**Proiect 1 – Client-Server**

Obiectiv:

- Folosirea executiei concurente prin apeluri asincrone.

- Folosirea mecanismelor: future/promises si thread\_pool.

- Analiza imbunatatirii performantei executiei unei aplicatii (de tip business) prin

programare concurenta.

**Sala spectacole**

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.

Sistemul foloseste un mecanism de tip ‘Thread-Pool’ poentru rezolvarea a taskurilor. Pentru acest mecanism este nevoie sa se poata specifica numarul de threaduri folosite.

Pentru testare se va considera ca fiecare client initiaza/creeaza la interval de 2 sec o noua cerere de vanzare

bilete folosind date generate aleatoriu (nr\_de\_bilete, locuri) si se primeste de la server o notificare – vanzare reusita sau vanzare nereusita. Nu este necesara interfata grafica!

Pentru verificare se cere salvarea pe suport extern (fisier text) a rezultatelor operatiilor de verificare executate periodic: data, ora, sold\_per spectacol, lista vanzarilor per spectacol, ‘corect/incorect’.

Serverul se inchide dupa un interval de timp precizat si notifica clientii activi referitor la inchidere.

Model

Spectacol (ID\_spectacol, data\_spectacol, titlu, pret\_bilet, lista\_locuri\_vandute, sold)

Vanzare (ID\_spectacol, data\_vanzare, nr\_bilete\_vandute, lista\_locuri\_vandute, suma)

Sala(nr\_locuri, Lista<Spectacol>, Lista<Vanzari>)

Taskuri posibile

* Vanzare bilete
* Verificare

Limbajul de implementare: la alegere!!!

Pentru implementarea conexiunii intre client si server se pot folosi oricare dintre tehnologiile folosite la cursurile anterioare!!!

**Testare:**

Nr\_locuri =100;  
3 spectacole (S1, S2, S3)

S1 pret\_bilet=100;

S2 pret\_bilet=200;

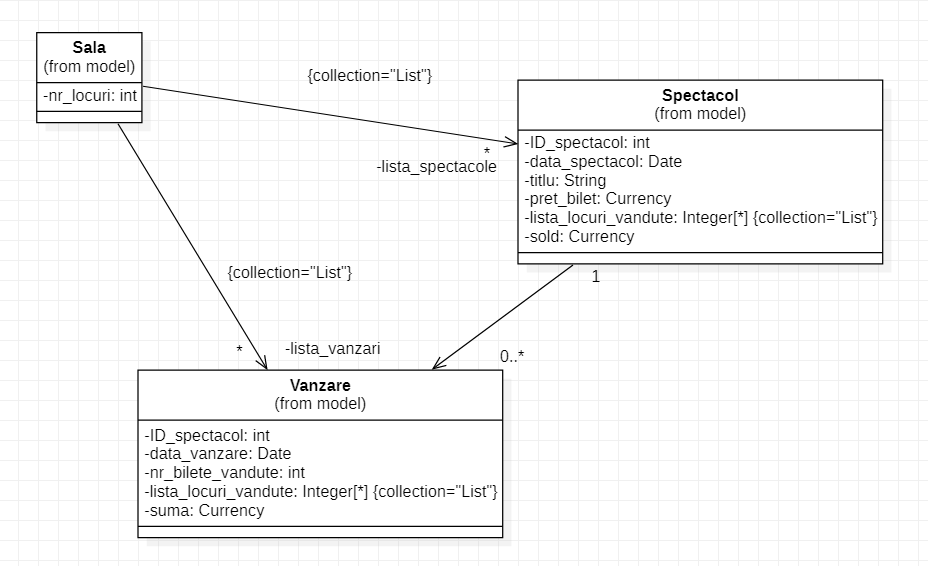
S3 pret\_bilet=150;

Serverul lucreaza 2 minute.

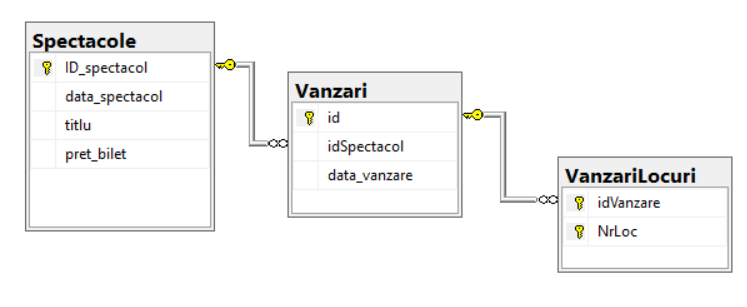
Deadline: saptamana 13

Indicatii:

Un model posibil este cel din fig. de mai jos:



Salvarea datelor poate fi facuta intr-o baza de date de tipul:



**Sau in fisiere text:**

**Spectacol.txt:**

ID\_spectacol1, data\_spectacol1, titlu1, pret\_bilet,

1,5,6,7,3 ## lista\_locuri\_vandute

sold;

ID\_spectacol2, data\_spectacol2, titlu2, pret\_bilet,

1,2,3,6,7,8

sold;

.

.

**Vanzare.txt:**

ID\_spectacol1, data\_vanzare1, nr\_bilete\_vandute,

1,5 ## lista\_locuri\_vandute

suma;

ID\_spectacol1, data\_vanzare2, nr\_bilete\_vandute,

6,7,3

suma;

ID\_spectacol2, data\_vanzare1, nr\_bilete\_vandute,

1,2,3

suma;

.

.

**Sala.txt:** nr\_locuri,

ID\_spectacol1, ID\_spectacol2, ID\_spectacol3,…etc ## lista\_spectacole

ID\_spectacol1, data\_vanzare1; ID\_spectacol1, data\_vanzare2, …etc ## lista\_vanzari

ID\_spectacol2, data\_vanzare1; ID\_spectacol2, data\_vanzare2, …etc