Studiu comparativ al modelelor de comunicare pentru microservicii: analiza abordărilor sincrone (cerere-răspuns) și asincrone (event-driven) aplicate pe diferite scenarii

PREZENTARE PROIECT (MILESTONE 1)

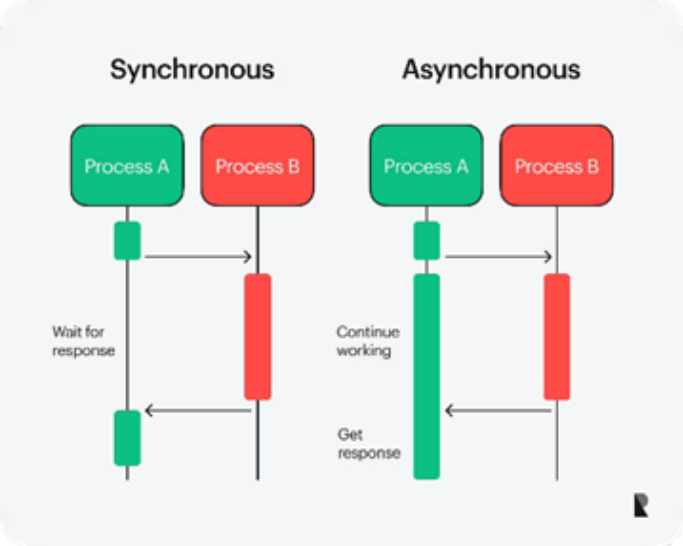
Membrii echipei: Ioana Profeanu - EGOV-2, Daria Ciobanu - EGOV-2, Alexandra Dumitrescu - SCPD-2, Anana-Catalina Gîrlea - SCPD-2

1. Introducere și Obiective

Proiectul presupune implementarea a **două versiuni** fundamental diferite ale aceluiași set de microservicii. Scopul este de a realiza un studiu științific și cantitativ pentru a răspunde la întrebarea: "Cum influențează alegerea unui stil de comunicare (sincron vs. asincron) atributele cheie ale unui sistem distribuit?".

Obiectivul principal este de a măsura și compara cele două arhitecturi în contextul a 6 scenarii de utilizare distincte, care simulează sarcini de lucru din lumea reală.

Livrabilul final va fi un articol științific care prezintă arhitectura sistemului, metodologia de testare, datele colectate (performanță, latență, utilizare resurse, reziliență) și concluziile trase în urma experimentului.



2. Metodologia de Comparație

Pentru a asigura că experimentul este valid științific, vom standardiza toate componentele, cu excepția variabilei de test: **modelul de comunicare**.

Elemente Standardizate (Constante):

- **Limbaj/Framework:** Toate microserviciile vor fi scrise în același limbaj și vor folosi același framework (de ex., **Node.js cu Fastify** sau **Python cu FastAPI**) pentru a elimina orice influență a performanței limbajului.
- **Baza de Date:** Toate serviciile care necesită persistență vor folosi **PostgreSQL**.
- **Orchestrare:** Ambele sisteme vor rula în containere **Docker**, gestionate prin fișiere **Docker Compose** separate.
- **Broker de Mesaje:** Sistemul asincron va folosi **RabbitMQ**.

Elemente Comparate (Variabile):

Arhitectura A: Sincronă (Cerere-Răspuns)

- **Descriere:** O implementare care urmează un stil arhitectural strict de tip Cerere-Răspuns.
- **Mecanism:** Serviciile comunică între ele prin apeluri API **HTTP (REST) directe, blocante**.
- **Comportament:** Când Serviciul A apelează Serviciul B, el trebuie să aștepte activ un răspuns (sau un timeout) înainte de a-și putea continua propria execuție. Acest lucru creează un **cuplaj temporal** strâns.

Arhitectura B: Asincronă (Event-Driven)

- **Descriere:** O implementare care urmează un stil arhitectural orientat pe evenimente (event-driven).
- **Mecanism:** Serviciile comunică decuplat prin intermediul unui **Message Broker (RabbitMQ)**.
- **Comportament:** Când Serviciul A dorește o acțiune de la Serviciul B, el publică un eveniment (un mesaj) în broker și își continuă imediat execuția. Serviciul B (și oricare alt serviciu interesat) este abonat la acel eveniment și va acționa asupra lui independent, în propriul său ritm. Acest lucru elimină cuplajul temporal.

