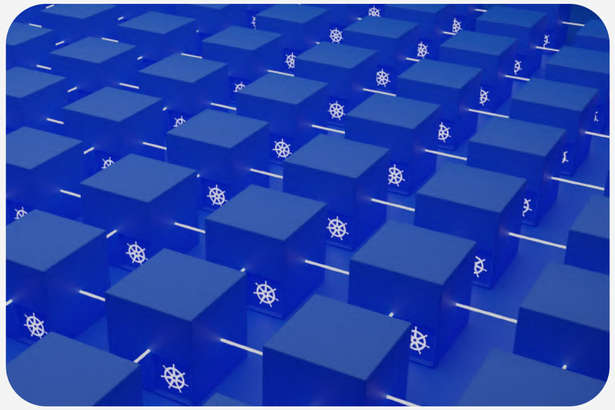
Journey into Microservices



Cuprins

Ce sunt microserviciile și cum se diferențiază de monolit

Cuprins

Cum definim microserviciile & granularitate
Business Capability în proiectul nostru

Cuprins

Topologiile Broker și Mediator

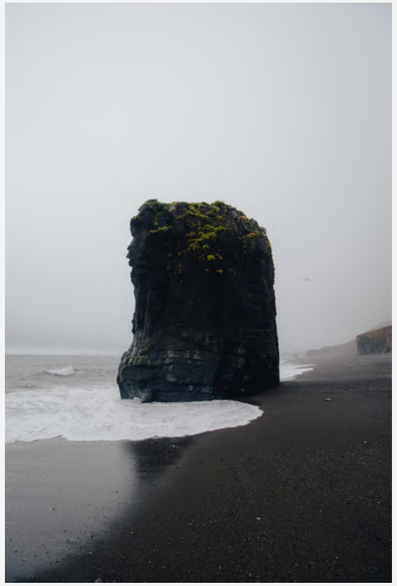
Cuprins

Modalități de comunicare între microservicii
Prevenirea pierderii datelor

Cuprins

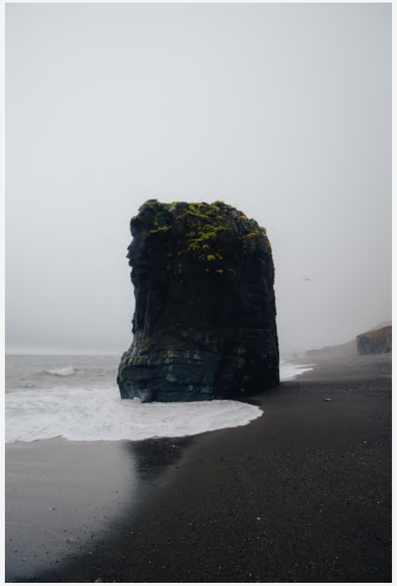
Monolith vs Microservices – Ce sunt microserviciile și cum se diferențiază de monolit

Un monolit este un tip de arhitectură software în care întreaga aplicație este construită ca o singură unitate.



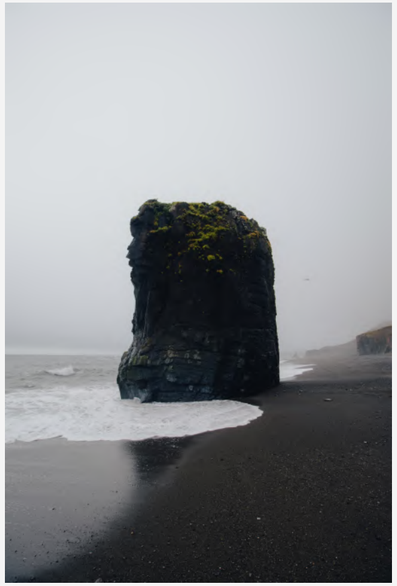
Monolith vs Microservices – Ce sunt microserviciile și cum se diferențiază de monolit

componentele și funcționalitățile sunt strâns legate între ele



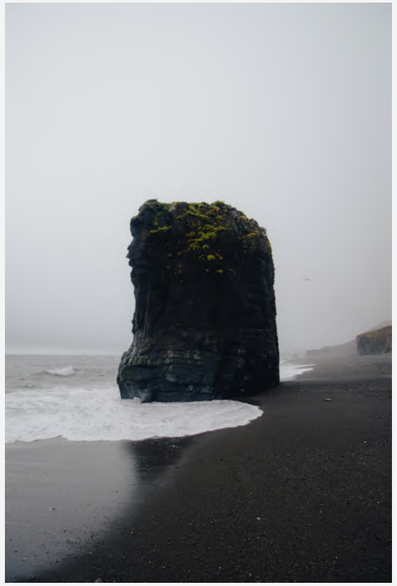
Monolith vs Microservices – Ce sunt microserviciile și cum se diferențiază de monolit

Dezvoltare și implementare unitară



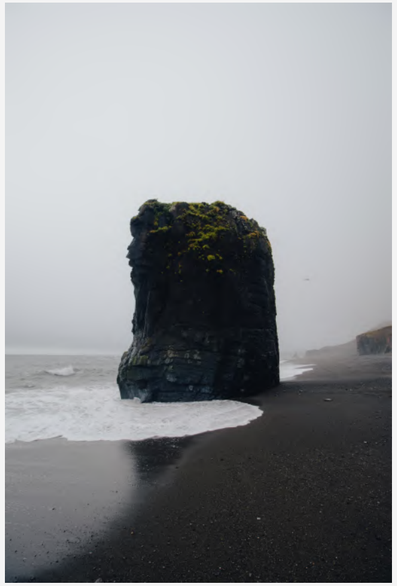
Monolith vs Microservices – Ce sunt microserviciile și cum se diferențiază de monolit

- Dezvoltare și implementare unitară
- Comunicare interna directa



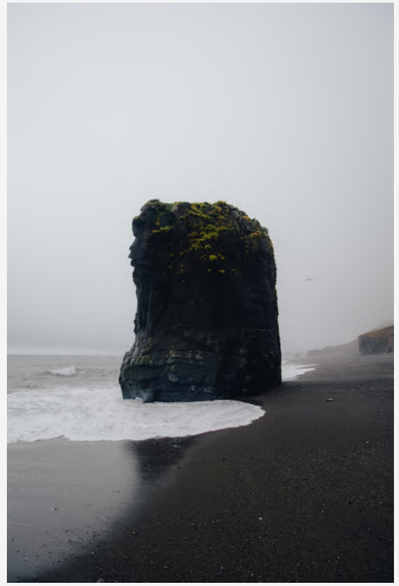
Monolith vs Microservices – Ce sunt microserviciile și cum se diferențiază de monolit

- Dezvoltare și implementare unitară
- Comunicare interna directă
- Potențial pentru complexitate



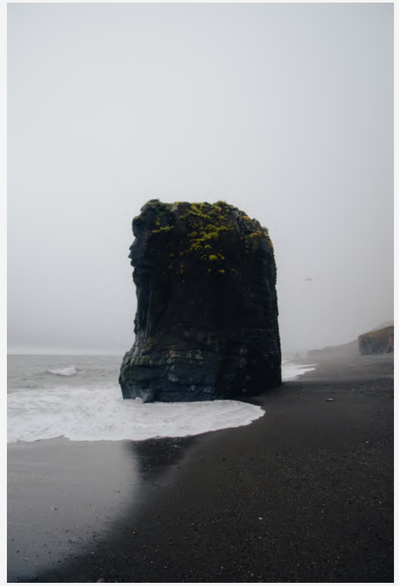
Monolith vs Microservices – Ce sunt microserviciile și cum se diferențiază de monolit

- Dezvoltare și implementare unitară
- Comunicare interna directa
- Potențial pentru complexitate
- Gestionare centralizată



Monolith vs Microservices – Ce sunt microserviciile și cum se diferențiază de monolit

- Dezvoltare și implementare unitară
- Comunicare interna directa
- Potențial pentru complexitate
- Gestionare centralizată
- Scalare verticală



Monolith vs Microservices – Ce sunt microserviciile și cum se diferențiază de monolit

Microserviciile sunt o abordare arhitecturală în dezvoltarea de software în care o aplicație este structurată ca o colecție de servicii mici.

Monolith vs Microservices – Ce sunt microserviciile și cum se diferențiază de monolit

independente și ușor de întreținut

Monolith vs Microservices – Ce sunt microserviciile și cum se diferențiază de monolit

independente și ușor de întreținut
fiecare rulează în propriul proces

Monolith vs Microservices – Ce sunt microserviciile și cum se diferențiază de monolit

independente și ușor de întreținut
fiecare rulează în propriul proces
comunică cu celelalte prin mecanisme ușoare

Monolith vs Microservices – Ce sunt microserviciile și cum se diferențiază de monolit

independente și ușor de întreținut
fiecare rulează în propriul proces
comunică cu celelalte prin mecanisme ușoare
pot fi desfășurate independent

Monolith vs Microservices – Ce sunt microserviciile și cum se diferențiază de monolit

independente și ușor de întreținut
fiecare rulează în propriul proces
comunică cu celelalte prin mecanisme ușoare
pot fi desfășurate independent
sunt scalabile individual

Monolith vs Microservices – Ce sunt microserviciile și cum se diferențiază de monolit

Independență

Monolith vs Microservices – Ce sunt microserviciile și cum se diferențiază de monolit

- Independență
- Separare a responsabilităților

Monolith vs Microservices – Ce sunt microserviciile și cum se diferențiază de monolit

- Independență
- Separare a responsabilităților
- Isolare și scalabilitate individuală

Monolith vs Microservices – Ce sunt microserviciile și cum se diferențiază de monolit

- Independență
- Separare a responsabilităților
- Isolare și scalabilitate individuală
- Tehnologii variate

Monolith vs Microservices – Ce sunt microserviciile și cum se diferențiază de monolit

- Independență
- Separare a responsabilităților
- Isolare și scalabilitate individuală
- Tehnologii variate
- Gestionarea datelor proprii

Monolith vs Microservices – Ce sunt microserviciile și cum se diferențiază de monolit

- Independență
- Separare a responsabilităților
- Isolare și scalabilitate individuală
- Tehnologii variate
- Gestionarea datelor proprii
- Reziliență și gestionare a erorilor

Monolith vs Microservices – Ce sunt microserviciile și cum se diferențiază de monolit

- Independență
- Separare a responsabilităților
- Isolare și scalabilitate individuală
- Tehnologii variate
- Gestionarea datelor proprii
- Reziliență și gestionare a erorilor
- Monitorizare și gestionare centralizată

Monolith vs Microservices – Ce sunt microserviciile și cum se diferențiază de monolit

- Independență
- Separare a responsabilităților
- Isolare și scalabilitate individuală
- Tehnologii variate
- Gestionarea datelor proprii
- Reziliență și gestionare a erorilor
- Monitorizare și gestionare centralizată
- Dezvoltare și livrare continuă

Descompunerea în Microservicii - Cum definim microserviciile & granularitate

Descompunerea în Microservicii - Cum definim microserviciile & granularitate

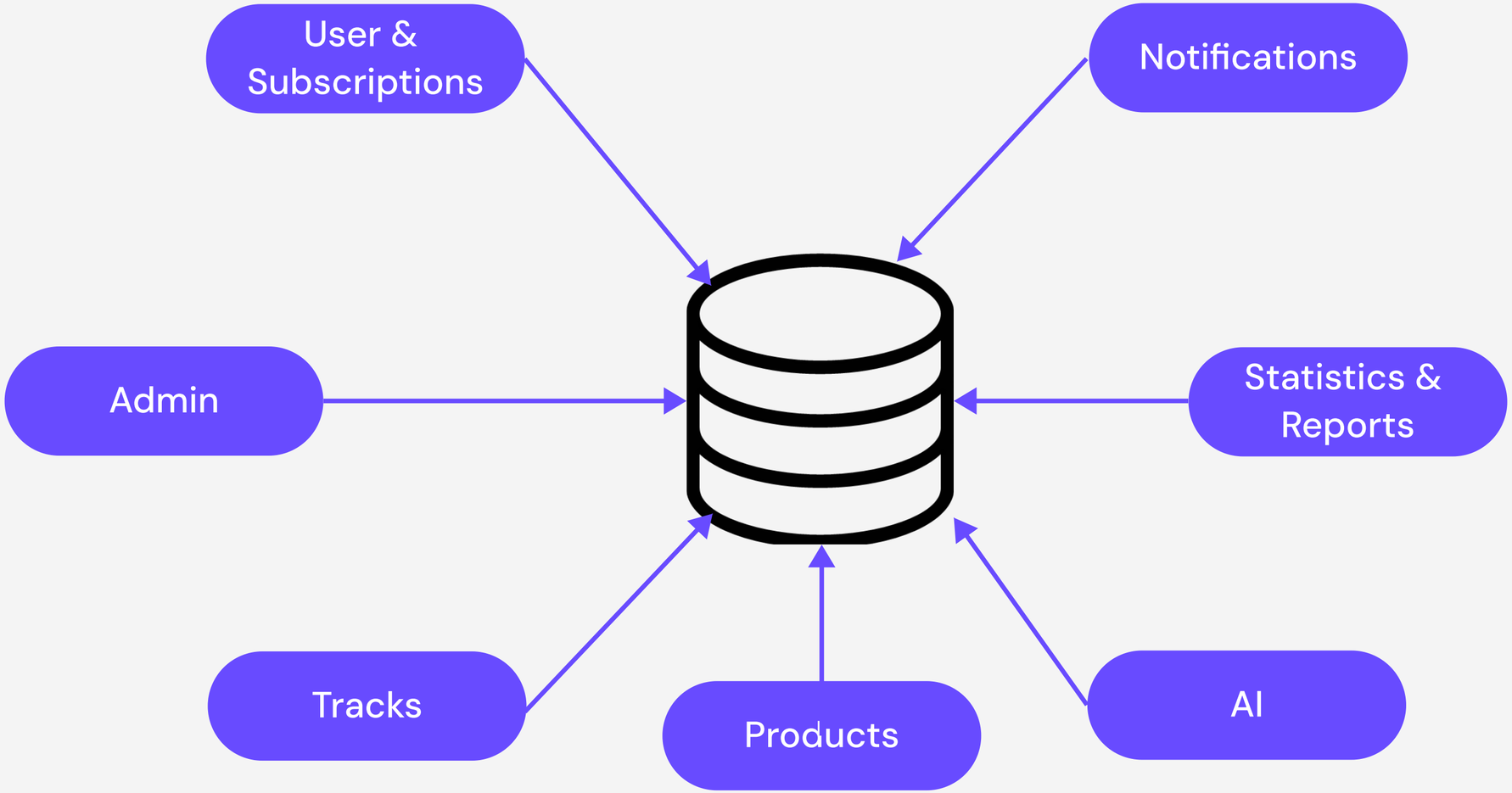
"A Big Ball of Mud is a haphazardly structured, sprawling, sloppy, duct-tape-and-baling-wire, spaghetti-code jungle. These systems show unmistakable signs of unregulated growth, and repeated, expedient repair. Information is shared promiscuously among distant elements of the system, often to the point where nearly all the important infor- mation becomes global or duplicated.

