Universitatea Tehnică a Moldovei
Facultatea Calculatoare, Informatică și Microelectronică
Departamentul Ingineria Software și Automatică

Proiect de an

la disciplina
Analiza și Modelarea Orientată pe Obiecte

Tema: Analiza și modelarea sistemului „Lanasatorul jocului Liga Legendelor”

Studentul gr. TI-173: Heghea Nicolae
Conducător: lector universitar, Sava Nina
lector universitar, Melnic Radu

Chișinău 2019

Cuprins

Scopul

Studiul tipurilor de diagrame UML, caracteristicile lor, elementele principale, aplicațiile și regulile de construcție.

Sarcina

Efectuați analiza și modelarea sistemului “Lanasatorul jocului Liga Legendelor” utilizând diagrame UML, utilizând software-ul de modelare Enterprise Architect .

Introducere

Această lucrare va explora principiile care stau la baza analizei și modelarea orientate pe obiecte, vor analiza și implementa cunoștințele despre diagramele UML - reprezentări din diferite perspective ale proceselor care apar în aplicație, care ne permit să înțelegem mai bine funcționarea programului, modul în care acesta este realizat și, în cazul originea problemei este sursa ei.

Înainte de a proiecta un sistem de orice scară, orice inginer trebuie să efectueze un studiu al problemei, apoi să transfere problema practică la un model matematic și apoi să construiască un model al sistemului real folosind acest model. În această lucrare, cea mai mare prioritate nu este o descriere detaliată a întregului sistem, ci studiul metodelor de analiză și proiectare a sistemelor.

Pentru cele mai vizuale și modele simplificate și modele, există un limbaj unificat de modelare UML 2.0. Numărul relativ mare de tipuri de diagrame UML ne oferă diferite posibilități de reprezentare a sistemului selectat, creând spațiu liber și liber pentru crearea unei diagrame.

Unul dintre instrumentele care acceptă specificația limbajului UML 2.0 este software-ul de modelare Enterprise Architect. Acest instrument va fi folosit, în comparație cu alte instrumente care suportă specificațiile limbii UML 2.0, Enterprise Architect are o interfață mai intuitivă și mai convenabilă.

Despre importanța sistemului ales

Sistemul informaţional al “Lansatorului jocului Liga Legendelor” va conține statistici, crearea inventariilor, cumpărături în magazinul propus, jocul în sine, interacțiuni cu alți jucători, indeplinirea misiunilor, și feadback.

Clientul este programul din care jucătorii interacționează cu League of Legends, fără a juca acest joc.

1 Proiectarea sistemului informatic

Sistemul informaţional al “Lansatorului jocului Liga Legendelor” este separat în secțiuni:

Meniul „***Redare”***

Pagina implicită la care intri în momentul în care vă conectați. În partea din stânga sus, ar trebui să vedeți butonul albastru "Redare". Folosind-o, puteți alege modul dorit și introduceți o coadă în cazul unui joc potrivit, căutați un joc manual folosind funcția de joc personalizat sau chiar antrenați jucând tutorialele.

Meniul „***Social***”

În meniul din dreapta sunt prezentate informațiile de bază ale utilizatorului: pictograma, numele summoner, mesajul personalizat și notificările. De asemenea, puteți vedea clubul dvs., dacă aveți unul, lista de prieteni și lista persoanelor cu care ați jucat recent. În partea dreaptă jos a clientului puteți vedea trei butoane: mesajele, misiunile dvs. și "Raportați un bug

Suma ***RP*** **/** ***BE*** (**R**iot **P**oints / **B**lue **E**ssences)

Puteți vedea suma dvs. de monedă lângă meniul social. Numărul din stânga indică suma dvs. de puncte de revoltă și Blue Essences dreapta. Puteți utiliza Riot Points și Blue Essences în magazin. O informație mai inteligentă despre chematorul tău poate fi văzută făcând clic pe butonul "Profil".

Pagina „Acasă”

În pagina de pornire, în fila "Prezentare generală", puteți vedea evenimentele curente, iar cele mai recente campioni sau care avatare sunt lansate. Fila "Știri" prezintă toate știrile și viitoarele modificări planificate. Fila "Patch Notes" vă arată ultimele informații despre patch-uri.

Pagina Profilului

Puteți accesa informațiile despre profilul dvs. făcând clic pe "Profil" în partea de sus a paginii de pornire.

Pagina „Colecție”

De asemenea, puteți vedea toați campionii, avatarele, emoti, rune, vrăji summoner și seturi de articole pe care le aveți în fila "Colecție", în partea de sus a paginii de pornire.

1.1 Etapele creării unui program

Procesul de creare a aplicațiilor software complexe este imposibil de imaginat fără a împărți ciclul de viață al software-ului în etape. Ciclul de viață al programului este următoarea totalitate a etapelor existenței sale:

- analiza domeniului și crearea specificațiilor tehnice (în acest stadiu proiectul există doar "pe hârtie");
- proiectarea structurii programului - stadiul creării documentației (diagramei) care descrie interacțiunile din cadrul și cu sistemul;
- codificare - stadiul implementării directe a sistemului în conformitate cu documentația de proiect;
- testarea și depanarea - etapa de verificare a respectării cerințelor de implementare, identificarea erorilor și eliminarea acestora etc. ;
- introducerea programului - stadiul în care produsul finit este transferat clientului, ieșirea produsului pe piață;
- întreținerea programului - etapa de susținere a proiectului, adăugarea sau schimbarea funcționalității, rezolvarea problemelor și a erorilor care nu au fost identificate la etapa de testare și verificare;
- eliminarea - utilizarea directă a produsului.

1.2 Noțiuni teoretice

Unified Modeling Language (UML) este un limbaj standard pentru descrierea de modele și specificații pentru software. Limbajul a fost creat de către consorțiul Object Management Group (OMG) care a mai produs printre altele și standardul de schimb de mesaje intre sisteme CORBA. UML a fost la bază dezvoltat pentru reprezentarea complexității programelor orientate pe obiect, al căror fundament este structurarea programelor pe clase, și instanțele acestora (numite și obiecte). Cu toate acestea, datorită eficienței și clarității în reprezentarea unor elemente abstracte, UML este utilizat dincolo de domeniul IT. Așa se face că există aplicații ale UML-ului pentru management de proiecte, pentru business Process Design etc.

UML este un limbaj grafic al vizualizării, descrierea parametrilor, proiectarea și documentarea diferitelor sisteme (în special programe). Graficele sunt create folosind instrumente speciale CASE (Inginerie Software pentru Inginerie Software), cum ar fi Rational Rose și Enterprise Architect (utilizate de noi). Pe baza tehnologiei UML, este construit un singur model de informare. Instrumentele CASE menționate mai sus sunt capabile să genereze coduri în diferite limbi orientate pe obiecte și au, de asemenea, funcția de inginerie inversă (procesul de creare a unui model grafic din codul de program existent și comentariile la acesta).

Tipurile de diagrame cele mai utilizate pentru vizualizarea modelului:

- diagramele cazurilor de utilizare;
- diagramele de secvență;
- diagramele de colaborare;
- diagramele de clase;
- diagramele de stari;
- diagramele de activitați;
- diagramele de componente;
- diagramele de amplasare.

Graficele, prin programare, sunt împărțite în două tipuri principale: diagramele de comportament și diagramele de structură.

1.2.1 **Diagrame de comportament**

Cele cinci diagrame comportamentale de bază din UML sunt utilizate pentru a reprezenta, preciza, construi și documenta aspectele dinamice ale unui sistem.

- diagramele cazurilor de utilizare;
- diagramele de secvențe;
- diagramele de colaborare;
- diagramele de stari;
- diagramele de activitați.

Diagrama Cazurilor de Utilizare

Sistemul proiectat este reprezentat ca un set de entități sau actori care interacționează cu sistemul cu ajutorul așa-numitelor precedente. Cu alte cuvinte, fiecare caz de utilizare definește un set de acțiuni efectuate de sistem în timpul unui dialog (interacțiune) cu un actor. În același timp, nu sunt specificate detalii despre organizarea și implementarea interacțiunii dintre actor și sistem. Pur și simplu, descriem "ce poate face un actor", dar nu spuneți "cum reacționează sistemul".

Un actor este orice entitate care interacționează cu sistemul din exterior. Poate fi o persoană, un dispozitiv tehnic, un program sau orice alt sistem care poate servi drept sursă de influență asupra sistemului simulat.

Un caz de utilizare (ca componentă) este un element folosit pentru a descrie capacitățile pe care sistemul le oferă unui actor. Cu alte cuvinte, fiecare caz de utilizare definește un anumit set de acțiuni efectuate de sistem în timpul unui dialog cu un actor.

Interfața este utilizată pentru a specifica parametrii modelului care sunt vizibili din exterior fără a specifica structura lor internă. Interfața nu numai că separă specificațiile operațiilor sistemului de la implementarea lor, ci definește și limitele generale ale sistemului proiectat.

Relațiile privind diagrama cazurilor de utilizare:

- asociația (asociere) servește pentru a indica rolul specific al actorului într-un caz de utilizare separată. Cu alte cuvinte, această relație stabilește rolul specific pe care îl joacă un actor atunci când interacționează cu o instanță a unui caz de utilizare;

- extinderea relației definește relația instanțelor unui caz de o singură utilizare cu o versiune mai generală, ale cărei proprietăți sunt determinate pe baza metodei de asociere comună a acestor instanțe;

- generalizarea (relația de generalizare) indică faptul că un caz de utilizare A poate fi generalizat pentru a folosi cazul B. În acest caz, opțiunea A va fi o specializare a opțiunii B. În acest caz, B este numit strămoș sau părinte A, iar opțiunea A este descendent în ceea ce privește cazul de utilizare B;

- includerea - indică faptul că un comportament specificat pentru un caz de utilizare este inclus ca o componentă integrală în secvența de comportament a unui alt caz de utilizare.

Diagrama secvențelor

Pentru a simula interacțiunea obiectelor în limba UML, se utilizează diagrame de interacțiune. Interacțiunea obiectelor poate fi văzută în timp și apoi o diagramă de secvență este utilizată pentru a reprezenta caracteristicile temporale ale transmiterii și recepționării mesajelor între obiecte. Interacțiunea obiectelor schimbă unele informații (mesaje) între ele. Diagrama de secvențe arată numai acele obiecte care sunt direct implicate în interacțiune și nu prezintă asocieri posibile posibile cu alte obiecte.

Linia de viață a obiectului - este reprezentată ca o linie verticală punctată asociată cu un singur obiect din diagrama succesivă. Linia de viață este utilizată pentru a indica perioada de timp în care un obiect există în sistem și, prin urmare, poate participa potențial la toate interacțiunile sale.

Stare activă - în timpul funcționării sistemelor orientate pe obiecte, unele obiecte pot fi în stare activă, executând în mod direct anumite acțiuni sau într-o stare de așteptare pasivă pentru mesaje de la alte obiecte. Pentru a distinge în mod explicit o astfel de activitate a obiectelor, se folosește un concept special în limba UML, numită concentrarea controlului.

Un mesaj - este o informație completă care este trimisă de la un obiect la altul.

UML furnizează câteva acțiuni standard care trebuie luate ca răspuns la primirea mesajului corespunzător. Acestea pot fi indicate în mod clar pe diagrama succesivă sub forma unui stereotip de lângă mesajul lor. În acest caz, acestea sunt scrise în citate. Următoarea notație este utilizată pentru a modela acțiunile:

- "*call*" (apel) - un mesaj care necesită un apel la operația sau procedura obiectului care primește;
- "*return*" (întoarcere) - un mesaj care returnează valoarea operației finalizate la obiectul care a numit-o;
- "*create*" (crearea) - un mesaj care necesită crearea unui alt obiect pentru a efectua anumite acțiuni;
- "*destoy*" (distruge) - un mesaj cu o cerere explicită de distrugere a obiectului corespunzător. Se trimite atunci când este necesar să se oprească acțiunile nedorite de la un obiect existent în sistem sau atunci când un obiect nu mai este necesar și trebuie să elibereze resursele de sistem pe care le angajează;
- "*send*" (trimite) - indică trimiterea către un alt obiect a unui semnal, inițiat asincron de un obiect și recepționat (interceptat) de un alt obiect.

