Bontida are nevoie de ajutor

Student: Gabor George Cătălin
Grupa: 30239

March 25, 2018

Cuprins

1 Analiza cerințelor

1.1 Specificare cerinte

Se cere realizarea unei aplicatii desktop pentru vanzarea de bilete pentru festivalul Electric Castle. Aceasta va avea doua tipuri de utilizatori:

- administrator;
- casier.

si patru tipuri de bilete:

- Ziua 1, pret 200 RON;
- Ziua 2, pret 200 RON;
- Ziua 3, pret 200 RON;
- Toate zilele, pret 400 RON.

De asemenea, in fiecare zi pot fi un numar maxim de oameni la festival.

1.2 Cerinte functionale

Aplicatia trebuie sa puna la dispozitie utilizatorilor urmatoarele functionalitati:

- autentificarea prin LOGIN si redirectionarea la operatii specifice utilizatorului;
- pentru casier:
  - vanzarea de bilete si crearea automata a unui fisier pe disk cu detalile biletului. In momentul cand este vandut un bilet urmatoarea constrangere trebuie respectate: Biletele din Ziua x + Biletele Toate zilele <MAX_CAPACITY;
  - Vizionarea istoricului biletelor vandute de catre el ordonate dupa timp+data (ultimele bilete vandute fiind primele).
- pentru administrator:
  - operatii CRUD pe casier;
  - Modificarea numarului MAX_CAPACITY;
  - Raport cu numarul de bilete vandute per casier;
  - Raport cu incasarile pe zi + totale bazate pe biletele vandute.

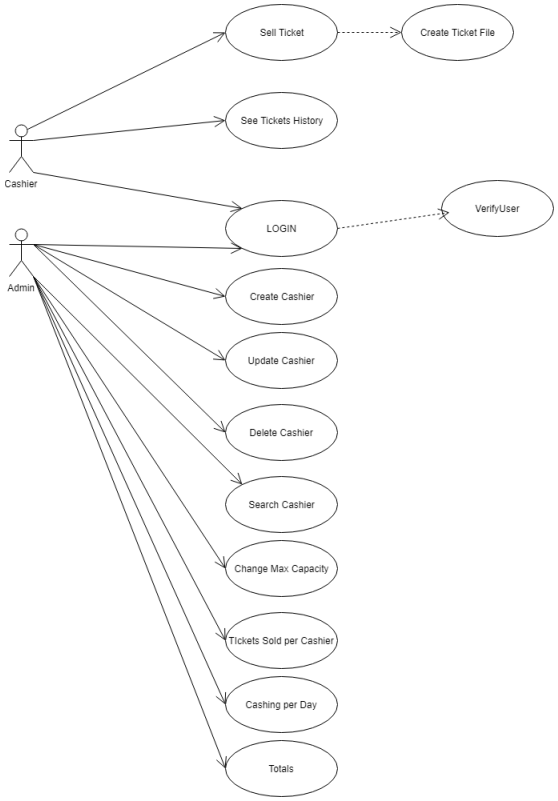
1.3 Cerinte non-functionale

Urmatoarele cerinte trebuie respectate:

- timpul de raspuns al sistemului pentru orice operatie trebuie sa fine mai mic de 2 secunde;
- arhitectura sistemului trebuie sa permina adaugarea de noi operatii sau modificarea celor curente;
- sistemul trebuie sa fie interactic, sa ofere mesaje utilizatorului pentru al ghida.

2 Use Case Model

Use case: Vanzare bilet;
Level: User-goal;
Actor princital: Casier;
Scenariu de succes: casierul se logheaza, alege tipul biletului si apasa butonul pentru vanzare. Un mesaj este afisat pentru a confirma reusita operatiei;
Extensii: Operatia nu poate esua decat daca intervine o problema in baza de date.