The overall structure of the system may never have been well defined.
If it was, it may have eroded beyond recognition. Programmers with a shred of architectural sensibility shun these quagmires. Only those who are unconcerned about architecture, and, perhaps, are comfortable with the inertia of the day-to-day chore of patching the holes in these failing dikes, are content to work on such systems."

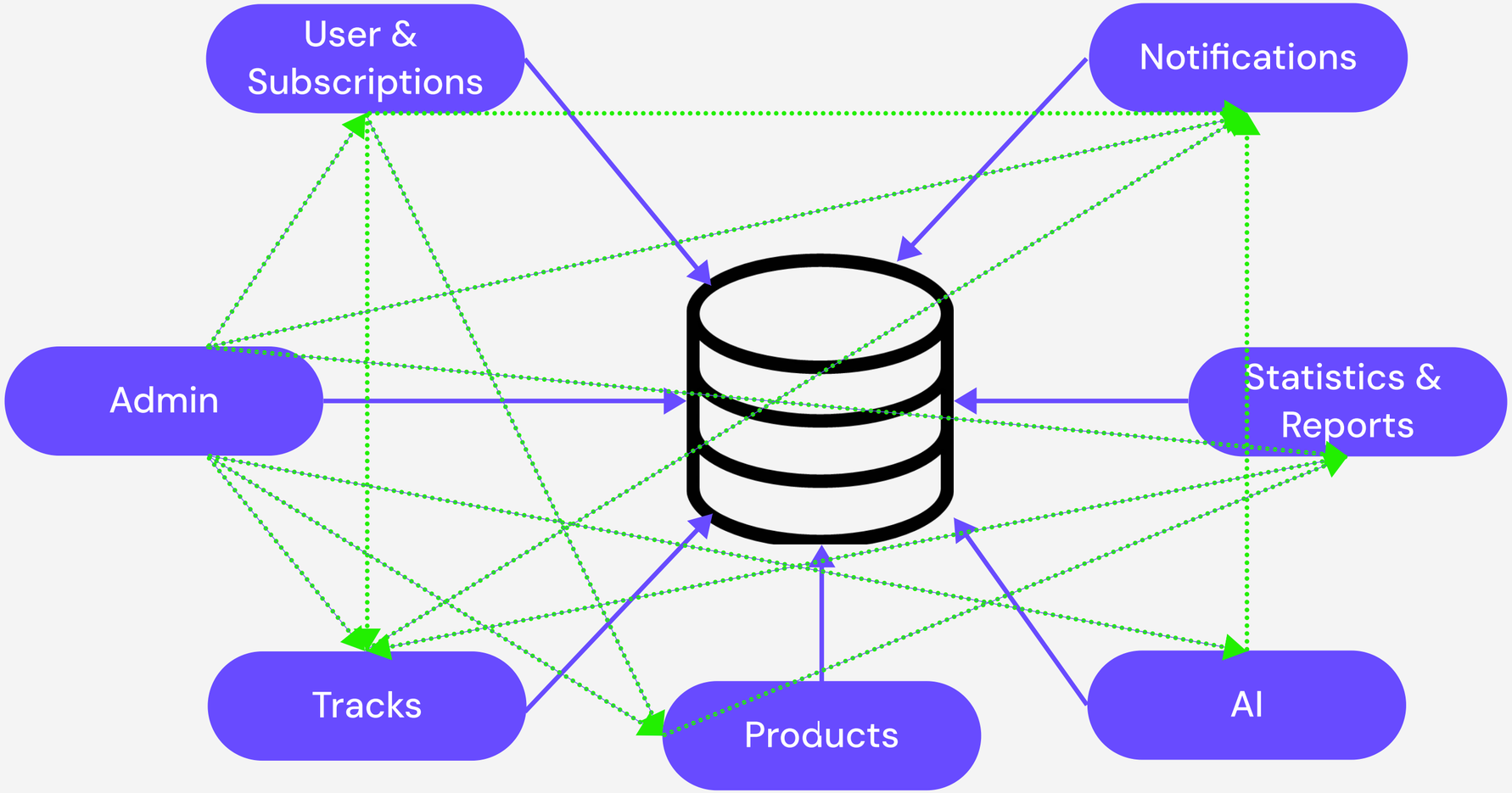
—Brian Foote and Joseph Yoder, 1997

Black Friday Deal Tracker

Black Friday Deal Tracker



Black Friday Deal Tracker



Cât de mic trebuie sa fie un microserviciu?

Cum putem defini limitele corecte ale unui microserviciu în ceea ce privește funcționalitatea sa?

Ce beneficii și dezavantaje pot apărea atunci când microserviciile sunt prea mici sau prea mari?

Cât de multe microservicii putem avea?

Granularity desintegrators vs granularity integrators

Nu orice parte din sistem trebuie sa fie un microserviciu.

Granularity desintegrators

Decomposition by Business Capability
- se concentrează pe funcționalități de afaceri

Granularity desintegrators

Decomposition by Business Capability
discutie pe exemplu

Granularity desintegrators

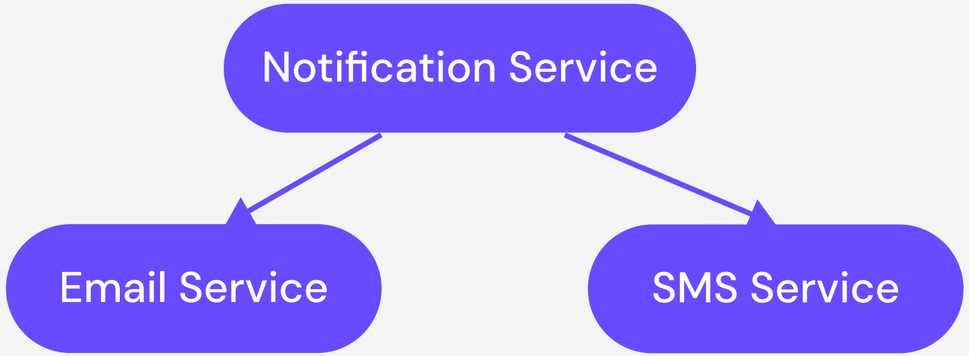
Decomposition by Subdomain
- se bazează pe cunoașterea și activitățile din cadrul domeniului

Granularity desintegrators

Decomposition by Subdomain
discutie pe exemplu

Granularity desintegrators

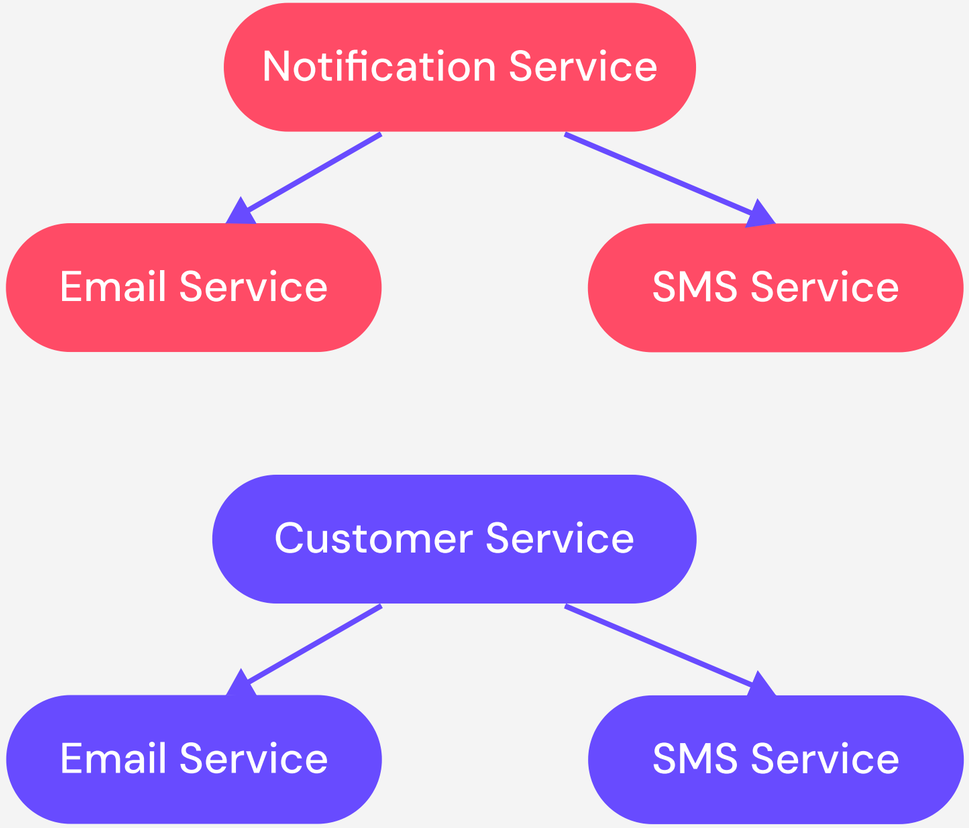
Aplicabilitate



Serviciul face prea multe lucruri care nu au legătură?

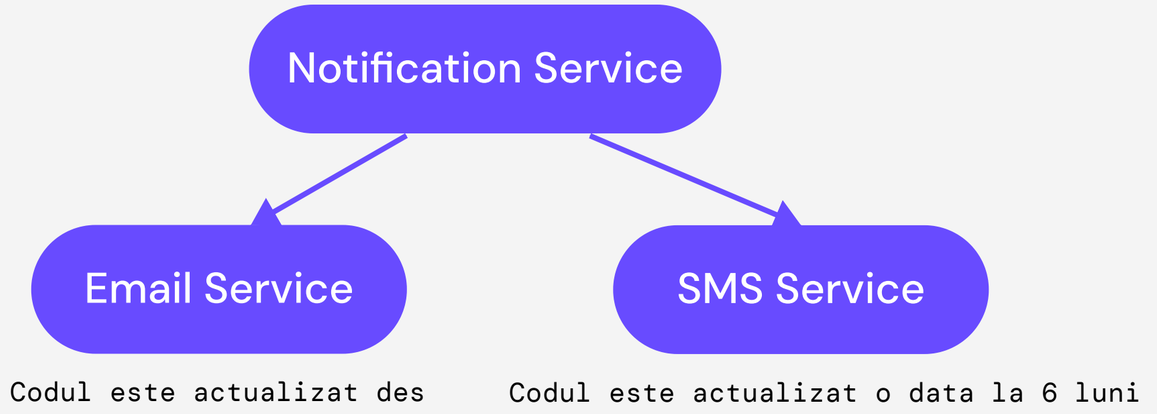
Granularity desintegrators

Aplicabilitate



Granularity desintegrators

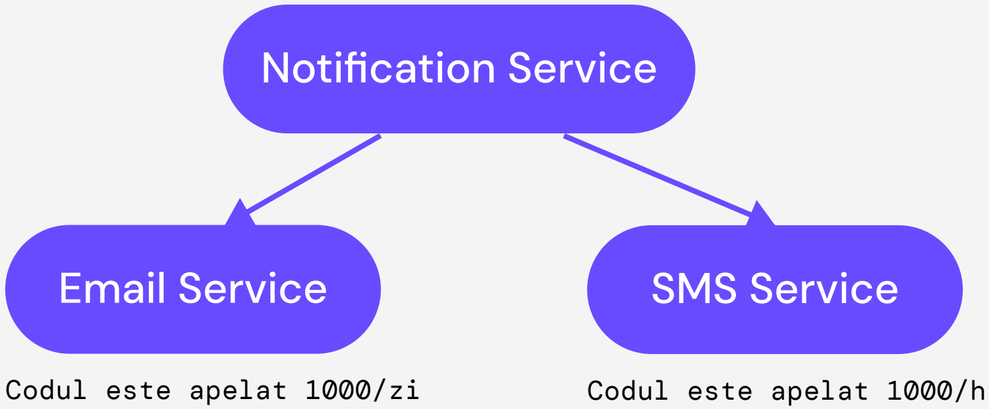
Aplicabilitate
Volatilitate



Modificările sunt izolate doar pentru o singură parte a serviciului?