Diagrama de colaborare

Pe diagrama de cooperare, dreptunghiurile descriu obiectele care participă la o interacțiune, conținând numele obiectului, clasa sa și, eventual, valorile atributului. Ca și în diagrama de clasă, asociațiile dintre obiecte sunt indicate sub forma unor linii de conectare diferite. În acest caz, puteți specifica în mod explicit numele asociațiilor și rolurile jucate de obiectele din asociația dată. Spre deosebire de diagrama de secvențe, numai relațiile dintre obiectele care joacă anumite roluri în interacțiune sunt reprezentate în diagrama de colaborare.

Pentru reprezentarea grafică a obiectelor în diagrama de cooperare, același simbol dreptunghi este utilizat ca și pentru clase.

Un obiect este o instanță separată a unei clase care este creată în stadiul de execuție a programului. Poate avea propriul nume și valori specifice ale atributelor. Pentru obiecte, formatul de formatare al șirului de clasificatori este completat de numele obiectului și are următoarea formă:

<*Numele obiectului*> '/' <*Numele rolului clasificatorului*> ':' <*Denumirea clasificatorului*>

Un multiobiect este un întreg set de obiecte la unul dintre capetele unei asociații. În diagrama de cooperare, un Multi-Object este utilizat pentru a afișa operațiile și semnalele care sunt adresate întregului set de obiecte, și nu doar unul.

Un obiect compozit sau un obiect container este destinat să reprezinte un obiect care are structura proprie și firele interne de control. Un obiect compus este o instanță a unei clase compozite (clasa container) care este legată de o relație de agregare sau compoziție cu părțile sale.

O legătură este o instanță sau un exemplu de asociere arbitrară. Comunicarea ca element al limbajului UML poate avea loc între două sau mai multe obiecte. O conexiune poate avea unele stereotipuri care sunt scrise lângă unul dintre capetele sale și indică particularitatea implementării acestei conexiuni. Următoarele stereotipuri pot fi utilizate în UML în acest scop:

- "asociere" este o asociere (asumată implicit, prin urmare acest stereotip poate fi omis);
- "parametru" - parametru de metodă. Obiectul corespunzător poate fi doar un parametru al unei anumite metode;

- "local" - variabilă a metodei locale. Domeniul său de aplicare este limitat la obiectul vecin;
- "global" este o variabilă globală. Domeniul său de aplicare se extinde la întreaga diagramă a cooperării;
- "auto" - o conexiune reflexivă a unui obiect cu el însuși, care permite obiectului să transmită un mesaj la sine. În diagrama de cooperare, relația reflexivă este reprezentată de o buclă în partea superioară a dreptunghiului obiectului.

Conceptul de colaborare (colaborare) este folosit pentru a desemna setul de obiecte care interacționează cu un scop specific în contextul general al sistemului simulat. Scopul cooperării în sine este de a specifica caracteristicile implementării unor operațiuni individuale cele mai semnificative din sistem.

Diagrama de stari

Scopul principal al acestei diagrame este de a descrie posibilele secvențe de stări și tranziții care caracterizează colectiv comportamentul elementului model în timpul ciclului său de viață. O diagramă de stare reprezintă comportamentul dinamic al entităților, pe baza specificării răspunsului lor la percepția anumitor evenimente specifice.

O stare - este o metaclasă abstractă utilizată pentru a modela o situație particulară în timpul căreia este îndeplinită o anumită condiție.

O tranziție - simplă este o relație între două state succesive, ceea ce indică faptul că o stare se schimbă în alta.

Un eveniment (eveniment), formal, este o precizare a unui fapt care are loc în spațiu și în timp. Despre evenimente se spune că "se întâmplă", în timp ce evenimentele individuale trebuie ordonate la timp. După apariția unui anumit eveniment, nu se poate reveni la evenimentele anterioare, cu excepția cazului în care această posibilitate este prevăzută în mod explicit în model. În limba UML, evenimentele joacă rolul de stimuli care inițiază tranziții de la un stat la altul. Semnalele, apelurile, sfârșitul perioadelor fixe sau momentele de încheiere a anumitor acțiuni pot fi considerate evenimente.

Condiția de pază, dacă există, este întotdeauna scrisă în paranteze drepte după evenimentul declanșator și este o expresie booleană. Introducerea condiției watchdog vă permite să specificați în mod explicit semantica declanșării acesteia.

O expresie de acțiune este executată dacă și numai dacă se declanșează tranziția. Este o operație simplă care se efectuează imediat după declanșarea tranziției corespunzătoare înainte de începerea oricăror acțiuni în starea țintă.

Diagrama de activitate

O diagramă de activitate este o diagramă, al cărei scop principal este simularea procesului de efectuare a operațiunilor. Diferența față de diagrama de stare constă în semantica statelor, care sunt folosite pentru a reprezenta nu activitățile, ci acțiunile și absența semnăturilor evenimentelor la tranziții. Fiecare stare din schema de activitate corespunde performanței unei anumite operații elementare, iar trecerea la starea următoare funcționează numai atunci când această operație este finalizată în starea anterioară.

Starea acțiunii este un caz special de stare. Starea acțiunii nu poate avea tranziții interne, deoarece este elementară. Utilizarea obișnuită a unei stări de acțiune este aceea de a simula o singură etapă de executare a unui algoritm (procedură) sau a unui flux de control.

O tranziție simplă este o relație între două state succesive, ceea ce indică faptul că o stare se schimbă în alta.

1.2.2 **Diagrame de structură**

UML are, de asemenea, diagrame pentru vizualizarea, specificarea, proiectarea și documentarea aspectelor statice ale unui sistem care formează coloana vertebrală relativ puternică. Așa cum aspectele statice ale unei case arată modul în care vor fi amplasate într-o clădire (pereți, uși, ferestre, țevi, cabluri electrice, orificii de aer etc.), aspectele statice ale sistemelor software reflectă prezența și aranjarea claselor, cooperative, componente, noduri și alte entități.

Numele diagramelor structurale ale UML corespund denumirilor grupurilor principale de entități utilizate în modelarea sistemului:

- diagrame de clase - clase, interfețe și cooperări;
- diagrame de componente - componente;
- diagrame de amplasare - noduri.

Diagrama de clasă

Diagrama de clasă (diagramă de clasă) este utilizată pentru a reprezenta structura statică a modelului de sistem în terminologia claselor de programare orientată obiect. Diagrama de clasă poate reflecta, în special, diferitele interrelații între entități separate ale domeniului, cum ar fi obiectele și subsistemele, și descrie, de asemenea, structura lor internă și tipurile de relații. Această diagramă nu indică informații privind aspectele temporale ale funcționării sistemului. Din acest punct de vedere, diagrama de clasă reprezintă o dezvoltare ulterioară a modelului conceptual al sistemului proiectat.

Clase

O clasă de obiecte reprezintă un grup de obiecte care au:

- proprietăţi similare (atribute);
- un comportament comun (operaţii);
- relaţii comune cu alte obiecte şi o aceeaşi semantică.

Regulile de vizibilitate se aplica atat atributelor cat si operațiilor din clase si se refera la domeniul de acces permis la un membru al unei clase. Fiecare nivel de vizibilitate este reprezentat printr-un simbol:

- private (–) : accesibiltate numai din interiorul clasei;
- public (+) : accesibiltate la nivelul intregului sistem;
- protected (#) : accesibiltate in arborele de mostenire;
- package (~) : accesibiltate din interiorul pachetului care contine clasa.

Dependenţa:

Apare când o clasă foloseşte pentru scurt timp o altă clasă. Apare între două elemente dintre care unul este unul independent şi altul dependent. Orice modificare a elementului independent va fi reflectată şi asupra elementului dependent. Putem avea o relaţie de dependenţă de exemplu în cazul în care o clasă foloseşte ca parametru un obiect al altei clase, sau o clasă accesează un obiect global al altei clase, sau o operaţie a unei clase este apelată într-o altă clasă.

Asocierea:

Legătură (conexiune) între două clase (relaţie şi între obiecte, instanţe ale celor două clase) ce permite claselor respective să comunice între ele. Pot exista asocieri unidirectionale sau bi-directionale (indică dacă fiecare clasă transmite mesaje celeilalte sau doar una poate transmite mesaje).

Agregarea:

Este o formă specială de asociere a claselor, utilizată în cazul în care relaţia dintre cele două clase este de tipul parte din întreg.

Compoziția:

Compunerea este un concept similar cu agregarea, însă mai puternic deoarece implică faptul că întregul nu poate exista fără părţi.

Generalizarea:

Relaţie între o clasă şi subclasele sale, este prezentă ori de câte ori se semnalează de-a lungul unei ierarhii proprietăţi comune sau operaţii ce evidenţiază comportament comun.

Folosirea diagramelor de clase:

- în modelarea conceptuală (analiza oriectată pe obiect);
- pentru specificarea software;
- în proiectarea de detaliu și implementare.

Diagrame de componente

Schema componentei, spre deosebire de diagramele considerate anterior, descrie caracteristicile reprezentării fizice a sistemului. Diagrama componente vă permite să definiți arhitectura sistemului dezvoltat prin stabilirea dependențelor între componentele software, în rolul sursei, codului binar și al codului executabil care poate acționa. În multe medii de dezvoltare, un modul sau o componentă corespunde unui fișier.

Componenta implementează un anumit set de interfețe și servește pentru o desemnare generală a elementelor reprezentării fizice a modelului. Deoarece componenta ca element al implementării fizice a modelului reprezintă un modul de cod separat, uneori este comentat cu indicarea unor simboluri grafice suplimentare care ilustrează caracteristicile specifice ale implementării acestuia.

Diagrama de componente se elaborează pentru urmatoarele scopuri:

- vizualizarea structurii comune a codului sursă a unui sistem de program;
- specificarea variantei executabile a unui sistem de program;
- asigurarea utilizării repetate a unor fragmente ale codului sursă;
- reprezentarea conceptuală şi fizică a schemelor bazei de date.

Componenta realizează un set de interfeţe şi desemnează elementele reprezentării fizice a unui model.

Interfaţa este reprezentată în formă de circumferinţă, care este legat cu componentul cu ajutorul liniei fără săgeată.

Diagrama de amplasare

Diagrama de amplasare (diagramă de desfășurare) este destinată vizualizării elementelor de program și a componentelor care există doar în stadiul de execuție (execuție). În acest caz, sunt reprezentate numai componentele programului, care sunt fișiere executabile sau biblioteci dinamice. Acele componente care nu sunt utilizate la rulare nu sunt prezentate în diagrama de implementare.

Diagrama de implementare conține reprezentări grafice ale procesoarelor, dispozitivelor, proceselor și conexiunilor dintre ele. Spre deosebire de diagramele de prezentare logică, diagrama de implementare este uniformă pentru sistem ca întreg, deoarece trebuie să reflecte pe deplin caracteristicile implementării sale. Această diagramă, de fapt, completează procesul de analiză și proiectare orientată pe obiecte pentru un sistem informatic specific, iar dezvoltarea acestuia este, de regulă, ultimul pas în specificarea modelului.

Nodul (node) reprezintă un anumit element fizic a sitemului, care are o anumită resursă de calculare. Ca resursă de calculare a nodului poate fi o valoarea electronică a memoriei sau procesorului.

Conectările sunt un fel de asociere şi sunt prezentate în formă de linii fără săgeţi.

2 Diagramele UML

În acest compartiment sunt analizate și modelate toate diagramele descrise mai sus. Ele sunt descrise la teme aleasă, „Lanasatorul jocului Liga Legendelor”.

