

UNIVERSITATEA "ALEXANDRU-IOAN CUZA" DIN IAȘI

**FACULTATEA DE INFORMATICĂ**



LUCRARE DE LICENȚĂ

**GenRap**

**Sistem Automatizat pentru Raportarea  
Publicațiilor și Citărilor Academice**

propusă de

**Clara-Daniela Sima**

**Sesiunea: iulie, 2024**

Coordonator științific

**Conf. Dr. Arusoaie Andrei**

UNIVERSITATEA "ALEXANDRU-IOAN CUZA" DIN IAȘI

**FACULTATEA DE INFORMATICĂ**

**GenRap**

**Sistem Automatizat pentru Raportarea  
Publicațiilor și Citărilor Academice**

**Clara-Daniela Sima**

**Sesiunea: iulie, 2024**

Coordonator științific

**Conf. Dr. Arusoaie Andrei**

Avizat,  
Îndrumător lucrare de licență,  
Conf. Dr. Arusoaie Andrei.

Data: ..... Semnătura: .....

## **Declarație privind autenticitatea conținutului lucrării de licență**

Subsemnata **Sima Clara-Daniela** domiciliată în **România, jud. Iași, sat. Tomești(com. Tomești) Șos. Veche, nr. 16**, născută la data de **17 ianuarie 2001**, identificată prin **CNP 6010117440021**, absolventă a **Universității „Alexandru Ioan Cuza” din Iași , Facultatea de informatică** specializarea **informatică**, promoția 2022, declar pe propria răspundere, cunoscând consecințele falsului în declarații în sensul art. 326 din Noul Cod Penal și dispozițiile Legii Educației Naționale nr. 1/2011 art.143 al. 4 și 5 referitoare la plagiat, că lucrarea de licență cu titlul **GenRap - Sistem Automatizat pentru Raportarea Publicațiilor și Citărilor Academice** elaborată sub îndrumarea domnului **Conf. Dr. Arusoaie Andrei**, este autentică, îmi aparține și îmi asum conținutul său în întregime.

De asemenea, declar că sunt de acord ca lucrarea mea de licență să fie verificată prin orice modalitate legală pentru confirmarea autenticității, consimțind inclusiv la introducerea conținutului său într-o bază de date în acest scop. Declar că lucrarea de față are exact același conținut cu lucrarea în format electronic pe care profesorul îndrumător a verificat-o prin intermediul software-ului de detectare a plagiatului.

Am luat la cunoștință despre faptul că este interzisă comercializarea de lucrări științifice în vederea facilitării falsificării de către cumpărător a calității de autor al unei lucrări de licență, de diplomă sau de disertație și în acest sens declar pe proprie răspundere că lucrarea de față nu a fost copiată, ci reprezintă rodul cercetării pe care am întreprins-o.

Data: .....

Semnătura student:

.....

## Declarație de consimțământ

Prin prezenta declar că sunt de acord ca lucrarea de licență cu titlul **GenRap - Sistem Automatizat pentru Raportarea Publicațiilor și Citărilor Academice** , codul sursă al programelor și celelalte conținuturi (grafice, multimedia, date de test, etc.) care însoțesc această lucrare să fie utilizate în cadrul Facultății de informatică.

De asemenea, sunt de acord ca Facultatea de informatică de la Universitatea "Alexandru-Ioan Cuza" din Iași, să utilizeze, modifice, reproducă și să distribuie în scopuri necomerciale programele-calculator, format executabil și sursă, realizate de mine în cadrul prezentei lucrări de licență.

Absolvent **Clara-Daniela Sima**

Iași, Data: .....

Semnătura:

.....

# Cuprins

<b>1</b>	<b>Introducere</b>	<b>2</b>
1.1	Context . . . . .	2
1.2	Motivație . . . . .	2
1.3	Descrierea problemei . . . . .	3
1.4	Contribuțiile lucrării . . . . .	3
<b>2</b>	<b>Soluția propusă</b>	<b>4</b>
2.1	Tehnologii Folosite . . . . .	4
<b>3</b>	<b>Proiectarea Sistemului</b>	<b>7</b>
3.1	Arhitectura Client-Server . . . . .	7
3.1.1	Scalabilitate și Flexibilitate . . . . .	7
3.1.2	Autentificare și Autorizare . . . . .	7
3.2	Diagrama generală a sistemului . . . . .	8
3.3	Proiectarea bazei de date . . . . .	9
3.3.1	Publicații . . . . .	9
3.3.2	Citări . . . . .	10
3.4	Diagrame de flux . . . . .	11
3.4.1	Fluxul de Autentificare și Autorizare a Utilizatorilor . . . . .	12
3.4.2	Meniu Principal . . . . .	13
3.4.3	Fluxul de gestionare a Publicațiilor . . . . .	13
3.4.4	Fluxul de generarea a Rapoartelor . . . . .	14
3.4.5	Fluxul de gestionare a citărilor . . . . .	14
3.5	Arhitectura MVC (Model-View-Controller) . . . . .	14
3.6	Principiul DRY . . . . .	15
3.7	JWT . . . . .	16
3.7.1	Cum funcționează JWT în aplicație? . . . . .	16

3.8	Variabile de mediu . . . . .	17
3.9	SPA . . . . .	17
3.10	Securitatea Aplicației . . . . .	17
<b>4</b>	<b>Extragerea automată a Datelor</b>	<b>19</b>
4.1	Publicații DBLP: <code>dblp.py</code> . . . . .	20
4.2	Publicații Google Scholar: <code>scholar.py</code> . . . . .	21
4.2.1	Selenium . . . . .	21
4.2.2	Beautiful Soup . . . . .	24
4.3	Citări Google Scholar <code>citations.py</code> . . . . .	24
4.3.1	Scraper API . . . . .	24
4.3.2	Organizare în pagini . . . . .	25
4.3.3	Beautiful Soup . . . . .	26
<b>5</b>	<b>Descrierea REST API-ului</b>	<b>27</b>
5.1	Proprietăți RESTful . . . . .	27
5.2	Autorizare endpoint-uri . . . . .	27
5.3	Endpoint-uri . . . . .	28
5.3.1	Autentificare și Autorizare . . . . .	28
5.3.2	Utilizatori . . . . .	29
5.3.3	Publicație unică . . . . .	31
5.3.4	Publicații . . . . .	35
5.3.5	Citări . . . . .	36
5.3.6	Generare Rapoarte . . . . .	40
5.4	Formate de Cerere/Răspuns și Exemple de Utilizare . . . . .	41
5.5	Structura codului . . . . .	41
5.6	Detalierea funcționalităților . . . . .	43
5.6.1	Înregistrare . . . . .	43
5.6.2	Autentificare . . . . .	44
5.6.3	Detectarea Duplicatelor . . . . .	44
5.6.4	Refresh Optim . . . . .	46
5.6.5	Generarea Rapoartelor . . . . .	46
<b>6</b>	<b>Interfața VueJS</b>	<b>47</b>
6.1	Pagina de Întâmpinare pentru Persoanele Neautentificate . . . . .	47

6.2	Înregistrare . . . . .	48
6.3	Autentificare . . . . .	49
6.4	Meniul principal din interfață . . . . .	49
6.5	Gestionarea Publicațiilor (My Publications) . . . . .	51
6.5.1	Butonul Update data form dblp and scholars . . . . .	51
6.5.2	Tag-ul NEW! . . . . .	51
6.5.3	Prima accesare a unui utilizator nou . . . . .	51
6.5.4	Afișarea publicațiilor deja extrase . . . . .	52
6.5.5	Status cu numărul de publicații obținute . . . . .	52
6.5.6	Butonul Add Publication . . . . .	53
6.5.7	Butonul Edit: Editare Manuală . . . . .	54
6.5.8	Publicații asemănătoare . . . . .	55
6.5.9	See citations button . . . . .	56
6.6	Generarea Rapoartelor: Create publications report . . . . .	56
6.7	Publicațiile cu citări: My Citations . . . . .	58
6.8	Profilul utilizatorului: My Profile . . . . .	58
6.9	Profilul utilizatorului: Log out . . . . .	58
6.10	Vizualizarea grafică a unei acțiuni în curs . . . . .	59
6.11	Pagina de Citări ale unei Publicații . . . . .	59
6.12	Înfățișare adaptabilă (Reactive) . . . . .	61
6.13	Structura codului . . . . .	63
<b>7</b>	<b>Ghid de Configurare</b>	<b>66</b>
7.1	Configurarea Aplicației Vue.js . . . . .	66
7.1.1	Precondiții . . . . .	66
7.1.2	Pași . . . . .	66
7.2	Configurarea API-ului . . . . .	67
7.2.1	Precondiții . . . . .	67
7.2.2	Pași . . . . .	68
7.3	Testarea API-ului . . . . .	68
7.3.1	Testarea în Terminal . . . . .	68
7.3.2	Testarea cu Insomnia . . . . .	69

<b>8</b>	<b>Analiză Comparativă cu alte Proiecte Similare</b>	<b>71</b>
8.1	Zotero . . . . .	71
8.2	Google Scholar . . . . .	72
8.3	Scopus . . . . .	72
8.4	Funcționalități adiționale ale aplicației propuse . . . . .	73
<b>9</b>	<b>Concluzii</b>	<b>74</b>
9.1	Planuri de îmbunătățire . . . . .	74
9.2	Concluzie finală . . . . .	75
	<b>Bibliografie</b>	<b>76</b>



# Capitolul 1

## Introducere

### 1.1 Context

În mediul academic, profesorii și cercetătorii trebuie să facă periodic anumite rapoarte care cuprind lucrările publicate și citările primite. Aceste rapoarte sunt esențiale pentru ei și reflectă impactul și performanța lor academică. Cu toate acestea, colectarea și modificarea conținutului acestor rapoarte necesită multă atenție la detalii, răbdare și timp.