Granularity desintegrators

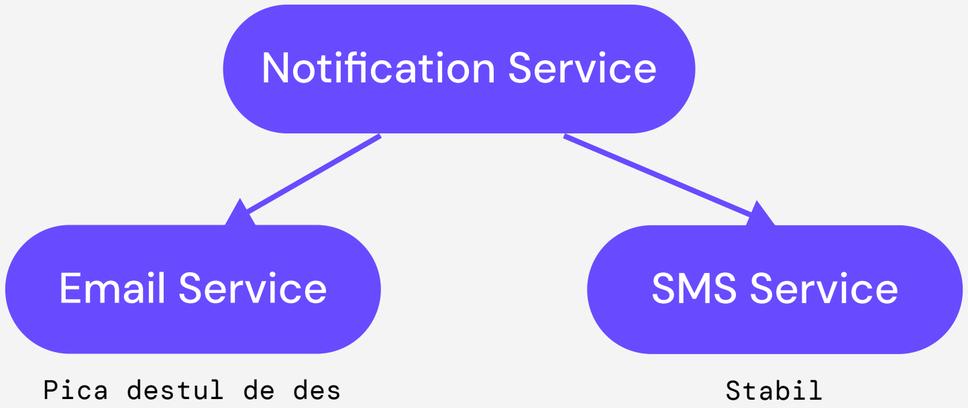
Aplicabilitate
Volatilitate
Scalabilitate



Părțile serviciului trebuie să se scaleze diferit?

Granularity desintegrators

Aplicabilitate
Volatilitate
Scalabilitate
Toleranță la erori

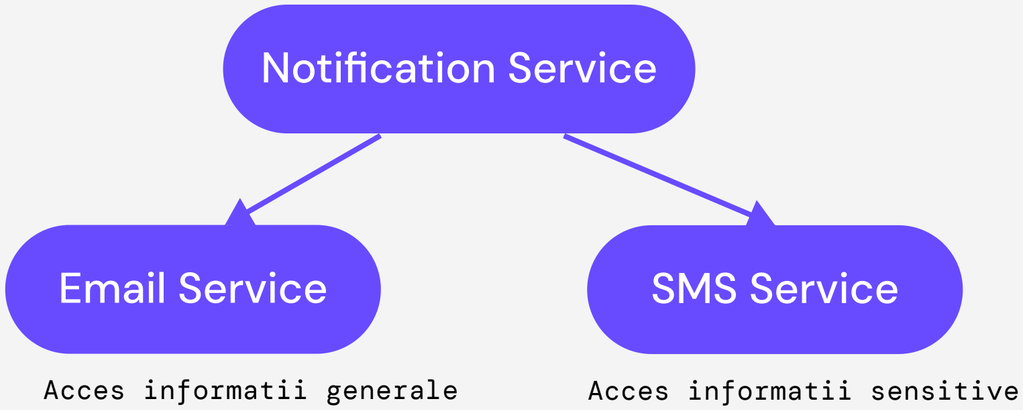


Există erori care cauzează eșecul funcțiilor critice în cadrul serviciului?

Granularity desintegrators

Aplicabilitate
Volatilitate
Scalabilitate
Toleranță la erori
Securitate

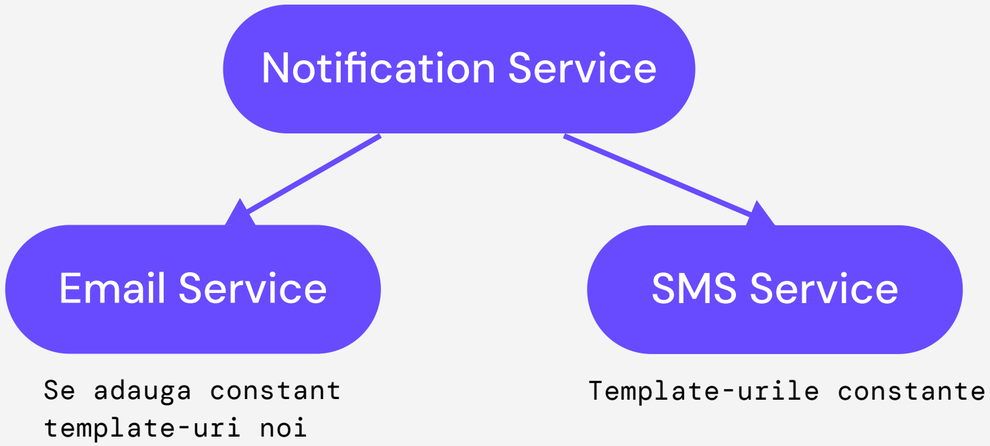
Unele părți ale serviciului necesită niveluri de securitate mai ridicate decât altele?



Granularity desintegrators

Serviciul se extinde mereu pentru a adăuga contexte noi?

Aplicabilitate
Volatilitate
Scalabilitate
Toleranță la erori
Securitate
Extensibilitate

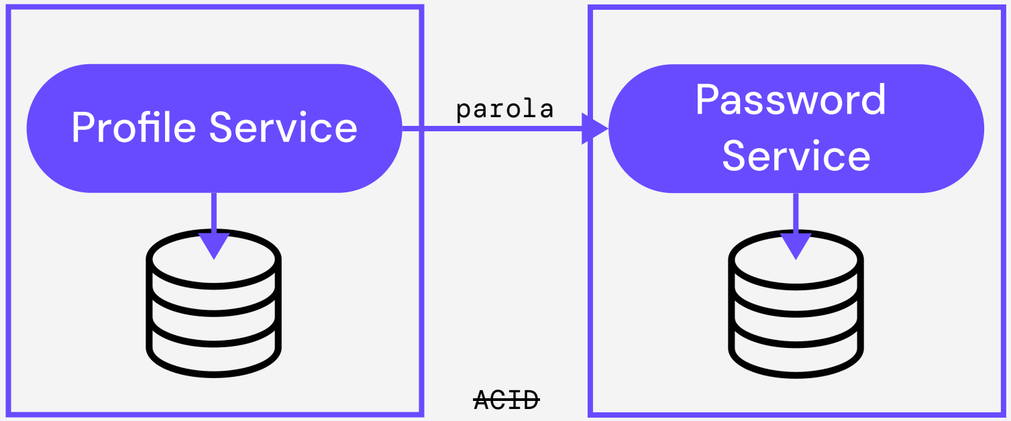


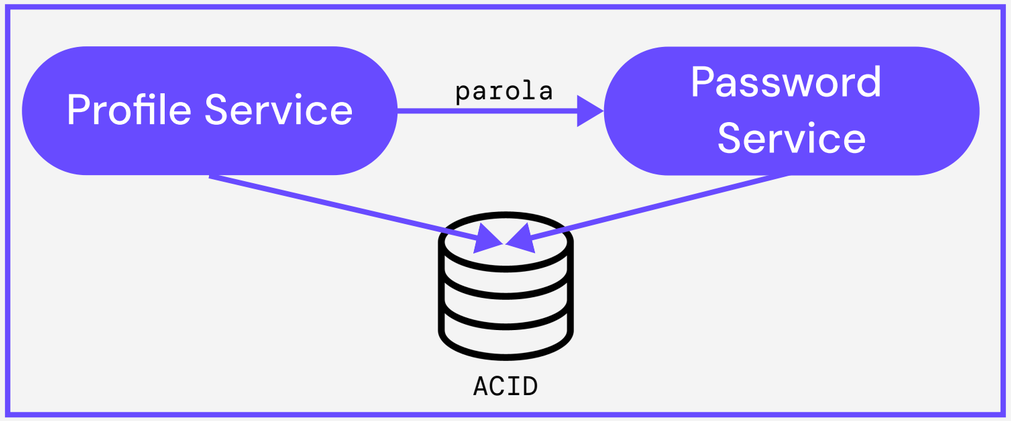
Granularity integrators

Tranzacții cu baze de date

Este necesară o tranzacție ACID între servicii separate?

Exemplu: Înregistrarea unui user

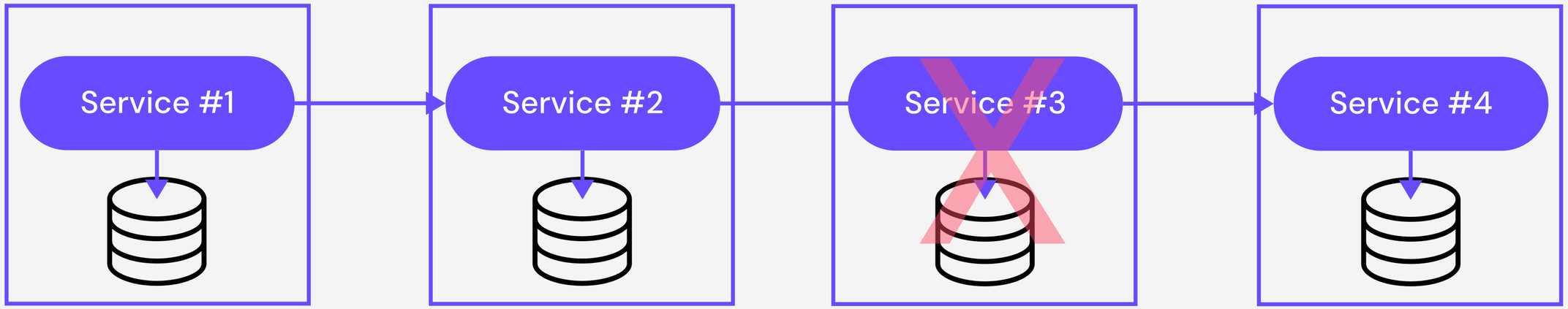




Granularity integrators

Serviciile trebuie să discute între ele?

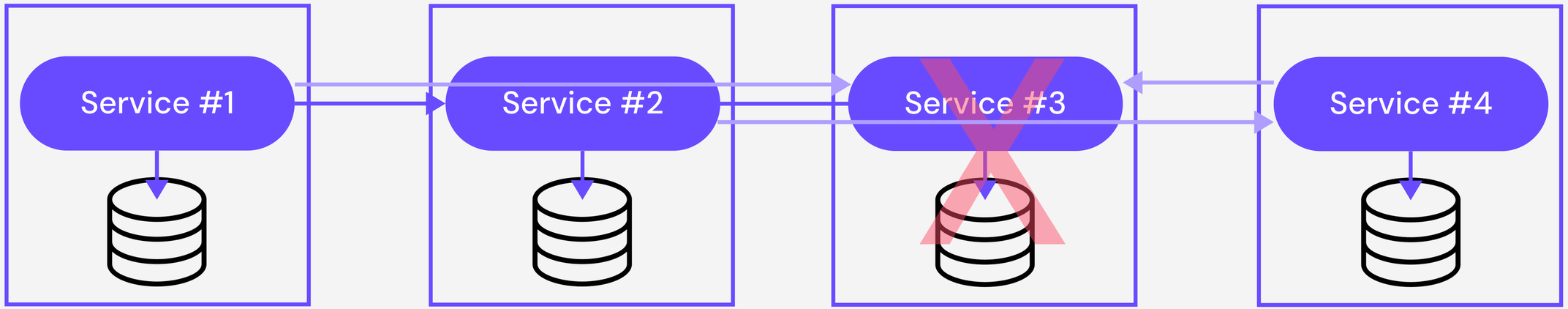
Tranzacții cu baze de date
Flux de lucru și coregrafie



Granularity integrators

Serviciile trebuie să discute între ele?

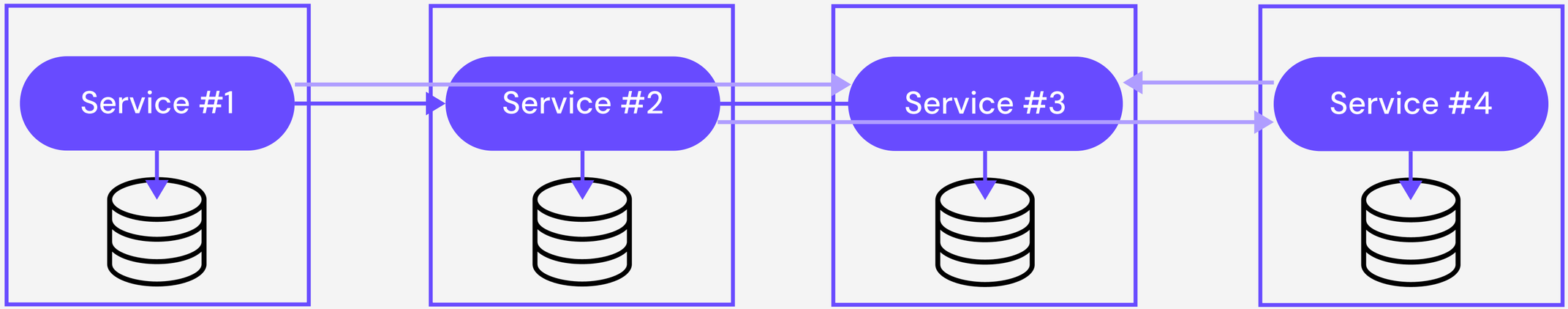
Tranzacții cu baze de date
Flux de lucru și coregrafie



Granularity integrators

Serviciile trebuie să discute între ele?

