

Agenție de rezervare bilete de tren

PROIECTARE SOFTWARE

Ardeleanu Mădălin-Florin

GRUPA 30233

2020

CUPRINS

1. Specificarea cerințelor	3
2. Instrumente utilizate	6
3. Etapa de analiză	7
3.1. Diagrama cazurilor de utilizare	7
3.2. Diagrame de activități	9
4. Etapa de proiectare	19
4.1. Diagrame de clase	19
4.2. Diagrame de secvență	23
4.3. Diagrama bazei de date	32
5. Prezentarea aplicației	33

1. Specificarea cerințelor (enunț problemă)

Tema 1:

Dezvoltați (analiză, proiectare, implementare) o **aplicație desktop** care poate fi utilizată de către o agenție de rezervare bilete de tren.

Aplicația va avea 3 tipuri de utilizatori: călător, angajat și administrator.

Utilizatorii de tip călător pot efectua următoarele operații fără autentificare:

- Vizualizarea listei trenurilor după stație de plecare, destinație, durată;
- Vizualizarea listei trenurilor dintre 2 locații, inclusiv preț și disponibilitate locuri libere;
- Căutarea unui tren după număr.

Utilizatorii de tip angajat pot efectua următoarele operații după autentificare:

- Toate operațiile permise utilizatorilor de tip călător;
- Operații CRUD în ceea ce privește persistența trenurilor și biletelor vândute;
- Vânzarea unui bilet către un călător;
- Vizualizarea unor statistici legate de biletele vândute: procente după stație de plecare, destinație, preț utilizând grafice (structură radială, structură inelară, de tip coloană, etc.);
- Salvare rapoarte / liste cu informații despre trenuri în mai multe formate: csv, xml, json.

Utilizatorii de tip administrator pot efectua următoarele operații după autentificare:

- Operații CRUD pentru informațiile legate de utilizatorii de tip angajat.

Constrângeri ale aplicației:

- Datele utilizate în aplicație vor fi socate într-un fișier binar;
- Utilizați şablonanele de proiectare creaționale **Factory Method** și **Builder**.

Cerințe:

În **faza de analiză** realizați diagrama cazurilor de utilizare și diagrame de activități corespunzătoare tuturor cazurilor de utilizare.

În **faza de proiectare** realizați diagrama de clase și diagrame de secvență corespunzătoare tuturor cazurilor de utilizare.

În **faza de implementare** scrieți cod pentru îndeplinirea tuturor funcționalităților precizate de diagrama cazurilor de utilizare utilizând unul dintre următoarele limbaje de programare: C#, C++, Java, Python.

Finalizarea temei va consta în predarea unui director ce va cuprinde:

- Un fișier word care cuprinde numele studentului, grupa, enunțul problemei și justificarea limbajului de programare ales.
- Un fișier cu diagramele UML realizate;
- Directorul cu aplicația implementată.

Tema 2:

Transformați aplicația desktop implementată la **tema 1** într-o aplicație client-server. Mare parte din modelul logic (inclusiv persistența) va face parte din server, mai puțin partea care poate fi determinată din datele primite de la server (generare rapoarte, statistică, etc.).

În **faza de analiză** realizați diagrama cazurilor de utilizare.

În **faza de proiectare** realizați diagrama de clase corespunzătoare aplicației **server** și diagrama de clase corespunzătoare aplicației **client**.

În **faza de implementare** scrieți cod pentru îndeplinirea tuturor funcționalităților precizate de diagrama cazurilor de utilizare utilizând unul dintre următoarele limbaje de programare: C#, C++, Java, Python.

Finalizarea temei va consta în predarea unui director ce va cuprinde:

- Un fișier word care cuprinde numele studentului, grupa, enunțul problemei și instrumentele utilizate.
- Un fișier cu diagramele UML realizate;
- Directorul cu aplicația implementată.

Constrângeri ale aplicației:

- Datele utilizate în aplicație vor fi socate într-o bază de date (MySQL, SQL Server, Oracle, etc.).
- Se cere utilizarea arhitecturii client/server, **NU** web client/server.

Tema 3:

Optimizați aplicația client/server implementată la **tema 2** astfel încât să utilizați şablonul de proiectare comportamental **Observer** în dezvoltarea aplicației client. În plus, interfața grafică a aplicației client va fi disponibilă în cel puțin 3 limbi de circulație internațională (implicit limba română).

În **faza de analiză** realizați diagrama cazurilor de utilizare și diagramele de activități corespunzătoare tuturor cazurilor de utilizare.

În **faza de proiectare** realizați:

- diagrama de clase corespunzătoare aplicației **server**;
- diagrama de clase corespunzătoare aplicației **client**;
- diagrame de secvență corespunzătoare tuturor cazurilor de utilizare.

În **faza de implementare** scrieți cod pentru îndeplinirea tuturor funcționalităților precizate de diagrama cazurilor de utilizare utilizând unul dintre următoarele limbaje de programare: C#, C++, Java, Python.

Finalizarea temei va consta în predarea unui director ce va cuprinde:

- Un fișier word care cuprinde numele studentului, grupa, enunțul problemei, instrumentele utilizate și detalii despre modul în care a fost utilizat şablonul de proiectare **Observer**.
- Un fișier cu diagramele UML realizate;
- Directorul cu aplicația implementată.

Observație:

Pentru persistență se va utiliza o bază de date relatională (MySQL, SQL Server, Oracle, etc.). Se va utiliza baza de date doar în aplicațiile unde este necesară persistență.

2. Instrumente utilizate

Pe parcursul celor 3 teme realizate, am lucrat doar în limbajul Java, mai precis în mediul de lucru Eclipse 2018/09, unde am folosit versiunea 11 de Java SE Development Kit.

La prima temă a fost nevoie de folosirea unui fișier extern de jar, anume json-simple-11.jar, acesta fiind folosit pentru tipărirea trenurilor existente într-un fișier cu extensia json, aceste trenuri fiind salvate inițial într-un fișier txt.

La a 2-a temă, s-a utilizat din nou jar-ul de mai sus, doar ca de această dată, trenurile erau salvate într-o bază de date, spre deosebire de prima temă, unde toate datele erau salvate în fișiere de persistență.

În plus, s-a mai utilizat și jar-ul mysql-connector-java-5.1.48.jar, deoarece cum am precizat, de această dată, datele erau salvate într-o bază de date, iar pentru stocarea acestor date era necesară mai întâi o conectare la baza de date respectivă.

Iar în final pentru ultima temă, toate uneltele precizate mai sus au rămas valabile, adică s-au folosit în continuare, însă a fost necesară și folosirea a două jar-uri suplimentare: activation-1.1.1.jar și javax.mail.jar ce s-au folosit pentru configurarea, conectarea și trimiterea unui mail din aplicația pe care am conceput-o, acest lucru fiind realizat și cu ajutorul unui design pattern numit Observer.

3. Etapa de analiză

3.1 Diagrama cazurilor de utilizare

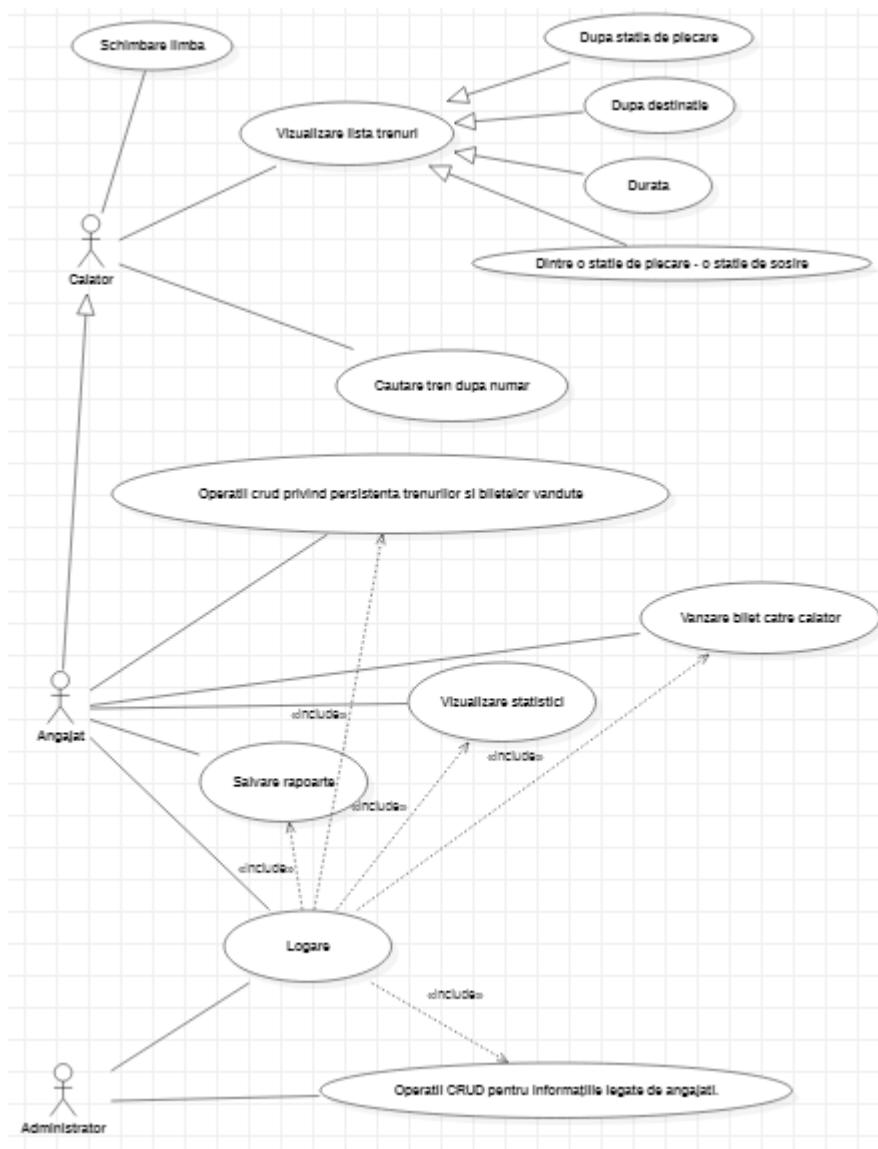


Fig 1. Diagrama cazurilor de utilizare

După cum se poate observa în diagrama de use-case, în aplicația mea există 3 tipuri de utilizatori (actori) : un călător, un angajat și un administrator.

Călătorul poate fi percepțut ca un guest, toate persoanele care vor rula aplicația inițial vor fi percepute ca niște călători, chiar dacă unele dintre persoane au un grad mai mare de acces, sistemul inițial nu cunoaște acest lucru decât după autentificarea unui angajat sau administrator, abia atunci li se oferă anumite drepturi ‘suplimentare’.

Revenind, călătorii sunt acei utilizatori care pot să efectueze anumite operații precum:

- să selecteze o limbă pentru interfața grafică
- să vizualizeze o anumită listă de trenuri obținută după anumite criterii de filtrare precum: stația de plecare, stația de sosire, durata cursei sau stația de plecare – stația de sosire
- să caute un tren după identificatorul unic al acestuia

Angajații sunt acei utilizatori care după autentificare, pe lângă toate operațiile permise călătorilor, li se permite efectuarea unor operații suplimentare precum :

- operații de inserare, selectare, modificare sau ștergere asupra trenurilor și biletelor vândute care sunt stocate în baza de date
- vânzarea unui bilet către un călător
- vizualizarea unor statistici cu privire la prețurile biletelor de tren sau a numărului de trenuri care pleacă sau sosesc într-un oraș într-o zi
- salvarea unor rapoarte cu privire la trenurile ce sunt stocate în baza de date, aceste rapoarte sunt salvate în mai multe fișiere de formate diferite: csv, xml sau json

Administratorii, după autentificare, pot să efectueze operații de inserare, selectare, modificare sau ștergere asupra angajaților, care de asemenea sunt stocați în baza de date.

3.2. Diagrame de activități

Diagrama de activități pentru vizualizarea trenurilor după anumite criterii

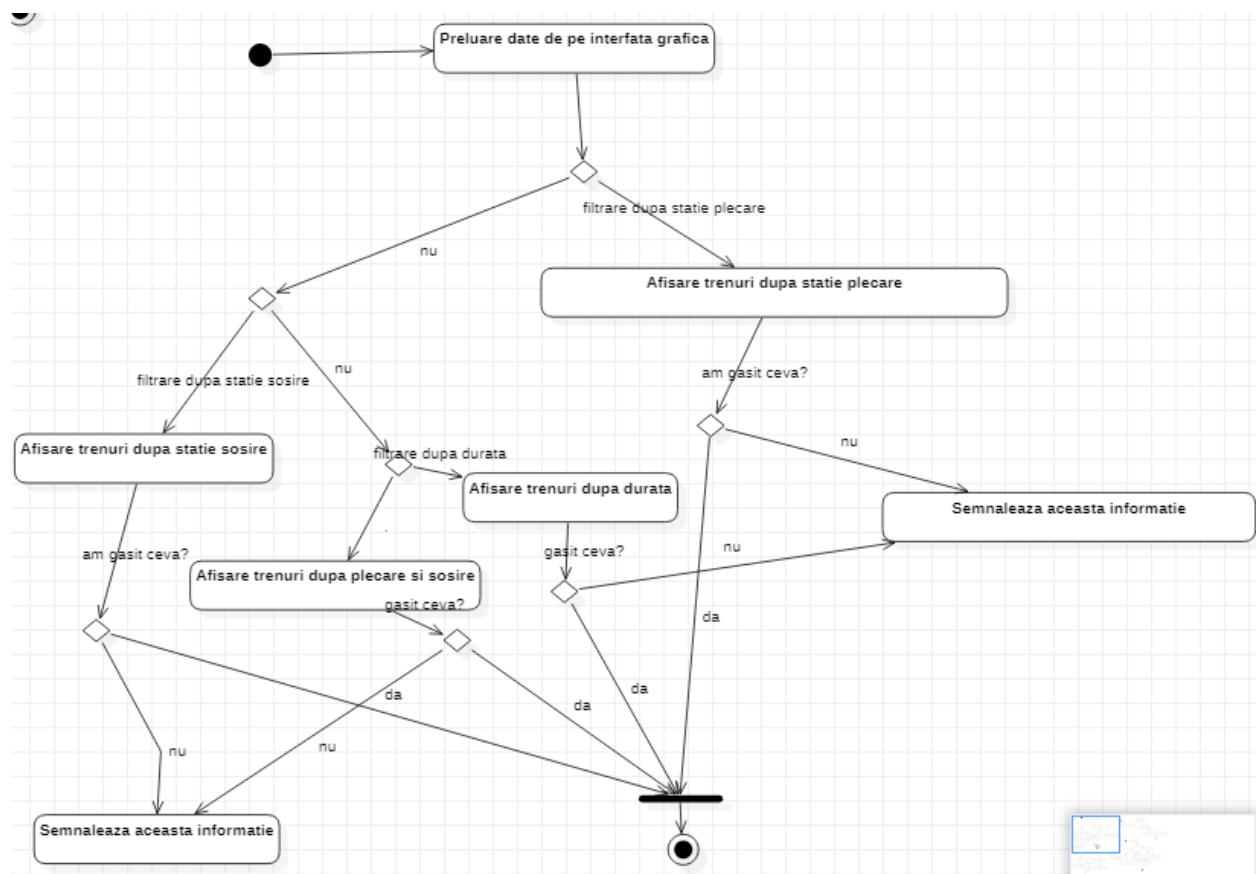


Fig 2. Diagrama de activități pentru vizualizarea trenurilor după anumite criterii

Pentru vizualizarea trenurilor după anumite criterii, mai întâi trebuie să preluăm datele cu privire la criteriul respectiv de pe interfață grafică, chiar dacă datele respective fac referire la un criteriu precum stația de plecare, stația de sosire, durata cursei sau stația de plecare – stația de sosire, principiul este acelaș, vom căuta anumite trenuri care satisfac acest criteriu, în cazul în care s-a găsit măcar un tren sau mai multe, acestea vor fi afișate într-un tabel, dacă nu, se va semnala ca nu s-a găsit niciun tren după criteriul ales.

Diagrama de activități pentru căutarea unui tren

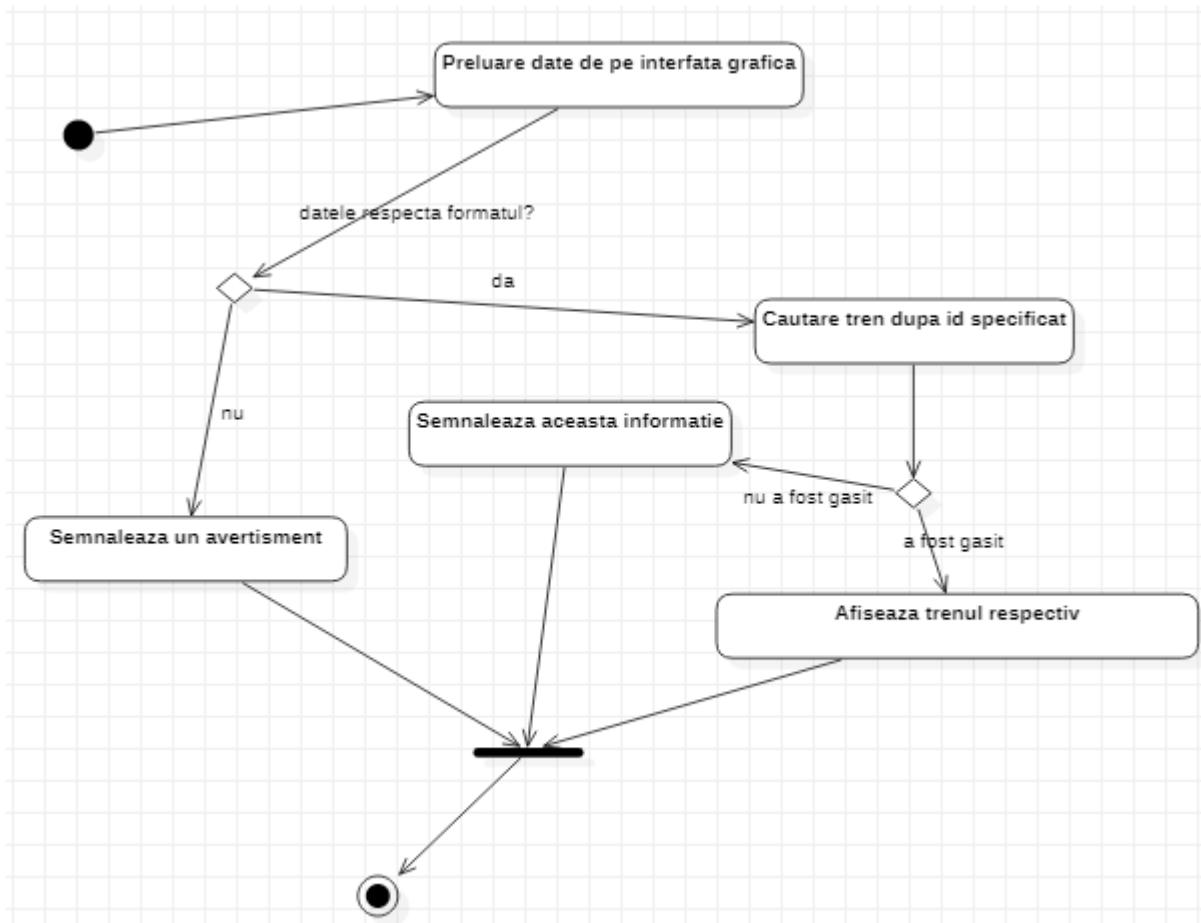


Fig 3. Diagrama de activități pentru căutarea unui tren

Pentru căutarea unui tren trebuie să preluăm anumite date de pe interfață grafică, anume identificatorul unic al trenului, iar pe baza acestui identificator se va căuta trenul, dacă este găsit, acesta va fi afișat, dacă nu, se va semnala acest lucru.