2.1 Diagramele cazurilor de utilizare

Prima diagramă reprezintă funcționalul de bază oferită de lansator (figura 2.1.1). Diagrama “**Descrierea Generală**”. Primul funcțional oferit de aplicație este autentificare în aplicație, apoi ne oferă o metodă de a începe o partidă de joc, diferite metode de vizualizări (informare, statistică), deasemenea realizarea cumpărăturilor în magazin și comunicarea cu alți utilizatori.

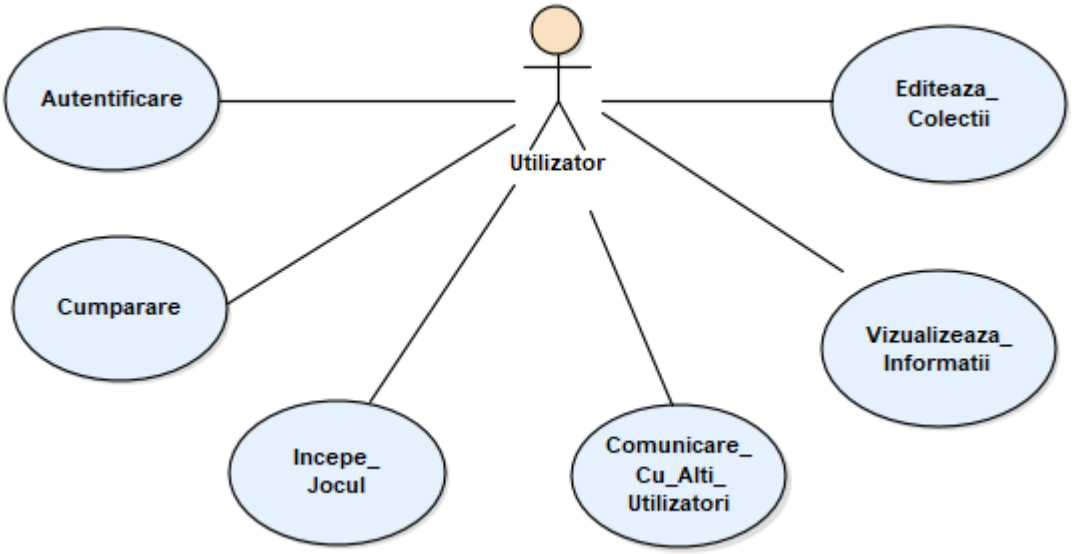

Figura 2.1.1 Descrierea generală

Pentru a realiza autentificarea, pentru prima dată va trebui să creezi un acoount, apoi cu acest acount ne vom putea loga în aplicație. Iar în caz că nu ne amintim „numele” sau „parola”, ele aplica’ia ne permite restabilirea lor. Acestă multitudine de operații sunt reprezentate în diagrama „**Operațiile de creare acount, autentificare și restabilire acount**”, arătate în (figura 2.1.2).

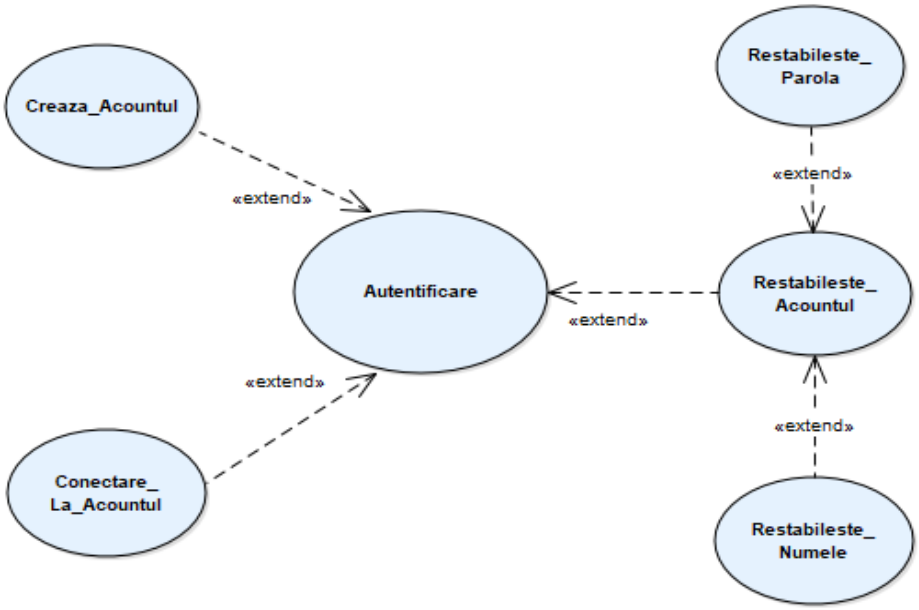

Figura 2.1.2 Operațiile de creare acount, autentificare și restabilire acount

Pentru a începe un o pardidă de joc va trebui să îndeplinim câteva operați. Și amume selectarea tipului de joc, harta pe care vrei să te joci și modul, campionul cu care vrei sa te joci, poziția pe hartă și desigur oferă și posibilitatea de a renunța pregătirea. Funcționalul este prezentat în diagrama „**Operațiile de începere a jocului**”, care este prezentată în figura 2.1.3.

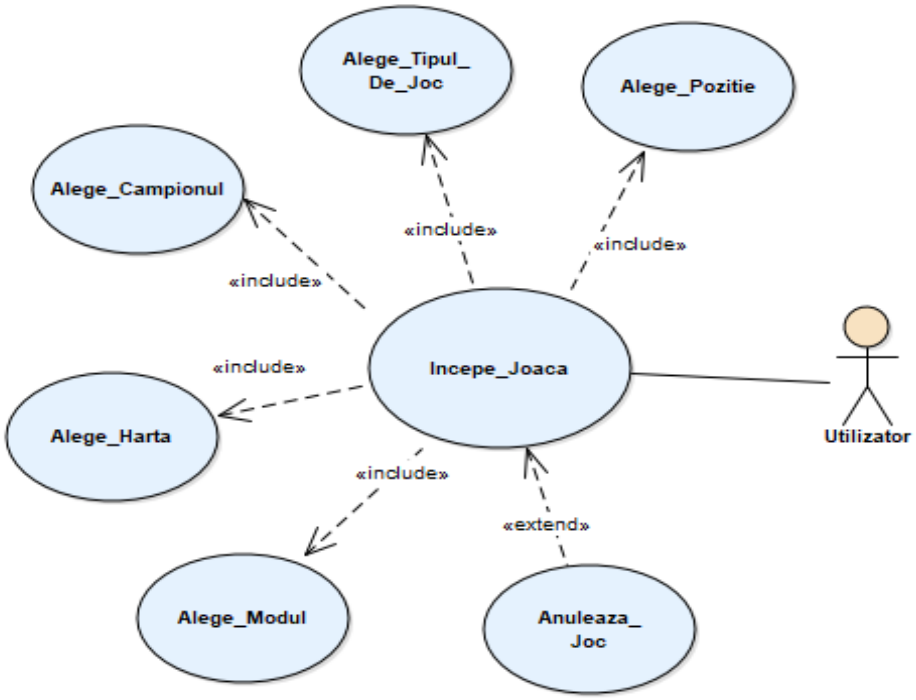

Figura 2.1.3 Operațiile de începere a jocului

Jocul pentru care a fost creat acest lansator poate fi jucat și singur, dar e plictisitor. De aceea acest joc se joacă cu prietenii sau alți jucători (utilizatori). Pentru aceasta aplicația ne ofera un compartiment aparte de socializare. Unde putem adăuga sau elimina prieteni în lista personală de prieteni. Ai invita, de a juca înpreună. Și de asemenea de a comunica cu acesti prieteni. Funcționalul este reprezentat în diagrama „**Funcționalul de Socializare**”, aratat în figura 2.1.4.

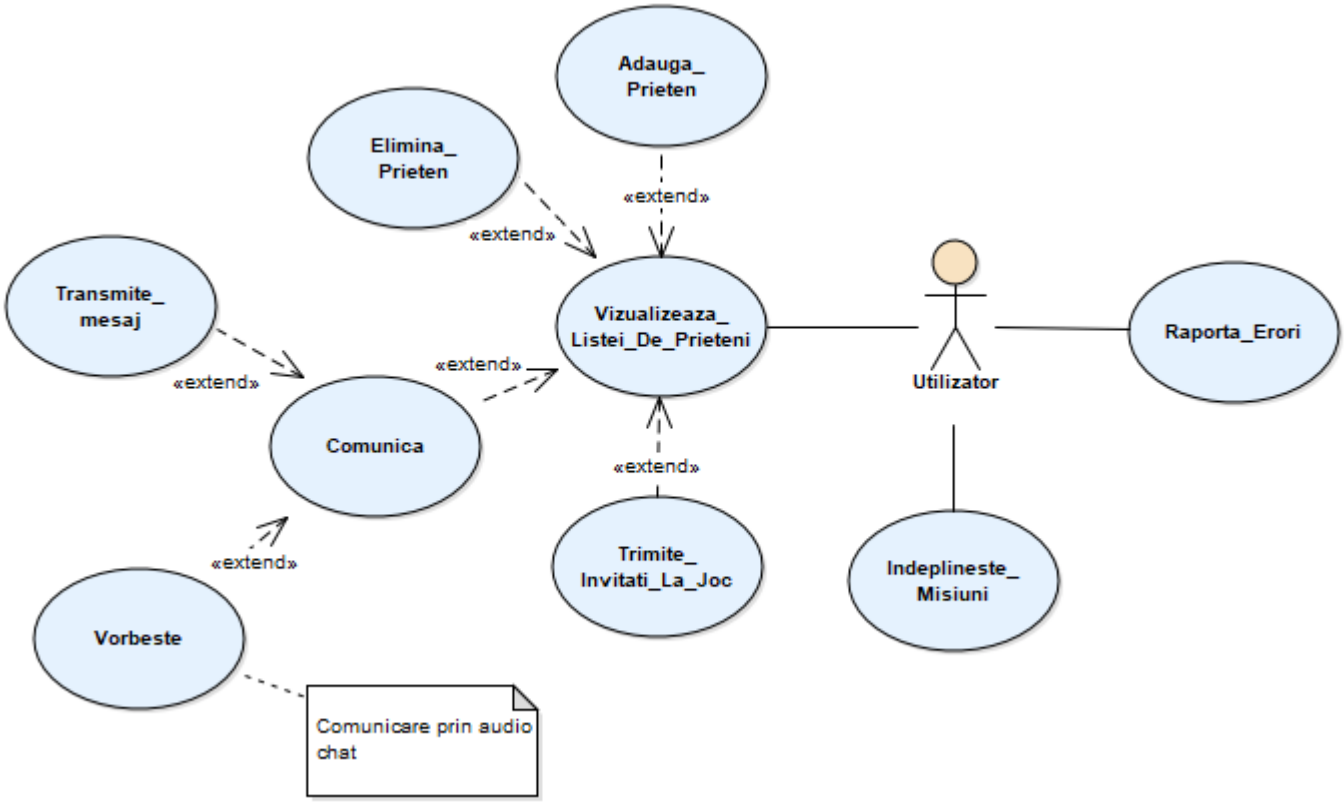

Figura 2.1.4 Funcționalul de Socializare

2.2 Diagramele de secvență

În următoarea diagramă de segvență este arătat ativitatea utilizatorului și a aplicației în timpul logării. Sunt reprezentați toți pașii necesari pentru o logare în aplicație. Începem cu pornirea la aplicație, introducerea numelui și parolei, transmiterea datelor pentru verificare în baza de date la server, validarea ulterioară a datelor, și accesarea acountului în aplicație. Toți pași sunt prezentați în diagrama „**Pașii de creare acount, autentificare și restabilire acount**”, din figura 2.2.1.

