



ACADEMIA DE STUDII ECONOMICE BUCUREȘTI
FACULTATEA DE CIBERNETICĂ, STATISTICĂ ȘI INFORMATICĂ
ECONOMICĂ

LUCRARE DE LICENȚĂ

SISTEM INFORMATIC PENTRU ÎNSCRIEREA

CANDIDAȚILOR ÎN CADRUL UNEI UNIVERSITĂȚI

Coordonator științific:

Prof. Univ. Dr. PAUL POCATILU

Absolvent:

BIȘAG ALEXANDRU-ȘTEFAN

BUCUREȘTI

2019

Cuprins

Introducere.....	3
Capitolul 1 - Etapa de analiză	6
1.1 Analiza contextului.....	6
1.2 Analiza sistemului informatic	7
1.2.1 Identificarea nevoilor și a funcționalităților	7
1.2.2 Diagramele generale ale cazurilor de utilizare	8
1.2.3 Diagramele de activitate	10
1.2.4 Diagramele de stare.....	13
1.2.5 Diagramele de secvență.....	14
Capitolul 2 – Etapa de proiectare.....	17
2.1 Diagrama de clase detaliată.....	17
2.2 Proiectarea bazei de date.....	18
2.3 Proiectarea interfețelor grafice	19
2.4 Diagrama de componente	22
2.5 Diagrama de desfășurare.....	23
Capitolul 3 – Etapa de implementare.....	24
3.1 Prezentarea tehnologiilor utilizate	24
3.2 Arhitectură & Scalare	28
3.3 Prezentarea modulelor	29
3.4 Publicare	31
Capitolul 4 – Prezentarea aplicației	35
4.1 Scenariul Student.....	35
4.2 Scenariul Voluntar	38
4.3 Scenariul Casier	39
4.4 Scenariul Operator.....	40
4.5 Scenariul Administrator	41
Concluzii.....	44
Bibliografie	46

Introducere

Despre mediul universitar știm, încă din copilărie, că reprezintă cea mai înaltă formă de învățământ la care o persoană poate să aspire. Indiferent de domeniul de activitate, cei mai renumiți teoreticieni și cercetători au avut, la bază, o pregătire temeinică sinonimă, de cele mai multe ori, cu studierea ariilor de interes în cadrul unei universități la care, ulterior, au revenit în calitate de profesori. În acest sens, mediul academic a reprezentat, încă de la apariția primelor astfel de instituții, la începutul Evului Mediu, un centru al culturii, al educației și al inovării, statut care a reușit să se propage până în prezent. Fie că vorbim de studii care au revoluționat lumea sau de descoperiri care au schimbat fundamental modul de gândire al generațiilor succesoare, aceste rezultate au fost, de-a lungul timpului, strâns legate de această formă de educație ale cărei origini datează încă din Grecia Antică.

Totuși, revoluțiile tehnologice care au schimbat radical modul de viață al strămoșilor noștri și au îmbunătățit într-un mod imposibil de prevăzut calitatea vieții au fost, până recent, temperate de bariera materială: oamenii au început să producă mai mult și mai eficient, au început să deprindă din ce în ce mai multe abilități elevate și să acceadă la o condiție mai bună, însă munca era, încă, percepută ca o activitate fizică, iar mediul academic a rămas, multă vreme, în ipostaza de instrument auxiliar, separat de lumea muncitorească.

Acest lucru a început să se schimbe odată cu nașterea celei mai recente revoluții tehnologice, anume revoluția digitală, care a început să șteargă cu pași rapizi distanța între muncă și efortul psihic, oamenii începând să aibă nevoie de cunoștințe din ce în ce mai avansate pentru a putea munci și, poate mai important, pentru a se integra în societate. Astfel, dintr-o instituție destinată în special acelor oameni interesați de cercetare sau care doreau să profeseze în domenii mai avansate, cum ar fi medicina sau unele ramuri ale ingineriei, universitatea, ca instituție, a devenit un pilon al societății într-un mod mult mai pragmatic, în atribuția ei intrând, aproape instantaneu, obligația efectivă de a pregăti forța de muncă la un nivel cantitativ mult mai ridicat decât înainte. Acest eveniment a avut un impact major asupra academiilor, ce au trebuit să își adapteze planurile de școlarizare pentru a face față atât cerințelor, cât și numărului din ce în ce mai mare de studenți, precum și asupra modului în care studiile superioare au început să fie văzute în cadrul societății, acestea devenind, dintr-o calificare facultativă, așa cum îi spune însuși numele, un necesar pentru toți oamenii ce doresc să activeze în societatea modernă și să țină pasul cu evoluția.

Numărul din ce în ce mai mare de potențiali studenți au impus universităților să se dezvolte, să crească în dimensiune, acestea devenind, dintr-o instituție clasică, un adevărat agent economic ce are nevoie de fonduri din ce în ce mai numeroase pentru a-și putea susține activitatea. Ținând cont de acest eveniment, dublat și de apariția marketing-ului ca știință economică și de prevalarea imaginii în detrimentul esenței, prestigioasa academie a fost pusă în situația de a se adapta la piață și de a concura cu numeroase forme de învățământ alternativ ce nu au ezitat în a profita de toate avantajele pe care tehnologia le pune la dispoziție. Astfel, imaginea publică și relațiile cu potențialii clienți au devenit un centru de interes pentru orice entitate academică, în contextul unei “piețe” libere a educației, fiecare interacțiune cu viitorii studenți fiind importantă nu doar pentru creșterea prestigiului, ci și pentru garantarea integrității economice.

Dacă acest fenomen nu este de ajuns pentru a considera că instituțiile clasice de învățământ superior au fost puse în fața unor schimbări mai mult sau mai puțin dramatice, instaurarea epocii vitezei a adus în ecuație o condiție care aproape că impune oricărei universități adoptarea revoluției digitale: informația circulă mai repede decât oricând. Un prim efect al acestui fenomen îl reprezintă flexibilitatea ridicată a viitorilor studenți în a aborda mai multe profile ce nu sunt, neapărat, conexe. Acest lucru scade considerabil plaja de clienți siguri pe care, în funcție de profil, o universitate poate aprecia că îi va avea, uneori simpla impresie creată în urma interacțiunii cu o facultate putând să îl atragă sau să îl îndepărteze pe potențialul student. Iar, atunci când vorbim de crearea impresiilor, există numeroase studii care au demonstrat faptul că primul contact creează în mintea părților o imagine foarte puternică ce poate influența masiv atitudinea reciproc manifestată. În contextul expus mai sus, acest prim contact s-ar traduce prin procesul de admitere pe care un viitor student trebuie să îl parcurgă pentru a putea fi acceptat în cadrul unei unități de învățământ superior.

Surprinzător sau nu, acest proces este tratat superficial în multe dintre cazuri, lucru care, din punctul meu de vedere, afectează puternic imaginea unei instituții, putând, uneori, să evidențieze probleme mult mai mari ce pot fi sesizate în mod indirect de către persoanele ce sunt puse în fața deciziei de a se înscrie în programele educaționale organizate de către unitățile în cauză.

În primul rând, apelând la ipoteza menționată anterior despre importanța primului contact, un proces de admitere slab organizat transmite viitorului student o imagine nefavorabilă, ce poate cântări greu în contextul în care acesta are la dispoziție mai multe opțiuni. Desigur, este absurd să considerăm că organizarea deficientă a unui astfel de proces reprezintă, într-adevăr, un semnal de alarmă cu privire la calitatea serviciilor oferite, însă, persoanele din exterior pot percepe indirect acest lucru și pot lua decizii în consecință.

În al doilea rând, în contextul unor generații ce sunt etichetate, din ce în ce mai des, drept native digital, un proces de admitere care pune în fața tinerilor mult blamata barieră a birocrăției și a a perioadelor interminabile de așteptare o să afecteze prematur gradul de încredere pe care aceștia îl acordă sistemului academic ce, din perspectiva lor, va părea că nu reușește să se adapteze nevoilor curente și să “îmbrățișeze” tehnologia ce a devenit, fără doar și poate, o prezență ubicuă în orice domeniu. Această lipsă a digitalizării aduce în discuție un subiect adesea regăsit în campaniile de promovare ale programelor de studii alternative, anume deconectarea dintre cerințele reale impuse de către o industrie și programele adesea învechite pe care le propun facultățile ce ar trebui să “producă” specialiști pregătiți să penetreze piața muncii imediat după finalizarea studiilor. Acest lucru este cu atât mai grav cu cât concentrăm discuția asupra facultăților cu profil tehnic, conduse de specialiști în domeniu, care nu reușesc să își recomande specificul prin sisteme care să funcționeze eficient și care să demonstreze abilitatea tehnică a personalului.

Plecând de la aceste probleme, am încercat să elaborez o soluție care să simplifice procesul de admitere, folosind modalități de comunicare familiare generațiilor curente, și care să eficientizeze efortul depus de către personalul implicat, transmițând tuturor părților ideea unei infrastructuri calitative, digitalizate, ce rezolvă o problemă actuală cu ajutorul unor mijloace moderne.

Soluția la care am ajuns în urma desfășurării etapelor descrise în cadrul lucrării curente se prezintă sub forma unui sistem informatic de gestiune, denumit Flow, ce include toți actorii implicați

în cadrul procesului de admitere și care este construit pe baza unei structuri flexibile, ce poate fi adaptată pentru majoritatea instituțiilor de învățământ superior din România. Totuși, analiza a avut ca punct de plecare procesul de admitere desfășurat de către Academia de Studii Economice din București, fapt pentru care scenariul implementat reproduce etapele acestuia. Obiectivul final al lucrării îl reprezintă obținerea unui sistem fiabil, ușor de întreținut și îmbunătățit, ce poate fi scalat printr-un procedeu bine documentat astfel încât să poată acoperi nevoile reale, imediate. Un punct cheie ce definește aplicația și, prin urmare, întregul proiect, se dorește a fi pragmatismul: rezolvarea unei probleme existente, identificate în cadrul unui mediu atent observat.

Capitolele următoare prezintă, gradual, etapele pe care orice produs software trebuie să le parcurgă pentru a putea fi definitivat, implementat și, mai important, documentat, astfel încât orice dezvoltator să înțeleagă nevoia ce a condus la apariția sistemului, modul în care sistemul a fost modelat, detaliile de implementare ale modulelor ce îl compun și toate funcționalitățile puse la dispoziția utilizatorilor finali.

Astfel, primul capitol prezintă, succint, procesul de analiză întreprins ce m-a ajutat să scot în evidență acele obiective principale ce trebuie atinse, să diferențiez necesarul ce trebuie implementat de funcționalitățile ce nu prezintă beneficii imediate și să creionez un nivel abstract, de ansamblu al întregului sistem pe care să îl transpun, ulterior, în termeni tehnici.

Al doilea capitol tratează proiectarea sistemului și reprezintă o etapă de tranziție ce racordează modelele generale, observate natural la echivalentul lor tehnic, dar fără a fi extrem de specific. Importanța acestui capitol constă, însă, în validarea informațiilor obținute în urma analizei desfășurate, cazurile incongruente putând fi ușor observate în momentul traducerii lor într-o formă mai apropiată de mediul tehnologic.

Cel de-al treilea capitol, implementarea, prezintă în detaliu și cu un aport mare de termeni tehnici întregul proces de creare a sistemului informatic, fiind explicate atât tehnologiile utilizate, motivațiile din spatele alegerilor făcute, avantajele sistemului și arhitectura ce asigură funcționarea acestuia, cât și pașii ce trebuie făcuți pentru publicarea aplicației, scalarea acesteia și asigurarea mentenanței.

Ultimul capitol major, prezentarea detaliată a aplicației, surprinde toate particularitățile produsului final, explicând funcționalitățile existente și exemplificând utilizarea lor, grupându-le după tipului de utilizator căruia îi sunt destinate.

Lucrarea se încheie prin formularea unor concluzii ce vor conține atât o parte de sinteză, implicând toate capitolele anterioare și apreciind rezultatul final din punct de vedere al calității și utilizabilității, cât și o parte în care vor fi definite potențiale direcții de dezvoltare a sistemului.

Capitolul 1 - Etapa de analiză

1.1 Analiza contextului

Înainte de a recurge la modalitățile clasice de analiză și modelare, ce constituie prima etapă în cadrul procesului de proiectare și construire a unui produs software, am considerat că este necesară realizarea unei sinteze a tuturor motivelor pentru care, din punctul meu de vedere, lucrarea curentă are relevanță și poate rezolva o problemă existentă ce afectează negativ imaginea publică a unei instituții de învățământ superior de prestigiu. În elaborarea soluției am ținut cont de experiența personală pe care am acumulat-o în urma observării procesului din trei ipostaze distincte ce mi-au permis să îmi fac nu doar o părere de ansamblu, ci și să acumulez informații despre modul în care întregul proces de admitere se derulează.

Prima interacțiune a fost reprezentată, bineînțeles, de participarea în calitate de viitor student, când am putut să observ cel mai bine impactul negativ resimțit de către toți cei prezenți: perioadele lungi de așteptare, combinate cu temperaturile ridicate, inconsistența fluxurilor de persoane și numărul mare de participanți au “reușit” să deterioreze părerea multora. În cazul câtorva persoane cu care am discutat ulterior, au reușit, chiar, să altereze decizia finală cu privire la alegerea universității ce se va clasa în vârful opțiunilor personale, motivația având la bază mai buna organizare a procesului de admitere în cadrul instituțiilor concurente ce a reușit să transmită, indirect, faptul că acestea beneficiază de o infrastructură modernă, mai eficientă și acordă mai multă atenție studenților.

A doua poziție din care am putut să observ mai atent și să iau parte activ la procesul de admitere a fost participarea ca voluntar, din ipostaza de membru al Sindicatului Studenților din Cibernetică. Ajuns de partea cealaltă a barierei, am remarcat încă din prima zi efortul uriaș depus de către organizatori pentru a face față tuturor factorilor menționați anterior ce aveau un efect la fel de puternic, îngreunând munca depusă și scăzând eficiența acțiunilor demarate. În acest sens, în poziția de operator pe care am ocupat-o în două dintre cele trei zile, am resimțit, la persoana întâi, dificultatea cu care sistemul utilizat pentru confirmarea înscrierilor reușea să gestioneze numărul mare de participanți. Această experiență a reprezentat primul impuls ce m-a determinat să apreciez posibilitatea implementării unui sistem nou, auxiliar, care să ofere sprijin și să disemineze rapid informațiile relevante atât în rândul elevilor, cât și în rândul studenților voluntari și a personalului universității, fie că vorbim despre operatori, casieri sau persoane responsabile de validarea dosarelor.

Cea de-a treia ipostază și, poate, cea mai relevantă pentru motivația de a aborda această temă, a fost reprezentată de elaborarea unui modul informatic care să țină evidența numerelor de ordine curente și să aproximeze, pe baza unor calcule statistice, timpul mediu de așteptare al unei serii variabile de persoane, fiind un instrument informativ atât pentru organizatori, cât și pentru viitori studenți, accesul la adresa pe care aplicația a fost expusă fiind public. Deși am fost sceptic, inițial, rezultatele observate nu au făcut decât să confirme că implementarea unui sistem nou, interactiv reprezintă o nevoie și nu doar un act de inovare: cozile mari din fața clădirii facultății s-au redus considerabil, elevii apreciind perioada în care ar trebui să revină la facultate pe baza bonului de ordine și a estimărilor făcute de sistem, fapt ce a crescut, implicit, transparența procesului și a permis persoanelor implicate să cunoască, cel puțin aproximativ, durata etapelor.

Aceste trei perspective au fost esențiale atât pentru a înțelege problema, motivele apariției ei, punctele critice existente în cadrul procesului, cât și pentru a identifica principalele caracteristici ale unei soluții care să ofere un sprijin real și a cărui impact pozitiv să fie ușor de apreciat, astfel încât

efortul dezvoltării sistemului și a implementării sale într-un context real, în cadrul unui proces de admitere, să fie motivabil.

1.2 Analiza sistemului informatic

1.2.1 Identificarea nevoilor și a funcționalităților

Sistemul informatic ce urmează a fi implementat are ca scop, așa cum am menționat anterior, simplificarea și eficientizarea procesului de înscriere a studenților în cadrul unei universități, adoptând, sub forma de caz particular, procesul de admitere al Academiei de Studii Economice din București. Aplicația rezultată în urma implementării acestui sistem are ca scop reducerea timpului de așteptare în ziua admiterii, precum și digitalizarea unor etape și operații ce îngreunează stocarea și prelucrarea datelor, folosind tehnologii actuale, adoptate ca standard de către industrie.

Pentru a putea oferi un sprijin real în cadrul desfășurării procesului de admitere, aplicația trebuie să asigure atât facilități elementare, existente într-o formă sau alta în implementarea actuală, cât și facilități suplimentare sau îmbunătățite, care să particularizeze sistemul și să îi garanteze un avantaj competitiv.

Astfel, printre facilitățile de bază putem identifica:

- înscrierea studenților la concursul de admitere
- validarea datelor și posibilitatea de actualizare a acestora
- selectarea opțiunilor dorite în ordinea preferințelor
- generarea documentelor rezultate în urma înscrierii
- identificarea unică a unui participant pe baza codului numeric personal

În continuarea acestora, este implementat un set nou de funcționalități care să ofere sistemului posibilitatea de a putea fi utilizat în cadrul unor scenarii complexe, ce modelează realitatea, simplificând numeroase subprocese:

- informarea în timp real cu privire la numărul curent de înscriși
- generarea bonurilor digitale de ordine
- asocierea bonurilor de ordine participanților în funcție de timpul de sosire al acestora și ținând cont de locația curentă
- achiziționarea creditelor necesare selectării unui număr de opțiuni prin intermediul platformei
- calcularea timpului mediu de așteptare pentru un student
- estimarea timpului de așteptare pentru fiecare student
- notificarea participanților ce urmează să intre în incinta facultății
- organizarea voluntarilor prezenți la înscrieri prin implementarea unui modul de management a resursei umane și a pozițiilor disponibile
- comunicarea bidirecțională, prin notificări și cereri, între voluntari și coordonatorul activității
- informarea participanților cu privire la pașii ce trebuie urmați pentru finalizarea procesului de înscriere
- revizuirea, corectarea, completarea și confirmarea datelor participanților
- identificarea unică a unui participant pe baza bonului de ordine
- distribuirea documentelor generate prin intermediul poștei electronice

- repartizarea studenților în cadrul facultăților, în funcție de opțiuni și de criteriile de admitere
- informarea studenților, prin intermediul poștei electronice, cu privire la rezultatele intermediare și finale ale concursului de admitere

Toate diagramele incluse în lucrarea curentă sunt construite respectând specificațiile limbajului de modelare universal (UML), pentru generare fiind folosit programul *Visual Paradigm Community Edition 2015*.

1.2.2 Diagramele generale ale cazurilor de utilizare

Toate funcționalitățile menționate mai sus, ce reprezintă fundația întregului sistem, pot fi exemplificate cu ajutorul diagramelor cazurilor de utilizare ce surprind, într-o formă naturală, ușor de urmărit, principalele acțiuni pe care diferitele tipuri de utilizatori finali le pot efectua în cadrul aplicației, precum și succesiunea acestora, în cazul în care reprezintă o condiție necesară.

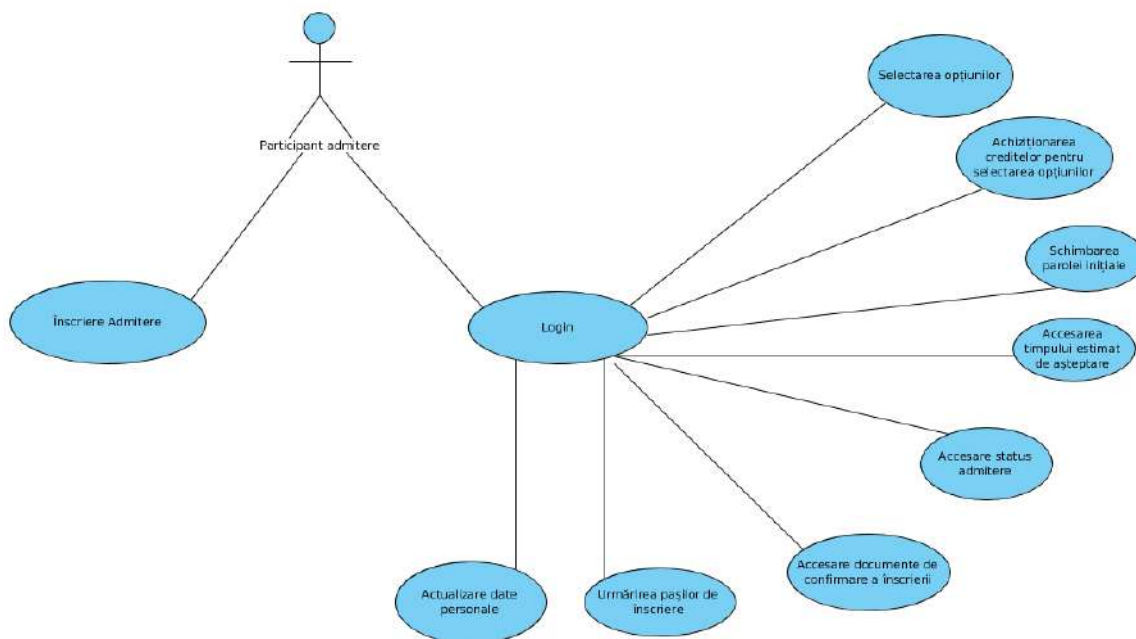


Figura 1.1 – Diagrama cazurilor de utilizare – Participant admitere

Diagrama din *Figura 1.1* prezintă toate acțiunile cărora cel mai important tip de utilizator, participantul la admitere, le poate da curs după momentul înregistrării și, ulterior, a autentificării în sistem. Un astfel de utilizator își poate actualiza datele personale în orice moment ulterior înscrierii, dar până la confirmarea acestora, poate urmări o scurtă prezentare cu pașii pe care trebuie să îi parcurgă în ziua depunerii dosarului, poate solicita un bon de ordine și poate accesa în permanență statusul curent al procesului, precum și timpul personal de așteptare estimat de către sistem pe baza unor calcule statistice. În plus, acesta poate achiziționa așa-numitele credite în baza cărora va putea selecta un număr echivalent de opțiuni de care se va ține cont în momentul repartizării.

