

UNIVERSITATEA BUCUREȘTI  
FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ  
SPECIALIZAREA INFORMATICĂ



Lucrare de licență

ROBOTUL ORICĂRUI STUDENT

Profesor coordonator

Prof.dr. ALIN ȘTEFĂNESCU

Absolvent

SIMONA POP

*București 2021*

# Elliot, Your University Robot

Lucrare de licență

ROBOTUL ORICĂRUI STUDENT

Profesor coordonator

Prof.dr. ALIN ȘTEFĂNESCU

Absolvent

SIMONA POP

*București 2021*

# Cuprins

<b>Introducere</b>	<b>3</b>
<b>1 Scopul și preceptele automatizării</b>	<b>6</b>
1.1 Un robot pentru fiecare student . . . . .	6
1.2 Robotul studentului FMI UNIBUC . . . . .	7
<b>2 Analiza proceselor-candidat</b>	<b>8</b>
2.1 Obiectivele automatizării . . . . .	8
2.2 Descrierea AS-IS (înaintea automatizării) . . . . .	9
2.2.1 Tabelul proceselor AS-IS . . . . .	9
2.2.2 Diagramele AS-IS . . . . .	10
2.2.3 Detalierea etapelor din diagramele AS-IS . . . . .	11
2.3 Descrierea TO-BE (în urma automatizării) . . . . .	13
2.3.1 Diagrama TO-BE . . . . .	14
2.3.2 Detalierea etapelor din diagrama TO-BE . . . . .	19
<b>3 Tehnologii utilizate în dezvoltarea aplicației</b>	<b>37</b>
3.1 Robotic Process Automation (RPA) . . . . .	37
3.2 UiPath Studio și UiPath Robot . . . . .	39

3.3	VisualBasic.NET (VB.NET) . . . . .	39
3.4	Python . . . . .	40
3.5	PyQt și QtDesigner . . . . .	40
3.6	Go . . . . .	40
3.7	Node.js . . . . .	41
<b>4</b>	<b>Arhitectura aplicației</b>	<b>42</b>
4.1	Interfața grafică . . . . .	42
4.1.1	Galeria paginilor . . . . .	42
4.1.2	Harta paginilor . . . . .	47
4.1.3	Folderul cu utilități . . . . .	51
4.2	Executabilele pentru Moodle . . . . .	53
4.3	Flowchart al roboților software . . . . .	56
4.4	Dependințe . . . . .	57
4.5	Orizonturi și perspective de progres . . . . .	58
4.6	Notițe de final . . . . .	58
	<b>Bibliografie</b>	<b>59</b>

# Introducere

## Câte ceva despre RPA și UiPath

Oare ce înseamnă *"a ține pasul cu tehnologia"* ? Cu siguranță semnifică pe deplin altceva pentru un dezvoltator software, un programator, un tehnician IT sau un inginer față de ceea ce semnifică pentru o persoană oarecare situată profesional în afara ariei de acoperire a domeniului informatic. Pe când informaticienii se confruntă neîntrerupt cu ritmul în care avansează posibilitatea alcătuirii unor programe din ce în ce mai complexe, cât și diversitatea uneltelor puse la dispoziție de proaspete descoperiri datorate cercetării, individul obișnuit se adaptează încă la traiul înconjurat de dispozitive tot mai capabile și de calculatoare tot mai performante. Cert este însă că orice participant la civilizație, indiferent de pregătirea sa didactică sau activitatea sa de zi cu zi, își ridică la un moment dat problema propriei sincronizări cu vastul concept de tehnologie.

Odată evidențiată relativitatea sub spectrul căreia se găsește definirea ideii de a face față progresului tehnologic, natural ar fi ca totuși să convenim asupra unei definiții universale care să înglobeze fundamentele relației om - tehnologie, făcând abstracție de divergențele apărute în urma particularizării elementului uman. Propun atunci să considerăm că *"a ține pasul cu tehnologia"* înseamnă pur și simplu a căuta permanent căi de a-ți îmbunătăți activitatea (oricare ar fi aceea) solicitând și acceptând ajutorul resurselor tehnologice la care îți poți procura acces. Desigur, intenția de integra aceste resurse într-un anumit nivel al vieții tale trebuie însoțită și de o cunoaștere cel puțin parțială a ofertelor și modului lor de funcționare, cum de altfel întotdeauna este nevoie să și stăpânești puterea pe care o ai în posesie.

Înspre aceeași temă de gândire și-au ațintit atenția și specialiștii de la UiPath, cea mai prestigioasă companie de Software RPA din lume. Acronimul RPA aparține expresiei "Robotic Process Automation" și se referă la procedeul prin care sarcinile recurente ale operatorului uman în cadrul interacțiunii sale cu calculatorul sunt îndeplinite de către un așa-numit robot. Înlocuirea efortului omului cu performanța robotului poartă denumirea de automatizare a unor procese la care este supus utilizatorul și reprezintă, conform celor de la UiPath, o inovație tehnolo-

logică esențială, ce trebuie popularizată în rândul întregii populații. Ei sugerează o abordare intitulată "*A Robot For Every Person*", inițiativă promovată prin sloganul următor:

“Just as Bill Gates envisioned a world with a PC for every desktop, UiPath envisions a world with a robot for every person. When every worker has a robot to help with mundane tasks, it frees employees to use time for higher-level work. Workers feel less stressed and become happier and more productive.”

Proiectul a luat naștere odată cu sedimentarea mentalității "*Automation First Era*", al cărei obiectiv declarat presupune facilitarea experienței unui angajat la locul de muncă prin intervenția automatizării. Noțiunile și statisticile ce urmează sunt preluate direct din prezentarea [6] realizată de UiPath conceptului.

Oricare ar fi sectorul industrial sub aripa căruia se situează o organizație ce apelează la tehnologia RPA, un aport substanțial la eficiența și productivitatea performanței sale tot este garantat. S-au luat în considerare două tipuri de strategii:

1. *Top-Down Automation First* - Centrul de operare robotică (ROC) al unei organizații identifică cele mai comune sarcini repetitive ce trebuie săvârșite în cadrul acesteia și le atribuie roboților. Căutarea unei soluții se petrece per companie, iar roboții sunt programați așa încât să satisfacă nevoile firmei într-un context general.
2. *Bottom-Up Automation First* - Angajații unei companii sunt echipați cu roboți concepuți exclusiv în favoarea poziției lor profesionale, ba chiar încurajați și îndrumați de către centrul de operare robotică (ROC) să își genereze singuri automatizări proprii. Astfel, schemele de automatizare sunt executate per angajat, într-o formulă particularizată.

În timp ce un procent de vreo 3-7% din mulțimea sarcinilor dintr-o firmă este selectat de către ROC ca prezentând activități recurente și obositoare pentru operatorul uman, dar cu perspectivă de automatizare, peste 40% dintre sarcini sunt semnalate de către angajați ca fiind repetitive și preferabil de dat spre automatizare. Acest aspect conduce la superioritatea strategiei Bottom-Up în raport cu cea Top-Down. Iar dintr-o asemenea remarcă reies și avantajele formării de perechi persoană - robot.

Pe scurt, în ziua de astăzi se ridică la mare cinste personalizarea modelului RPA în funcție de cerințele și nevoile individului ce intră în posesia sa.

# Capitolul 1

## Scopul și preceptele automatizării

### 1.1 Un robot pentru fiecare student

Cum viziunea *"A Robot For Every Person"* se află constant în curs de extindere și tot mai multe companii adoptă tehnologia RPA, nu este o surpriză apariția grabnică a proiectului geamăn, *"A Robot For Every Student"*. Intențiile prevăzute de acest nou proiect și estimările ritmului în care vor fi puse în practică, deopotrivă relatate în lucrarea [4], pornesc de la premisa familiarizării viitorilor corporatiști și antreprenori cu tehnologia atât de valoroasă pieței actuale. Înmarmată încă din studenție cu aptitudinile necesare controlării roboților, forța de muncă va beneficia de un atu deosebit în ceea ce privește gestionarea timpului de lucru, calitatea produselor și serviciilor oferite, satisfacția postului și evitarea erorilor cauzate de factori umani precum oboseala sau neatenția. De asemenea, trebuie evidențiat că aducerea tinerilor la un oarecare nivel comun de cunoștințe despre automatizare, plan a cărui desfășurare se află abia în stare incipientă, se petrece treptat în diverse universități ale lumii și vizează o uniformitate ce nu ține seama de profilul academic. Cu alte cuvinte, se dorește predarea disciplinei RPA fiecărui program de studiu cuprins în structura unei universități, în ciuda posibilei lipse de tehnicitate a departamentului didactic în sine.

Lucrarea de față are ca scop întocmai construcția unui robot software atașat unui student ”purtător” și servește drept model al manierei în care poate fi utilizat mediul oferit de UiPath întru procurarea companionilor virtuali. Prin urmare, vor fi realizate automatizări complete ale câtorva procese recurente și mai puțin plăcute prin care trece în mod normal un student, iar aplicația va fi concepută cât să se plieze pe demersurile specifice Facultății de Matematică și Informatică a Universității din București.

## 1.2 Robotul studentului FMI UNIBUC

Păstrând ca orizont efectele satisfăcătoare ale strategiei *Bottom-Up* definite și elaborate în introducerea acestei documentații, se deduce imediat că autorul unui robot cu adevărat competent în acompanierea unui student nu poate fi altul decât studentul în sine. Firește, nici administrația universității, nici conducerea facultății și uneori nici măcar un student reprezentant al întregii comunități studentești nu pot anticipa pe deplin suita de responsabilități tehnice recurente ce pot interveni în viața unui student particular. Nimeni altcineva nu se află în poziția de a proiecta un robot mai util unei cauze anume decât însuși studentul în nevoie. Având acum aceste noțiuni evidențiate, se poate avansa la o caracterizare mai concretă a aplicației ce va fi prezentată în continuare. "Elliot, Your University Robot", cum am numit-o, servește drept model primar de robot UiPath pentru un student FMI UNIBUC și, implicit, drept exemplu de proiect pe care studenții oricărei facultăți din lume îl pot implementa în scopul facilitării propriei experiențe universitare.

Orice persoană integrată într-un context instituțional se confruntă constant cu aceleași activități monotone și obositoare care de cele mai multe ori nu depind direct de prestația ei propriu-zisă în cadrul instituției. Printre respectivele activități se numără trimiterea de emailuri standard, consultarea repetată a unor documente, solicitarea și predarea actelor-tip, consemnarea zilnică a progresului, mânuirea câtorva programe adiacente. Pe lângă risipa de timp și abaterea de la preocupările serioase pe care efectuarea acestora le presupune, intervine adesea factorul de epuizare umană, adică o formă de irosire a resurselor de energie deținute de participanții la dinamica organizației. Referindu-ne strict la situația studenților, interacțiunea cu secretariatul facultății în vederea încheierii operațiunilor birocratice sau accesarea unei multitudini de platforme pentru vizualizarea profilului digital creează întotdeauna bătaie de cap mai mult sau mai puțin sesizabile. Stăpânirea aptitudinii de a folosi RPA, deși probabil nu neapărat vitală în mediul facultăților ce știu să își gestioneze treburile administrative prin alte căi, nu conduce numai la asigurarea satisfacției utilizatorilor în perioada studenției, ci îi învață pe aceștia să își simplifice singuri munca odată angajați într-o companie, după cum am specificat și mai devreme. În sprijinul acestui obiectiv sar iarăși specialiștii UiPath sub forma parteneriatului "*Academic Alliance*". Parteneriatul, fondat în octombrie 2018, a cooptat încă din primul său an de activitate aproximativ 400 de universități membre, pătrunzând astfel pe teritoriul a 33 de țări, conform statisticii oferite de articolul [5]. Dintre respectivele țări, România și-a creat chiar un renume datorită investiției făcute de Universitatea Babeș-Bolyai în predarea unor cursuri semestriale de anvergură destinate automatizării. Între timp, aceleași cursuri au fost încorporate și în programa de studiu FMI UNIBUC.



# Capitolul 2

## Analiza proceselor-candidat

### 2.1 Obiectivele automatizării

După cum este și natural, premergător încercării de îmbunătățire a unor procese, intervine stadiul de investigare a terenului. În primă etapă, ne vom raporta, categoric, la perspectiva beneficiarului. Așadar, din poziția de student, ne punem următoarea întrebare: "Ce anume mă deranjează la felul în care sunt organizate mijloacele de contact dintre mine și facultate și cum l-aș putea schimba în avantajul meu?".

Trei fenomene principale au fost considerate la conceperea mecanismului lui Elliot:

- (1) interacțiunea studentului cu secretariatul la solicitarea și depunerea de acte necesare în context administrativ sau juridico-legal;
- (2) menținerea la zi a studentului cu starea disciplinelor la care participă din punct de vedere al materialelor postate de către profesori și notelor căpătate;
- (3) invocarea și obținerea ajutorului din partea secretariatului sau a corpului de profesori ori de câte ori studentul are nevoie.

Bineînțeles, întrebarea ce se conturează fără întârziere în mințile cititorilor acestei liste este dacă cele trei fenomene întruchipează chiar procesele cu care vom jongla în continuare. Răspunsul însă nu este pe cât de categoric poate ni l-am dori. Un proces nu constituie o unitate atât de bine definită încât să i se poată trasa granițele într-o mulțime de alte procese sau să-și piardă caracterul de proces prin divizare ori completare. Este limpede că putem privi fenomenele de mai sus ca fiind procese în sine. Dar la fel de ușor le putem comasa într-un singur proces mai mare sau le putem chiar fragmenta individual spre obținerea mai multor procese

concluse pe baza fiecăruia dintre ele. Spre exemplu, un proces conținut de fenomenul (2) ar putea fi procesul prin care studentul capătă materialele cu teoria predată la un curs anume. În plus, nu pierdem din vedere faptul că automatizăm sarcini pentru un student pe cont propriu și nu pentru un angajat dependent de contextul unei corporații, corporație care ar fi convenit deja la nivel global asupra structurii proceselor întâmpinate în cadrul ei. Deci modelarea activităților supuse automatizării va rămâne slujba proiectantului.

Acestea fiind punctate, vom păstra deocamdată procesele analizate în formele (1), (2) și (3), urmând ca automatizarea să le unească într-un flux unic, după cum se va remarca. De altfel, centralizarea într-o singură aplicație a tuturor acestor cerințe ridicate de studentul în nevoie oferă un beneficiu consistent automatizării.

## 2.2 Descrierea AS-IS (înaintea automatizării)

Vom considera procesele intitulate:

- (1) Get Help
- (2) Check Courses
- (3) Deal with Paperwork

### 2.2.1 Tabelul proceselor AS-IS

Denumire proces	Get Help	Check Courses	Deal with Paperwork
<b>Descriere proces</b>	Când întâmpină o problemă, studentul solicită ajutorul autorității competente să i-l ofere (profesorilor prin Email, secretariatului prin Helpdesk).	Studentul se pune la curent cu starea fiecărui curs. Pentru cele deja examinate caută să își afle nota, iar pentru celelalte își descarcă materialele încărcate de profesor.	Studentul poate opta pentru solicitarea de la secretariat a eliberării unui document în scop extern. Verificând periodic site-ului facultății, el află ce documente îi cere facultatea, putând să le descarce șablonul și să îl completeze.
<b>Date intrare</b>	textul cu descrierea problemei	denumirile cursurilor	denumirea oficială a tipului de cerere
<b>Date ieșire</b>	ticket pe Helpdesk; Email către profesor	arhivă de fișiere cu materiale; excel cu note la examene	șablon document; Email către secretariat
<b>Servicii invocate</b>	platforma Helpdesk; serviciu de Email (Microsoft Outlook)	Microsoft Teams; Moodle UNIBUC	serviciu de Email (Microsoft Outlook); Site-ul facultății (deschis în browser)