În cazul în care identificatorul nu respectă formatul propus de noi (trebuie să fie un întreg) , se va semnala un avertisment.

Diagrama de activități pentru selectare limbii

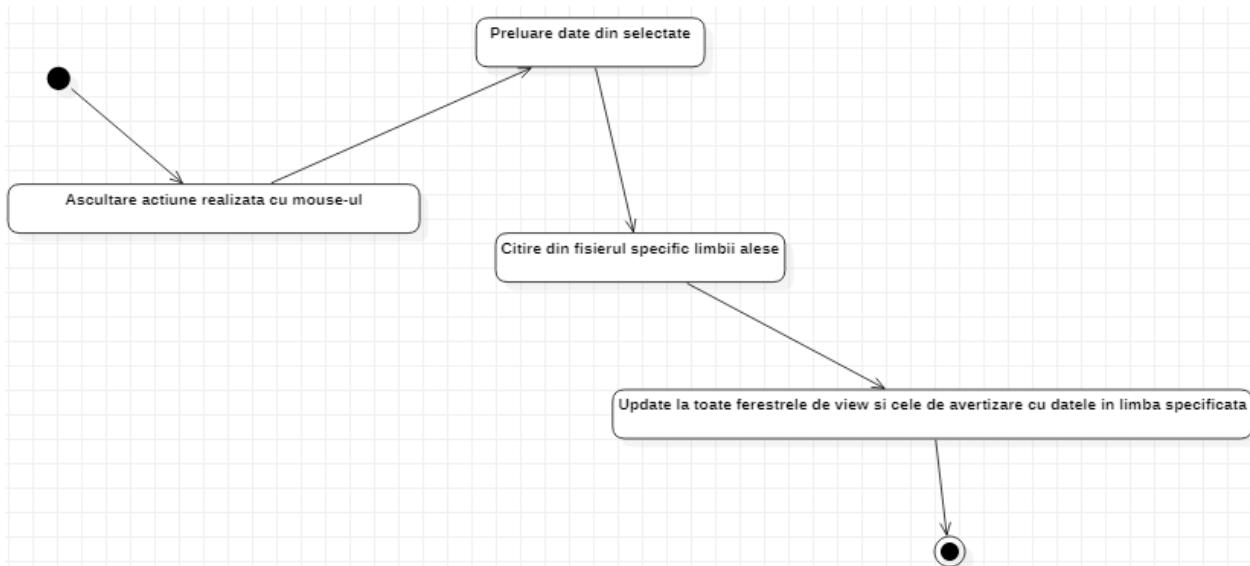


Fig 4. Diagrama de activități pentru selectare limbii

Pentru selectarea limbii, se va aștepta o acțiune ce constă în apăsarea unui click pe o anumită limbă din meniul principal, acea limbă selectată va fi preluată sub forma unei date, iar pe baza acesteia se va citii un anumit fișier ce corespunde limbii alese, fișier în care se găsesc toate traducerile pentru ferestrele ce țin de interfață grafică, urmând ca toate datele din acele ferestre să fie modificate în limba aleasă.

Diagrama de activități pentru autentificarea unui angajat sau administrator

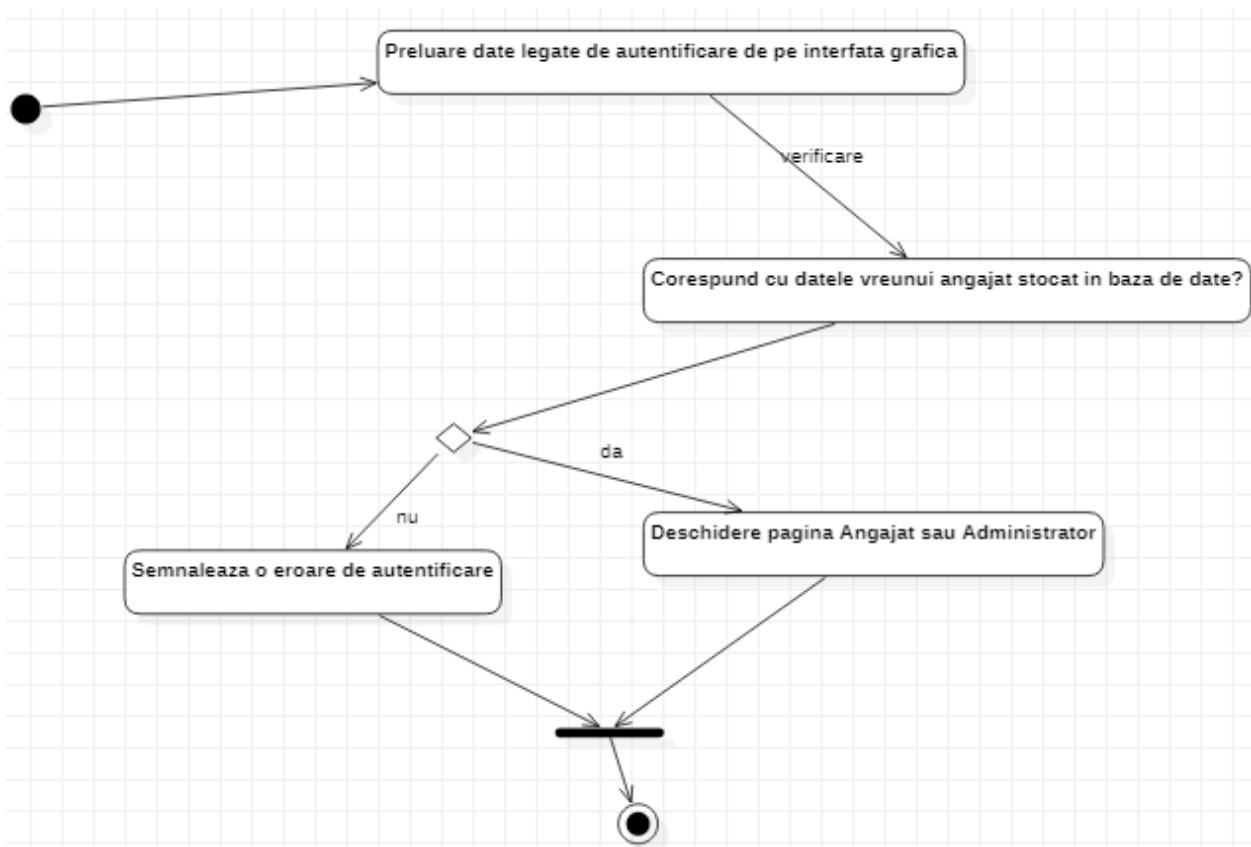


Fig 5. Diagrama de activități pentru autentificarea unui angajat sau administrator

În momentul în care un angajat sau un administrator încearcă să se autentifice, se vor prelua respectivele date ce au legatură cu autentificarea de pe interfață grafică, ulterior se va verifica corectitudinea datelor ce au fost introduse, în cazul în care datele introduse sunt corecte, adică se găsesc în baza de date, se va deschide o nouă pagină specifică pentru un angajat sau un administrator, dacă datele nu sunt corecte, se va semnala o eroare de autentificare.

Diagrama de activități pentru vizualizarea statisticilor legate de prețurile și stațiile de plecare, respectiv sosire pentru biletele vândute

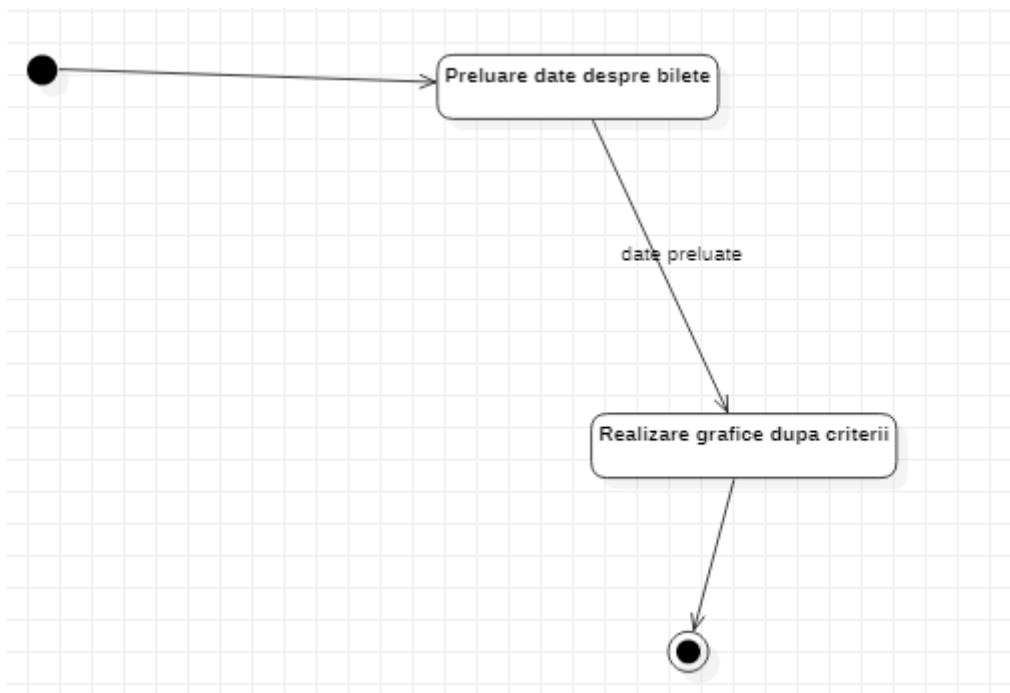


Fig 6. Diagrama de activități pentru vizualizarea statisticilor legate de prețurile și stațiile de plecare, respectiv sosire pentru biletele vândute

Ideea de bază este destul de simplă, se vor prelua datele despre biletele care au fost vândute, iar pe baza acestor date, se vor realiza niște grafice legate de prețurile acestor bilete și de numărul de trenuri care vin și pleacă dintr-un anumit oraș, făcând referire bineînțeles doar la trenurile care se găsesc pe biletele vândute.

Diagrama de activități pentru salvarea rapoartelor cu privire la lista de trenuri în diferite fișiere care au formate diferite

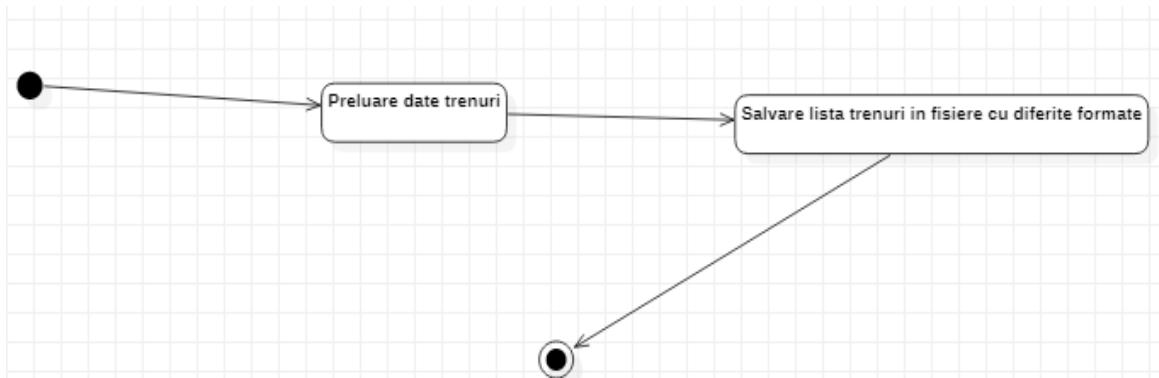


Fig 7.Diagrama de activități pentru salvarea rapoartelor cu privire la lista de trenuri în diferite fișiere care au formate diferite

Pentru acest caz de utilizare, mai întâi trebuie să preluăm informațiile despre trenuri din locul unde acestea sunt stocate, fie că vorbim de o bază de date sau pur și simplu de un fișier de persistență.

După aceea, aceste date le vom scrie în mai multe fișiere cu diferite formate : json, xml, csv.

Diagrama de activități pentru inserarea, modificare, selectare și ștergerea unor informații despre angajați în cazul administratorilor

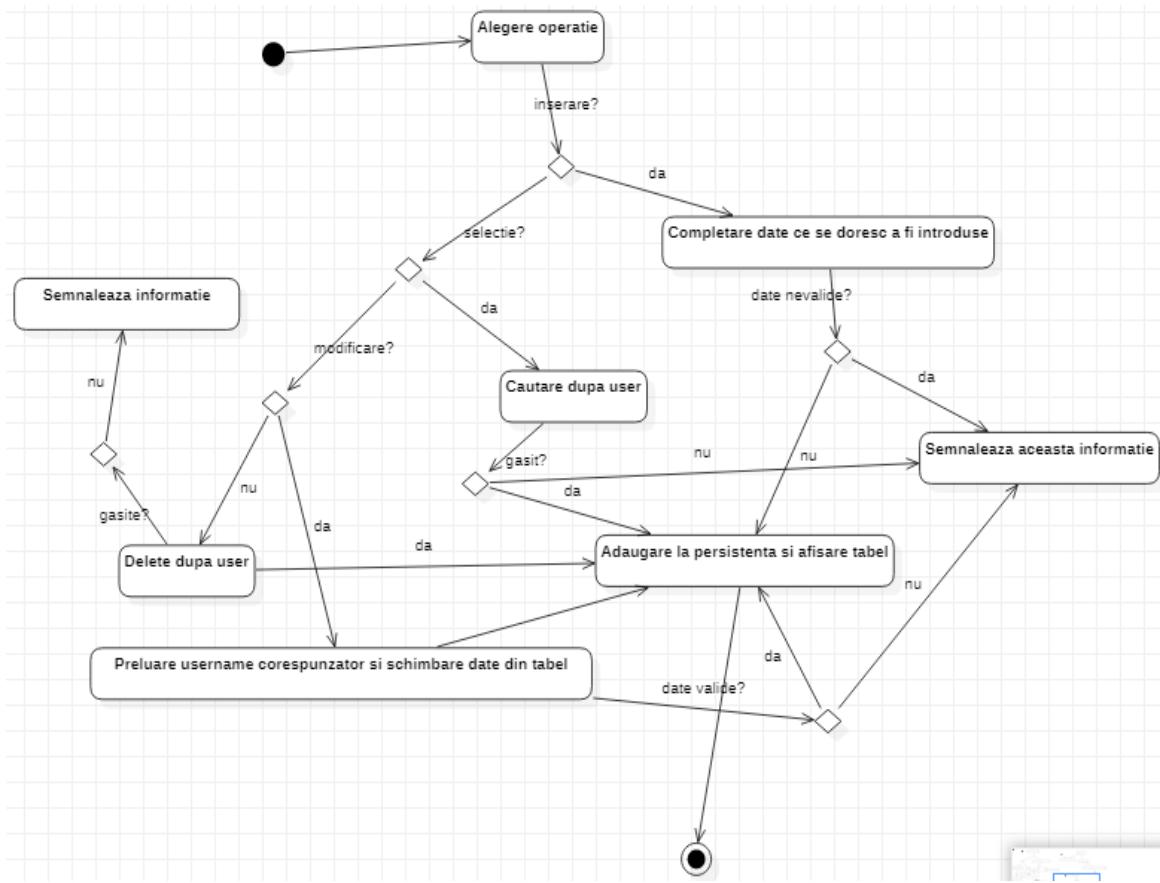


Fig 8. Diagrama de activități pentru inserarea, modificare, selectare și ștergerea unor informații despre angajați în cazul administratorilor

În cazul acestui caz de utilizare, se va prelua inițial de pe interfață grafică operația care se dorește a fi aplicată pe un anumit angajat, dacă va fi vorba de inserare, ulterior va trebui să fie completate detaliile referitoare la noul angajat, iar în cazul în care toate acestea respectă formatul impus de noi, noul angajat va fi adăugat la persistență.

În cazul selecției, va trebui introdus user-ul angajatului care este căutat, în cazul în care user-ul este valid, se va afișa un tabel cu informațiile despre angajatul respectiv, dacă nu e valid, se va semnala acest lucru.

În cazul modificării, se va introduce din nou user-ul ce corespunde angajatului a cărui date vrem să le modificăm, urmând să se completeze noile informații, în cazul în care totul a mers bine, modificările vor fi salvate și afișate ulterior într-un tabel, dacă ceva nu a mers bine, cum ar fi faptul că user-ul nu era valid sau informațiile care au fost completate ulterior nu respectau un anumit format, se va semnala un avertisment cu privire la cauza insuccesului.

Și nu în ultimul rând, în cazul în care vrem să stergem un angajat, va trebui introdus ulterior user-ul acestuia, dacă user-ul introdus este valid, ștergerea se va efectua cu succes și se va afișa noua listă de useri existenți pentru a verifica că user-ul pe care l-am șters nu mai există.