### 1.2 Motivație

Profesorii trebuie să își caute publicațiile și citările corespunzătoare acestora pe diverse website-uri internaționale, precum Google Scholar și DBLP, și să le editeze manual. Acest proces este ineficient și poate consuma mult timp și efort. Scopul aplicației acesteia este să automatizeze acest proces și să îi ajute pe utilizatori să importe automat publicațiile și citările, să le editeze așa cum își doresc ei și să genereze rapoarte detaliate în diferite formate Excel, CSV și Text. Astfel, prin reducerea efortului manual și minimizarea erorilor umane, aplicația îmbunătățește eficiența și acuratețea procesului de raportare pentru profesori.

## 1.3 Descrierea problemei

Google Scholar și DBLP conțin seturi de date diferite. Unele publicații pot coincide, dar altele pot apărea doar pe unul dintre site-uri. Acest lucru complică procesul de colectare a datelor. Profesorii trebuie să verifice conținutul în mai multe locuri pentru rapoartele lor, preluând și modificând manual datele, astfel consumând foarte mult timp în întocmirea raportului.

## 1.4 Contribuțiile lucrării

- Automatizarea extragerii publicațiilor de pe Google Scholar și DBLP
- Posibilitatea unui refresh periodic cu un efort minimal
- Editare manuală a datelor extrase automat
- Adăugare publicație și posibilitatea de a adăuga un câmp nou pentru publicații și citări
- Generarea unor rapoarte cu detaliile dorite în format Excel, CSV și Text
- Posibilitatea de a selecta ce câmpuri să apară în rapoarte
- Posibilitatea de a selecta automat publicațiile și citările dintr-un anumit interval de timp pentru generarea rapoartelor
- Interfață prietenoasă în care poți vizualiza și gestiona publicațiile și citările proprii
- Fiecare utilizator are un cont în care se păstrează evidența publicațiilor și citărilor deja extrase și editate
- Detectarea automată a duplicatelor (publicațiile care apar și în DBLP și în Google Scholar) și posibilitatea de a alege care dintre ele este mai bună sau, în caz că profesorul consideră că detectarea automată nu a funcționat corect, să le separe în două publicații (una DBLP, una Scholar)
- Adăugarea automată a câmpurilor: tip (conference, journal, workshop, book), categorie (A, B, C, D), index (BDI, ISI, Unranked) și factor de impact în funcție de tipul publicației, care urmează a fi completate cu valori mai apoi de către profesor

# Capitolul 2

## Soluția propusă

Creând un sistem automat pentru extragerea publicațiilor și a citărilor din multiple surse, precum Google Scholar și DBLP, acest proiect urmărește să rezolve ineficiența în colectarea acestora. De asemenea, permite utilizatorilor să perfecționeze datele extrase manual și oferă posibilitatea de a genera rapoarte în diferite formate.

### 2.1 Tehnologii Folosite

- **Python**
  - **Descriere:** Python este un limbaj de programare de înalt nivel, cunoscut pentru sintaxa sa simplă, care facilitează citirea și scrierea codului.
  - **Utilizare:** A fost utilizat în dezvoltarea backend-ului aplicației pentru scrierea scripturilor de extragere a datelor și pentru implementarea api-ului
- **Flask**
  - **Descriere:** Flask este un framework pentru Python cunoscut pentru instrumentele folosite pentru a crea aplicații web în Python
  - **Utilizare:** A fost utilizat pentru dezvoltarea REST API-ului care gestionează cererile de la client și interacționează cu baza de date.

- **Vue.js**

- **Descriere:** Vue.js este cunoscut ca fiind un framework JavaScript ușor de utilizat, performant și flexibil pentru construirea interfețelor de utilizator
- **Utilizare:** Am ales VueJs pentru a dezvolta partea de front-end a acestei aplicații

- **MongoDB**

- **Descriere:** MongoDB este o bază de date NoSQL, document-oriented, documentele sunt stocate în format JSON-like, astfel fiind posibilă o gestionare flexibilă a datelor.
- **Utilizare:** A fost folosit pentru stocarea publicațiilor, citărilor și a utilizatorilor.

- **DBLP**

- **Descriere:** DBLP este o bază de date de bibliografie informatică care oferă acces la o mare varietate de publicații academice în domeniul informaticii.
- **Utilizare:** A fost utilizat ca sursă pentru extragerea automată a datelor publicațiilor.

- **Google Scholar**

- **Descriere:** Google Scholar este un motor de căutare dedicat literaturii academice, care indexează articole, teze, cărți și alte lucrări științifice.
- **Utilizare:** A fost utilizat pentru extragerea automată a citărilor și publicațiilor.

- **ScraperAPI**

- **Descriere:** ScraperAPI este un serviciu care permite extragerea datelor de pe web prin intermediul API-ului său, gestionând problemele legate de proxy și CAPTCHA.
- **Utilizare:** A fost utilizat pentru a obține HTML-ul paginilor de citări de pe Google Scholar.

- **Openpyxl**

- **Descriere:** Openpyxl este o bibliotecă Python pentru citirea și scrierea fișierelor Excel 2010.
- **Utilizare:** A fost folosit pentru generarea rapoartelor în format Excel.

- **CSV**

- **Descriere:** CSV (Comma Separated Values) este un format de fișier simplu pentru păstrarea datelor sub formă de tabel
- **Utilizare:** A fost folosit pentru generarea rapoartelor în format CSV.

- **io.StringIO**

- **Descriere:** StringIO este un modul din Python care implementează un fișier în memorie.
- **Utilizare:** A fost utilizat pentru manipularea datelor text în memorie înainte de generarea fișierelor TXT și CSV.

# Capitolul 3

## Proiectarea Sistemului

### 3.1 Arhitectura Client-Server

Arhitectura Client-Server este un model de proiectare a software-ului care împarte aplicația în două părți de bază: partea de client, cea care interacționează direct cu utilizatorul final, și partea de server, cea care gestionează și distribuie funcționalitățile necesare unei aplicații web. Scopul acestei arhitecturi este organizarea și împărțirea sistematică a rolurilor în aplicație.

#### 3.1.1 Scalabilitate și Flexibilitate

Acest tip de arhitectură permite scalabilitatea și flexibilitatea aplicației. Clientul și serverul pot fi dezvoltate independent, adăugând noi funcționalități, gestionând număr mare de utilizatori etc. De asemenea, utilizarea bazei de date NoSQL MongoDB permite adaptarea rapidă la schimbările de structură a datelor.

#### 3.1.2 Autentificare și Autorizare

Un aspect important al arhitecturii Client-Server este gestionarea autentificării și autorizării utilizatorilor, astfel utilizatorii se autentifică pe baza unor credențiale (nume de utilizator și parolă), iar serverul generează un token JWT (JSON Web Token) care este stocat în localStorage pe partea de client. Acest token este trimis cu fiecare cerere ulterioară pentru a verifica autorizarea utilizatorului.