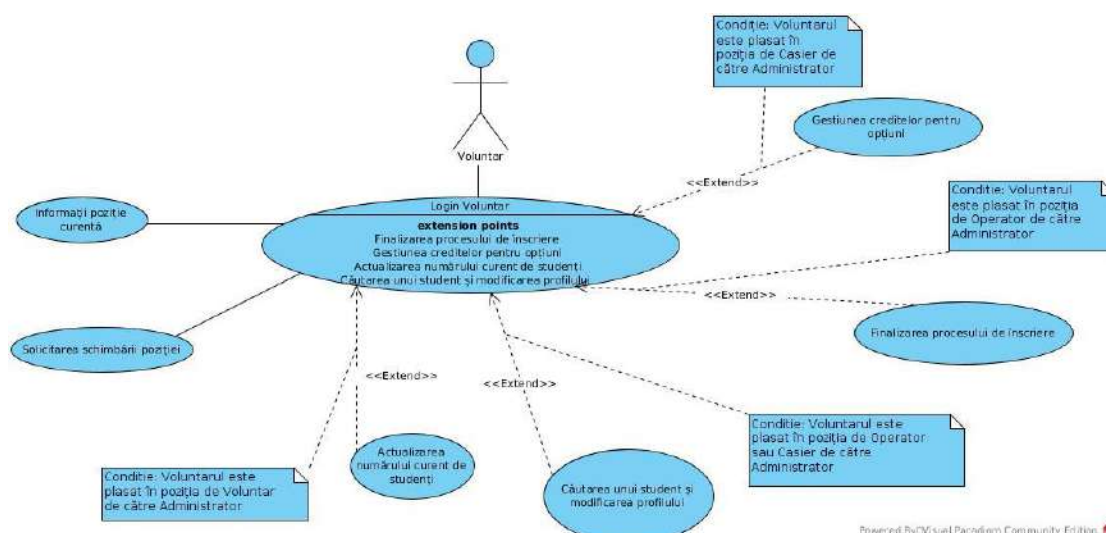


Figura 1.2 – Diagrama cazurilor de utilizare – Voluntar

Fiecărui utilizator de tip Voluntar îi este atribuit un rol, ce poate varia între Voluntar, Casier și Operator. În baza fiecăruia dintre acestea, funcționalitățile la care are acces utilizatorul se modifică pentru a-i permite îndeplinirea sarcinilor revenite. Astfel, pe lângă funcționalitățile comune, consultarea informațiilor despre poziția curentă și posibilitatea de a solicita o schimbare de poziție, voluntarii ce ocupă rolul standard vor avea acces la un modul ce le va permite modificarea numărului de studenți cărora le-a fost permis accesul în clădire, acordarea de bonuri de ordine în cazuri speciale și notificarea următoarelor serii de studenți ce trebuie să se pregătească pentru intrare. Spre deosebire de aceștia, casierii și operatorii vor avea acces la un modul ce va permite căutarea unui student pe baza codului numeric personal sau a bonului de ordine, precum și actualizarea condiționată a profilului acestuia. Casierii vor putea adăuga credite studenților, iar operatorii vor putea confirma cererile de înscriere, punând astfel în funcțiune mecanismul de generare a documentelor rezultate.

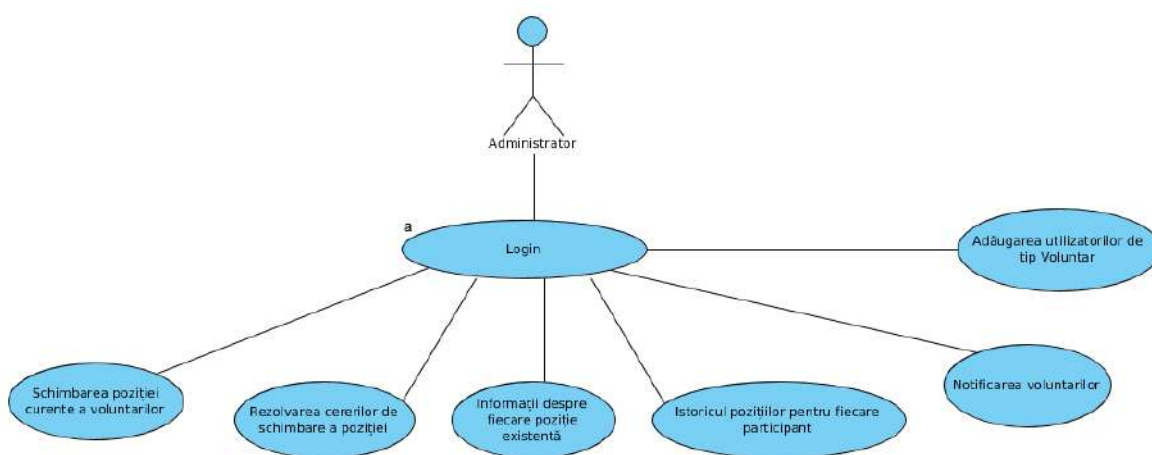


Figura 1.3 – Diagrama cazurilor de utilizare – Administrator

Ultima diagramă majoră, expusă în *Figura 1.3*, surprinde acțiunile pe care un Administrator le poate întreprinde, acestea având o puternică orientare managerială. Un utilizator de acest tip poate, în orice moment, să acceseze pozițiile fiecărui voluntar prezent, împreună cu un istoric al acestora, și să le modifice în funcție de nevoie sau pentru a soluționa cererile de schimbare a poziției primite. Tot în atribuțiile sale intră și crearea conturilor de tip Voluntar, precum și transmiterea directă de indicații către aceștia prin intermediul notificărilor.

1.2.3 Diagramele de activitate

Spre diferență de diagramele cazurilor de utilizare, diagramele de activitate tratează în detaliu noțiunea de proces, scoțând în evidență pașii ce trebuie executați în cadrul unor acțiuni bine definite, ce pot fi privite independent unele de altele în momentul derulării. Totalitatea acestor acțiuni determină comportamentul final al sistemului și gestionează fluxul de date existent. Pentru exemplificare, am selectat trei dintre activitățile cele mai importante atât din punct de vedere logic, cât și tehnic.

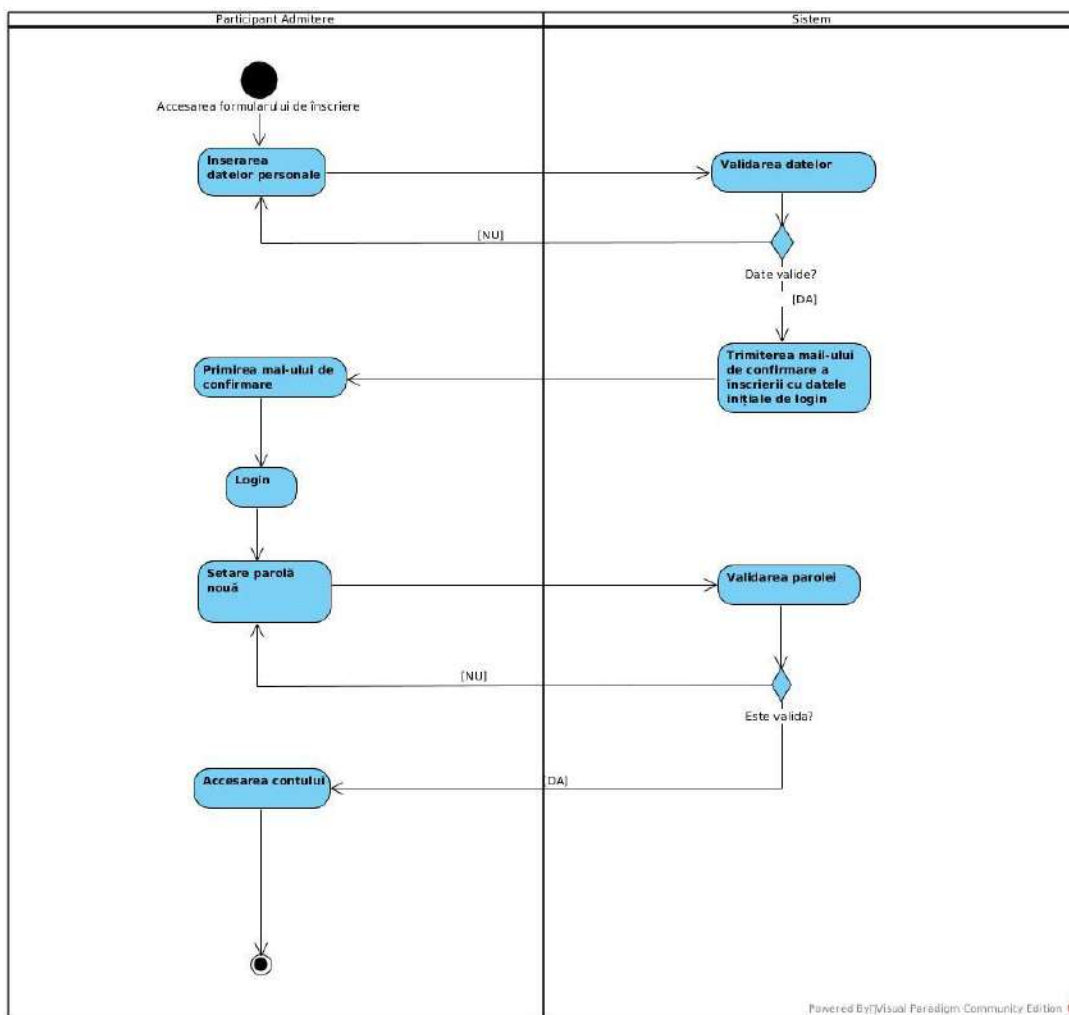


Figura 1.4 – Diagrama de activitate – Înregistrarea unui participant în sistem

Prima diagramă, atașată în *Figura 1.4*, surprinde, chiar, primul pas pe care un participant trebuie să îl facă, anume înscrierea în sistem. Datele furnizate de acesta în cadrul formularului sunt evaluate și validate, în condițiile unui rezultat pozitiv fiind creată o nouă înregistrare și trimis un e-mail ce conține pașii următori ce trebuie urmați. Astfel, după înregistrare, utilizatorul se poate loga, fiind, însă, redirecționat, în momentul primei autentificări, către o interfață de schimbare a parolei implicite generate de către sistem cu o parolă nouă. Dacă această parolă verifică standardele de securitate implicate, utilizatorul poate accesa toate resursele și funcționalitățile specifice puse la dispoziție de către sistem.

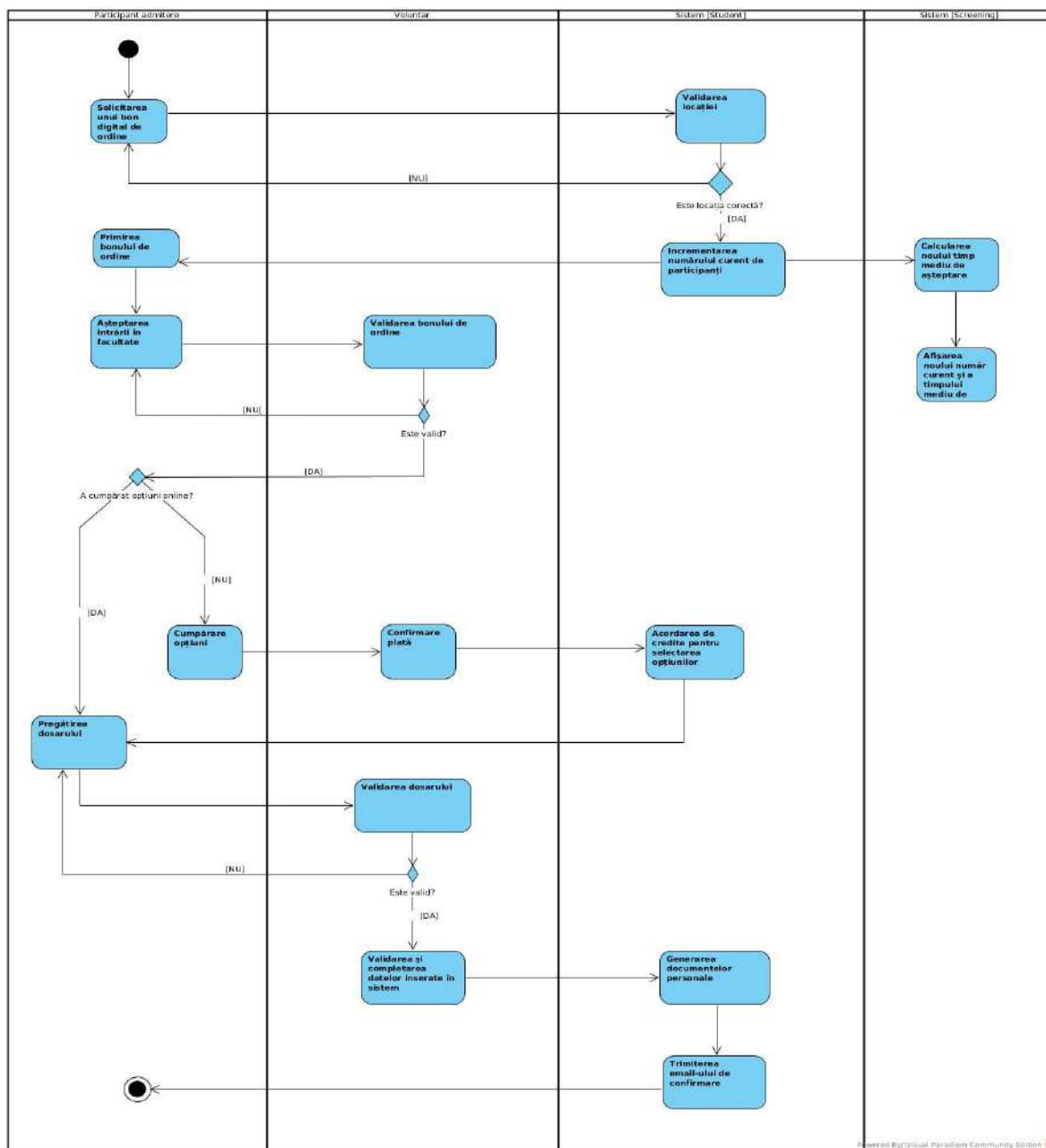


Figura 1.5 – Diagrama de activitate – Procesul de înscriere

Cea de-a doua diagramă înglobează activitatea de bază a sistemului, procesul de admitere parcurs în ziua depunerii dosarului, și identifică toate acele entități care interacționează pentru a valida datele și a finaliza înscrierea unui participant. Pentru a ajunge la acest rezultat, sunt angrenate multiple subprocese, cum ar fi generarea bonului de ordine, validarea locației, achiziționarea creditelor și selectarea opțiunilor, ce determină o schemă bogată de interacțiune atât între modulele ce compun aplicația, cât și între actorii implicați. Această diagramă este atașată în *Figura 1.5*.

Ultima diagramă de activitate, echivalentă *Figurii 1.6*, descrie, spre diferență de celelalte, un proces destinat personalului auxiliar, anume voluntarilor și administratorilor, și are scopul de a gestiona resursa umană: activitatea de schimbare a poziției unui voluntar. Pe lângă schimbarea directă a poziției pe care administratorul o poate face în orice moment, un voluntar poate trimite o cerere, ce va conține poziția curentă și poziția pe care dorește să se transfere. Această cerere este analizată de către un administrator, decizia acestuia fiind transmisă către inițiator sub forma unei notificări.

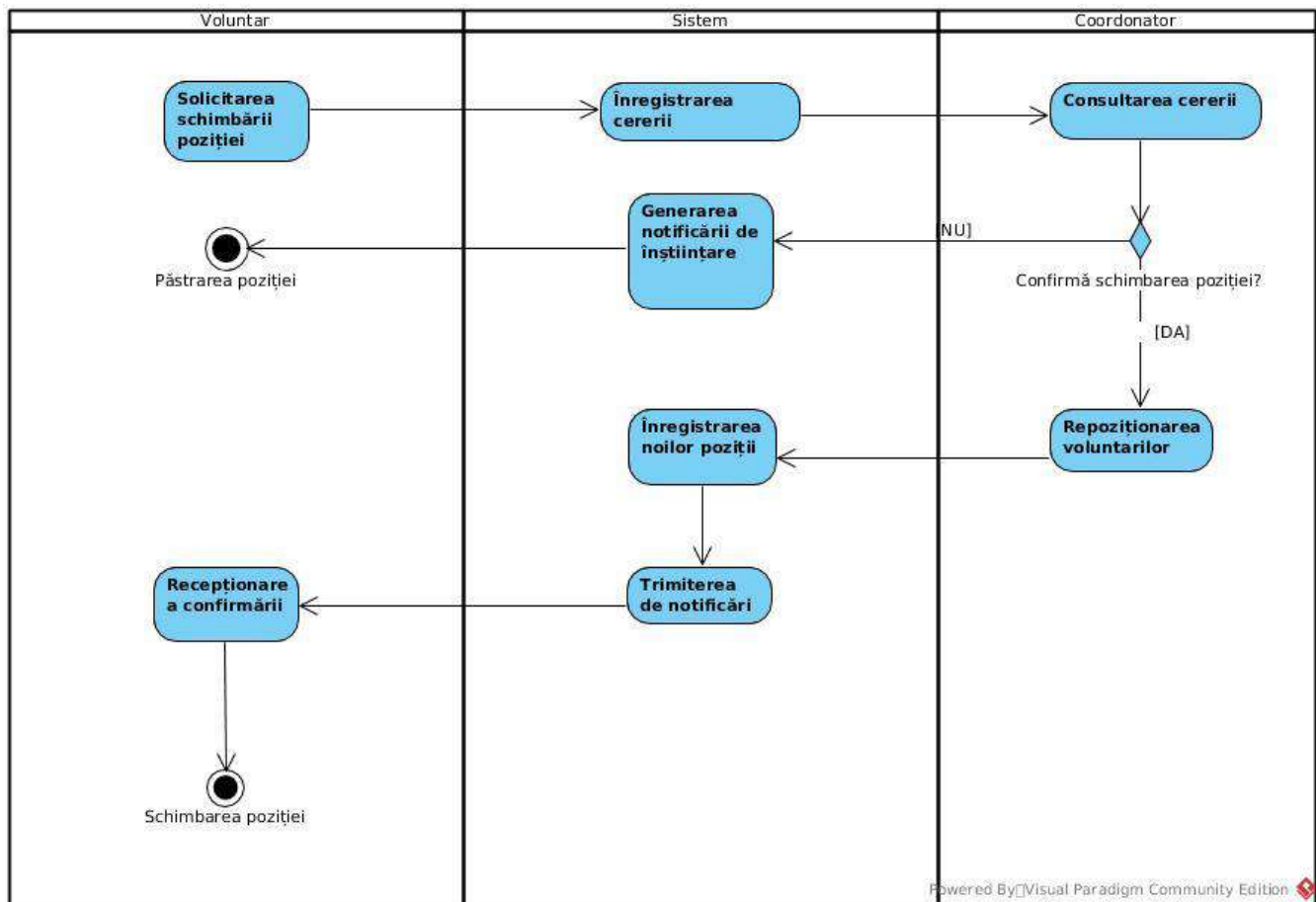


Figura 1.6 – Diagrama de activitate – Schimbarea poziției unui voluntar

1.2.4 Diagramele de stare

Principalul scop al diagramelor de stare este identificarea facilă a tuturor pozițiilor în care o entitate existentă în sistem se poate afla într-un anumit moment. Pe baza acestor informații se poate analiza dacă logica aflată în spatele proceselor conduce la apariția unor cazuri conflictuale care să afecteze integritatea unui obiect, validând, astfel, fluxurile concepute.