3 Design-ul arhitectural al sistemului

3.1 Descrierea modelului arhitectural

Este utilizat pattern-ul arhitectural Layers.
Aplicatia este compusa din trei nivele:

- Presentation - contine interfata cu utilizatorul;
- Logic - structurile de date necesare pentru realizarea operatilor;
- DataAccess - conexiunea cu baza de date si extragerea, schimbarea datelor din aceasta.

3.2 Diagrame

Urmatoarea diagrama prezinta structura pachetelor din sistem :

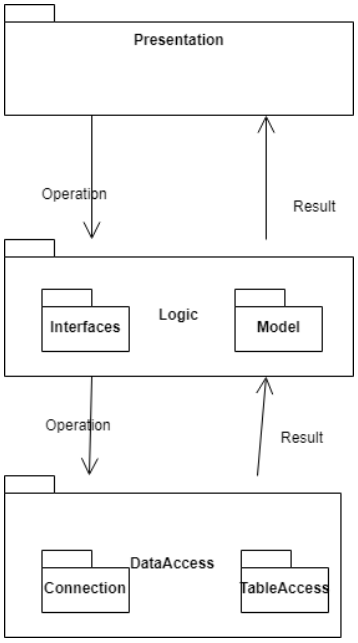


Diagrama de componente:

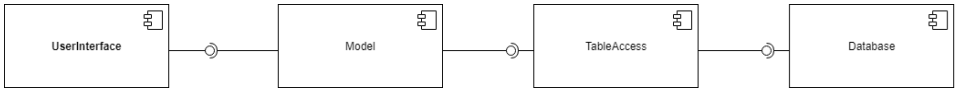
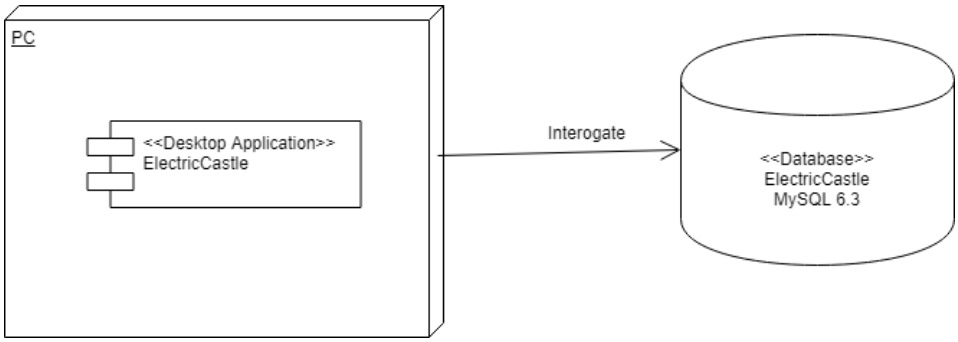
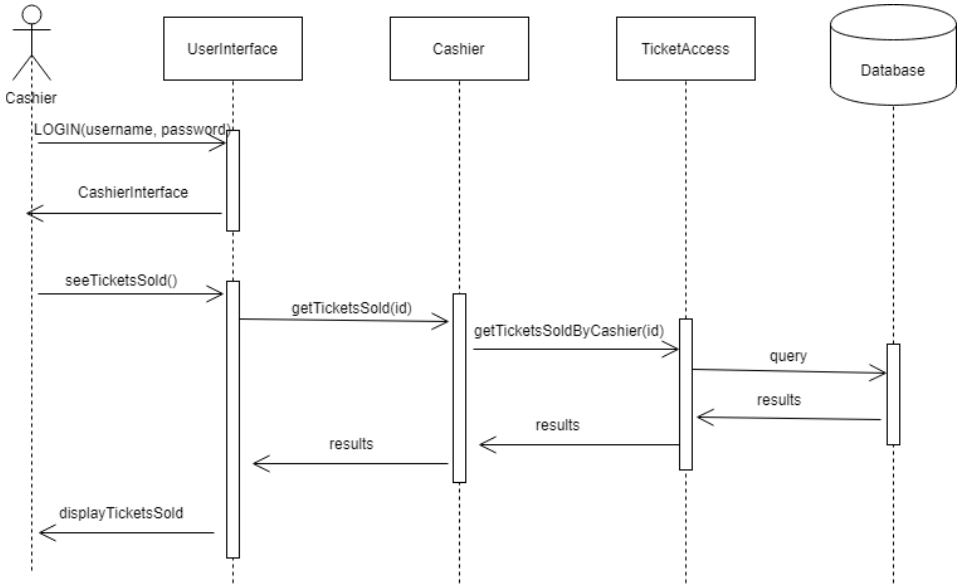