## 3.2 Diagrama generală a sistemului

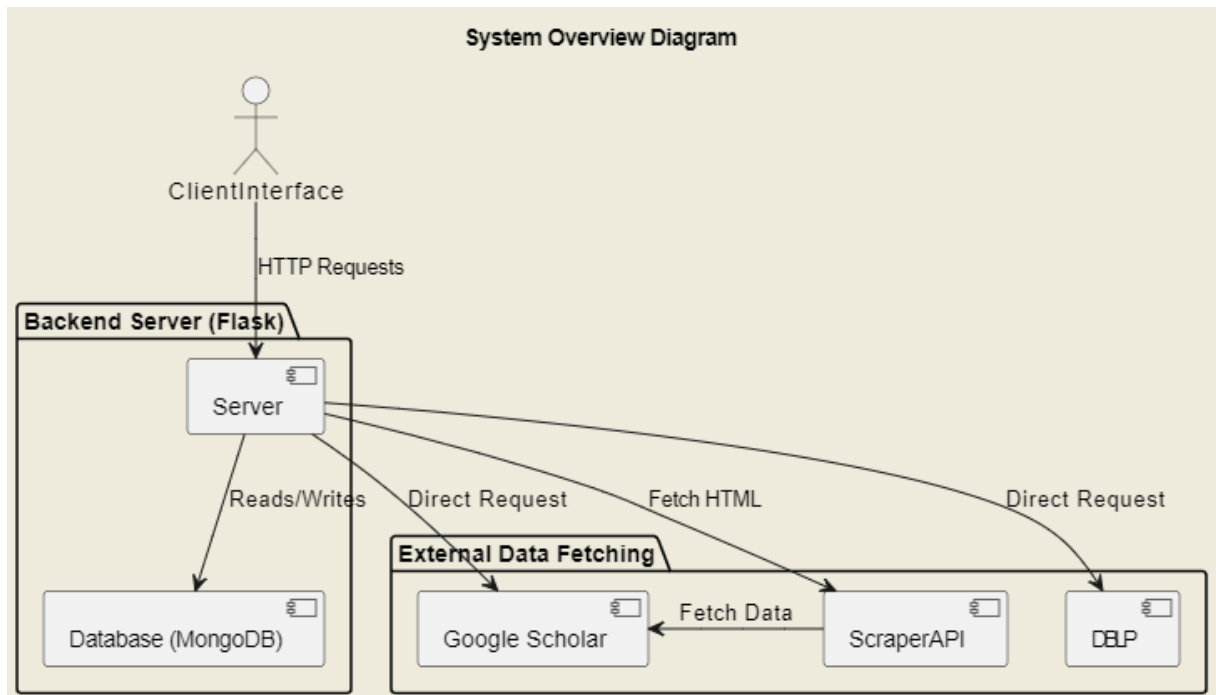


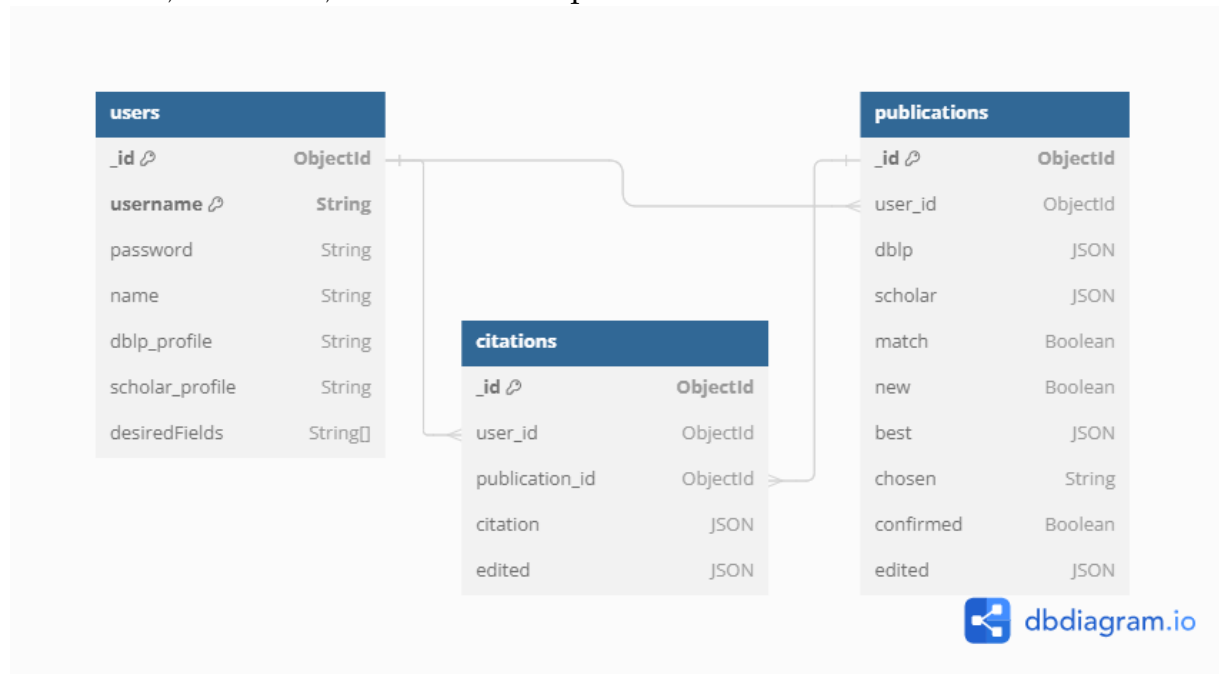
Diagrama generală a sistemului ilustrează arhitectura proiectului și fluxul de date dintre componente. Aceasta include următoarele elemente:

- **ClientInterface:** Reprezintă interfața utilizatorului (clientul), care trimite cereri HTTP către server și primește răspunsuri HTTP.
- **Backend Server (Flask):** Serverul backend care utilizează ScraperAPI pentru a obține HTML-ul paginilor de citări de pe Google Scholar. ScraperAPI, apoi preia datele de pe Google Scholar și le trimite serverului backend.
- **Database (MongoDB):** Baza de date MongoDB care stochează informațiile despre publicații, citări și utilizatori. Serverul backend citește și scrie date în această bază de date.
- **External Data Fetching:**
  - **DBLP:** O sursă externă de unde serverul backend extrage direct informații despre publicații.
  - **Google Scholar:** O altă sursă externă de unde serverul backend extrage direct informații despre publicații și citări.
  - **ScraperAPI:** Un serviciu extern utilizat pentru a obține HTML-ul paginilor de citări de pe Google Scholar, atunci când este necesar. ScraperAPI trimite datele către serverul backend.

## 3.3 Proiectarea bazei de date

Baza de date cuprinde 3 colecții: users(utilizatori), publications(publicații) și citations(citări). Fiecare document din fiecare colecție are un id.

Colecțiile interacționează între ele prin folosind id-urile.



### 3.3.1 Publicații

Fiecare document de publicație poate conține mai multe variante ale aceleiași publicații.

Dacă s-a constatat că o anumită publicație indexată de DBLP și alta indexată de Google Scholar că ar fi aceeași publicație, atunci acele publicații apar în același document însoțite de câmpul match:true.

În această fază, utilizatorul are două opțiuni. El poate alege o variantă a articolului și anume pe cea din DBLP sau pe cea din Scholar. Ceea ce va alege el va fi stocat în interiorul câmpului best. În plus, se va adăuga câmpul chosen, care va fi un șir de caractere cu valorile "scholar" sau "dblp". De asemenea, se va adăuga un câmp confirmed:true.

De altfel, utilizatorul poate considera că publicațiile nu reprezintă de fapt aceeași publicație. În acest caz match va deveni:false și se vor crea două publicații diferite.

Dacă utilizatorul va modifica publicația, acele modificări vor fi salvate în câmpul edited.

Schema edited vine cu câmpuri suplimentare:



- **În mod implicit:** Când utilizatorul alege opțiunea de a edita o publicație se vor adăuga câmpurile:
  - **type:** Acesta reprezintă tipul publicației și poate fi de 4 tipuri: Conference, Journal, Workshop sau Book
  - **indexed:** Acesta are 3 opțiuni: ISI, BDI, Unranked
- **Care apar în urma altor selecții:** Unele câmpuri nu apar de la început, ci sunt specifice numai anumitor publicații:
  - **category:** Acesta apare numai dacă s-a selectat **type** Conference sau Workshop și are 4 opțiuni: A,B,C,D
  - **impactFactor:** Acesta reprezintă factorul de impact și apare numai în două cazuri:
    - \* A fost selectat **type** Journal și **indexed** ISI.
    - \* A fost selectat **type** Workshop
- **Personalizat:** Profesorul are opțiunea de a adăuga noi câmpuri, personalizate oricărei publicații, câmpuri ce pot avea orice nume.



### 3.3.2 Citări

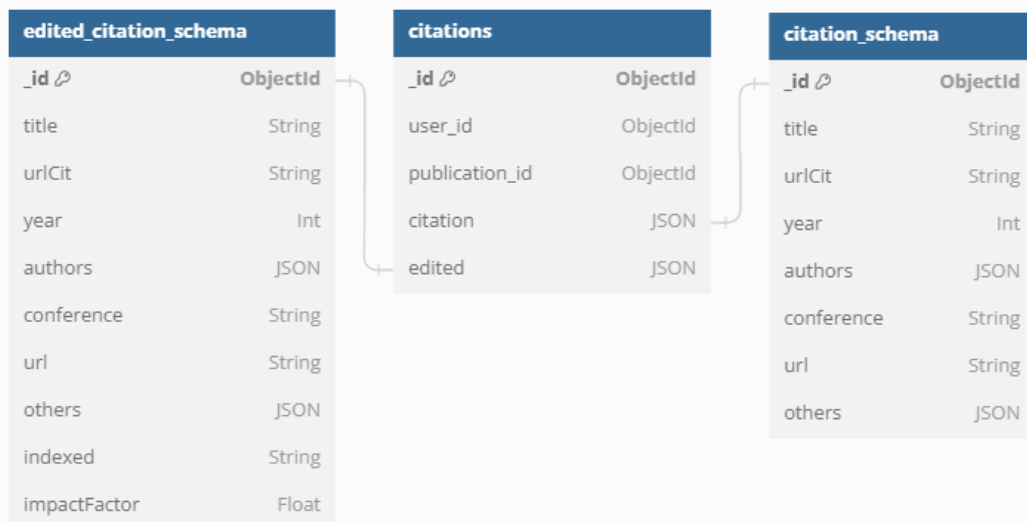
Acestea conțin datele de bază ale citării, care au fost extrase de pe Google Scholar:

- **title:** Titlul publicației care citează
- **urlCit:** Linkul care duce spre publicația care citează

- **year:** Anul în care s-a publicat acea publicație
- **conference:** Conferința unde s-a publicat acea publicație
- **url:** Linkul spre domeniul publicației.
- **others:** Alte informații care au mai apărut, dar nu se știe exact ce reprezintă.

De asemenea, în urma editării citării de către profesor, noi câmpuri vor apărea.

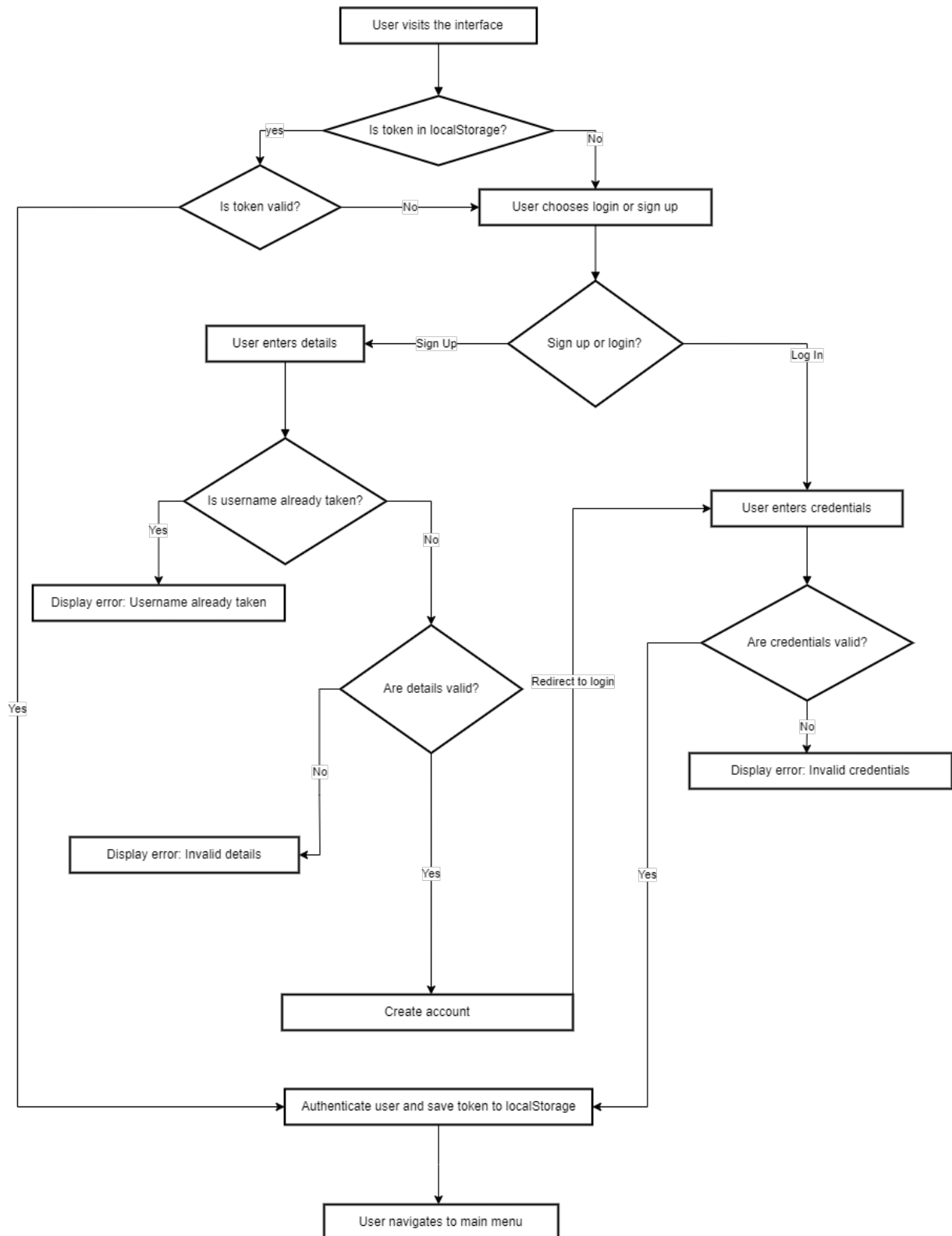
- **În mod implicit:** Când utilizatorul alege opțiunea de a edita o publicație se vor adăuga câmpurile:
  - **indexed:** Acesta are 3 opțiuni: ISI, BDI, Unranked
  - **impactFactor:** Acesta reprezintă factorul de impact și apare mereu, diferit de publicațiile utilizatorului.
- **Personalizat:** Profesorul are opțiunea de a adăuga noi câmpuri, personalizate oricărei citări, câmpuri ce pot avea orice nume.