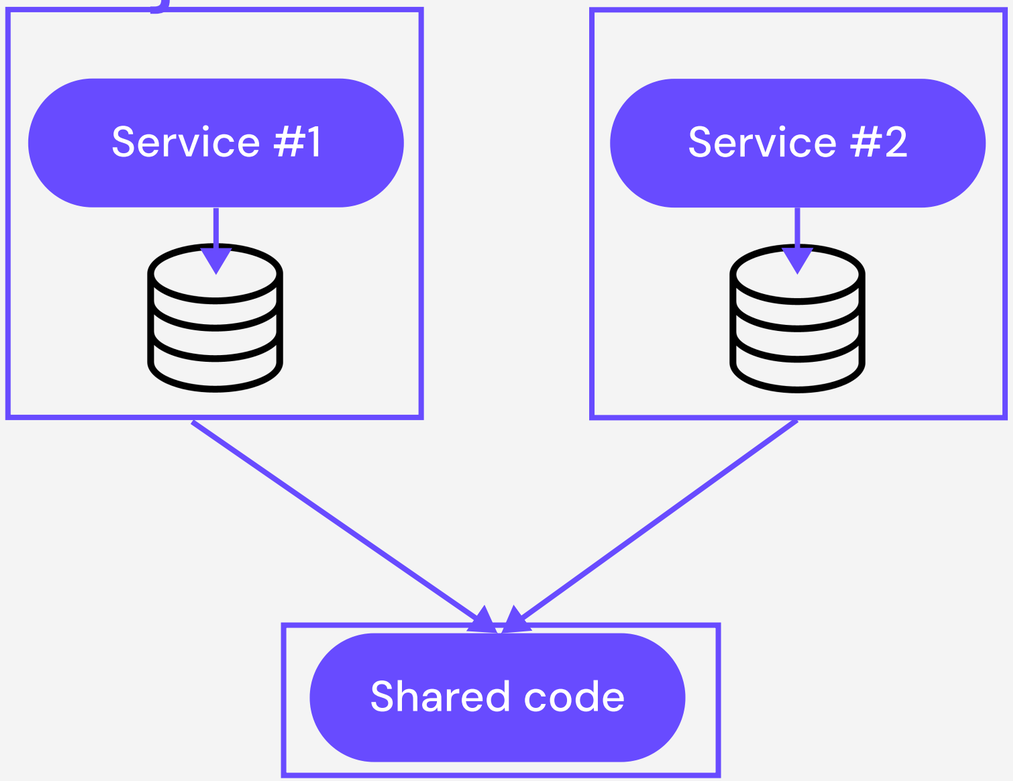
Tranzacții cu baze de date
Flux de lucru și coregrafie



Granularity integrators

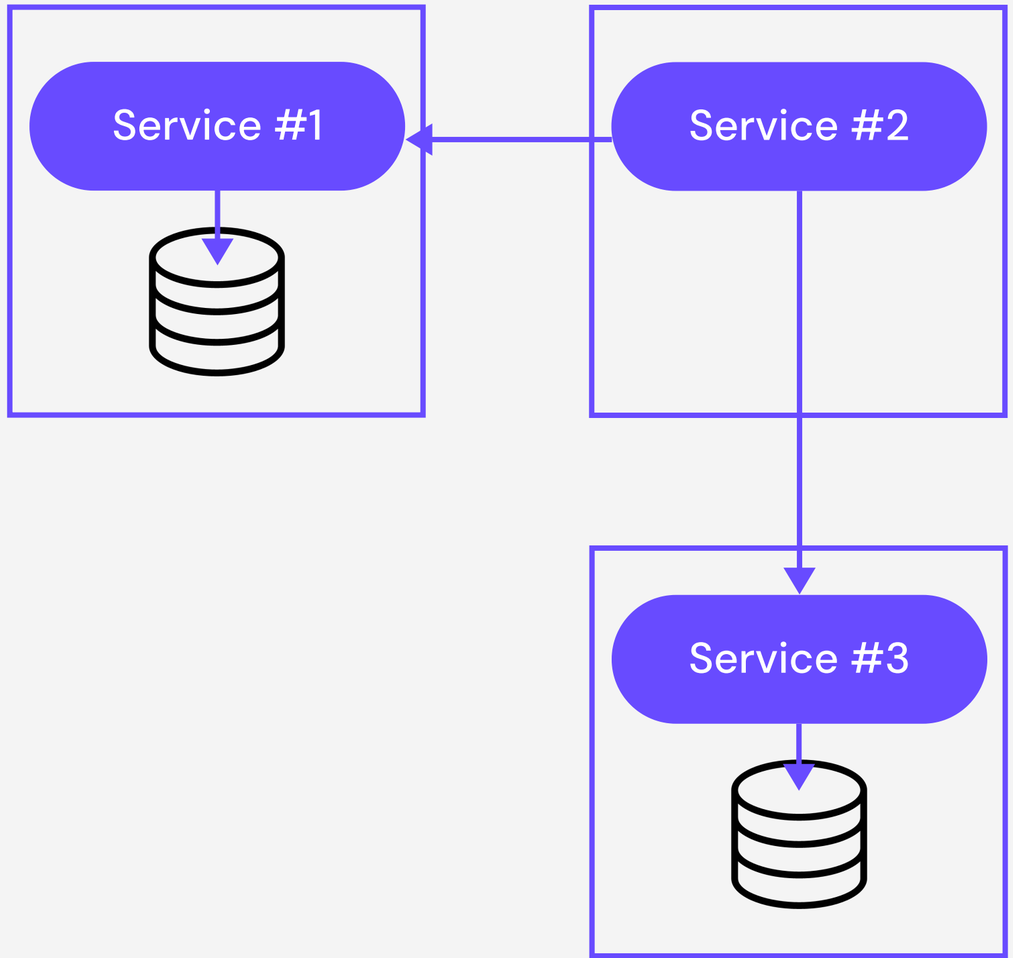
Serviciile trebuie să partajeze codul între ele?

Tranzacții cu baze de date
Flux de lucru și coregrafie
Shared code



Granularity integrators

Poate dependență de un microserviciu să afecteze funcționalitatea unui alt microserviciu?

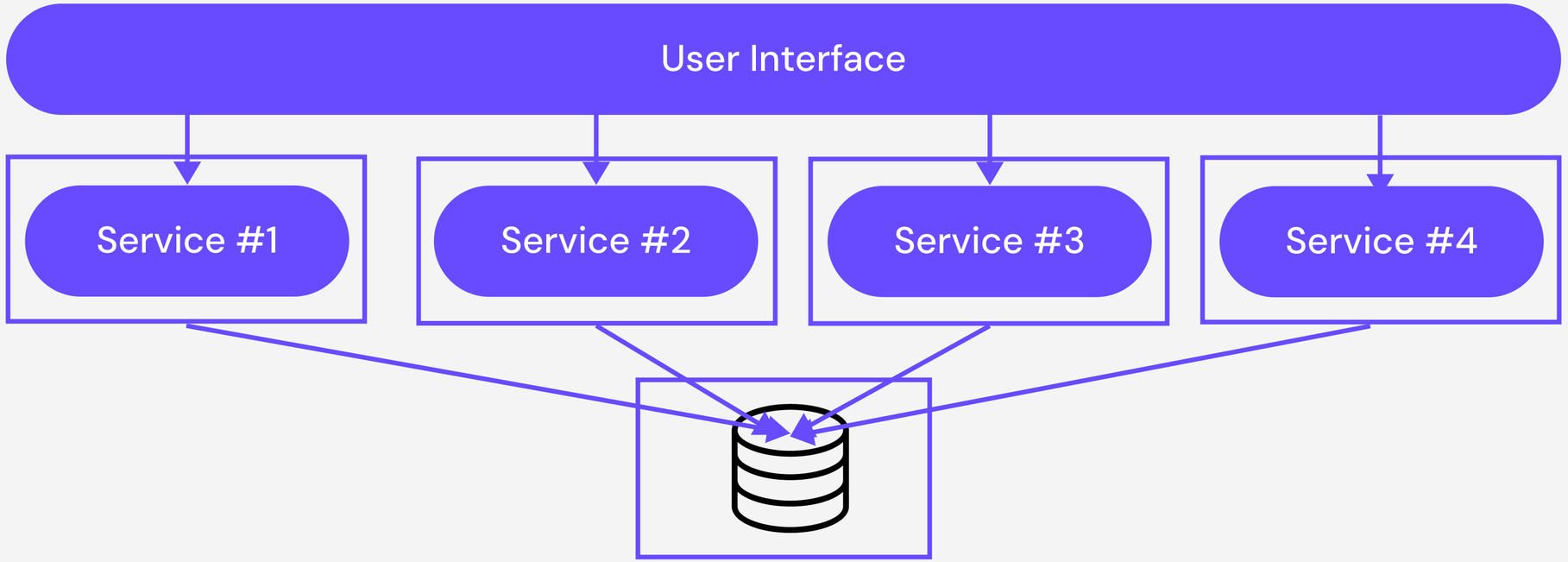


Tranzacții cu baze de date
Flux de lucru și coregrafie
Shared code
Relații cu baze de date

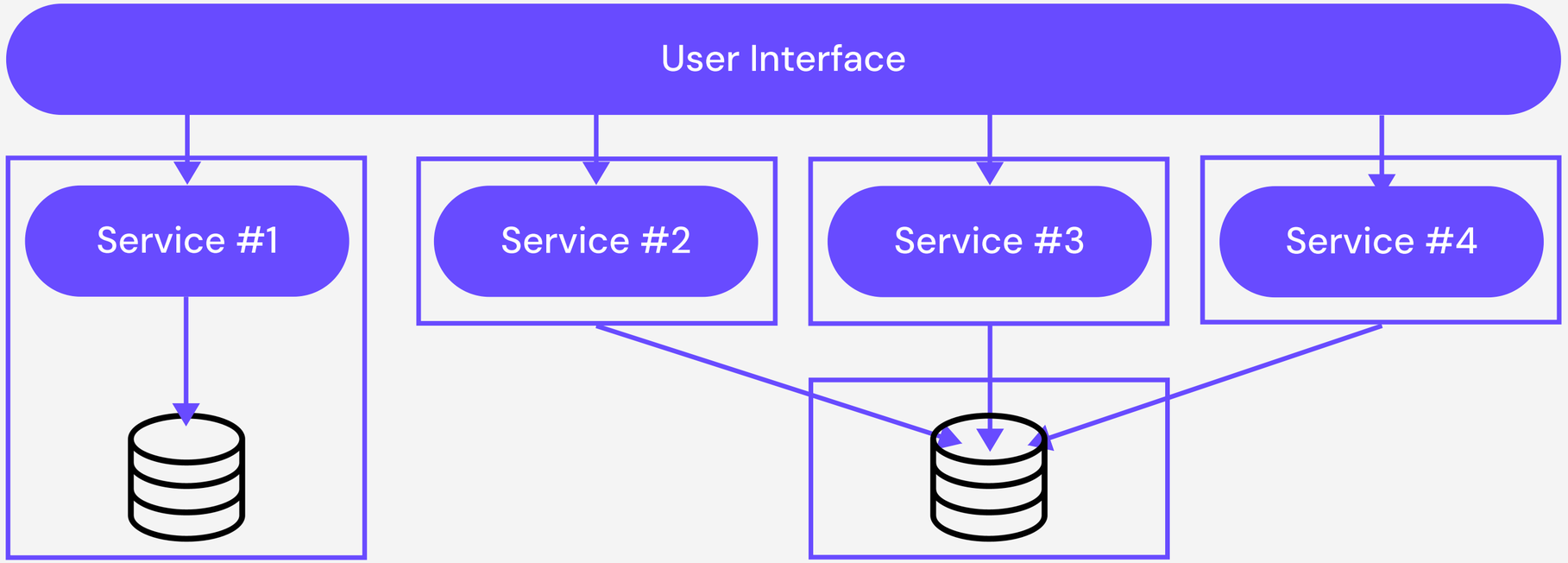
Granularitatea aplicației depinde de un echilibru între dezintegratori și integratori.

Care sunt cele mai bune practici pentru a face ajustări în granularitatea microserviciilor pe măsură ce cerințele proiectului evoluează?