Diagrama de deployment:



4 Diagrame de secventa UML

Consideram scenariul in care un anumit casier doreste sa vada lista cu biletele vandute de el. Avem urmatoarea diagrama de secventa:



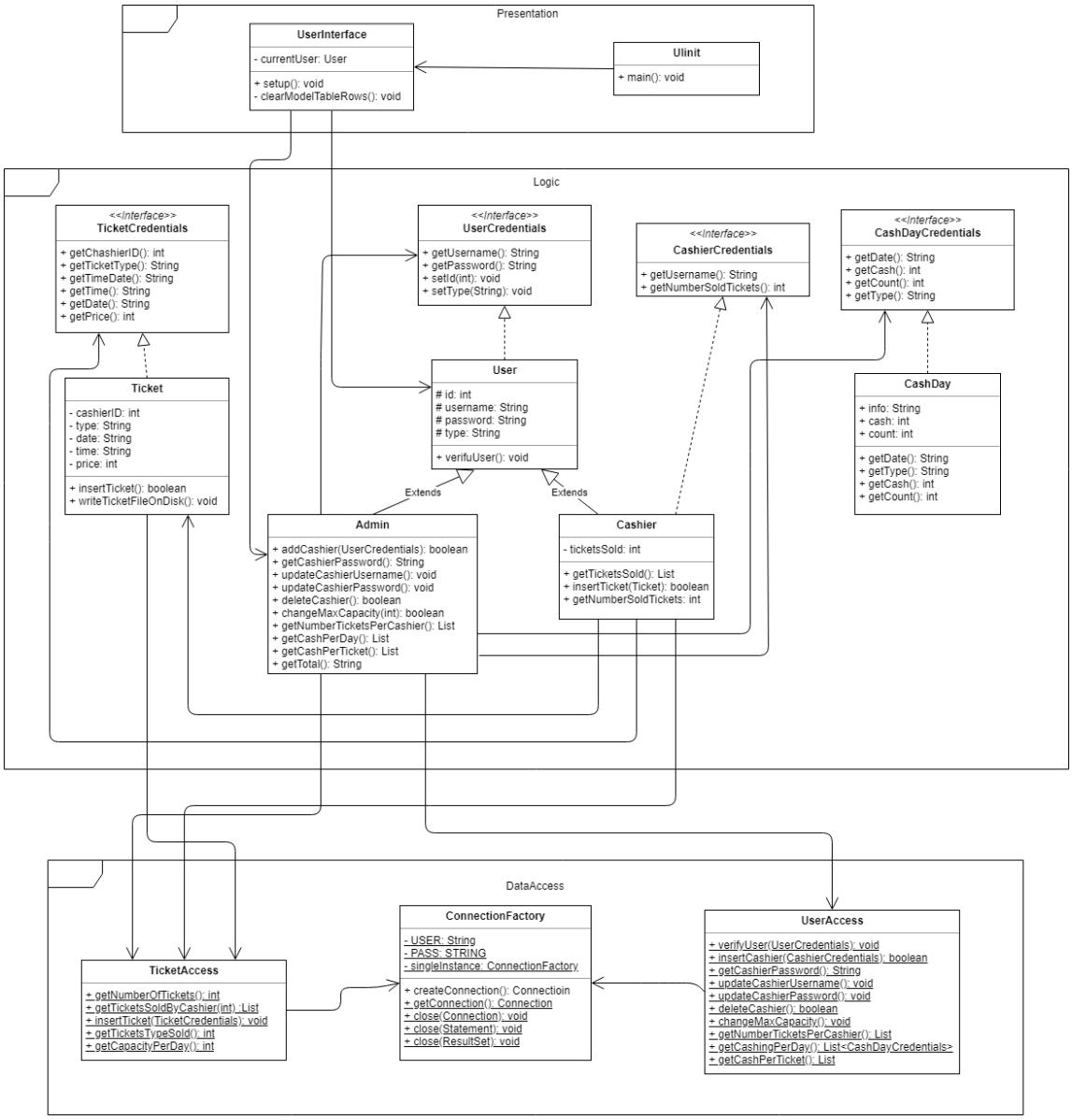
5 Design-ul claselor

5.1 Descriere design pattern-uri utilizate

Au fost utilizate urmatoarele:

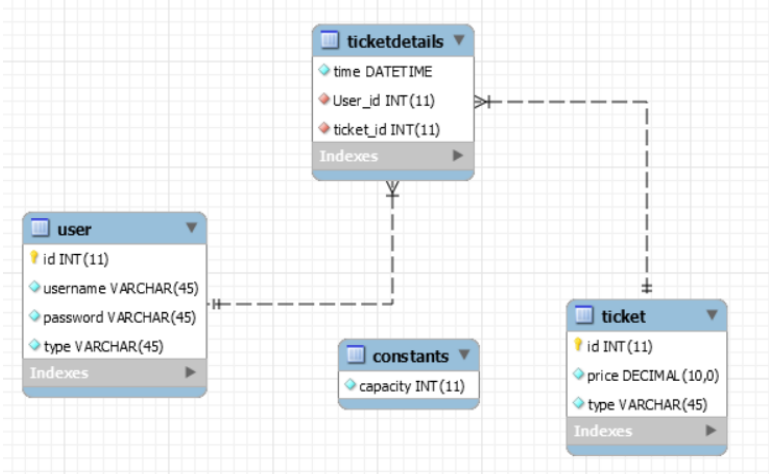
- Data Mapper - Clasele User, Ticket, Cashier si Admin nu contin interogari SQL sau connexiuni la baza de date. Toate datele care trebuie scrise sau citite sunt transferate prin clasele TicketAccess si UserAccess;
- Singleton - Exista o singura instanta a clasei ConnectionFactory;
- Delegation - Clasele acceseaza metode doar ale claselor vecine.

5.2 Diagrama de clase



6 Model de date

Baza de date in care sunt stocate informatiile este urmatoare:



Dupa cum se poate, vedea exista patru tabele:

- User - atributul "type" este folosit pentru a diferentia intre casieri si administrator;
- Ticket - modeleaza tipurile de bilete, atributul "type" reprezinta perioada de access la festival, exemplu "Day1";
- Ticketdetails - folosit pentru a evita relatia de n la n intre tabelele User si Ticket si asociaza casierul cu tipul biletului vandut;
- Constants - are un singur rand si un singur atribut pentru a retine capacitatea maxima de persoana pentru festival.

7 Testarea sistemului

S-a folosit o strategie de testare incrementala, functionalitatile au fost verificate pe masura ce au fost adaugate in sistem. Astfel, principalele metode de testare au fost data-flow (au fost date anumite intrari si s-a observat raspunsul sistemului), white-box (codul a fost disponibil) si boundary analysis (pentru valoarea capacitatii intr-o zi de festival).

8 Bibliografie

Pentru UML:

- `https://www.youtube.com/watch?v=OkC7HKtiZCO&list=PLGLfVvz_LVvQ5G-LdJ8RLqe-ndo7QITYc`

Pentru anumite erori, pattern-uri:

- `https://stackoverflow.com/`
- `https://www.youtube.com/`
- `https://en.wikipedia.org/wiki/Main_Page`