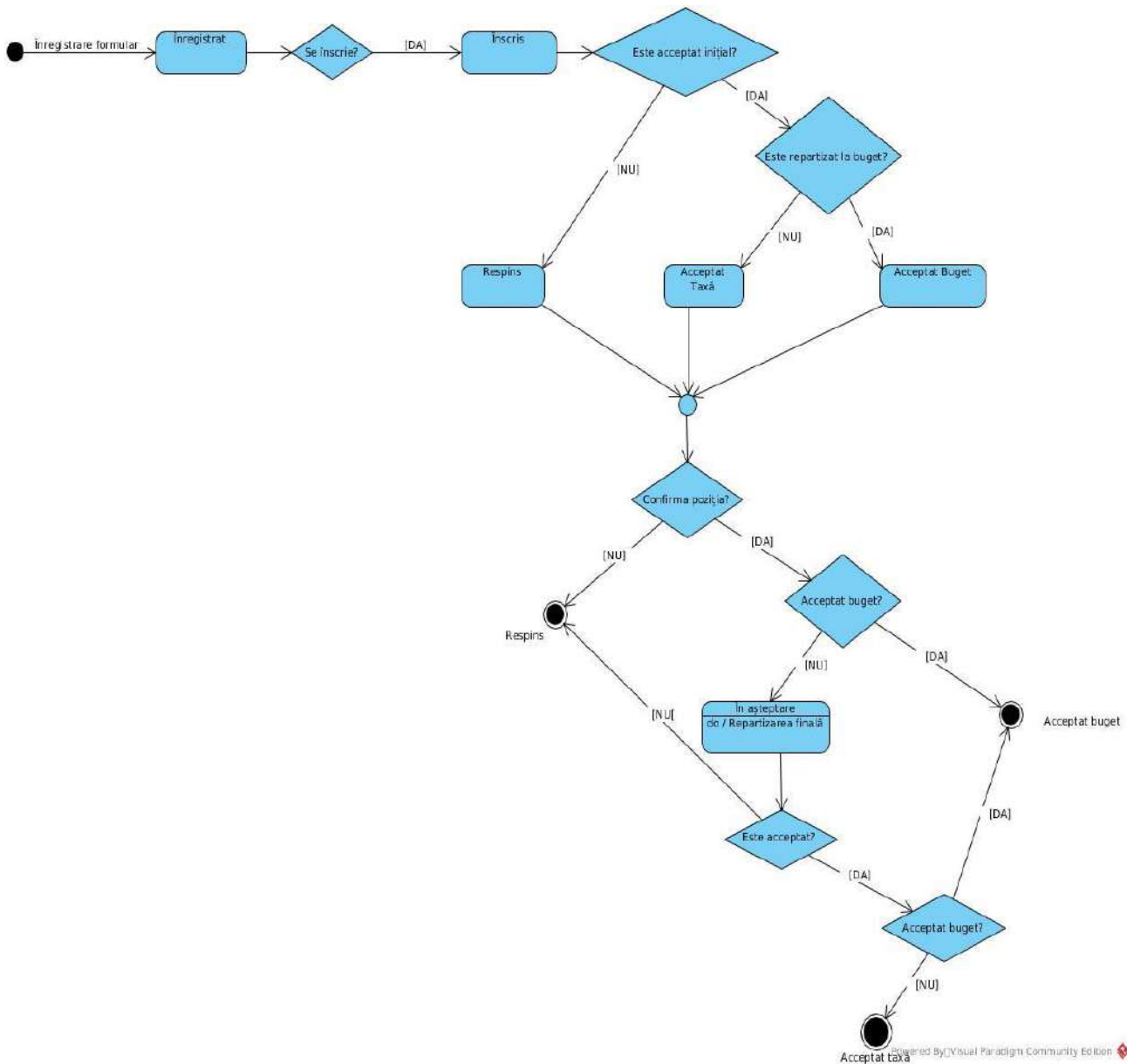


Figura 1.7 – Diagrama de stare – Participant la admitere

Cea mai complexă entitate existentă în sistem, participantul, cunoaște mai multe stări ce descriu, indirect, cât de mult a avansat în cadrul procesului de admitere, fiind un mod concis de a exprima condiția acestuia. Așa cum se poate observa din diagrama atașată în Figura 1.7, stările pe

care le poate avea un obiect de tipul participant sunt înregistrat, înscris, respins, acceptat buget sau acceptat taxă, acestea reușind să descrie exhaustiv toate rezultatele ce pot apărea la finalul concursului de admitere sau a etapelor intermediare.

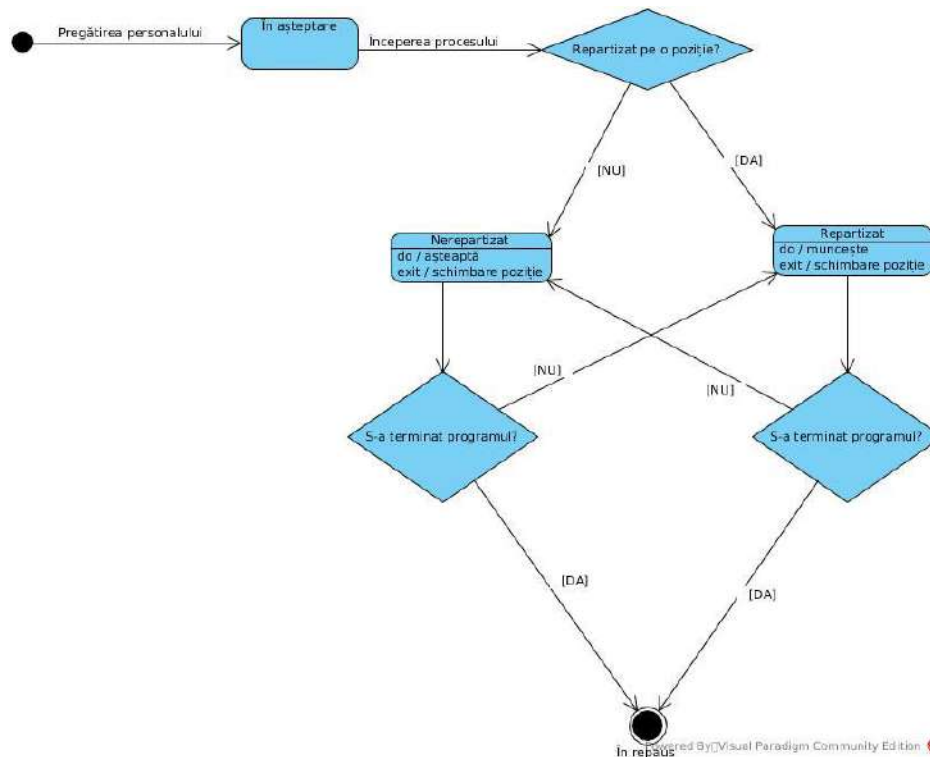


Figura 1.8 – Diagrama de stare - Utilizator

Spre deosebire de administrator, în cazul căruia nu pot fi identificate mai multe stări de existență, voluntarul poate avea o stare inițială, denumită stare de așteptare, ce se încheie odată cu începerea efectivă a procesului. Ulterior, pe baza repartizărilor făcute de către administrator, un voluntar poate fi repartizat sau nerepartizat, putând părăsi oricare dintre aceste două stări fie prin adresarea unei cereri de schimbare a poziției, fie prin acțiunea directă a administratorului. La finalul programului, voluntarul intră în starea finală, cea de repaus.

1.2.5 Diagramele de secvență

Diagramele de secvență dețin, dintr-un anumit punct de vedere, cel mai înalt grad de complexitate deoarece introduc în discuție factorul temporal, absent în cazul tuturor celorlalte reprezentări. În ceea ce privește conținutul, acestea se aseamănă cu diagramele de activitate, surprinzând acțiunile întreprinse succesiv de diferiți actori pentru finalizarea unui scenariu unic, bine definit. Pentru exemplificare, am ales tratarea a trei cazuri relevante care să ilustreze, în detaliu, scenarii fundamentale ce angrenează pe parcursul desfășurării funcționalității de bază.

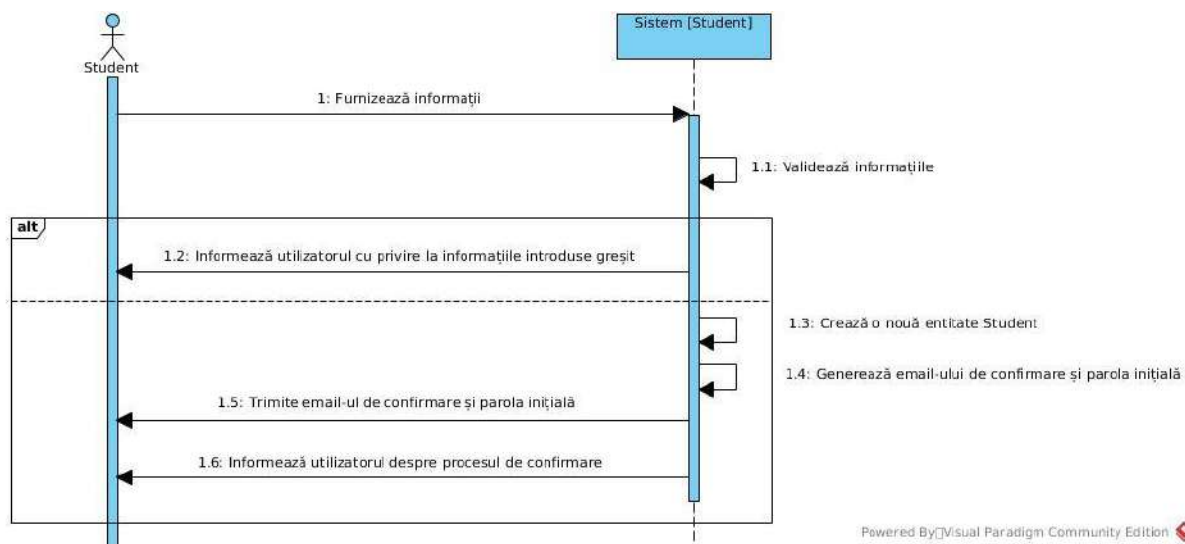


Figura 1.9 – Diagrama de secvență – Înregistrare Student

Diagrama de secvență din *Figura 1.9* descrie scenariul de înregistrare a unui student și setează, ca agent principal, participantul care, prin furnizarea datelor necesare va căpăta calitatea de Student. Din punctul de vedere al aplicației, este determinată apariția unei noi entități de tipul Student în sistem ce îl identifică unic pe înscris și de care acesta se poate folosi pentru a utiliza toate funcționalitățile ce îi sunt destinate.

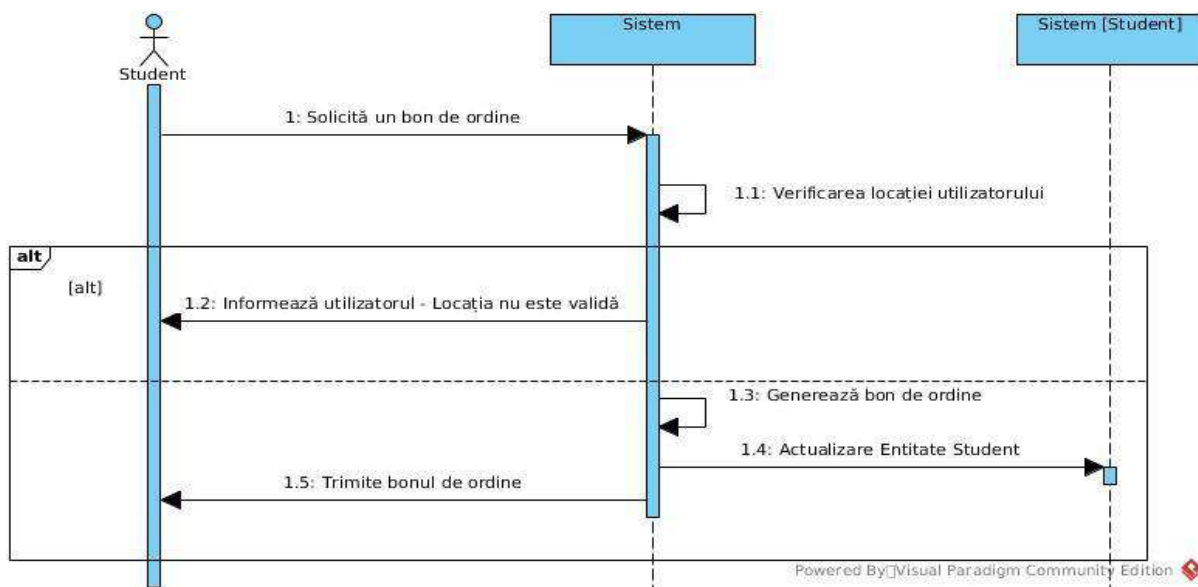


Figura 1.10 – Diagrama de secvență – Solicitarea bonului de ordine

Diagrama de secvență ce descrie scenariul de solicitare a unui bon de ordine, inclusă în *Figura 1.10*, surprinde acțiunile ce vor fi întreprinse de către un Student în ziua înscrierii la facultate. Pe baza unor verificări bazate pe locație, acesta va primi un bon de ordine cu ajutorul căruia vor fi generate informații personalizate despre ora la care trebuie să se prezinte la intrarea în incinta facultății.

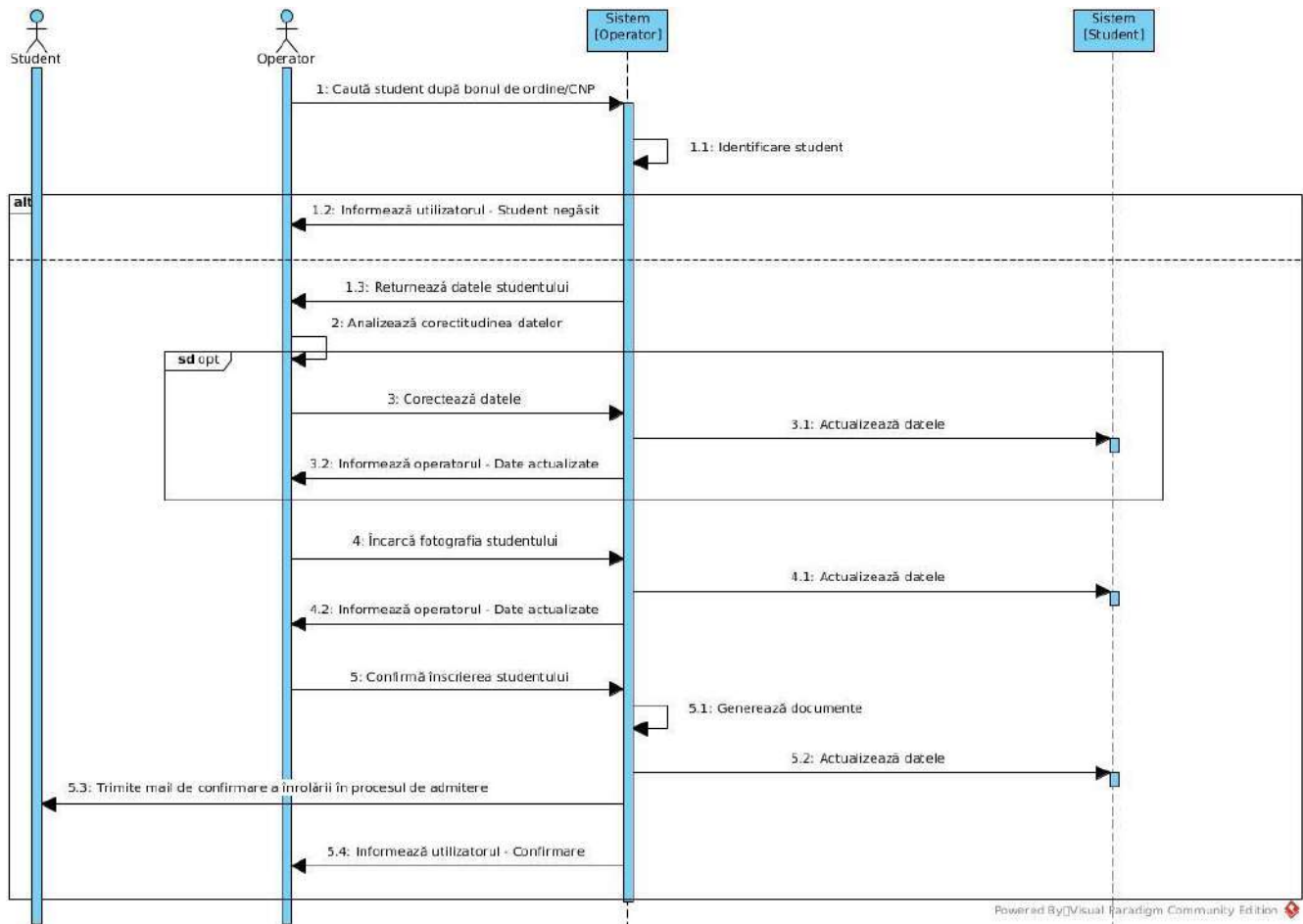


Figura 1.11 – Diagrama de secvență – Confirmarea înscrierii unui student

Ultima diagramă, cea care implică cel mai mare grad de complexitate, surprinde pasul ce precede finalizarea procesului de înscriere a unui student, validarea datelor și confirmarea cererii de înscriere. Un utilizator cu rolul de operator va analiza veridicitatea datelor înscrise în sistem și va derula procesul de confirmare. Această acțiune va genera documente specifice și va trimite e-mail-uri de confirmare studentului în cauză. Diagrama poate fi consultată în *Figura 1.11*.

Capitolul 2 – Etapa de proiectare

Capitolul de proiectare are ca principal obiectiv realizarea unei asocieri între entitățile observate în cadrul etapei de analiză și mediul informatic, reprezentând un stadiu intermediar foarte important cu ajutorul căruia se realizează translația realității modelate la un limbaj tehnic, utilizat, ulterior, pentru implementarea efectivă a sistemului. Fiecare subcapitol tratează, dintr-un unghi diferit, procesul menționat, descompunând în pași succesivi cele mai importante nivele ale sistemului: nivelul de reprezentare a datelor, nivelul software și nivelul hardware.

2.1 Diagrama de clase detaliată

Diagrama de clase detaliată are o semnificație aparte în descrierea nivelului de reprezentare a datelor și are rolul de a surprinde totalitatea claselor utilizate la nivelul sistemului informatic, precum și a acțiunilor pe care acestea le pot invoca la un moment dat. Altfel spus, are ca finalitate descrierea atentă a tuturor entităților ce interacționează pe baza unor reguli bine definite pentru a conferi sistemului funcționalitatea dorită.

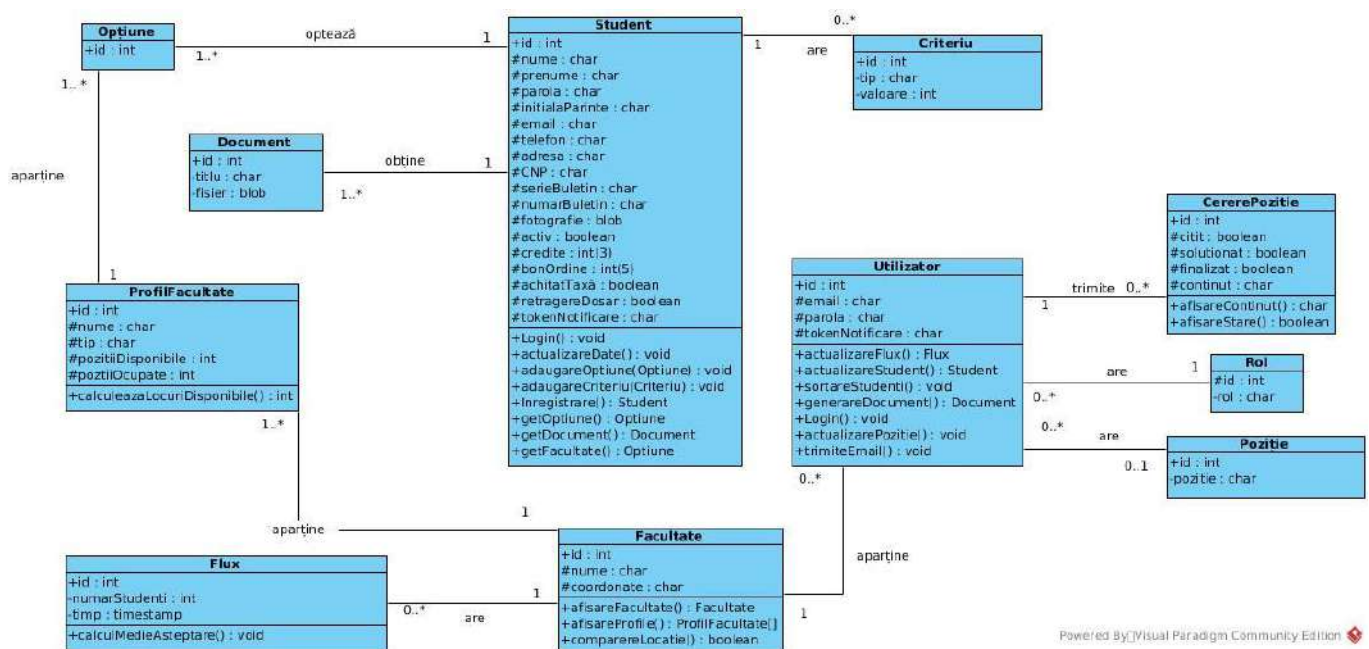


Figura 2.1 – Diagrama de clase detaliată

2.2 Proiectarea bazei de date

Pornind de la diagrama de clase detaliată expusă anterior, în *Figura 2.1*, procesul de proiectare a bazei de date translatează informațiile acumulate despre actorii existenți în cadrul fenomenului observat la o formă tehnică generală, agnostică față de tehnologiile existente, ce va putea fi implementată ulterior în orice sistem ce respectă principiile fundamentale ale bazelor de date relaționale.