### 3.4 Diagrame de flux

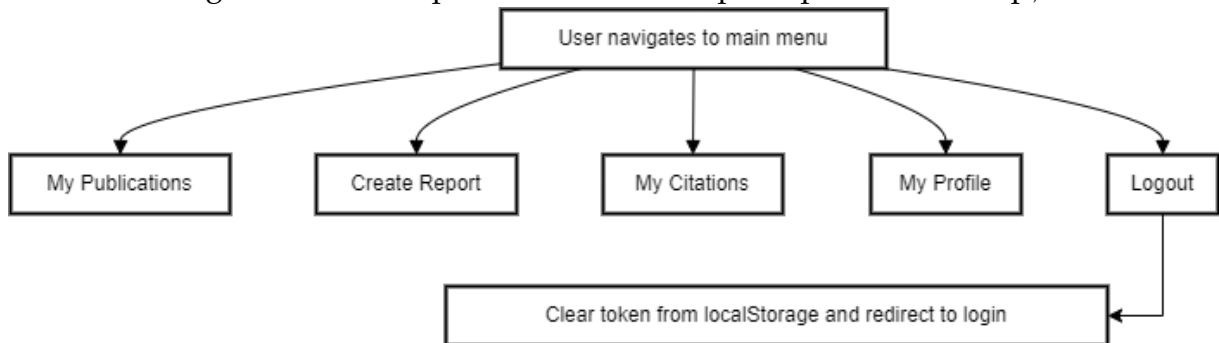
Diagramele de flux ale aplicației ajută la înțelegerea fluxului de acțiuni în tot sistemul prin reprezentarea vizuală clară a interacțiunilor utilizatorilor și punctele de decizie din cadrul sistemului.

### 3.4.1 Fluxul de Autentificare și Autorizare a Utilizatorilor

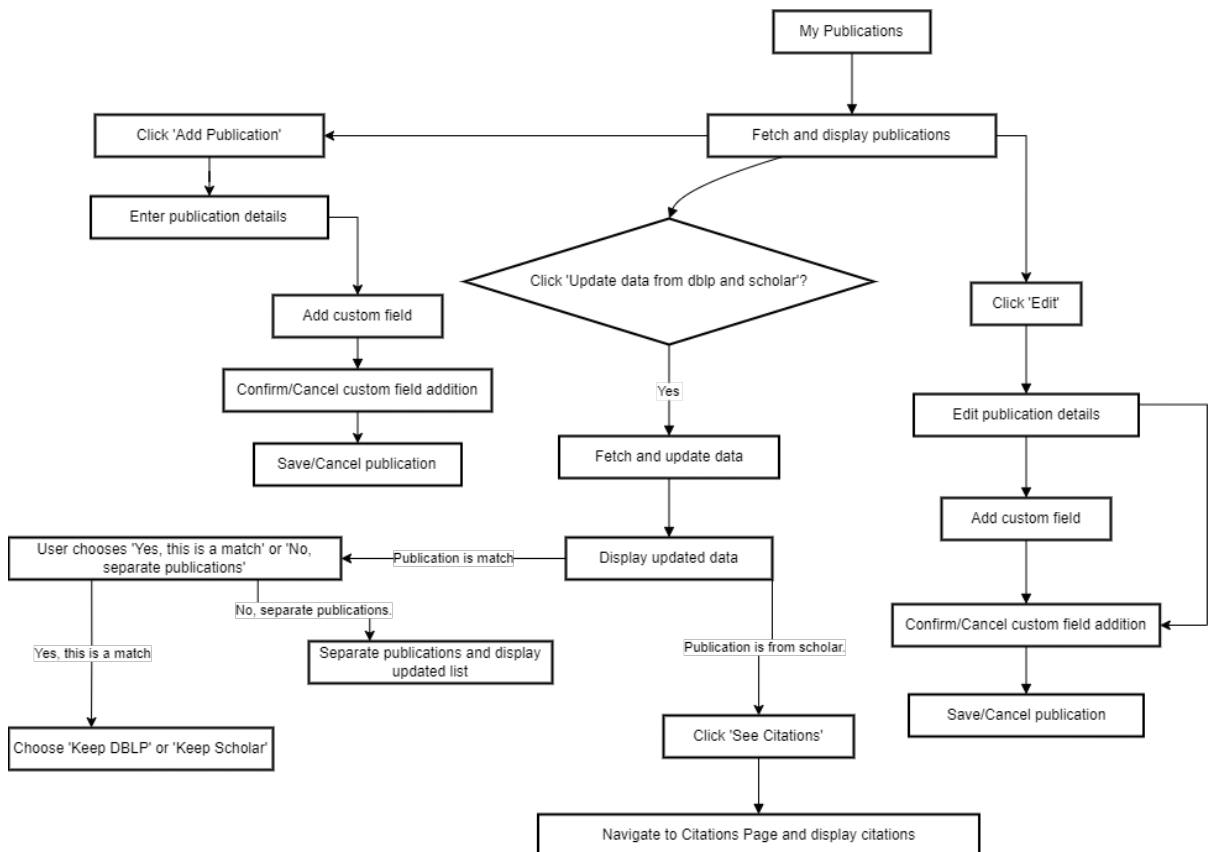


### 3.4.2 Meniu Principal

Odată logat, utilizatorul poate vedea meniul principal care are 5 opțiuni:



### 3.4.3 Fluxul de gestionare a Publicațiilor



### 3.4.4 Fluxul de generarea a Rapoartelor

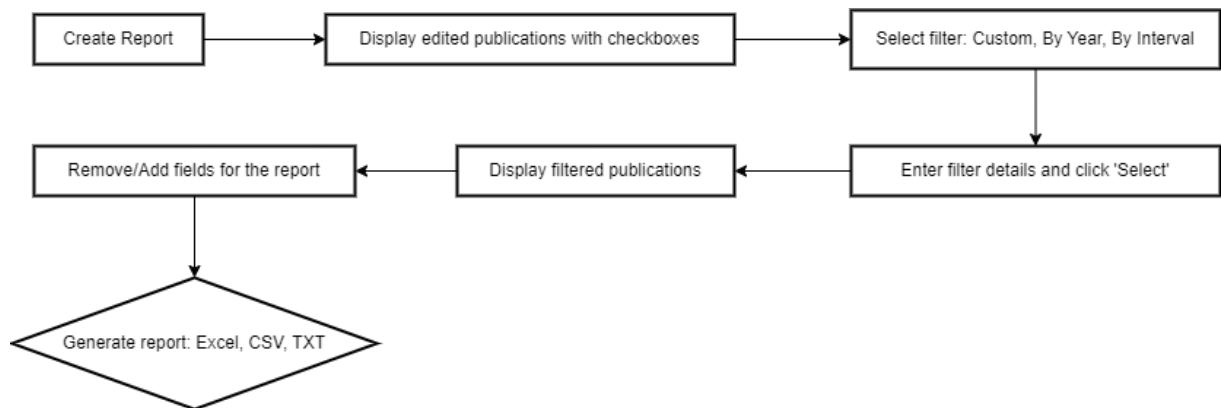


Diagrama de flux pentru generarea de rapoarte descrie procesul de creare a rapoartelor detaliate pe baza filtrelor și câmpurilor selectate de utilizator.

