Documentatie

Tema 2

Nume: Saca Victor-Valentin

1. Obiectivul temei

(i) Obiectivul principal al temei este dezvoltarea unei aplicații de simulare a unui sistem de gestionare a sarcinilor pentru un mediu de serviciu, utilizând tehnici de programare orientată pe obiecte și planificare a resurselor.

(ii) Obiectivele secundare:

- Implementarea unui scheduler eficient pentru distribuirea sarcinilor către servere.
- Dezvoltarea unei interfețe grafice pentru afișarea și monitorizarea sarcinilor și a serverelor în timp real.
- Realizarea unui sistem de înregistrare a evenimentelor pentru analiza ulterioară a performanței aplicației.

2. Analiza problemei, modelare, scenarii, cazuri de utilizare

Cerințe funcționale:

- Simularea sosirii și procesării sarcinilor de către servere.
- Monitorizarea și afișarea stării serverelor și a sarcinilor într-o interfață grafică.

Cerințe non-funcționale:

- Performanță: Simularea să fie realizată în timp real, fără întârzieri semnificative.
- Fiabilitate: Asigurarea că aplicația funcționează corect și rezistent la erori.

2. Cazuri de utilizare:

1. Utilizatorul deschide aplicația de simulare.
2. Utilizatorul setează parametrii simulării (timpul maxim, timpul minim de procesare, etc.).
3. Simularea începe, iar sarcinile sunt distribuite către servere conform algoritmului de planificare.
4. Utilizatorul poate vizualiza starea actuală a sarcinilor și a serverelor în interfața grafică.
5. Simularea se oprește automat după expirarea timpului maxim setat de utilizator.

3. Proiectare

- Interfețele grafice vor fi proiectate pentru a afișa informațiile relevante despre starea simulării.
- Se vor utiliza structuri de date eficiente pentru stocarea și gestionarea sarcinilor și a serverelor.

4. Implementare

- Clasele principale ale aplicației, cum ar fi **SimulationManager**, **Scheduler**, **Server**, și **Task**, vor fi implementate conform proiectării OOP.

SimulationManager: contine bulca repetitive pentru simulare, folosind fire de executie, si are metoda generateNRandomTasks, care genereaza client cu campuri aleatorii, avand arrivalTime si serviceTime.

Scheduler: trimite clientii catre cozi, respecandu-se strategia aleasa

Task: este clasa pentru un client, avand un id, arrivalTime si serviceTime, pentru fiecare client

Server: reprezinta cozile la care sunt asignati clientii

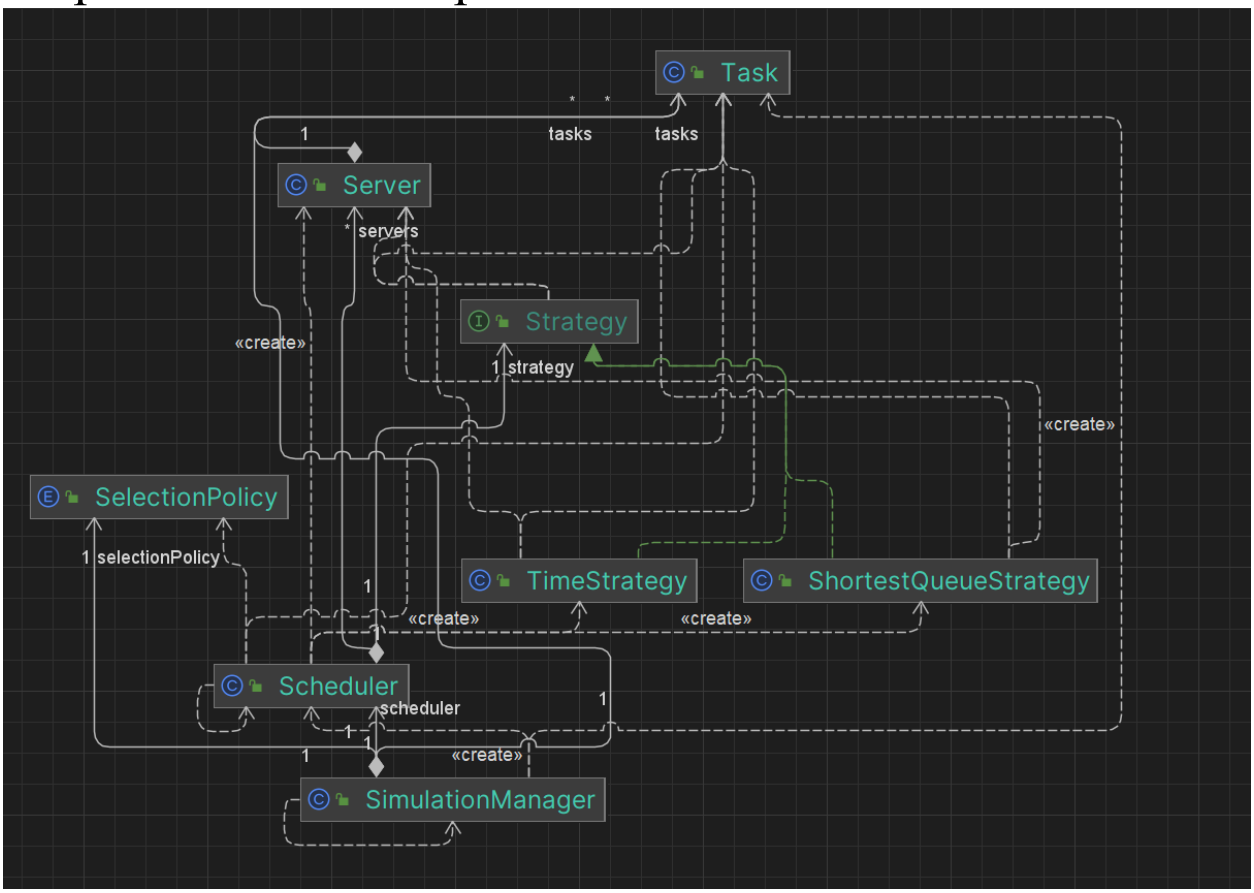
Clasa contine o infinita pentru thread-uri, prin care se proceseaza clientii in cozi unul cate unul, tinand cont de waitingTime.

Scheduler: Creeaza Q threaduri pentru Q cozi si schimba strategia selectata de utilizator. Aici este implementata si metoda de dispatch task, care trimite trimite un client la coada lui corespunzatoare, si seteaza timpul de asteptare a clientului respective.

Strategy: este o interfata care contine o metoda de addTask, prin care se adauga clientul in coada folosind strategia selectata.

ShortestQueueStrategy: este o clasa care implementeaza interfata Strategy, si implementeaza metoda addTask, dupa strategia de asginare in coada la coada cea mai scurta la momentul respective.

TimeStrategy: reprezinta cealalta clasa care implementeaza Strategy, dupa strategia de asignarea clientilor in coada, dupa strategia de asignare in cozi dupa timpul minim de asteptare.



5. Concluzii

- În urma dezvoltării temei, s-au învățat principii importante ale programării orientate pe obiecte și gestionării resurselor în timp real.
- Dezvoltarea ulterioară a aplicației ar putea include optimizări ale algoritmilor de planificare și extinderea funcționalității interfeței grafice.

6. Bibliografie

-https://stackoverflow.com/

- https://www.jetbrains.com/

- https://dsrl.eu/courses/pt/ (resursele cu fisiere pdf ale facultatii)