TeamWork Chat
Actor Model
vs.
Arhitectura clasica client-server

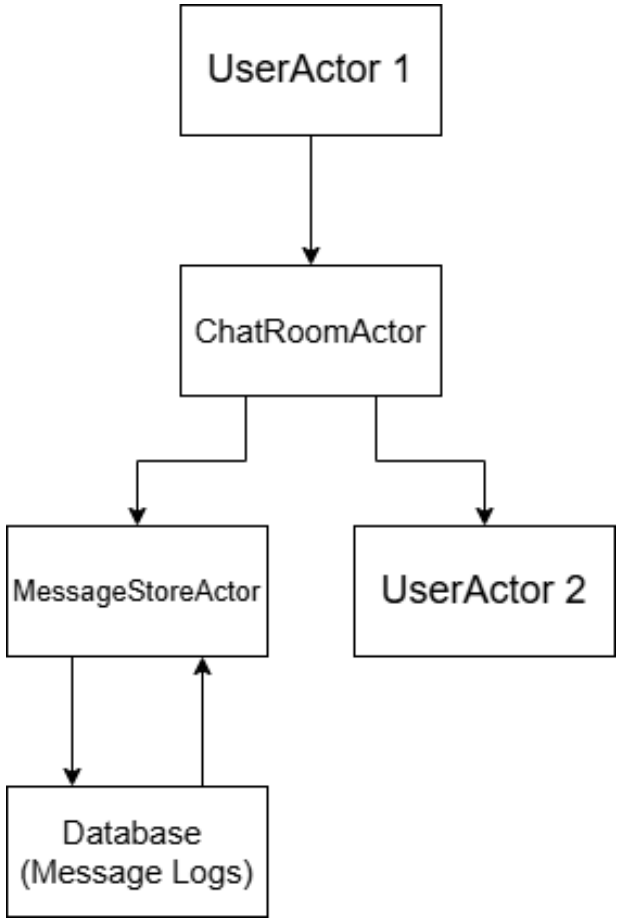
Alexandra Camelia Stefan, Toma Andrei Sacuiu

Scop

- Implementarea unei aplicații chat
- Organizarea activității in sprint-uri
- Impărțirea implementării in sarcini per membru al echipei
- 2 variante ale aplicației:
  - Arhitectură clasică client-server
  - Implementare bazată pe modelul Actor
- Compararea calitativă si cantitativă a celor 2 versiuni
- Folosirea unor metrici adecvate pentru performanță, scalabilitate, mentenanță și ușurința în dezvoltare.

Descrierea aplicatiei

- Sistem de comunicație în timp real între utilizatori
- Permite trimiterea și recepționarea de mesaje text
- Crearea unui mediu distribuit, analizând impactul arhitecturii folosite asupra performanței și scalabilității
- Permiterea unui numar considerabil de utilizatori sa primeasca si sa trimita in acelasi timp date



Flux al utilizării

- Utilizatorul pornește clientul aplicației.
- Clientul se conectează la server / sistemul de actori.
- Utilizatorul poate trimite mesaje către alți utilizatori conectați.
- Serverul / actorii primesc mesajele și le distribuie către destinatari.
- Clientul afișează mesajele primite în interfața utilizatorului.

Beneficii model Actor

- **Concurență nativă:** Actorii gestionează mesaje asincrone fără blocări.
- **Scalabilitate ridicată:** Ușor de extins la un numar mare de utilizatori simultan.
- **Izolare a stării:** Fiecare actor își gestionează propria stare internă, reducând erorile de concurență.
- **Rezilienta la erori:** Actorii pot fi reporniți sau înlocuiți fără afectarea întregului sistem.
- **Flexibilitate și modularitate:** Comportamente complexe pot fi împărțite în actori simpli și reutilizabili.

Comparatie: Actor Model vs. Client-Server

- Evaluarea diferențelor între cele două modele pentru aplicația de chat, în termeni de performanta, scalabilitate si usurinta in implementare.
- In continuare, detaliem comparatia in doua categorii:
  - Calitativa
  - Cantitativa

Analiza calitativa

- Vom oberva diferenta intre cele doua implemntari privind urmatoarele aspecte:
  - Concurenta
  - Rezistenta la erori
  - Scalabilitate
  - Usurinta implementarii
  - Modularitate

Analiza cantitativa

- **Timp de răspuns:** Măsurăm timpul mediu și maxim de la trimiterea la primirea unui mesaj.
- **Throughput:** Numărul de mesaje procesate pe secundă.
- **Consumul de resurse:** CPU și memorie utilizate la diferite numere de clienți.
- **Număr de utilizatori concurenți susținuți:**
  - Teste cu un număr variat de clienți: 10, 50, 100+
- **Mesaje scurte vs mesaje lungi**
- **Monitorizare resurse și timpi cu instrumente de profilare** (Prometheus, Grafana etc.)

Organizarea pe sprint-uri

- Sprint 1:

TASK	PEOPLE
Documentare despre Actor model (istoric, beneficii, concept, implementari)	Toma, Alexandra
Documentare despre pattern-uri alternative	Toma, Alexandra

- Sprint 2:

TASK	PEOPLE
Software design pentru implentarea ineficienta	Toma, Alexandra
Implementarea aplicatiei fara utilizarea Actor Model	Toma, Alexandra

Organizarea pe sprint-uri

- Sprint 3:

TASK	PEOPLE
Software design pentru implentarea cu Actor Model	Toma, Alexandra
Implementarea aplicatiei utilizand Actor Model	Toma, Alexandra

- Sprint 4:

TASK	PEOPLE
Bug fixes, minor improvements	Toma, Alexandra

Organizarea pe sprint-uri

- Sprint 5:

TASK	PEOPLE
Testare performanta, scalabilitate, mantainability	Toma, Alexandra
Comparatie, concluzii finale	Toma, Alexandra

- Note: Fiind o echipa de dimensiune redusa, task-urile vor fi realizate de ambii membrii, avand subtask-uri si impartindu-le corespunzator (i.e. peer programming)