Proiect 1 PPD

Mihai Ungureanu, Mihai Vulcan

# Cerinta

O sala de spectacole vinde bilete la spectacolele organizate printr-o aplicatie client-server.

Sala organizeaza cel mult un spectacol pe zi.

Sala de concerte are un numar maxim - ‘nr\_locuri’ - de locuri numerotate de la 1 la ‘nr\_locuri’.

Pentru fiecare spectacol avem informatii de tip (data, titlu, pret\_bilet).

Permanent sala mentine o evidenta actualizata pentru:

- informatii despre bilete pentru fiecare spectacol - (ID\_spectacol, lista\_locuri\_vandute);

- vanzarile efectuate: lista de vanzari; vanzare = (data\_vanzare, ID\_spectacol, numar\_bilete, lista\_locurilor) ;

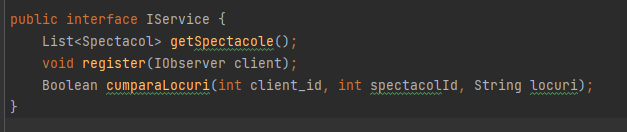
- soldul total (suma totala incasata).

Periodic sistemul (2 cazuri testare: 5, 10 secunde) face o verificare a locurilor vandute prin verificarea

corespondentei corecte intre locurile libere si vanzarile facute, sumele incasate per vanzare si soldul total.

# Implementare networking

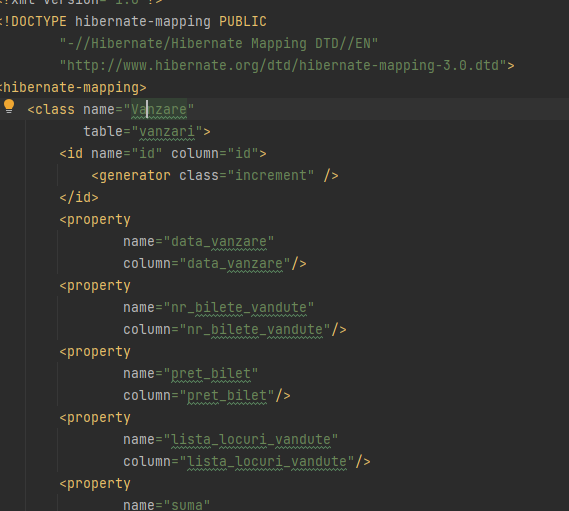
Am folosit Spring RMI pentru conectarea dintre server si client. Se foloseste un proxy prin interfata IService care defineste functiile cu care lucreaza serverul



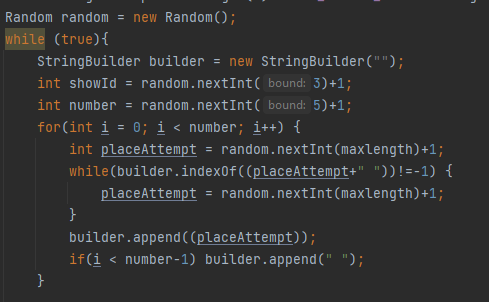
Functia getSpectacole() este folosita de client pentru generarea requesturilor aleatoare astfel:

# Persistenta datelor si dependinte

Datele sunt persistate intr-o baza de date folosind PostgreSQL. Pe server se foloseste Hibernate pentru maparea si accesul la date. Folosim Log4j2 pentru mentinerea log-urilor, spring rmi pentru networking



# Client



Clientul trimite astfel requesturi din 2 in 2 secunde serverului cerand un numar variabil de locuri alese aleator. Clientul implementeaza interfata IObserver, care este folosita de server pentru identificarea acestuia si notificarea cu privire la inchiderea acestuia.