În cazul în care user-ul introdus de noi nu este valid, se va semnala această informație.

Diagrama de activități pentru inserarea, modificare, selectare și ștergerea unor informații despre trenuri și bilete în cazul angajaților

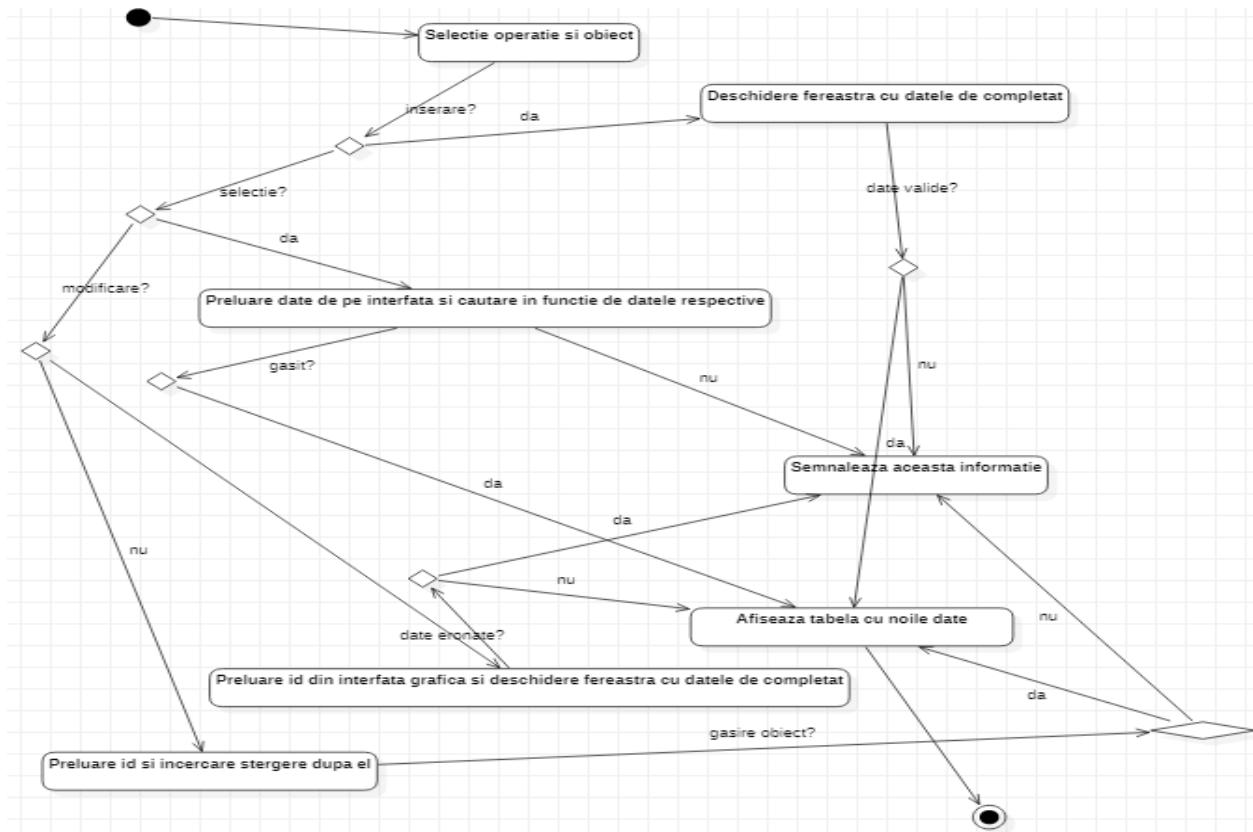


Fig 9. Diagrama de activități pentru inserarea, modificare, selectare și ștergerea unor informații despre trenuri și bilete în cazul angajaților

În acest caz de utilizare, va trebui inițial să fie preluate de pe interfață grafică date referitoare la operația pe care vrem să o executăm și pe ce obiect dorim să efectuăm această operație.

În cazul în care vorbim de inserare ca operație, trebuie să completăm noile date referitoare la obiectul selectat pe care dorim să îl introducem, dacă aceste date respectă formatul impus, se va adăuga la persistență și se va afișa noul tabel ce conține toate obiectele de tipul selectat, printre care și cel adăugat anterior de către noi, dacă nu toate datele introduse respectă acel format, se va semnala un advertisement.

În cazul selecției unui anumit obiect, fie că vorbim de un tren sau de un bilet, va trebui introdus identificatorul acestuia care este unic, dacă acest identificator introdus respectă formatul și chiar există un obiect cu acel id, acesta va fi afișat într-un tabel, dacă respectă formatul însă nu se găsește obiectul cu id-ul respectiv, se va semnala lipsa aceluiași obiect, iar dacă identificatorul nu respectă formatul, se va semnala un advertisement.

În cazul operației de modificare, va trebui introdus din nou id-ul obiectului care se dorește a fi modificat, dacă acesta respectă formatul, se vor putea completa noile detalii despre obiect și ulterior aceste date vor fi salvate, dacă id-ul obiectului nu respectă formatul sau este de negăsit, se va semnala acest lucru.

Diagrama de activități pentru vânzarea unui bilet

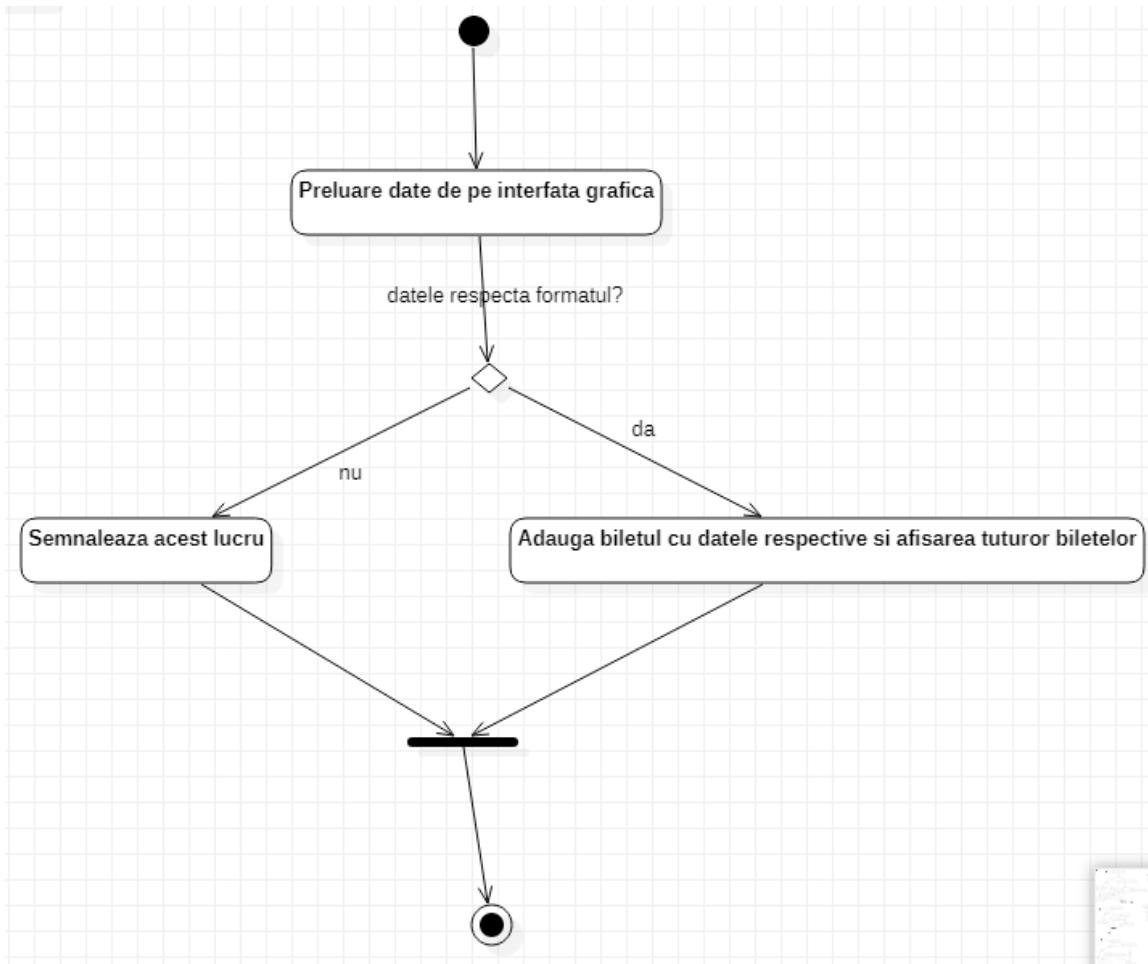


Fig 10. Diagrama de activități pentru vânzarea unui bilet

Se preiau datele despre biletul care se dorește să fie introdus, aceste date fac referire la: id-ul biletului, numele, prenumele, numărul de telefon și adresa de e-mail a cumpărătorului, id-ul trenului pentru care este cumpărat biletul și prețul acestuia.

În cazul în care toate datele preluate respectă formatul impus, se va crea un nou bilet și va fi adăugat la lista de bilete vândute, lista care ulterior va fi afișată.

În caz contrar, se va semnala un advertisman.

4. Etapa de proiectare

4.1 Diagrame de clase

Diagrama de clasă pentru client

În această clasă se poate observa clasa AppClient, de unde se pornește practic aplicația pe partea de client.

MainView este clasa ce definește pagina principală a aplicației, pe lângă această clasă care face parte din view, mai avem alte clase specifice de view, fiecare clasă reprezentând practic o fereastră, toate au ca și componente în general : butoane, jlabel-uri, jTextField-uri, jcombobox-uri, gettere pentru jcombobox-uri și jTextField-uri și ascultători pentru butoanele existente.

Clasa ControlerOperatiilorCalator este practic punctul de legătura între partea de client și de server, aici întâmplându-se toate operațiile existente.

Tot aici se găsesc și multe clase interne de tip ButtonListeneri, fiecare clasă implementând o anumită funcționalitate la apăsarea unui anumit buton.

Iar clasa BarChartSimple este folosită pentru desenarea graficului cu privire la situația prețurilor pentru biletele vândute și numărul de trenuri care pornesc din / sosesc într-un anumit oraș.

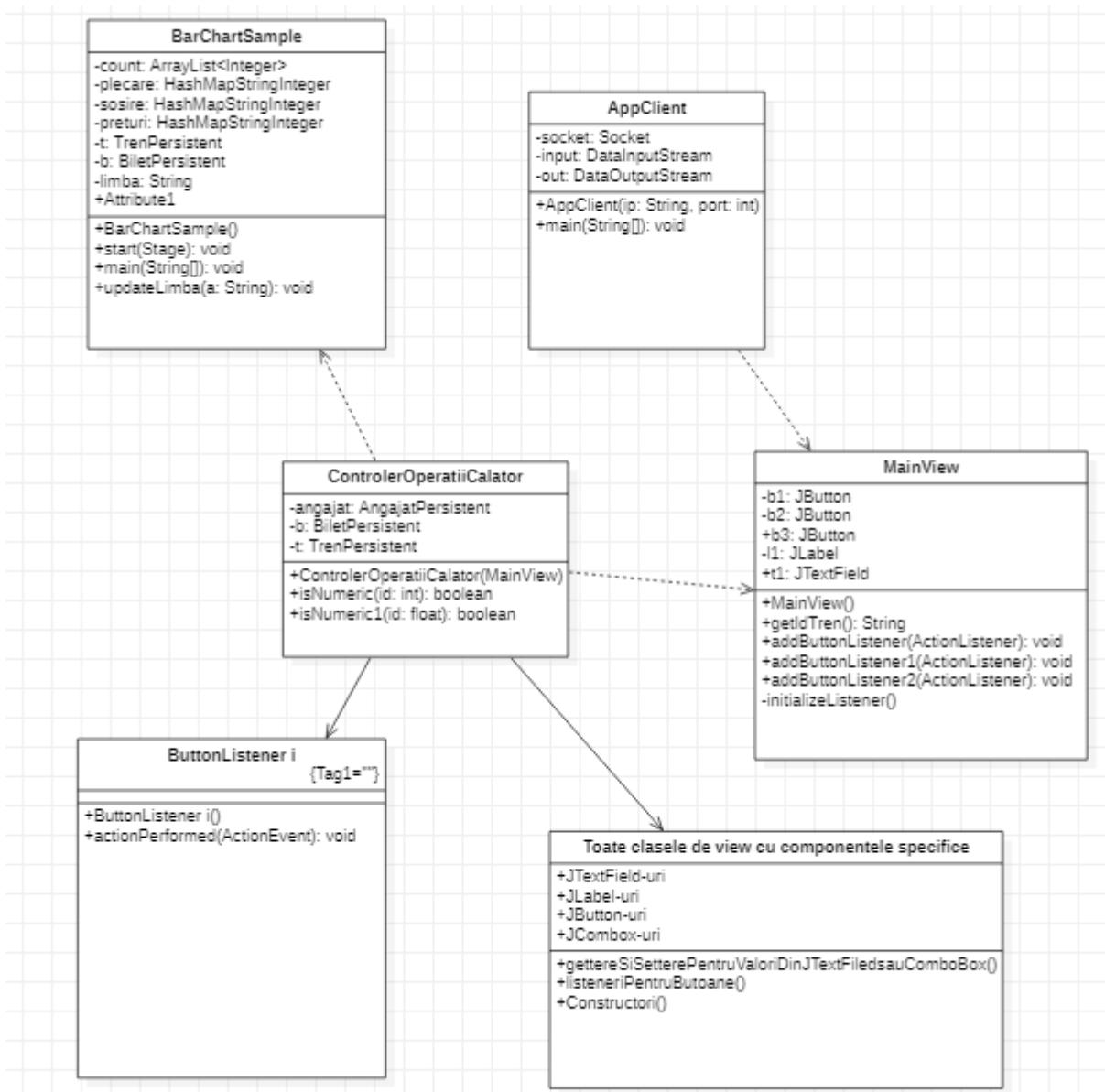


Fig 11. Diagrama de clasă pentru client

Diagrama de clasă pentru server

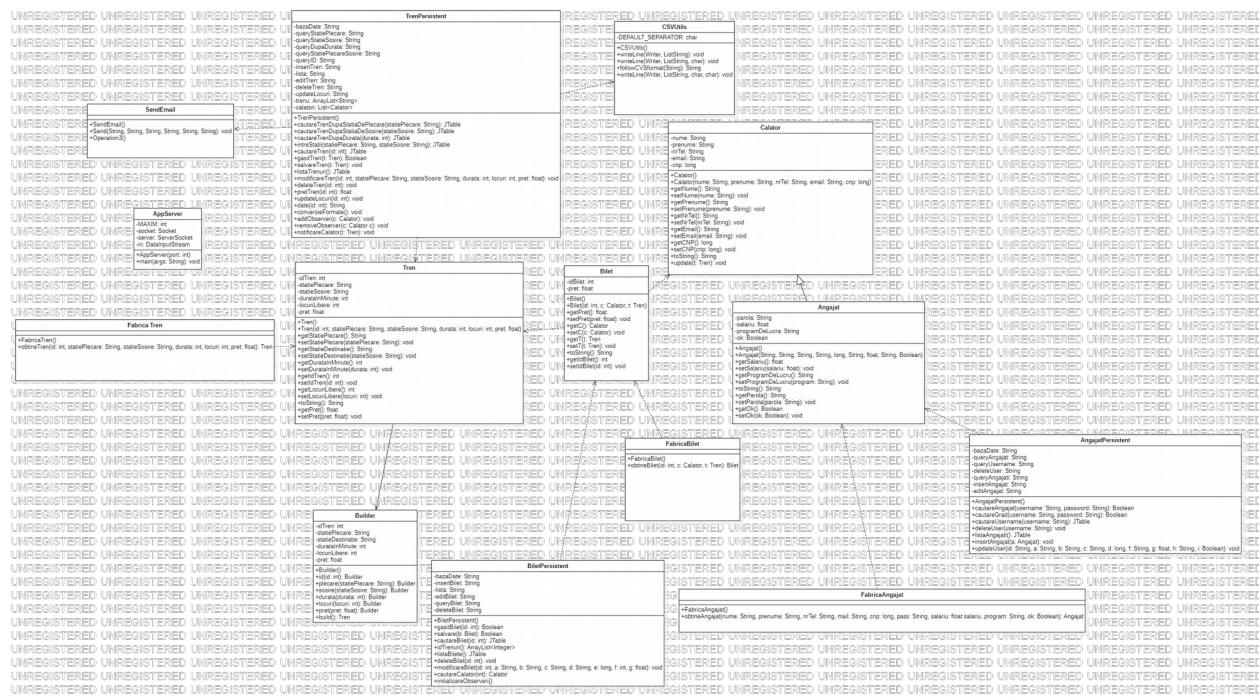


Fig 12. Diagrama de clasă pentru server

Cu ajutorul clasei AppServer, se pornește aplicația pe partea de server.

Clasele BilePersistent, TrenPersistent și AngajatPersistent sunt folosite pentru accesarea, inserarea, modificarea sau ștergerea datelor legate de bilete, trenuri și angajați din baza de date.

În plus, se observă că în clasa de TrenPersistent s-a implementat design pattern-ul Observer, această clasă are o listă de observeri de tip Călător și metodele specifice pentru adăugarea unui observer, ștergerea unui observer și notificarea tuturor observerilor.

Această notificare va consta în trimitera a câte un e-mail pentru fiecare călător care a cumpărat un bilet până în prezent, notificarea va avea loc în momentul în care un nou tren este introdus în baza de date, deoarece vrem să notificăm toți clienții cu privire la introducerea unor noi trenuri, fiindcă există posibilitatea să fie interesați.

Trimiterea efectivă a e-mailului este realizată de către clasa SendEmail.

Clasele de Bilet, Tren și Călător reprezintă niște şabloane despre cum ar trebui să arate acestea, în general au doar atribute și gettere/settere pentru atributele respective.

Ies în evidență clasele Călător și Tren, deoarece în Călător se observă metoda de update specifică design pattern-ului Observer, iar în clasa Tren se observă implementarea design pattern-ului Builder cu ajutorul clasei interioare Builder.