Pentru a putea realiza transformarea diagramei de clase într-o schemă generală care să reproducă particularitățile unei baze de date, trebuie aplicat un set de transformări obligatorii:

- fiecare clasă identificată va deveni o tabelă
- atributele claselor vor deveni câmpuri identificate printr-un tip de dată în cadrul tabelelor
- acel atribut ce identifică unic o instanță va deveni cheie primară
- entitățile dependente vor deveni tabele dependente și vor conține, sub formă de cheie externă, cheia primară a entității de care depind
- în cazul relației mulți-la-mulți, ce nu poate fi implementată în cadrul unui model relațional, este necesară introducerea unei noi tabele de legătură ce va depinde de tabelele ce se aflau în relația inițială, deținând, sub formă de chei externe, referințe către cheile primare ale instanțelor ce participă la relație

Aplicând aceste instrucțiuni, vom încheia procesul prin a genera o schemă validă, ce poate fi interpretată de către orice sistem de gestiune a bazelor de date.

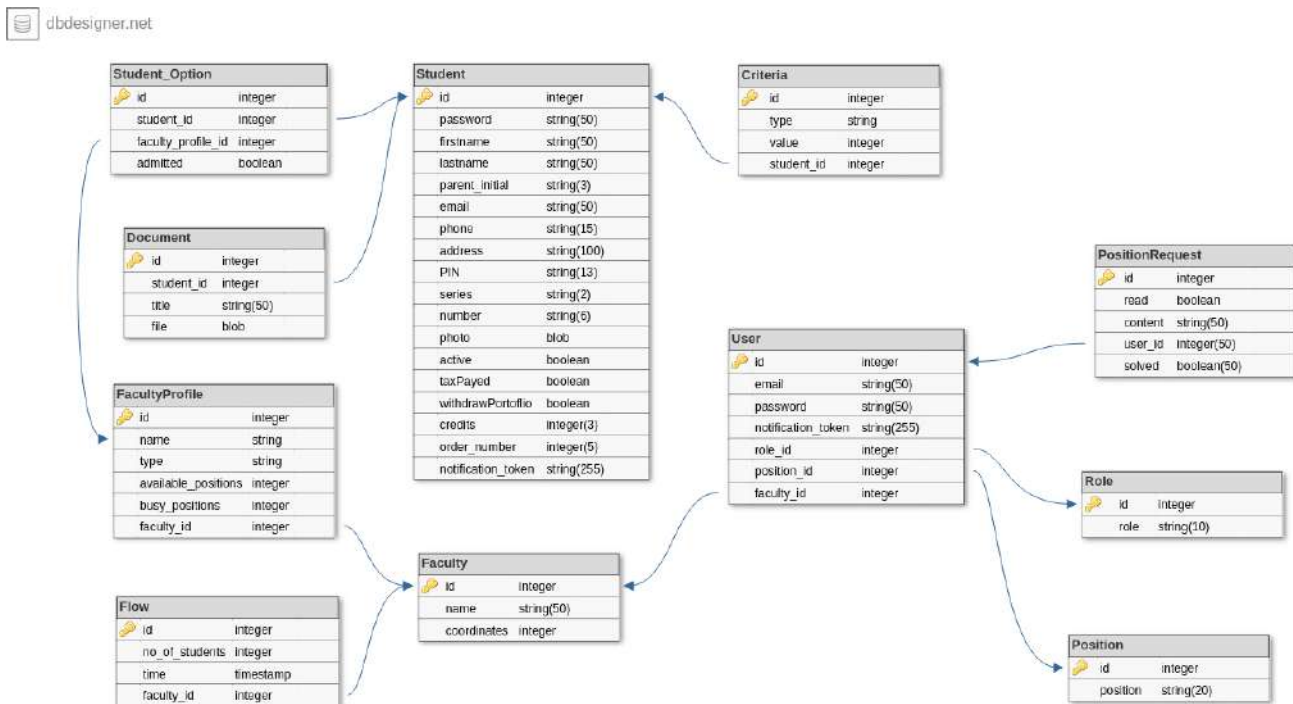


Figura 2.2 – Schema bazei de date

Astfel, analizând schema obținută, putem descrie punctual care este utilitatea fiecărei entități în cadrul sistemului.

Tabela Student va gestiona datele participanților ce se înscriu la concursul de admitere, deținând atât informații care descriu entitatea, cât și informații ce au rol activ în accesarea sistemului (parolă) sau utilizarea funcționalităților puse la dispoziție de acesta. (credite, număr de ordine)

Tabela Utilizator stochează informațiile tuturor actorilor înscriși în sistem ce vor avea un rol auxiliar în organizarea procesului de admitere. Aceasta conține, pe lângă câmpurile proprii, două chei externe ce vor identifica rolul unui utilizator, respectiv poziția pe care acesta o ocupă la un moment dat.

Tabela Rol conține toate rolurile disponibile în cadrul sistemului ce pot fi alocate unui utilizator: Voluntar, Admin, Casier și Operator.

Tabela Poziție înregistrează toate pozițiile fizice pe care un utilizator le poate ocupa în timpul procesului de înscriere, acestea având o dimensiune spațială, concretă, fără a se confunda sau a depinde de rolul asociat. (ex: Sala calculatoare, Ușă intrare, Verificare dosar)

Tabela CererePoziție se ocupă, așa cum poate fi remarcat încă din nume, de stocarea tuturor cererilor de schimbare a poziției pe care un utilizator de tip Voluntar le lansează. Acestea vor fi evaluate și rezolvate de către un administrator, putând conduce la o schimbare a poziției afiliate.

Tabela Criteriu va fi populată în urma procesului de înregistrare a unui Student în cadrul sistemului și va conține toate acele date ce vor fi utilizate ulterior la repartizarea acestuia. Deși tabela Student ar fi putut include toate aceste informații, putând fi observată relația de unu-la-mulți existente între acestea, am optat pentru utilizarea unei tabele separate pentru adăuga un nivel suplimentar de abstractizare ce va permite reutilizarea sistemului în cadrul unui context cu o logică distinctă de repartizare.

Tabela Document va stoca toate actele generate de către instituția universitară ce atestă înscrierea unui elev la concursul de admitere, acesta putând deține, conform relației unu-la-mulți, mai multe documente.

Tabela Facultate conține toate facultățile disponibile în cadrul universității, precum și informații despre locația acestora ce vor fi utilizate de către mecanismul de generare a bonurilor de ordine.

Tabela ProfilFacultate, legată printr-o relație de tip mulți-la-unu de tabela Facultate, identifică toate specializările puse la dispoziție de către o facultate și include, totodată, informații despre forma de finanțare, numărul disponibil de locuri și numărul de locuri ocupate într-un anumit moment.

Tabela Flux surprinde acele entități abstracte care descriu mișcările maselor de studenți pe parcursul procesului de înscriere, pe baza acestora fiind realizate calcule statistice care să informeze studenții despre ora aproximativă la care aceștia trebuie să se prezinte la ușa de intrare.

Tabela StudentOpțiuni este, prin excelență, un detaliu de implementare și realizează descompunerea relației de tip mulți-la-mulți dintre Studenți și Facultăți, stocând, suplimentar, un indicator de tip boolean despre starea curentă: dacă aceasta va deveni opțiunea finală în urma desfășurării procesului de sortare, câmpul își va schimba valoarea.

2.3 Proiectarea interfețelor grafice

Proiectarea interfețelor destinate utilizatorilor are ca scop schematizarea scenariilor de interacțiune dintre utilizatorul final și sistem. În baza acestora poate fi definit un flux de utilizare care să simplifice folosirea sistemului, trasând un set de pași naturali ce îmbunătățesc experiența clientului și, implicit, satisfacția marginală resimțită de acesta.

Ca urmare a structurii modulare și a numărului mare de actori implicați în întregul proces de înscriere gestionat cu ajutorul sistemului, în sistem se pot identifica cinci tipuri de utilizatori, fiecare dispunând de instrumente specifice poziției și de o interfață grafică adaptată.

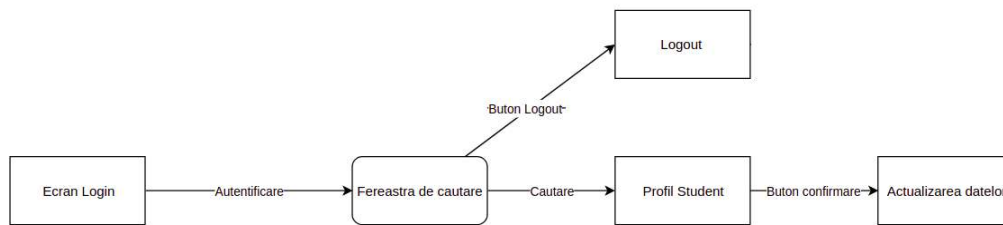


Figura 2.3 – Schemă interfață grafică – Operator și Casier

Două dintre cele patru tipuri particulare de utilizatori ce se ocupă de organizarea procesului de admitere, operatorul, respectiv casierul, împart, din punct de vedere structural, aceeași interfață grafică, principalele funcționalități de care aceștia pot beneficia fiind căutarea unui student pe baza CNP-ului sau a numărului de ordine, actualizarea datelor acestuia și filtrarea studenților pe baza unei condiții. Diferența între cele două instanțe se regăsește, însă, la nivel de detaliu, fiecare dintre acestea având dreptul de a modifica secțiuni diferite și de a invoca acțiuni distincte, cum sunt, de exemplu, finalizarea procesului de admitere, în cazul operatorului, și incrementarea numărului de credite necesare pentru selectarea opțiunilor, în cazul casierului.

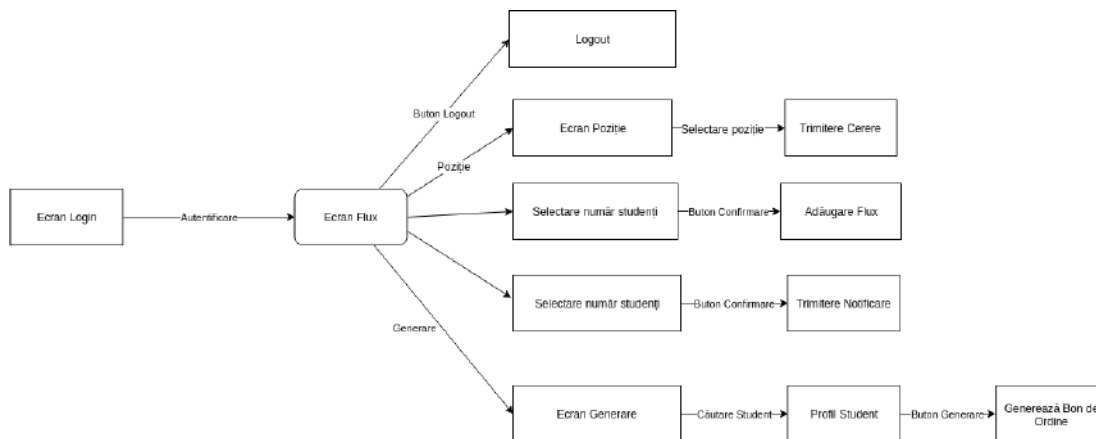


Figura 2.4 – Schemă interfață grafică – Voluntar

Voluntarul beneficiază de o interfață mai complexă, atât din punct de vedere al numărului de pagini, cât și în ceea ce privește acțiunile pe care le poate derula, deoarece rolul său prevede o interacțiune mai strânsă cu modulele sistemului. Odată autentificat, voluntarul poate înregistra fluxuri noi de participanți și, pe baza numerelor de ordine, poate să trimită notificări de înștiințare persoanelor ce trebuie să se prezinte la intrarea în incinta facultății. În cazul în care în ziua înscrierii se prezintă elevi ce nu pot solicita un bon de ordine prin intermediul platformei din cauza unor probleme tehnice ale terminalelor utilizate, voluntarii pot căuta un student și îi pot asocia un număr de ordine. Totodată, fiind rolul cel mai dinamic, voluntarul poate trimite, din fereastra de solicitare, cereri de schimbare a poziției ce vor fi, ulterior, evaluate de către un administrator.

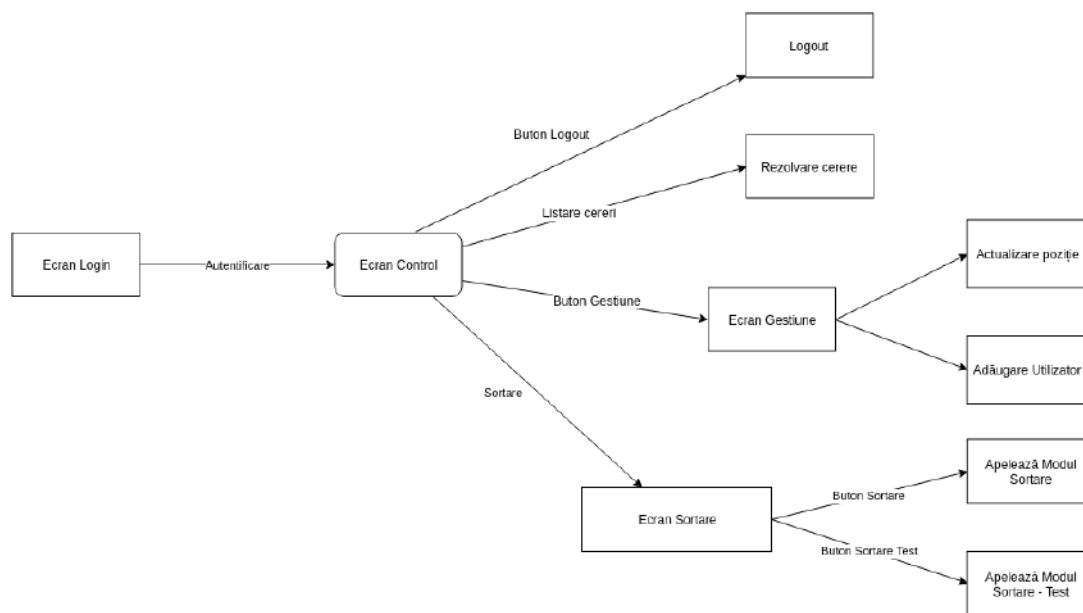


Figura 2.5 – Schema interfață grafică – Administrator

Din poziția de administrator, un utilizator are acces la un panou de control ce expune numeroase funcționalități care să asigure transparența procesului și să pună la dispoziția utilizatorului în cauză instrumente de coordonare a personalului auxiliar. Conform *Figurii 2.5*, un administrator poate actualiza pozițiile voluntarilor fie pe baza cererilor trimise de către aceștia, fie printr-o acțiune independentă, poate adăuga noi utilizatori în sistem cărora le va fi alocat unul dintre cele patru roluri existente și, cel mai important, poate demara procesul de sortare atât sub formă de test, cât și sub formă finală, notificând toți participanții cu privire la rezultatele obținute.

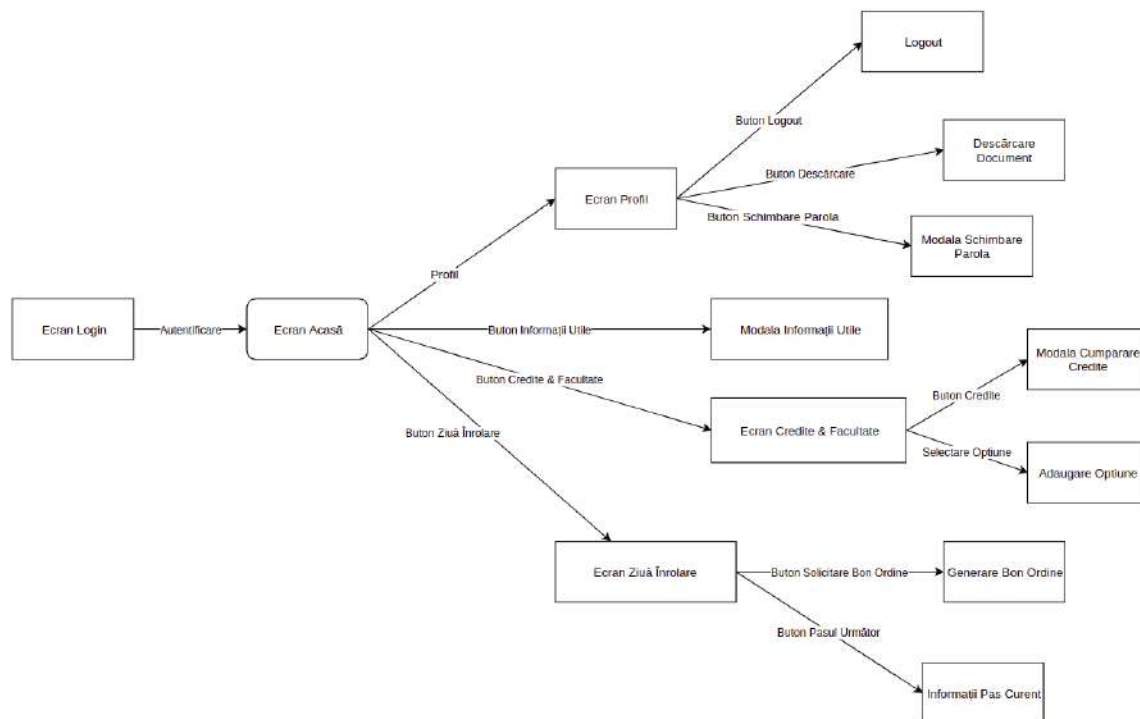


Figura 2.6 – Schemă interfață grafică - Participant

Participantul la admitere sau studentul, așa cum este recunoscut de către sistem, are acces la cea mai complexă interfață, lucru ce poate fi observat cu ușurință prin consultarea *Figurii 2.6*. Navigând prin paginile puse la dispoziție după momentul autentificării, utilizatorul poate să obțină informații atât despre starea curentă a înscrierilor, cât și despre starea personală, fie că vorbim despre timpul estimat de așteptare, consultarea rezultatelor concursului de admitere sau a documentelor generate. Totodată, studentul poate accesa o pagină în care îi vor fi prezentate informații generale despre admitere și despre pașii pe care trebuie să îi urmeze în ziua depunerii dosarului, putând, în plus, să solicite, în cadrul aceluiași meniu, bonul de ordine pe baza căruia va fi identificat de către sistem. Adicional, acesta poate actualiza în orice moment topul personal al facultăților la care dorește să aplice pe baza unor credite ce pot fi achiziționate atât prin intermediul platformei, cât și fizic, cu suportul unui casier.