## 2.2.2 Diagramele AS-IS

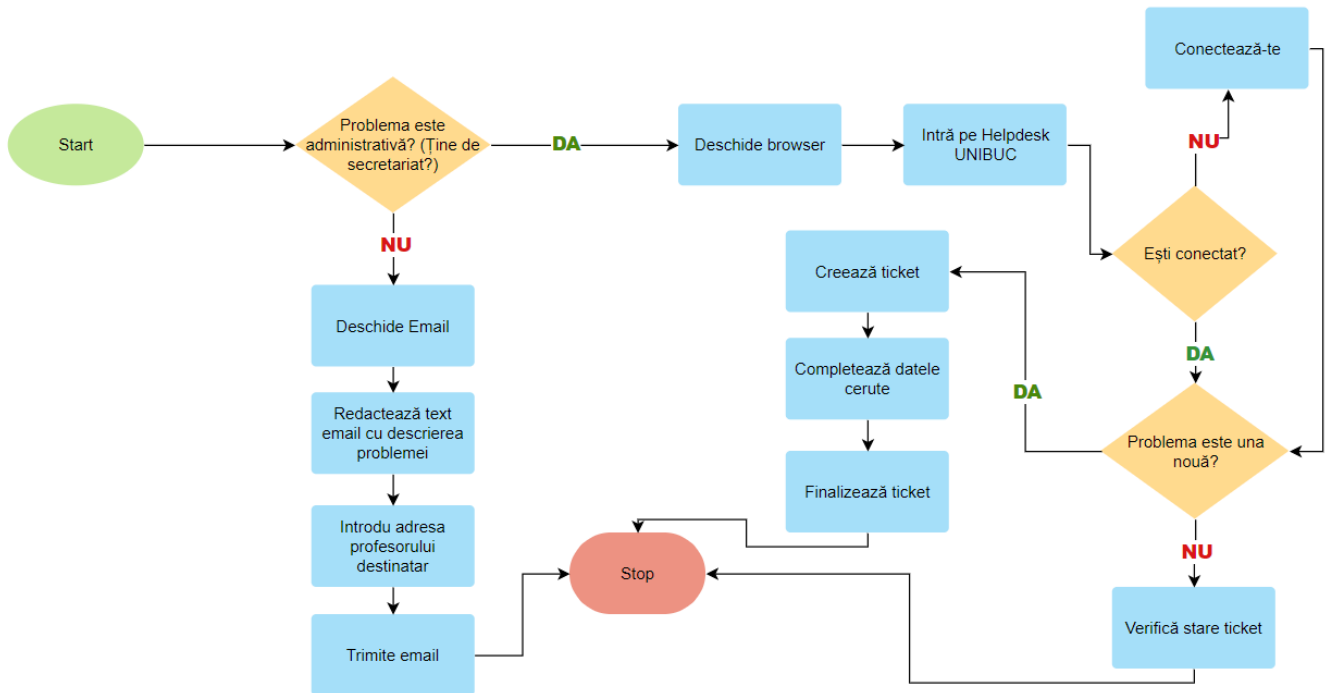


Fig. 2.1: Get Help AS-IS

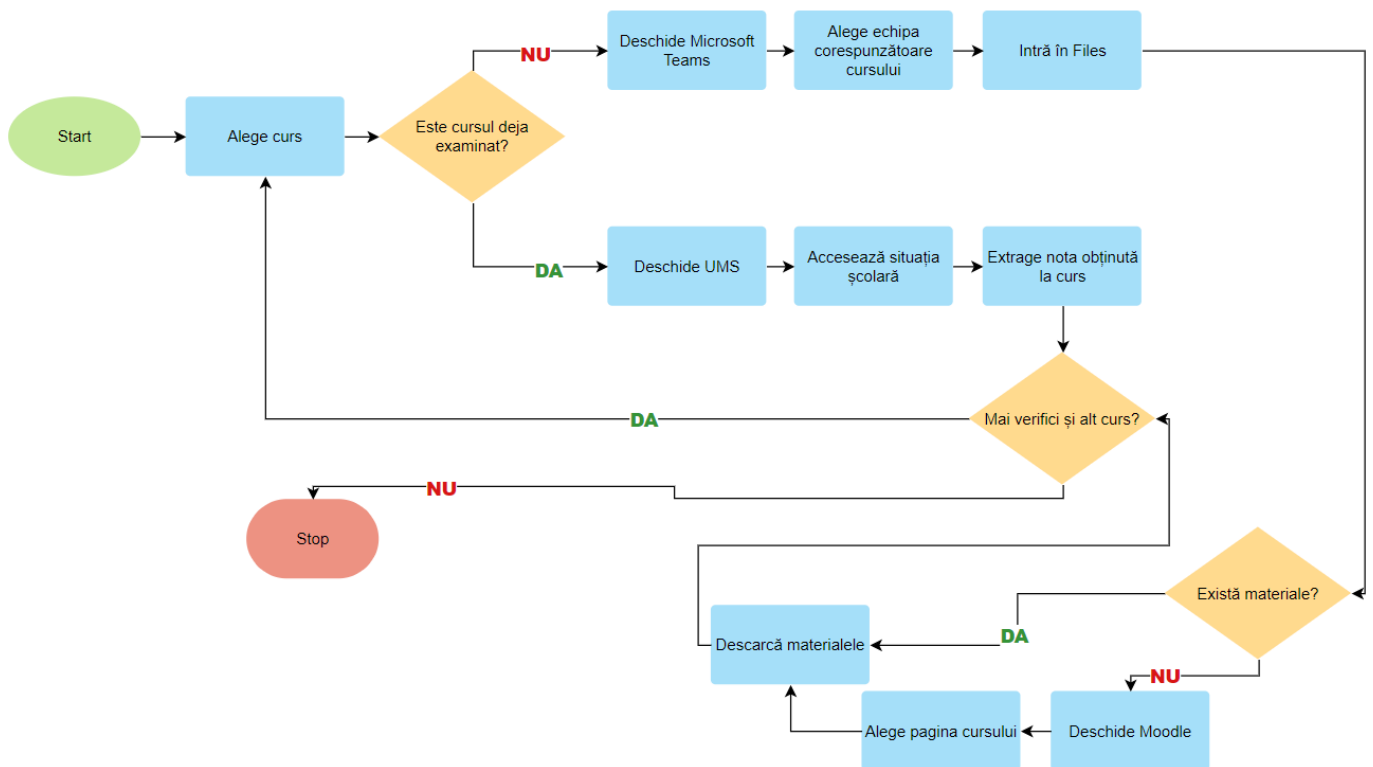


Fig. 2.2: Check Courses AS-IS

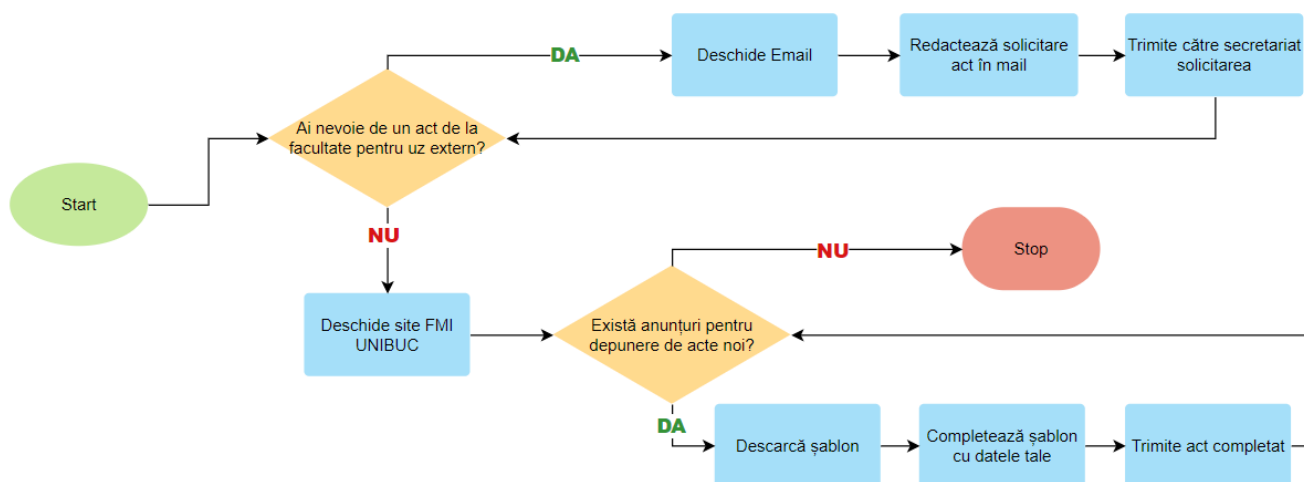


Fig. 2.3: Deal with Paperwork AS-IS

### 2.2.3 Detalierea etapelor din diagramele AS-IS

#### (1) Get Help

Procesul este reluat de fiecare dată când studentul se confruntă cu o problemă ce necesită intervenția unei autorități superioare în facultate (secretariat/ conducere/ decanat/ departament/ corp de profesori). În general, problemele studenților sunt de natura uneia dintre categoriile: nelămuriri (neclarități serioase sau simple curiozități), constatări ale unor potențiale nereguli provenind din partea autorității respective, rugăminți speciale etc.

Intuitiv, în primă instanță ar trebui stabilită autoritatea căreia ne vom adresa. În funcție de această alegere, vom decide ulterior și canalul pe care îl vom folosi. Odată cunoscut caracterul problemei, adoptăm etapele:

- Dacă problema este una administrativă și necesită ajutor din partea secretariatului, deschidem platforma Helpdesk, ne autentificăm în cazul în care încă nu avem contul conectat, iar apoi ne exprimăm problema sub forma unui tichet (formular) pe care îl completăm cu datele noastre personale și îl trimitem prin platformă. Evident, putem opta și pentru consultarea stării unui tichet trimis în trecut (descoperind dacă a fost sau nu soluționată o problemă semnalată anterior).
- Dacă problema nu ține de atribuțiile secretarilor, ci este legată mai degrabă de structura vreunui curs, modalități de evaluare ale unei discipline, teorie subordonată unei materii studiate, elaborarea lucrării de licență etc., atunci folosim serviciul de Email (Microsoft Outlook). Hotărâm cine ar trebui trecut ca destinatar, îi introducem adresa de email și ne formulăm problema în căsuța de text, după care trimitem mesajul.

## (2) Check Courses

Procesul prevede seria de acțiuni pe care un student le întreprinde în scopul actualizării progresului fiecărui curs la care este înscris. În cazul disciplinelor la care a fost deja evaluat, aceste acțiuni se îndreaptă către obiectivul aflării notei. De aceea am pretins accesarea carnetului virtual UMS ca sursă oficială a punctajelor obținute la cursuri. Desigur, odată ce studentul pătrunde în platformă, sunt urmați pașii consacrați de ghidul UMS, adică selectarea rubricii "Situație școlară" și extragerea notei căutate. Pentru disciplinele la care studentul urmează să fie examinat este vizată dobândirea materialelor postate de către profesori, care vor fi folosite la învățarea (sau recapitularea) materiei.

Presupunând că materialele apar fie în echipa de Teams a cursului, fie în secțiunea de pe Moodle dedicată cursului, inspectăm în ordine cele două medii:

- Statistic, cei mai mulți profesori FMI optează pentru Microsoft Teams când vine vorba de publicarea materialelor. Din acest motiv, cu aceeași aplicație și începem. Ulterior selectării echipei care ne interesează, accesăm pagina Files atașată ei și suntem liberi să pornim descărcarea materialelor, evident, dacă acestea și există.
- Lipsa materialelor din Teams indică prezența lor pe Moodle. Atunci, deschidem cea de-a doua platformă și intrăm pe pagina cursului, de unde putem descărca în sfârșit fișierele căutate.

Ciclul format în schemă asigură parcurgerea tuturor cursurilor cu progresul cărora ne dorim să ajungem la zi.

## (3) Deal with Paperwork

Procesul reprezintă un liant al fluxului de acte îndreptat de la student la conducere și viceversa. În principiu, diagrama a fost schițată astfel încât să dea curs evenimentelor după următoarea succesiune:

- Solicitarea de acte: Mai întâi, studentul încercă să se aprovizioneze cu toate actele (eliberate de către facultate) necesare pentru uz extern la momentul inițierii procesului. Așadar, el redactează și expediază pe Email, către secretariat, câte o solicitare pentru fiecare exemplar în parte. Am luat în calcul documente precum:
  - Adeverință de student (prin care facultatea care a eliberat-o atestă calitatea de student a persoanei trecute ca posesoare a actului);
  - Adeverință de licență (înlocuitoare de diplomă care garantează statutul de licențiat al posesorului);
  - Situație școlară (fișă matricolă ce conține toate notele și creditele căpătate de student de la începutul studiilor în facultate până la momentul eliberării).

- Depunerea de acte: Apoi studentului îi revine îndatorirea de a consulta site-ul facultății și a se pune în temă cu noile anunțuri. Dacă pe site este semnalată obligația completării și predării unui act de care studentul nu s-a ocupat până atunci, el își va procura un șablon pe care îl va completa, îl va semna și apoi îl va expedia conform procedurii cerut (prin formular, Email sau alte căi). Am considerat reprezentative documentele:
  - Fișă de lichidare (păstrează evidența datoriilor financiare pe care le are studentul față de instituție la încheierea contractului de studiu);
  - Adeverință de cercetare în vederea elaborării lucrării de licență (declarație a faptului că studentul a înaintat suficient în construcția proiectului);
  - Declarație de autenticitate a lucrării de licență (declarație a faptului că studentul nu a plagiat în construcția proiectului);
  - Cererea de înscriere la licență;
  - Cerere de cazare la cămin;
  - Cerere de echivalare a unui curs (prin care studentul solicită recunoașterea notei primite la o disciplină promovată în trecut ca notă a unei discipline la care va trebui să i se încheie o situație în viitor);
  - Cerere de transfer în altă grupă sau într-un an superior;
  - Acte de practică (care sunt adăugate la dosarul studentului și constituie dovada legală a desfășurării activității de practică a acestuia);
  - Dovada plății unei taxe.

## 2.3 Descrierea TO-BE (în urma automatizării)

Am specificat și mai devreme că o modificare dintre schema AS-IS și schema TO-BE, nu atât cea mai spectaculoasă cât cea mai vizibilă, este contopirea proceselor de dinainte de automatizare într-o unitate logică, într-un proces mai complex care se despică în trei trasee, macazul fiind schimbat de un parametru ce reține comanda operatorului. În exact același ritm, și cele trei procese amintite sunt la rândul lor comasări de alte procese controlate prin parametrii de comandă la alegerea pe care utilizatorul o va transmite din interfață.

Pe de-o parte, ne așteptăm ca schema să mai difere și din prisma simplului fapt că vom alege o configurație cât mai prielnică țelului nostru. Pe de altă parte, independent de preferințele noastre arhitecturale, numărul stărilor din alcătuirea proceselor va crește inevitabil, generându-se stări auxiliare de care robotul va avea nevoie.