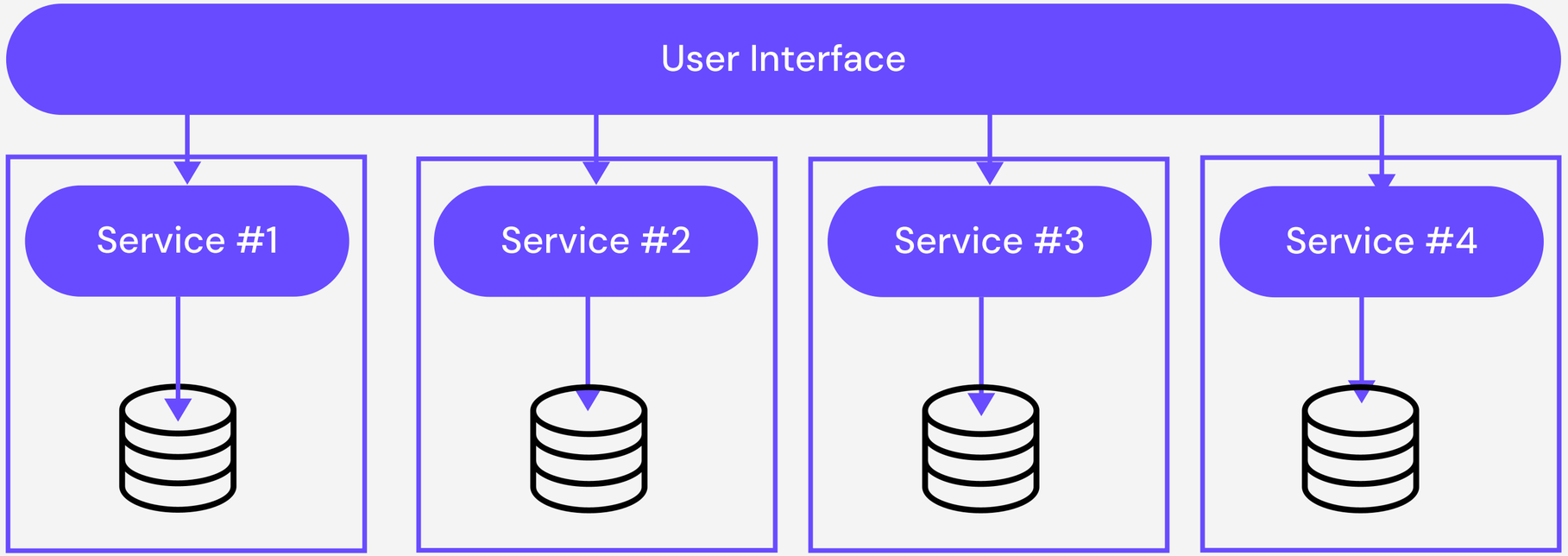
Descompunerea în Microservicii - Cum definim microserviciile & granularitate



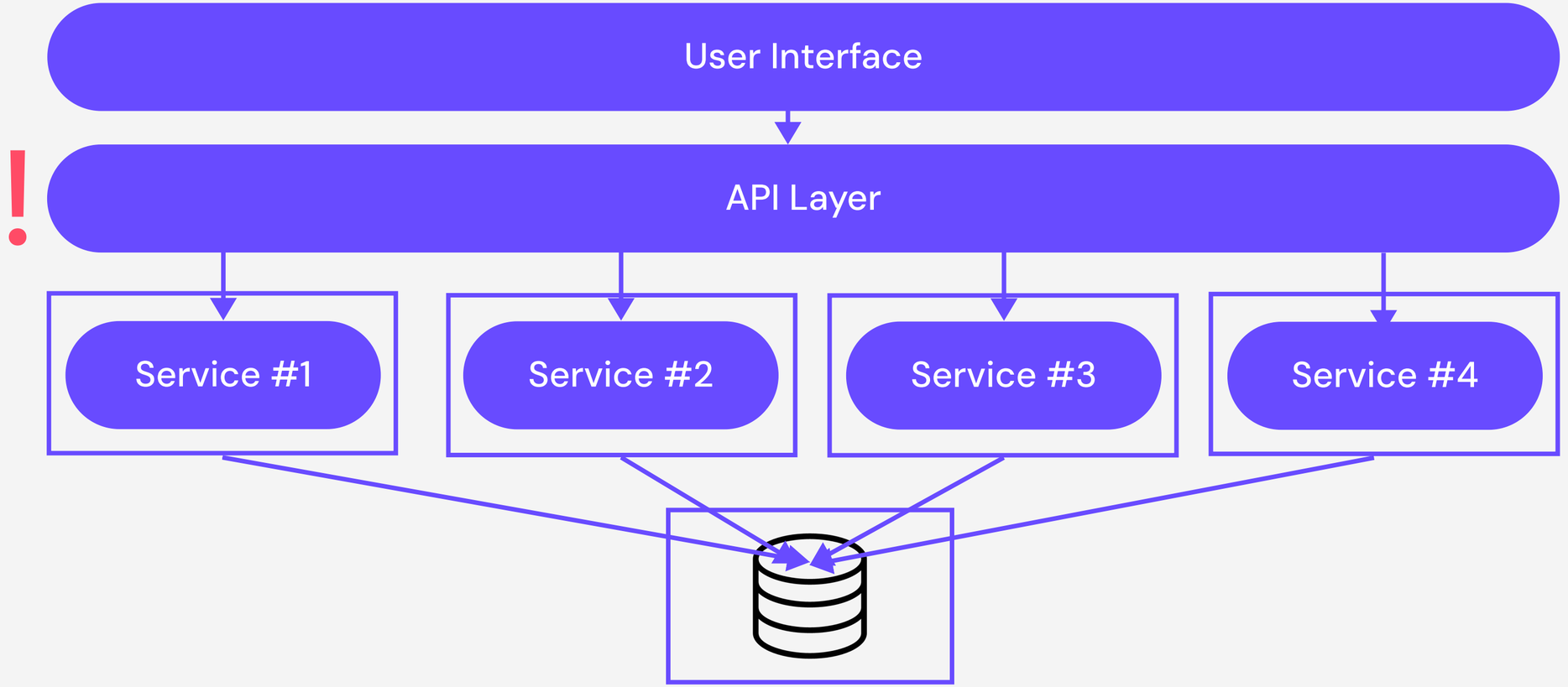
Descompunerea în Microservicii - Cum definim microserviciile & granularitate



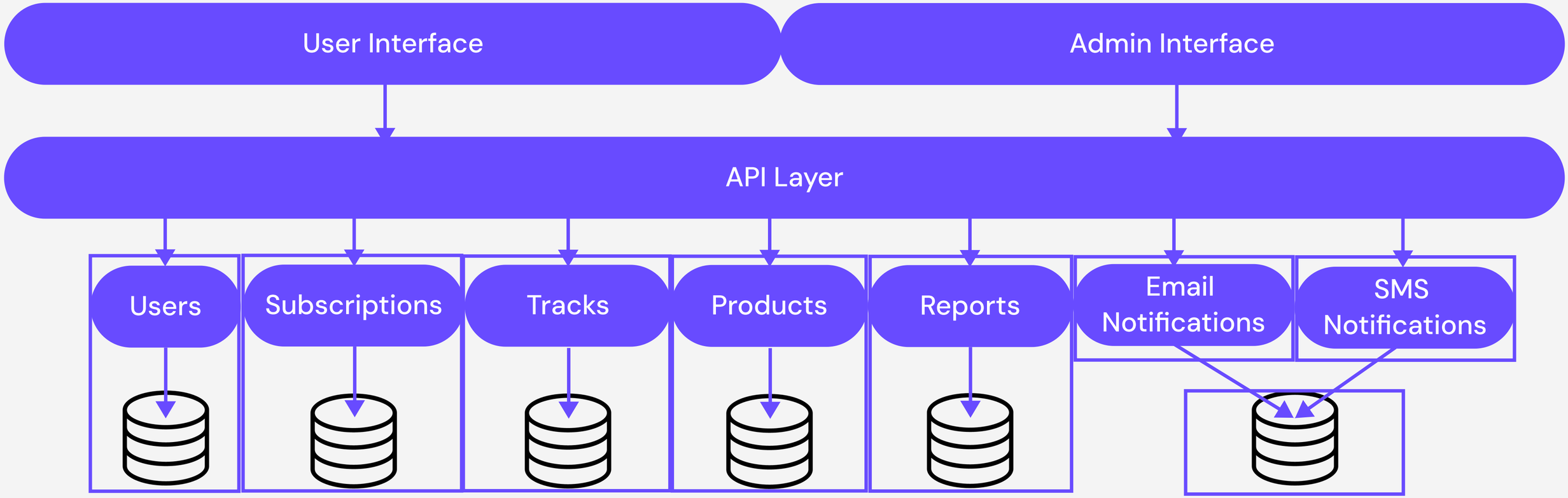
Descompunerea în Microservicii - Cum definim microserviciile & granularitate



Descompunerea în Microservicii - Cum definim microserviciile & granularitate



Black Friday Deal Tracker



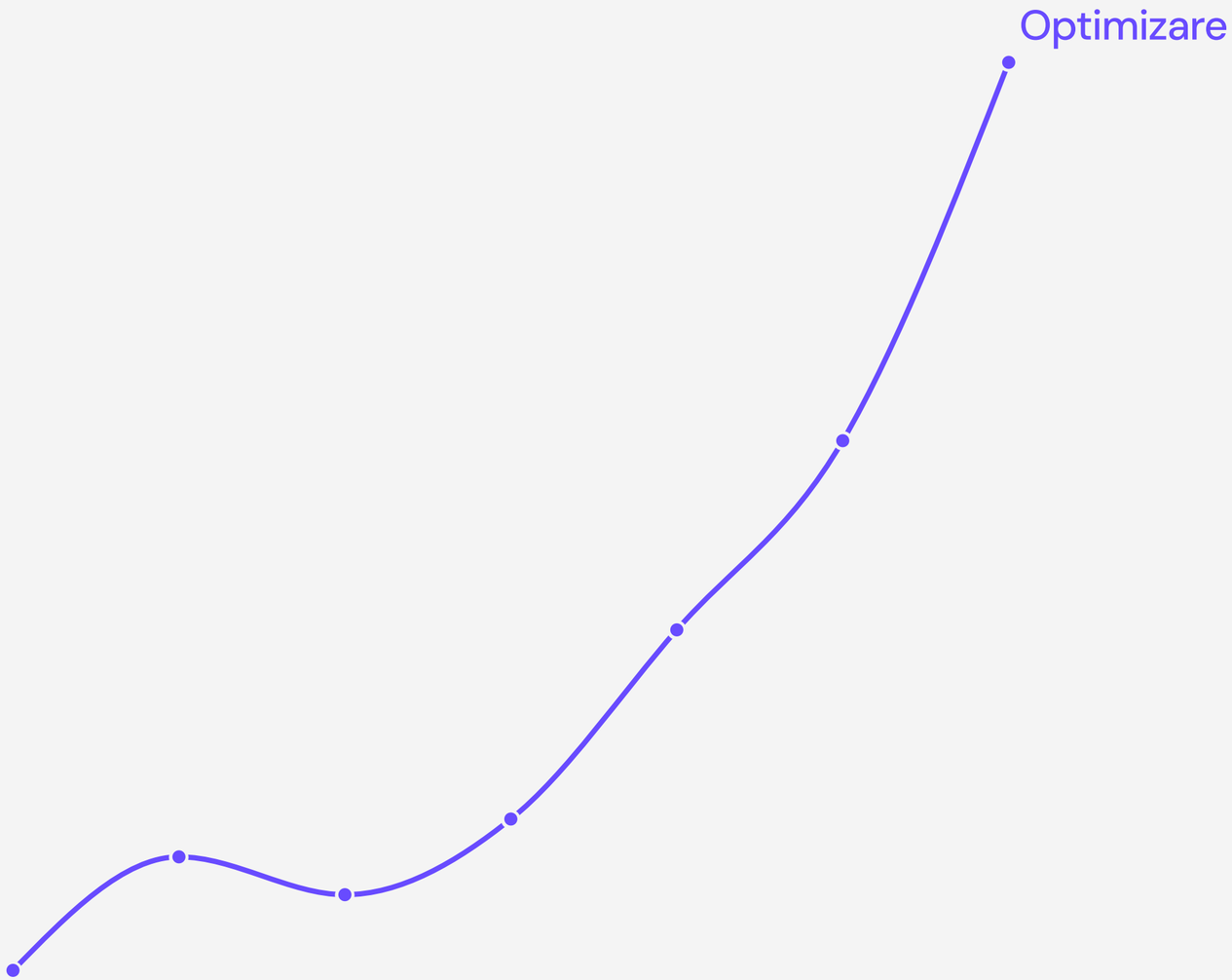
Descompunerea în Microservicii – Business Capability în proiectul nostru

Complexitatea este bună, dar pentru un motiv.

Complexitatea este dificilă atunci când există prea multe părți în mișcare.