# Server

Serverul mentine o lista a clientilor cunoscuti pentru a-i anunta cand se inchide astfel:



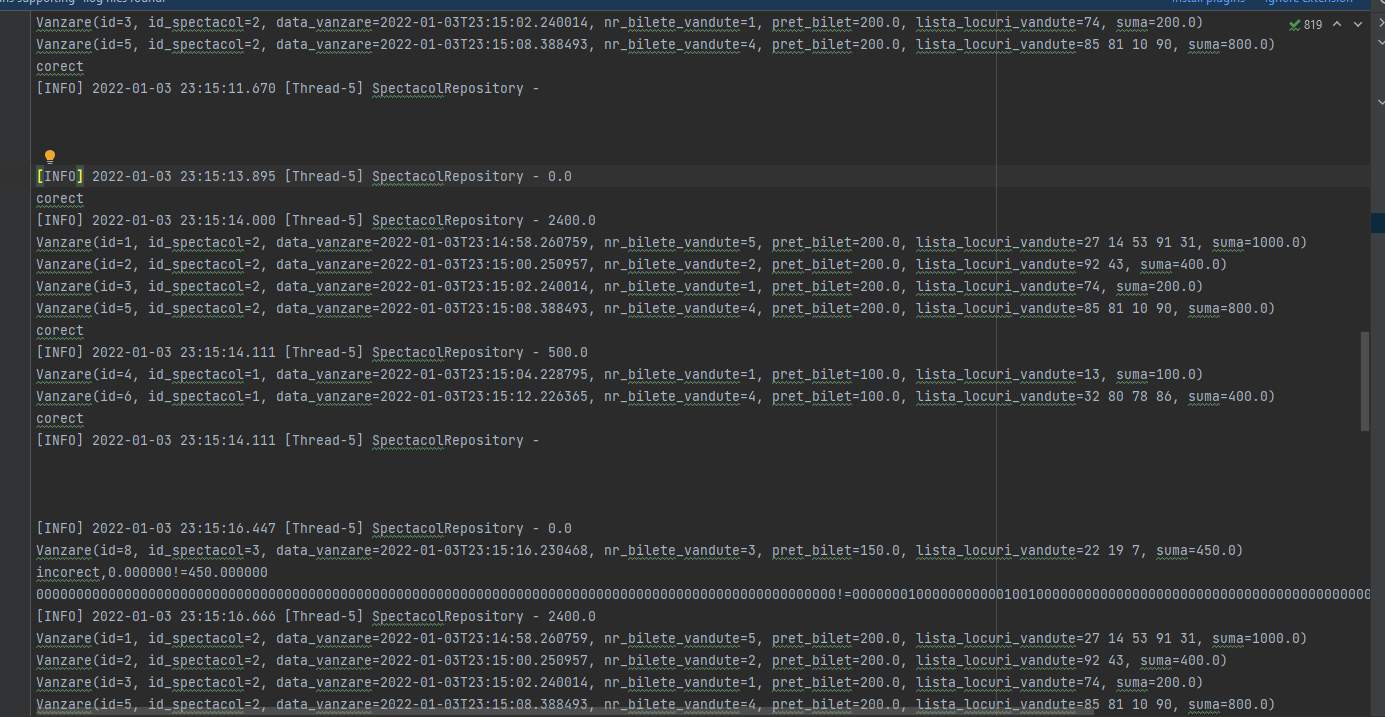
De asemenea in implementarea serverului am folosit un Thread Pool

ExecutorService executor = Executors.*newFixedThreadPool*(10);

Iar fiecare request este servit prin intermediul acestui executor.

# Verificare si Logging

La fiecare X secunde, serverul realizeaza o verificare a datelor din baza de date. Se vor recalcula sumele totale si locurile libere pe baza vanzarilor si se vor compara cu datele mentinute in obiectele de tip Spectacol. Intr-un fisier numit app.log se mentine evidenta acestor verificari cu data, ora, suma totala pentru spectacol, lista de vanzari, si “corect”/”incorect” in functie de rezultatul verificarii.



# Sincronizare

1. Clientul va trimite requestul pe un thread separat, iar pe thread-ul principal va putea fi apelata din partea cealalta functia ServerClosed() din interfata IObserver pe care o implementeaza.
2. Serverul foloseste functii declarate ca Synchronized pentru tratarea cazurilor de concurenta. Astfel, functiile sensibile cum ar fi buy() blocheaza obiectul pentru a nu se modifica de doua ori obiectul in acelasi timp, trecand peste verificarile de rigoare intr-unul din aceste cazuri. De asemenea, se foloseste si un mecanism de future-promise