Rolul acestui design pattern este acela că nu întotdeauna avem nevoie de definerea tuturor atributelor clasei la o instanțiere, cum a fost în cazul nostru când doream să instantiem un obiect de tipul Bilet, acesta avea nevoie de 1 obiect de tip Călător și un obiect de tip Tren, însă în legătură cu trenul, nu ne interesau toate detaliile despre acesta, ci doar identificatorul și prețul cursei.

Alte clase despre care nu am vorbit ar fi clasele FabricaAngajat, FabricaBilet și FabricaTren, acestea implementează design pattern-ul FactoryMethod, fiecare dintre acestea având o metodă ce returnează un obiect de un anumit tip.

Iar clasa CSVUtils este folosită la salvarea raporturilor cu privire la lista de trenuri într-un fișier cu extensia csv.

4.2. Diagrame de secvență

Diagrama de secvență pentru alegerea limbii

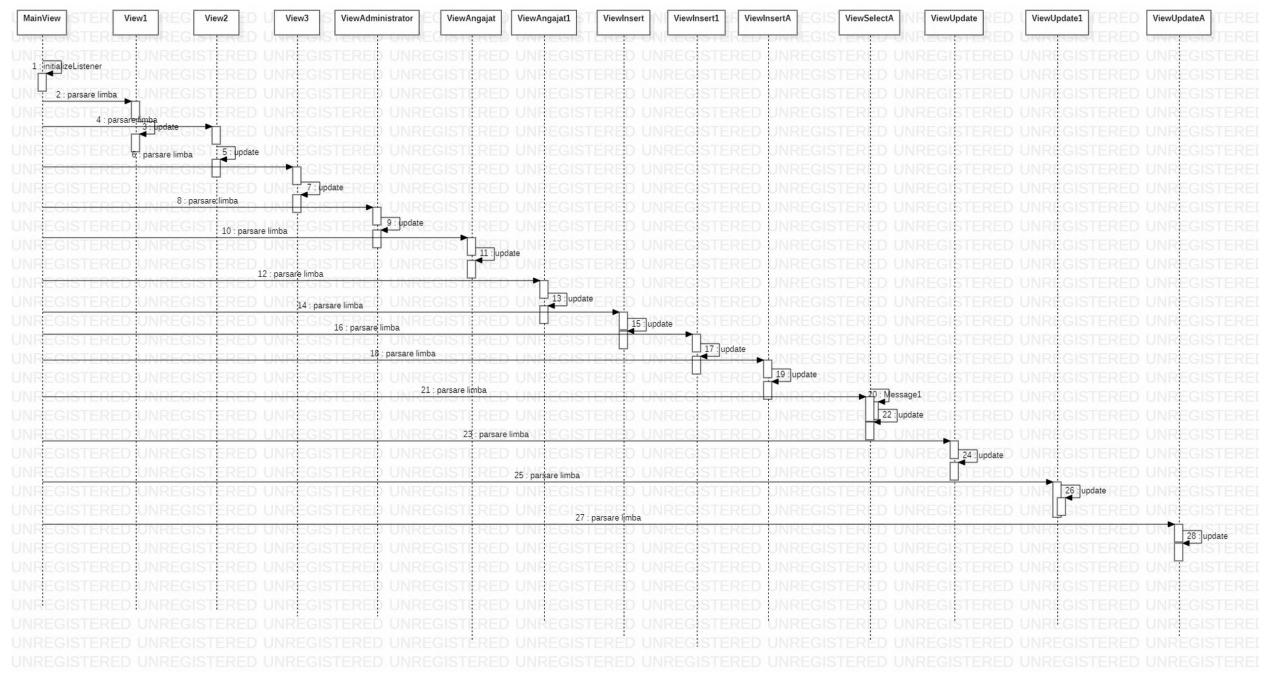


Fig 13. Diagrama de secvență pentru alegerea limbii

În momentul apăsării unui click pe o anumită limbă de pe pagina principală se apelează metoda `initializeListener()` care preia informația cu privire la limba selectată, iar în funcție de aceasta se vor citii traducerile pentru fiecare cuvânt vizibil pe interfață dintr-un fișier specific limbii alese, iar aceste traduceri vor fi transmise către fiecare fereastră de view ce aparține de interfață.

Diagrama de secvență pentru vizualizarea trenurilor după anumite criterii

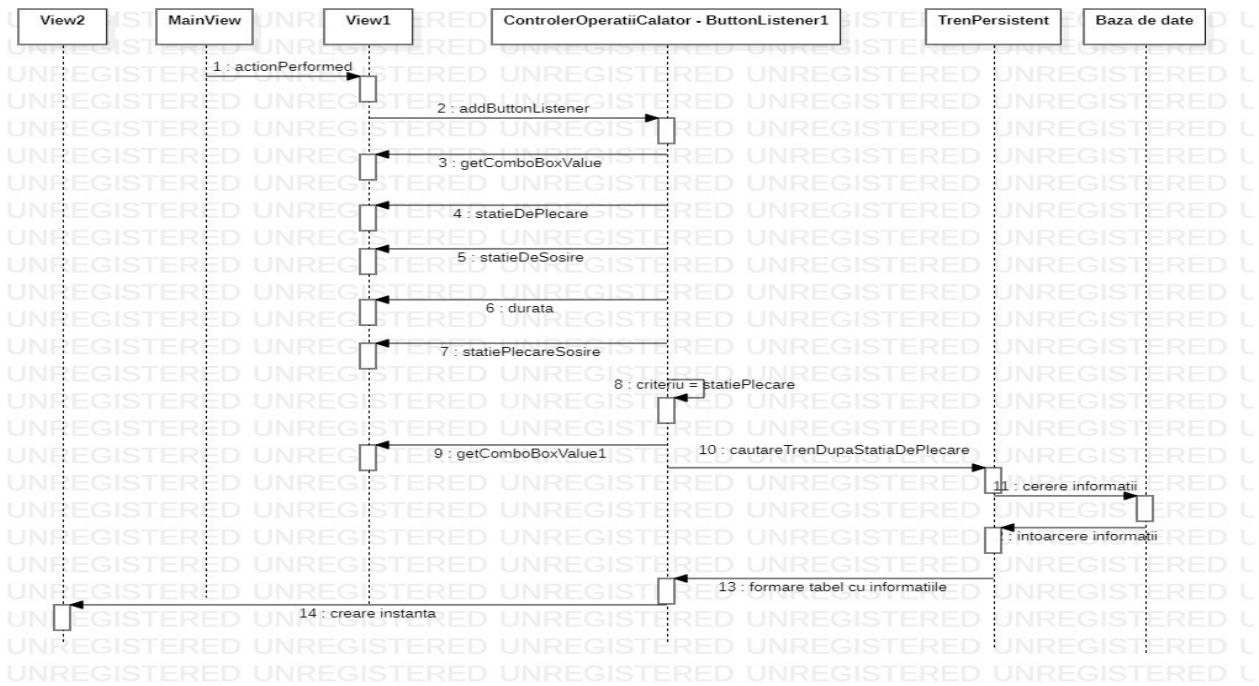


Fig 14. Diagrama de secvență pentru vizualizarea trenurilor după stația de plecare

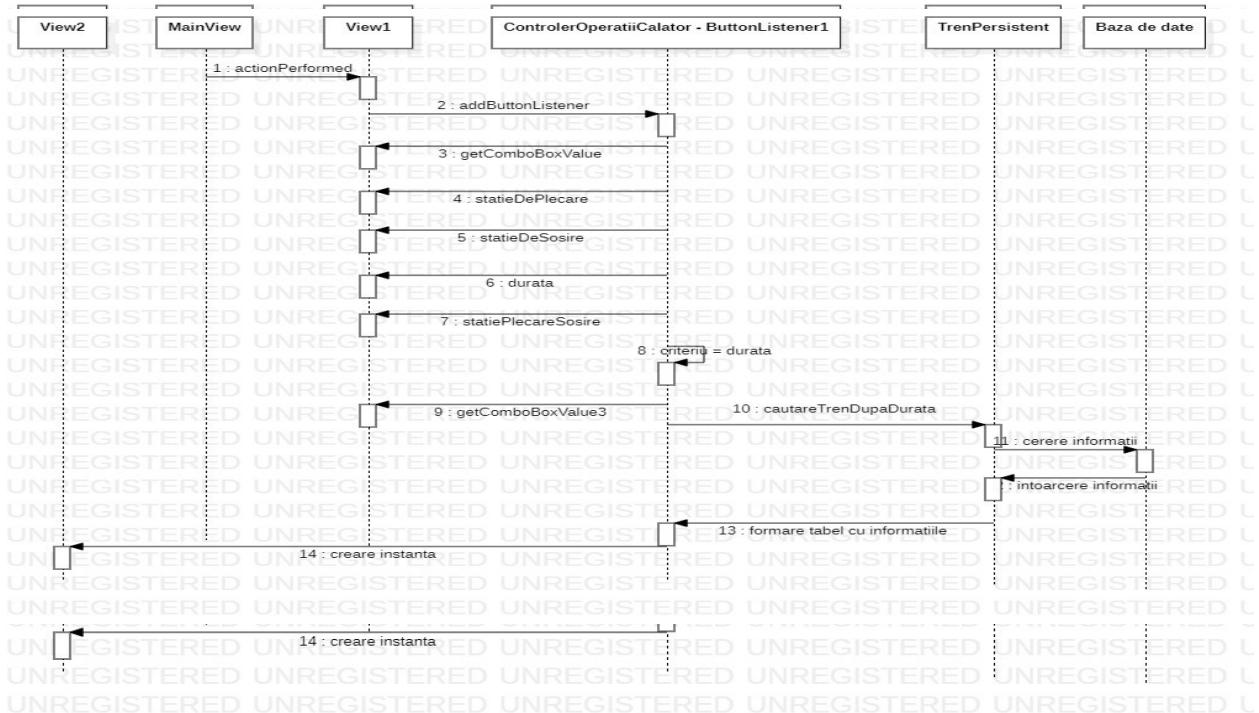


Fig 15. Diagrama de secvență pentru vizualizarea trenurilor după durată

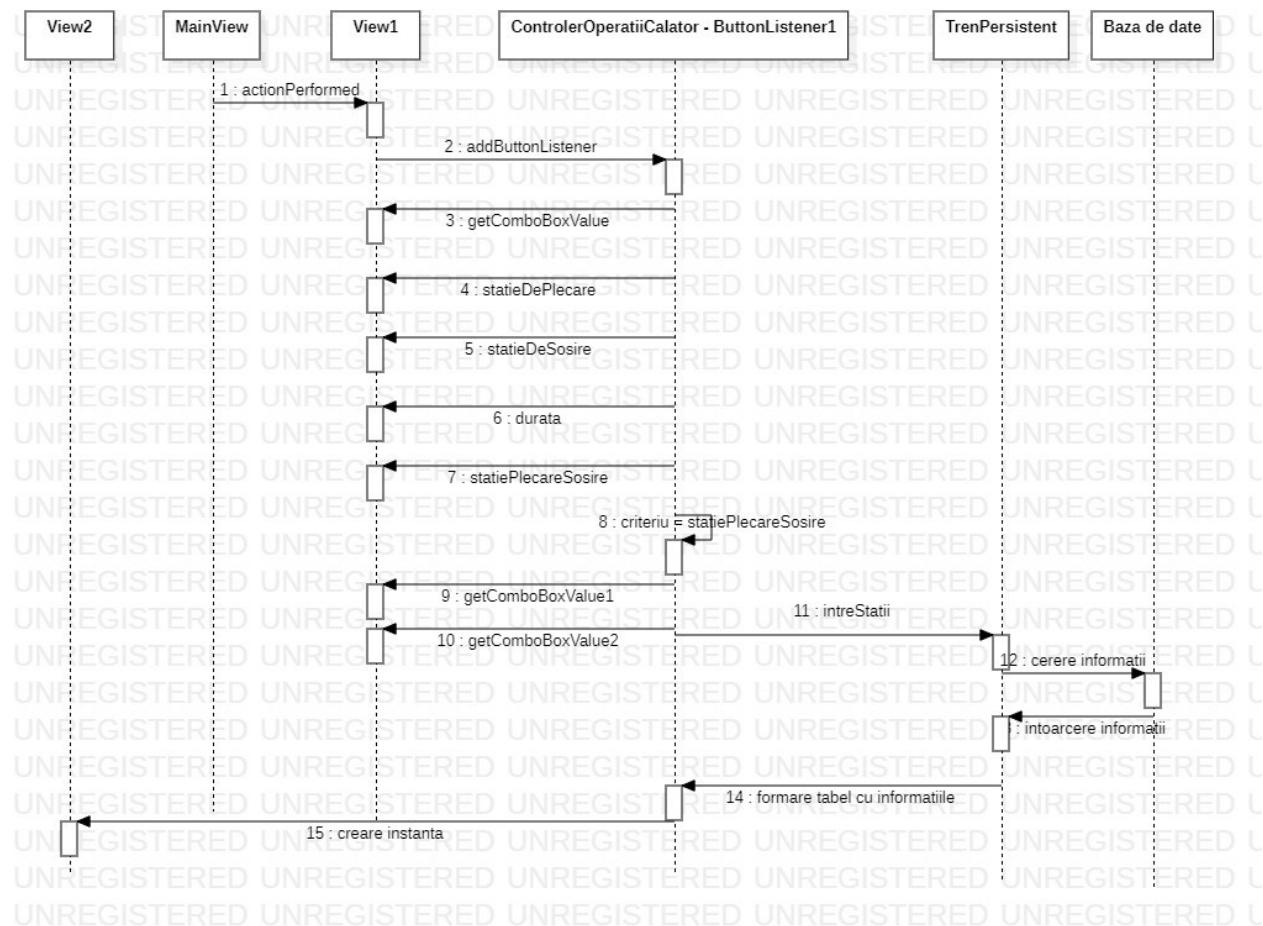


Fig 16. Diagrama de secvență pentru vizualizarea trenurilor după stația de plecare – stația de sosire

În fereastra principală se accesează butonul pentru Vizualizarea trenurilor, ceea ce duce la deschiderea unei ferestre de tip View1, acolo se preiau informații precum criteriu ales și cele 4 criterii disponibile după apăsarea butonului de OK.

În funcție de criteriu ales, se vor prelua informații relevante ce au legătura cu acel criteriu și se vor căuta trenurile după informațiile respective în baza de date.

Trenurile găsite vor fi returnate și se va forma un tabel cu acestea care se va pute viziona într-o fereastră de tip View2.

Diagrama de secvență pentru căutarea unui tren după ID

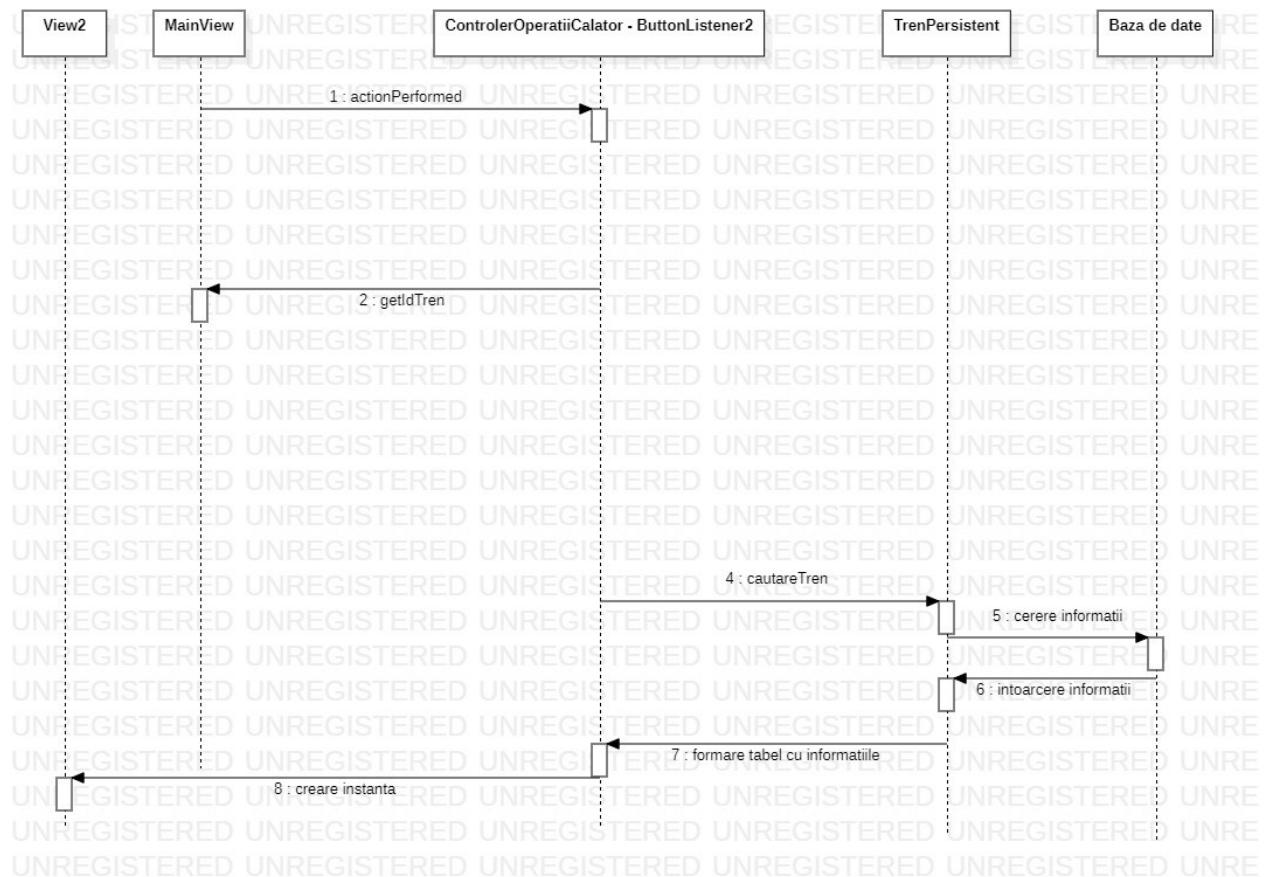


Fig 17. Diagrama de secvență pentru căutarea unui tren după ID

În pagina principală de MainView, la apăsarea butonului de Căutare tren, se va prelua id-ul introdus pe aceeași pagină la rubrica intitulată Introdu id și se va căuta trenul care corespunde acestui id, în cazul în care acesta este găsit se va returna un tabel cu informațiile despre acesta, în caz contrar, tabelul returnat va fi gol.