## 2.3.1 Diagrama TO-BE

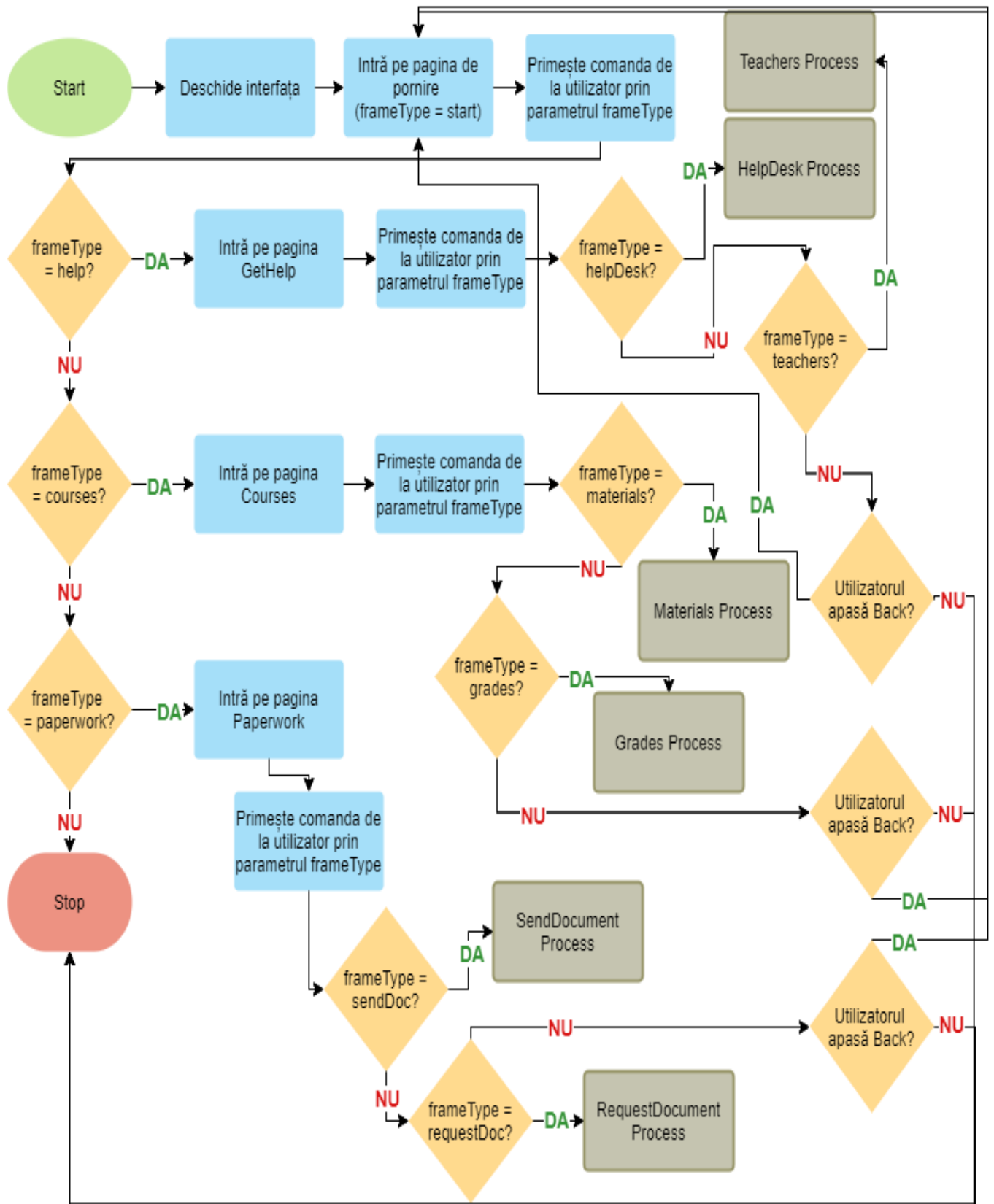


Fig. 2.4: Main TO-BE

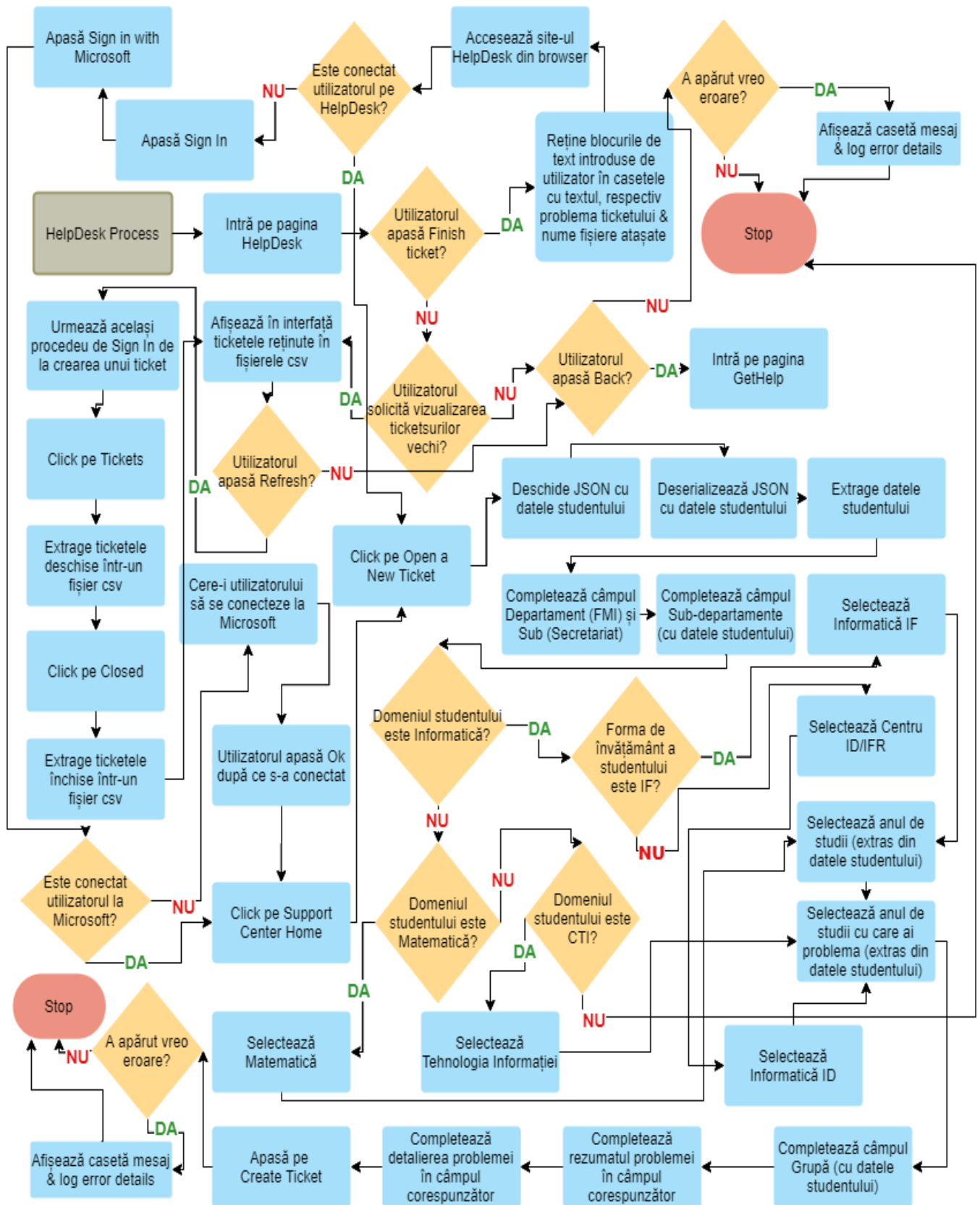


Fig. 2.5: HelpDesk TO-BE



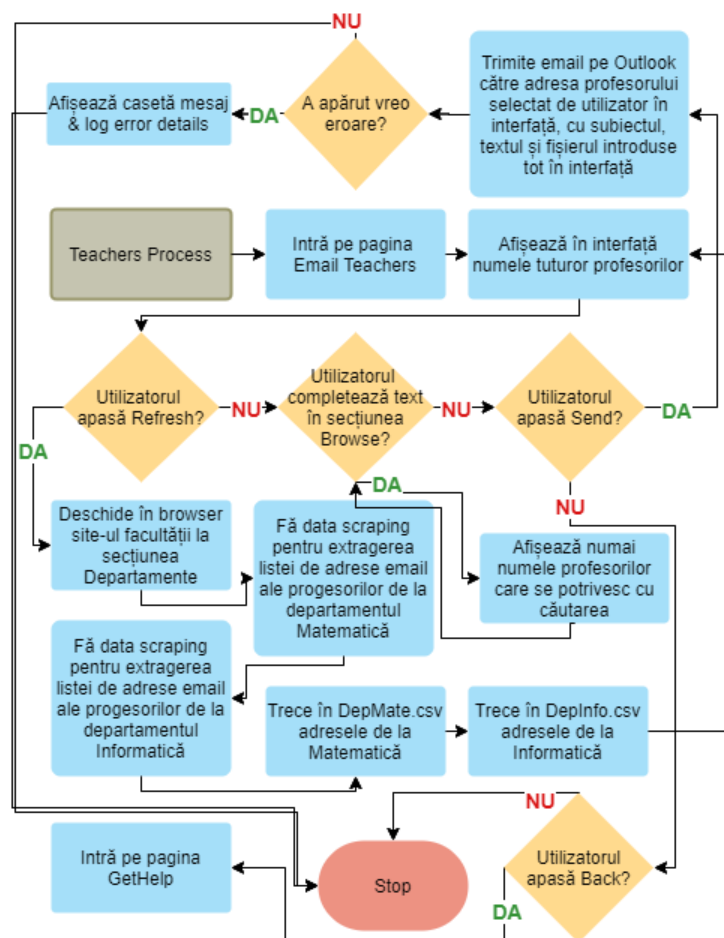


Fig. 2.6: EmailTeacher TO-BE

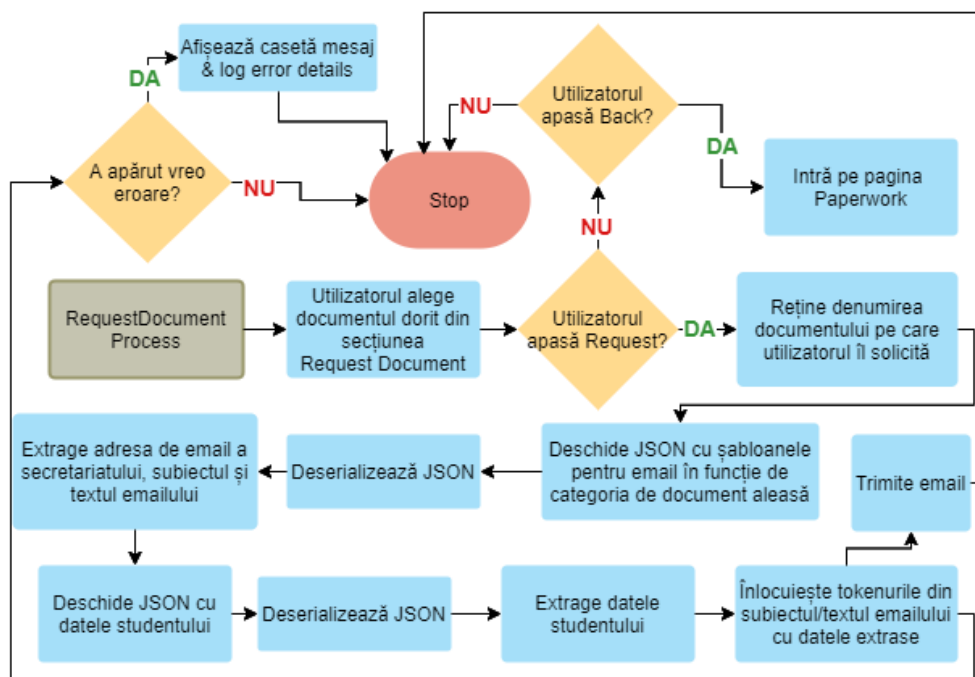


Fig. 2.7: RequestDocument TO-BE



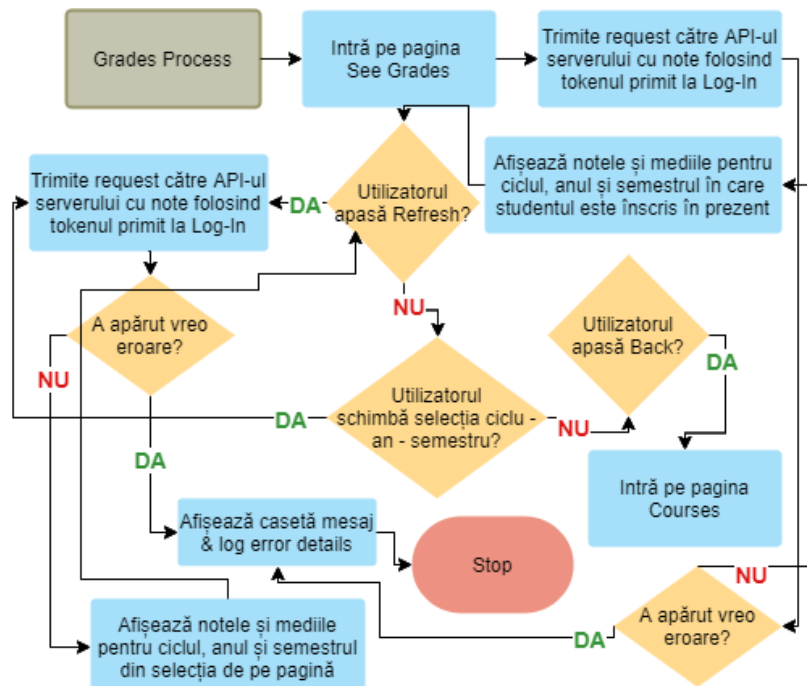


Fig. 2.9: SeeGrades TO-BE

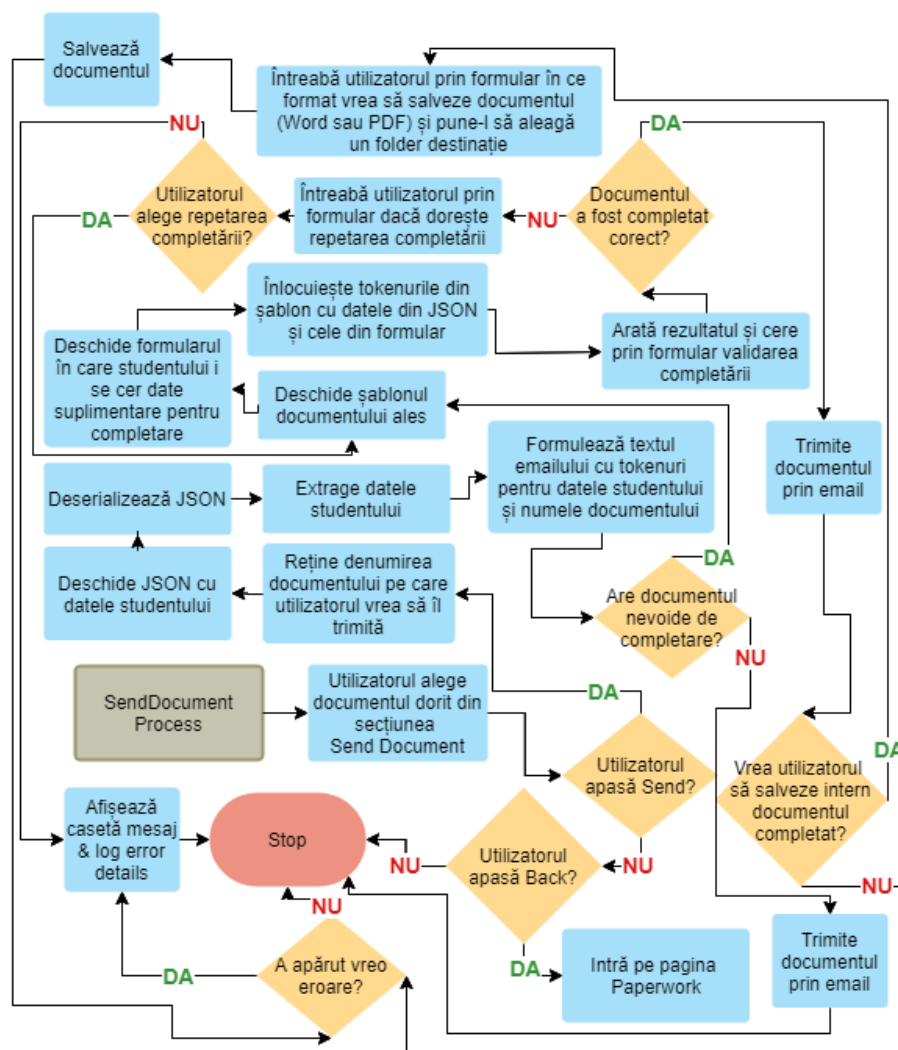


Fig. 2.10: SendDocument TO-BE

### 2.3.2 Detalierea etapelor din diagrama TO-BE

Diagrama anterioară intenționează a surprinde atât mișcările declanșate de utilizator direct din interfață, cât și traseul urmat de roboții invocați de aceste mișcări. Funcționarea ambelor sisteme (interfață și roboți) va fi expusă amănunțit într-un alt capitol. Deocamdată ne interesează schema automatizării strict la nivel de procese.

- Fig. 2.4: Fiecare proces al schemei de automatizare pornește de la preluarea unei comenzi din partea utilizatorului. Firește, precedent interceptării instrucțiunilor acestuia, este pregătit canalul de comunicare, adică este deschisă interfața la pagina de pornire, iar de aici înainte opțiunile studentului vor fi înregistrate și transmise mai departe ca parametrii.

Pe pagina de pornire sunt trecute trei dispoziții care corespund celor trei procese analizate în faza AS-IS. Selectarea uneia dintre cele trei va provoca automat deschiderea altei pagini din interfață (parametrul `frameType` servind la recunoașterea ei). Noua pagină va expune la rândul ei propriile ramuri disponibile pentru continuarea procesului, despicând (1), (2) și (3) din AS-IS în câte două fluxuri de acțiune și înființând paginile corespunzătoare în interfață, pagini dintre care tot utilizatorul alege (opțiunea fiind transmisă iarăși prin parametrul `frameType`):

- (i) Pagina de pornire → Pagina GetHelp → Pagina HelpDesk.
- (ii) Pagina de pornire → Pagina GetHelp → Pagina Email Teachers.
- (iii) Pagina de pornire → Pagina Courses → Pagina Get Materials.
- (iv) Pagina de pornire → Pagina Courses → Pagina See Grades.
- (v) Pagina de pornire → Pagina Paperwork → Secțiunea Send Document.
- (vi) Pagina de pornire → Pagina Paperwork → Secțiunea Request Document.

Fiecare dintre paginile derivate (toate paginile cu excepția celei de pornire) este dotată cu comanda Back, prin activarea căreia utilizatorul revine la pagina imediat anterioară celei curente. În diagramă, comanda Back este reprezentată ca un case al switch-ului inherent deschiderii oricărei noi pagini.

#### Fișierul *StudentDataJSON.json*

În vederea îndeplinirii comenzilor ulterioare, este necesară existența unei configurații inițiale a datelor personale ale studentului. Acestui scop îi servește stocarea internă (în aplicație) a fișierului *StudentDataJSON.json*. El conține informațiile esențiale despre proprietarul aplicației, cum ar fi numele, prenumele, adresa instituțională de email, domeniul, specializare, formele de învățământ, respectiv de finanțare, grupa, anul de studiu, anul universitar, anul începerii facultății, anul absolvirii dacă e cazul, numărul de telefon. JSON-ul arată astfel:

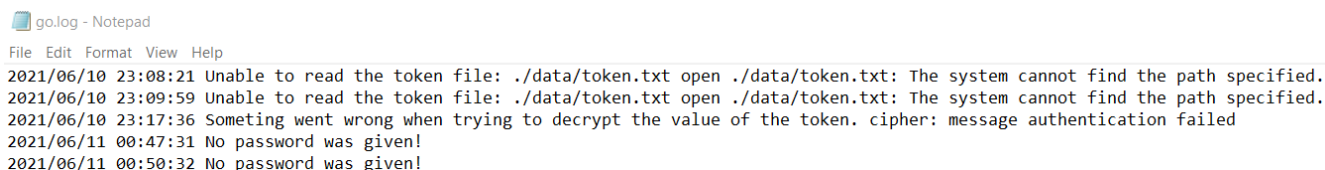
```
1 {
2   "EmailAddress": "simona.pop@s.unibuc.ro",
3   "Surname": "Pop",
4   "FirstName": "Simona",
5   "Phone": "+40723165394",
6   "Class": "342",
7   "Year": "III",
8   "UniversityYear": "2020-2021",
9   "Domain": "Informatica",
10  "Specialization": "Informatica",
11  "StudyForm": "IF",
12  "FinanceType": "taxa",
13  "beginYear": "2017",
14  "endYear": "2020"
15 }
```

Rămâne a fi explicată mai târziu și procedura de actualizare a fișierului. Dar mai întâi aruncăm o privire peste schemele logice care succed comenzile date de utilizator în paginile de pe nivelul al doilea (GetHelp, Courses, Paperwork).

### Fișierele-jurnal pentru erori log

O altă familie de fișiere aferentă aplicației și de importanță generală acesteia este familia fișierelor-jurnal. Putem percepe un fișier-jurnal ca pe o condică în care este păstrată evidența erorilor. La întâlnirea unei erori, fișierului i se mai adaugă o secvență de tipul:

"Could not *do what you asked for* due to : " + Exception.Message.ToString + " at " +  
Exception.Source.ToString



```
go.log - Notepad
File Edit Format View Help
2021/06/10 23:08:21 Unable to read the token file: ./data/token.txt open ./data/token.txt: The system cannot find the path specified.
2021/06/10 23:09:59 Unable to read the token file: ./data/token.txt open ./data/token.txt: The system cannot find the path specified.
2021/06/10 23:17:36 Something went wrong when trying to decrypt the value of the token. cipher: message authentication failed
2021/06/11 00:47:31 No password was given!
2021/06/11 00:50:32 No password was given!
```

Fig. 2.11: Exemplu de conținut al fișierului log pentru erorile din Go

Am creat mai multe fișiere log (mai precis trei), câte unul pentru fiecare unitate tehnologică folosită: *UiPath.log*, *go.log*, *python.log*.

- Fig. 2.5: Procesul HelpDesk, declanșat de selectarea paginii HelpDesk din interfață, îi oferă utilizatorului două posibilități principale, lăsând la o parte opțiunea Back discutată mai sus. Se poate deci înainta fie spre crearea unui nou tichet descriind o problemă pe care studentul nu a semnalat-o încă, fie spre vizualizarea stării tichetelor vechi, cu probleme deja semnalate. Schematic, am ilustrat decizia aceasta în diagramă prin switch-ul care procesează comanda utilizatorului în funcție de butonul apăsat în interfață.

(i) **Butonul Finish:** La apăsarea lui, robotul presupune încheiată redactarea unui nou tichet și, deci, extrage textele introduse de utilizator în cele două casete din interfață, anume titlul problemei semnalate și caracterizarea ei, iar automatizarea poate începe:

- Robotul creează singur noul tichet, accesând platforma *HelpDesk* prin browserul *Microsoft Edge*. În cazul ivirii vreunei erori, robotul anunță utilizatorul printr-o notificare că procesul nu a fost finalizat, afișându-i o casetă de mesaj cu sursa erorii, și completează în fișierul-jurnal (log) detaliile erorii.
- Robotul verifică dacă utilizatorul este conectat pe platforma HelpDesk (trimițându-i o notificare din care el să aleagă starea de logged-in).
- Dacă utilizatorul nu este conectat pe HelpDesk, robotul completează singur pașii de conectare (apăsând Sign In, apoi Sign In with Microsoft), iar când ajunge în punctul de legătură cu Microsoft, verifică tot printr-o notificare dacă utilizatorul este conectat la profilul său instituțional sau nu.
- Dacă utilizatorul nu este conectat pe Microsoft, robotul îi cere să își treacă direct în fereastra deschisă în browser datele de autentificare (am evitat intermedierea prin robot ca măsură de securitate). Odată conectat la Microsoft, utilizatorul apasă Ok pentru a îi transmite robotului că treaba poate continua.
- Fiind conectat la Microsoft și, implicit, la HelpDesk, robotul accesează Support Center Home, apasă *Open a New Ticket*, deschide fișierul *StudentDataJSON.json* și îl deserializează pentru a putea extrage din el datele studentului și completa cu ele câmpurile de pe pagina web de creare a tichetului.
- Mai întâi, independent de informațiile extrase, robotul selectează *Facultatea de Matematică și Informatică* în câmpul Departamente și *Secretariat* în primul câmp Sub-departamente. La completarea celui de-al doilea câmp Sub-departamente intră în scenă domeniul de studii al studentului, cunoscut din JSON. Fiindcă, în funcție de acesta, nu numai valorile trecute în câmpuri vor diferi, ci chiar alcătuirea părții rămase din formular.
- Pentru domeniul de studii Informatică, trebuie cunoscută și forma de învățământ din programul studentului (aflată tot în JSON). Dacă acesta studiază cu frecvență, atunci robotul alege Informatică IF în câmpul imediat următor, iar câmpurile ce survin sunt, în ordine: Anul de studii și Subiectul de ajutor (care

de fapt va fi tot anul de studii), pe care robotul le completează cu informațiile reținute. Pentru învățământ la distanță sau învățământ cu frecvență redusă, robotul alege Centru ID/IFR, selectează Informatică ID în următorul câmp și apoi completează anul de studii.

- Pentru domeniul de studii Matematică, forma de învățământ nu se consideră relevantă, robotul selectând pur și simplu Matematică în Sub-departamente și completând câmpurile ce apar cu anul de studii, respectiv subiectul de ajutor (din JSON).
- Pentru domeniul de studii CTI, robotul alege Tehnologia Informației și trece anul de studii în câmpul cu subiectul.
- Aici iau sfârșit tipurile distincte de rubrici, rămânând ca robotul să completeze, indiferent de domeniu (Informatică, Matematică, CTI), aceleași câmpuri. La Grupă se notează valoarea preluată din JSON, iar în rubricile de Rezumat al problemei și Detaliere a problemei cor fi plasate blocurile de text introduse de utilizator în interfață. Dacă utilizatorul atașează un fișier textului scris în interfață, acel fișier va fi transportat către HelpDesk și inclus în tichet.

Departamente	robotul alege mereu Facultatea de Matematică și Informatică
Sub-Departamente	robotul alege mereu Secretariat
Sub-Departamente	completat de robot cu datele din JSON
An	completat de robot cu datele din JSON
Subiect	completat de robot cu datele din JSON
Grupă	completat de robot cu datele din JSON
Titlu tichet	completat de robot cu titlul scris de utilizator în interfață
Text tichet	completat de robot cu textul scris de utilizator în interfață

- (ii) **Butonul See the status of older tickets:** Apăsarea sa deschide în interfață o pagină cu tichetele înregistrate de contul Microsoft al utilizatorului pe platforma HelpDesk și starea acestora. Există două liste cu asemenea tichete: una dintre ele conținându-le pe cele soluționate, iar cealaltă pe cele aflate în curs de soluționare. Listele sunt reținute intern de aplicație, fiecare în câte un fișier csv separat (*OpenTickets.csv* și *ClosedTickets.csv*) și corespund cu listele tichetelor utilizatorului cu care operează direct platforma HelpDesk. Totuși, întrucât pe HelpDesk pot apărea oricând modificări precum schimbarea stării unui tichet sau adăugarea/ștergerea altuia, fișierele csv au nevoie de o actualizare periodică. Această actualizare se invocă de către utilizator prin apăsarea butonului Refresh din interfață și prevede pașii:

- Robotul respectă întocmai procedul de sign-in pentru HelpDesk (și eventual Microsoft) în browserul *Microsoft Edge*. În plus, tratează erorile descoperite



prin aceeași strategie de notificare și notare în fișierul log.