2.4 Diagrama de componente

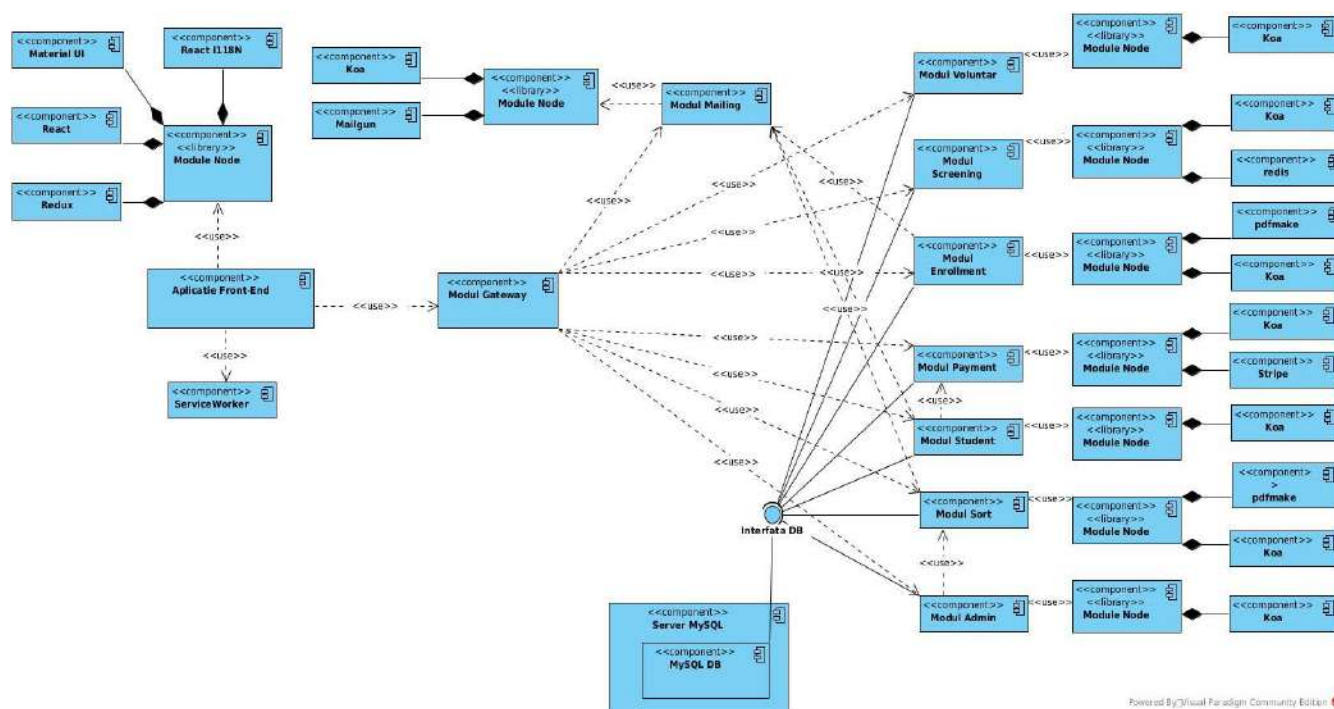


Figura 2.7 – Diagrama de componente

Diagrama de componente reunește toate modulele utilizate de către sistem, surprinzând relațiile de dependență ce apar între acestea și resursele pe care le solicită.

Punctual, diagrama prezintă diferitele subsisteme software ale aplicației și conexiunile ce apar între acestea la momentul execuției. Spre diferență de subcapitolele anterioare, ce au tratat subiectele abordate dintr-o ipostază teoretică, independentă de o implementare concretă, diagrama de componente anunță, pentru prima dată, tehnologiile ce vor fi utilizate și le atribuie, încă înainte de debutul implementării propriu-zise, poziții bine definite în cadrul sistemului, creionând schița unui ecosistem tehnic. Diagrama poate fi consultată în *Figura 2.7*, prezentarea tehnologiilor surprinse de aceasta fiind obiectul central al capitolului următor.

2.5 Diagrama de desfășurare

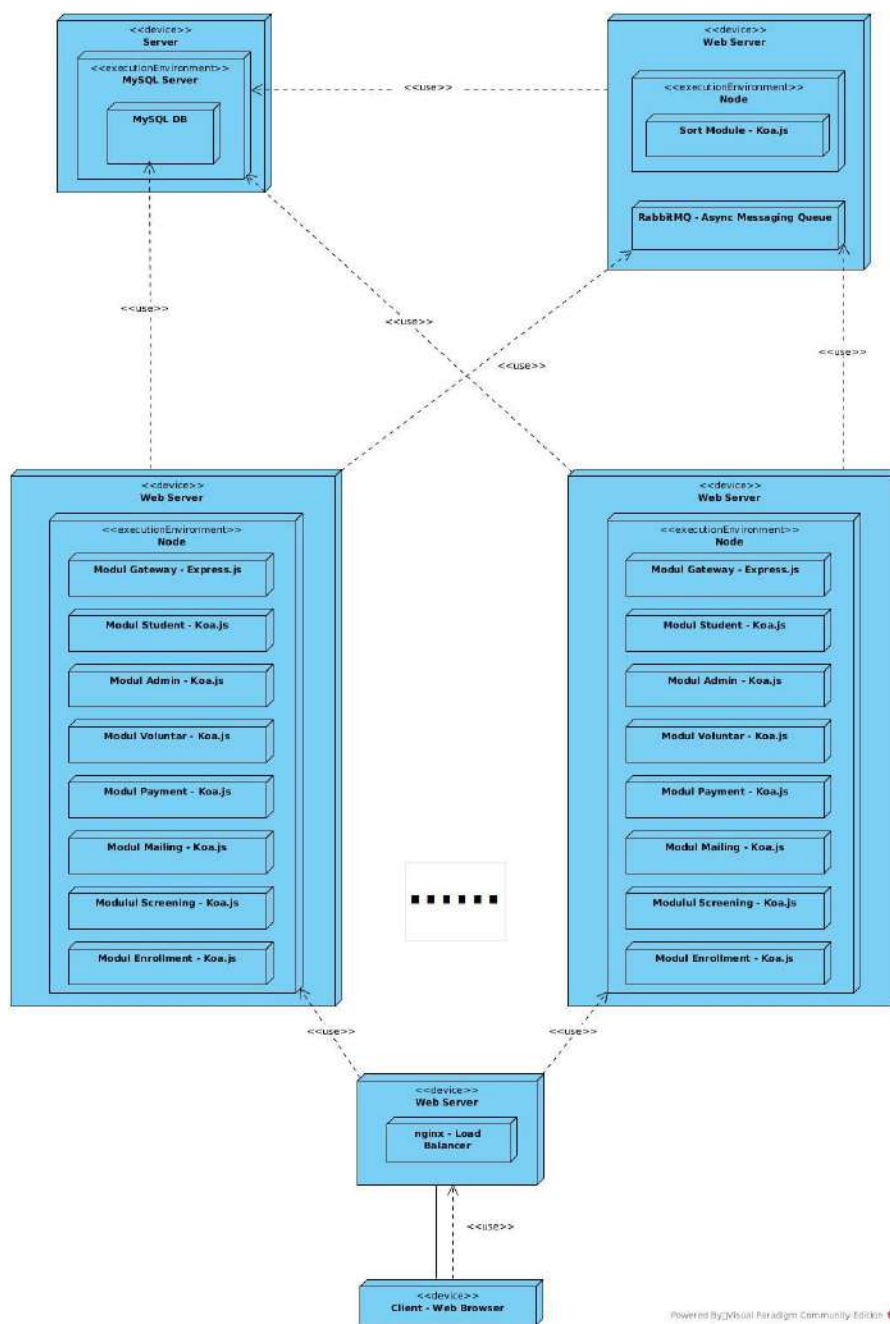


Figura 2.8 – Diagrama de desfășurare

Diagrama de desfășurare, atașată în *Figura 2.8*, înglobează totalitatea dispozitivelor fizice și logice ce interacționează în momentul execuției, precum și a relațiilor ce apar între acestea. Elementele constitutive sunt descrise pe baza unei structuri de incluziune, fiecare sistem individual fiind prezentat de la exterior către interior, fiind realizată gradual tranziția de la mediul hardware la cel software. Această reprezentare surprinde arhitectura generală a sistemului ce joacă un rol important atât în procesul de publicare a aplicației, cât și în cel de scalare.

Capitolul 3 – Etapa de implementare

3.1 Prezentarea tehnologiilor utilizate

Aplicația Flow este dezvoltată cu ajutorul platformei Web și, majoritar, a consacratei arhitecturi client-server. În unele contexte este utilizată *tehnologia WebSockets* [1] pentru a permite o conexiune bidirecțională între părțile comunicante, folosind principiul transmiterii asincrone de mesaje prin intermediul receptării și emiterii de evenimente.

Totodată, fiind o *aplicație progresivă* [2], sunt utilizate tehnologii moderne pe care platforma le pune la dispoziție pentru a reduce distanța dintre programele native, dedicate unui sistem de operare unic, și cele portabile, ce pot fi accesate din orice mediu. Sistemul implementează funcționalități precum trimiterea de notificări persistente, interogarea locației, suportul pentru plățile online, instalarea aplicației în sistem, asemenea unei aplicații dedicate, și salvarea resurselor frecvent utilizate într-o zonă de caching. În plus, este posibilă rularea aplicației în lipsa unei conexiuni active la rețea, un program auxiliar, denumit *Service Worker* [3], suplinind absența acestuia prin returnarea unor resurse salvate, pe baza unor instrucțiuni special dezvoltate. Aceste tehnologii cu ajutorul cărora se transcede aria de funcționalitate a World Wide Web-ului clasic devin, pe zi ce trece, mai bine conturate, mai intuitive și mai pregătite pentru a fi adoptate la scară largă de o industrie care este, la propriu, reprezentantul schimbării și al inovației.

Suportul fundamental pentru implementarea interfețelor grafice îl reprezintă instrumentele standard recomandate de către W3Consortium, HTML5, CSS3 și JavaScript ES6. Totuși, pentru a simplifica procesul și a scădea timpul de dezvoltare am utilizat biblioteci auxiliare bine documentate. Astfel, pentru partea de Front-End am ales să lucrez cu celebra bibliotecă propusă de Facebook, React, datorită conceptelor de care aceasta se folosește și a structurii bazate pe componente ce permite reutilizarea codului deja existent. În continuare, am adăugat plugin-ul Redux ce se ocupă de gestiunea datelor la nivel de aplicație prin utilizarea unui container unic, denumit store, ce poate fi accesat din orice componentă, indiferent de profunzimea la care se plasează în arborele de componente existent.

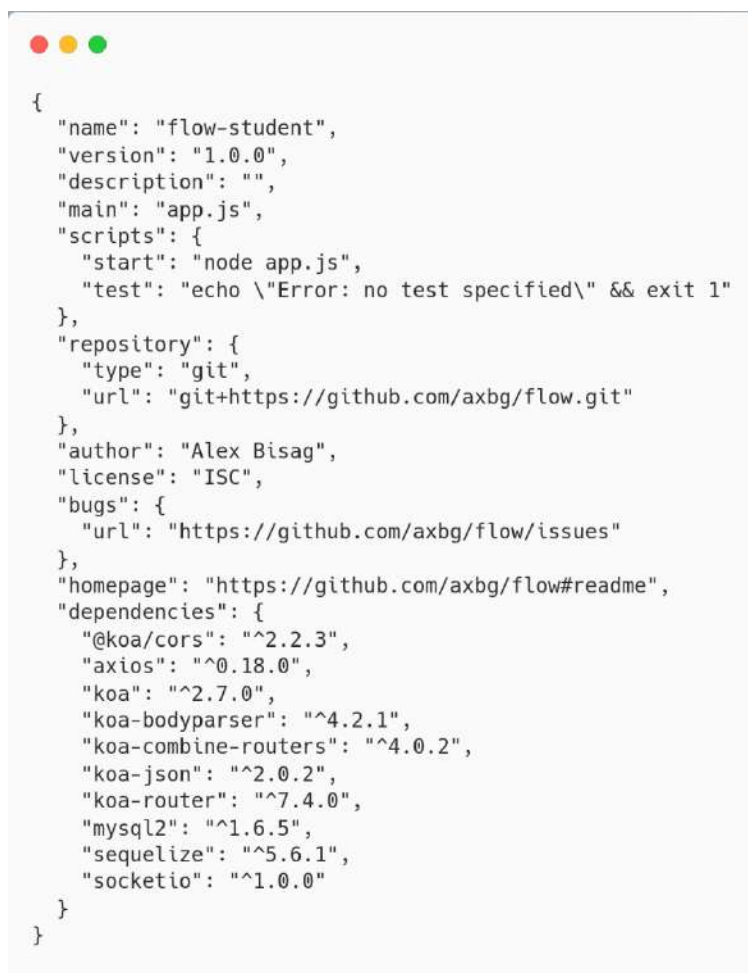
Setul de tehnologii nu putea fi complet fără alegerea unei biblioteci de design. În acest sens, am optat pentru MaterialUI, un auxiliar ce implementează principiile teoretice ale *manifestului cu același nume propus de inginerii de la Google* [4] și care este omniprezent atât pe dispozitivele Android, cât și în cadrul produselor arhicunoscute ale companiei. Suplimentar, am utilizat bibliotecile react-loaders, pentru afișarea unor animații în momentele în care aplicația execută o operație intensă de procesare sau se află într-un schimb de date cu serverul, axios, pentru construirea rapidă de cereri HTTP, google-react-charts, pentru generarea graficelor, și i18n-react, necesară pentru internaționalizarea interfețelor prin utilizarea unor structuri JSON asociate traducerilor ce pot fi alternate de către utilizator prin simpla apăsare a unui buton. Pe lângă toate acestea, demnă de menționat este și existența a două fișiere speciale, manifest.json și service-worker.js, care specifică aspectul, respectiv comportamentul aplicației din punct de vedere al progresivității. Mai precis, definesc setul de pictograme folosit în momentul instalării aplicației în sistem, numele acesteia, tema cromatică, comportamentul offline, strategiile de caching și afișarea notificărilor persistente cu ajutorul sistemului nativ de notificări al dispozitivului în cauză.

Implementarea zonei de Back-End prezintă o complexitate mult mai mare, fiind construită în jurul unui model arhitectural de graniță între arhitectura tradițională, monolitică, și cea modernă, bazată pe microservicii: monolitul modular.

Am ales să urmăresc acest șablon deoarece, spre diferență de o arhitectură complet distribuită, necesită mai puține resurse în momentul publicării și prezintă un flux de dezvoltare mai simplu, iar, contrar unui monolit, poate fi scalat nu doar vertical, ci și orizontal, într-o oarecare măsură.

Punctul comun al tuturor instanțelor îl reprezintă o bază de date unică, situată în afara schemei de scalare. În termeni de specialitate, accesul la baza de date ar putea ajunge, în acest context, un “*bottleneck*” ce ar putea bloca întregul sistem și ar face inutilă implementarea semi-modulară descrisă anterior. În cazul în care o singură instanță a bazei de date devine insuficientă și pune în pericol disponibilitatea sistemului, aceasta va fi replicată și va fi “ascunsă” în spatele unui proxy, traficul fiind direcționat către una dintre instanțele existente în funcție de nivelul de solicitare al acestora.

Această arhitectură mi-a permis ca, în funcție de nevoie, să utilizez tehnologii diferite pentru implementarea logicii executate pe server și să dezvolt fiecare modul în paralel, integrându-l în schema generală abia în momentul în care prezența acestuia a fost necesară. Astfel, majoritatea modulelor au fost realizate utilizând mediul de execuție JavaScript Node.js și *framework-ul* Koa [5], cunoscut pentru gradul înalt de modularitate, stilul elegant de dezvoltare și performanța dovedită în contextul operațiilor IO. Totodată, Koa este privit de către o bună parte a comunității drept un reprezentat al generației următoare de framework-uri ce va crește masiv în popularitate în perioada următoare. Un exemplu de configurație a pachetelor utilizate poate fi consultat în *Figura 3.1*.



```
{
  "name": "flow-student",
  "version": "1.0.0",
  "description": "",
  "main": "app.js",
  "scripts": {
    "start": "node app.js",
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "repository": {
    "type": "git",
    "url": "git+https://github.com/axbg/flow.git"
  },
  "author": "Alex Bisag",
  "license": "ISC",
  "bugs": {
    "url": "https://github.com/axbg/flow/issues"
  },
  "homepage": "https://github.com/axbg/flow#readme",
  "dependencies": {
    "@koa/cors": "^2.2.3",
    "axios": "^0.18.0",
    "koa": "^2.7.0",
    "koa-bodyparser": "^4.2.1",
    "koa-combine-routers": "^4.0.2",
    "koa-json": "^2.0.2",
    "koa-router": "^7.4.0",
    "mysql2": "^1.6.5",
    "sequelize": "^5.6.1",
    "socketio": "^1.0.0"
  }
}
```

Figura 3.1 – Configurarea generală a unui modul

Pe lângă acesta, am utilizat în dezvoltarea unui modul același mediu de execuție împreună cu celebrul framework Express, datorită bibliotecilor specializate ce au fost perfecționate de-a lungul timpului și a experienței facile de diagnosticare a problemelor, existând peste 54000 de probleme soluționate pe platforma StackOverflow.

Ecosistemul acestor tehnologii, format din sute de pachete auxiliare, este gestionat folosind un sistem devenit, încă de la apariție, un adevărat standard în industrie: npm (*Node Package Manager* [6]), ce deține peste un milion de pachete dedicate mediului Node.js.

Pentru nivelul de stocare a datelor, am utilizat sistemul de gestiune MySQL datorită experienței personale, a nativității operațiunilor necesare pentru replicare și a popularității, orice problemă care ar fi putut fi apărut pe parcursul dezvoltării având, în mod aproape cert, o rezolvare ușor de adaptat.

Sistemul de autentificare este unul simplu, fără sesiuni stocate pe server, lucru aproape necesar în contextul unui mediu modular, și folosește principiile JWT (*JavaScript Web Tokens* [7]), materializate cu ajutorul bibliotecii *javascriptwebtoken*, fiecare utilizator primind, în momentul autentificării, un token dinamic, criptat, ce conține informații proprii, menite să realizeze identificarea acestuia. Token-ul este trimis către server la fiecare cerere în câmpul Authorization existent în zona de Header și este decriptat, serverul acceptând sau respingând cererea, în funcție de rezultat.

Această operațiune de confirmare a identității se realizează într-un punct unic, în modulul Gateway al fiecărei instanțe, ce acționează, grație bibliotecii *http-proxy-middleware*, asemeni unui proxy, reprezentând, totodată, unica expunere către Internet a sistemului. Înainte de a realiza redirectarea către resursa solicitată ce poate fi oferită de către unul dintre modulele interne, cererea HTTP este rescrisă, în zona de Header fiind incluse, sub niște chei cunoscute în prealabil de către toate instanțele, informațiile relevante pentru identificarea utilizatorului. Modulele vor intercepta cererea și vor extrage informațiile de autentificare, de care se va folosi, ulterior pentru a putea construi un răspuns adecvat.

Pentru comunicarea asincronă am utilizat două tehnologii distincte, în funcție de natura procesului tratat. În primul context, atunci când discutăm de comunicarea asincronă, bidirecțională, între client și server, am optat pentru utilizarea standardului WebSockets și, în particular, a bibliotecii *socket.io* ce îmbracă operațiile asincrone sub forma unui API intuitiv, ce respectă principiile programării JavaScript. În cel de-al doilea context, am utilizat o tehnologie derivată din Redis, *Redis Messaging Queue* [8], ce acționează în momentul comunicării între două module din zona de server, indiferent de mașinile fizice pe care rulează sub forma de proces. Cu ajutorul acestei implementări, schimbările determinate de către o operație pot fi propagate în întregul sistem, fiecare instanță ce prezintă un interes pentru datele modificate putând reacționa, amintind, în acest sens, de implementarea standard a modelului de design Observer.

Un dezavantaj inherent al arhitecturilor distribuite sau semi-distribuite îl reprezintă volumul mare de operații ce trebuie făcute în momentul publicării aplicației, fiecare modul putând avea un comportament diferit, un set diferit de parametri necesari și o întreagă serie de configurări necesare. În acest sens, pentru a simplifica și, chiar, elimina din schema publicării necesitatea configurărilor individuale, am ales să utilizez Docker pentru construirea unei imagini complete, ce va putea să reproducă, prin câteva instrucțiuni, întreaga infrastructură necesară pentru introducerea unei noi instanțe. Astfel, imaginile se vor instanția sub formă de containere, incluzând toate modulele ce se pretează replicării pe care le va configura pe baza unui fișier de proprietăți ce va fi completat, în prealabil. Suplimentar, am optat pentru crearea unui script Bash care să gestioneze operațiile necesare

punerii în funcțiune a containerului Docker, schema de publicare a întregului sistem fiind, deci, redusă la rularea unui script în linia de comandă a mașinii.

În ceea ce privește expunerea instanțelor către Internet am optat în a plasa toate modulele accesibile în spatele unor servere web nginx, *configurate sub formă de proxy invers* [9], atât datorită robusteții acestora și a ușurinței cu care pot fi optimizate, cât și a sistemului elaborat de logging ce păstrează, sub formă de istoric, fiecare cerere primită de către o anumită instanță. Fiind unul dintre cele mai populare servere web, nginx beneficiază de multiple module și aplicații auxiliare dezvoltate independent, lucru ce s-a dovedit util în momentul adăugării certificatelor SSL pe care le-am generat cu ajutorul platformei Let's Encrypt. Pe lângă gratuitatea completă, Let's Encrypt pune la dispoziția dezvoltatorilor o aplicație de tip consolă denumită *Certbot* [10] ce simplifică procesul, automatizând fiecare pas.

În ciuda intenției inițiale de a dezvolta toate funcționalitățile de la zero, unele dintre acestea s-au dovedit fie prea complexe, depășindu-mi cunoștințele actuale, fie prea costisitoare din punct de vedere temporal și am optat pentru externalizarea lor, utilizând resurse sau servicii puse la dispoziție de către dezvoltatori specializați.

