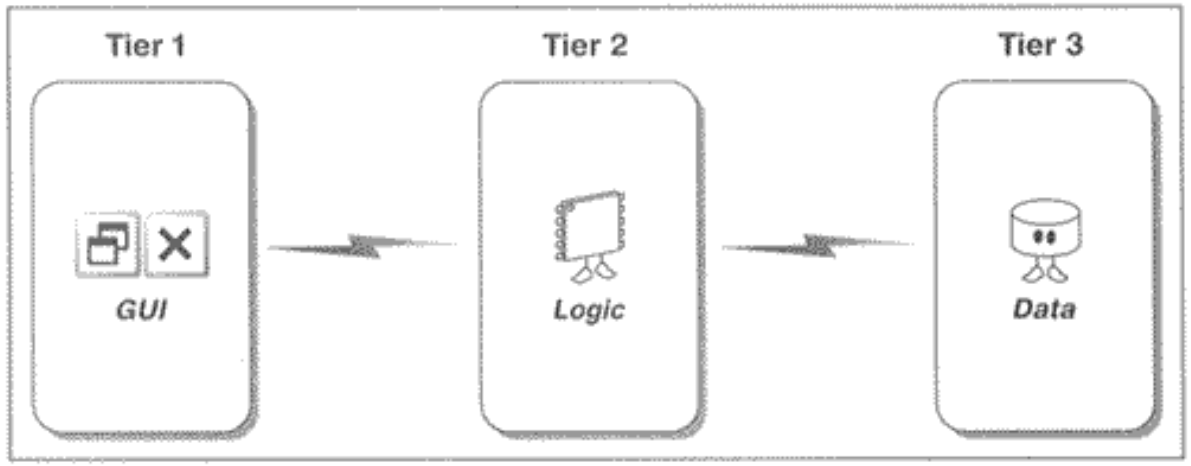


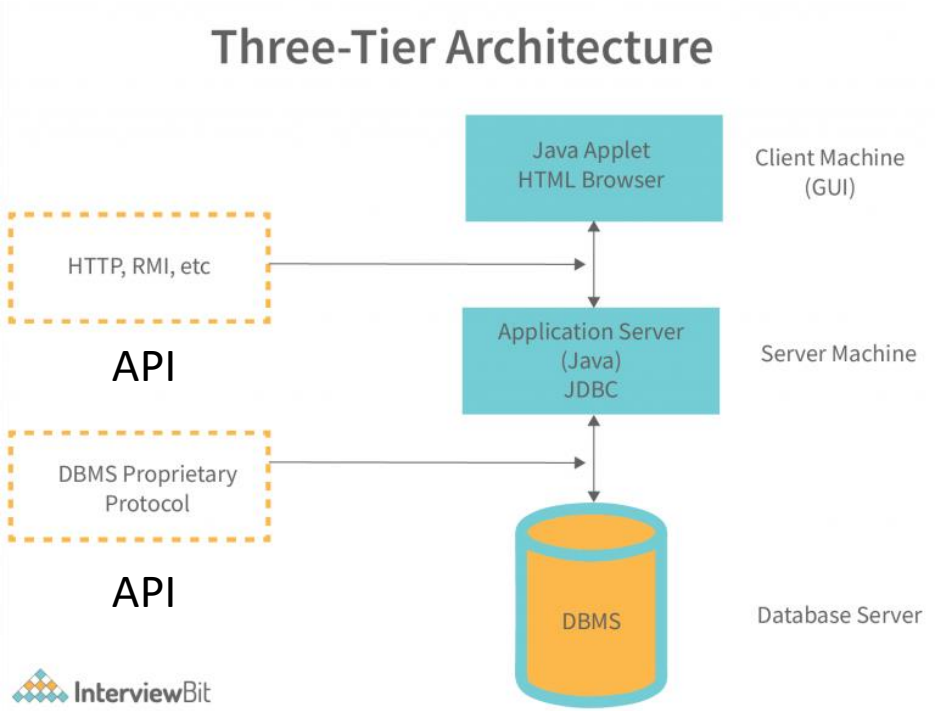


Sursa: JDBC Architecture, InterviewBit, 2023

Arhitectura “three-tier”