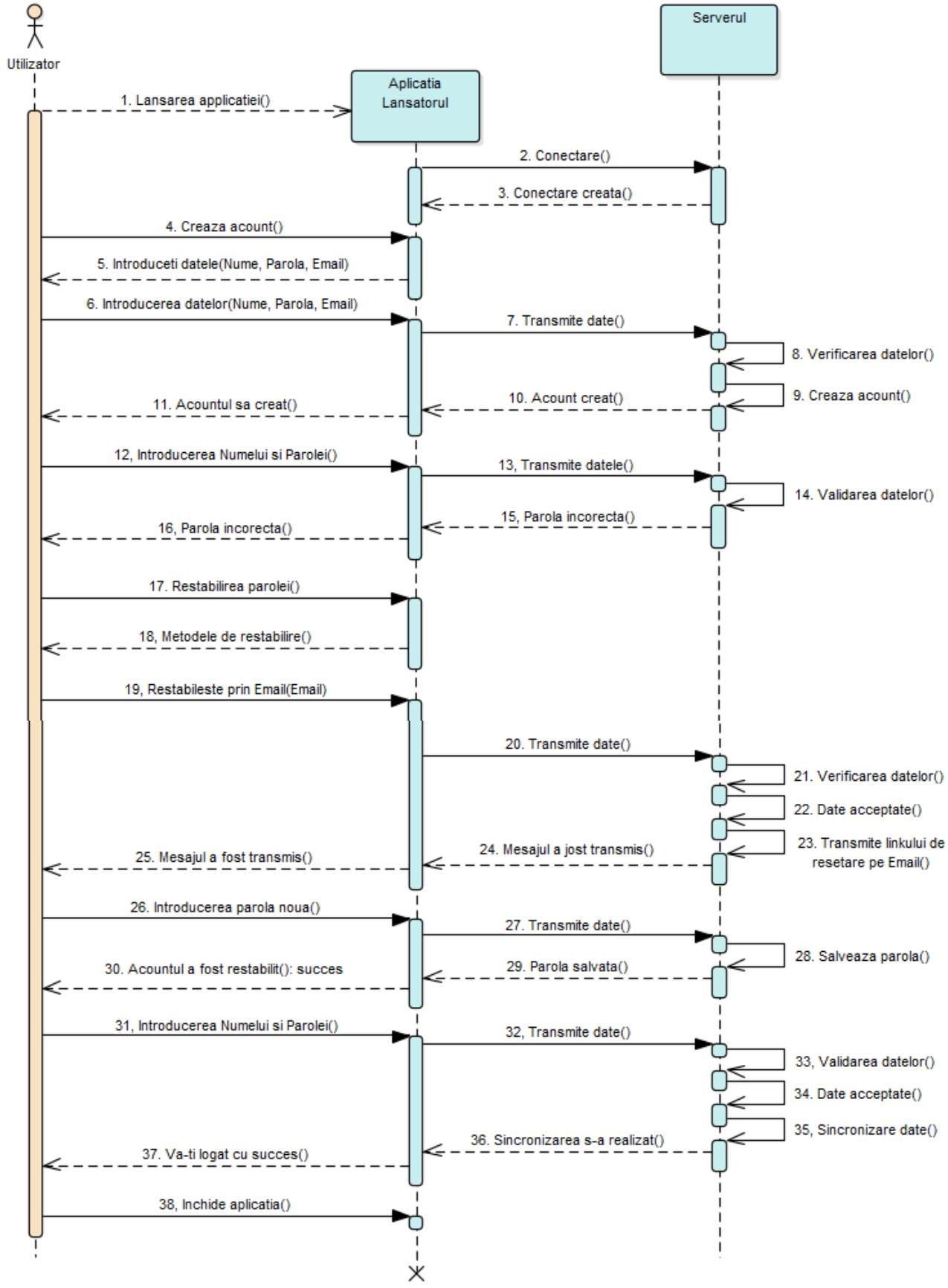

Figura 2.2.1 Pașii de creare acount, autentificare și restabilire acount

În următoarea diagramă de segvență este arătat întreaga etapă de configurare (creare) a unui joc, și anume pașii de unde utilizatorul trbuie să alege tipul, harta, modul și poziția, apoi alegerea campionul și controlul campionului în joc, și cum ne întoarcem înapoi la lansator (după ce terminăm jocul), pentru a vizualiza informațiile despre jocul jucat. Toți pași sunt prezentați în diagrama „**Pașii de începere a unui joc**”, din figura 2.2.2.a și figura 2.2.2.b.

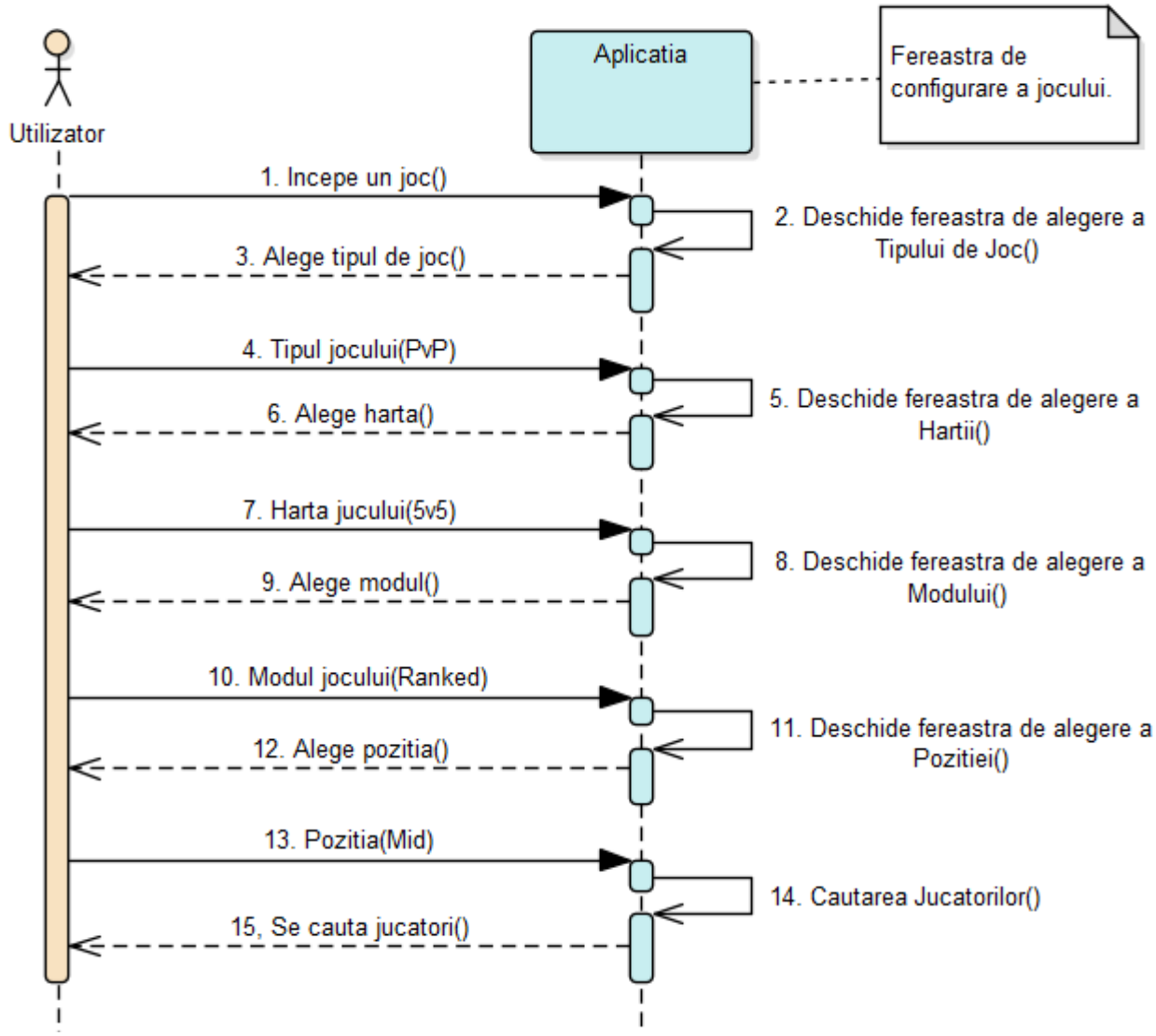

Figura 2.2.2.a Pașii de începere a unui joc

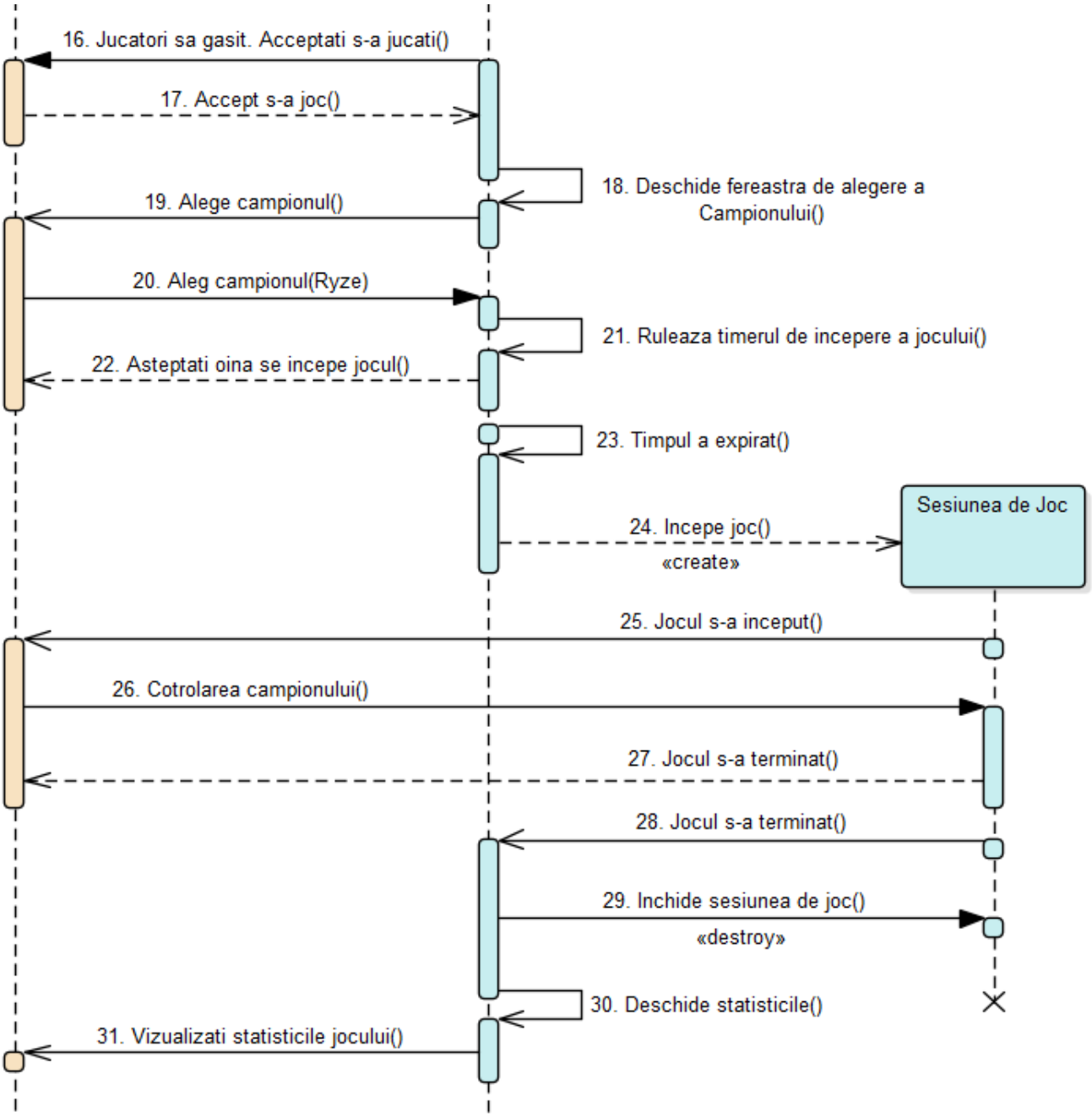

Figura 2.2.2.b Pașii de începere a unui joc

2.3 Diagramele de colaborare

În diagrama „ **Începerea unui joc**”, reprezentată în figura 2.3.1. Această diagramă este de nivel de specificare și ne arată interacțiunea dintre utilizator și o sesiune de joc, anume la etapa de începere.