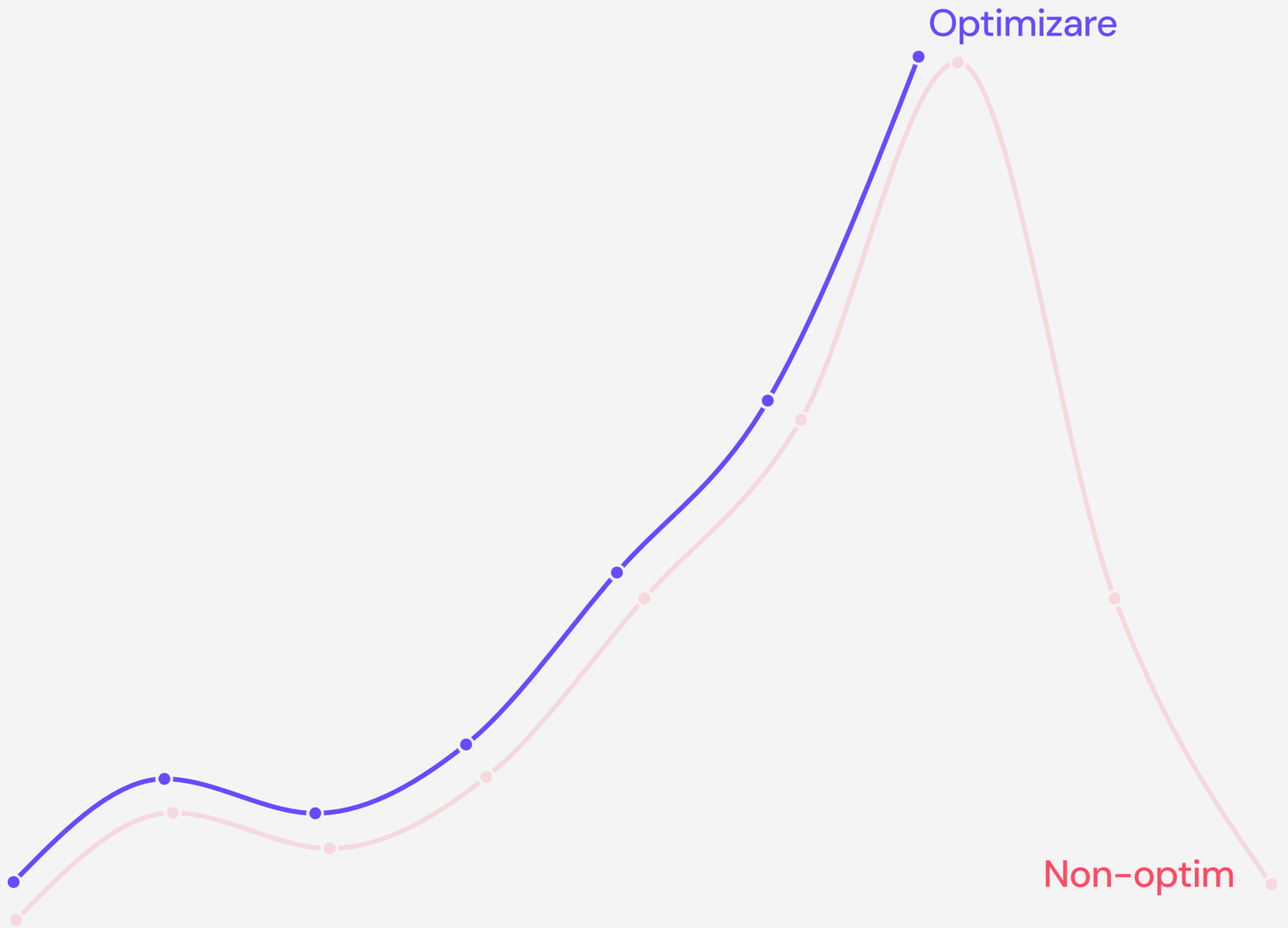
Descompunerea în Microservicii - Business Capability în proiectul nostru

- Independență
- Separare a responsabilităților
- Isolare și scalabilitate individuală
- Tehnologii variate
- Gestionarea datelor proprii
- Rezilienţă și gestionare a erorilor
- Monitorizare și gestionare centralizată
- Dezvoltare și livrare continuă



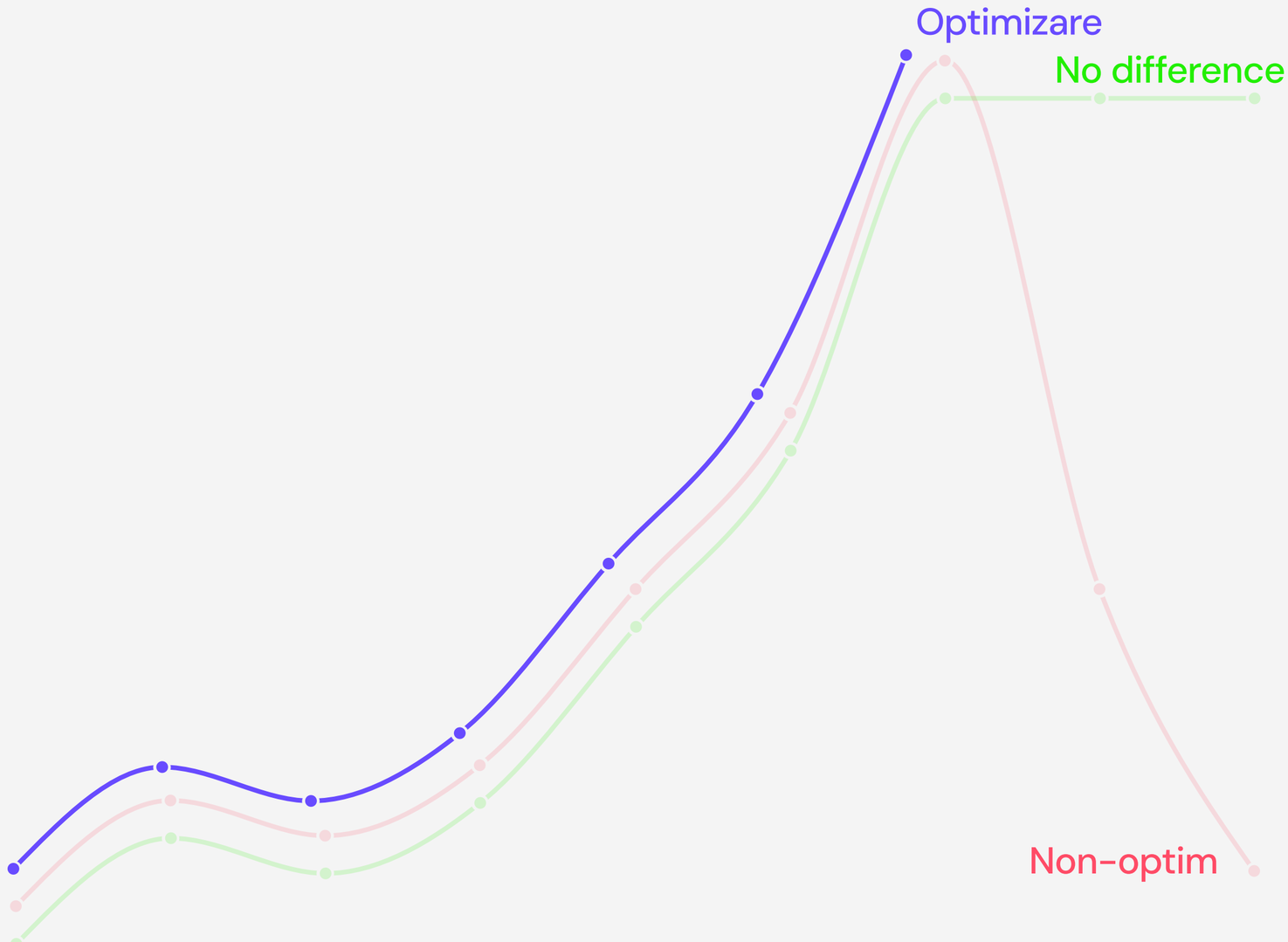
Descompunerea în Microservicii - Business Capability în proiectul nostru

- Independență
- Separare a responsabilităților
- Isolare și scalabilitate individuală
- Tehnologii variate
- Gestionarea datelor proprii
- Rezilienţă și gestionare a erorilor
- Monitorizare și gestionare centralizată
- Dezvoltare și livrare continuă



Descompunerea în Microservicii - Business Capability în proiectul nostru

- Independență
- Separare a responsabilităților
- Isolare și scalabilitate individuală
- Tehnologii variate
- Gestionarea datelor proprii
- Rezilienț și gestionare a erorilor
- Monitorizare și gestionare centralizată
- Dezvoltare și livrare continuă

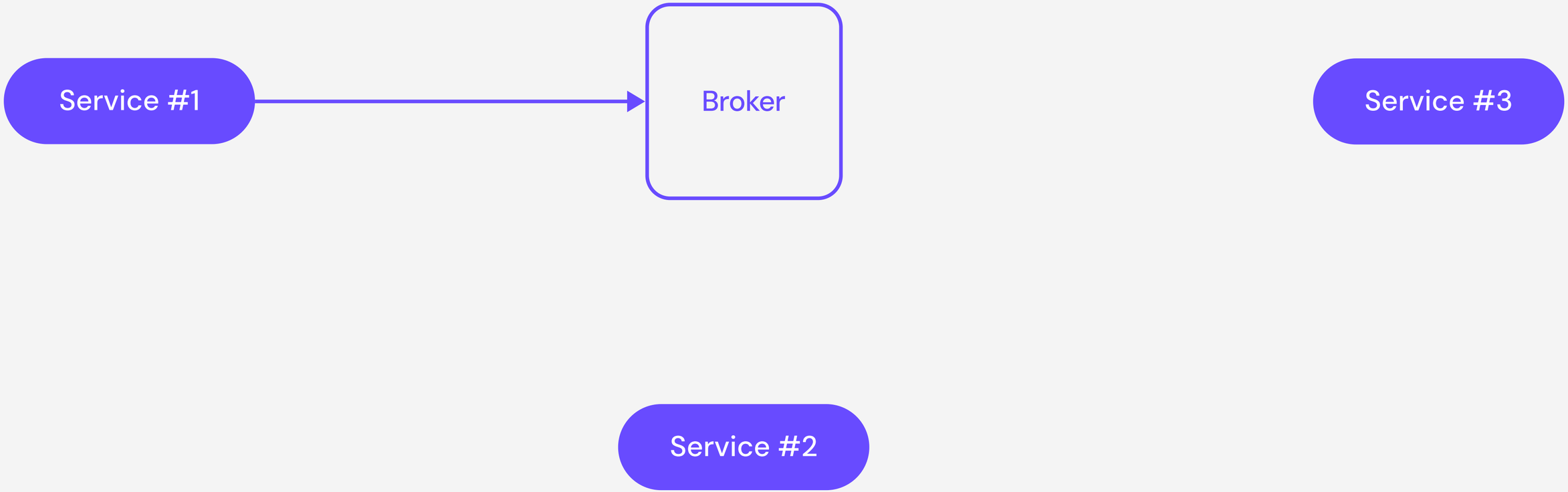


Arhitectura bazată pe evenimente – Topologiile Broker și Mediator

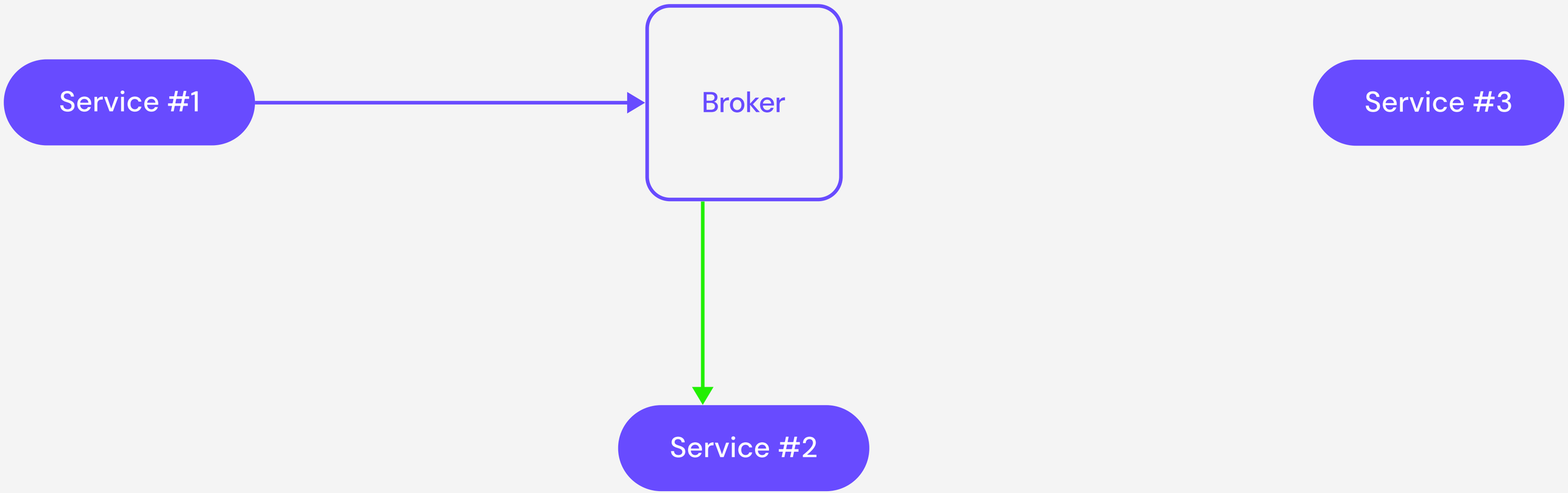
Brokerul nu este altceva decât o multitudine de evenimente înlănțuite fără a fi nevoie de un serviciu central.

Fluxul de mesaje este distribuit între consumatorii evenimentului.