Primul serviciu de acest gen este reprezentat de Stripe, o platformă ce pune la dispoziție un set de API-uri REST și biblioteci specializate pentru limbajele de programare majore cu ajutorul cărora pot fi realizate și monitorizate plățile online. Mai mult decât atât, infrastructura este foarte bine documentată, fiecare operație disponibilă beneficiind de exemple bine descrise ce pot fi replicate cu ajutorul unor date de test generale, furnizate în platformă. Am ales să utilizez Stripe deoarece este un serviciu foarte popular, foarte ușor de folosit, ce nu prezintă restricții de platformă ca în cazul API-ului nativ Google Payments ce a reprezentat, într-adevăr, prima mea opțiune, dar la care am fost nevoit să renunț din cauza lipsei de suport manifestată, în special, pe dispozitivele mobile. Din punct de vedere al gestiunii, Stripe pune la dispoziție prin intermediul unei aplicații web o modalitate de contorizare a tuturor tranzacțiilor realizate de către sistem și de resetare a acestora, în cazul apariției unor situații problematice.

Al doilea serviciu integrat este Mailgun ce acoperă zona de poștă electronică, suplinind infrastructura necesară pentru susținerea unui astfel de sistem și oferind o disponibilitate net superioară unei soluții dezvoltate de la zero. Principalul motiv pentru care am ales să utilizez setul de servicii expus de către Mailgun îl reprezintă consistența inter-platforme, lucru ce mi-a ridicat mari probleme în trecut, în dezvoltarea altor sisteme ce necesitau o astfel de funcționalitate. Totodată, Mailgun se pretează excelent integrării într-o platformă distribuită ce poate genera un număr foarte mare de cereri, deoarece, intern, folosește o coadă asincronă de mesaje ce va stoca toate e-mail-urile ce se doresc a fi trimise și le va expedia gradual, pe măsură ce serverele implicate se eliberează. Un alt avantaj îl reprezintă aplicația web ce prezintă starea fiecărui e-mail transmis, punând la dispoziția dezvoltatorului un set de instrumente pentru remedierea erorilor.

Ultimul modul extern ce are o importanță majoră în cadrul sistemului este reprezentat, de această dată, de o bibliotecă JavaScript și nu de un serviciu. Pdfmake expune un set de metode cu ajutorul cărora vor fi generate și personalizat documentele de confirmare a înscrierii unui participant. Principalul motiv pentru care am ales utilizarea acestei biblioteci îl reprezintă, înainte de toate, documentația bogată formulată de dezvoltatori, ce prezintă toate opțiunile disponibile și ghidează, pas cu pas, procesul de integrare al bibliotecii atât în aplicațiile ce rulează cu ajutorul platformei Node.js, cât și în browser.

Pasul final al implementării propriu-zise este reprezentat de împachetarea sistemului și de pregătirea procesului de publicare. Fiind un sistem dezvoltat pe baza tehnologiilor web, scenariul, este, oarecum, simplu, fiind necesară distribuirea modulelor back-end pe o suită de servere

specializate, cu o anumită configurație, ce vor comunica, în mod uni sau bidirecțional, cu clientul front-end. Însă, cum aplicația este una progresivă și prezintă caracteristici particulare menite să ofere utilizatorului final o experiență îmbunătățită, am considerat că ștergerea barierei dintre nativ și portabil reprezintă un obiectiv important ce merită un efort suplimentar. În acest sens, m-am documentat cu privire la o nouă formă de publicare a aplicațiilor progresive apărută în decursul ultimelor luni și susținută intens de către compania mamă Google: conceptul de *Trusted Web Activities* [11]. Pe baza acestui set de tehnologii distincte ce lucrează împreună, aplicațiile progresive pot fi transformate în aplicații native destinate sistemului de operare Android, putând fi, de asemenea, publicate în cadrul magazinului oficial, Google Play Store. Deși tehnologia este disponibilă de aproximativ un an, procedura de generare a fișierului sursă .apk este, încă, destul de complexă, fiind necesară utilizarea mediului de dezvoltare Android Studio pentru compilarea surselor și pentru producerea artefactelor necesare.

3.2 Arhitectură & Scalare

După cum a fost menționat în capitolul anterior, arhitectura ce a servit drept model implementării sistemului este cea *de monolit modular* [12], o arhitectură de graniță între abordarea clasică și cea modernă. Din dorința de a motiva alegerea făcută, am analizat avantajele generale și dezavantajele monolitului modular prin comparație cu arhitectura bazată pe microservicii, rezultatul obținut confirmând că, în contextul de față, această abordare a determinat existența unui raport efort-rezultat subunitar.

Avantaje

- poate fi scalată vertical, dar și orizontal, într-o oarecare măsură
- permite existența unei surse unice de adevăr pentru toate modulele, datele putând fi consultate direct
- permite replicarea bazei de date pentru a evita scăderea de performanță din cauza acesteia, în cazul unui număr foarte mare de conexiuni
- fiecare modul are propria logică și poate fi dezvoltat independent
- prin comparație cu arhitectura bazată pe microservicii, necesită mai puține operații de rețea, iar cererile inter-modulare aflate în interiorul instanțelor replicate pot fi făcute cu ajutorul unei rețele interne
- pentru asigurarea unei publicări complete sunt necesare mai puține mașini fizice
- publicarea poate fi gestionată manual, fără utilizarea unor instrumente complexe de orchestrare (ex: Kubernetes, Ansible)
- modulele pot fi transformate, ulterior, în microservicii, în contextul în care cazurile de utilizare solicită acest lucru

Dezavantaje

- pentru publicarea modificărilor realizate la nivelul unui modul, toată aplicația trebuie repornită
- modulele nu sunt proprietare asupra datelor, acestea putând fi modificate de către alte module
- scalarea orizontală necesită o nouă instanță a întregului sistem, nu doar a unui modul independent

- o instanță se poate supraîncărca răspunzând cererilor unui singur modul, fapt ce nu va îngreuna balansarea de sarcini pentru toate celelalte module, aceasta nemaifiind egală la nivelul întregului sistem

Luând toate aceste lucruri în considerare, am realizat o schemă de scalare a sistemului care surprinde elemente comune cu diagrama de desfășurare, dar care, spre diferență de aceasta, prezintă într-un mod mai puțin formal toate posibilitățile existente ce pot sau nu să fie aplicate în funcție de nivelul de cereri anticipat.

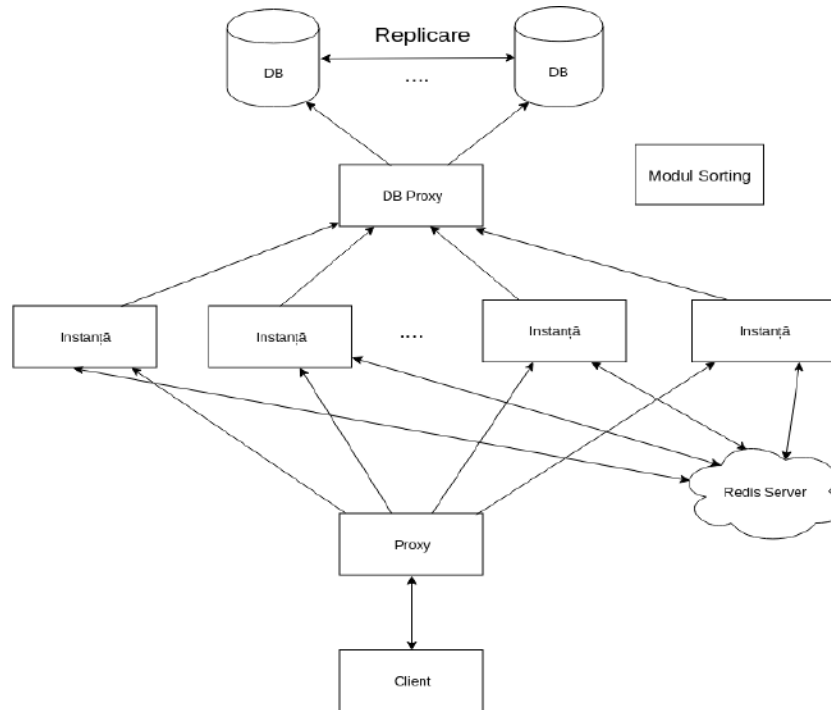


Figura 3.2 – Schema de scalare

3.3 Prezentarea modulelor

Modulele dezvoltate și interacțiunile ce apar între acestea alcătuiesc sistemul informatic de gestiune, fiecare unitate având un rol specializat, existând, pe de-o parte, module majore, ce se mulează pe cazul de utilizare al unui anumit tip de utilizator, și module utilitare ce expun servicii care pot fi invocate, evitând astfel replicarea codului și încălcarea principiului DRY. Modulele sunt slab cuplate, acestea comunicând prin redirectări realizate de modulul Gateway, astfel încât, într-o etapă viitoare a dezvoltării, printr-un număr redus de modificări, pot fi transformate în microservicii.

Modulul Gateway

- singurul serviciu expus pe Internet
- acționează ca un proxy, redirectionând toate cererile către modulele interne
- realizează procesul de autentificare, utilizând, în acest sens, conceptul de Javascript Web Tokens (JWT)
- asigură funcționalitatea de logare, generând JWT personalizate

Modulul Mailing

- este conectat la serviciul Mailgun
- primește cereri din partea tuturor celorlalte module pe care, urmând o structură bine determinată, le transformă în e-mail-uri ce vor fi expediate prin API-ul Mailgun

Modulul Student

- permite înregistrarea unui participant în sistem
- permite vizualizarea profilului și, implicit, a datelor înregistrate în sistem
- permite schimbarea parolei curente
- permite achiziționarea creditelor necesare pentru selecția facultății prin intermediul serviciului Stripe
- permite selectarea opțiunilor în ordinea preferințelor
- permite generarea unui bon de ordine, validând locația curentă a participantului prin comparație cu aria permisă de către fiecare facultate
- permite descărcarea documentelor generate în momentul confirmării înscrierii
- afișează numărul curent de ordine pentru facultatea aflată în topul opțiunilor
- afișează timpul personal, estimat de așteptare

Modulul Screening

- afișează bonul de ordine curent pentru o facultate selectată
- utilizează tehnologia WebSockets pentru a permite reîncărcarea instantanee a informațiilor actualizate
- se conectează la coada asincronă Redis, astfel încât, indiferent de instanța fizică în care se află, să fie înștiințat când are loc o actualizare a datelor
- afișează timpului estimat de intrare pentru următorul grup
- afișează o fereastră modală ce prezintă o colecție link-uri utile

Modulul Volunteer

- permite introducerea de fluxuri de studenți în sistem
- se conectează la coada asincronă Redis, notificând, indirect, modulul Screening atunci când sunt adăugate date noi
- permite trimiterea de notificări persistente următorului grup de studenți ce trebuie să se prezinte la intrare
- permite trimiterea cererilor de schimbare a poziției către administratori
- permite căutarea studenților în sistem
- permite generarea unui bon de ordine pentru un student
- recepționează notificări persistente în momentul în care cererea de schimbare a poziției este rezolvată

Modulul Admin

- calculează fluxul total de studenți pentru fiecare oră
- permite pornirea sau oprirea sesiunii de înscrieri

- poate vizualiza cererile de schimbare a pozițiilor trimise de către voluntari și le poate soluționa
- poate schimba poziția curentă a voluntarilor
- poate determina începerea procedurii de sortare fie sub formă de test, fie sub formă definitivă
- poate crea utilizatori noi
- poate accesa listele de repartizare produse finalul procesului de sortare

Modulul Sort

- singurul modul ce nu necesită replicare, datorită numărului redus de cereri pe care le va recepționa și a intensității operațiilor întreprinse ce nu pot fi distribuite atât de ușor
- implementează un algoritm de sortare a candidaților în funcție de o logică particulară, ce ține cont de criteriile de sortare
- generează liste de repartizare în format PDF pentru fiecare profil

Modulul Payment

- permite fiecărui student să cumpere credite pe care le asociază profilului său după efectuarea plății
- permite fiecărui casier adăugarea/scăderea numărului de credite asociate unui student

Modulul Enrollment

- permite căutarea unui student și accesarea profilului
- permite modificarea profilului unui student
- permite setarea sau resetarea opțiunilor unui student în baza creditelor pe care acesta le deține
- permite finalizarea procesului de înscriere a unui student
- generează documente de confirmare în format PDF la finalizarea procesului de înscriere

3.4 Publicare

Pe tot parcursul dezvoltării am tratat cu un grad ridicat de atenție procesul de publicare, încercând, atât prin structura aplicației, cât și prin tehnologiile auxiliare utilizate, să automatizez un procent cât mai mare din numărul de operații necesare, astfel încât sistemul să poată fi instalat cu ușurință de către un operator de sistem, fără a solicita cunoștințe avansate de programare sau o înțelegere profundă a modului de funcționare.

Pentru a fi sigur de fiabilitatea instrumentelor utilizate, am reprodus o parte din pașii de publicare ce trebuie urmați în cazul unei lansări în producție. Pentru infrastructură, am ales serviciul Cloud Platform pus la dispoziție de către Google datorită fondurilor de testare pe care aceștia le oferă, dar și a ușurinței cu care mașinile virtuale pot fi create și configurate. Din punct de vedere al sistemului de operare, am optat pentru Ubuntu 18.04 LTS, varianta server, principalul motiv fiind popularitatea uriașă, buna documentare și experiența personală de utilizare

<input type="checkbox"/> Name ^	Zone	Recommendation	In use by	Internal IP	External IP	Connect
<input type="checkbox"/> flow-database	europa-west4-a			10.164.0.8 (nic0)	35.204.250.24	SSH ▾ ⋮
<input type="checkbox"/> flow-instance	europa-west4-a			10.164.0.4 (nic0)	35.204.163.215	SSH ▾ ⋮
<input type="checkbox"/> flow-proxy	europa-west4-a			10.164.0.6 (nic0)	34.90.179.149	SSH ▾ ⋮
<input type="checkbox"/> flow-redis	europa-west4-a			10.164.0.7 (nic0)	34.90.5.118	SSH ▾ ⋮
<input type="checkbox"/> flow-sort	europa-west4-a			10.164.0.5 (nic0)	35.204.201.226	SSH ▾ ⋮

Figura 3.3 – Serverele utilizate pentru publicarea aplicației

Printr-o comparație a arhitecturii potențiale, expusă în cadrul subcapitolelor anterioare, cu numărul de mașini instanțiate, se poate observa că, pentru exemplificarea procesului de publicare, am ales o implementare minimalistă, ce se pretează unui mediu de test. Utilizarea unui număr mare de servere și a schemei de scalare este costisitoare atât din punct de vedere al resurselor folosite, cât și din punct de vedere al timpului investit, iar obiectivele vizate, testarea instrumentelor auxiliare, ar fi atinse în aceeași proporție.

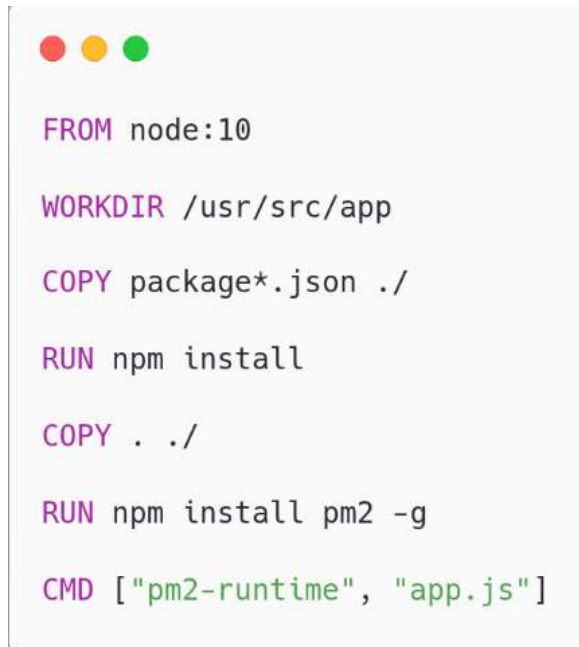
Mașinile flow-database și flow-sort sunt cele care găzduiesc și expun sistemul de gestiune MySQL, respectiv modulul de sortare, și sunt configurate în prealabil, acestea nefăcând parte din schema de scalare și nefiind, așadar, incluse în fluxul de publicare automatizat.

Mașina flow-redis găzduiește serverul Redis ce va fi utilizat pentru asigurarea comunicării asincrone între module, acesta transmitând mesajele publicate de către un modul către toate instanțele ce așteaptă propagarea unui eveniment specific.

Mașina flow-proxy reprezintă punctul de contact dintre client și sistem, acesta redirecționând cererile către instanța cea mai puțin utilizată, asigurând, astfel, o funcționalitate de load-balancing mai mult decât necesară în cazul utilizării intense. Totodată, în contextul testării, aplicația front-end poate fi găzduită pe acest server, reducând, astfel, numărul de unități hardware utilizate.

Ultima instanță fizică, mașina flow-instance, reprezintă mediul în care vor fi publicate toate modulele incluse în grila de automatizare, reprezentând punctul din sistem ce poate fi replicat pentru a disipa orizontal numărul de cereri primite. Kit-ul de publicare conține un număr total de cinci fișiere utilizate pentru configurarea mașinii fizice, publicarea containerelor Docker, configurarea mediului de execuție Node.js și expunerea modulului Gateway către Internet prin implementarea unui mecanism de proxy invers cu ajutorul serverului web nginx. Toate aceste fișiere au fost încărcate într-un depozit Github Gist de unde pot fi descărcate cu ușurință pe orice mașină.

Pentru containerizarea modulelor am creat un script clasic, Dockerfile, ce produce o imagine Docker care descrie un set de instrucțiuni de configurare a containerului rezultat la rularea acesteia

A terminal window with a light gray background and three colored window control buttons (red, yellow, green) in the top left corner. The text is a Dockerfile configuration for a Node.js application, with each command on a new line and color-coded: FROM, WORKDIR, COPY, RUN, and CMD are in purple, while the file names and paths are in green.

```
FROM node:10

WORKDIR /usr/src/app

COPY package*.json ./

RUN npm install

COPY . ./

RUN npm install pm2 -g

CMD ["pm2-runtime", "app.js"]
```

Figura 3.4 – Configurarea imaginii Docker

Cum toate modulele împart același mediu de execuție, a fost suficientă compunerea unui singur script, ce poate fi observat în *Figura 3.4*. Pe scurt, în momentul instanțierii, fiecare container va fi dezvoltat incremental, folosind ca și bază imaginea standard Node.js, pe care o va utiliza pentru a executa aplicația. Pentru a putea observa cu ușurință starea de sănătate a fiecărei aplicații, am optat pentru rularea acestora cu ajutorul utilitarului pm2, ce asigură resetarea modulelor în cazul apariției erorilor și salvarea automată a tuturor problemelor întâmpinate. Toate imaginile construite au fost publicate pe DockerHub, o platformă dedicată stocării imaginilor Docker, de unde vor fi descărcate automat în momentul utilizării script-ului de publicare.

Odată create, imaginile pot fi instanțiate, sub formă de containere, prin executarea unor comenzi specifice. Totuși, dat fiind numărul destul de mare de imagini rezultate, raportul față de numărul de module fiind de unu la unu, am optat pentru utilizarea unui instrument de orchestrare nativ, denumit Docker Compose, ce permite punerea simultană în execuție a mai multor containere prin intermediul unui script yaml de configurare.

Pe lângă identificarea, descărcarea și instanțierea imaginilor, script-ul creează o rețea internă prin care containerele pot comunica și transmite mediului de execuție din fiecare container un fișier de configurare ce permite sistemului să se conecteze la baza de date și la serverul Redis. Acest fișier trebuie completat de către operator înainte de începerea procesului.

Asemeni fișierului de configurare, kit-ul conține încă un fișier auxiliar ce descrie setările care trebuie aplicate serverului web nginx pentru a permite redirectarea cererilor de pe portul 443 pe portul asociat containerului ce ține în execuție modulul Gateway, realizând, astfel, expunerea pe Internet a sistemului Flow.

Ultimul fișier este reprezentat de scriptul Bash care se ocupă de orchestrarea tuturor operațiilor necesare pentru pregătirea mașinii și pornirea, efectivă, a containerelor.

```
#update apt
sudo apt-get update

#install docker
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs)
stable"
sudo apt-get update
sudo apt-get install -y docker-ce

#install docker compose
sudo curl -L https://github.com/docker/compose/releases/download/1.18.0/docker-compose-`uname -s`-
`uname -m` -o /usr/local/bin/docker-compose
sudo mv /usr/local/bin/docker-compose /usr/bin/docker-compose
sudo chmod +x /usr/bin/docker-compose

#deploy instances
sudo docker-compose up -d

#installing and configuring nginx
sudo apt-get --assume-yes install nginx
sudo cp nginx /etc/nginx/sites-available/default
sudo service nginx restart

clear
echo "Flow instances deployed successfully"
```

Figura 3.5 – Scriptul de configurare și publicare a modulelor

Acesta script, atașat în *Figura 3.5*, parcurge mai mulți pași, începând prin a actualiza referințele de actualizare ale sistemului, permițând accesul la noi versiuni ale programelor instalate, urmând ca, gradual, să instaleze și să pregătească toate instrumentele necesare: Docker, Docker-Compose, nginx și să execute scriptul ce va realiza instalarea sistemului informatic prin instanțierea separată a fiecărui modul implicat.