Diagrama de secvență pentru autentificare

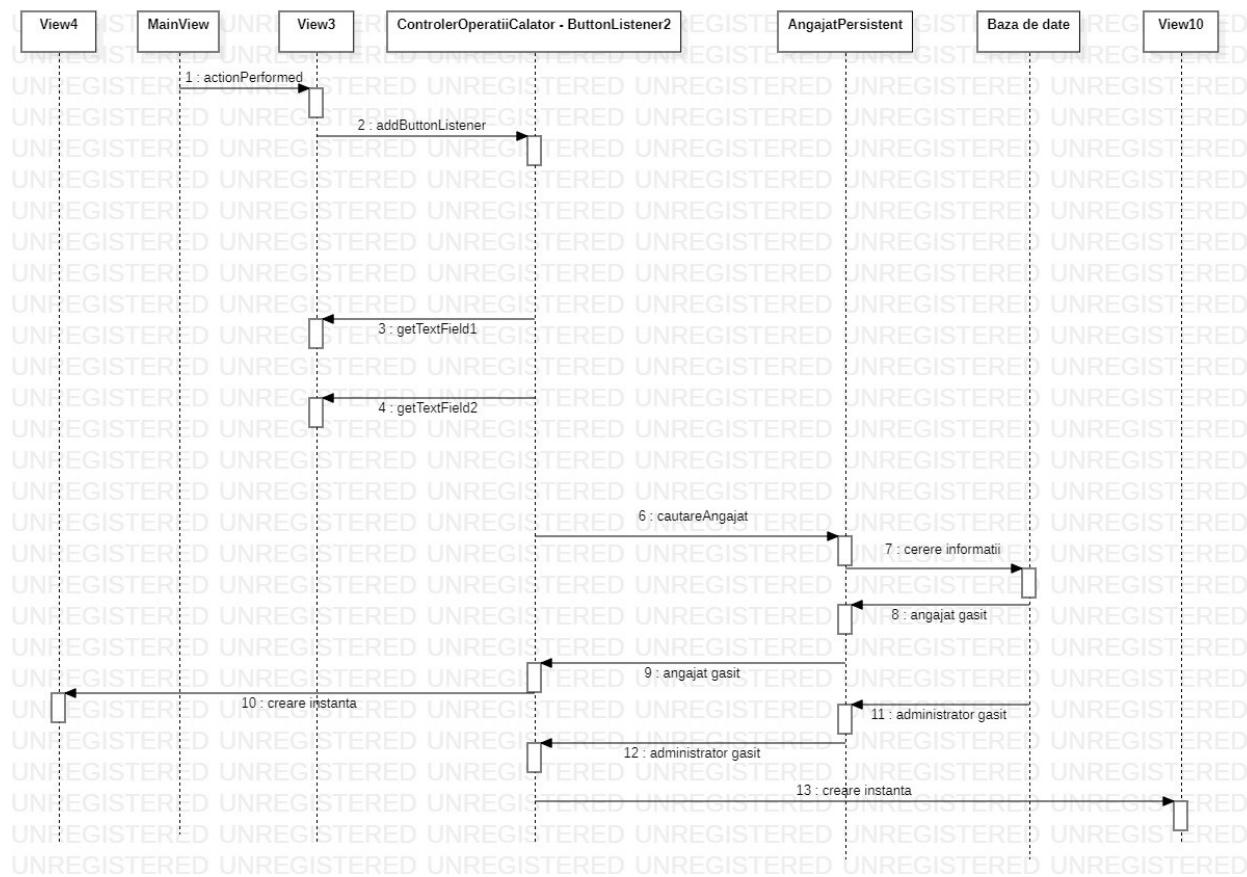


Fig 18. Diagrama de secvență pentru autentificare

În pagina principală de MainView, după apăsarea butonului de Conectare, se va deschide o nouă fereastră de tip View3, acolo trebuie completate numele utilizatorului și parola acestuia, după completare și apăsarea butonului de Autentificare, se vor prelua datele introduse și se va apela o metodă numită cautareAngajat din clasa AngajatPersistent care primește datele preluate și verifică dacă există în baza de date un astfel de angajat.

În cazul în care găsește un angajat cu acces limitat, se va deschide o nouă fereastră de tip View4.

Dacă găsește un angajat cu acces mai mare, înseamnă ca e vorba de un administrator, iar pentru acesta, se va deschide o nouă fereastră de tip View10.

Diagrama de secvență pentru vânzarea unui bilet către un călător

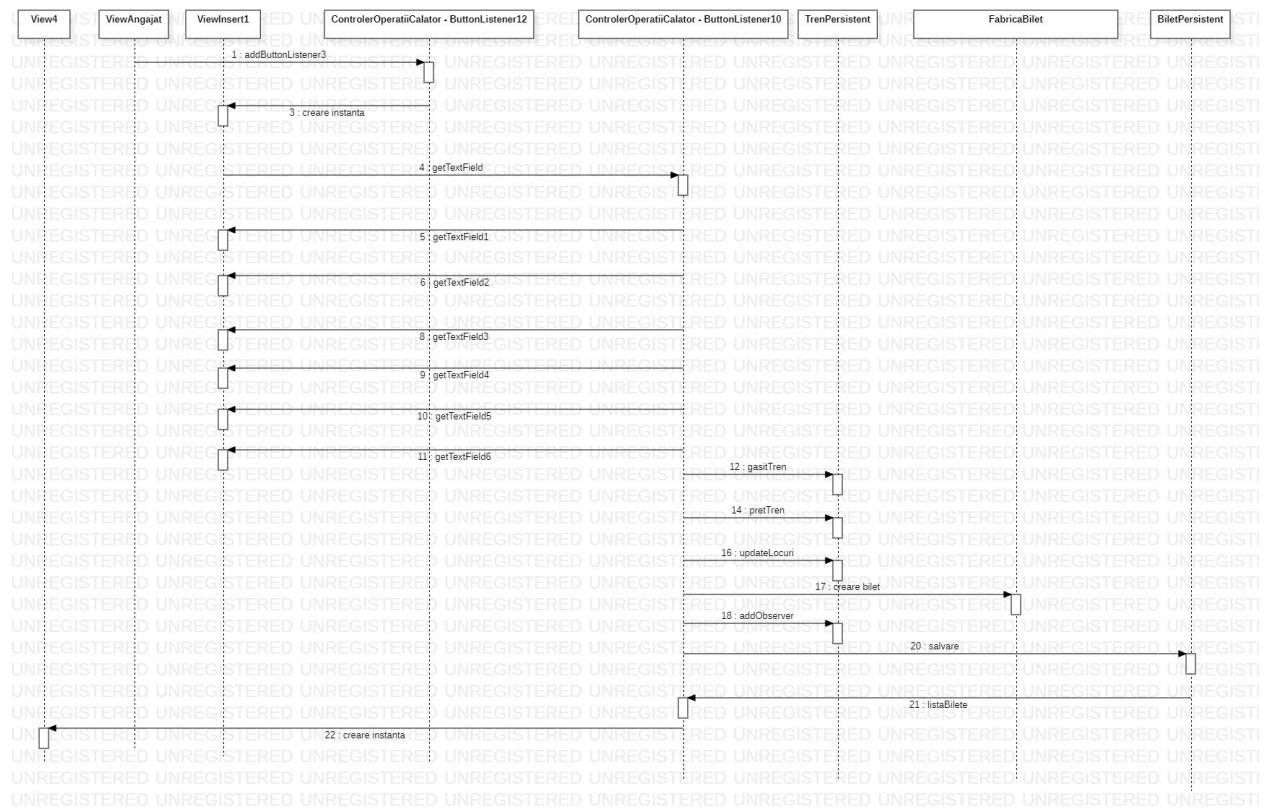


Fig 19. Diagrama de secvență pentru vânzarea unui bilet către un călător

Pornind de la pagina angajatului, la apăsarea butonului de Vânzare bilet, se va deschide o nouă pagină de tip ViewInsert1, după ce se completează datele acolo și este apăsat butonul de OK, vor fi preluate toate datele care s-au completat : id-ul biletului, nume călător, prenume călător, telefon călător, e-mail călător , cnp călător, id-ul trenului și se va căuta trenul al cărui id a fost introdus, pentru aflarea prețului biletului,

În continuare se va modifica numărul de locuri disponibile pentru acel tren, se va crea biletul respectiv cu datele care au fost completate, iar

călătorul care a fost adăugat pe bilet va fi trecut și în lista de observatori pentru a fi anunțat când apar alte trenuri.

Biletul va fi salvat în baza de date și i se vor afișa datele sub forma unui tabel într-o fereastră de tip View2.

Diagrama de secvență pentru salvarea rapoartelor pentru liste de trenuri în fișiere cu diferite formate : xml, json, csv

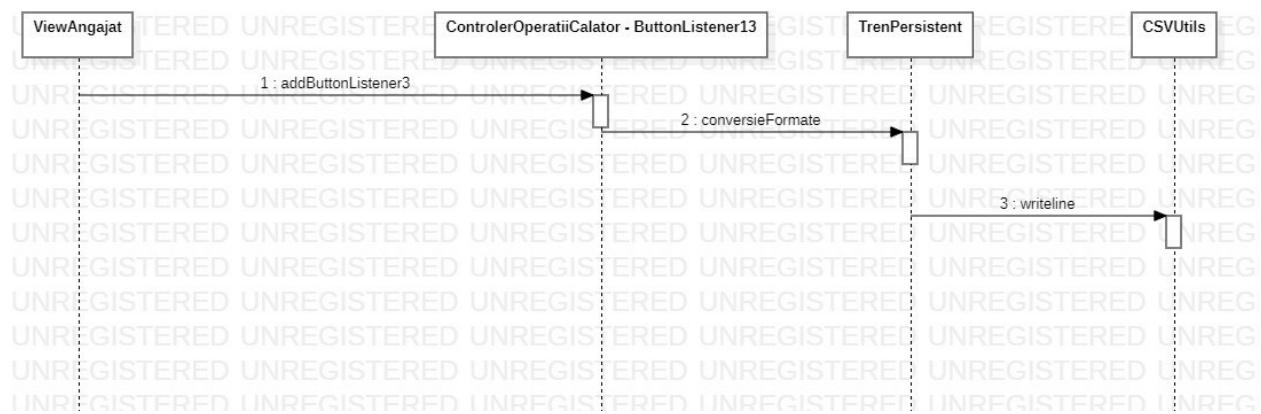


Fig 20. Diagrama de secvență pentru salvarea rapoartelor pentru liste de trenuri în fișiere cu diferite formate : xml, json, csv

Pornind de la pagina angajatului, la apăsarea butonului de Salvare rapoarte, se va apela metoda de conversieFormate() din clasa TrenPersistent, unde se vor citii toate trenurile salvate în baza de date și se vor scrie pe rând în 3 fișiere cu formate diferite: xml, json, csv.

Pentru scrierea în fișierul.csv, metoda din TrenPersistent va apela o altă metodă din clasa CSVUtils.

Diagrama de secvență pentru generarea statisticilor biletelor

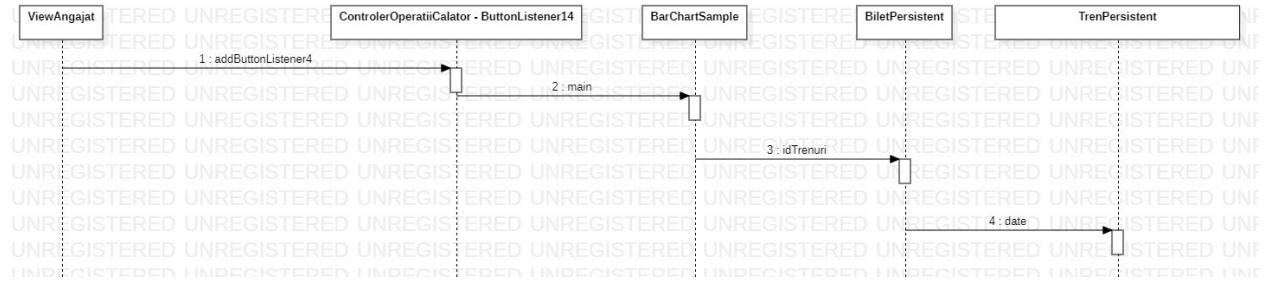


Fig 21. Diagrama de secvență pentru generarea statisticilor biletelor

Pornind de la pagina angajatului, după apăsarea butonului de Vizualizare statistici, se va apela metoda main din clasa de BarChartSample unde se generează practic graficul, acolo se va apela metoda idTrenuri() din clasa BiletPersistent pentru a afla pentru ce trenuri au fost cumpărate biletele respective, iar apoi se va apela metoda date() pentru a salva detaliile care sunt de interes pentru noi cu privire la acele trenuri, anume prețurile lor, de unde pleacă și unde se duc.

Diagrama de secvență pentru operațiile CRUD realizate de către administrator pe angajați

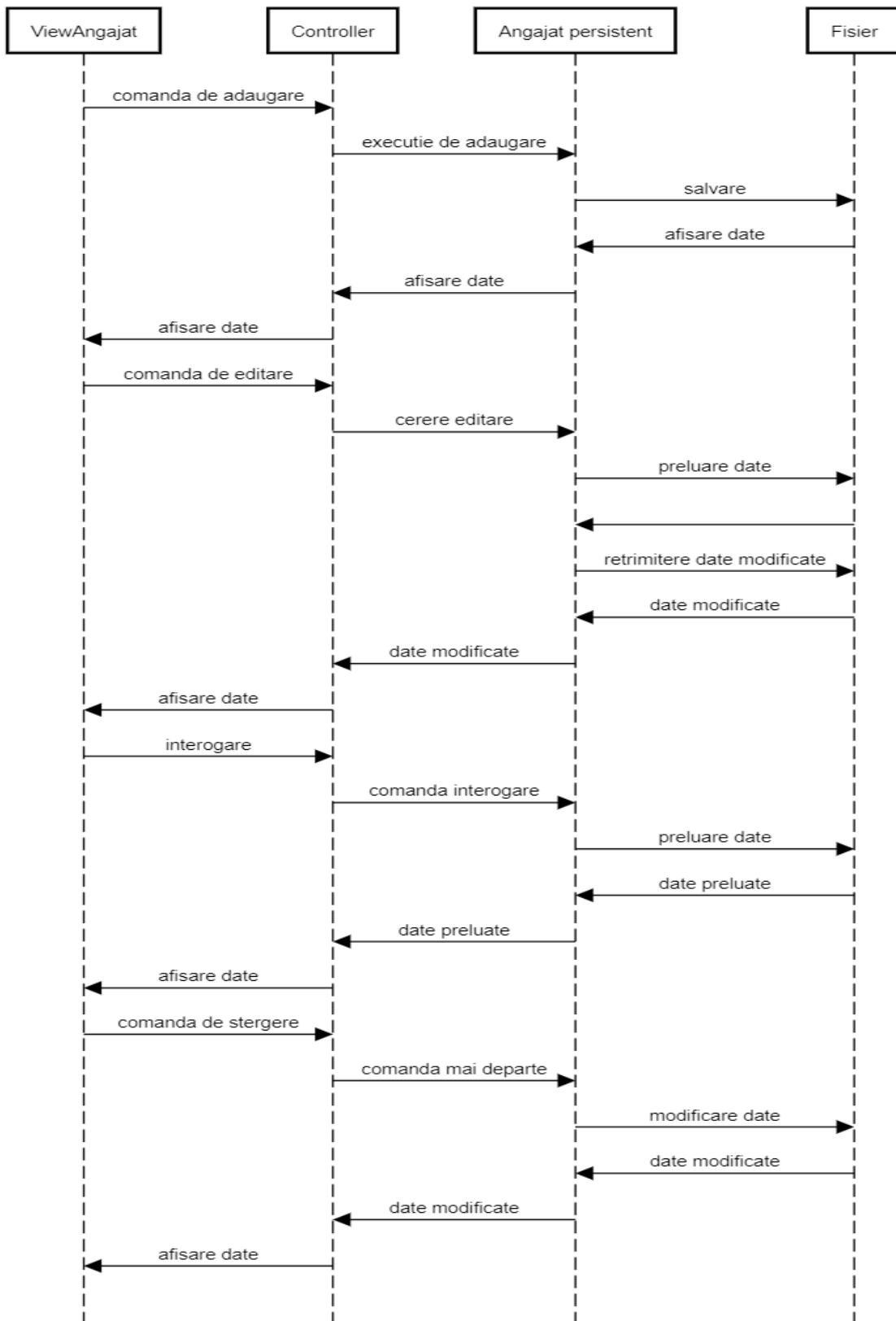


Fig 22. Diagrama de secvență pentru operațiile CRUD realizate de către administrator pe angajați

Deoarece o diagramă de secvențiere pentru operațiile CRUD ar ocupa un spațiu foarte mare atât pe verticală cât și pe orizontală datorită multiplelor metode care sunt accesate, am decis să le exprim în limbaj natural.

Pornind de la pagina administratorului, în funcție de criteriu ales, este posibil să inserăm, selectăm, modificăm sau ștergem un angajat, oricare variantă ar fi, se va apela o metodă specifică din clasa AngajatPersistent, care după caz va salva noul angajat introdus în baza de date în cazul unei inserări, va solicita datele despre un anumit angajat, le va modifica și le va salva pe acelea în locul celor originale în cazul modificării, doar va solicita date despre un anumit angajat în cazul selecției sau va șterge un anumit angajat după numele de utilizator și va actualiza baza de date.

Procesul decurge la fel și în cazul operațiilor CRUD realizate pe trenuri sau pe bilete, lucrându-se pe datele lor specifice.