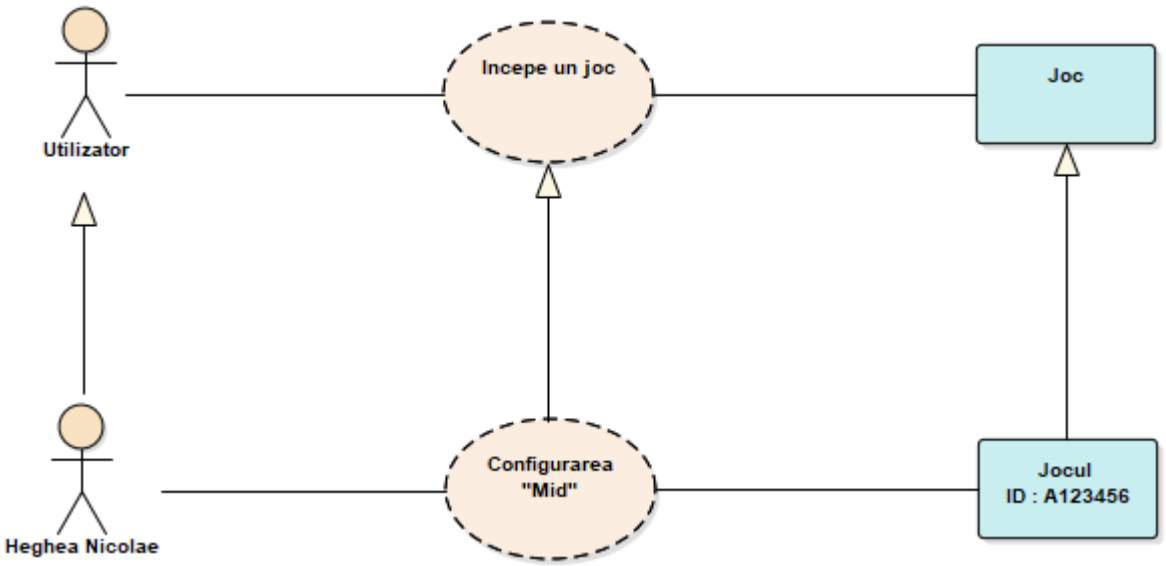

Figura 2.3.1 Începerea unui joc

În diagrama „**Alimentarea contului**”, reprezentată în figura 2.3.2. Aceasta diagramă este de nivel de specificare. Și ne arată interacțiunea dintre un utilizator și server la alimentarea unui cont.

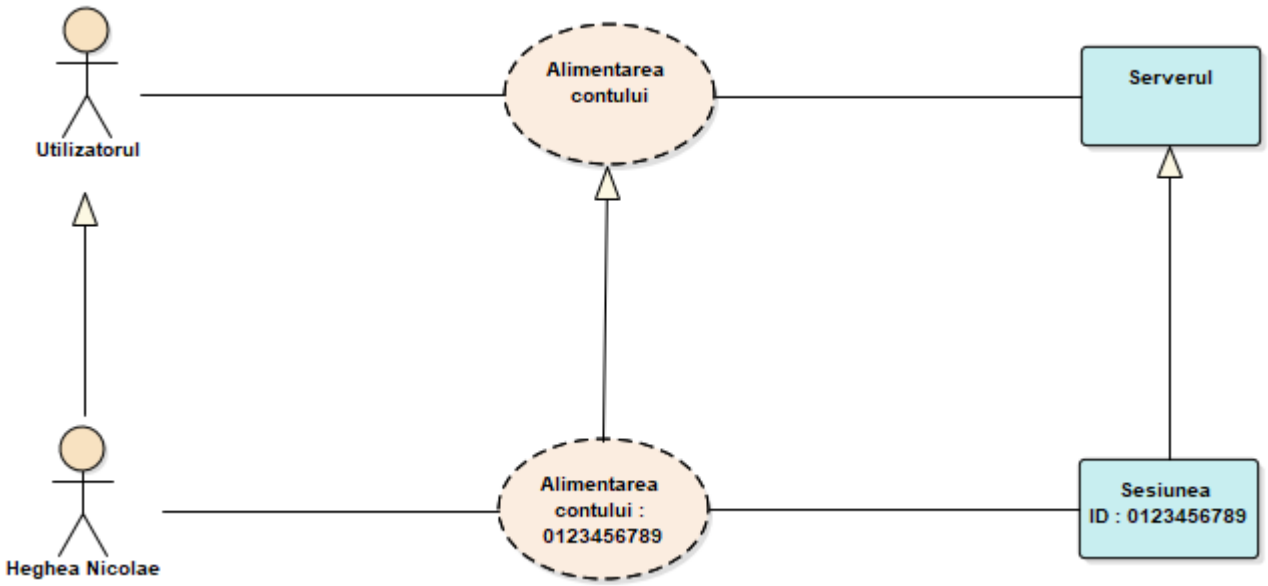

Figura 2.3.2 Alimentarea contului

În diagrama „**Modificarea și afișarea unui inventariu nou**”, reprezentată în figura 2.3.3. Aceasta diagramă este de nivel de exemple. Ne arată interacțiunea dintre utilizator, coleții, server și fereastra de editare a unui nou inventariu. Un utilizator poate crea, edita, sterge, vizualiza un inventariu.

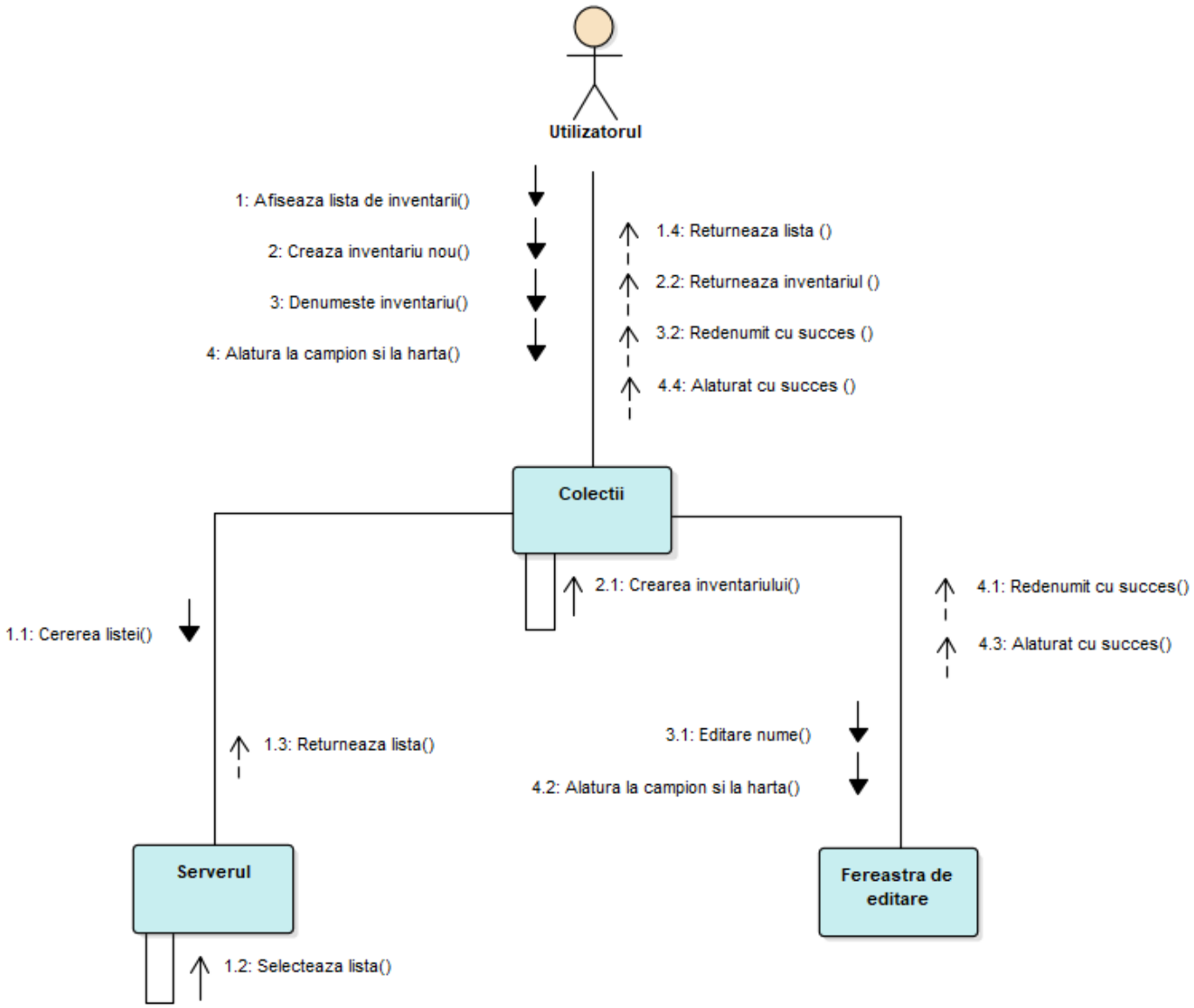

Figura 2.3.3 Modificarea și afișarea unui inventariu nou

2.4 Diagramele de clase

În figura 2.4.1 este reprezentată diagrama de clase „**Interfața principală**”. Această diagramă ne arată modelarea interacțiunilor între aplicație și componentele care se află pe ea. Aici este arătată interfața principală a aplicației, și anume că fiecare aplicație conține o fereastreă, un meniu, o secțiune de socializare, și la aplicație este conectat un singur profil, și deasemenea această aplicație este conectată la server.