- Robotul vizualizează pe HelpDesk tichetele vechi în același mod în care le-ar vizualiza și operatorul uman: apasă pe Tickets pentru citirea acelor tichete aflate în curs de procesare și pe Closed Tickets pentru tichetele declarate ca soluționate de către secretariat. Deosebirea este că, în situația robotului, citirea tichetelor se rezumă la aplicarea activității de *Data-Scraping* și distribuirea tabelului obținut în urma acesteia către fișiere csv stocate intern *OpenTickets.csv* și *ClosedTickets.csv* (care vor fi suprascrise).

### Fișierele *OpenTickets.csv* și *ClosedTickets.csv*

Având tipul csv (i.e. comma-separated-values) și fiind rezultate din data-scraping (acțiune care salvează datele sub formă de tabele), fișierele în discuție se vor deschide implicit în *Microsoft Excel* și vor fi vizibile în format de tabel.

	A	B	C	D	E
1	Ticket #	Create Date	Status	Subject	Department
2	948	20.11.2020	Inchis	URGENT!	Anul III

Fig. 2.12: Exemplu de conținut al fișierului *ClosedTickets.csv*

- Fig. 2.6: Procesul Teachers marchează schimbarea paginii GetHelp cu pagina Email Teachers și afișarea unei liste în interfață cu numele tuturor profesorilor din facultate. Pagina mai cuprinde și o casetă în care studentului i se cere să adauge subiectul mesajului digital și încă una în care să redacteze corpul mesajului. Dacă dorește, i se permite și să atașeze un fișier în pagină astfel încât robotul să i-l trimită profesorului odată cu mesajul scris. Posibilitățile de atac ale utilizatorului aplicației sunt reprezentate tot prin intermediul unei structuri switch cuprinzând și case-ul de click pe Back. Erorile apărute parcursul traseului explicat mai sus.
  - (i) **Butonul Refresh:** Când butonul este apăsat, robotul deschide browserul *Microsoft Edge* și intră pe site-ul facultății, la secțiunea Departamente. Alege departamentul Matematică, cheamă funcția de data-scraping și depune în fișierul *DepMate.csv* tabelul rezultat la extragerea datelor (numele și adresele profesorilor din departamentul de matematică). Aceeași serie de activități se repetă pentru departamentul Informatică, fișierul *DepInfo.csv*.
  - (ii) Se comportă ca orice altă casetă de căutare. Șirul de caractere introdus de utilizator reduce numărul elementelor listei de profesori la numărul elementelor care se potrivesc căutării.
  - (iii) **Butonul Send:** Odată apăsat, blochează datele pe care studentul a apucat să le introducă în interfață până în clipa apăsării. La nivel logic, funcționează ca o validare a inputului. Așadar, presupunem că studentul a introdus subiectul și textul emailului (cu sau fără fișier atașat) și a selectat din listă numele destinatarului.



În punctul acesta, procesul Teachers culminează cu trimiterea efectivă a emailului. Vom folosi mediul *Microsoft Outlook* pentru serviciul de email și datele oferite de utilizator pentru compunere.

### Fișierele *DepMate.csv* și *DepInfo.csv*

Din aceste două fișiere csv sunt scoase numele profesorilor facultății care apar în interfață, pe pagina Email Teacher. Selectarea validată (după apăsarea butonului Send) a unui nume din lista paginii trimite robotul către adresa de email asociată numelui respectiv, care se va găsi într-unul dintre cele două fișiere.

	A	B
1	Column-0	Email
2	Prof.dr. Marian Aprodu	marian.aprodu@fmi.unibuc.ro
3	Prof.dr. Cristian Bereanu	cristian.bereanu@fmi.unibuc.ro
4	Prof.dr. Lucian Beznea	lucian.beznea@fmi.unibuc.ro
5	Prof.dr. Daniel Bulacu	daniel.bulacu@fmi.unibuc.ro
6	Prof.dr. Aurelian Cernea	acernea@fmi.unibuc.ro

Fig. 2.13: Exemplu de conținut al fișierului *DepMate.csv*

- Fig. 2.8: Procesul Materials începe din pagina Get Materials deschisă de pe pagina Courses și respectă același circuit logic ca procesele detaliate până acum. Comenzile așteptate de la utilizator apar grafic pe scheletul unui switch din care nu lipsește case Back.

La încărcarea paginii Get Materials, interfața așază în centrul ferestrei lista cursurilor la care studentul este înscris la momentul respectiv. Procesul este bifurcat în succesiuni logice izolate, dictate de utilizator din interfață prin selectorul de platformă. Opțiunile oferite de selector sunt *Moodle* și *MS Teams*, iar denumirile apărute în lista cursurilor diferă, evident, de la selecția unei platforme la selecția alteia. Mai exact, atunci când opțiunea aleasă din fereastra Get Materials este Moodle, elementul tip view va oglindi conținutul fișierului *coursesMoodle.csv* unde sunt stocate numele cursurilor întocmai cum apar și pe Moodle. În schimb, atunci când sursa precizată este MS Teams, în pagină vor fi vizibile cursurile din *coursesTeams.csv*, intitulate așa cum sunt intitulate echipele corespunzătoare lor pe Teams. Acestea fiind lămurite, prezentăm restul elementelor din pagină.

### Fișierele *coursesMoodle.csv* și *coursesTeams.csv*

Am specificat mai devreme că ambele fișiere sunt umplute cu denumirile date cursurilor pe platformele referite de titlurile lor. Ceea ce nu am subliniat încă este că denumirile cursurilor nu sunt singurele elemente ale fișierelor, cu toate că sunt într-adevăr singurele elemente care se și afișează în interfață.

Practic, fișierul *coursesMoodle.csv* conține tupluri de forma: *denumire\_curs* - *id\_curs*. Id-urile (codurile) cursurilor sunt extrase de pe Moodle prin același mijloc și în același timp cu denumirile cursurilor și sunt esențiale, după cum se va observa, căutării materialelor de la o disciplină aleasă.

Totodată, vizualizat ca excel, fișierul *coursesTeams.csv* este alcătuit din două coloane: *denumire\_curs* și *descriere\_curs*. Actualizarea descrierilor coincide ca proces cu actualizarea denumirilor cursurilor, dar nu este neapărat importantă pentru funcția de extragere a materialelor.

În sfârșit, celelalte elemente ale paginii Get Materials sunt:

- (i) **Caseta Browse:** Pagina actuală este dotată cu o casetă Browse, în interiorul căreia șirul de caractere tastat de student este procesat, comparat cu titlurile din csv-ul platformei selectate și întrebuințat la filtrarea listei din view, pe pagină rămânând numai titlurile de cursuri care includ secvența dată la tastatură.
- (ii) **Butonul Refresh:** Firește, apăsarea butonului Refresh are drept consecință rescrierea fișierului csv asociat platformei din selector și, automat, modificarea listei din interfață. Butonul este indispensabil în ipoteze de natura simplei treceri dintr-un semestru în altul (ori chiar dintr-un an în altul), transferului la altă specializare, apariției întârziate a unei echipe de Teams sau a unui curs pe Moodle, cât și oricărei alte metamorfozări subite ale programei urmate de student.

Cum tranziția de la o stare a listei de cursuri la o alta se produce destul de alambicat și se deosebește radical în scenariul Moodle de scenariul Teams, vom analiza separat firele de execuție:

- a) **Refresh Moodle:** În extragerea oricărei informații de pe platforma Moodle primează comunicarea robotului cu API-ul site-ului. În vederea asigurării acestei comunicări, robotului îi este crucială posesia unui token de securitate care să ateste că studentul este cel care a lansat cererea către API.

Dacă în memoria aplicației nu a fost stocat încă un astfel de token, robotul intră pe pagina principală a site-ului Moodle și urmează demersurile de conectare. Află de la utilizator dacă nu cumva este deja stabilită conexiunea corectă la platformă. În caz contrar, precum ar proceda și studentul manual, apasă butonul Autentificare și bifează sugestia de autentificare prin Microsoft. Dacă studentul nu este conectat nici pe Microsoft, robotul așteaptă până când acesta își introduce datele de sign-in și apasă Ok în fereastra deschisă de Elliot. Odată pătruns pe contul corect de Moodle, robotul apasă pe iconița ce indică pagina de profil a studentului, coborând astfel un meniu, din care selectează elementul Preferences, iar pe urmă dă click pe Security Keys.

Asupra zonei proaspăt răsărite în pagină și afișând chei de securitate, robotul aplică mecanismul de data-scraping, iar din tabelul rezultat extrage nu-

mai cheia pentru mobile API. În acest punct robotul intră în posesia tokenului de securitate căutat, iar studentul este consultat cu privire la stocarea acestui token. Dacă stocarea este dorită de către student, robotul îi cere setarea unei parole cu ajutorul căreia executabilul *encrypt.exe* să creeze tokenul, iar valoarea criptată se reține într-un fișier txt.

Poposind la scenariul în care tokenul a fost extras pe loc de pe site (căci nu exista în fișierul token.txt), indiferent dacă studentul alege să îl salveze sau nu, în pasul următor este apelat executabilul *moodle-api-request.exe* care, cu tokenul extras, trimite un request la API-ul site-ului Moodle. Așadar, date fiind corectitudinea tokenului și desfășurarea neperturbată a comunicării cu API-ul, se va izbuti actualizarea fișierului *coursesMoodle.csv* și, automat, actualizarea afișajului.

Am văzut că, pentru salvarea tokenului de securitate, studentul își setează o parolă. Atunci, intuitiv, pentru accesarea unui token stocat este necesar ca studentul să își introducă parola și să își demonstreze astfel identitatea. Așadar, pe ecran se deschide o fereastră cu un câmp de parolă și următoarele opțiuni:

- **Forgot Password:** Redă implicit execuția firului de copiere a tokenului din secțiunea Security Keys cu toate etapele acesteia.
- **Submit:** Anunță robotul că studentul a terminat de tastat parola. Atunci robotul apelează iar executabilul *moodle-api-request.exe*, de această dată în scopul decriptării tokenului. Introducerea unei parole corecte de către student conduce la decriptarea unui token corect și mai târziu la posibilitatea robotului de a trimite, apelând același executabil, request la API-ul Moodle. Așa se obține din nou actualizarea listei de cursuri. În schimb, introducerea unei parole greșite redirecționează studentul la etapa în care i se cere introducerea parolei pentru decriptarea tokenului.
- **Close:** Închide pur și simplu fereastra de parolă și întoarce studentul la view-ul de până atunci.

### Fișierul token.txt

Fișierul token.txt, stocat intern în aplicație și actualizat la fiecare schimbare de parolă, este format dintr-o singură linie. Pe acea linie se găsește un conglomerat de bytes alcătuit prin criptarea tokenului cu un algoritm de criptare ce implica și "salt"-uirea parolei (de aceea fișierul se modifică simultan cu parola).

Pentru claritate, să considerăm și un exemplu. Fie tokenul de securitate:

hsk6hc0p1m11hc6az9908lopvgd4285j

Și setăm parola:

parola22exemplu

Aplicarea programului *encrypt.exe* acestor două argumente generează un sir de bytes care tradus în caractere arată așa:

ⱡ€b´a‡eç(đëáⱡâPLPĂłquf–ňUâv%ÍNBm´{´´™†^zqRⱡ{ }ⱡsáOxŁøeçⱡ(ţfc°ŌiPĐŬş"ńÇ•â–‘É

- b) **Refresh Teams:** Rescrierea fișierului *coursesTeams.csv* începe cu accesarea paginii principale de pe site-ului Teams. Aici este plasată o stare decizională care divide schema în două căi de atac. Dacă studentul nu este conectat pe Teams, robotul îi cere să își completeze singur username-ul și parola de Microsoft, iar după ce se vede logat să apese Ok pentru ca fluxul să poată continua, unindu-se cu firul principal (cel în care studentul era deja conectat). Apoi robotul mimează acțiunile operatorului uman, dând click succesiv pe Teams, pe Setting și pe Manage Teams, ajungând pe pagina cu înșiruirea tuturor cursurilor. Pe ea face data-scraping, consemnând în csv rezultatul obținut, care, preluat fiind de alt fir de execuție, se actualizează și în interfață.
- (iii) **Selectorul de platformă:** Presupunem că studentul nu știe neapărat pe care dintre cele două platforme (Teams sau Moodle) își postează un profesor suportul de curs și materialele adiționale. De aceea, Elliot îi oferă oportunitatea de a căuta fișiere în ambele medii. Schimbarea platformei se realizează printr-un selector din interfață și se manifestă identic cu apăsarea butonului Refresh.
- (iv) **Butonul Get:** În teorie, studentul apasă Get după ce a selectat atât platforma dorită, cât și cursul pentru care vrea să descarce materiale (căci, prin apăsarea sa, decizia este înregistrată și robotul chemat). Cum logica din spate diferă iarăși în funcție de platforma din selector, discernem aceleași cazuri de mai devreme:
- a) **Get from Moodle:** Studentul este rugat să aleagă din dispozitiv un folder destinație pentru materialele extrase. Dacă nu alege niciunul, o face robotul pentru el, salvând fișierele descărcate în folderul default *data*.

Până la încercarea efectivă de download, care ar putea fi, desigur, întâmpinată de erori (de aici termenul ”încercare”), robotul reiterează pașii de la apăsarea butonului Refresh, respectiv scoaterea tokenului de securitate din fișier, transcrierea lui de pe site în fișier sau utilizarea lui ca atare din site. Totul se petrece la fel ca pentru actualizarea listei de cursuri până în punctul trimiterii requestului cu tokenul către API-ul Moodle inclusiv.

Acum intră în joc id-urile păstrate în csv. De vreme ce studentul a comunicat prin interfață de materialele cărui curs are nevoie, robotul cotrobăie prin csv ca printr-un dicționar și face rost de id-ul acelui curs. În situația fericită în care nu intervine nicio întrerupere a procesului, Elliot lucrează cu id-ul la descărcarea în folderul destinație a materialelor cursului identificat prin acesta.

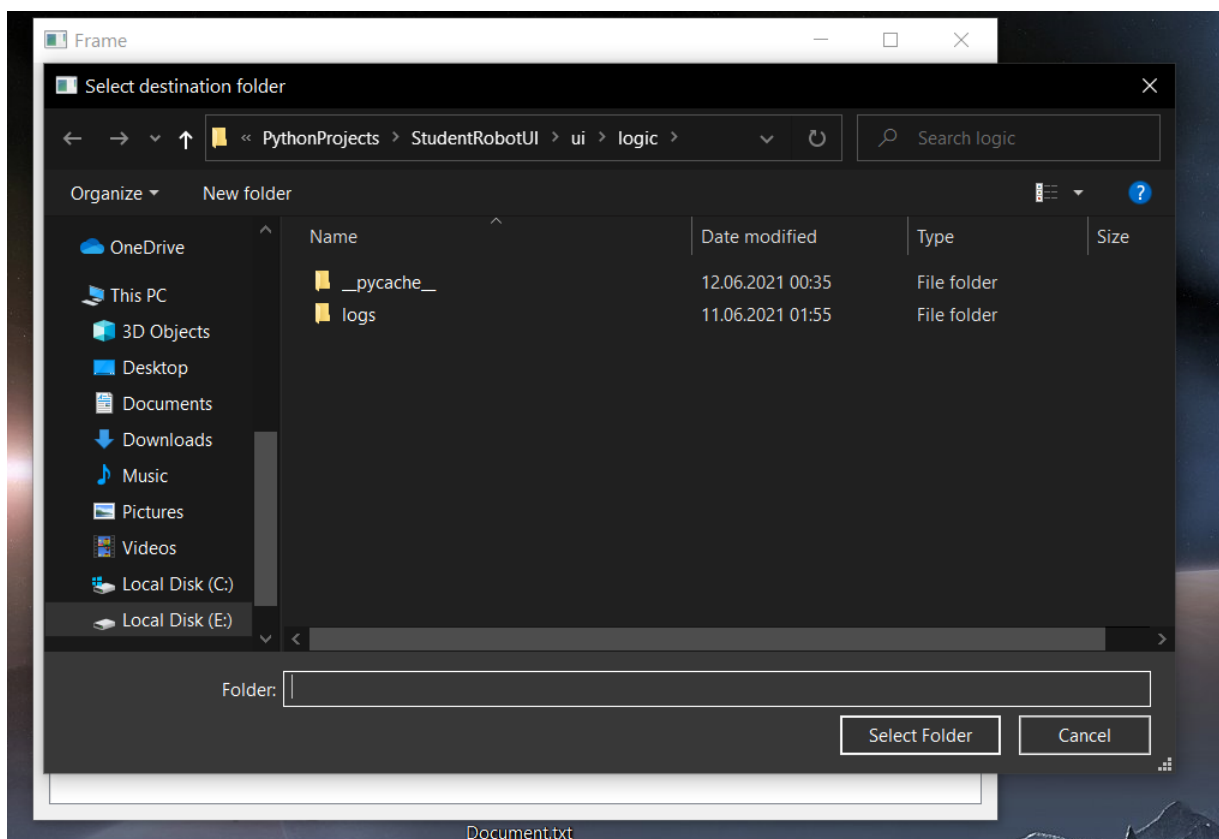


Fig. 2.14: Fereastra de alegere a destinației materialelor de pe Moodle

- b) **Get from Teams:** Și pentru Teams sunt respectate preponderent aceleași demersuri ca la butonul Refresh, încetând, în principiu, să mai semene din punctul de click pe Manage Teams. Preluat, la comanda studentului, din csv-ul aferent platformei, numele cursului este completat de robot în bara de search a paginii curente din Teams.

Rezultatul căutării va fi întocmai echipa dorită, în care robotul va intra pentru o altă scanare. În cadrul acestei noi scanări (data-scraping) vor fi vizate canalele administrate de echipa respectivă, ale căror denumiri vor fi memorate de robot. Cunoscând mulțimea canalelor echipei, robotul le va parcurge. Din fiecare canal, va pătrunde în secțiunea Files și va apăsa *Download*. La finalul procesului, studentul va găsi în folderul Downloads al dispozitivului său toate fișierele aferente cursului dorit existente pe platforma Teams.

Să recapitulăm succint principalele trăsături definitorii ale procesului Materials în raport cu fiecare dintre platforme. Ne interesează asemănările și deosebirile dintre operările cu acestea două din considerente precum metodă de accesare, modalitate de extragere a fișierelor și de salvare a lor.

Numele platformei	Teams	Moodle
Accesarea platformei	prin browserul Microsoft Edge	prin browserul Microsoft Edge
Autentificare pe platformă	prin tokenul de securitate memorat în token.txt (luat de pe site)	făcută manual de către student și dirijată de robot
Extragerea materialelor	trimiterea unui request către API-ul site-ului cu tokenul de securitate	activitatea RPA de data-scraping
Stocarea materialelor	în folderul indicat de student	în folderul default pentru descărcări

- Fig. 2.9: Procesul Grades, pornit la pătrunderea în pagina See Grades (derivată din Courses) și respectând clasică convenție cu switch-ul în diagramă, butonul de Back în interfață și înregistrarea erorilor în fișierul-jurnal, are atuul că inovează în cea mai mare măsură modalitatea prin care un student își află notele și își calculează mediile.