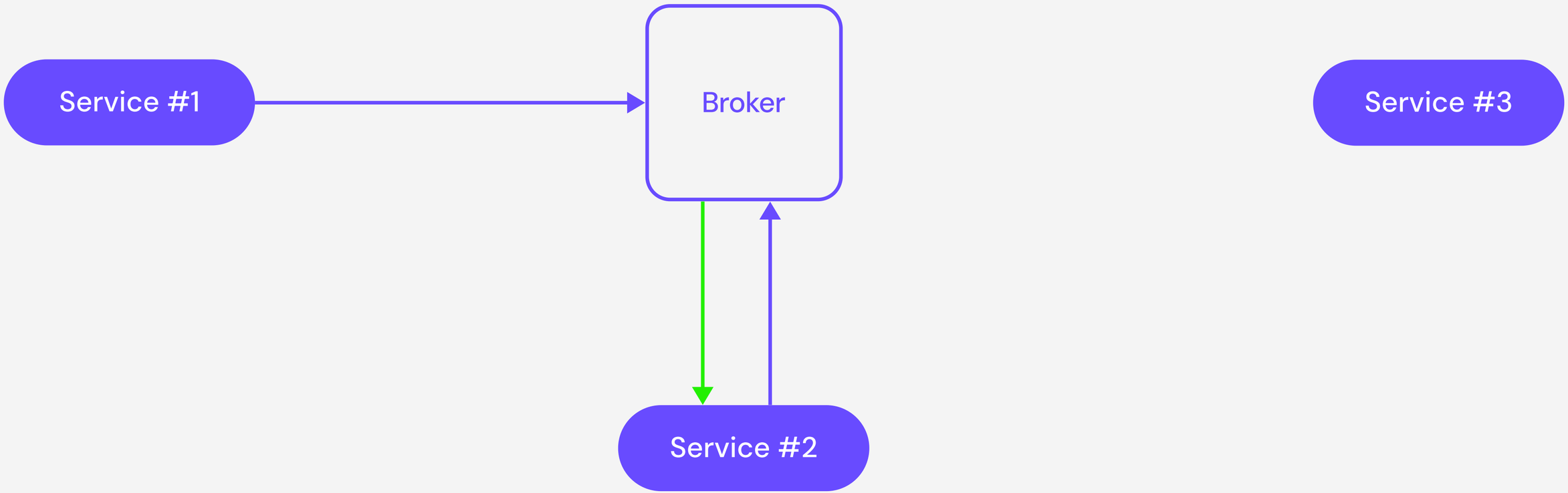
Arhitectura bazată pe evenimente - Topologiile Broker și Mediator



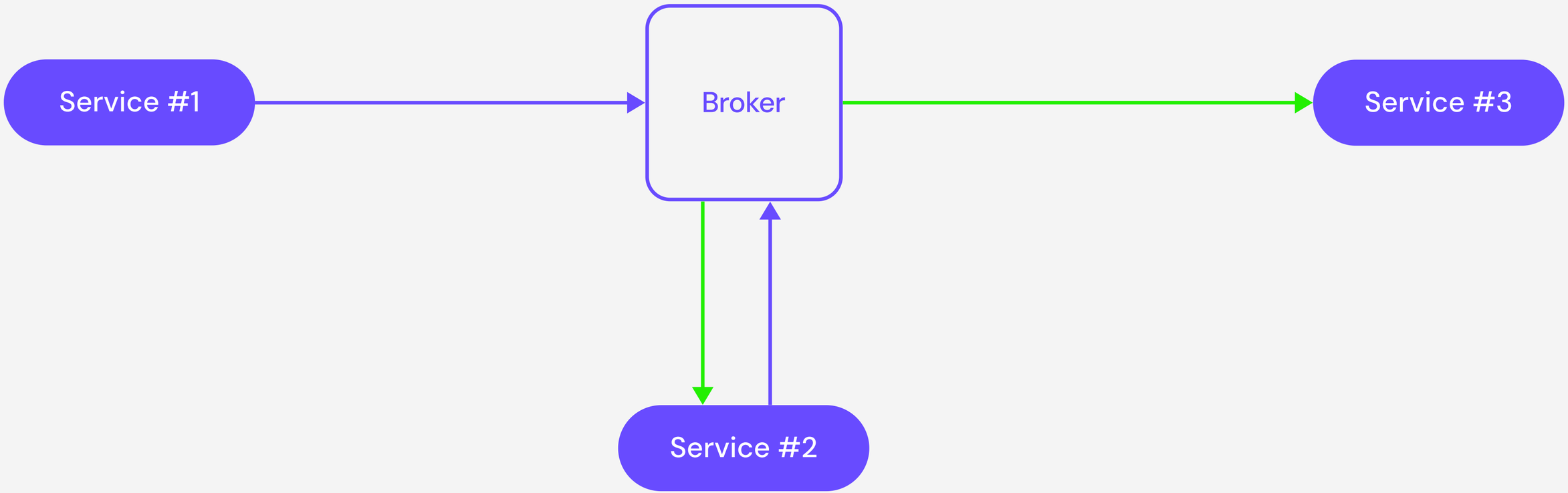
Arhitectura bazată pe evenimente - Topologiile Broker și Mediator



Arhitectura bazată pe evenimente – Topologiile Broker și Mediator



Arhitectura bazată pe evenimente – Topologiile Broker și Mediator



exemplu cod

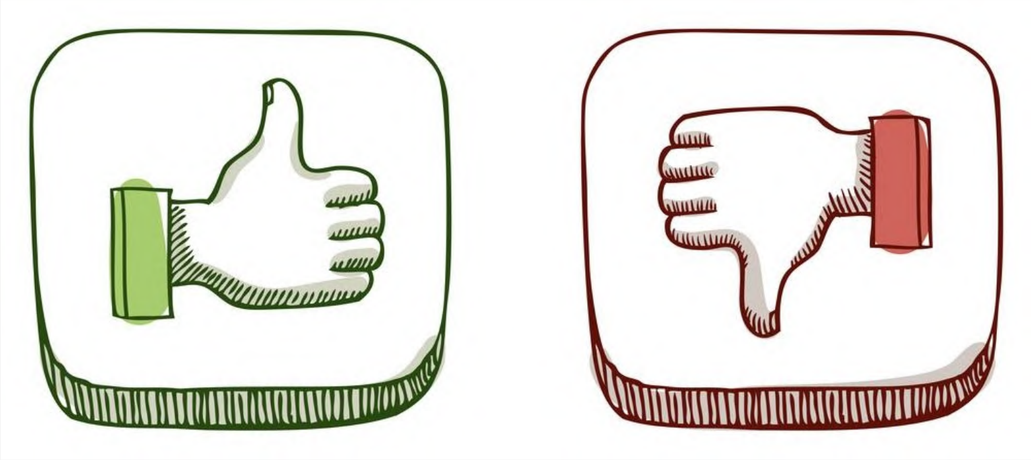
Arhitectura bazată pe evenimente – Topologiile Broker și Mediator

Scalabilitate

Decuplare

Ușor de implementat

Arhitectura bazată pe evenimente – Topologiile Broker și Mediator

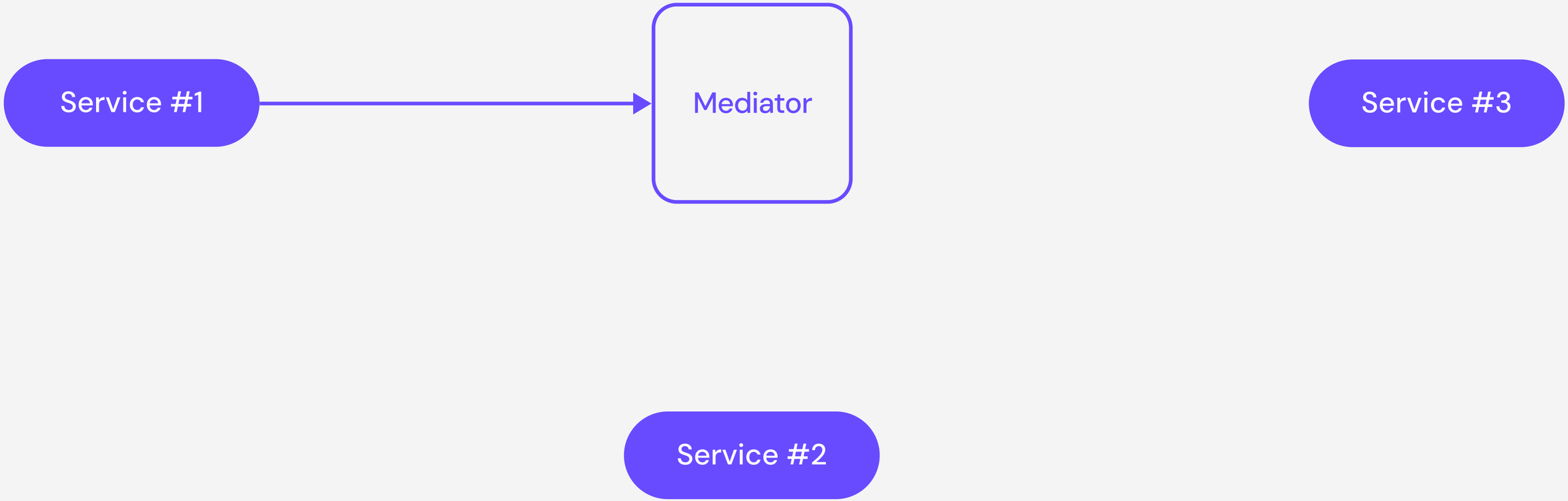


Scalabilitate
Decuplare
Ușor de implementat

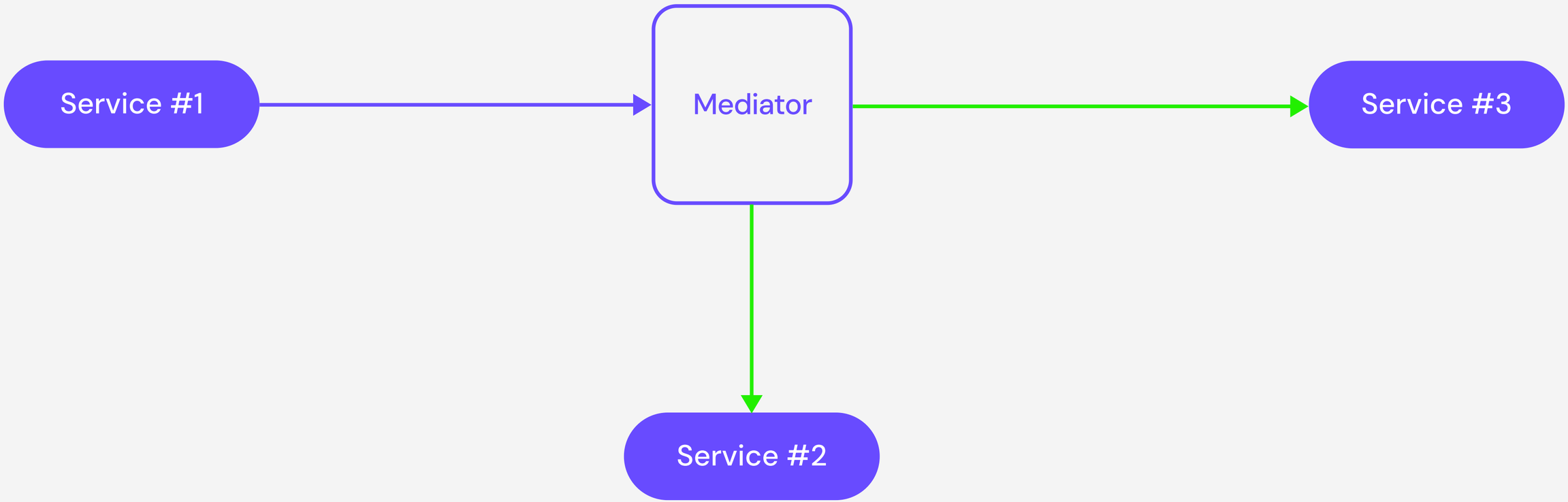
Testarea poate fi dificilă
Dezvoltarea poate fi complicată

Topologia mediatorului este folosită în mod obișnuit atunci când trebuie să orchestrăm mai mulți pași într-un eveniment printr-un mediator central.