Capitolul 4 – Prezentarea aplicației

Conform capitolului dedicat etapei de implementare, aplicația Flow include cinci tipuri distincte de utilizatori ce dețin atribuții și instrumente specifice.

Din cauza acestui fapt realizarea unei prezentări care să urmeze un flux exact ar ridica destul de multe probleme, mai ales în formă scrisă. Astfel, pentru simplificarea acestei sarcini și pentru păstrarea unei cursivități în discurs, am ajuns la concluzia că cea mai bună formulă de prezentare este aceea de a parcurge, în mod natural, toate scenariile destinate utilizatorilor finali, dinspre exterior spre interior, respectiv din ipostaza participantului la admitere, înspre cea a personalului auxiliar, cu toate formele în care acesta este reprezentat în cadrul sistemului.

Totodată, fiind vorba despre o aplicație progresivă, trebuie menționat faptul că aceasta se adaptează dimensiunii ecranului pe care rulează, având, deci, atât o formă dedicată terminalelor de tip desktop, cât și una specifică rezoluțiilor terminalelor mobile. Totuși, fiecare rol implică un grad diferit de mobilitate, determinând utilizarea majoritară a aplicației într-una din cele două opțiuni de vizualizare. Scenariile descrise în continuare țin cont de acest factor, prezentând, în funcție de caz, forma cea mai bine adaptată la tipul de utilizator vizat.

4.1 Scenariul Student

Primul scenariu tratează interacțiunea dintre participantul la admitere, echivalentul entității Student, și aplicația de gestiune. În mod firesc în cazul primei interacțiuni între un utilizator și un sistem informatic, prima etapă este cea de înregistrare, participantul fiind îndrumat, prin intermediul unui formular etapizat, să introducă datele personale relevante.

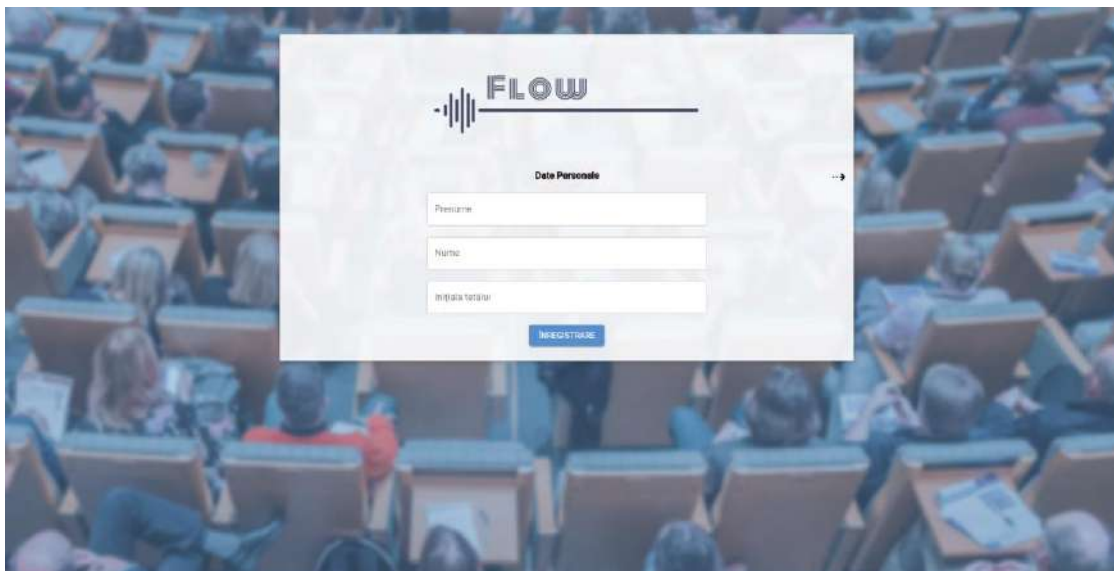


Figura 4.1 – Ecranul de înregistrare

Câmpurile formularului sunt grupate în funcție de categoria datelor colectate, fiind prezente, astfel, secțiunile de date personale, date de contact, date oficiale, foarte sensibile, și datele de concurs ce vor fi utilizate pentru evaluarea candidatului și repartizarea acestuia. Ulterior înregistrării,

participantul va primi un e-mail de confirmare ce va conține credențialele cu ajutorul cărora va putea accesa sistemul.

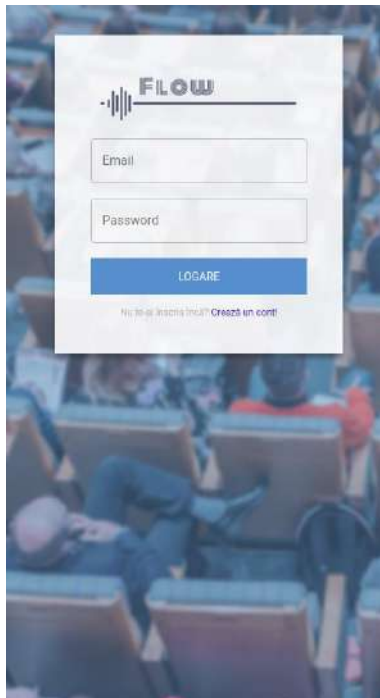


Figura 4.2 – Ecranul de autentificare

Ecranul de logare, ce poate fi observat în *Figura 4.2*, este, însă, un portal comun atât pentru viitorii studenți, cât și pentru personalul organizator, acesta filtrând datele și redirecționând utilizatorul către o interfață specializată în funcție de datele de conectare introduse.



Figura 4.3 – Ecranul de înscriere

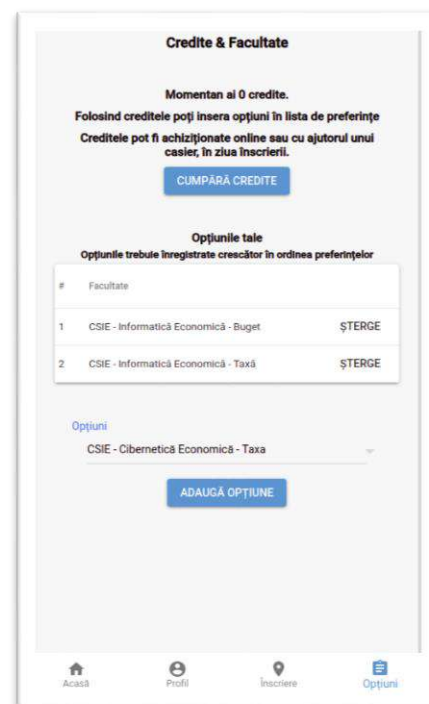


Figura 4.4 – Ecranul de selecție a opțiunilor

În zilele anterioare etapei de depunere a dosarelor, utilizatorul poate accesa informații generale despre pașii pe care trebuie să îi urmeze pentru a finaliza procesul de admitere, ce sunt afișate conform *Figurii 4.3*, și poate achiziționa credite cu ajutorul cărora să își seteze opțiunile de repartizare în ordinea preferințelor personale, așa cum se poate observa în *Figura 4.4*.

În ziua depunerii dosarului, participantul va solicita un bon, utilizând butonul de generare din ecranul Înscriere. O cerere va fi emisă și, în funcție de locația curentă a terminalului, un bon de ordine va fi generat.

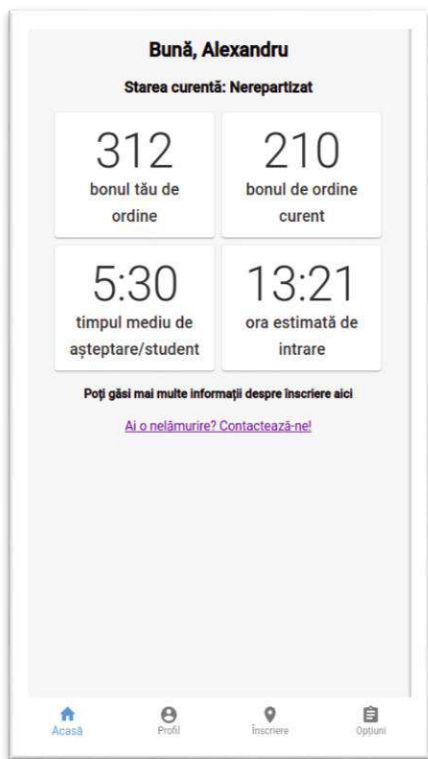


Figura 4.5 –Ecranul Acasă cu informații generale

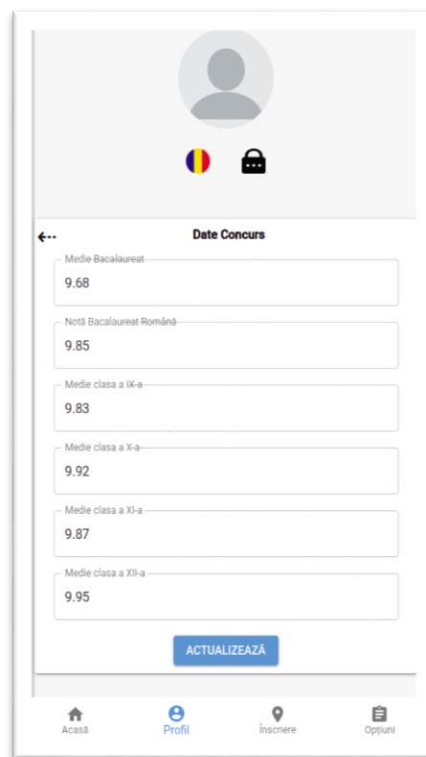


Figura 4.6 –Ecranul Profil

Pe baza acestui bon de ordine, fiecare utilizator va avea acces la un set de date personalizat în funcție de facultatea aflată în topul listei de preferințe și de numărul total de candidați prezenți, pe care le poate consulta în ecranul Acasă, ce poate fi observat în *Figura 4.5*. Tot prin intermediul acestui ecran, utilizatorul va fi înștiințat, pe parcursul etapelor de repartizare, despre situația curentă. Suplimentar, acesta va primi email-uri de informare la finalizarea fiecărei etape de repartizare.

Ultimul ecran destinat candidaților este ecranul Profil, ce permite corectarea și actualizarea datelor din momentul înregistrării până în momentul depunerii dosarului, precum și alternarea limbii în care meniurile sunt afișate, în prezent fiind suportate limbile română și engleză. Totodată, acest ecran permite deschiderea unei ferestre modale prin intermediul căreia utilizatorul își poate schimba parola generată de către sistem, acesta fiind un pas ce trebuie realizat, obligatoriu, după prima logare.

4.2 Scenariul Voluntar

Primul tip de utilizator cu care un participant interacționează este voluntarul care, asemenea acestuia, ocupă o poziție mobilă, putând fi repartizat în diferite puncte cheie atât în interiorul clădirii în care are loc procesul de admitere, cât și în afara acesteia. Principala sarcină a voluntarului este aceea de a gestiona fluxurile de candidați, controlând, în acest sens, modulul Flux, ce se ocupă cu calcularea statistică a timpilor de intrare a fiecărui participant, calcularea timpului mediu de așteptare și trimiterea de notificări informative.

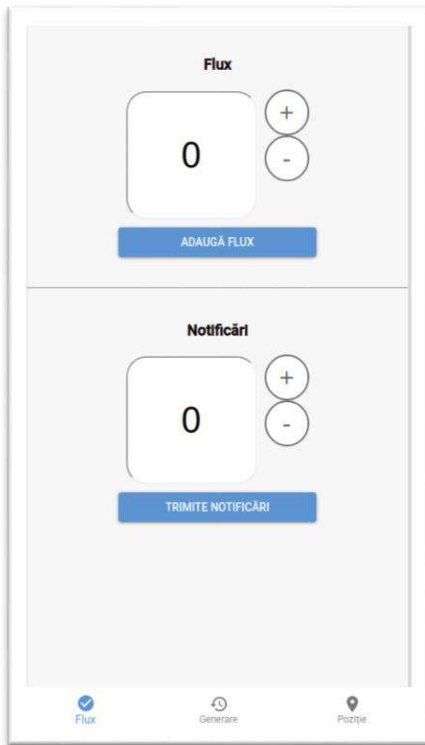


Figura 4.7 – Ecranul de control al modulului Flux

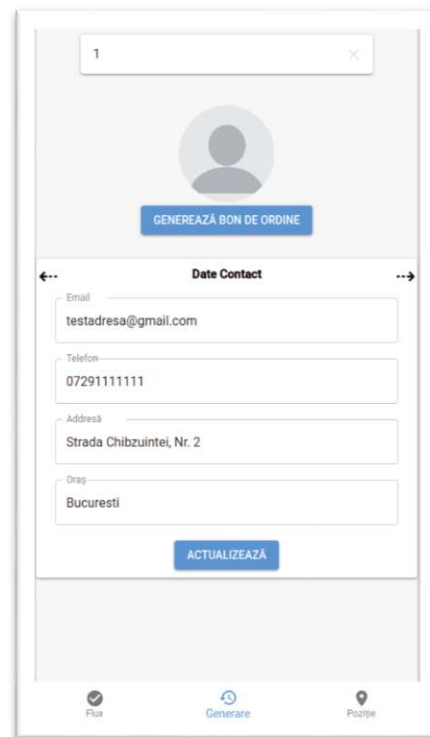


Figura 4.8 – Ecranul de generare a bonului de ordine

Interfața de control a modulului Flux, atașat în *Figura 4.7*, permite voluntarilor să trimită facil cereri către acesta, fiind observate, în acest sens, cele două structuri de inserare similare, plasate pe ecran astfel încât să fie ușor de recunoscut și utilizat. Totodată, fiind un proces ireversibil, butonul de trimitere a notificărilor va determina apariția unui mesaj de confirmare ce va atrage atenția utilizatorului în cazul unei erori.

Deși aplicația a fost testată în diferite rânduri pe o gamă largă de terminale, posibilitatea apariției unor erori este încă prezentă, lucru, de altfel, comun în rândul produselor software aflate la prima versiune. În acest sens, am evaluat toate punctele critice care, în condițiile apariției unei erori, ar putea afecta buna desfășurare a procesului de admitere și am considerat că cel mai important dintre acestea este reprezentat de etapa generării unui bon de ordine. În acest sens, cu intenția de a oferi o alternativă în cazul unor condiții neprevăzute, voluntarii, în special cei ce ocupă o poziție aflată în proximitatea intrării, pot căuta, pe baza CNP-ului, un candidat și îi pot acorda un bon de ordine. Totodată, funcționalitatea de căutare permite identificarea participanților pe baza unui bon de ordine, în contextul în care este necesară regenerarea acestora. *Figura 4.8* ilustrează ecranul Generare ce include toate funcționalitățile menționate mai sus.

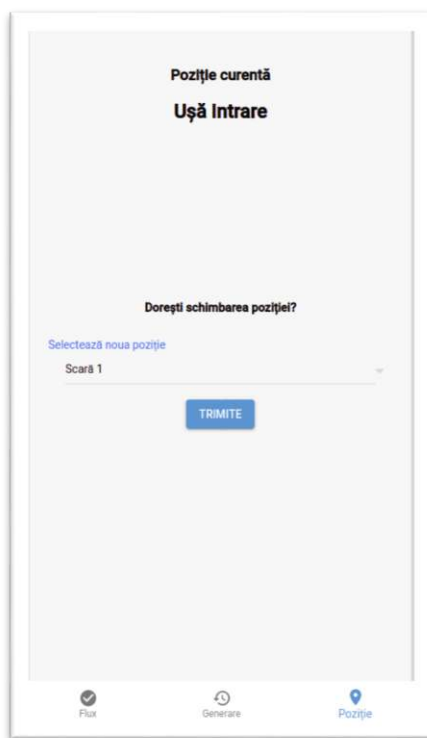


Figura 4.9 – Ecranul de solicitare a schimbării poziției

Ultima funcționalitate destinată voluntarilor, anume posibilitatea de generare a cererilor de schimbare a poziției, are ca scop implicarea directă a acestora în procesul de gestiune a personalului și reducerea timpilor morți de tranziție dintre posturi.

Pe baza cererilor, administratorii își pot face o imagine mai clară despre modul în care decurge procesul, identificând zonele ce necesită suplimentarea personalului și cunoscând, în orice moment, poziția fiecărei persoane implicate.

4.3 Scenariul Casier

Rolul casierului este acela de a oferi suport direct participanților prin gestionarea procesului fizic de achiziționare a creditelor. Conform interfeței din *Figura 4.10*, acest tip de utilizator are la dispoziție un ecran de căutare a candidaților după numărul de ordine sau CNP. După identificare, casierul poate confirma identitatea participantului prin consultarea profilului și poate adăuga un număr de credite în sistem echivalent sumei plătite.

Totodată, casierul, poate fi implicat în procesul de plată a taxei de școlarizare ce se desfășoară ulterior depunerii dosarului, din intermediul aceluiași panou putând marca persoanele ce au achitat suma necesară.

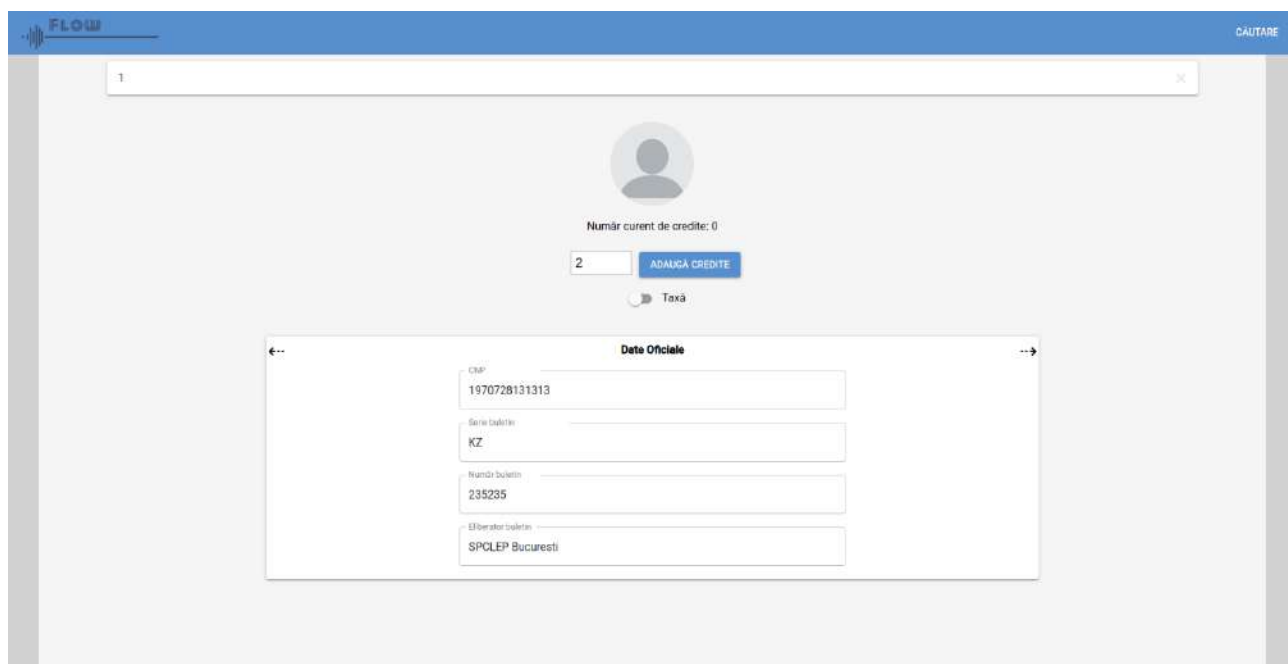


Figura 4.10 – Ecranul Casier

4.4 Scenariul Operator

Operatorul reprezintă ultimul actor cu care un participant poate să interacționeze în mod direct, acesta gestionând etapa finală a procesului de înscriere: validarea datelor și confirmarea. La fel ca în cazul casierului, operatorul dispune de un panou de căutare a candidaților după CNP sau după numărul de ordine, însă, spre diferență de acesta, are control complet asupra datelor și opțiunilor introduse în sistem, cu scopul de a valida și de a corecta eventualele greșeli.

La finalul evaluării profilului, operatorul va apăsa pe butonul de confirmare, disponibil în zona inferioară a ecranului.