În definitiv, am dezvoltat un traseu complet diferit de cel pe care îl parcurg studenții în prezent de la intenția cunoașterii notei obținute la un examen până la cunoașterea ei faptică. Realitatea actuală arată cam așa: Imediat după susținerea unui examen, studenții așteaptă primirea notelor prin email, fie în forma unui mesaj direct de la profesor, afișându-le numai propria notă (întru respectarea principiului de confidențialitate), fie în forma unui tabel (excel) cu notele întregii serii de la un reprezentant al acesteia (care, la rândul său, l-a dobândit de la profesorul titular). Cea mai rapidă cale de a descoperi o nouă notă este cu siguranță aceea a emailului cu tabelul de note, care însă încalcă principiul de confidențialitate (ușurând slujba profesorului, care nu mai este nevoit să trimită individual notele fiecărui student și nici la fel de predispus la a greși nota comunicată). Unii profesori mai optează în favoarea postării notelor pe Moodle sau Teams, în rubricile dedicate evaluării, și renunțării la email. Din perspectiva profesorului, aducerea notelor la cunoștința studenților constituie numai prima fază din încheierea unei situații școlare, căci, pentru a avea într-adevăr valoare oficială, ele trebuie trimise către secretariat, de unde vor fi puse pe UMS, carnetul virtual al studentului. Există și studenți care nu figurează în grupul de email, echipa de Teams sau programul Moodle, iar pentru aceștia, UMS reprezintă unica resursă, iar așteptarea este obositoare. Dar chiar și

numărându-te printre primele persoane care și-au văzut nota, peste o vreme, după ce nu ți-o vei mai aminti exact și vei avea nevoie să îți calculezi media de pildă, tot la UMS vei recurge (fiindcă un carnet virtual este cum mult mai stabil și mai de încredere decât răsfoirea de diverse platforme și fișiere în căutarea unei note). Ca atare, în diagrama AS-IS am socotit UMS sursa formală a notelor.

Cu toate acestea, niciuna dintre modalitățile existente nu este ideală în ansamblu pentru procurarea notelor. Hibeale uneia în raport cu celelalte sunt compensate de calități pe care le posedă exclusiv. Ce-i lipsește uneia apare în cadrul alteia și viceversa.

Să fixăm un set de proprietăți pe care le așteptăm de la o asemenea modalitate ideală:

- circulație rapidă a notelor de la profesor la studenți;
- șansa studentului de a-și vedea nota la un examen în același timp cu colegii săi;
- respectarea principiului de confidențialitate (un student primește de la profesor numai nota proprie, nu și pe ale colegilor);
- centralizarea fișierelor cu note la toate materiile la care studentul a fost vreodată evaluat (astfel încât să nu fie forțat să-și caute prin locații online distincte nota la o materie anume).

Vom cerceta acum, raportându-ne la acest set, cât de adecvate sunt platformele folosite la momentul actual când vine vorba de vizualizarea notelor. Reamintim variantele conductoare ale procesului implicate în studiu: *Moodle Unibuc*, *Ms Teams*, *UMS Unibuc*, distribuire excel pe grup, email cu nota.

Alternativa propusă în lucrarea de față se declară soluția optimă a comunicării de note, care întrunește toate cele patru atribute specificate:

### Serverul lui Elliot

Un discurs mai amplu cu referire la mediul intitulat *Serverul lui Elliot* va rămâne în seama capitolului rezervat aspectelor tehnice de care se bucură Elliot. Deocamdată trebuie clarificat faptul că nu am construit un server în adevărata accepțiune a cuvântului, ci mai degrabă am simulat existența unuia. Evident, la scară largă se va lucra cu un server real, dar mostra improvizată momentan include o bază de date în care sunt reținute fișiere JSON și satisface intenția prototipică a aplicației. Așadar, ne prefacem că notele au fost plasate de profesori pe server și ilustrăm mecanismul prin care le va replica robotul. Această soluție le depășește pe cele curente în toate cele patru arii considerate, comparația putând fi studiată în tabelul de mai jos:

Platformă	Circulație rapidă a notelor	Primire notei simultan de către toți studenții	Principiul de confidențialitate	Centralizare
Moodle (Grades)	DA platforma sprijină postarea notelor cu ușurință și vizibilitatea instantanee	NU în general lucrările sunt corectate pe rând, iar notele postate pe parcurs (după notarea unei lucrări, profesorul scrie nota și aici)	DA notele asignate unui profil de student sunt vizibile doar pentru proprietarul contului	NU nu toți profesorii aleg această variantă, la unele materii nota se primește pe o cale diferită
Teams (Grades)	DA platforma sprijină postarea notelor cu ușurință și vizibilitatea instantanee	NU în general lucrările sunt corectate pe rând, iar notele postate pe parcurs (după notarea unei lucrări, profesorul scrie nota și aici)	DA notele asignate unui cont MS nu pot fi citite decât din contul respectiv	NU nu toți profesorii aleg această variantă, la unele materii nota se primește pe o cale diferită
Excel cu toate notele pe grup	DA imediat ce termină de notat lucrările și completează excelul pe care oricum îl va trimite la secretariat, profesorul trebuie doar să îl atașeze într-un mail către un grup al studenților	DA toți studenții primesc excelul pe grup în același timp, deci își afla nota simultan	NU fiecare student are acces la excel integral, putând să citească și notele colegilor	NU nu toți profesorii aleg această variantă, la unele materii nota se primește pe o cale diferită
Email personal	DA nu există intermediari între profesor și studenți, nota trece direct prin canalul de comunicare (email)	NU profesorul trimite (manual) pe rând notele fiecărui student, deci unii își vor primi nota înaintea altora	DA fiecare student primește strict propria notă obținută la un examen	NU nu toți profesorii aleg această variantă, la unele materii nota se primește pe o cale diferită
UMS (carnet de note virtual)	NU nota trece de la profesor la secretariat înainte de a fi încărcată pe platformă, iar mijlocirea consumă timp	NU carnetele studenților sunt alterate individual, independent unele de altele	DA studentul nu are acces decât la propriile sale note (la propriul carnet virtual)	DA notele la absolut toate materiile (toți anii, toate semestrele) se găsesc aici
Serverul lui Elliot	DA imediat ce termină de notat lucrările și de completat excelul pe care îl va trimite oricum către secretariat, profesorul încarcă ușor excelul pe serverul lui Elliot	DA odată puse pe server, notele vor fi disponibile pentru toți roboții, fiecare student fiind liber să își afle nota prin Elliot în același timp ca ceilalți studenți	DA robotul unui student poate trimite request către API-ul serverului numai cu tokenul de securitate dobândit la autentificare, primind înapoi doar notele sale	DA în scenariu în care soluția s-ar implementa la nivelul facultății, fiecare profesor ar încărca exceluri cu note pe server (ar fi la un loc)

Table 2.1: Tabel de comparație a metodelor de transmitere note



Etapa incipientă de Log-In pe aplicația *Elliot, Your University Robot* nu a mai fost menționată până acum din motivul irelevanței acesteia în raport cu procesele precedente. De altfel, nici nu ar exista operațiunea Log-In pe Elliot fără proiectarea sistemului cu server de note. Contextul ne cere totuși acum să conturăm raționamentul din spatele conectării la server și, implicit, la robot, așa că asta vom și face.

### Sign-In

Primordială tuturor este înregistrarea utilizatorului pe server (în acest caz ne referim la serverul mimat despre care am scris mai sus). Înregistrarea se va face cu adresa de email a studentului și cu o parolă hotărâtă de el. Parola setată va fi hash-uită (prin funcția hash bcrypt) și stocată în baza de date a serverului (în noua formă). La orice conectare pe server (cu email și parolă corecte), serverul va răspunde cererii de logare cu un token de securitate efemer.

Cu fiecare deschidere a aplicației *Elliot, Your University Robot*, utilizatorului i se va solicita introducerea emailului și a parolei (stabilite pentru server), iar robotul se va conecta automat la server, făcând rost de tokenul de securitate și fiind astfel capabil să acceseze notele cu el. La expirarea tokenului (expirare pentru care îl numim efemer), capacitatea de a vedea notele din server se întrerupe, utilizatorului cerându-i-se iarăși logarea de către aplicație.

Când este invocat, procesul Grades trimite un request către API-ul serverului împreună cu tokenul efemer care i s-a repartizat la ultimul Log-In. Atunci când se deschide pagina See Grades, afișajul implicit din interfață (inclus într-un element de tip view) se alcătuieste din notele strânse în ciclul, anul și semestrul în care aplicația deduce că studentul a fost ultima oară evaluat la o disciplină (deducție făcută pe baza datelor recrutate tot din server).

Cu excepția părăsirii paginii (luată în calcul la expunerea posibilelor evenimente pe care le provoacă utilizatorul), demersurile disponibile în fereastră sunt:

- (i) **Butonul Refresh:** Apăsarea sa dă naștere unui fir de execuție, paralel cu afișarea propriu-zisă a notelor în interfață, care se comportă identic cu firul inițiat la deschiderea ferestrei See Grades (analizat mai sus). Deosebirea se resimte în punctul actualizării afișajului, fiindcă, deși requestul către API cu tokenul temporar coincide, colecția de note afișate corespunde selecției făcute între timp de utilizator (nu se mai iau notele sin cel mai recent triplet ciclu-an-semestru, ci din tripletul setat de student). Separarea disciplinelor examinate și notate în categoriile: afișate (cerute de selecția utilizatorului) și ignorate, are loc pe firul de execuție paralel, după ce toate notele au fost deja preluate din server, iar firul desprins urmează a se reuni cu firul principal.
- (ii) **Selecția ciclu - an - semestru:** Selecția poate fi schimbată de utilizator din interfață (fereastra curentă) ori de câte ori acesta dorește vizualizarea notelor din

trecut și este trată de aplicație întocmai cum este tratată și apăsarea butonului Refresh. Adică la detectarea oricărei modificări din selecție, robotul se leagă iarăși la server, dând start aceleiași succesiuni de pași .

- Fig. 2.7: Fereastra Paperwork diferă de celelalte două ferestre din al doilea nivel al aplicației prin absența opțiunii de redirectionare către un al treilea nivel. Chiar și așa, în ea se instaurează tot două procese independente, pagina fiind împărțită în două secțiuni: Request Document și Send Document, care în linii mari se pliază pe același tipar cu care deja ne-am obișnuit (switch cu opțiunea Back pentru revenirea în fereastra anterioară, tratarea erorilor și raportarea lor în fișierul log).

Secțiunea Request Document semnifică instaurarea procesului RequestDocument și îi pune utilizatorului la îndemână o listă cu documentele a căror eliberare poate fi solicitată. Bineînțeles, documentele sunt aceleași deja enumerate în cadrul schemei AS-IS:

- Adeverință de student;
- Adeverință de licență;
- Situație școlară.

Procesul încorporează un fișier JSON *paperwork.json* (stocat în aplicație) care păstrează adresa de email a secretariatului și subiecte, respectiv texte de email prefabricate și personalizate pentru solicitarea fiecărui act în parte. Datele personale ale studentului, obligatorii pentru prezentarea și identificare sa în instituție, sunt doar simbolizate, prin tokenuri sugestive, iar înlocuirea tokenurilor cu datele reale la trimiterea emailului rămâne datoria aplicației.

### Fișierul paperwork.json

```

1      {
2          "Adeverinta de student": {
3              "email": "secretariat@fmi.unibuc.ro",
4              "subject": "Adeverinta de student",
5              "text": "Buna ziua! \n Sunt <NUME>, student(a) in
                        anul <AN> la Facultatea de Matematica si
                        Informatica, specializarea <SPECIALIZARE>, grupa <
                        GRUPA>. Solicit eliberarea unei adeverinte de
                        student pentru a imi servi la atestarea acestei
                        calitati. \n Multumesc!"
6          },
7
8          "Adeverinta de licenta": {
9              "email": "secretariat@fmi.unibuc.ro",
10             "subject": "Adeverinta de licenta",

```

```
11      "text": "Buna ziua! \n Sunt <NUME>, absolvent(a) la
          Facultatea de Matematica si Informatica,
          specializarea <SPECIALIZARE>, fosta grupa <GRUPA>
          din anul universitar <ANUNIVERSITAR>. Solicit
          eliberarea unei adeverinte inlocuitoare de diploma
          pentru a imi servi la atestarea calitatii de
          licentiat(a) pana la primirea diplomei. \n
          Multumesc!"
12  },
13
14  "Situatie scolara": {
15    "email": "secretariat@fmi.unibuc.ro",
16    "subject": "Situatie scolara",
17    "text": "Buna ziua! \n Sunt <NUME>, student(a) in
          anul <AN> la Facultatea de Matematica si
          Informatica, specializarea <SPECIALIZARE>, grupa <
          GRUPA>. Solicit eliberarea situatiei scolare in
          scopul depunerii interne sau externe a acesteia la
          cererea unei institutii. \n Multumesc!"
18  }
19 }
```

Studentul nu trebuie decât să aleagă din interfață actul pe care și-l dorește, iar apoi să apese **butonul Request**. Odată apăsă, e rândul robotului să intre în scenă. Deschizând fișierul *paperwork.json* și aplicându-i deserializarea, robotul extrage valorile corespunzătoare cheii date de numele documentului ales de student. Apoi, deschizând și fișierul *StudentDataJSON.json*, deserializându-l și extrăgând informațiile particulare studentului, robotul finalizează redactarea emailului traducând tokenurile cheii "text" în elementele potrivite găsite printre acele informații particulare. Subiectul emailului este dat, evident, de valoarea cheii "subject", iar adresa destinatarului de valoarea cheii "email".

- Fig. 2.10: Secțiunea Send Document, cealaltă jumătate a paginii Paperwork, așa cum am explicat mai devreme, declanșează procesul SendDocument atunci când studentul apasă **butonul Send**. Reiterăm lista actelor participante la proces, care va apărea de altfel ca zonă interactivă în interfață:

- Fișă de lichidare;
- Adeverință de cercetare în vederea elaborării lucrării de licență;
- Declarație de autenticitate a lucrării de licență;

- Cererea de înscriere la licență;
- Cerere de cazare la cămin;
- Cerere de echivalare a unui curs;
- Cerere de transfer în altă grupă sau într-un an superior;
- Acte de practică;
- Dovada plății unei taxe.

În mod normal, se află în puterea studentului să își procure șabloane pentru actele ce trebuie completate. Cu cât face rost de mai multe șabloane, cu atât crește ajutorul oferit de automatizare și vom înțelege imediat de ce.

În versiunea curentă a aplicației *Elliot, Your University Robot*, au fost recoltate șabloane ale actelor: cerere de înscriere la licență, adeverință de cercetare în elaborarea lucrării de licență, declarație de autenticitate a lucrării de licență, fișă de lichidare și cerere de echivalare. Actele prezente în lista secțiunii Send Document care nu se regăsesc în această colecție vor fi catalogate drept acte complete (adică acte fără un șablon anume sau acte în care câmpurile au fost deja completate) și nu vor fi pasate procedurii *FillDocument* (umplerea documentului).

### Fișierele-șablon

Fișierele-șablon, salvate sub numele generic *Tip-Scop-Template.extensie*, sunt fișiere *docx* sau *doc* în corpul cărora câmpurile libere (scrise ca spații goale, "....." sau "\_\_\_\_\_") au fost înlocuite cu tokenuri. Pentru claritate, anexăm exemplul:

Doamnă Decan,

Subsemnatul/subsemnata <NUME>, student(ă) în anul III/IV CTI (<AN-UNIV>) la Facultatea de Matematică și Informatică a Universității din București, domeniul de licență <DOMENIU>, specializarea <SPECIALIZARE>, grupa <GRUPA>, forma de învățământ <FORMAINV> vă rog să-mi aprobați susținerea lucrării de licență, în sesiunea <SESIUNE>, cu titlul <TITLU>, profesor coordonator <NUME-PROF>.

Data: <DATA>

Semnătura:

Fig. 2.15: Exemplu de template

Procesul se înființează în momentul în care utilizatorul alege un document din interfață

și comandă trimiterea acestuia. Observăm în diagramă că reapare fragmentul cu deserializarea fișierului *StudentData.JSON.json*. Asemănător cu ce s-a întâmplat și la Request Document, se formulează textul unui email cu datele particulare studentului și documentului figurând ca tokenuri. În continuare, robotul se interesează dacă actul selectat este etichetat drept complet sau incomplet, căci numai actele incomplete vor fi supuse procedurii de umplere.

Unui act incomplet robotul îi deschide din memorie șablonul, plănuiind să înlocuiască tokenurile din acesta cu datele corecte. În mare parte, informațiile esențiale completării documentului au fost deja extrase de robot la pasul privind manevrarea fișierului JSON. Iar acelea lipsă îi vor fi cerute utilizatorului mai departe.

Robotul este implementat în așa fel încât să cunoască de la bun început informațiile date spre completare în corpul documentului care nu se numără printre componentele JSON-ului. De aceea și știe ce anume să ceară ca input în chip de formular. Intrând în posesia răspunsurilor din formular și având așadar toate ingredientele completării șablonului, robotul duce la capăt înlocuirea de tokenuri și afișează rezultatul. Tot printr-un formular, studentul este întrebat dacă umplerea a fost un succes, caz în care documentul este trimis la secretariat ca atașament al unui email având la subiect titlul documentului, iar ca mesaj textul prefabricat. Dacă varianta completată nu îl mulțumește pe student, atunci el este întrebat printr-un alt formular dacă i-ar plăcea să se repete încercarea completării, iar o decizie afirmativă înseamnă de fapt o nouă execuție a aceluiași algoritm.

Ulterior trimiterii prin email a unui document completat corect, studentul mai este întrebat dacă vrea să îl și salveze în computer, în ce format vrea să îl salveze (word sau pdf) și ce folder alege ca destinație. Robotul urmează instrucțiunile acestuia și salvează actul dacă este cazul.

# Capitolul 3

## Tehnologii utilizate în dezvoltarea aplicației

### 3.1 Robotic Process Automation (RPA)

Tehnologia RPA a fost deja prezentată punctual în introducerea lucrării, unde s-a discutat mai ales despre strategiile de luptă ale instituțiilor care și-au integrat-o în sistem. Companiile care apelează la automatizare cu gândul de a-și înmulți, metaforic vorbind, numărul de angajați (prin mâna de lucru a roboților) se bucură de ceea ce singure numesc *"un mediu în care oamenii și roboții lucrează cot la cot"* (din [1]).

Definită și ca preschimbarea software-ului în *"virtualized FTE"* (full-time equivalent), tehnologia se remarcă prin aptitudinea manevrării de aplicații întocmai ca un operator uman. Strămoșii roboților software nu sunt niciunii alții decât macrourele (din macrosubstituție), așa cum afirmă și articolul [1]. Iar, moștenind preceptele de funcționare ale acestor strămoși, roboții software slujesc strict la mânuirea aplicațiilor externe, neavând puterea sau rolul de a le înlocui pe acestea din urmă ori a le manipula codul.

Cu siguranță, o slăbiciune severă a tehnologiei RPA este plaja îngustă de procese ce pot fi automatizate. Numai evenimentele cursive, clare, cu rezultat predictibil pot fi practicate de roboți software, acțiunile complexe sau purtătoare de anumite vulnerabilități revenindu-i persoanei din spatele robotului. Un exemplu de apariție și deopotrivă de combatere a unei slăbiciuni de acest gen în codul lui Elliot este instabilitatea la Sign-In din browser pe contul de Microsoft al studentului. Mulțimea neregulilor, sau cel puțin a fenomenelor neprevăzute, care pot surveni este destul de costisitor de îngrijit. Pe scurt, ne referim la fenomene precum: încercarea de sign-in atunci când contul studentului sau, de ce nu, chiar un cu totul alt cont, este deja conectat (dificultatea robotului de a-și da seama dacă este nevoie de sign-out acompaniat de sign-in, doar sign-in sau niciuna dintre ele); introducerea unei parole greșite (dificultatea robotului de a sesiza o asemenea situație pentru a se redresa); ivirea bruscă a notificărilor în

browser care să perturbe cursul sign-in-ului; nevoia de a reține parola cu o securitate maximă etc. Din cauza acestor incertitudini și altora asemănătoare, am preferat implementarea operațiunii în colaborare cu inputul uman.