4.3. Diagrama bazei de date

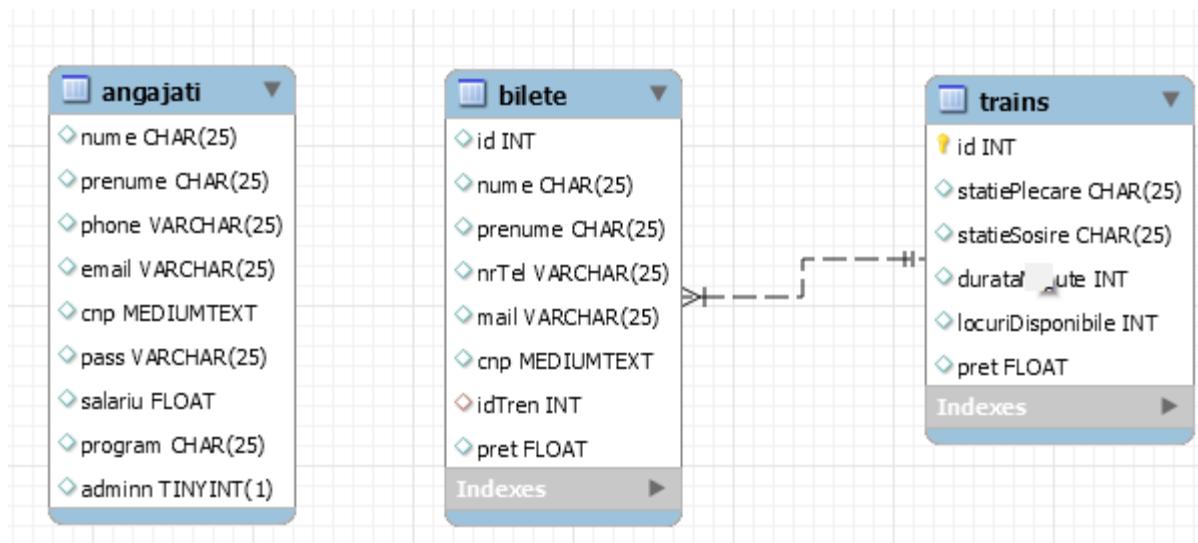


Fig 23. Diagrama bazei de date

Această diagramă este formată din 3 tabele, deoarece vrem să reținem anumite date despre angajați, bilete și trenuri.

Un angajat este descris de un nume, prenume, număr de telefon, e-mail, cnp, parola de la contul său, salariu pe care îl are, programul de lucru și adminn, care este un câmp ce descrie accesul privilegiat al acestuia, adică dacă este administrator sau nu, acesta are valoarea 1 când este vorba de un administrator și valoarea 0, când este vorba de un angajat.

Un bilet este descris de un identificator unic, numele, prenumele, numărul de telefon, e-mailul și cnp-ul persoanei care a cumpărat biletul, identificatorul trenului pentru care a fost cumpărat biletul și prețul acestuia.

Iar un tren este descris cu ajutorul unui identificator unic (care trebuie să fie acelaș cu cel din tabelul de la bilete), stația de plecare, stația de sosire, durata cursei care este contorizată în minute și prețul pentru cursa respectivă.

5. Prezentarea aplicației

Se va pornii mai întâi aplicația de pe partea de server.

În momentul în care aceasta va rula, inițial va afișa în consolă următoarele informații:

„Server started
Waiting for a client...”

Acum dacă vom rula aplicația și de pe partea de client, în consola serverului se va preciza că un anumit client cu un anumit ip s-a conectat la server:

„Client with ip 192.168.1.2 connected” de exemplu.

Iar în consola clientului se va afișa mesajul „Connected” și se va deschide pagină principală a interfeței grafice pentru aplicația propiu-zisă care arată în felul următor:

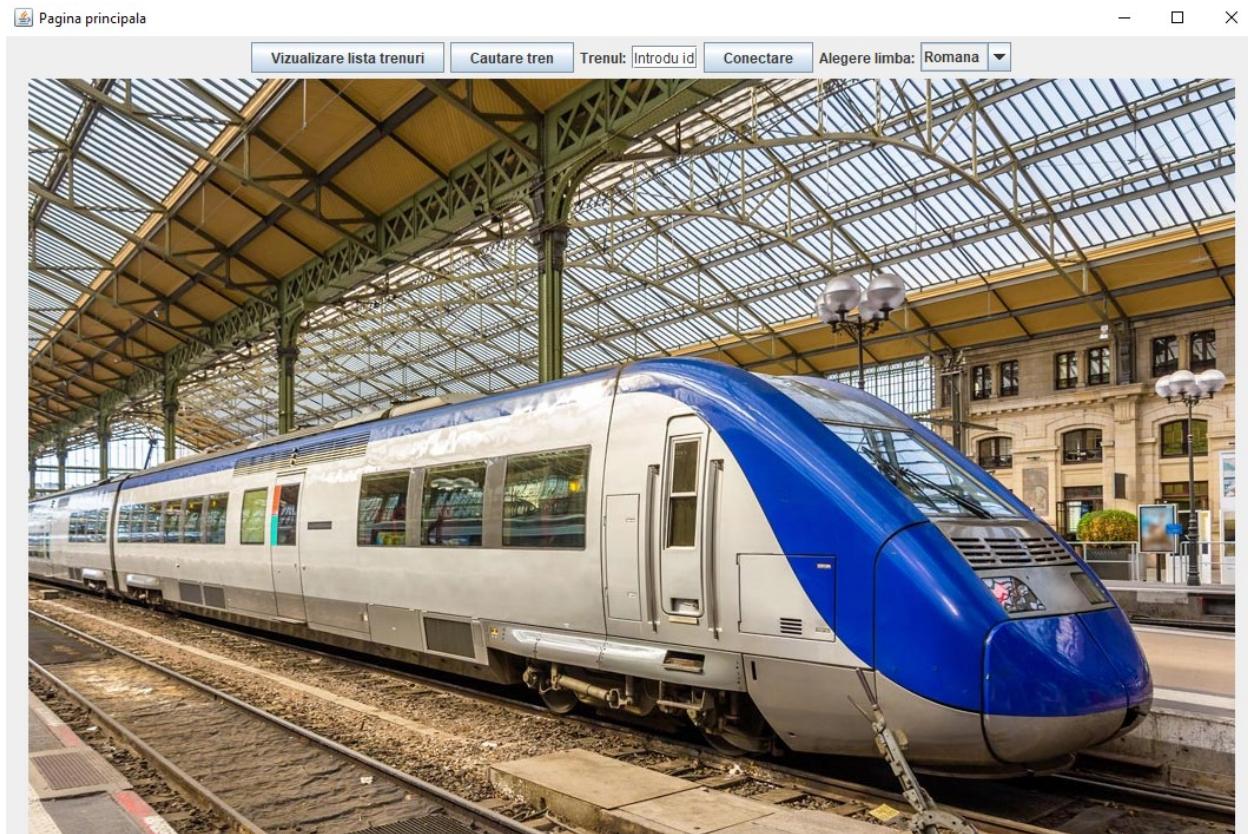


Fig 24. Meniul principal

Această pagină este accesibilă inițial oricărui utilizator, se poate observa că acesta poate să aleagă limba pentru interfața grafică accesând săgeata din zona denumită Alegere limba, moment în care se va deschide un JComboBox cu mai multe limbi, iar în momentul în care vom selecta o limbă prin intermediul unui click, toate detaliile din punct de vedere textual vor fi traduse în limba specificată.



Fig 25. Selectarea limbii

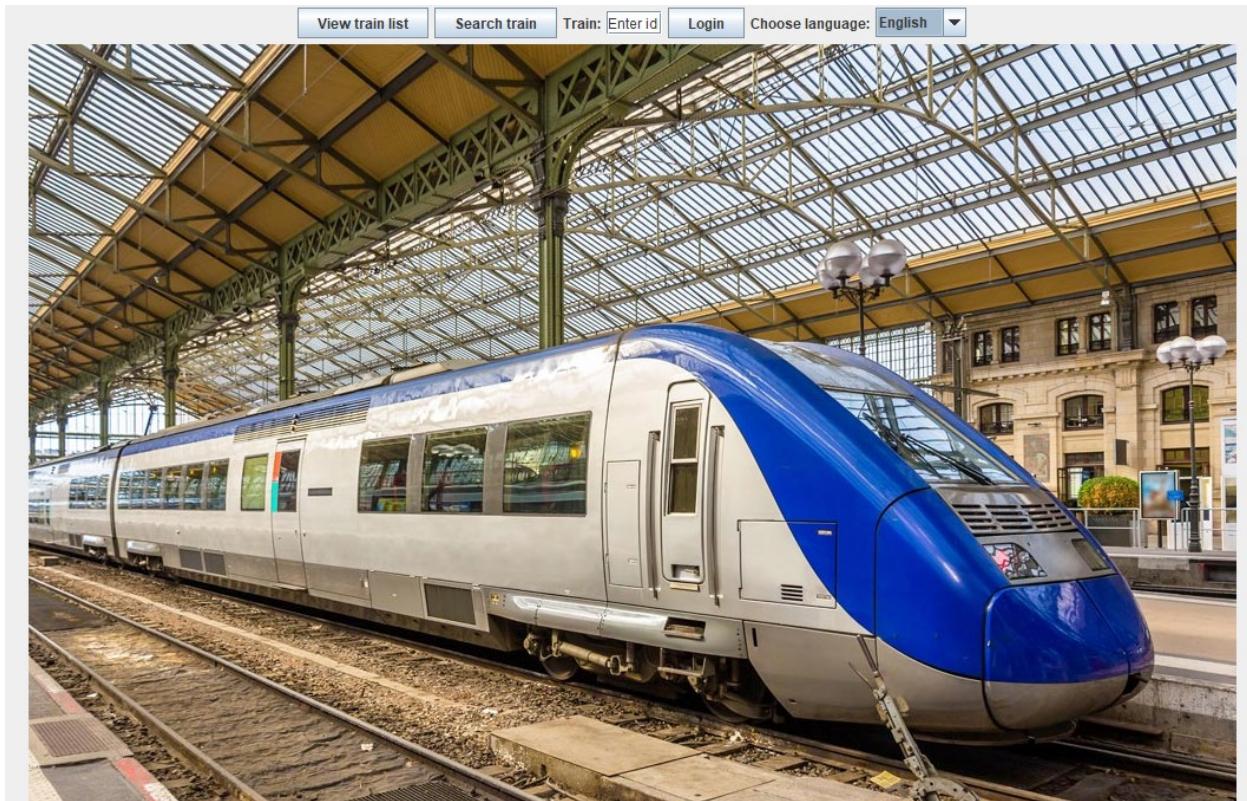


Fig 26. Meniul principal având ca limbă setată engleza

Utilizatorii mai pot să filtreze lista de trenuri după un anumit criteriu, astfel după apăsarea butonului de Vizualizare listă trenuri, se va deschide o nouă fereastră de forma:

A screenshot of a Java Swing application window titled "Vizualizare trenuri". The window contains several dropdown menus and buttons for filtering train searches. At the top right are standard window control buttons (-, □, X). Below the title, there is a dropdown menu labeled "Alegere criteriu" with the option "Statie plecare" selected. To its right is a "Alege" button. Below this, there are two more dropdown menus: "Statie plecare" set to "Bucuresti" and "Statie sosire" also set to "Bucuresti". At the bottom, there is a dropdown menu for "Durata" with the option "< 1h" selected.

Fig 27. Vizualizarea trenurilor

Se observă că trebuie ales mai întâi un criteriu de filtrare: acestea pot fi stația de plecare, stația de sosire, durata cursei sau stația de plecare – stația de sosire.

În cazul în care criteriul selectat este stația de plecare, ne va interesa doar din ce oraș pleacă trenul, respectiv valoarea selectată din combobox-ul specific stației de plecare.

Acest lucru decurge la fel și pentru cazul în care criteriul ales este stația de sosire sau durata, atunci ne va interesa doar valoarea din combobox-ul statiei de sosire, respectiv duratei.

Însă în cazul în care criteriul de filtrare ales este de forma stație de plecare – stație de sosire, ne vor interesa ambele valori din combobox-urile specifice stației de plecare și stației de sosire.

Mai jos se pot observa câteva rezultate după câte o filtrare după fiecare criteriu:

- după stația de plecare unde s-a ales București



Id-ul trenului	Stația de pl...	Stația de s...	Durata în m...	Locuri disp...	Pret
3	Bucuresti	Craiova	650	35	40
13	Bucuresti	Craiova	320	24	49
15	Bucuresti	Pitesti	650	12	40
33	Bucuresti	Craiova	330	84	40
51	Bucuresti	Pitesti	170	103	14
75	Bucuresti	Mangalia	480	111	60
83	Bucuresti	Craiova	650	44	40
105	Bucuresti	Pitesti	650	89	40
313	Bucuresti	Craiova	330	85	43.8

Fig 28. Lista trenurilor care pleacă din București

- după stația de sosire, unde s-a ales Craiova

Id-ul trenului	Statia de pl...	Statia de s...	Durata in m...	Locuri disp...	Pret
3	Bucuresti	Craiova	650	35	40
13	Bucuresti	Craiova	320	24	49
33	Bucuresti	Craiova	330	84	40
63	Cluj-Napoca	Craiova	540	5	85
83	Bucuresti	Craiova	650	44	40
313	Bucuresti	Craiova	330	85	43.8

Fig 29. Lista trenurilor care sosesc în Craiova

- după durată, unde s-a ales < 4h

Id-ul trenului	Statia de pl...	Statia de s...	Durata in m...	Locuri disp...	Pret
7	Iasi	Vaslui	102	49	9.4
51	Bucuresti	Pitești	170	103	14
300	Craiova	Targu-Jiu	125	86	50

Fig 30. Lista trenurilor a căror cursă durează sub 4 ore, echivalentul a 240 de minute

- după stația de plecare – stația de sosire, unde s-a ales București – Pitești

Id-ul trenului	Statia de pl...	Statia de s...	Durata in m...	Locuri disp...	Pret
15	Bucuresti	Pitești	650	12	40
51	Bucuresti	Pitești	170	103	14
105	Bucuresti	Pitești	650	89	40

Fig 31. Lista trenurilor care pleacă din București și ajung în Pitești

Un utilizator mai poate să caute un tren după id-ul acestuia, astfel în pagina principală după introducerea id-ului trenului care dorește să îl caute în căsuța marcată cu Introdu id și apăsarea butonului de Căutare tren, va fi afișat trenul căutat în cazul în care id-ul introdus respectă formatul impus (este număr întreg) și chiar există un tren cu respectivul id.

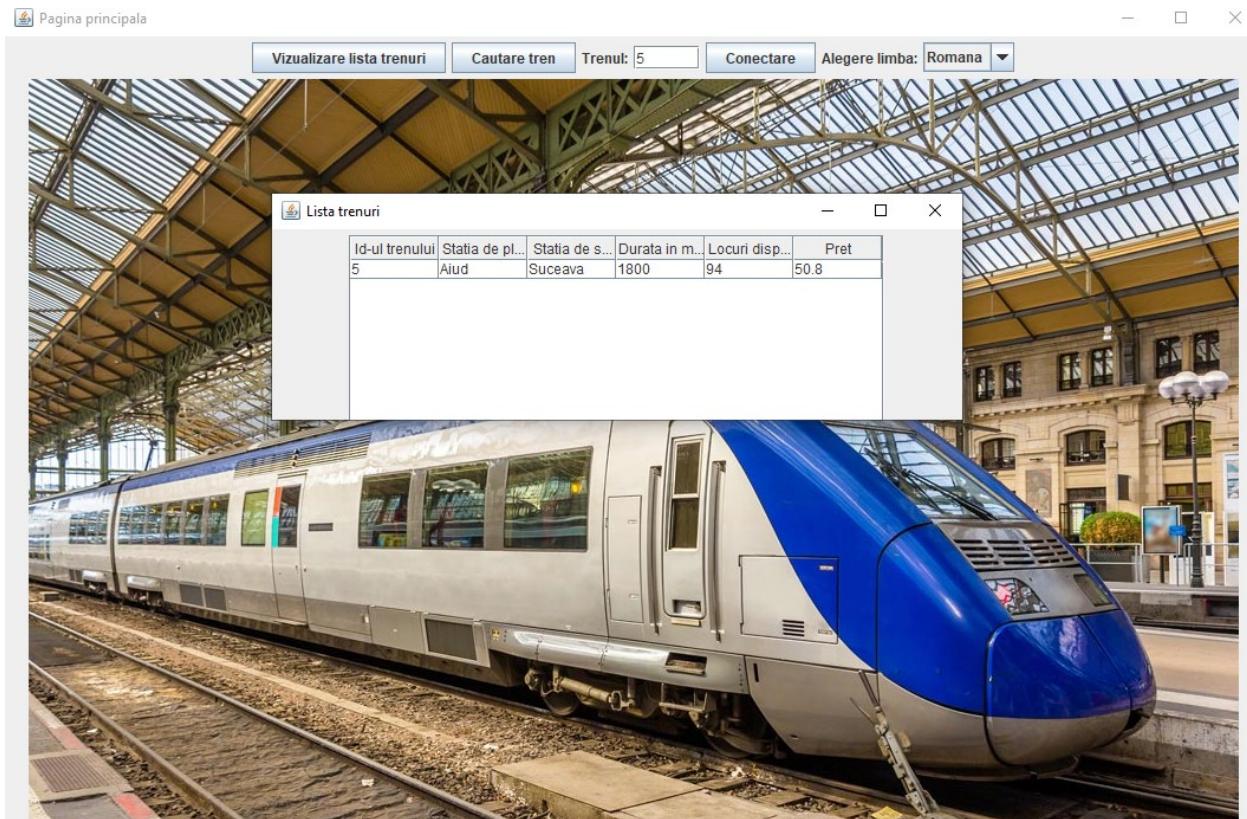


Fig 32. Căutarea trenului cu id-ul 5

În cazul în care id-ul introdus respectă formatul impus însă nu există un tren cu acest id, se va deschide o fereastră care să ne avertizeze cu privire la acest lucru:

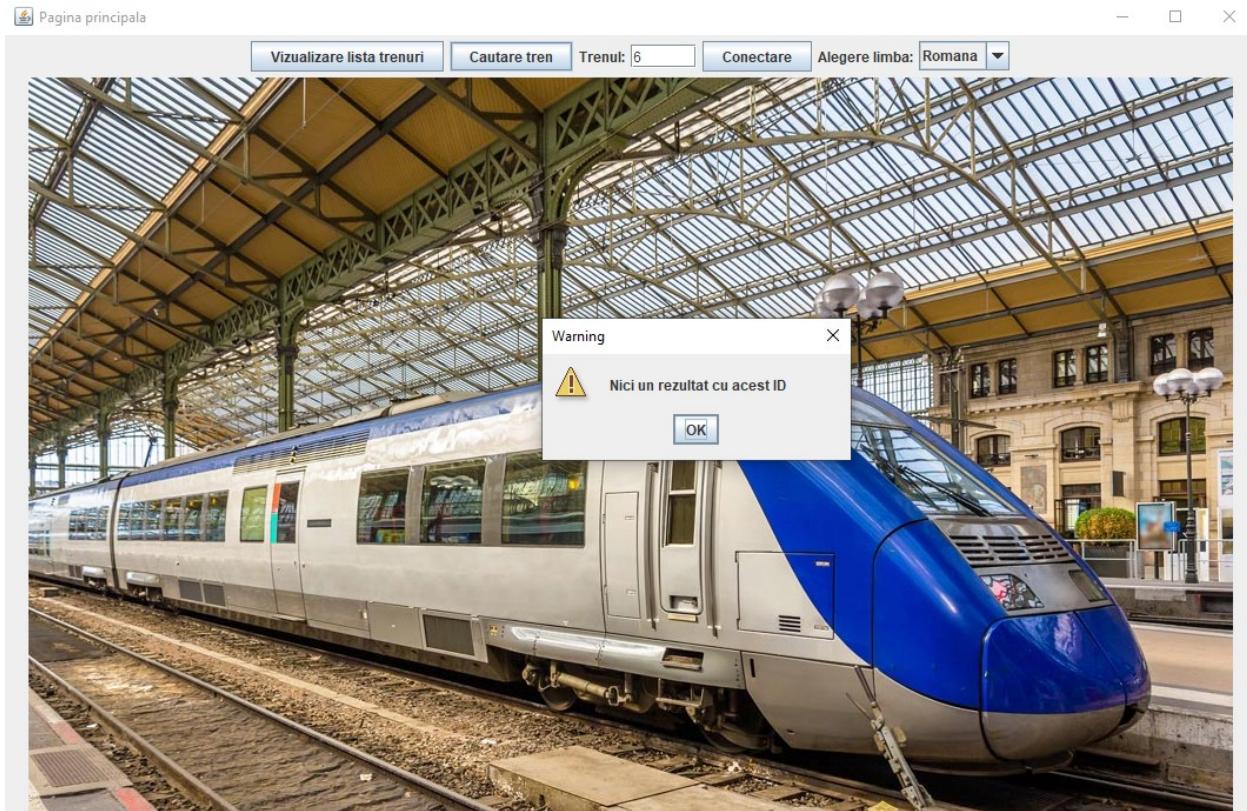


Fig 33. Avertizare cu privire la lipsa trenului care a fost căutat

Și nu în ultimul rând, în cazul în care se introduce un id ce nu respectă formatul impus, se va deschide o altă fereastră care ne va semnala acest lucru:

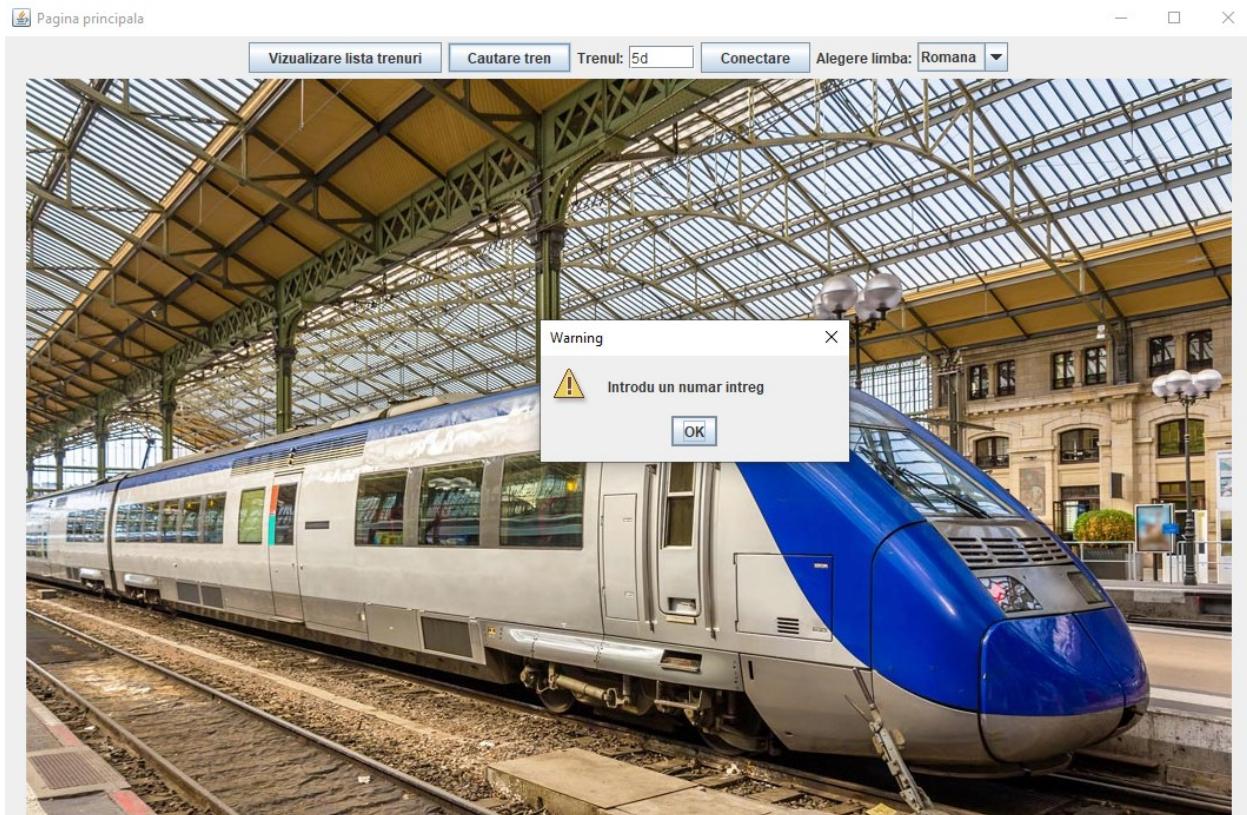


Fig 34. Avertizare cu privire la formatul id-ului introdus

Dacă vorbim de un angajat sau administrator, acesta va încerca să se conecteze apăsând butonul de conectare de pe pagina principală, după care se va deschide o nouă pagină în care vor trebui introduse numele și parola pentru contul de angajat/administrator:

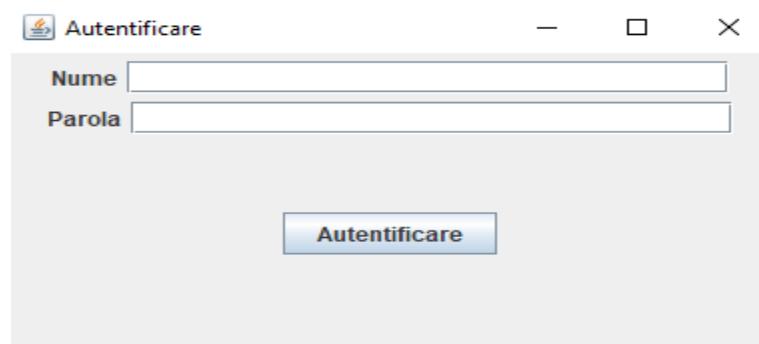


Fig 35. Fereastra de autentificare

În cazul în care datele introduse, corespundeau unui angajat, se va deschide o pagină specifică pentru acesta:

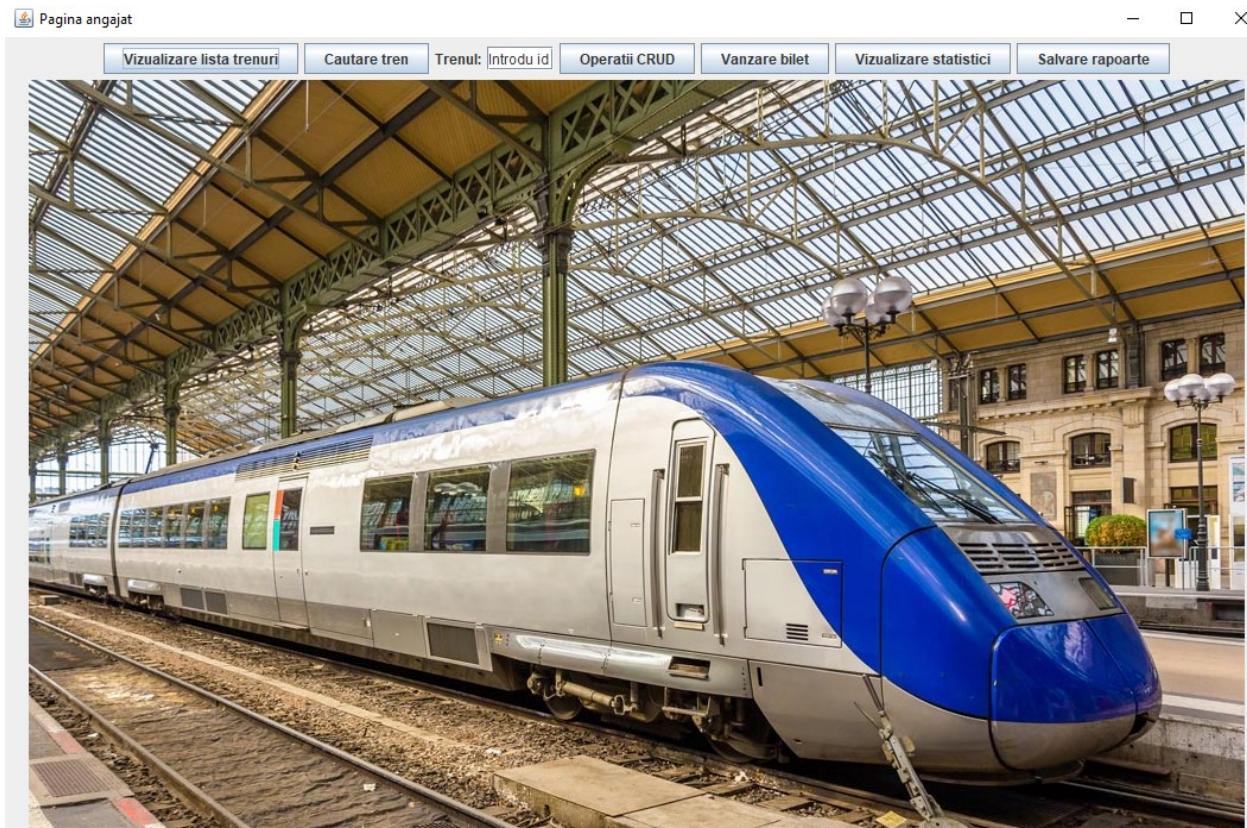


Fig 36. Pagina principală a angajatului

Dacă era vorba de un administrator, se va deschide o altă pagină:

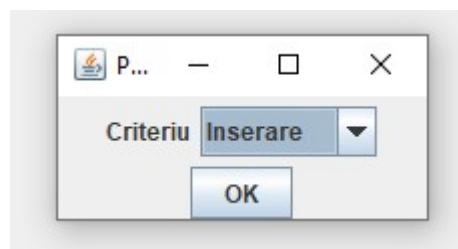


Fig 37. Pagina principală a administratorului

Iar dacă datele introduse nu au fost valide, se va deschide o fereastră care ne va avertiza cu privire la acest fapt:

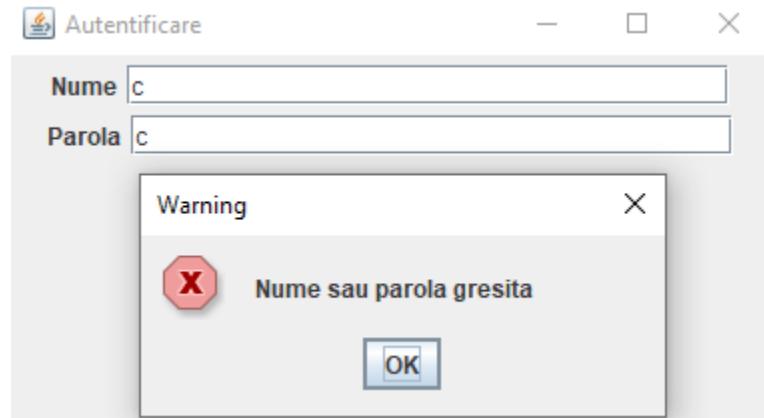


Fig 38. Autentificare eșuată

Ca angajat, putem realiza aceleasi operații care puteau fi realizate de către un utilizator normal, în plus se mai pot realiza operații precum :

Vizualizarea statisticilor în legătură cu prețul biletelor care au fost vândute și câte trenuri pentru care s-a tipărit un bilet pleacă/vin dintr-un/într-un anumit oraș.

Pentru această funcție trebuie doar apăsat butonul de Vizualizare statistici, după care se va deschide o nouă fereastră în care se va anunța dacă operațiunea a fost realizată cu succes urmând ca mai apoi să ne arate cele 2 grafice:

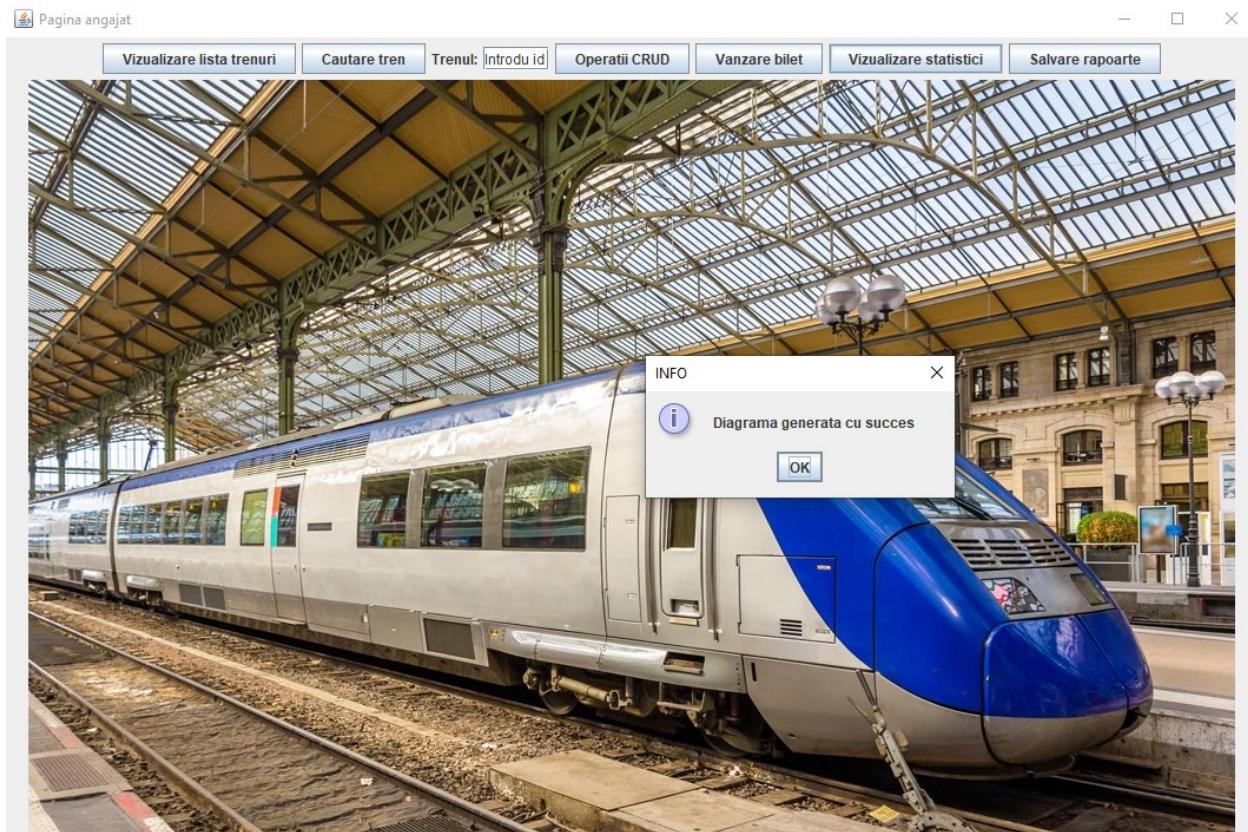


Fig 39. Notificare cu privire la reușita generării diagramei

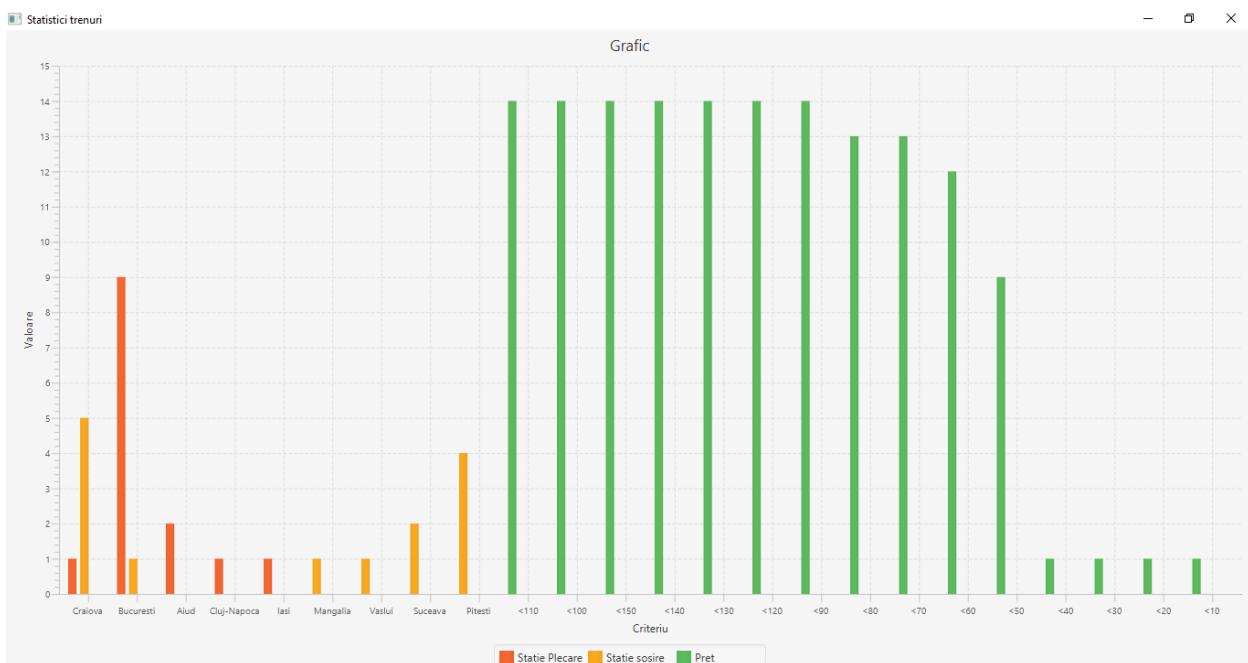


Fig 40. Graficul realizat

În cazul în care un angajat dorește să salveze rapoarte cu privire la trenurile existente în baza de date, acesta va apăsa butonul de Salvare rapoarte, în urma căruia se va afișa un mesaj cu reușita operației, după care în zona locală a proiectului, pe parte de client, vor apărea 3 fișiere cu extensii diferite: xml, json și csv.