Odată confirmată înscrierea unui participant, sistemul va genera documentele personalizate și le va expedia prin intermediul poștei electronice. Simultan, participantul nu va mai putea să își modifice datele sau opțiunile înscrise în sistem, fiind inclus, pe baza acestora, în procesul oficial de repartizare al universității.

Totodată, operatorul dispune de două moduri de căutare, pe lângă cel menționat anterior, putând fi listați toți acei studenți care, după afișarea rezultatelor intermediare sau finale, și-au semnalat dorința de retragere a dosarului. Cu ajutorul acestei liste, operatorii pot organiza, în ziua recuperării dosarelor, o coadă specială, mai rapidă, dedicată acelor candidați care au utilizat funcționalitatea disponibilă în platformă.

Interfața dedicată utilizatorului de tip operator poate fi consultată în *Figura 4.11*.

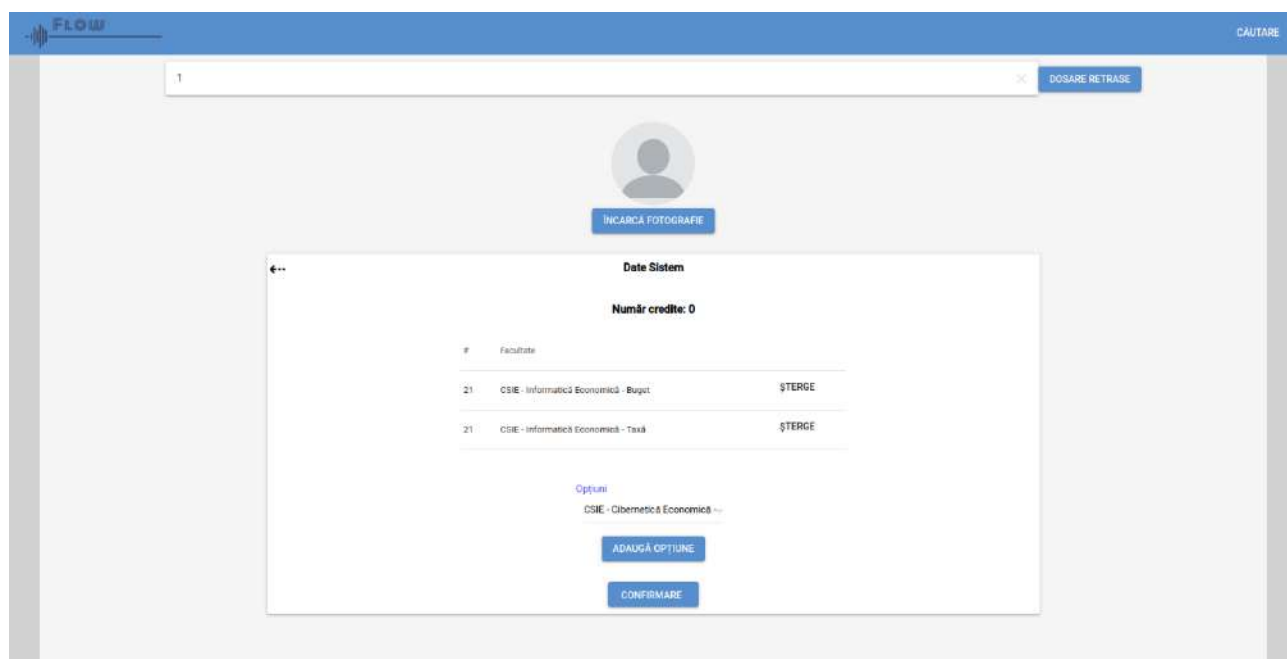


Figura 4.11 – Ecranul Operator

4.5 Scenariul Administrator

Ultimul rol auxiliar, cel de administrator, dispune de un set numeros de instrumente ce simplifică gestiunea personalului și remedierea situațiilor particulare.

Primul ecran, atașat în Figura 4.12, surprinde panoul general de informare. Acesta este actualizat în timp real, administratorul putând ține evidența numărului de participanți înscriși în ziua curentă, grupați în funcție de ora la care operatorul a făcut confirmarea datelor, și a cererilor de schimbare a poziției emise de către voluntari.

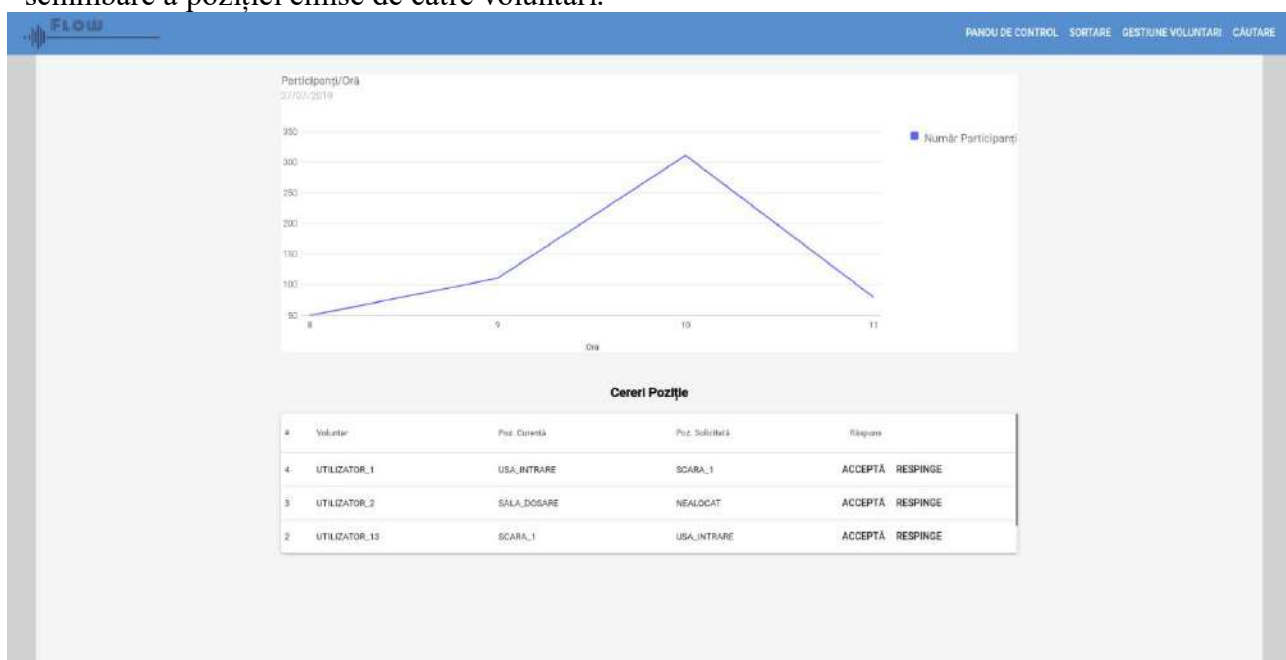


Figura 4.12 – Ecranul de control

Tot în scopul organizării personalului, ecranul de gestiune, afișat în *Figura 4.13*, permite unui administrator adăugarea unui utilizator nou în sistem și asocierea implicită a unei poziții, precum și reorganizarea rolurilor și pozițiilor ocupate de către voluntarii existenți.

#	Voluntar	Rol	Poziție
1	VOLUNTAR_1	Voluntar	USA_INTRARE
2	CASIER_1	Casier	-
3	VOLUNTAR_2	Voluntar	NEALOCAT
1	OPERATOR_1	Operator	-

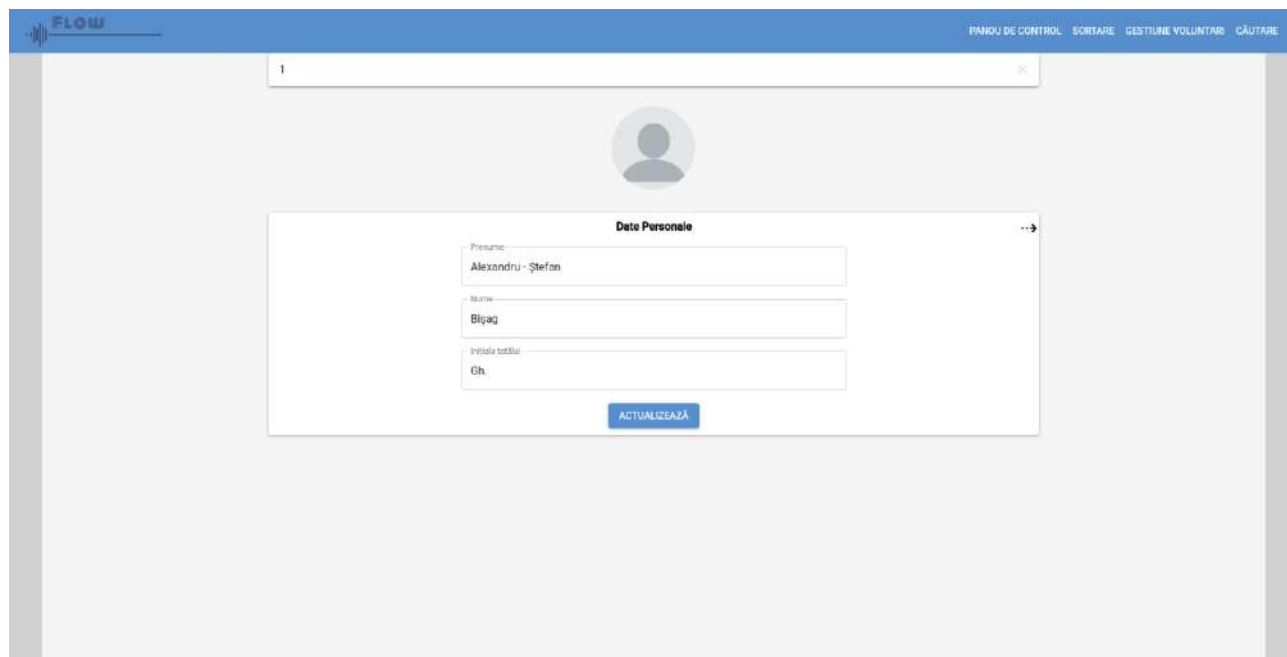
Figura 4.13 – Ecranul de gestiune a personalului

Odată cu finalizarea perioadei de înscriere, administratorul poate determina apelarea modului Sort ce va efectua repartizarea participanților în funcție de criteriile preconfigurate și de datele înscrise în sistem. Ecranul ce permite apelarea acestor procedee poate fi observat în *Figura 4.14*.

#	Iterație	Documente
1	TEST 1	DESCARCĂ
2	REPARTIZARE_1 INT	DESCARCĂ

Figura 4.14 – Ecranul de control al sortării

Operațiunea poate fi executată atât sub formă oficială, generând documentele de repartizare, actualizând informațiile tuturor candidaților și trimițând acestora e-mail-uri de înștiințare, cât și în regim de test, generând fișiere care să permită personalului să consulte în prealabil rezultatele și să identifice potențialele erori de calcul.



The screenshot shows a web application interface with a blue header bar. On the left of the header is the 'FLOW' logo. On the right are navigation links: 'PANO DE CONTROL', 'SORTARE', 'GESTIUNE VOLUNTAR', and 'CĂUTARE'. Below the header, there is a search bar containing the number '1'. In the center of the page is a large, light gray rectangular area. At the top of this area is a circular profile icon. Below the icon is a form titled 'Data Personale' with a right-pointing arrow. The form contains three input fields: 'Prenume' with the value 'Alexandru - Ștefan', 'Nume' with the value 'Bîșag', and 'Inițiala tatălui' with the value 'Gh.'. Below these fields is a blue button labeled 'ACTUALIZEAZĂ'.

Figura 4.15 – Ecranul de actualizare a datelor

Un ultim instrument auxiliar, adăugat preventiv, cu scopul de a remedia potențialele erori, îl reprezintă ultimul ecran de control destinat administratorilor, ecranul de căutare a candidaților. Spre diferență de toți ceilalți utilizatori, ce pot actualiza datele unui participant doar într-o anumită măsură sau până la apariția unui eveniment, administratorul deține drepturi depline cu scopul de a corecta erorile observate sau semnalate de către candidați în zilele ulterioare perioadei de depunere a dosarelor.

Această interfață este prezentată în *Figura 4.15*.

Concluzii

Aplicația Flow sau, formal, sistemul informatic de gestiune a participanților la un concurs de admitere organizat de către o universitate, reprezintă produsul finit rezultat în urma procesului descris în detaliu în capitolele de mai sus. La finalul lucrării, aceasta se află într-o stare optimă de funcționare, putând fi publicată și scalată conform instrucțiunilor expuse și a schemelor atașate anterior, ideea în jurul căreia a fost proiectată arhitectura și implementat fiecare modul nefiind alta decât automatizarea.

Din punctul meu de vedere, această aplicație ar putea reprezenta un prim pas în definirea unui instrument digital standardizat care să uniformizeze modul de desfășurare a proceselor de admitere organizate de către instituțiile publice de învățământ superior din România. Principalele beneficii aduse sunt, pe lângă performanță și modernitate, îmbunătățirea imaginii externe în raport cu potențialii studenți și eliminarea gradului de confuzie creat în rândul participanților ce se înscriu la concursurile de admitere ale mai multor universități, aceștia fiind expuși unui traseu similar, cunoscând, după o primă experiență, toți pașii necesari finalizării procesului.

Privit la nivel macro, un sistem comun, utilizat de către toate universitățile majore din țară ar putea determina o strânsă colaborare între acestea pentru îmbunătățirea sistemului, marcând, implicit, apariția unei infrastructuri digitale partajate la nivelul întregii țări. Acest lucru, însă, pare, în prezent, departe de materializare, principalii factori fiind costurile foarte ridicate și cantitatea necesară de muncă ce trebuie depusă nu doar pentru dezvoltarea unor sisteme fiabile, ci și pentru întreținerea și îmbunătățirea continuă a acestora. În acest sens, aplicația Flow ar putea reprezenta, încă o dată, un punct solid de plecare, deoarece, ulterior susținerii lucrării curente, aceasta va fi publicată în regim open-source, sub *licență MIT* [13] ce permite oricărui doritor să acceseze, modifice și să distribuie codul sursă în orice altă formă, fără existența unui cadru legal coercitiv.

În ciuda încheierii lucrării curente, sistemul, deși ajuns la o versiune stabilă, este departe de a-și cunoaște forma finală, *un produs software neputând fi niciodată finalizat, ci doar abandonat* [14]. În acest sens, am remarcat, încă din momentele incipiente ale etapei de implementare, câteva direcții viitoare de dezvoltare ce ar putea atât să îmbunătățească performanța netă a aplicației, cât și să simplifice considerabil procesele de scalare, mentenanță și depanare.

Prima și cea mai importantă modificare ce s-ar putea face în contextul creșterii performanței potențiale a sistemului o reprezintă transformarea fiecărui modul într-un microserviciu. Prin însăși observarea arhitecturii și a detaliilor de implementare, se poate deduce că acesta este următorul pas firesc în dezvoltarea sistemului, piesele necesare fiind, deja, pregătite pentru această schimbare majoră. Trecerea la o arhitectură bazată pe microservicii ar permite utilizarea unor tehnologii auxiliare foarte puternice pentru orchestrarea și observarea instanțelor ce compun partea de back-end a aplicației. Un prim astfel de instrument poate fi sistemul de orchestrare Ansible introdus pentru a asigura configurarea mașinilor într-o manieră automatizată, mult mai eficientă decât prin rularea script-ului Bash dezvoltat în prezent. Împreună cu publicarea modulelor, acum devenite micro-servicii, într-un cluster Kubernetes, suportat chiar și de către furnizorul IaaS actual, Google Cloud Platform, publicarea și scalarea aplicației încep să se îndepărteze, câte puțin, de zona de operații manuale înspre mult discutatul domeniu DevOps ce crește, pe zi ce trece, în popularitate [15].

Un avantaj major al structurii modulare îl reprezintă posibilitatea de dezvoltare, în paralel, cu tehnologii complet distincte, a fiecărei unități din sistem, în funcție de tipul de operații pe care acestea trebuie să le execute. Dintre modulele existente, modulul Sort, singurul modul ce, de altfel, nu este inclus în schema de replicare, se remarcă prin cantitatea ridicată de operații matematice pe care trebuie să le soluționeze, fiind responsabil pentru repartizarea tuturor studenților înscriși. Un dezavantaj cunoscut al platformei Node.js îl reprezintă *performanța computațională scăzută prin comparație cu*

limbajele compilate [16]. În acest sens, o modificare ce ar conduce la creșterea performanței generale sistemului o reprezintă rescrierea modulului Sort în Java, utilizând, pe post de framework dedicat, Spring Boot.

Cea de-a treia direcție de dezvoltare se concentrează pe procesul de mentenanță și depanare a sistemului, proces care, în contextul unei scheme complexe de scalare, poate solicita foarte mult timp. Astfel, o potențială direcție de dezvoltare ar fi adăugarea unui sistem de colectare, prezentare și interogare a log-urilor scrise de către toate instanțele. Opțiunea cea mai fiabilă este, fără doar și poate, trio-ul Elasticsearch, Logstash și Kibana, mai bine cunoscut sub numele de ELK Stack. Suplimentar acestui instrument foarte puternic de analiză și diagnosticare, poate fi implementat și un sistem de monitorizare și colectare a datelor analitice, cum este, de exemplu, Datadog, cu ajutorul căruia administratorii de sistem își pot face o imagine de ansamblu mai clară asupra performanței și a nivelului de cereri primite de către sistem.

Adăugarea acestor funcționalități și sisteme auxiliare la baza deja solidă oferită de forma actuală a aplicației Flow ar reprezenta un pas important în definitivarea sistemului și transformarea sa într-un produs software complet, complex și fiabil, indiferent de condițiile de utilizare cărora este expus.

Bibliografie

- [1] Wikipedia, „WebSocket,” [Interactiv]. Available: <https://en.wikipedia.org/wiki/WebSocket>. [Accesat Iunie 2019].
- [2] P. LePage, „Your First Progressive Web App,” Google, 29 Mai 2019. [Interactiv]. Available: <https://developers.google.com/web/fundamentals/codelabs/your-first-pwapp/>. [Accesat Iunie 2019].
- [3] M. Gaunt, „Service Workers: an Introduction,” Google, [Interactiv]. Available: <https://developers.google.com/web/fundamentals/primers/service-workers/>. [Accesat Iunie 2019].
- [4] „Introduction - Material Design,” Google, [Interactiv]. Available: <https://material.io/design/>. [Accesat Iunie 2019].
- [5] „Koa.js - Introduction,” [Interactiv]. Available: <https://koajs.com/#introduction>. [Accesat Iunie 2019].
- [6] NPM, „NPM,” [Interactiv]. Available: <https://npmjs.com>. [Accesat 2019 Iunie].
- [7] Auth0, „Introduction to JSON Web Tokens,” [Interactiv]. Available: <https://jwt.io/introduction/>. [Accesat Iunie 2019].
- [8] R. Fajar, „Using Redis Pub-Sub with Node.js,” 20 Martie 2018. [Interactiv]. Available: <https://medium.com/@ridwanfajar/using-redis-pub-sub-with-node-js-its-quite-easy-c9c8b4dae79f>. [Accesat Iunie 2019].
- [9] R. Saive, „TecMint - How to configure nginx as reverse proxy for Node.js app,” 17 Noiembrie 2018. [Interactiv]. Available: <https://www.tecmint.com/nginx-as-reverse-proxy-for-nodejs-app/>. [Accesat Iunie 2019].
- [10] Electronic Frontier Foundation, „Certbot Instructions,” [Interactiv]. Available: <https://certbot.eff.org/>. [Accesat Iunie 2019].
- [11] J. Delaney, „Trusted Web Activity - PWA to Play Store Guide,” Fireship, 20 Februarie 2019. [Interactiv]. Available: <https://fireship.io/lessons/pwa-to-play-store/>. [Accesat Iunie 2019].
- [12] S. Brown, „Coding Architecture - Modular Monoliths,” 14 Octombrie 2015. [Interactiv]. Available: <http://www.codingthearchitecture.com/presentations/sa2015-modular-monoliths>. [Accesat Iunie 2019].
- [13] „Choose a License - MIT License,” [Interactiv]. Available: <https://choosealicense.com/licenses/mit/>. [Accesat Iunie 2019].

- [14] J. Saddington, „Software: Never Finished, Only Abandoned,” 1 Noiembrie 2014. [Interactiv]. Available: <https://john.do/software/>. [Accesat Iunie 2019].
- [15] „Learnitguide.net - Install Kubernetes Cluster using Ansible,” 20 Ianuarie 2019. [Interactiv]. Available: <https://www.learnitguide.net/2019/01/install-kubernetes-cluster-using-ansible.html>. [Accesat Iunie 2019].
- [16] J. Jenson, „Performance Comparison: Java vs Node,” [Interactiv]. Available: <https://www.tandemseven.com/blog/performance-java-vs-node/>. [Accesat Iunie 2019].