Elliot întreabă utilizatorul într-un InputDialog dacă este sau nu conectat la Microsoft (pe contul corect), iar un răspuns negativ cheamă un MessageBox în care utilizatorul este rugat să se autentifice singur:

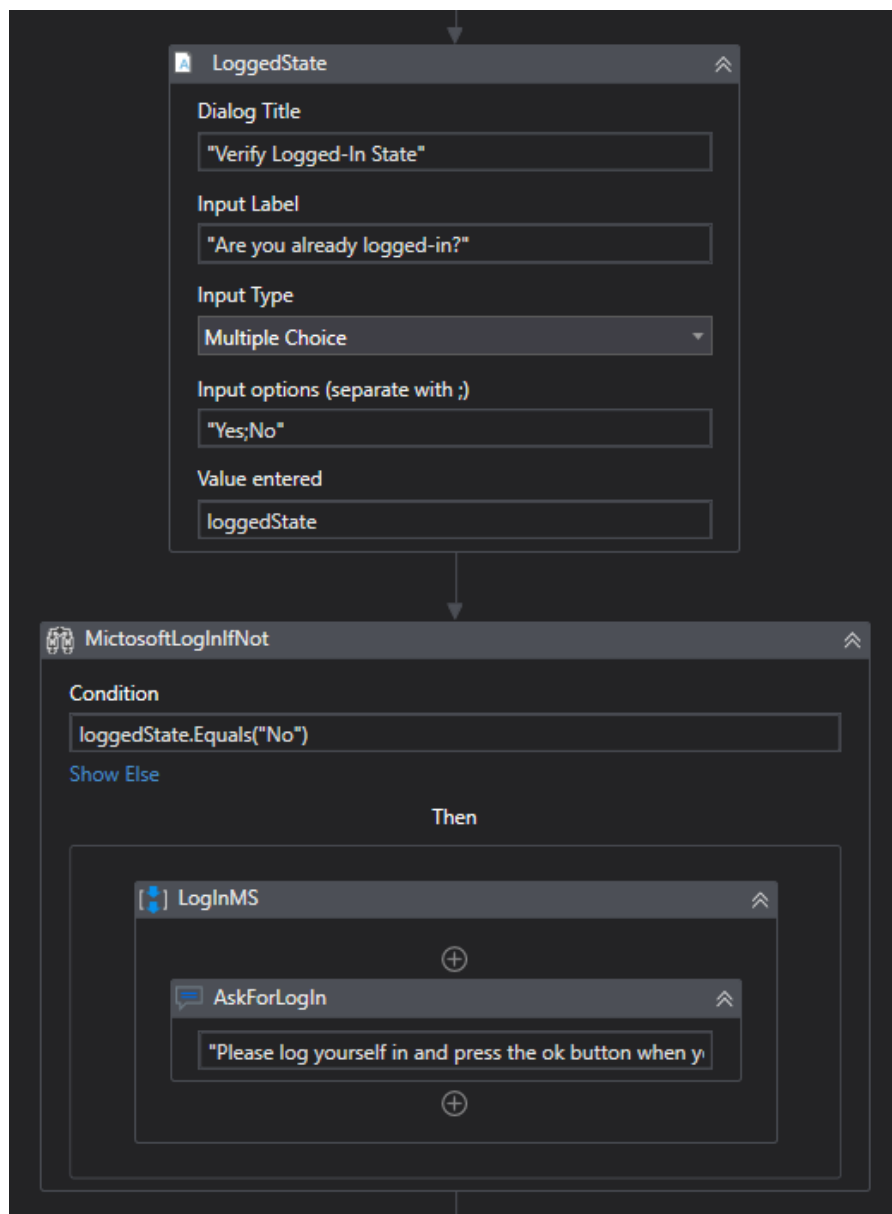


Fig. 3.1: Exemplu RPA - Combaterea slăbiciunii la sign-in Microsoft

În ciuda neajunsului descris, există un avantaj de rezonanță semnificativă: adaptabilitatea roboților software, care determină o cantitate necesară minimă de ajustări făcute proceselor și aplicațiilor implicate în automatizare (din [1]).

## 3.2 UiPath Studio și UiPath Robot

*UiPath Studio* (mediul schematizării) și *UiPath Robot* (executabilul schematizării) alcătuiesc împreună instrumentul de bază prin care software-ul Elliot a fost fabricat.

UiPath Studio se identifică drept un program de dezvoltare a roboților software și este menit proiectării de fluxuri logice (workflows), designului unei hărți pentru întreaga automatizare și stabilirii unei matrice use-case. Studioul este cel cu care utilizatorul interacționează nemijlocit, beneficiind de un UI sugestiv și prietenos.

Între timp, UiPath Robot constituie componenta ascunsă utilizatorului. În calitate de aplicație instalată pe mașina gazdă (fizică sau virtuală), rolul robotului constă în executarea fluxurilor proiectate în studio. Robotul venit împreună cu IDE-ul este *UiRobot.exe*.

Zona pachetelor NuGet reprezintă intersecția dintre studio și robot, fluxurile studioului fiind rulate de robot prin apel la pachete. Pe lângă importul pachetelor, IDE-ul mai permite și invocarea codului în limbaj VB.NET și C# (dintre care numai cel dintâi a fost utilizat la construcția lui Elliot).

## 3.3 VisualBasic.NET (VB.NET)

Cele mai cunoscute limbaje de programare dezvoltate în framework-ul .NET sunt de departe C#, F# și VB.NET. Dintre acestea trei, numai Visual Basic și-a făcut loc în implementarea lui Elliot.

Aparținând Microsoft, VB.NET se face remarcant datorită posturii sale de limbaj orientat obiect și suportat de UiPath (alături de C#). Fragmentele de cod scrise în VB se introduc în sistemul de workflow-uri din UiPath Studio cu activitatea *Invoke code*.

```
Dim Name As String
Name = Surname & " " & Firstname
Text = Text.replace("<NUME>", Name)
Text = Text.replace("<GRUPA>", Grupa)
Text = Text.replace("<SPECIALIZARE>", Specialization)
Text = Text.replace("<AN>", Year)
Text = Text.replace("<ANUNIVERSITAR>", UnivYear)
```



## 3.4 Python

*Python* a devenit celebru în raza limbajelor de programare interpretate datorită însușirilor sale impresionant de comode pentru programatori. Fiind, întâi de toate, un limbaj *open-source*, deci dat în folosință publicului larg, *Python* se laudă cu oferta unui confort inedit pentru orice persoană intenționează să se acomodeze cu programarea multi-paradigmă și orientată obiect. Ușurința înțelegerii noțiunilor și practicilor caracteristice tipului de programare menționat se trage din scurtăturile pe care acest tip i le îndește programatorului tangent la scrierea codului. Ne referim aici la semantica dinamică și nivelul înalt (*high-level*) pe care limbajul le deține.

În proiectarea aplicației *Elliot, Your University Robot* limbajul *Python* joacă un rol de egală însemnătate cu roboții software. Blocurile de cod redactate în acesta, cu ajutorul IDE-ului *PyCharm*, au făcut posibile atât existența unei interfețe grafice (care va fi prezentată mai jos), cât și procesarea răspunsurilor API-ului Moodle prin fabricarea executabilului *process-moodle-api.exe*, care la bază are tot cod scris în *Python* și despre care se vor detalia mai multe în capitolul următor. În plus, la transformarea programelor *Python* în aplicații, s-a recurs la pachetul *PyInstaller*.

## 3.5 PyQt și QtDesigner

Framework-ul *Qt*, scris în *C++* pentru *C++*, are menirea de a dezvolta interfețe grafice cu utilizatorul aplicațiilor Desktop. *PyQt*, varianta din *Python* a aceluiași framework, transferă funcțiile fundamentale *Qt* într-o unealtă externă limbajului, dar compatibilă cu acesta. Un program dezvoltat în *PyQt* se va compune în totalitate din "widget"-uri, adică obiecte încărcate cu funcționalități (cum ar fi butoane, căsuțe de bifat, bară de căutare etc).

O veste încântătoare pentru un dezvoltator *front-end* este că nu mai are obligația de a construi de la zero codul *PyQt* al unei interfețe grafice. Pentru a-i ușura munca, a fost eliberat produsul *QtDesigner*, un *GUI Builder* care vine împreună cu framework-ul și servește la simplificarea organizării interfeței grafice cu utilizatorul pe partea de design.

În spatele transformării schiței profilate cu drag-and-drop de pe *QtDesigner* în cod *Python* specific framework-ului *PyQt* se ascunde un alt utilaj digital extern: executabilul *PyUIC*. Builderul *QtDesigner* memorează, la fel ca *UiPath Studio* de altfel, proiectul sub formă de fișiere *xaml*. Pentru a preschimba fișierele *xaml* în fișiere *py* se execută *PyUIC*.

Toate acestea alcătuiesc sfera uneltelor cu care a fost construită interfața grafică a lui *Elliot*, pe care o vom reliefa în paginile de mai jos (s-a utilizat *PyQt5*).

## 3.6 Go

Limbajul compilat creat de cei de la *Google*, *Go*, se aseamănă din multe puncte de vedere cu *C*, având semantica statică și eficiență la run-time. *Go* apare în implementarea aplicației

*Elliot, Your University Robot* datorită modulului [8], librărie scrisă în și pentru limbajul *Go* și recomandată oficial pe site-ul Moodle, ce are puterea de a interacționa cu API-ul platformei.

Tot în *Go* a fost compus și executabilul *moodle-api-request.exe*, caracterizării căruia lucrarea de față îi dedică un subcapitol ulterior. Codul executabilului adaugă în modulul pomenit (i.e. [8] din bibliografie) o funcție suplimentară care extrage conținutul cursurilor de pe Moodle. De asemenea, aplicația de criptare a tokenului de securitate, *encrypt.exe*, este programată în *Go* din considerente de uniformitate la transferul tokenului criptat în requestul la API-ul Moodle.

## 3.7 Node.js

Mediul de execuție *Node.js*, deopotrivă *open-source* și *cross-platform*, este apreciat la scară largă datorită abilității sale de a permite rularea codului de *JavaScript* în afara unui browser, profitând pe deplin de performanța motorului V8.

Simularea serverului de pe care se presupune că Elliot preia notele studentului-proprietar a fost atribuită pachetelor *json-server* și *json-server-auth* din *npm* (*Node Package Manager*). Autentificarea se produce prin intermediul unor *http-request-uri* așa cum am specificat deja în secțiunea *Log-In* a diagramei Grades TO-BE.

Iată câteva exemple de *http-request-uri* cu răspunsurile lor:

```
POST /login/ 200 100.362 ms — 208
POST /login/ 400 101.831 ms — 20
POST /login/ 400 97.672 ms — 20
POST /login/ 200 102.950 ms — 208
POST /login/ 200 100.927 ms — 208
POST /login/ 200 117.306 ms — 208
POST /login/ 200 121.225 ms — 208
POST /login/ 200 133.696 ms — 208
POST /login/ 200 111.074 ms — 208
GET /__rules 200 4.564 ms — 96
GET /db 200 17.734 ms — —
```

# Capitolul 4

## Arhitectura aplicației

### 4.1 Interfața grafică

Am promis în rândurile anterioare o aducere în prim plan a interfeței grafice cu utilizatorul. A sosit timpul să ne ținem de cuvânt și să croim itinerarul paginilor de care *Elliot, Your University Robot* dispune în GUI.