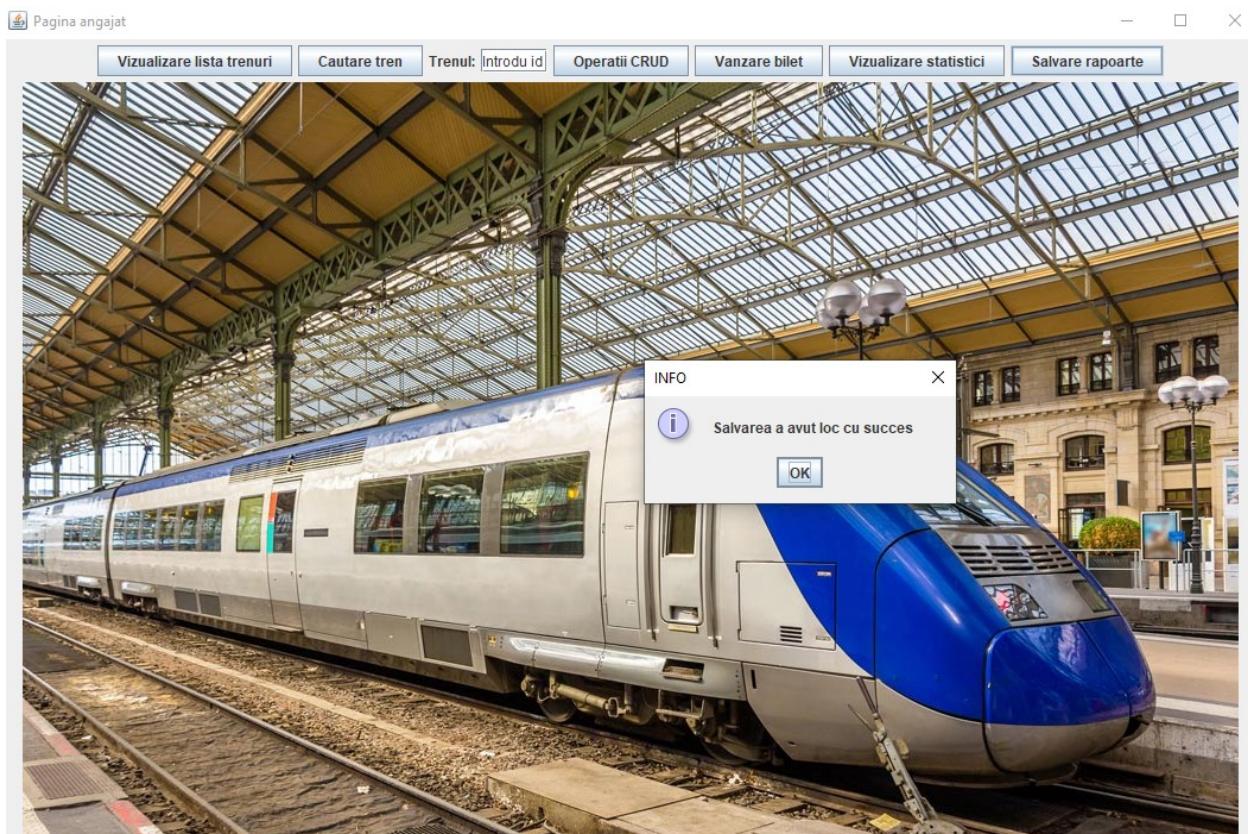


Fig 41. Raport cu privire la succesul informației

trains	5/25/2020 6:57 PM	OpenOffice.org 1....	1 KB
trains	5/25/2020 6:57 PM	Source code file	31 KB
trains	5/25/2020 6:57 PM	XML Document	9 KB

Fig 42. Fișierele create după operația de salvare a rapoartelor despre trenuri

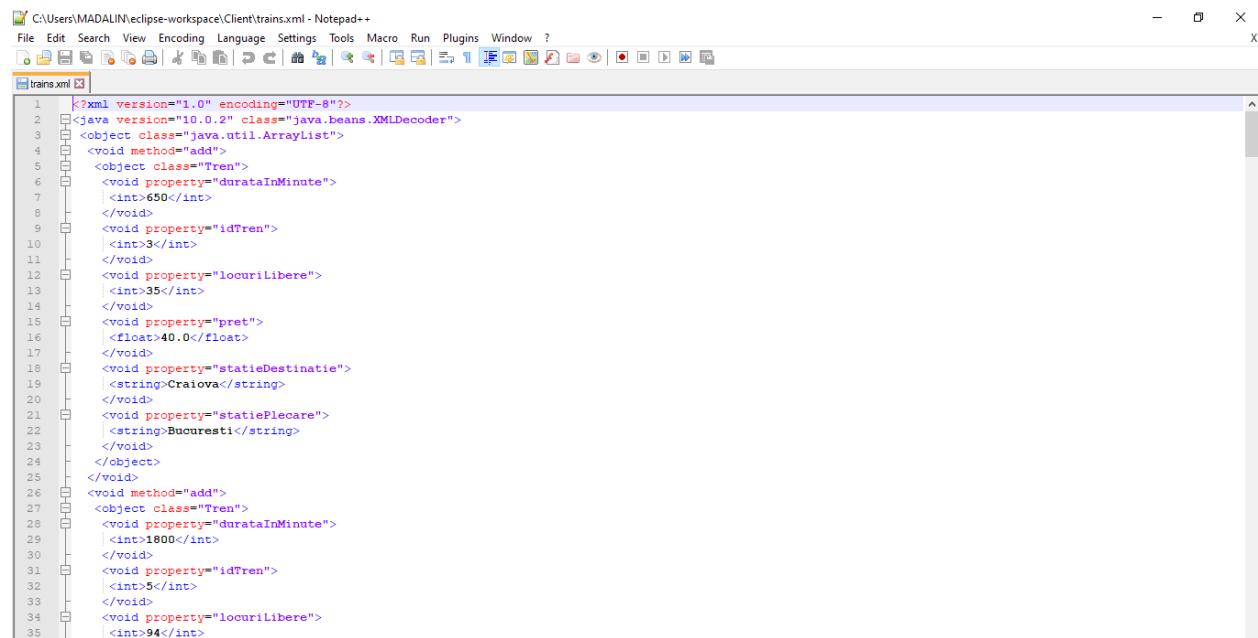
Conținutul acestor fișiere arată în felul următor:

Fișierul csv:

	A	B	C	D	E	F
1	3	Bucuresti	Craiova	650	35	
2	5	Aiud	Suceava	1800	94	
3	7	Iasi	Vaslui	102	49	
4	13	Bucuresti	Craiova	320	24	
5	15	Bucuresti	Pitesti	650	12	
6	33	Bucuresti	Craiova	330	84	
7	51	Bucuresti	Pitesti	170	103	
8	63	Cluj-Napoca	Craiova	540	5	
9	69	Budapest	Gheorghe	333	54	
10	75	Bucuresti	Mangalia	480	111	
11	77	Targu-Jiu	Budapest	444	53	
12	83	Bucuresti	Craiova	650	44	
13	93	Craiova	Bucuresti	333	69	
14	105	Bucuresti	Pitesti	650	89	
15	200	Iasi	Cluj-Napoca	540	13	
16	300	Craiova	Targu-Jiu	125	86	
17	313	Bucuresti	Craiova	330	85	

Fig 43. Conținutul fișierului csv

Fișierul xml:



```

<?xml version="1.0" encoding="UTF-8"?>
<java version="10.0.2" class="java.beans.XMLDecoder">
<object class="java.util.ArrayList">
<void method="add">
<object class="Tren">
<void property="durataInMinute">
<int>650</int>
</void>
<void property="idTren">
<int>3</int>
</void>
<void property="locuriLibere">
<int>35</int>
</void>
<void property="pret">
<float>40.0</float>
</void>
<void property="statieDestinatie">
<string>Craiova</string>
</void>
<void property="statiePlecare">
<string>Bucuresti</string>
</void>
</object>
</void>
<void method="add">
<object class="Tren">
<void property="durataInMinute">
<int>1800</int>
</void>
<void property="idTren">
<int>5</int>
</void>
<void property="locuriLibere">
<int>94</int>

```

Fig 44. Conținutul fișierului xml

Fisierul json:

Fig 45. Conținutul fișierului json

Un angajat de asemenea poate să vândă un bilet către un călător, pentru a realiza acest lucru se va apăsa pe butonul Vânzare bilet, după care o nouă fereastră se va deschide, unde vor trebui completate date cu privire la id-ul biletului nou care va fi introdus (acesta trebuie să fie unic), numele, prenumele, telefonul, adresa de e-mail și cnp-ul cumpărătorului, id-ul trenului pentru care se dorește să se cumpere bilet.

În funcție de id-ul trenului pentru care se cumpără bilet, se va calcula automat și prețul biletului.

În cazul în care id-ul biletul introdus nu era valid în sensul că exista deja un bilet cu acel id sau formatul id-ul introdus nu este permis, ni se va atrage atenția cu privire la acest lucru, acest lucru se repetă și în cazul în care id-ul trenului nu este valid, deoarece ca să cumpărăm un bilet la un anumit tren, trenul respectiv trebuie să existe în primul rând.

Formatul mai trebuie respectat și la introducerea CNP-ului clientului, deoarece știm ca CNP-ul este percepțut ca un număr întreg de dimensiuni foarte mari (long) .

Datele ce le vom introduce:



Fig 46. Completare detalii bilet

Rezultatul introducerii biletului:

Id-ul bil...	Nume c...	Prenum...	Nr. telef...	E-mail	CNP	Id-ul tre...	Pretul bi...
53	Maria	Ioana	076000...	getfit98...	2020	5	50.8
13	Maria	Ioana	076000...	getfit98...	2020	3	40
52	Maria	Ioana	076000...	getfit98...	2020	15	40
43	Maria	Ioana	076000...	getfit98...	2020	63	85
57	Maria	Ioana	076000...	getfit98...	2020	75	60
19	Maria	Ioana	076000...	getfit98...	2020	83	40
20	Maria	Ioana	076000...	getfit98...	2020	7	9.4
38	Maria	Ioana	076000...	getfit98...	2020	93	59.4
56	Maria	Ioana	076000...	getfit98...	2020	105	40
15	Maria	Ioana	076000...	getfit98...	2020	3	40
222	Maria	Ioana	076000...	getfit98...	2020	3	40
333	Maria	Ioana	076000...	getfit98...	2020	105	40
115	Maria	Ioana	076000...	getfit98...	2020	105	40
77	Ana	Maria	076000...	getfit98...	12345	5	50.8
97	Florea	Adrian	076666...	java@y...	1234	5	50.8

Fig 47. Lista completă a biletelor vândute

Și nu în ultimul rând, un angajat mai poate efectua operații CRUD în ceea ce privește trenurile și biletele stocate în baza de date.

Pentru realizarea acestei operații, se va apăsa butonul de Operații CRUD din meniul principal al angajatului, după care se va deschide o nouă fereastră de unde vom putea selecta operația pe care dorim să o efectuăm (inserare, selecție, modificare sau ștergere) și obiectul pe care dorim să efectuăm operația aleasă (tren sau bilet).

În cazul în care alegem operația de inserare, nu va mai trebui completat id-ul obiectului cerut în fereastra curentă, deoarece îl vom completa ulterior în fereastra ce se va deschide după apăsarea butonului de OK împreună cu restul datelor pentru noul obiect ce se dorește a fi introdus.

Id-ul obiectului care ne este cerut în prima fereastră, trebuie completat doar la operațiile de selectare, modificare sau ștergere, deoarece trebuie să știm de dinainte pe ce obiect vom aplica operația respectivă.

Bineînțeles, la fel ca și la alte operații, și aici sunt anunțate eventualele greșeli în introducerea datelor, fie că vorbim de formatul acestora sau de simpla lipsă a obiectului pe care se dorește aplicarea unei anumite operații.

Aceste greșeli împreună cu fereastrele ce ne-ar avertiza nu vor mai fi ilustrare datorită cazuri multiple de combinații care pot exista și a încercării de a limita cât de mult posibil dimensiunea acestui document.

Câteva imagini de test:

- pentru inserarea unui tren

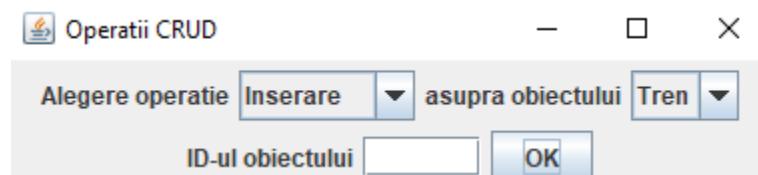


Fig 48. Selectarea operației de inserare

ID: 242
Statie plecare: Cluj-Napoca
Statie sosire: Targu-Jiu
Durata: 467
Locuri libere: 68
Pret: 84.25

Fig 49. Introducere date tren pentru inserare

Id-ul trenului	Statia de pl...	Statia de s...	Durata in m...	Locuri disp...	Pret
3	Bucuresti	Craiova	650	35	40
5	Aiud	Suceava	1800	93	50.8
7	Iasi	Vaslui	102	49	9.4
13	Bucuresti	Craiova	320	24	49
15	Bucuresti	Pitesti	650	12	40
33	Bucuresti	Craiova	330	84	40
51	Bucuresti	Pitesti	170	103	14
63	Cluj-Napoca	Craiova	540	5	85
69	Budapest	Gheorgheni	333	54	131
75	Bucuresti	Mangalia	480	111	60
77	Targu-Jiu	Budapest	444	53	55
83	Bucuresti	Craiova	650	44	40
93	Craiova	Bucuresti	333	69	59.4
105	Bucuresti	Pitesti	650	89	40
200	Iasi	Cluj-Napoca	540	13	92.35
242	Cluj-Napoca	Targu-Jiu	467	68	84.25
300	Craiova	Targu-Jiu	125	86	50
313	Bucuresti	Craiova	330	85	43.8

Fig 50. Afisarea tuturor trenurilor, impreuna cu cel inserat anterior

- pentru selecția unui tren



Fig 51. Selectare operației de selecție

Id-ul trenului	Stația de pl...	Stația de s...	Durata în m...	Locuri disp...	Pret
242	Cluj-Napoca	Targu-Jiu	435	85	88.3

Fig 52. Afisare date despre trenul selectat

- pentru modificarea unui tren



Fig 53. Selectare operației de modificare

Operatii C...

Statie plecare	Targu-Jiu
Statie sosire	Cluj-Napoca
Durata	500
Locuri	99
Pret	92.32

OK

Fig 54. Completare date pentru modificare

Lista trenuri

Id-ul trenului	Stacia de plecare	Stacia de sosire	Durata in minute	Locuri disponibile	Pret
3	Bucuresti	Craiova	650	35	40
5	Aiud	Suceava	1800	93	50.8
7	Iasi	Vaslui	102	49	9.4
13	Bucuresti	Craiova	320	24	49
15	Bucuresti	Pitesti	650	12	40
33	Bucuresti	Craiova	330	84	40
51	Bucuresti	Pitesti	170	103	14
63	Cluj-Napoca	Craiova	540	5	85
69	Budapest	Gheorgheni	333	54	131
75	Bucuresti	Mangalia	480	111	60
77	Targu-Jiu	Budapest	444	53	55
83	Bucuresti	Craiova	650	44	40
93	Craiova	Bucuresti	333	69	59.4
105	Bucuresti	Pitesti	650	89	40
200	Iasi	Cluj-Napoca	540	13	92.35
242	Targu-Jiu	Cluj-Napoca	500	99	92.32
300	Craiova	Targu-Jiu	125	86	50
313	Bucuresti	Craiova	330	85	43.8

Fig 55. Afisarea listei tuturor trenurilor in care se observa modificarea datelor pentru trenul cu id-ul respectiv

- pentru ștergerea unui tren



Fig 56. Selectare operație de stergere

The screenshot shows a window titled 'Lista trenuri'. It displays a table with the following columns: Id-ul trenului, Statia de plecare, Statia de sosire, Durata in minute, Locuri disponibile, and Pret. The table contains 18 rows of data. The last row shown is 313, which corresponds to the train that was deleted in Fig 56.

Id-ul trenului	Statia de plecare	Statia de sosire	Durata in minute	Locuri disponibile	Pret
3	Bucuresti	Craiova	650	35	40
5	Aiud	Suceava	1800	93	50.8
7	Iasi	Vaslui	102	49	9.4
13	Bucuresti	Craiova	320	24	49
15	Bucuresti	Pitesti	650	12	40
33	Bucuresti	Craiova	330	84	40
51	Bucuresti	Pitesti	170	103	14
63	Cluj-Napoca	Craiova	540	5	85
69	Budapesta	Gheorgheni	333	54	131
75	Bucuresti	Mangalia	480	111	60
77	Targu-Jiu	Budapeste	444	53	55
83	Bucuresti	Craiova	650	44	40
93	Craiova	Bucuresti	333	69	59.4
105	Bucuresti	Pitesti	650	89	40
200	Iasi	Cluj-Napoca	540	13	92.35
300	Craiova	Targu-Jiu	125	86	50
313	Bucuresti	Craiova	330	85	43.8

Fig 57. Afisarea tuturor trenurilor, astfel se observă ca trenul șters anterior nu mai există în baza de date

Iar ca administrator putem de asemenea să efectuăm operații CRUD, însă pe angajați, nu pe bilete sau pe trenuri.

Principiile sunt aceleiasi, cu excepția faptului ca angajații care se doresc a fi selectați, modificați sau ștersi, trebuie identificați pe baza adresei de e-mail.