- Arhitectura pe 3 niveluri

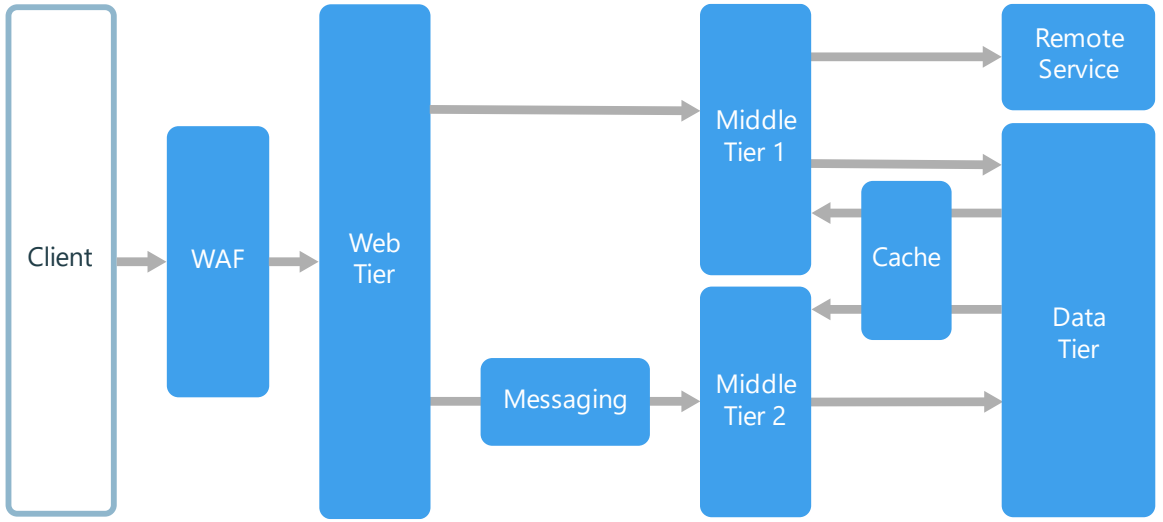




Sursa: JDBC Architecture, InterviewBit, 2023

Arhitectura N-Tier

- Dezvoltări ulterioare au dus la N niveluri (N-Tier)
  - Aplicații compozite
  - Fiecare nivel poate corespunde unor funcționalități sau servicii specifice
- Exemplu (4-Tier)
  - Presentation/User interface
  - Application/Business logic
  - Data Management
  - Infrastructure/Integration