### 3.4.5 Fluxul de gestionare a citărilor

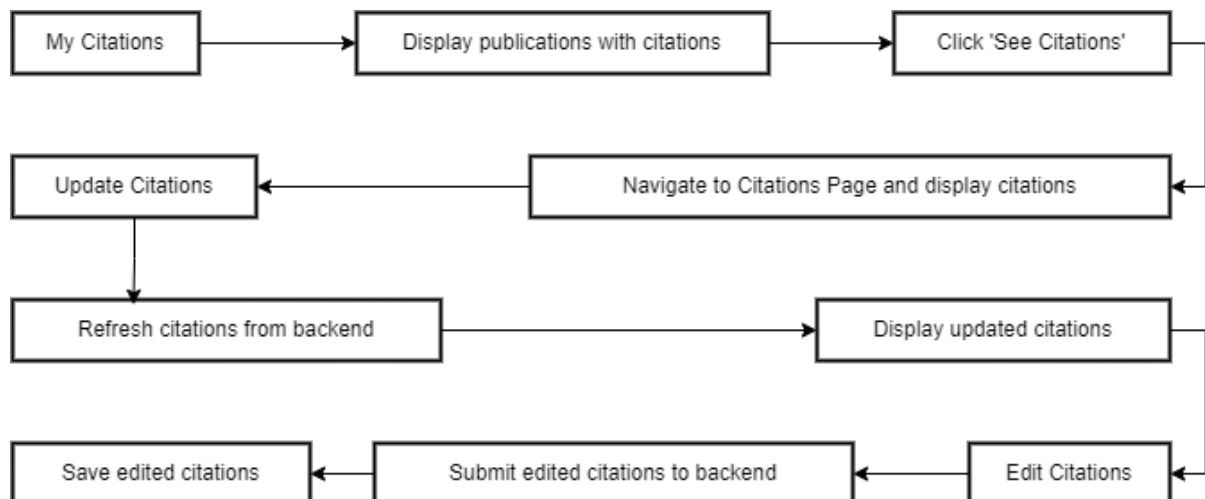


Diagrama de flux pentru gestionarea citărilor descrie procedurile pentru administrarea citărilor.

## 3.5 Arhitectura MVC (Model-View-Controller)

Arhitectura MVC (Model-View-Controller) este un model de design software care separă logica aplicației în trei componente interdependente, fiecare cu responsabilități distincte: Model, View și Controller, astfel facilitând dezvoltarea, testarea și întreținerea aplicațiilor.

În contextul aplicației, Model reprezintă structura de date a proiectului, organizată în 3 colecții (utilizatori, publicații și citări). Datele sunt gestionate eficient prin utilizarea operațiunilor CRUD (Create, Read, Update, Delete)

View-ul reprezintă interfața care este implementată folosind framework-ul Vue.js și utilizatorul. View-ul este responsabil pentru afișarea datelor și interacțiunea cu utilizatorul final. În aplicație există diferite pagini care pot fi vizualizate: Pagina de Sign In, de Log In, My publications, My profile, My citations.

Controller-ul este reprezentat de Api-ul implementat în Flask, acesta gestionează logica aplicației și interacțiunea dintre model și view. El primește cereri http de la client și extrage și prelucrează datele din baza de date. Gestionează cererile pentru generarea rapoartelor, actualizare a publicațiilor sau citărilor și cererile de înregistrare și autentificare.

### Modelul CRUD

Modelul CRUD este esențial pentru gestionarea eficientă a datelor într-o aplicație. Fiecare operațiune este realizată astfel:

- **Create (Creare):** Se pot adăuga utilizatori noi, publicații și citări în baza de date. De exemplu, utilizatorul își poate crea un cont nou sau adăuga o nouă publicație.
- **Read (Citire):** Se pot citi datele existente din baza de date. De exemplu, pentru a afișa publicațiile sau citările unui utilizator acestea trebuie citite mai întâi din baza de date.
- **Update (Actualizare):** Utilizatorul poate modifica datele existente prin actualizarea detaliilor unei publicații sau a unei citări, de exemplu.
- **Delete (Ștergere):** Se pot șterge datele existente din baza de date. De exemplu, ștergerea unei publicații care nu îi este utilă utilizatorului.

## 3.6 Principiul DRY

Principiul DRY (Don't Repeat Yourself) sugerează evitarea duplicării codului. Aplicarea acestui principiu se face prin: reutilizarea componentelor în interfață (de exemplu PublicationBasic.vue, reutilizarea stilului CSS în întreaga aplicație și folosirea rutelor și a funcțiilor din API.

## 3.7 JWT

JWT (JSON Web Token) este un standard pentru transmiterea informațiilor între părți sub forma unui obiect JSON. Aceste informații sunt semnate digital, asigurând astfel autenticitatea datelor. Un Json Web Token este ușor de trimis între server și client, acesta este semnat criptografic, prevenind modificările, iar utilizarea acestuia elimină necesitatea păstrării sesiunilor pe server.

Un token JWT este format din trei părți:

- **Header:** Tipul tokenului și algoritmul de semnare.
- **Payload:** Informațiile despre utilizator.
- **Signature:** Semnătura asigură integritatea tokenului.

### 3.7.1 Cum funcționează JWT în aplicație?

#### Autentificare:

1. Utilizatorul trimite o cerere de autentificare cu numele de utilizator și parola.
2. Serverul verifică credențialele, dacă acestea sunt corecte, generează un token JWT.
3. Tokenul este returnat clientului și stocat în *localStorage*.

#### Autorizare:

1. Fiecare cerere ulterioară către server trebuie să includă tokenul JWT în antetul cererii HTTP (*x-access-token*).
2. Serverul verifică validitatea tokenului. Dacă este valid, cererea este procesată.

Pentru encodarea și decodarea sigură a JWT-urilor se folosește o cheie secretă:

```
token = jwt.encode({'_id': user["_id"], 'exp': expiration},  
                  config.SECRET_KEY)  
data = jwt.decode(token, config.SECRET_KEY, algorithms=["HS256"])
```

## 3.8 Variabile de mediu

Variabilele de mediu (environment variables) se folosesc pentru a păstra anumite informații care sunt sensibile sau care se doresc a fi ușor de schimbat pentru diferitele medii de rulare, fără a modifica codul sursă.

În acest proiect există 3 variabile de mediu:

1. **SECRET\_KEY:** Această variabilă de mediu stochează cheia secretă utilizată pentru a encoda și decoda tokenurile JWT (JSON Web Token), asigurând securitatea aplicației, prin garantarea autenticității token-urilor. Dacă această cheie nu ar exista, orice persoană ar putea să acceseze informațiile din token.
2. **DB\_URI:** Această variabilă de mediu conține URI-ul de conexiune la baza de date MongoDB. Ea permite schimbarea bazei de date fără a modifica codul aplicației.
3. **SCRAPET\_API\_KEY:** Această variabilă de mediu conține cheia API pentru serviciul de scraping utilizat pentru a colecta date despre citările de pe Google Scholar. Aceasta trebuie să fie protejată deoarece acesta este un serviciu care necesită plată și trebuie ferit de posibilitatea de a fi accesat și utilizat fără autorizarea celui care plătește serviciul.

## 3.9 SPA

Interfața este o Single Page Application (SPA) deoarece interacțiunile utilizatorului sunt controlate pe o singură pagină web. Acest lucru oferă rapiditate datorită faptului că resursele sunt încărcate o singură dată. Conținutul se actualizează dinamic în urma interacțiunii utilizatorului cu interfața web, nefiind nevoie de niciun refresh al paginii.

## 3.10 Securitatea Aplicației

Aplicația implementată asigură securitatea datelor utilizatorilor utilizând următoarele bune practici:

- Utilizare de token-uri JWT pentru autentificare



- Token-urile JWT sunt utilizate pentru a valida cererile. Acestea reduc astfel riscul atacurilor CSRF
- Parolele utilizatorilor sunt criptate folosind algoritmul SHA-256 înainte de a fi inserate în baza de date
- Utilizatorul are reguli stricte pentru crearea unei parole și a celorlalte date introduse
- CORS (Cross-Origin Resource Sharing) este setat pentru a controla accesul la API
- Folosirea variabilelor de mediu

# Capitolul 4

## Extragerea automată a Datelor

Aplicația utilizează trei scripturi principale pentru extragerea, filtrarea și prelucrarea datelor din diverse surse academice: `citations.py`, `dblp.py` și `scholar.py`. Aceste scripturi sunt folosite pentru colectarea informațiilor despre publicațiile și citările utilizatorilor.

În urma analizării paginilor web ale website-urilor DBLP și Google Scholar am identificat structura generală a acestora și am reușit să găsesc un pattern de care mă pot folosi pentru detectarea elementelor din pagină care conțin informațiile despre publicații.

Astfel am identificat diferitele tag-uri, clase și attribute specifice informațiilor dorite. Pentru a găsi toate informațiile, se analizează componentele de la mare la mic. Adică întâi extragem tot html-ul, apoi din acel html extragem codul care reprezintă publicațiile, apoi pentru fiecare publicație extragem codul care conține titlul, autorii, anul, apoi din fiecare extragem doar textul.

Este de evidențiat că website-ul nu furnizează direct informațiile necesare, ci noi le identificăm. În caz că administratorii website-urilor din care luăm informațiile decid să își schimbe structura atunci și va fi necesară reanalizarea structurii acestora și găsirea unor noi pattern-uri care sunt folosite.

Dacă există un element care nu respectă structura generală a celorlalte, aplicația nu îl va detecta.

Cu acest gând, m-am gândit la o soluție pentru a ajuta utilizatorii să știe că ceva nu a fost detectat. **Soluția** găsită este următoarea: am decis să afișez numărul elementelor detectate în interfață.

## Cum ajută afișarea numărului elementelor extrase automat?

În website-uri pot găsi numărul de publicații/citări în html, și apoi îl pot compara cu câte publicații/citări am reușit să extrag eu.

Astfel nu pot preciza exact care publicație/citare nu a fost extrasă, dar pot preciza numărul acestora.