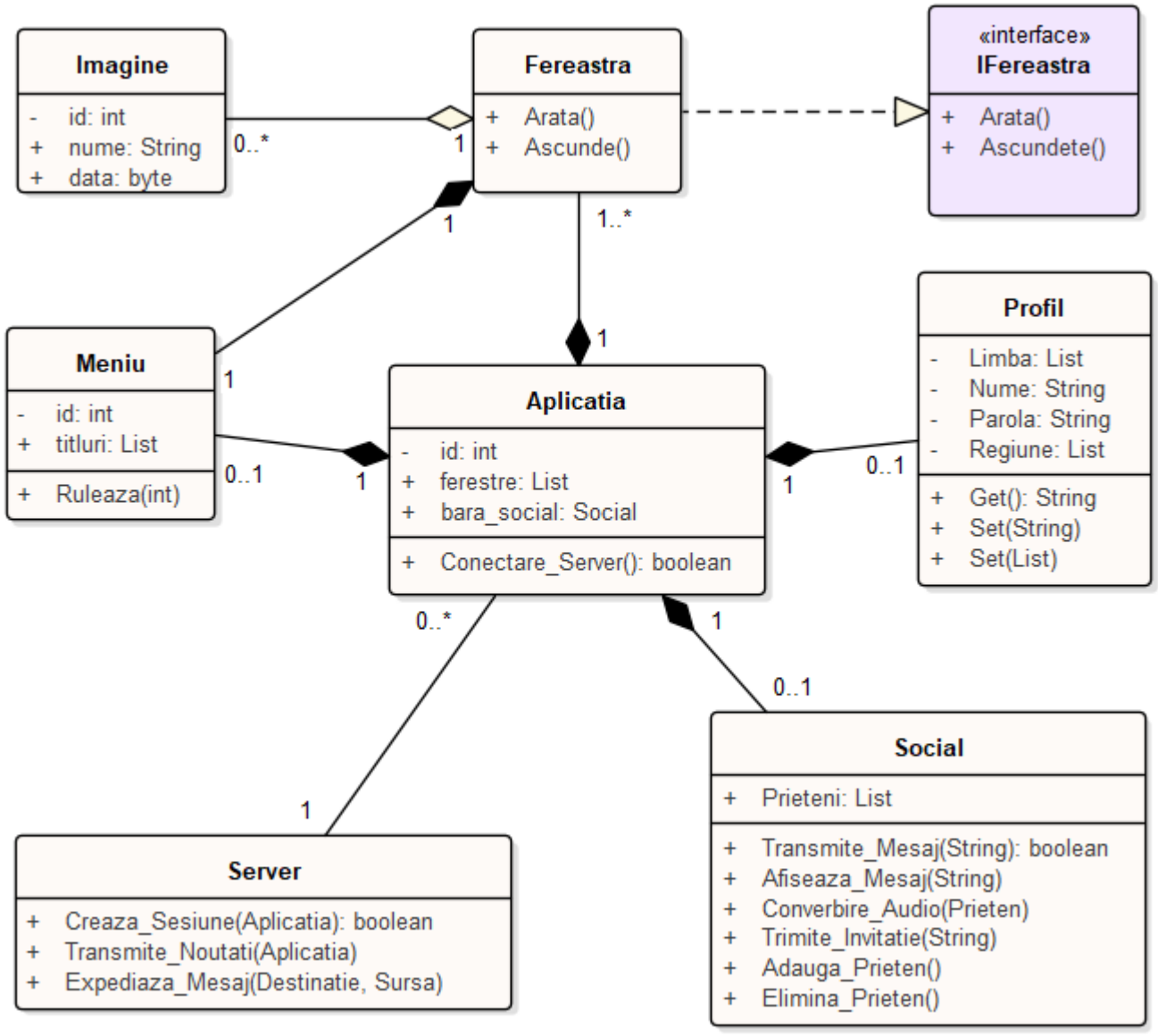

Figura 2.4.1 Interfața principală

În figura 2.4.2 este reprezentată diagrama de clase „**Autentificarea în aplicație**”. Această diagramă ne arată modelul ferestrei de autentificare și autentificare în aplicație. În diagramă este arătată fereastra de logare prin intermediul căreia se realizează logarea, restabilirea unui profil sau crearea unui profil nou. Și deasemenea această fereastra se conectează la server.

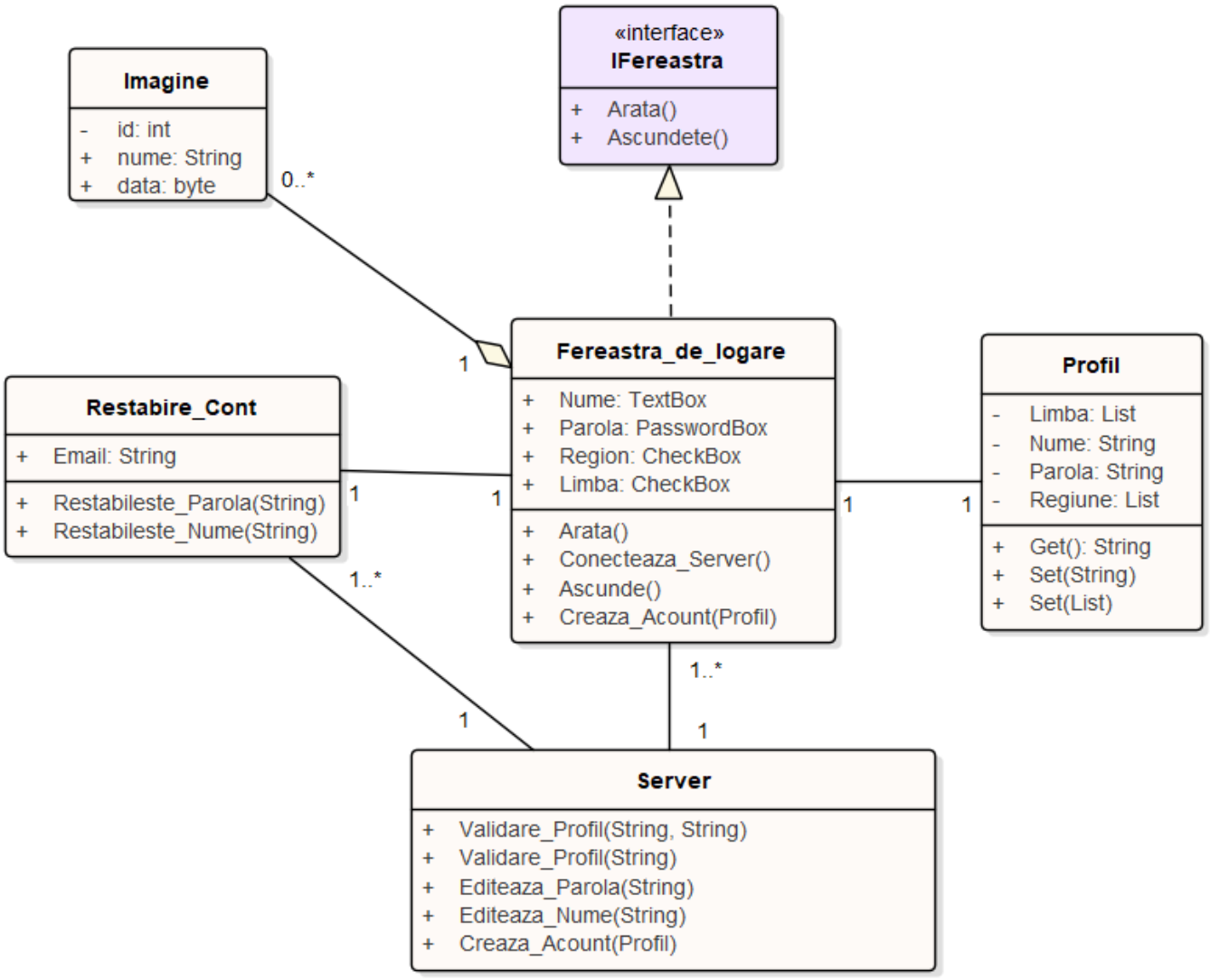

Figura 2.4.2 Autentificarea în aplicație

În figura 2.4.3 este reprezentată diagrama de clase „ **Modelul magazinului**”. În acest model este arătată fereastra magazinului care este alcătuită dintrun meniu, casete de oferte, de campioni și de avatare. De asemenea conține și posibilitatea alimentării contului, și realizarea diferitor cumpărături.

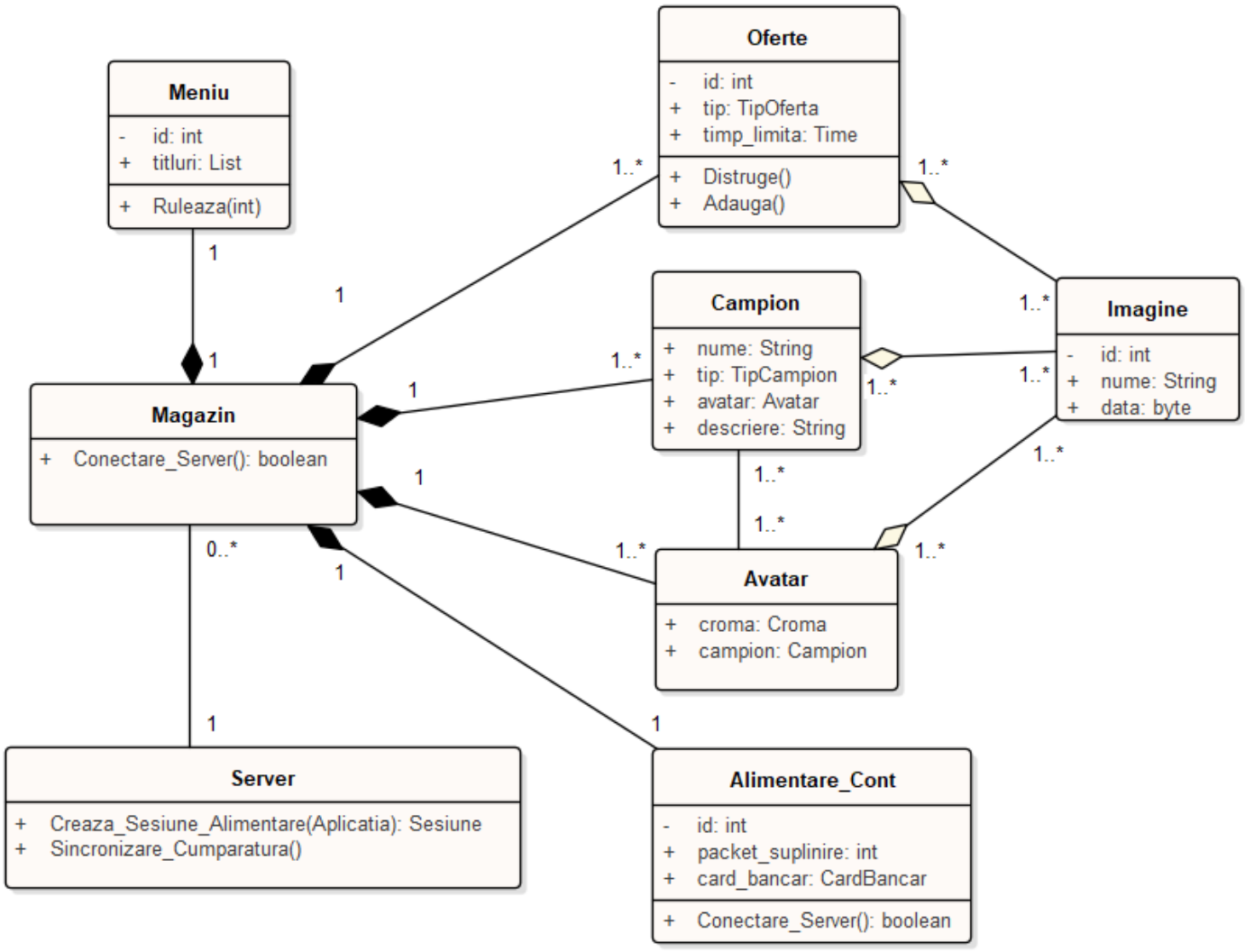

Figura 2.4.3 Modelul magazinului

2.5 Diagramele de stari și activitați

În figura 2.5.1 este reprezentată diagrama de stări „**Începerea unui joc**”. În acest model este arătat stările procesului de începere a unui joc. Procesul începe cu apăsarea butonului ”joaca”, apoi configurăm setările dorite. După accepare, se vor cauta jucari. Și după găsire toți își vor alege campionul, și vor începe jocul. După finisare vor vizualiza statisticile jocului, și dacă vor dori. Vor mai juca încă un joc, iar dacă nu, se vor întoarce în fereastra principală.