3. Scenariile de Implementare

Pentru a testa cele două arhitecturi, vom implementa 6 scenarii care simulează diverse provocări:

Scenariul 1: Decuplarea Sarcinii Non-Critice

- *Descriere:* Înregistrarea unui utilizator nou. Operațiunea critică este scrierea în UserService, iar cea non-critică este trimiterea unui e-mail de bun venit (simulată, 500ms delay) de către EmailService.
- *Obiectivul testului:* Măsurarea latenței percepute de client și a rezilienței la eșecul serviciului non-critic.

Scenariul 2: Proces de Lungă Durată Simulat (API Extern)

- *Descriere:* Procesarea unei plăți. PaymentService **simulează** un apel la un API extern (ex. Stripe) introducând un **delay artificial de 2 secunde**.
- *Obiectivul testului:* Măsurarea impactului major asupra timpului de răspuns și a numărului de conexiuni HTTP blocate pe server.

Scenariul 3: Flux "Fan-Out" (Acțiuni Paralele)

- *Descriere:* Actualizarea unui produs. O singură acțiune (ProductService) trebuie să declanșeze 3 acțiuni separate: re-indexarea în SearchService, invalidarea în CacheService și înregistrarea în AnalyticsService.
- *Obiectivul testului:* Compararea latenței totale (secvențială în sincron vs. paralelă în asincron) și a throughput-ului.

Scenariul 4: Sarcină CPU-Intensivă (Procesare Asincronă)

- *Descriere:* Generarea unui raport lunar. ReportService **simulează** o operațiune CPU-intensivă (ex. un calcul matematic complex sau hashing repetat) care durează 10 secunde.
- *Obiectivul testului:* Demonstrarea fezabilității. Sistemul sincron va eșua prin timeout HTTP, în timp ce cel asincron va gestiona sarcina în fundal.

Scenariul 5: Coregrafie și Compensare (Saga Pattern)

- *Descriere:* Plasarea unei comenzi care implică mai mulți pași ce pot eșua: 1. Rezervare stoc (InventoryService), 2. Procesare plată (PaymentService). Se testează eșecul plății.
- *Obiectivul testului:* Analiza complexității implementării logicii de compensare (anularea rezervării stocului) în sincron (manuală, în OrderService) versus asincron (coregrafiată, prin abonarea InventoryService la evenimentul PaymentFailed).

Scenariul 6: Ingestie de Date High-Throughput

- *Descriere:* Urmărirea clicurilor utilizatorilor. Fiecare click trimite o cerere mică către AnalyticsService.
- *Obiectivul testului:* Măsurarea throughput-ului maxim și a ratei de eșec. Sistemul sincron va claca sub mii de cereri/sec, în timp ce broker-ul asincron va acționa ca un buffer (amortizor), permițând procesarea fără pierderi de date.

4. Ce și Cum se Măsoară (Testare și Configurații)

Vom folosi o unealtă de testare de sarcină (ex. **k6**) pentru a executa teste automate pe ambele arhitecturi.

Ce se Măsoară (Metricile Cheie):

1. **Latența (Timpul de Răspuns):** Măsurat în milisecunde (ms), din perspectiva clientului (end-to-end), pentru fiecare scenariu.
2. **Throughput (Debit):** Măsurat ca numărul maxim de cereri pe secundă (RPS) pe care le poate gestiona fiecare arhitectură înainte de a depăși un prag de eroare (ex. 1% erori).
3. **Utilizarea Resurselor:** Monitorizarea **CPU (%)** și a **Memoriei (MB)** pentru fiecare container Docker în timpul testelor de sarcină (folosind docker stats).
4. **Rata de Eșec și Reziliența:** Procentajul de cereri eșuate (ex. erori HTTP 5xx) sub diferite niveluri de stres.

În ce Configurații se Măsoară:

Vom rula suita de teste pe ambele arhitecturi (Sincronă și Asincronă) în 3 configurații distincte:

1. **Configurația Nominală (Baseline):** O sarcină redusă și constantă (ex. 10 utilizatori virtuali) pentru a stabili o performanță de bază.
2. **Configurația de Stres (Stress Test):** O sarcină care crește progresiv (ex. de la 1 la 500 de utilizatori virtuali) pentru a identifica punctul de saturație (throughput maxim) și modul în care sistemul degradează.
3. **Configurația de Eșec (Failure Mode):** O sarcină nominală, dar în care un serviciu non-critic (ex. EmailService sau AnalyticsService) este oprit intenționat sau i se introduce o latență mare, pentru a măsura reziliența și impactul asupra celorlalte servicii.