### 4.1 Publicații DBLP: `dblp.py`

Acest script extrage informațiile despre publicațiile utilizatorilor de pe website-ul DBLP, folosind BeautifulSoup. Aceasta este structura html a unei publicații DBLP:

```
▼ <li class="entry" inproceedings toc" id="conf/icsoft/PistolA23" itemscope
  itemtype="http://schema.org/ScholarlyArticle">
  <link itemprop="additionalType" href="https://dblp.org/rdf/schema#Public
    ation">
  ▶ <div class="box"> ... </div>
  <div class="nr" id="c21">[c21]</div>
  ▶ <nav class="publ"> ... </nav>
  ▼ <cite class="data tts-content" itemprop="headline">
    ▼ <span itemprop="author" itemscope itemtype="http://schema.org/Person">
      ▼ <a href="https://dblp.org/pid/15/5770.html" itemprop="url">
        <span itemprop="name" title="Ionut Cristian Pistol">Ionut Cristian
          Pistol</span>
        </a>
      </span>
      ", "
    ▼ <span itemprop="author" itemscope itemtype="http://schema.org/Person">
      <span class="this-person" itemprop="name">Andrei Arusoaie</span>
      </span>
      ": "
      <br>
      <span class="title" itemprop="name">AIM-RL: A New Framework Supporting
        Reinforcement Learning Experiments.</span>
    ▼ <a href="https://dblp.org/db/conf/icsoft/icsoft2023.html#PistolA23">
      ▼ <span itemprop="isPartOf" itemscope itemtype="http://schema.org/Book
        Series">
        <span itemprop="name">ICSOFT</span>
      </span>
      <span itemprop="datePublished">2023</span>
    </a>
    ": "
    <span itemprop="pagination">412-419</span>
  </cite>
```

**Exemple:** Fiecare publicație se află într-un tag `li` cu clasa `entry`. În fiecare publicație se află un tag `cite` cu clasa `data tts-content` etc.

În unele cazuri vreau să caut mai multe elemente care au aceeași structură, în altele vreau să caut un singur element.

Când vreau să caut mai multe elemente folosesc `find_all`:

- caut toate publicațiile
- caut toți autorii

Când vreau să găsesc un singur element specific folosesc `find`:

- caut anul publicației
- caut conferința publicației

```
publications = soup.find_all("li", class_="entry")
author_spans = publication.find_all("span", itemprop="author")

date_span = publication.find('cite', class_='data tts-content')
               .find('span', itemprop='datePublished')
conference_span = publication.find('cite', class_='data tts-content')
               .find('span', itemprop='isPartOf')
```

Comenzile descrise mai sus îmi selectează tagurile în care se află informația necesară, dar acestea vin împreună cu mai multe informații, nu doar textul în sine. Pentru a extrage doar textul folosesc `get_text(strip=True)`:

```
author_name = author_span.get_text(strip=True)
pub_conference = conference_span.get_text(strip=True)
```

## 4.2 Publicații Google Scholar:scholar.py

### 4.2.1 Selenium

Acest script utilizează Selenium pentru a naviga și interacționa cu paginile Google Scholar.

## De ce este necesar Selenium?

Diferit de dblp.py, când se face un request la pagina unui utilizator pe Google Scholar nu se afișează direct toate publicațiile acestuia. Ci trebuia apăsat butonul "Show more" de mai multe ori până ce acesta ajunge să fie disabled.

Astfel pentru a obține toate datele trebuie să interacționăm cu interfața. De aceea aplicația utilizează Selenium pentru a naviga pe pagina Scholar și pentru a încărca toate publicațiile disponibile făcând click pe butonul "Show more".

Articles 1–100    ▼ SHOW MORE

```
▼ <div id="gsc_lwp">
  <span id="gsc_a_nn">Articles 1-100</span>
  ▼ <div id="gsc_bpf">
    ▼ <button type="button" id="gsc_bpf_more" class="gs_btnPD gs_in_ib gs_
      btn_flat gs_btn_lrge gs_btn_lsu">
      ▼ <span class="gs_wr">
        ::before
        <span class="gs_ico"></span>
        <span class="gs_lbl">Show more</span> == $0
      </span>
    </button>
  </div>

▼ <div id="gsc_lwp">
  <span id="gsc_a_nn">Articles 1-111</span>
  ▼ <div id="gsc_bpf">
    ▼ <button type="button" id="gsc_bpf_more" class="gs_btnPD gs_in_ib gs_
      btn_flat gs_btn_lrge gs_btn_lsu" disabled> == $0
      ▼ <span class="gs_wr">
        ::before
        <span class="gs_ico"></span>
        <span class="gs_lbl">Show more</span>
      </span>
    </button>
  </div>
```

```

# Automatically download and setup ChromeDriver
service = ChromeService(executable_path=ChromeDriverManager().install())
driver = webdriver.Chrome(service=service, options=chrome_options)

driver.get(url)

prev_span_text = ""
try:
    # Loop to repeatedly click the "Show more" button until it's disabled
    while True:
        button = driver.find_element(By.ID, 'gsc_bpf_more')

        # Check if the button is disabled
        if button.get_attribute('disabled'):
            span = driver.find_element(By.ID, 'gsc_a_nn')
            current_span_text = span.text
            if current_span_text != prev_span_text:
                prev_span_text = current_span_text
            else:
                html_copy = copy.deepcopy(driver.page_source)
                return html_copy

        # Click the "Show more" button
        button.click()
        time.sleep(2)

```

În codul afișat folosim web driver-ul Chrome pentru a găsi butonul Show more și îl apăsăm. Avem o buclă infinită pentru că nu știm de câte ori trebuie apăsate acest buton.

Pentru a ieși din această buclă infinită avem două condiții:

- Butonul Show more să fie disabled.
- conținutul actual să fie la fel ca cel anterior pentru tagul cu id-ul gsc\_a\_nn, acesta reprezintă stringul care arată așa: "Articles 1–100". Așadar se observă că nu mai există nicio modificare de la ultima apăsare a butonului. Acest lucru este necesar deoarece butonul "Show more" devine disabled de fiecare dată după ce este apăsate pt câteva momente. Așa că Dacă ar rămâne doar prima condiție am putea fi fals convinși că utilizatorul are mai puține publicații decât în realitate.

Pentru a nu face exces de request-uri am adăugat un sleep de 2 secunde, acesta dă timp butonului Show more să redevină enabled, dar ajută și la performanță. De asemenea, evită posibilitatea de a fi blocați de Google Scholar din cauza numărului de requesturi excesive.

## 4.2.2 Beautiful Soup

După ce am extras conținutul în întregime al profilului Google scholar putem folosi aceleași tehnici pentru a extrage datele specifice ale publicațiilor.

Diferit de dblp, pe google scholar extragem de asemenea numărul citărilor publicațiilor și link-ul care duce spre pagina acestora.

## 4.3 Citări Google Scholar `citations.py`

### 4.3.1 Scraper API

Acest script este responsabil pentru extragerea citărilor din Google Scholar. Diferit de extragerea publicațiilor aici se folosește API-ul ScraperAPI pentru a accesa și parsa paginile web.

Acest lucru a fost necesar deoarece în urma a mai multor request-uri de test IP-ul calculatorului a fost blocat. Așa că am avut nevoie de un intermediar pentru obținerea conținutului HTML al citărilor.

Nu am întâmpinat aceeași problemă la obținerea conținutului de la publicații, dar în caz că acest lucru s-ar întâmpla, utilizarea unui intermediar va fi necesară și acolo.

Pentru a folosi Scraper API trebuie făcut un cont. Fiecărui cont îi este asociată cheie care poate fi folosită ca parolă pentru obținerea datelor dorite.

```
def get_html(url):
    try:
        payload = { 'api_key': config.SCRAPER_API_KEY, 'url': url }
        response = requests.get('https://api.scraperapi.com/', params=payload)
        if response.status_code != 200:
            print("Error: Failed to retrieve the webpage")
            return
        html_content = response.content
        return html_content
    except Exception as e:
        print(f"error while getting html: {e}")
```

## 4.3.2 Organizare în pagini

scholar.google.com/scholar?start=20&hl=en&as\_sdt=2005&sciodt=0,5&cites=13920903996128028090&scpsc=

... Rewriting Logic and its Applications via Rusa (Ed.) 129 LNCS 11132 12th International Workshop, WRLA 2018 Held as a Satellite Event of ETAPS Thessaloniki, Greece, June 14–15 ...

☆ Save Cite Cited by 1 Related articles All 7 versions

[PDF] Local completeness logic on strategic term rewriting [PDF] github.io

N Piazzesi, R Bruni - piazzesianiccolo.github.io

Abstract In Abstract Interpretation, completeness is an ideal situation in which the abstraction does not introduce any false alerts. Completeness is notoriously difficult to achieve in ...

☆ Save Cite Related articles

Extending *Timbuk* to Verify Functional Programs [PDF] hal.science

T Genet, T Gillard, T Haudebourg... - ... Held as a Satellite Event of ..., 2018 - Springer

Abstract Timbuk implements the Tree Automata Completion algorithm whose purpose is to over-approximate sets of terms reachable by a term rewriting system. Completion is ...

☆ Save Cite Cited by 1 Related articles All 7 versions

Characterizing conclusive approximations by logical formulae [PDF] hal.science

Y Boichut, TBH Dao, V Murat - International Workshop on Reachability ..., 2011 - Springer

Considering an initial set of terms  $E$ , a rewriting relation  $R$  and a goal set of terms  $Bad$ , reachability analysis in term rewriting tries to answer to the following question: does there ...

☆ Save Cite Cited by 2 Related articles All 11 versions

Vers un outil de vérification formelle légère pour OCaml [PDF] hal.science

T Genet, B Kordy, A Vansynge - AFADL, 2015 - inria.hal.science

Si l'on décrit, par une grammaire, l'ensemble des entrées possibles d'un programme fonctionnel, peut-on connaître la grammaire des sorties de celui-ci? Il existe des outils en ...

☆ Save Cite Cited by 4 Related articles All 11 versions

< Googoooooogle >

Previous 1 2 3 4 5 6 Next

Diferit de obținerea html-ului pentru publicații care într-un final ajungeau să fie toate pe o pagină după direct(dblp) sau în urma unor acțiuni(Google Scholar) citările sunt organizate mereu în pagini.

Analizând fiecare pagină am găsit următoarele pattern-uri:

1. Fiecare pagină are câte 10 citări sau mai puțin în cazul ultimei pagini.
2. Pe lângă butoanele de interacționare din josul paginii de unde poți schimba paginile, în url-url paginii există o variabilă query care se schimbă, aceasta este reprezentată parametrul query `start` care de la o pagină la alta are cu o valoare crescută cu 10. Diferită fiind prima pagină în care acesta nu există.