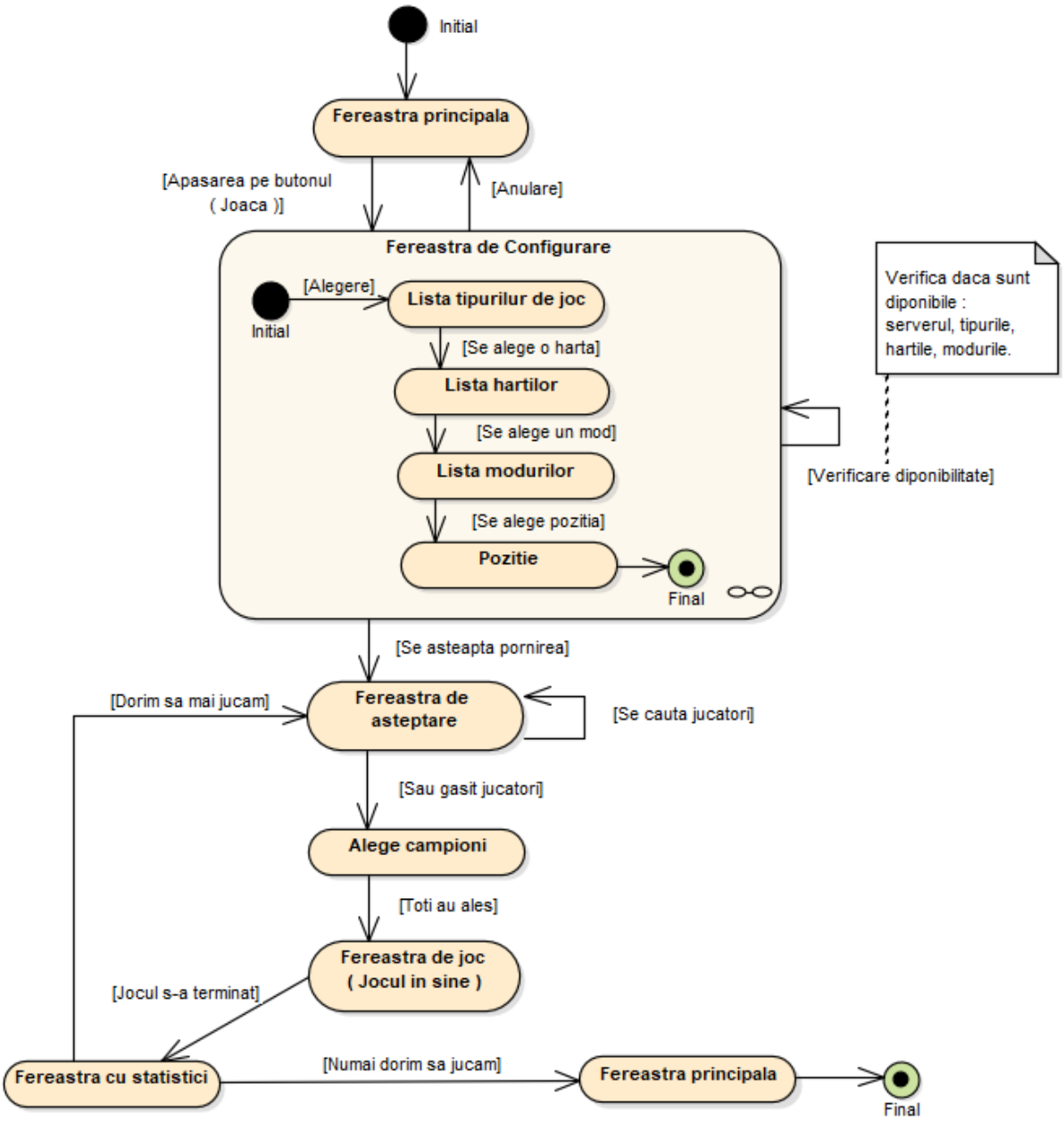

Figura 2.5.1 Începerea unui joc

În figura 2.5.2 este reprezentată diagrama „ **Autentificarea în aplicație**”. Această diagramă ne arată stările procesului de autentificare. În diagramă este arătată fereastra de pornire, prin care se fac updaturile, care necesită conectare la server. Apoi fereastra de logare prin intermediul căreia se realizează logarea sau restabilirea unui profil. Și deasemenea această fereastra se conectează la server. Diagrama se reprizintă prin stări disjuncte.

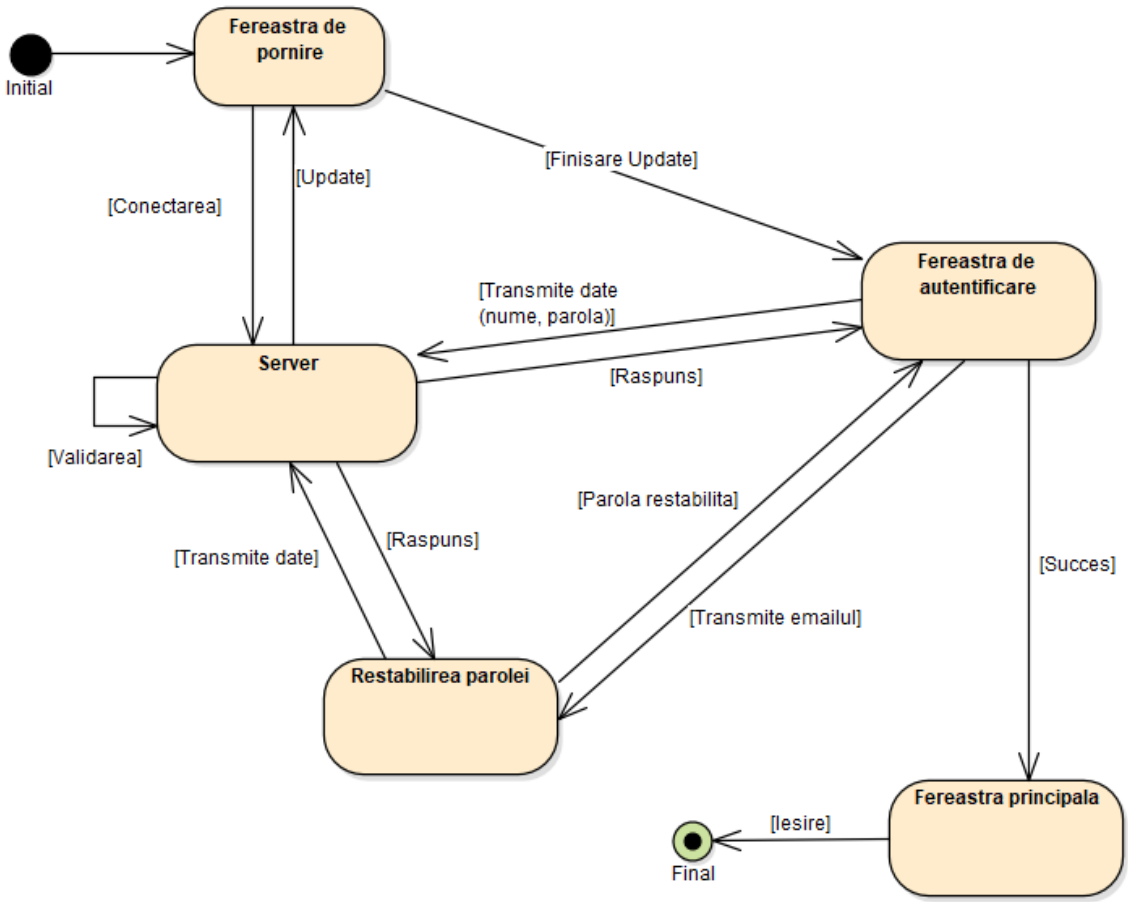

Figura 2.5.2 Autentificarea în aplicație

În figura 2.5.3 este reprezentată diagrama „**Crearea unui nou inventariu**”. În această diagramă este arată stările procesului decreare a unui nou inventariu. Acțiunile de editare se realizază paralel una față de alta.

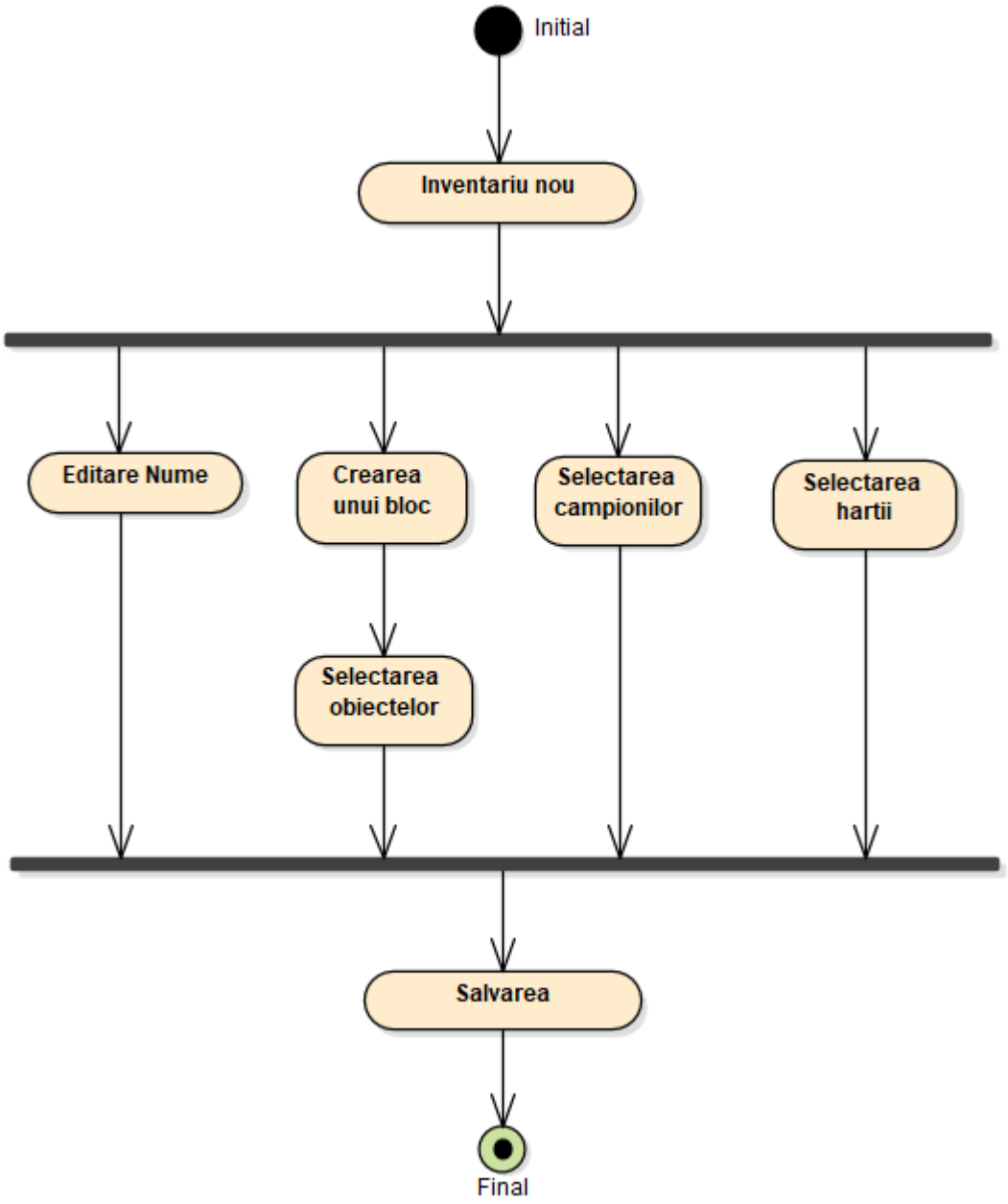

Figura 2.5.3 Crearea unui nou inventariu

2.6 Diagramele componentelor

În figura 2.6.1 este reprezentată diagrama „**Interfața principală**”. Această este diagrama de componente și ne arată modelarea interacțiunilor intre componentele aplicației. Aplicația este dependentă de ferestrele care o reprezintă. Fiecare dintre aceste ferestre au rolul și acțiunile sale. Aplicația are de asemenea acțiunile sale, de exemplu : crearea unui sesiuni de joc, cumpararea campionilor sau suplinirea contului și conectarea la server.