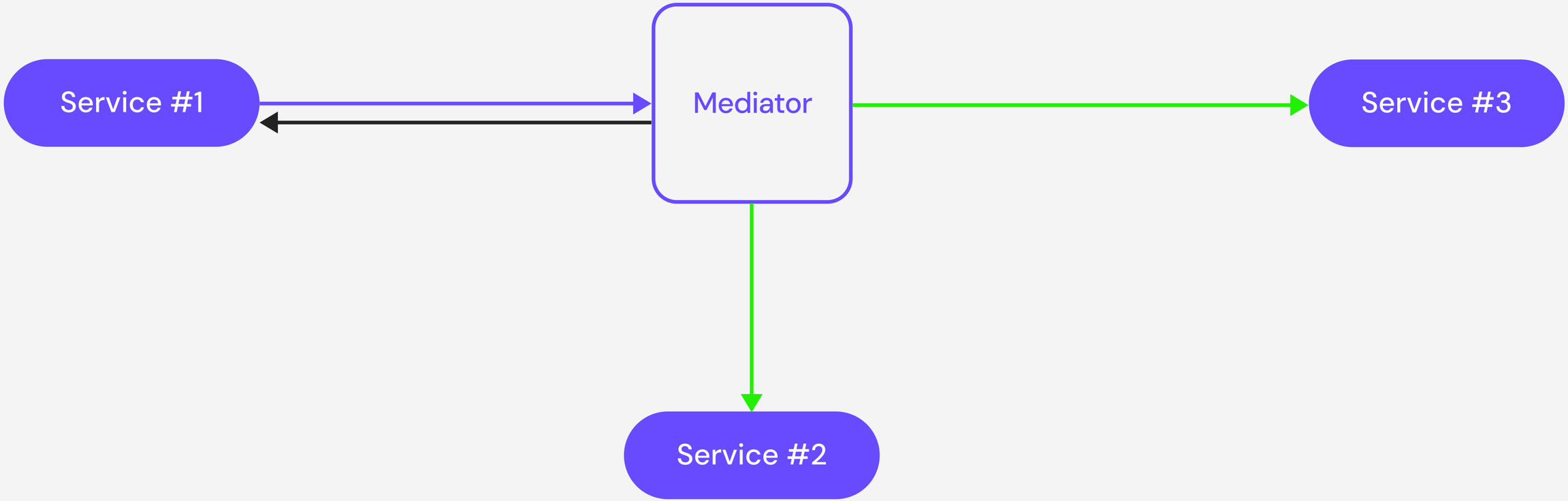
Arhitectura bazată pe evenimente - Topologiile Broker și Mediator



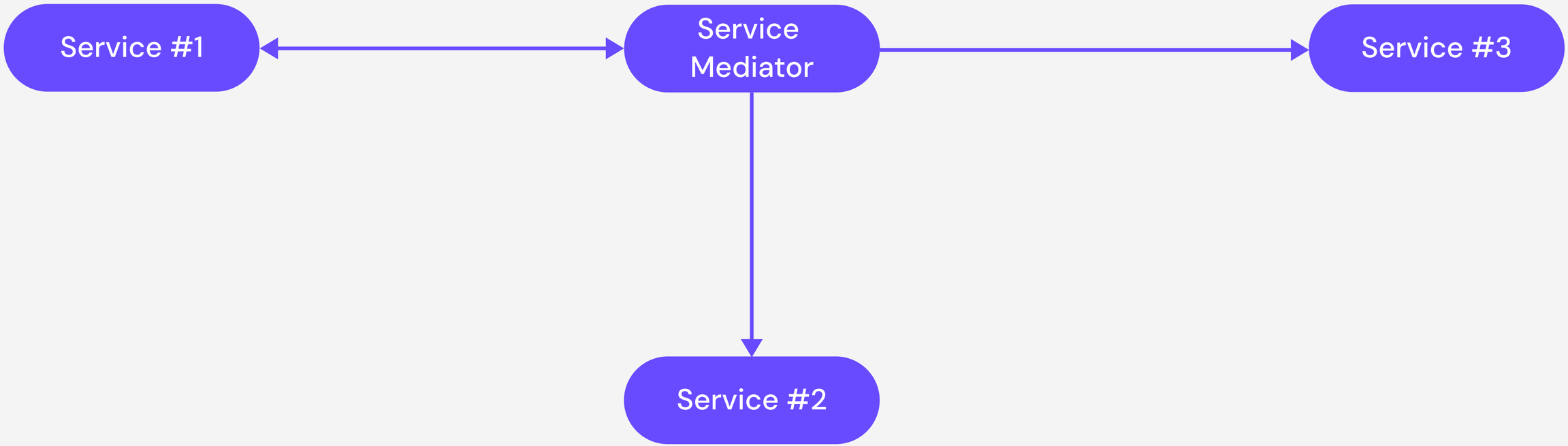
Arhitectura bazată pe evenimente – Topologiile Broker și Mediator



Arhitectura bazată pe evenimente - Topologiile Broker și Mediator



Arhitectura bazată pe evenimente - Topologiile Broker și Mediator



exemplu cod

Arhitectura bazată pe evenimente – Topologiile Broker și Mediator

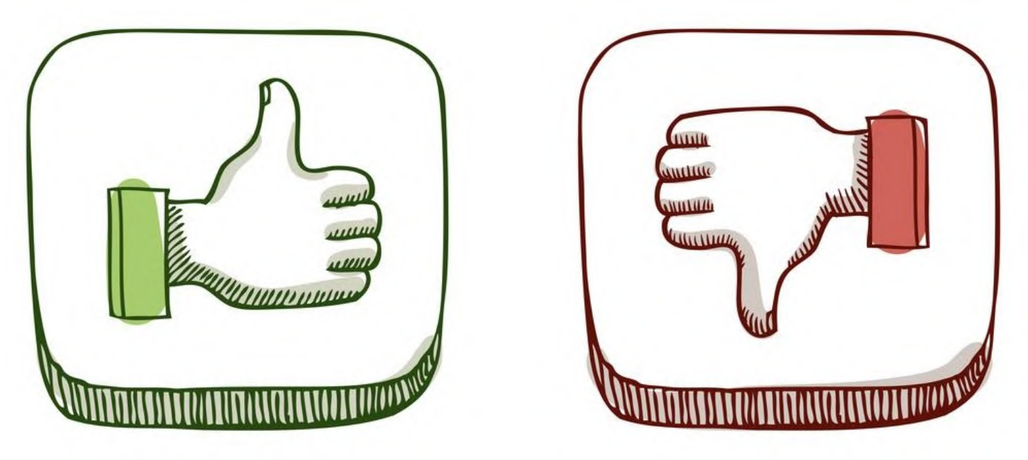
Decuplare

Comunicare simplificată

Control centralizat

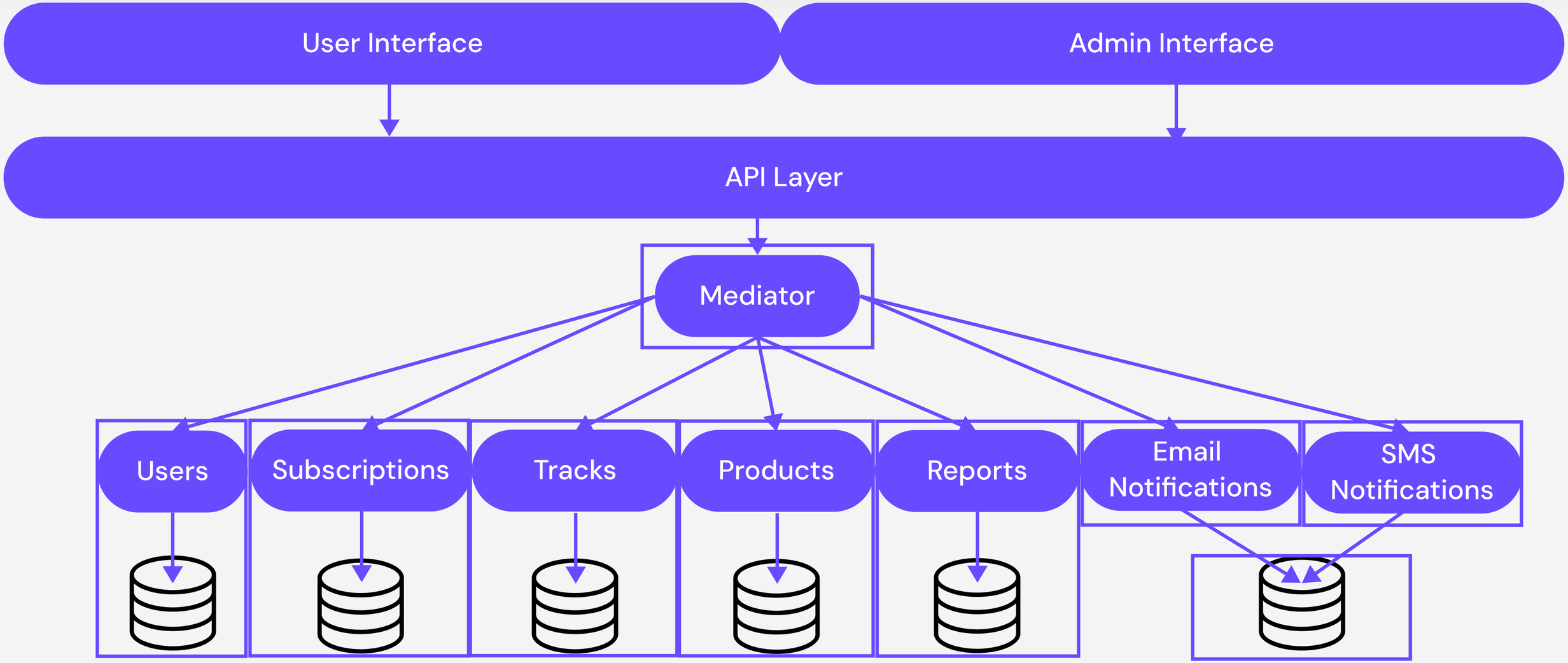
Flexibilitate

Arhitectura bazată pe evenimente – Topologiile Broker și Mediator

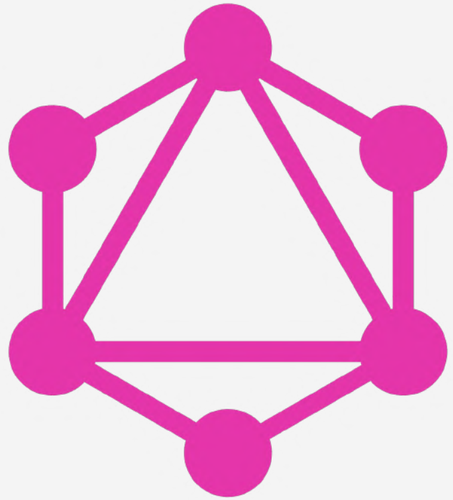


Decuplare	Punct unic de eșec
Comunicare simplificată	Complexitate
Control centralizat	Potențial blocaj
Flexibilitate	Latență crescută

Arhitectura bazată pe evenimente – Topologiile Broker și Mediator



Comunicarea și Protocol – Tipuri de comunicare între microservicii



exemplu cod

Circuit Breaker Pattern
previne accesul la un serviciu sau o funcționalitate care are probleme persistente. Acesta se inspiră din conceptul unui întrerupător electric, care oprește furnizarea de energie în caz de suprasarcină sau scurtcircuit.

exemplu in python de implementare

Retry Pattern

când o operațiune eșuează din cauza unei erori temporare a unui serviciu, avem posibilitatea sa încercăm din nou aceeași operațiune.

exemplu in python de implementare

Comunicarea și Protocol – Prevenirea pierderii datelor

Circuit Breaker și Retry, sunt importante în dezvoltarea aplicațiilor care trebuie să fie reziliente la eșecuri și să ofere o experiență robustă utilizatorilor, chiar și în condiții dificile sau în medii distribuite

Demo

Un monolit poate fi mai optim decât un set de microservicii.

Un monolit poate fi mai optim decât un set de microservicii.

Nu orice parte din sistem trebuie să fie un microserviciu.

Journey into Microservices – Lessons learned

Un monolit poate fi mai optim decât un set de microservicii.

Nu orice parte din sistem trebuie să fie un microserviciu.

Microserviciile sunt utile dacă sunt adaptate corect nevoilor.

Granularitatea aplicației depinde de un echilibru între dezintegratori și integratori.

Journey into Microservices – Lessons learned

Granularitatea aplicației depinde de un echilibru între dezintegratori și integratori.

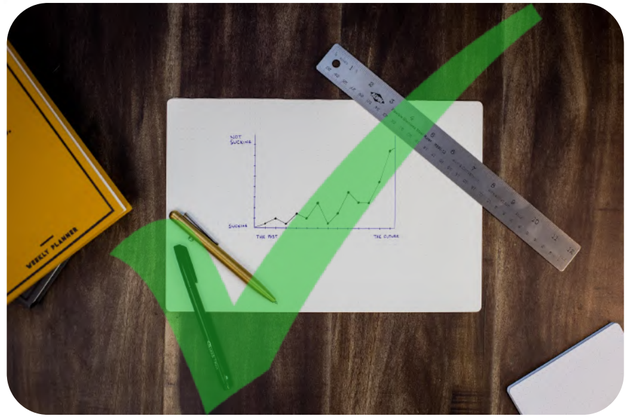
Complexitatea este bună, dar pentru un motiv.

Granularitatea aplicației depinde de un echilibru între dezintegratori și integratori.

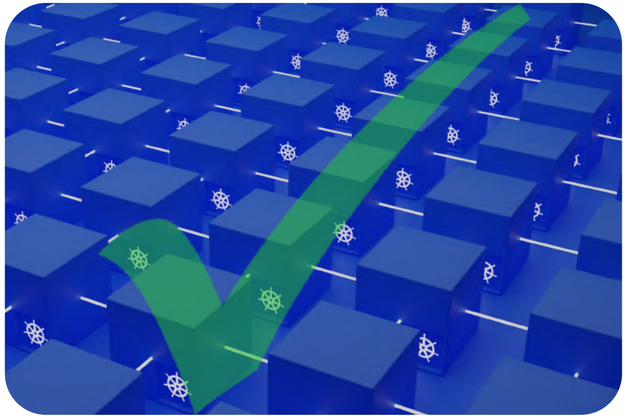
Complexitatea este bună, dar pentru un motiv.

Iterația este singura modalitate de a asigura un design bun al serviciului.

Boosting Performance with Data Caching



Performance Optimization Techniques



Journey into Microservices

Extra - 20-minute rule