Așadar pentru a obține toate citările unei publicații se începe cu url-ul dobândit din linkul de la numărul de citări din dreptul publicației corespunde, a căror informații le-am aflat în `scholar.py`, se obține html-ul prin utilizarea Scraper Api-ului și se incrementează valoarea parametrului `start` cu 10, creând un nou url cu ea a cărui conținut îl aflăm la fel, până când numărul de citări este mai mic decât 10 (am ajuns la ultima pagină) sau numărul de citări este 0(Ultima pagină a avut exact 10 citări).



### 4.3.3 Beautiful Soup

În momentul în care avem un url care conține citări, utilizăm Beautiful Soup pentru a afla valorile dorite.

Față de publicații, informațiile citărilor nu sunt la fel de clar determinate.

Titlul citării și linkul către aceasta sunt cel mai ușor de dobândit, pentru că respectă același procedeu descris și la publicații.

În schimb celelalte date sunt doar în chenarul încercuit cu roșu în poza de mai sus. Acest chenar uneori nu este complet, conținând "..." în anumite cazuri sau neavând unele câmpuri.

Analizând acel chenar cu informații am descoperit următorul pattern:

- câmpurile diferite sunt separate prin "-".
- Primul câmp este reprezentat de numele autorilor cu inițiala prenumelui și numele de familie.
- Al doilea câmp poate fi conferința, dar nu mereu.
- După conferință urmează anul, dar nu mereu.
- La finalul câmpului este un link.

Pentru a detecta anul se folosește o expresie regex, detectând toate numerele între 1900 și 2299:

```
year_match = re.search(r'\b(19\d{2}|20\d{2}|21\d{2}|22\d{2})\b',  
authors_string)
```

# Capitolul 5

## Descrierea REST API-ului

### 5.1 Proprietăți RESTful

API-ul proiectului este conceput conform principiilor RESTful:

- **Stateless:** Toate cererile de la client care ajung în server conține toate datele necesare pentru a fi prelucrate de server, nefiind necesar ca server-ul de reține nimic despre starea clientului între cereri. Acest lucru este posibil datorită autentificării și autorizării realizate prin intermediul JWT.
- **Uniform Interface:** API-ul proiectului utilizează metodele HTTP: GET, POST, PUT, DELETE, iar pentru identificarea resurselor se folosește URI (Uniform Resource Identifier) . Acestea sunt niște convenții standard pentru resurse.
- **Arhitectura Client-Server:** Separarea clientului de server permite evoluția independentă a ambelor părți.
- **Cacheable:** Răspunsurile de la server pot fi cache-uite pentru a îmbunătăți performanța.
- **Layered System:** În momentul în care clientul accesează serverul, acesta nu știe dacă el interacționează direct serverul sau cu un intermediar al său. De exemplu, aplicația utilizează Scraper API pentru a obține datele despre citări de pe Google Scholar, astfel acționează ca un intermediar între utilizator și sursa de date.

### 5.2 Autorizare endpoint-uri

Toate endpoint-urile, în afara celor de GET /login și POST /user trebuie să conțină tokenul într-un header 'x-access-token'.

```
header: 'x-access-token': 'eyJ0....'
```

În acest fel este asigurată autentificarea și securitatea informațiilor. Un utilizator are acces la informațiile sale doar dacă dovedește că el este realul deținător al acelor informații. El face acest lucru trimițând în request tokenul de autentificare primit la logare, atunci când el a introdus numele de utilizator și parola corectă.

GET /login este o excepție pentru că acest request reprezintă metoda de autentificare și obținere a tokenului.

POST /user este o excepție pentru că oricine are voie să își creeze un cont, fără niciun fel de autentificare prealabilă.

## 5.3 Endpoint-uri

### 5.3.1 Autentificare și Autorizare

[/login](#)

- **Metodă:** GET
- **Descriere:** Autentificarea utilizatorului.
- **Response:**

- **Status Code 200 OK:**

```
{  
  "token": "eyJ0..."  
}
```

- **Status Code 400 BAD REQUEST:**

```
{  
  "error": "Bad Request: Username and password required."  
}
```

- **Status Code 401 UNAUTHORIZED:**

```

{
    "error": "Could not verify"
}

sau

{
    "error": "Username does not exist in the database"
}

```

### 5.3.2 Utilizatori

[/user](#)

- **Metodă:** POST
- **Descriere:** Crearea unui cont nou de utilizator.
- **Request:**

```

{
    "username": "user1",
    "password": "parola",
    "name": "User Good",
    "dblp_profile": "https://dblp.org/pid/...",
    "scholar_profile": "https://scholar.google.com/..."
}

```

- **Response:**

– **Status Code 201 CREATED:**

```

{
    "result": "User added",
    "user_id": "60883h4..."
}

```

– **Status Code 400 BAD REQUEST:**

```
{
  "errors": "Field 'username' - Shorter than minimum length 1.
Field 'password' - Shorter than minimum length 1.
Field 'name' - Shorter than minimum length 1.
Field 'scholar_profile' - Not a valid URL.
Field 'dblp_profile' - Not a valid URL."
}
```

– **Status Code 409 CONFLICT:**

Description: Username already exists

– **Status Code 500 INTERNAL SERVER ERROR:**

Description: An error occurred: {e}

[/user/{user\\_id}](#)

- **Metodă:** GET
- **Descriere:** Obținerea informațiilor despre un utilizator.
- **Response:**

– **Status Code 200 OK:**

```
{
  "_id": "60c5baeb8c62b91440...",
  "username": "numeutilizator1",
  "name": "utilizator unu",
  "dblp_profile": "https://dblp.org/pid/...",
  "scholar_profile": "https://scholar.google.com/..."
}
```

– **Status Code 404 NOT FOUND:**

```
{
```

```
"error": "User not found"
}
```

– **Status Code 500 INTERNAL SERVER ERROR:**

Description: An error occurred: {e}

[/user/{user\\_id}](#)

- **Metodă:** DELETE
- **Descriere:** Ștergerea unui utilizator.
- **Response:**

– **Status Code 200 OK:**

```
{
    "result": "User deleted"
}
```

– **Status Code 404 NOT FOUND:**

Description: User not found

– **Status Code 500 INTERNAL SERVER ERROR:**

Description: An error occurred: {e}

### 5.3.3 Publicație unică

[/publications/{publication\\_id}/scholar](#)

- **Metodă:** GET
- **Descriere:** Obținerea informațiilor despre o anumită publicație Scholar.
- **Response:**

– **Status Code 200 OK:**

```
{  
  "title": "Publication Title",  
  "authors": ["Author One", "Author Two"],  
  "year": 2021  
}
```

– **Status Code 400 BAD REQUEST:**

```
{  
  "message": "Publication does not have scholar data."  
}
```

– **Status Code 404 NOT FOUND:**

```
{  
  "error": "Publication not found"  
}
```

– **Status Code 500 INTERNAL SERVER ERROR:**

Description: An error occurred: {e}

[/publications/{publication\\_id}/split](#)

- **Metodă:** POST
- **Descriere:** Separarea unei publicații considerate match în publicații distincte DBLP și Scholar.
- **Response:**

– **Status Code 200 OK:**

```
{  
  "message": "Publication split successfully",  
  "dblp_id": "60c5baeb8c62b91440...",  
}
```

```
    "scholar_id": "60c5baeb8c62b91441..."
  }
```

– **Status Code 400 BAD REQUEST:**

```
{
  "message": "Matched publication does not contain
  both DBLP and Scholar data"
}
```

– **Status Code 404 NOT FOUND:**

```
{
  "error": "Matched publication not found"
}
```

– **Status Code 500 INTERNAL SERVER ERROR:**

Description: An error occurred: {e}

[/publications/{publication\\_id}](#)

- **Metodă:** PUT
- **Descriere:** Actualizarea unei publicații existente sau crearea uneia noi.
- **Request:**

```
{
  "userId": "60c5baeb8c62b91440...",
  "updatedData": {
    "title": "Updated Title",
    "authors": ["Author One", "Author Two"],
    "year": 2021,
    "conference": "Sample Conference",
    "category": "A",
  }
}
```



```

        "indexed": "ISI",
        "type": "journal",
        "impactFactor": 3.5
    }
}

```

- **Response:**

- **Status Code 200 OK:**

```

{
    "message": "Publication updated successfully"
}

```

- **Status Code 201 CREATED:**

```

{
    "message": "Publication created successfully' "
}

```

- **Status Code 400 BAD REQUEST:**

```

{
    "message": "userId or updatedData missing"
}

```

- **Status Code 404 NOT FOUND:**

```

{
    "error": "Publication not found or does
            not belong to this user"
}

```

- **Status Code 500 INTERNAL SERVER ERROR:**

Description: An error occurred: {e}

sau

```
{'error': 'Failed to create publication'}  
sau  
{'error': 'Failed to update publication'}
```

### 5.3.4 Publicații

[/publications/user/{user\\_id}/refresh](/publications/user/{user_id}/refresh)

- **Metodă:** POST
- **Descriere:** Actualizarea publicațiilor utilizatorului din surse externe.
- **Response:**

– **Status Code 200 OK:**

```
[  
  {  
    "_id": "60c5baeb8c62b91440...",  
    "title": "Sample Publication",  
    "authors": ["Author One", "Author Two"],  
    "year": 2021,  
    "source": "DBLP"  
  },  
  ...  
]
```

– **Status Code 404 NOT FOUND:**

```
{  
  "error": "User not found"  
}
```

– **Status Code 500 INTERNAL SERVER ERROR:**

```
{"error": "An error occurred: {e}"}
```

[/publications/user/{user\\_id}](#)

- **Metodă:** GET
- **Descriere:** Obținerea publicațiilor unui utilizator.
- **Response:**

– **Status Code 200 OK:**

```
[
  {
    "_id": "60cdhhd7...",
    "title": "Publicație Titlu",
    "authors": ["Autorul Unu", "Nume Prenume"],
    "year": 2021,
    "source": "DBLP"
  },
  ...
]
```

– **Status Code 404 NOT FOUND:**

```
{
  "error": "No documents found for this user in the database.
           Try refreshing the data."
}
```

– **Status Code 500 INTERNAL SERVER ERROR:**

```
{"error": "An error occurred: {e}"}
```

### 5.3.5 Citări

[/citations/refresh](#)

- **Metodă:** POST

- **Descriere:** Actualizarea citărilor pentru o publicație specifică.