5. Planificarea Proiectului (Milestones și Sarcini)

Proiectul este împărțit în 4 milestone-uri:

Milestone 1: Definirea Proiectului (Termen: 25.10.2025)

- **Livrabil:** Acest document (prezentarea proiectului).
- **Sarcini:**
    - **Toată echipa (Ioana, Daria, Alexandra, Anana):**
        - Revizuirea, ajustarea și validarea finală a acestui plan de proiect.
        - Definirea schemelor de evenimente (structura mesajelor pentru RabbitMQ).
        - Alegerea finală a stack-ului tehnologic (ex. Node.js/Fastify).

Milestone 2: Implementarea Arhitecturii Sincrone (Termen: 15.11.2025)

- **Livrabil:** O aplicație complet funcțională (toate cele 6 scenarii) folosind apeluri HTTP sincrone. Un fișier docker-compose-sync.yml și codul sursă aferent.
- **Sarcini:**
    - **Ioana:** Setup infrastructură (Git, Docker, imagine de bază, CI de bază). Implementarea Scenariilor 3 (Fan-Out) și 4 (CPU-Intensiv).

- **Daria:** Setup template de bază pentru microservicii (structură foldere, logging, health checks). Implementarea Scenariului 1 (User/Email).
- **Alexandra:** Setup bază de date PostgreSQL. Implementarea Scenariului 5 (Saga - varianta sincronă, cu logică de compensare manuală).
- **Anana:** Setup inițial al proiectului de testare k6. Implementarea Scenariilor 2 (Payment Sim) și 6 (Ingestie Date).
- **Sarcina Colectivă (Săptămâna finală):** Integrarea tuturor serviciilor în docker-compose-sync.yml și asigurarea comunicării corecte între ele.

Milestone 3: Implementarea Arhitecturii Asincrone (Termen: 13.12.2025)

- **Livrabil:** O a doua aplicație, complet funcțională (toate cele 6 scenarii), refactorizată pentru a folosi RabbitMQ. Un fișier docker-compose-async.yml.
- **Sarcini:**
  - **Ioana:** Refactorizarea Scenariilor 3 și 4 pentru a folosi evenimente. Configurarea avansată RabbitMQ (exchanges, queues) în docker-compose-async.yml.
  - **Daria:** Refactorizarea Scenariului 1 pentru a publica/consuma evenimente.
  - **Alexandra:** Refactorizarea Scenariului 5 pentru a implementa coregrafia Saga (prin abonarea la evenimente de succes/eșec).
  - **Anana:** Refactorizarea Scenariilor 2 și 6 pentru a folosi cozi de mesaje. Adaptarea scripturilor k6 pentru a testa fluxurile asincrone (ex. verificarea statusului la un alt endpoint).

Milestone 4: Testare, Analiză și Redactare Articol (Termen: 17.01.2026)

- **Livrabil:** Articolul științific final (format PDF) și codul sursă complet, curățat și documentat.
- **Sarcini:**
  - **Anana:**
    - Executarea finală a suitelor de teste k6 pe ambele arhitecturi în toate cele 3 configurații (Nominal, Stres, Eșec).
    - Colectarea și centralizarea datelor brute (latență, throughput, erori).
    - Crearea graficelor și tabelelor comparative.
    - Redactarea secțiunilor "Metodologia de Testare" și "Abstract" ale articolului.
  - **Ioana:**
    - Executarea testelor de monitorizare a resurselor (CPU/Memorie) și colectarea datelor.
    - Redactarea secțiunilor "Introducere" și "Arhitectura Sistemului" (descriind implementările Sincronă și Asincronă).
    - Analiza și redactarea rezultatelor pentru Scenariile 3, 4, 6.
  - **Daria:**
    - Analiza și redactarea rezultatelor pentru Scenariile 1 și 5.
    - Redactarea secțiunilor "Concluzii" și "Discuții" (interpretarea rezultatelor).
  - **Alexandra:**

        - Analiza și redactarea rezultatelor pentru Scenariul 2.
        - Redactarea secțiunilor "Lucrări Viitoare" (Future Work) și "Managementul Datelor" (consistența datelor).
    - **Sarcina Colectivă (Ultimele 2 săptămâni):** Asamblarea tuturor secțiunilor, revizuirea integrală a articolului, corectarea și formatarea finală.

Prin compararea directă a celor două stiluri arhitecturale (sincron și asincron) pe 6 scenarii de lucru diverse și relevante, proiectul va genera date concrete care evidențiază compromisurile practice ale fiecărei abordări.