#### 4.1.1 Galeria paginilor

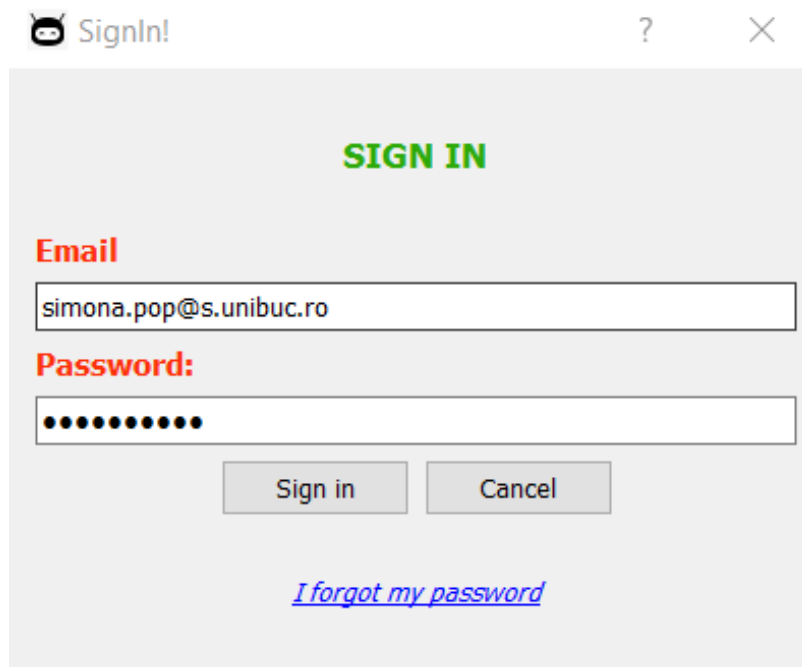


Fig. 4.1: Fereastra de Sign-In

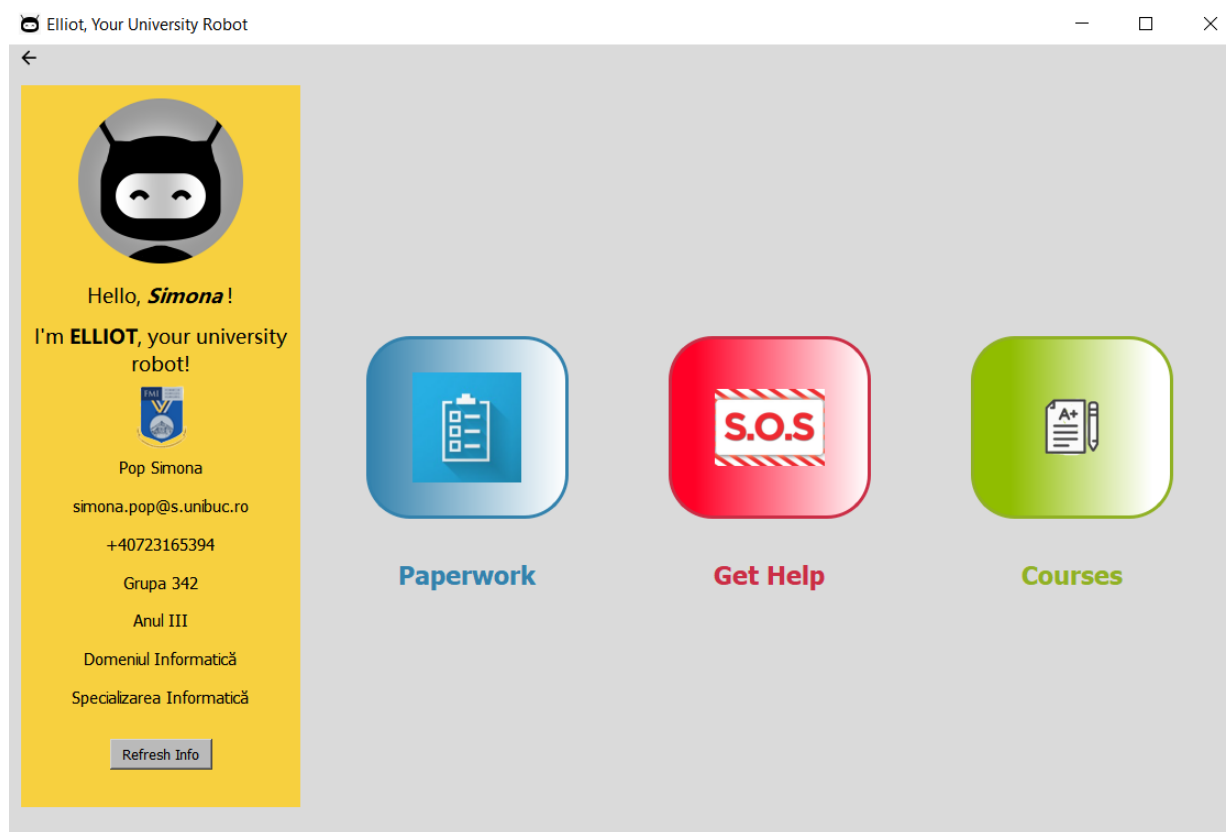


Fig. 4.2: Pagina de pornire

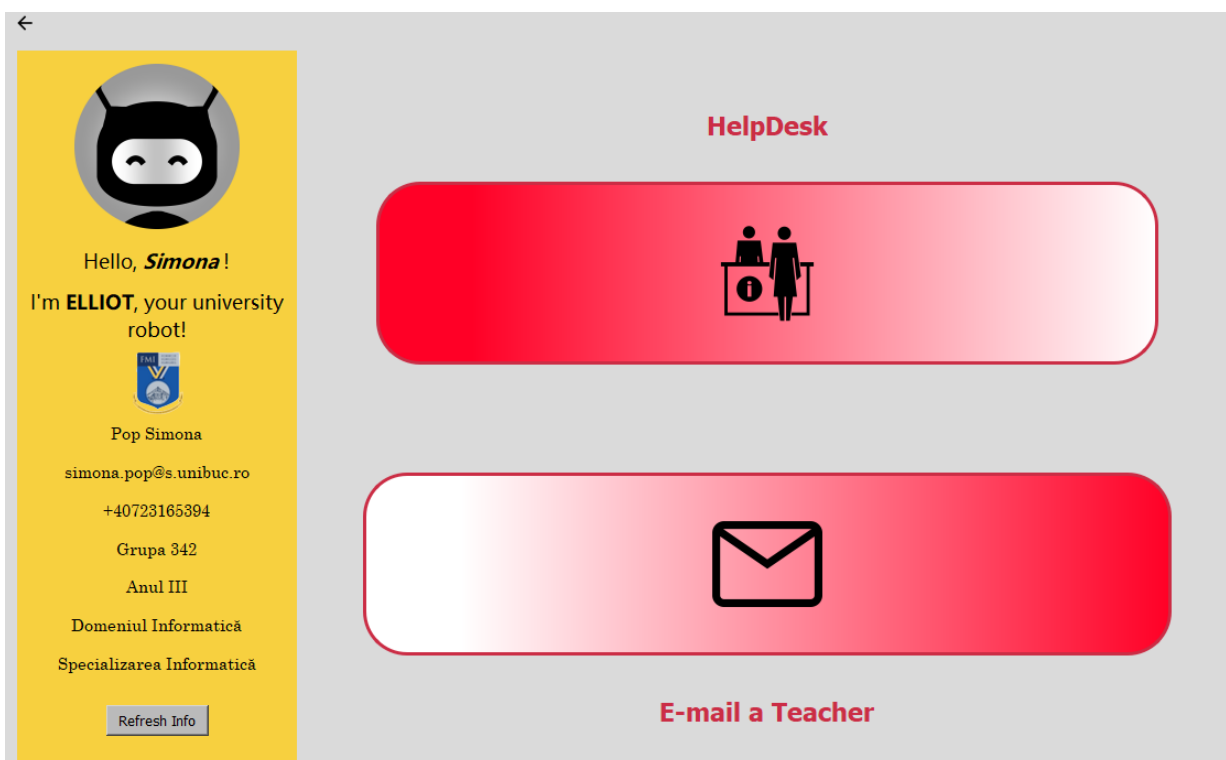


Fig. 4.3: Pagina GetHelp

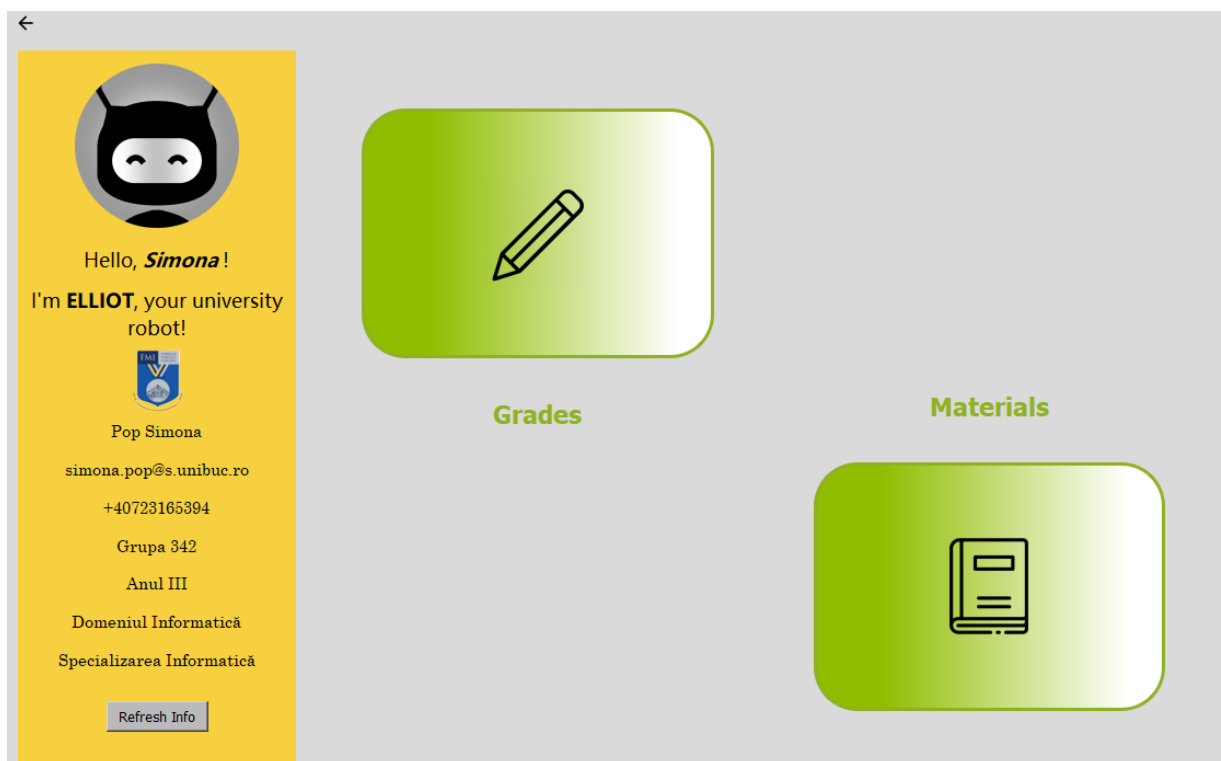


Fig. 4.4: Pagina Courses

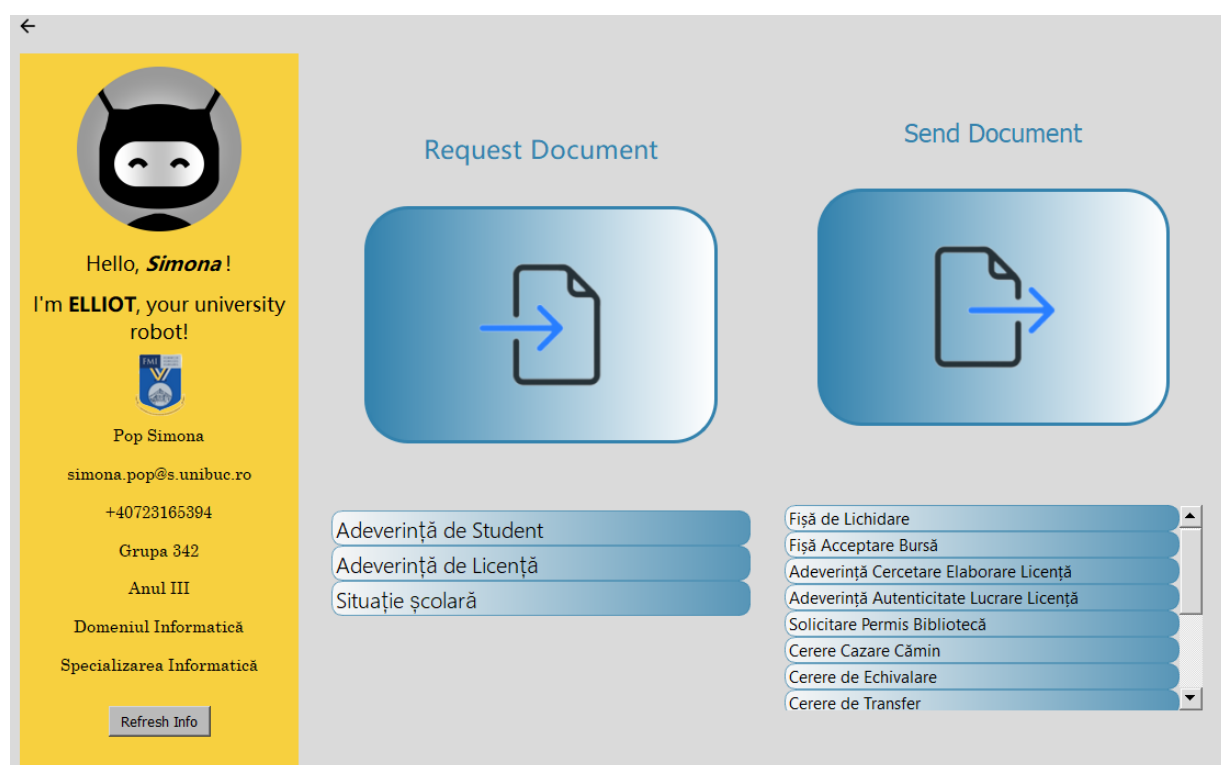


Fig. 4.5: Pagina Paperwork

The screenshot shows the 'HelpDesk' interface. On the left, a yellow sidebar contains a robot avatar, the text 'Hello, **Simona**!', 'I'm **ELLIOT**, your university robot!', a university logo, and contact information for Pop Simona (email: simona.pop@s.unibuc.ro, phone: +40723165394, group: Grupa 342, year: Anul III, department: Domeniul Informatică, specialization: Specializarea Informatică). A 'Refresh Info' button is at the bottom of the sidebar. The main area has a 'HelpDesk' title and a small robot icon. It prompts the user to 'Introduce a name for your ticket:' with a text input field, and then 'Write down below the text of your new ticket:' with a large text area. At the bottom, there are two buttons: 'Attachments' and 'Finish ticket', and a link: '→ Or go to see the status of your older tickets'.

Fig. 4.6: Pagina HelpDesk

The screenshot shows the 'HelpDesk' interface for viewing old tickets. On the left, the same yellow sidebar as in Fig. 4.6 is present. The main area has a 'HelpDesk' title and a 'Reload list of tickets' button. Below this, there are two tabs: 'Open Tickets' and 'Closed Tickets'. The 'Open Tickets' tab is active, showing a table with the following data:

Ticket #	Create Date	Status	Subject	Department
948	20.11.2020	Inchis	URGENT!	Anul III

Below the table, there is a large empty rectangular area.

Fig. 4.7: Pagina See Old Tickets

**E-mail a Teacher**

Introduce a title for your problem as mail subject:

Introduce here the text describing the problem you want to adress to a teacher:

Refresh Attachments Send

Browse:

Prof.dr. Marian Aprodu  
 Prof.dr. Cristian Bereanu  
 Prof.dr. Lucian Beznea  
 Prof.dr. Daniel Bulacu  
 Prof.dr. Aurelian Cernea  
 Prof.dr. Mihai Cristea  
 Prof.dr. Sorin Dăscălescu  
 Prof.dr. Liviu Marin  
 Prof.dr. Ion Mihai  
 Prof.dr. Gigel Militaru  
 Prof.dr. Liviu Ornea – director departament  
 Prof.dr. Ionel Popescu  
 Prof.dr. Gabriel Priopae  
 Prof.dr. Dragoș Ștefan  
 Prof.dr. Victor Vuletescu  
 Conf.dr. Cornel Baetica  
 Conf.dr. Cristian Cazacu  
 Conf.dr. Iulian Cîmpean  
 Conf.dr. Tiberiu Dumitrescu  
 Conf.dr. Cătălin Gherghe – prodecan  
 Conf.dr. Alexandru Gica  
 Conf.dr. Iulia Hirica  
 Conf.dr. Cristian Niculescu

Fig. 4.8: Pagina Email Teachers

**Materials**

Refresh Browse: Moodle GET

Algoritmica grafurilor  
 Calcul numeric - Seria 34  
 Competențe de bază într-o limbă străină (engleză)  
 Competențe de bază într-o limbă străină (franceză)  
 Competențe specifice într-o limbă străină (engleză)  
 Competențe specifice într-o limbă străină (franceză)  
 Criptografie și securitate (2021)  
 Dezvoltarea aplicațiilor Web - Seria 34  
 Ecuații diferențiale și cu derivate parțiale, IF - Seria 34  
 Educație fizică  
 Ingineria programării - Seria 34  
 Metode de dezvoltare software - Seriile 23 si 24  
 Printare și modelare 3D  
 Programare declarativă - Seria 33  
 Programare declarativă - Seria 34  
 Sisteme de Gestiune a Bazelor de Date (INFO, S33&S34)  
 Tehnici de compilare - Seria 34  
 Tehnici de optimizare - Seria 34  
 Tehnici de simulare - Seria 34

Fig. 4.9: Pagina Materials

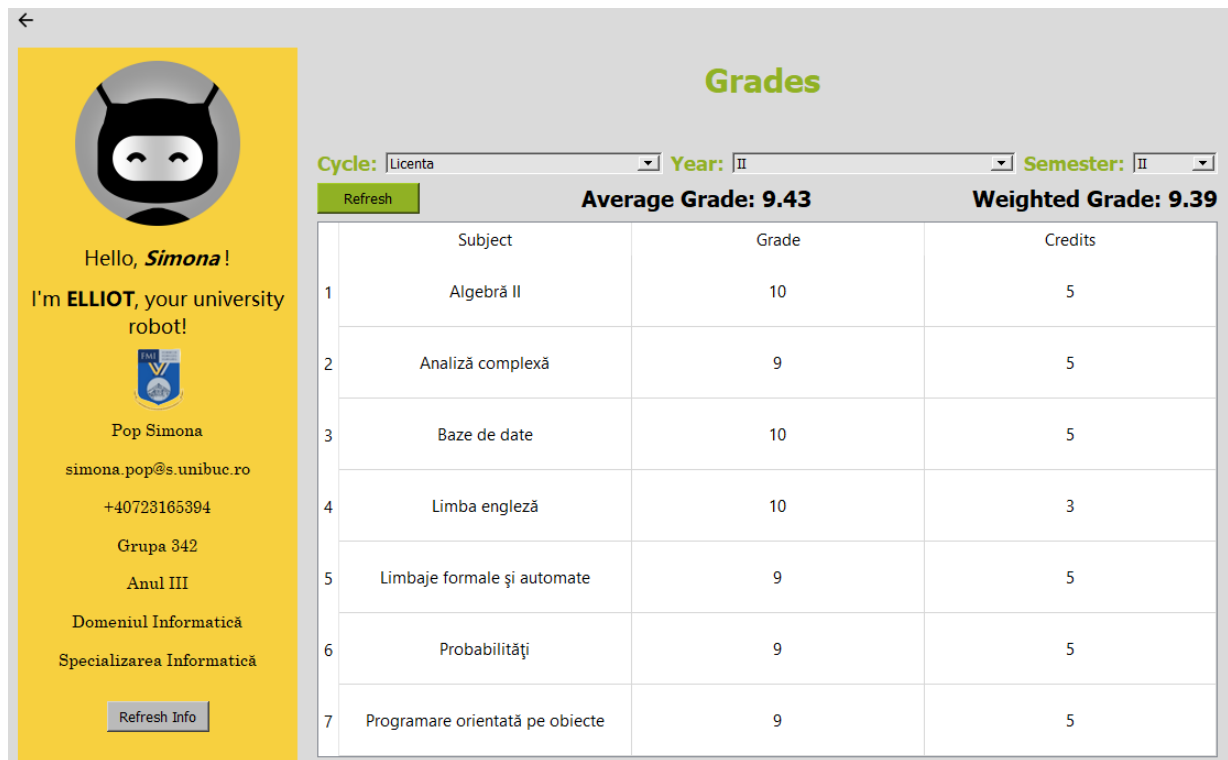


Fig. 4.10: Pagina Grades

### 4.1.2 Harta paginilor

Toate paginile interfeței înglobează în structura lor o panglică de prezentare a robotului lăsat în slujba studentului autentificat. Pentru acuratețe la gradul implementării tehnice, ar fi mai adecvat să spunem că panglica, similar butonului Back, întemeiază elementul GUI fixat, întotdeauna ieșit înaintea paginii curente. Panglica, poziționată în stânga ecranului, se alcătuiește din logo-ul *Elliot, Your University Robot*, sigla facultății la care studiază utilizatorul și date despre dânsul (numele complet, adresa de email, numărul de telefon, grupa, anul de studii, domeniul și specializarea). Adicional relatării datelor studentului, panglica găzduiește **butonul Refresh Info**, cu care (apăsându-l) sunt înnoite datele respective. Prin urmare, sesizând că o anumită informație a devenit neconformă cu realitatea, studentul apasă butonul, expediind astfel o cerere către server de pescuire a datelor (la zi) din acesta.

- Fereastra care întâmpină utilizatorul la deschiderea aplicației, anexată ca 4.1, îi solicită acestuia autentificare cu adresa de email și parola stabilită atunci când s-a înregistrat pe serverul de note. Dependența lui Elliot de server privitoare la intrarea în cont a fost explicată integral în Capitolul 2.

### Baza de date

Ca aplicația să poată rula, se impune pornirea serverului improvizat (un server real ar fi deschis permanent). Iată comanda de terminal care duce la deschiderea acestuia:



```
> npm run json:server
```

Baza de date a serverului, inscripționată în format JSON, reține (în teorie) conturile tuturor studenților membri ai programului robot universitar și notele din fișierele excel încărcate de profesori pe server, filtrate în funcție de ciclu, an și semestru. Dar, cum serverul este doar simulat deocamdată, au fost create (în practică) două conturi de student drept exemple și portofoliile lor de note introduse manual.

```
1  "users": [  
2  {  
3    "id": 1,  
4    "password": "$2a$10$PsS/f55zlbjSJAg/qMx0zuTl7y.snSTiQTb  
      8e6jomt7VgcERQpKEK",  
5    "email": "simona.pop@s.unibuc.ro",  
6    "Surname": "Pop",  
7    "FirstName": "Simona",  
8    "Phone": "+40723165394",  
9    "Class": "342",  
10   "Year": "III",  
11   "UniversityYear": "2020-2021",  
12   "Domain": "Informatica",  
13   "Specialization": "Informatica",  
14   "StudyForm": "IF",  
15   "FinanceType": "taxa",  
16   "beginYear": "2017",  
17   "endYear": "2020",  
18   "faculty": "Matematica-Informatica",  
19   "cycle": "Licenta",  
20   "registrationNumber": "133/2017",  
21   "finished": 0,  
22   "gradesId": 1  
23 }
```

```
1  "grades": [  
2  {  
3    "id": 1,  
4    "userId": 1,  
5    "Licenta": [  
6      {  
7        "Year": "I",
```

```
8      "Semester I": [  
9          {  
10             "Subject": "Analiza matematica I",  
11             "Grade": 9,  
12             "Credits": 7  
13         },  
14         {  
15             "Subject": "Algebra I",  
16             "Grade": 7,  
17             "Credits": 7  
18         },  
19         { "Subject": "Geometrie I", "Grade": 10, "Credits  
20             ": 7 },  
21         { "Subject": "Logica matematica si teoria  
22             multimilor", "Grade": 9, "Credits": 5 },  
23         { "Subject": "Programare procedurala", "Grade": 9  
24             , "Credits": 4 }  
25     ],  
26     "Semester II": [  
27         {  
28             "Subject": "Analiza matematica II",  
29             "Grade": 8,  
30             "Credits": 7  
31         },  
32         {  
33             "Subject": "Algoritmi paraleli",  
34             "Grade": 10,  
35             "Credits": 7  
36         },  
37         {  
38             "Subject": "Algoritmica grafurilor",  
39             "Grade": 9,  
40             "Credits": 5  
41         },  
42         {  
43             "Subject": "Geometrie II",  
44             "Grade": 9,  
45             "Credits": 7  
46         },  
47     ]
```

```

44         {
45             "Subject": "Introducere in software matematic",
46             "Grade": 10,
47             "Credits": 4
48         },
49         {
50             "Subject": "Limba engleza",
51             "Grade": 10,
52             "Credits": 3
53         }
54     ], ...
55
1     "relations": [
2     { "userName": "Pop-Simona", "userId": 1, "gradesId": 1 },
        ...

```

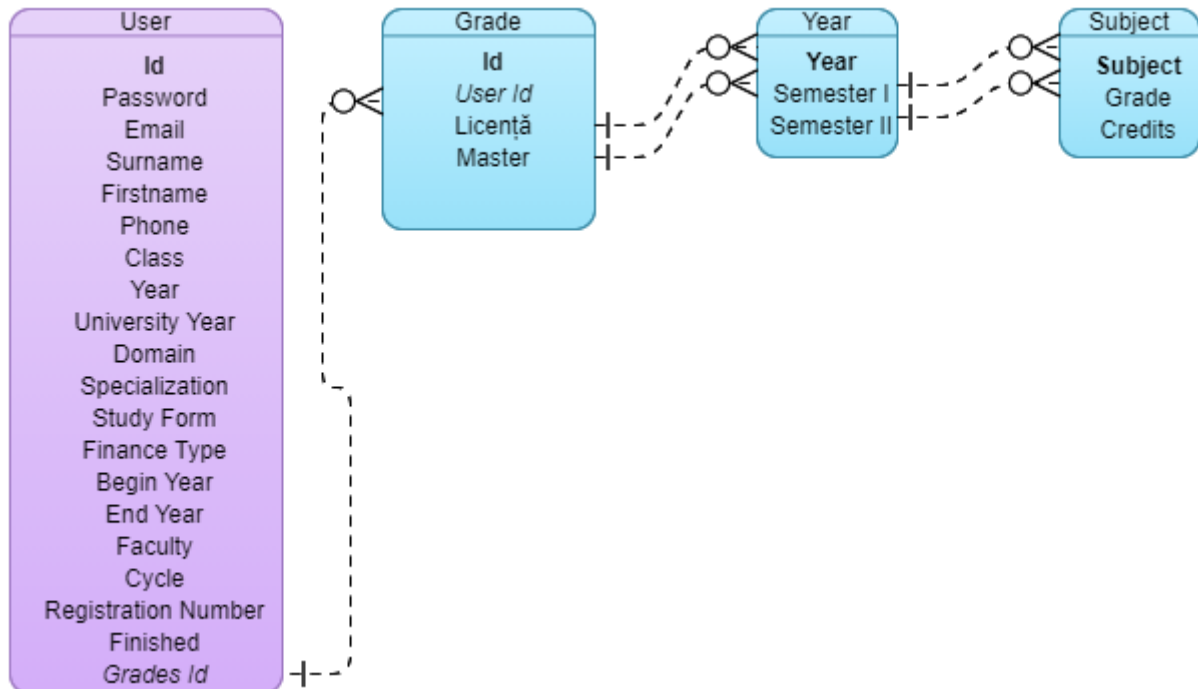


Fig. 4.11: Modelul bazei de date

- Despre pagina de pornire, ilustrată în figura 4.2, și paginile de pe al doilea nivel (figurile 4.3, 4.4, 4.5), nu au rămas prea multe de zis. Construite preponderent din bu-

toane, înfățișările lor sunt suficient de sugestive pentru înțelegerea mecanismului de funcționare, care a fost oricum explicat în analiza TO-BE.