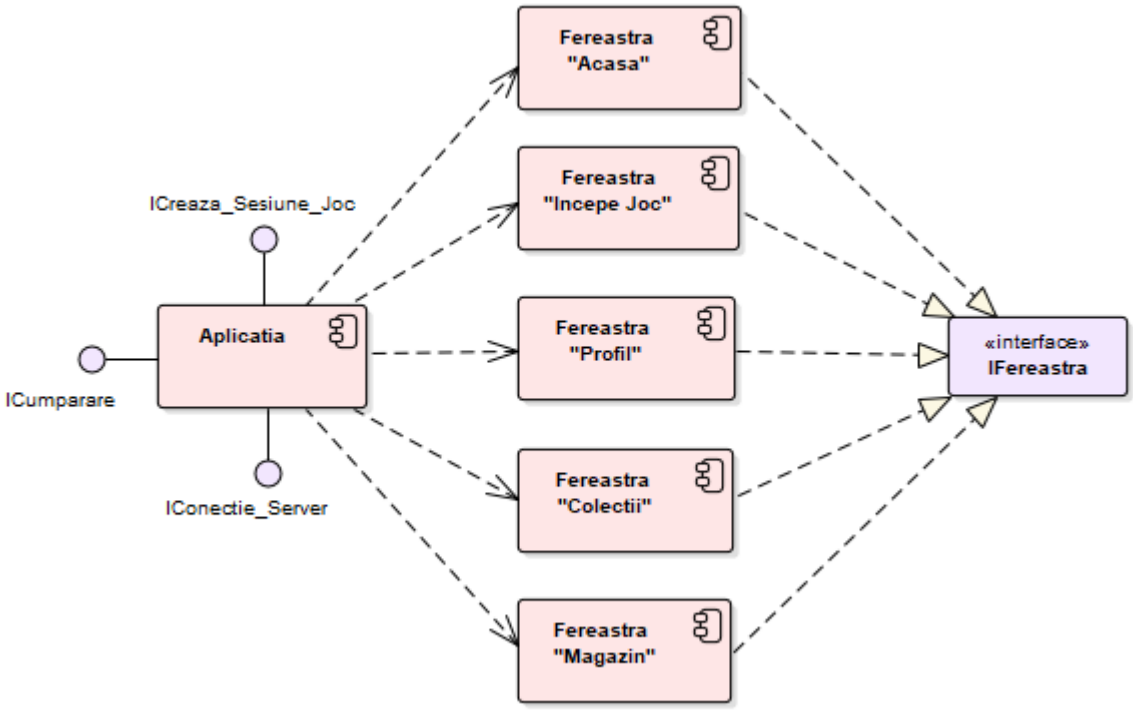

Figura 2.6.1 Interfața principală

În figura 2.6.2 este reprezentată diagrama „**Modulul de începere a unui joc**”. Această este diagrama de componente și ne arată modelarea interacțiunilor intre componentele modulului de începere a unui joc.

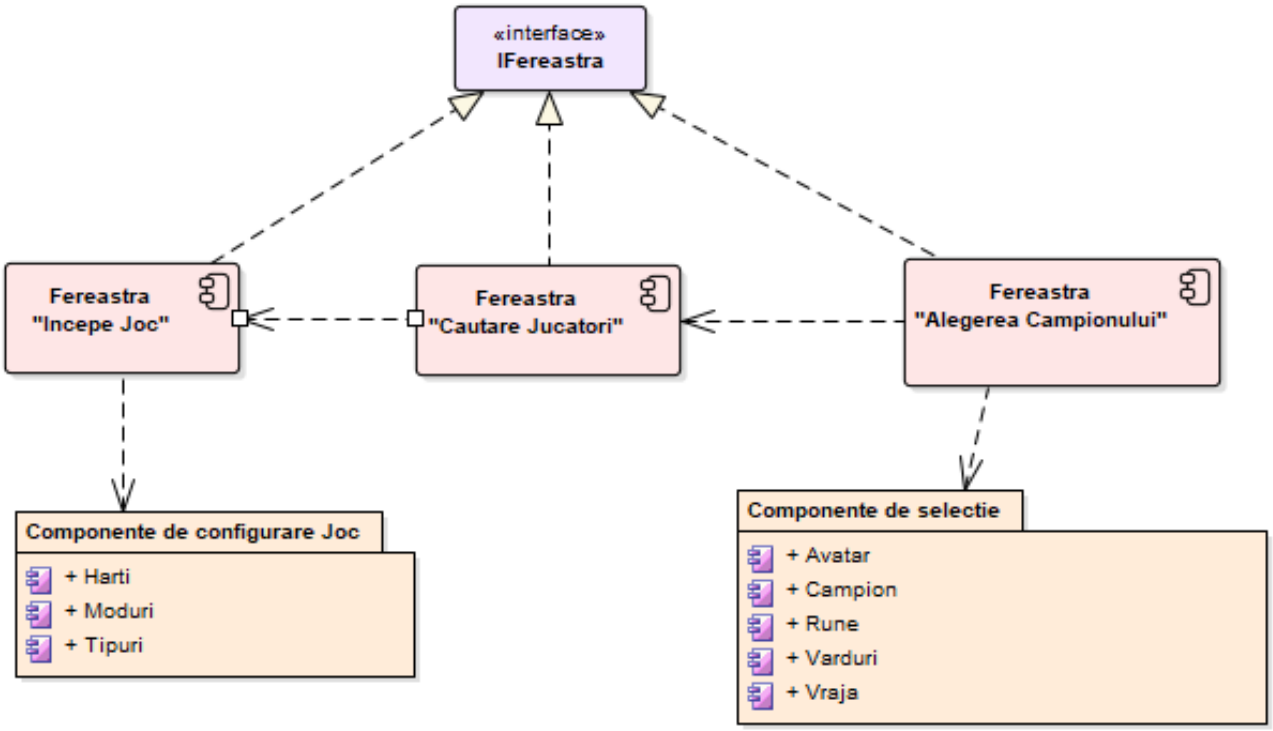

Figura 2.6.2 Fereastra magazinului

2.7 Diagramele de amplasare

În figura 2.7.1 este reprezentată diagrama „**Componentele de baza ale sistemului**”. Această este diagrama de plasere și ne arată modelarea interacțiunilor intre modulele sistemului, și anume : aplicație, joc, actualizator și server.

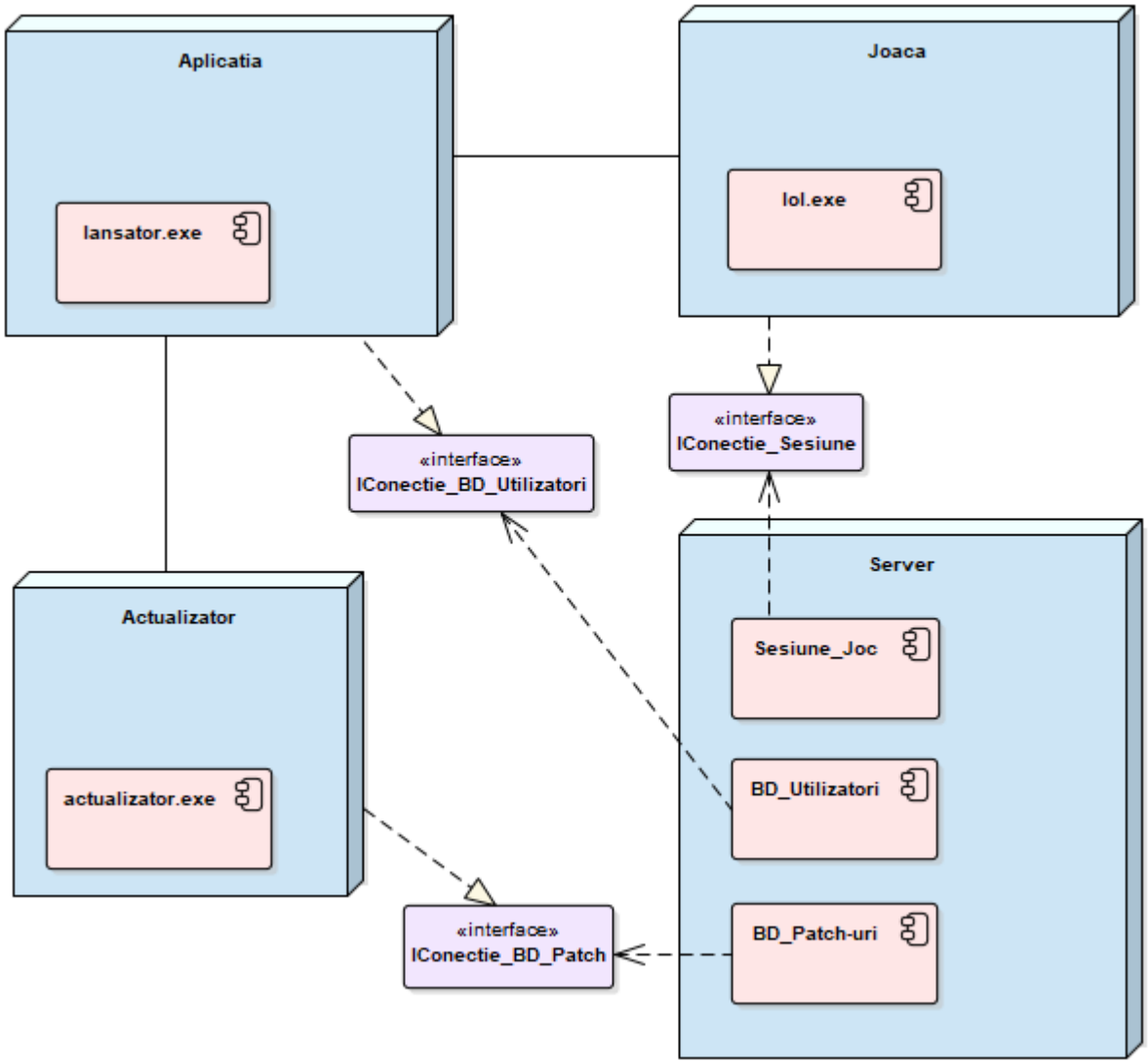

Figura 2.7.1 Componentele de baza ale sistemului

Concluzie

În ultimii ani modelarea sistemelor informatice sa dezvoltat foarte tare ,dar și a devenit o activitate foarte importantă în ingineria soft-urilor. Modelarea unui sistem informațional este realizată prin folosirea limbajului UML în mod rational de proiectare a unui sistem.

Toate relaţiile prezentate in aceasta lucrare sunt aduse la forma normală. În cadrul lucrării a fost elaborat un model de program pentru clientul (lansatorul) jocului „Liga Legendelor”.

Mai mult, s-au evidențiat diferite procese care au apărut în timpul funcționării aplicației. Acestea au fost mapate folosind o mașină de stare finită în diagramele de stare. Procesele algoritmice au fost reprezentate pe diagrame de activitate. În acest stadiu, comportamentul sistemului a fost determinat, descrierea structurii sale logice a fost finalizată. Ultima etapă este modelarea din punct de vedere fizic. A fost făcută o diagramă a componentei care arată dependența componentelor și o diagramă de amplasare. Pentru modelare, sa folosit limbajul UML - un limbaj descriptiv pentru modelarea obiectului în dezvoltarea de software, modelarea proceselor de afaceri, proiectarea sistemului și afișarea structurilor organizaționale.

În decursul creării acestei lucrări, sa dovedit că limbajul UML este un limbaj de vizualizare și design, deosebit de util, dar în același tim simplu și accesibil, care permite îndeplinirea cu succes a diferitor sarcini de proiectare. UML oferă un singur standard pentru toți, asigurându-se că, respectând standardul, chiar și specialiștii din diferite domenii pot înțelege structura și comportamentul sistemului proiectat. Universalitatea lui oferă un plus de încredere din momentul în care diagramele vor putea fi citite și înțelese eficient de orice persoană care cunoaște UML-ul.

În final pot afirma cu încredere că acest proiect a fost educativ, ajutându-mă să cunosc mai bine structura și componentele UML, dar cel mai util să le implimentez în practică.

Bibliografie

1. Site official https://www.uml.org
2. Enciclopedia Wikipedia [Resursa electronică]: www.wikipedia.com
3. Lucrările de laborator AMOO
4. Curs teoretic AMOO, partea 1: autor Melnic Radu
5. Helper-ul [Enterprise Architect]