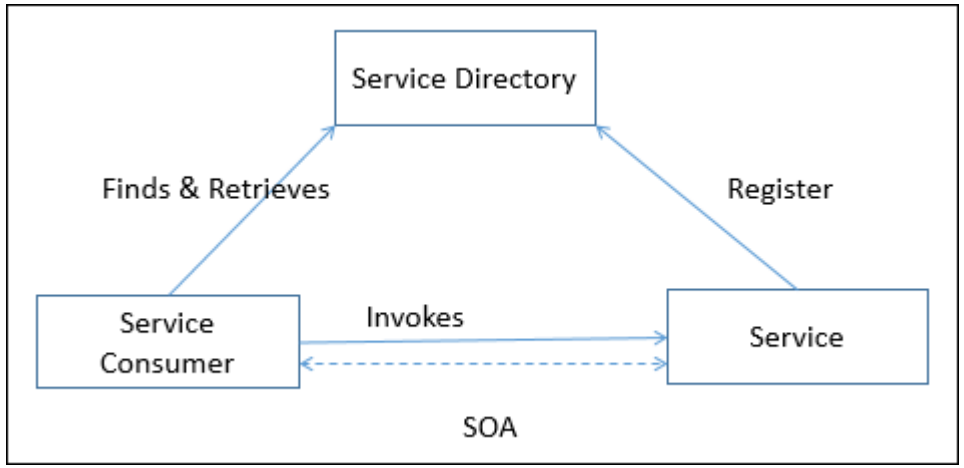


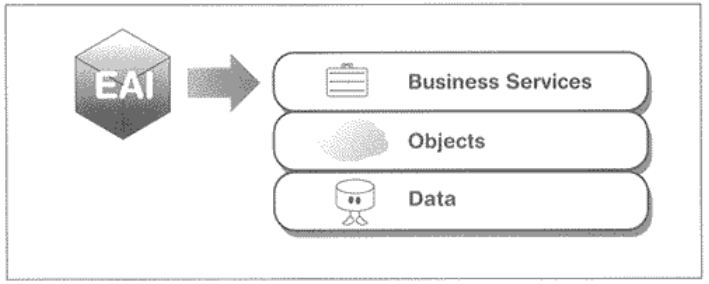
Sursa: Azure Architecture Center, Microsoft

Sursa: Azure Architecture Center, Microsoft

Service Oriented Architecture (SOA)

- Servicii
  - Componente independente cu funcționalități bine definite
- Arhitectură client-server orientată pe servicii
  - Servicii software
  - Consumatori de servicii
  - Registru de servicii
- Caracteristici
  - Arhitectură distribuită
  - Compoziție
  - Interoperabilitate
  - Reutilizabilitate

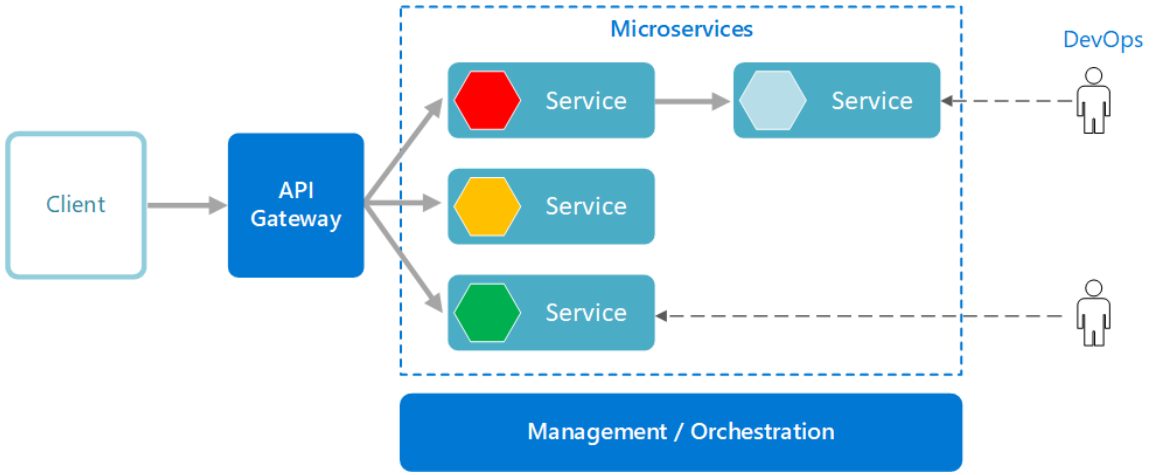




Sursa: Distributed Architecture, tutorialspoint

Arhitectura sistemelor – Microservicii

- Componente autonome, independente, care implementează o funcționalitate bine delimitată
- Pot fi dezvoltate și publicate independent
  - Sunt responsabile de nivelul **propriu** de persistență
  - Se interfațează și sunt accesibile printr-un **API Gateway**



Sursa: Azure Architecture Center, Microsoft

Arhitectura sistemelor

Arhitect de sistem?

- Cunoaște domeniul de modelare
- Este familiar cu multiple tehnologii, framework-uri, platforme
- Ia decizii privind arhitectura generală a sistemului
- Urmărește implementarea conformă cu modelul arhitectural
- Analiza continuă a arhitecturii și a tendințelor actuale
- Analiza tendințelor în industria de dezvoltare software
- Soft skills – comunicare, lucru în echipă, coordonare, facilitare

Sursa: B. Magri, [Summary — Chap 1] Fundamentals of Software Architecture, medium

Întrebări?