- Figura 4.6 reflectă câmpurile pentru introducerea subiectului, respectiv textului tichetului, din pagina HelpDesk, și butoanele destinate atașării de fișiere la noul tichet, finalizării lui și consultării stărilor în care se găsesc tichetele mai vechi. Figura 4.7 înfățișează view-ul cu tichete create în trecut, clasificate în două file: închise sau deschise. Evident, butonul Reload caută orice schimbări suferite de statusurile tichetelor din listă.
- În figura 4.8, se observă cum pagina Email Teachers este secționată în partea dedicată inputului studentului (subiectul și corpul mailului) și partea opțiunilor de adresă destinație. Sunt surprinse și facilitățile speciale aclamate în analiza TO-BE, adică atașarea de fișiere emailului, actualizarea listei de profesori, foarte utilă în cazul în care survin modificări ale catedrelor, și caseta de căutare (ca studentul să găsească mai rapid adresa dorită).
- Imaginea paginii Materials, figura 4.9, întărește ideea cititorului despre procesul analizat de către Capitolul 2. În aceasta, selectorul este comutat pe platforma Moodle, deci titlurile disciplinelor din view coincid cu cele de acolo.
- Aruncând o privire peste figura 4.10, putem constata că mediile generală, respectiv ponderată (cu numărul de credite), ale notelor obținute de studentul autentificat pe aplicație în anul II de licență, semestrul al doilea, sunt calculate corect de Elliot, deasupra tabelului principal. Ori de câte ori este apăsat butonul Refresh sau este modificată valoarea vreunui selector (Cycle, Year, Semester), se pornește un thread (*worker* în terminologia *PyQt*) din fișierul *workers.py* care, pe lângă vizita serverului pentru extragerea notelor, inițiază și calculul mediilor:

```
avgGrade = sum([g[0] for g in nonZeroGrades]) / len(
    nonZeroGrades)
weightGrade = sum(g[0] * g[1] for g in nonZeroGrades) /
    sum([g[1] for g in nonZeroGrades])
```

### 4.1.3 Folderul cu utilități

Folderul *utils* care agregă la programul scris în *Python* pentru interfața grafică este alcătuit din fișiere auxiliare fără de care GUI-ul nici nu ar arăta, nici nu s-ar comporta așa cum o face. Printre utilitățile aferente acestuia se numără funcția de logare a erorilor, funcția de ajustare a logo-ului Elliot, modelele view-urilor de pe paginile pe care sunt plasate astfel de widgeturi și fișierul cu *workeri*.

### Workerii din `workers.py`

Un *worker*, asemuit adesea cu un thread, este un proces care se execută în fundalul unei aplicații, astfel încât să nu îngreuneze firul principal al acesteia.

Fișierul *workers.py* din folderul *utils* al interfeței noastre este dotat cu funcții pentru procesarea datelor studentului din *StudentDataJSON.json*, extragerea organizată a notelor din server (conformă cu selecția), procurarea cursurilor de pe Moodle și apelarea roboților din UiPath. Drept exemplu vom oferi această funcție din urmă, probabil cel mai important worker GUI în cazul lui Elliot.

```
class CallUipathRobotWorker(QtCore.QObject):
    finished = QtCore.pyqtSignal(bool)

    def __init__(self, args: dict):
        super(CallUipathRobotWorker, self).__init__()
        self._args = args

    def run(self):
        roboCommand = "cmd /c " + getRobotPath() + " execute --file "
        " + abspath(r".\uipath\Main.xaml") + " --input " + "\"" +
        str(
            self._args) + "\""
        # print(roboCommand)
        returnCode = Popen(roboCommand, stdout=PIPE, stderr=PIPE)
        stdout, stderr = returnCode.communicate()
        if stderr:
            # print("Robot Error: ", stderr.decode('utf-8'))
            err = stderr.decode('utf-8')
            with open('./logs/uipath.log', 'a', encoding='utf-8') as
                logFile:
                    logFile.write(f" {datetime.datetime.now()} - {err} \
                        n")
        if stdout:
            # print("Robot Output: " + stdout.decode('utf-8'))
            out = json.loads(stdout.decode('utf-8'))['robotSuccess']
            self.finished.emit(out)
        else:
            self.finished.emit(False)
```

## 4.2 Executabilele pentru Moodle

Am adus adesea în discuție, cu diverse ocazii, cele trei executabile vitale procesului Materials (mai precis descărcării materialelor de pe Moodle), iar acum le vom cerceta mai îndeaproape. Debutând ca programe rudimentare în *Go* și *Python*, ele au fost convertite în fișiere tip *exe* cu ajutorul comenzii *go build* (care compilează codul Go și îl aduce în formă de executabil), respectiv a pachetului *PyInstaller*.

### (i) **encrypt.exe:**

Programul *encrypt.go*, compus în cod de *Go* cules din sursa [3] (din bibliografie), dar prelucrat și adaptat la nevoile lui Elliot, servește stocării sigure a tokenului de securitate de aplicație din contul Moodle al studentului. Programul recepționează tokenul asociat aceluși cont, parola setată de utilizator pentru comunicarea cu API-ul și calea folderului destinație pentru salvarea tokenului criptat (dată tot de utilizator). Făcând rost de aceste trei ingrediente, prepară în continuare criptarea după rețeta de mai jos:

- transformă tokenul clar și parola în bytes (vom numi șirurile de bytes rezultate în urma transformării: *text* - obținut din token și *key* - obținut din parolă);
- generează un *salt* cu lungimea de 16 bytes;
- aplică funcția de derivare **PBKDF2** (Password-Based Key Derivation Function) cheii *key* și *salt*-ului chemând de 4096 de ori funcția hash **SHA-256** și ajungând la un nou șir de 32 de bytes;
- criptează *text* cu algoritmul **AES** (Advanced Encryption Standard) în modul de operare **GCM** (Galois Counter Mode), cu *nonce*-ul generat aleator și cu cheia obținută din *key* și *salt* la pasul anterior.

Să notăm acum câteva noțiuni teoretice care vor clarifica termenii anteriori.

Algoritmul **AES**, standardul actual *NIST*, determină un sistem de criptare cu cheie simetrică echivalând cu o rețea substituție-permutare pe 128 de biți, al cărei număr de runde variază proporțional cu lungimea cheii. Practic, textul predat criptării AES este fragmentat în blocuri de câte 128 de biți, iar atât el cât și cheia sunt reprezentați ca matrice de 4x4. Starea mesajului inițial este modificată de-a lungul rundelor prin transformări intitulate convențional AddRoundKey, SubBytes, ShiftRows și MixColumns, iar mesajul criptat devine chiar ieșirea din ultima rundă.

Modul de operare **GCM** presupune participarea unui vector inițial la numerotarea blocurilor-cifru și criptarea rezultatelor numerotării cu cifrurile respective. Criptările acestea sunt *XOR*-ate cu fragmente din textul clar, iar blocurile de biți din final se compun în mesajul criptat.

De la momentul apariției și până în prezent, familia de funcții hash **SHA-2** a rămas considerată sigură în plan oficial și este încă recomandată de *NIST*. Un membru popular al acestei familii este funcția **SHA-256**, care însă nu se prea mai folosește de una singură din pricina lipsei de integritate oferită datelor primite ca argumente.

Funcția pseudo-aleatoare **PBKDF2** se ocupă cu derivarea unei chei prin hash-uire repetată (de un număr specificat de ori) a combinației dintre ea și o valoare *salt*. Aplicarea acesteia reduce întotdeauna vulnerabilitatea sistemului la atacurile cu forță brută.

(ii) **moodle-api-request.exe**:

S-ar cuveni să percepem fișierul *moodle-api-request.go* ca suplimentarul celui de mai sus întrucât definește funcția inversă aceluia, adică decriptarea. Nu trebuie să scăpăm din vedere faptul că lucrăm în sisteme de criptare cu cheie simetrică, așa deci operația de decriptare nu îmbogățește cu nimic ceea ce am făcut cunoscut în raport cu operația de criptare. În plus față de inversa funcției din *encrypt.go*, programul *moodle-api-request.go* încorporează și subprogramul prin care sunt scoase id-urile titlurilor de cursuri salvate în csv-ul despre care s-a vorbit deja.

Mai mult decât atât, de interes major fișierului este porțiunea în care se trimite request înspre API-ul Moodle cu tokenul de securitate proaspăt decriptat:

```
api := moodle.NewMoodleApi("http://moodle.unibuc.ro//  
webservice/rest/server.php", tokenValue)
```

Finalizarea acestei cereri este însoțită de două abordări alternative una față de cealaltă: setul de instrucțiuni pentru solicitare de tip Refresh și cel pentru Get (Get primind ca parametru id-ul cursului selectat de student din interfață).

(iii) **process-moodle-api.exe**:

Ultimul program al seriei de unelte construite pentru descărcarea materialelor de pe platforma Moodle, *process-moodle-api.py*, primește ca argumente răspunsul API-ului site-ului la cererea trimisă și calea folderului destinație, stabilită de utilizator. Informațiile conținute în răspunsul dat de Moodle sunt filtrate de executabil, acesta ignorând rubricile cu numele "Temă"/"Teme" ori "Proiect"/"Proiecte" (căci se pleacă de la premisa că acestea nu conțin materiale de studiu).

Rubricilor rămase la sfârșitul excluderii li se creează câte un director, iar fișierele găsite în rubricile respective sunt salvate în format pdf în interiorul directoarelor core-spunzătoare. Codul *Python* ce stă la baza executabilului arată astfel:

```
import json  
import sys  
from pathlib import Path
```

```

import requests

if __name__ == '__main__':
    if len(sys.argv) != 3:
        raise Exception("There must be 2 arguments: 1 – the
            moodle result file , 2 – the download destination path
            ")
    path = sys.argv[1]
    materiale = Path(sys.argv[2])
    with open(path, "r") as jFile:
        response = json.load(jFile)
    for item in response:
        if item['name'].strip() and "tem" not in item['name'].
            lower() and "proiect" not in item['name'].lower() and
            item['uservisibile']:
            p = materiale / item['name']
            p.mkdir(parents=True, exist_ok=True)
            for module in item['modules']:
                if 'contents' in module:
                    for content in module['contents']:
                        if 'fileurl' in content:
                            try:
                                request = requests.get(content['
                                    fileurl'], stream=True,
                                    allow_redirects=True)
                                filepath = p / content['filename
                                    ']
                                with open(filepath, "wb") as
                                    Pypdf:
                                    for chunk in request.
                                        iter_content(chunk_size
                                            =1024):
                                            if chunk:
                                                Pypdf.write(chunk)
                            except requests.exceptions.
                                RequestException as e:
                                    raise e

```



### 4.3 Flowchart al roboților software

Am arătat mai devreme cum se leagă interfața grafică de roboții software dezvoltati în UiPath. Insistăm acum pe felul în care aceștia sunt legați între ei. Iar în această direcție, flowchartul soluției software se va dovedi cel mai capabil îndrumător:

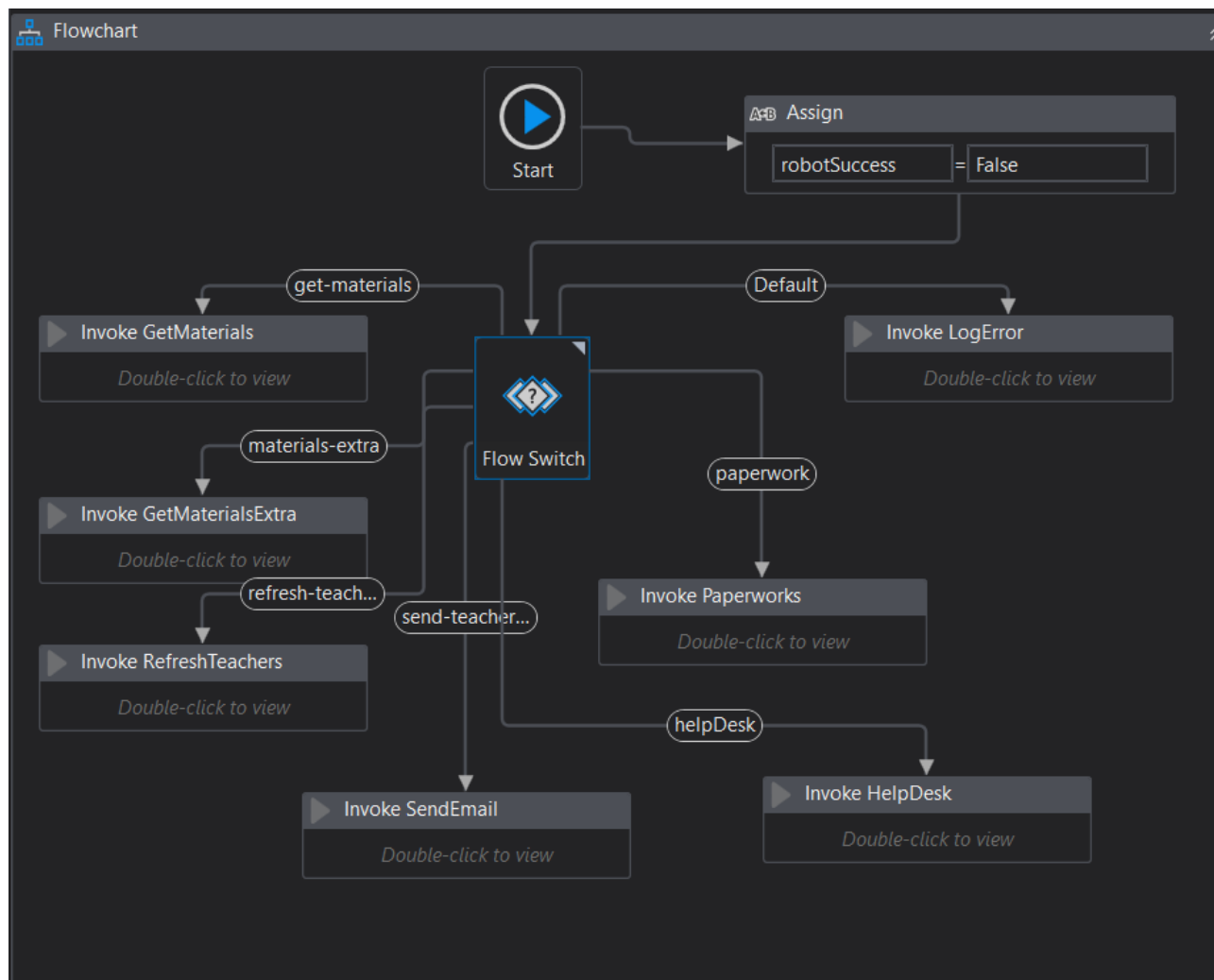


Fig. 4.12: Main Flowchart

Remarcăm cum scheletul automatizării se rezumă la un switch care invocă în parte roboții, depinzând de procesul solicitat. Desigur, acesta nu este unicul flowchart peste care s-a fondat logica lui Elliot. Fiecare proces din figura 4.12, la rândul său, se divide pe mai multe ramuri (ale unui switch) care corespund în mare măsură stărilor decizionale din diagrama TO-BE.

## 4.4 Dependințe

Funcționarea integrală a aplicației *Elliot, Your University Robot* se leagă strâns de fauna serviciilor de care studentul beneficiază în avans. Dependințele astfel definite vor fi enumerate în continuare.

Elliot reprezintă un program adânc înrădăcinat în solul produselor *Microsoft*. Aplicațiile de care depinde modul său de acțiune sunt:

- *Sistem de operare Windows pe 64 biți, orientativ versiunea Windows 10;*
- *UiPath Studio, minim versiunea 18.4:* pentru execuția de atribuții ale roboților software;
- *Microsoft Office Word, începând de la 2007:* strict pentru completarea actelor;
- *Microsoft Office Outlook, minim versiunea 16.0:* pentru serviciul de email;
- *Microsoft Edge, orientativ versiunea 91.0:* în calitatea de browser.

# Concluzie

## 4.5 Orizonturi și perspective de progres

S-a tot afirmat și confirmat pe parcursul lucrării de față că *Elliot, Your University Robot* îndeplinește pentru moment o funcție experimentală. Rolul său demonstrativ, împletit cu alocarea minimă de resurse din partea instituției competente, tinde la un potențial de viitor ridicat.

Capitolele de până acum au lămurit, fără îndoială, cititorul că distanța de la configurația actuală a aplicației până la cea dezirabilă pentru calibrul instituțional (pentru darea acesteia în folosința unei mâini factuale de studenți) constă preponderent în dezvoltarea unui server adevărat (implicit a bazei de date cu utilizatori și a mediului de postare a notelor). De asemenea, am trecut în agendă faptul că funcționalitatea de completare automată a actelor vizează momentan numai tipuri de documente alese aleator din mulțimea acelor ce sunt de regulă expediate către secretariat. Aria de acoperire a acestei funcționalități ar putea fi (ba chiar s-ar dori a fi) extinsă în așa fel încât studentul să aibă capacitatea de a trimite o sferă cât mai largă de documente completate direct de robot pe baza șabloanelor de pe site-ul facultății.

La scară mai înaltă, progresul lui Elliot s-ar putea îndrepta înspre lansarea unei aplicații mobile copil pentru vizualizarea notelor. În rest, cum am precizat în introducere, studenții sunt singurii cu adevărat capabili de a furniza idei de noi direcții pe care să o apuce robotul universitar, inspirându-se din nevoile și greutățile de care se împiedică în viața studentască.

## 4.6 Notițe de final

- La ilustrarea diagramelor din această lucrare s-a folosit programul web *Visual Paradigm* (vezi [7]).
- Prin programul *Install Forge*, se permite distribuția extinsă a executabilului Elliot prin crearea unui *set-up* ușor de întrebuințat și de propagat.

# Bibliografie

- [1] Markus Alberth and Michael Mattern | Managing Principals at Capco. *"Understanding robotic process automation (RPA)"*. Apaținând de: *JOURNAL - THE CAPCO INSTITUTE JOURNAL OF FINANCIAL TRANSFORMATION, AUTOMATION* Henley Business School – Capco Institute, Paper Series in Financial Services #46. 2017.
- [2] *Documentation on pbkdf2*. <https://pkg.go.dev/golang.org/x/crypto/pbkdf2>. 2021.
- [3] Elliot Forbes. *Go Encryption and Decryption using AES - Tutorial*. <https://tinyurl.com/AESenc>. 2018.
- [4] Senior Vice President of Learning UiPath Tom Clancy. *"How Schools are Enabling a Robot for Every Student"*. Accesat cu: <https://www.uipath.com/blog/schools-enabling-robots-for-students>. 2020.
- [5] Vice President of Global Communications at UiPath Toni Iafrate. *"Academic Alliance Momentum"*. Accesat cu: <https://tinyurl.com/AcademicAlliance>. 2019.
- [6] UiPath. *"Build a Thriving Enterprise in the Automation First Era"*. Accesat cu: <https://tinyurl.com/AutomationFirstEraPDF>. Material prezentat în cadrul conferinței: <https://tinyurl.com/AutomationFirstConference>. 2019.
- [7] *Visual Paradigm*. <https://www.visual-paradigm.com/>.
- [8] zaddok. *Moodle API Library*. <https://github.com/zaddok/moodle>. 2018.