- **Request:**

```
{  
  "userId": "60c5baeb8c654314...",  
  "publicationId": "60c42423eb8b91440..."  
}
```

- **Response:**

- **Status Code 200 OK:**

```
{  
  "message": "Citations refreshed successfully"  
}
```

- **Status Code 400 BAD REQUEST:**

```
{  
  "error": "userId or publicationId missing"  
}
```

- **Status Code 404 NOT FOUND:**

```
{"error": "User not found"}  
  
sau  
  
{"error": "Publication not found"}
```

- **Status Code 500 INTERNAL SERVER ERROR:**

Description: An error occurred: {e}

[/citations/update](#)

- **Metodă:** POST

- **Descriere:** Actualizarea unei citări existente.

- **Request:**

```
{
  "userId": "60c5baeb8c62b91440...",
  "citationId": "60c5baeb8c62b91441...",
  "updatedData": {
    "title": "New Citation Title",
    "authors": ["Author One", "Author Two"],
    "year": 2021,
    "conference": "Sample Conference",
    "indexed": "ISI",
    "impactFactor": 3.5
  }
}
```

- **Response:**

- **Status Code 200 OK:**

```
{
  "message": "Citation updated successfully"
}
```

- **Status Code 400 BAD REQUEST:**

```
{
  "error": "userId or updatedData missing"
}
```

- **Status Code 404 NOT FOUND:**

```
{
  "error": "Citation not found or does not belong
           to this user"
```

```
}
```

– **Status Code 500 INTERNAL SERVER ERROR:**

Description: An error occurred: {e}

[/citations/{publication\\_id}](#)

- **Metodă:** GET
- **Descriere:** Obținerea tuturor citărilor pentru o publicație specifică.
- **Response:**

– **Status Code 200 OK:**

```
[  
  {  
    "_id": "60c5baeb8c62b91440...",  
    "citation": {  
      "title": "Nume",  
      "authors": ["Autor1", "Autor2"],  
      "year": 2021,  
      "conference": "Conferinta",  
      "link": "http://exemplu.com"  
    }  
  },  
  ...  
]
```

– **Status Code 404 NOT FOUND:**

```
{  
  "error": "No citations found for this publication for the gi"  
}
```

– **Status Code 500 INTERNAL SERVER ERROR:**

Description: An error occurred: {e}

### 5.3.6 Generare Rapoarte

[/generate](#)

- **Metodă:** POST
- **Descriere:** Generarea unui raport în format TXT, Excel sau CSV.
- **Request:**

```
{  
  "format": "txt", // sau "excel", "csv"  
  "ids": ["60c5baeb8c62b91440...", "60c5baeb8c62b91441..."],  
  "type": "publications",  
  "fields": ["title", "authors", "year"]  
}
```

- **Response:**

- **Status Code 200 OK:**

```
(attachment: publications.txt / publications.xlsx /  
  publications.csv)
```

- **Status Code 400 BAD REQUEST:**

```
{"error": "Invalid format type"}
```

- **Status Code 500 INTERNAL SERVER ERROR:**

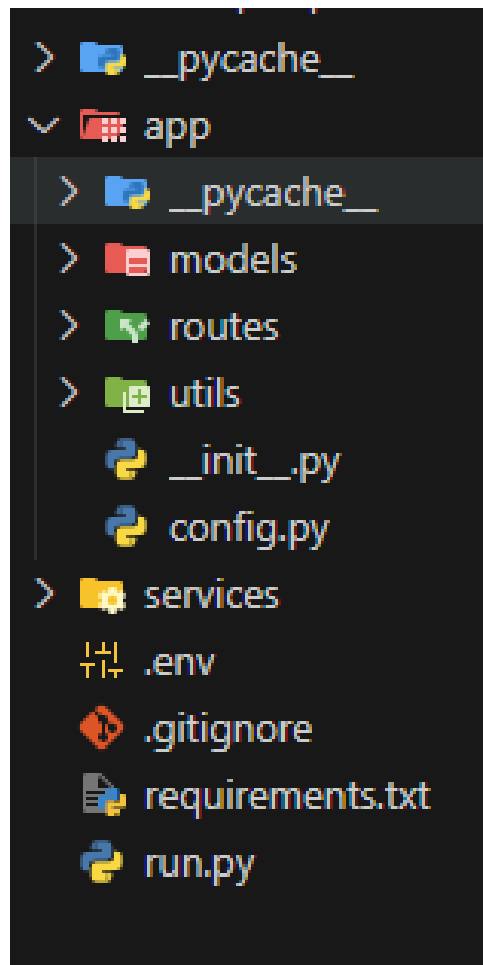
Description: An error occurred: {e}

## 5.4 Formate de Cerere/Răspuns și Exemple de Utilizare



































Toate cererile către API trebuie să includă un token de autentificare în antetul cererii. Exemplele de mai sus ilustrează formatul cererilor și răspunsurilor pentru diverse endpoint-uri.

Aceste exemple pot fi utilizate pentru a testa și valida funcționalitatea API-ului.

## 5.5 Structura codului





- ▼  app
  - >  \_\_pycache\_\_
- ▼  models
  - >  \_\_pycache\_\_
    -  \_\_init\_\_.py
    -  report.py
    -  user.py
- ▼  routes
  - >  \_\_pycache\_\_
    -  \_\_init\_\_.py
    -  auth.py
    -  citations.py
    -  publications.py
    -  report.py
    -  user\_publications.py
    -  user.py
- ▼  utils
  - >  \_\_pycache\_\_
    -  \_\_init\_\_.py
    -  auth.py
    -  publications.py
    -  report.py
    -  validation.py
    -  \_\_init\_\_.py
    -  config.py
- ▼  services
  - >  \_\_pycache\_\_
    -  citations.py
    -  dblp.py
    -  scholar.py
-  .env
-  .gitignore
-  requirements.txt
-  run.py

## 5.6 Detalierea funcționalităților

### 5.6.1 Înregistrare

Adăugarea unui nou utilizator în API este reprezentată de un request POST la entrypoint-ul `"/user"` alături de un JSON în body care reprezintă informațiile acestuia.

Prima dată se va face o verificare a schemei obiectului introdus în request, dacă aceasta respectă schema utilizatorului:

#### Nume de utilizator (Username)

- **Lungime minimă:** Numele de utilizator trebuie să aibă cel puțin 4 caractere.
- **Caracter valid:** Numele de utilizator poate conține doar litere și cifre (fără spații, caractere speciale sau simboluri).

#### Parolă

- **Majusculă:** Parola trebuie să conțină cel puțin o literă majusculă.
- **Minusculă:** Parola trebuie să conțină cel puțin o literă minusculă.
- **Cifră:** Parola trebuie să conțină cel puțin un număr.
- **Caracter special:** Parola trebuie să conțină cel puțin un caracter special dintre următoarele: `@ $ ! % * ? & + # ^ ( ) { }`
- **Lungime minimă:** Parola trebuie să aibă puțin 8 caractere.

#### Profil Scholar

- **URL Valid:** URL-ul profilului Scholar trebuie să conțină `scholar.google.com/` sau poate fi valoarea booleană `False`.

#### Profil DBLP

- **URL Valid:** URL-ul profilului DBLP trebuie să conțină `https://dblp.org/` sau poate fi valoarea booleană `False`.

În caz că nu respectă, se va returna Status Code 400 BAD REQUEST alături de informații despre ce câmp nu este corect sau lipsește în body.

Pentru a optimiza securitatea, parola se va cripta într-un sha256 și în baza de date se va introduce doar acest hash al parolei.

```
hashed_password = generate_password_hash(data['password'], method='sha256')
new_user = User(_id=data["username"], data=data, password=hashed_password)
```

Apoi se încearcă introducerea în baza de date a noului utilizator. Dacă numele de utilizator al acestuia există deja pentru altcineva, o eroare "duplicate key error" va fi aruncată și procesată, deoarece username este o cheie primară în baza de date pentru colecția utilizatorilor. Așadar, nu pot exista două persoane cu același username.

Se va returna Status Code 409 Conflict cu mesajul "Username already exists".

### 5.6.2 Autentificare

Inițial se verifică dacă s-au primit toate informațiile necesare în cerere (username și password), dacă nu s-au trimis se va returna status code 400 BAD REQUEST. Apoi se caută utilizatorul în baza de date după username și se compară parolele, dacă acestea coincid, se va genera un JWT (JSON WEB TOKEN) care va fi valabil 24 de ore. Acest token va fi răspunsul la loginul valid.

### 5.6.3 Detectarea Duplicatelor

Pentru a detecta duplicatele se extrag mai întâi toate publicațiile de pe DBLP, apoi toate publicațiile de pe Google Scholar și apoi se apelează următoarea funcție:

```

def find_similar_publications(dblp_data, scholar_data):
    matches = []
    scholar_keys_to_ignore = set()

    if isinstance(dblp_data, dict):
        for key1, pub1 in dblp_data.items():
            if not isinstance(pub1, dict):
                continue
            title1 = normalize_title(pub1.get('title', ''))
            authors1 = normalize_authors([author for author in pub1.get('authors', [])])
            year1 = pub1.get('year', '')

            matched = False
            if isinstance(scholar_data, dict):
                for key2, pub2 in scholar_data.items():
                    if not isinstance(pub2, dict):
                        continue

                    title2 = normalize_title(pub2.get('title', ''))
                    authors2 = normalize_authors(pub2.get('authors', []))
                    year2 = pub2.get('year', '')

                    if similar(title1, title2) > 0.7 and match_authors(authors1, authors2) and year1 == year2:
                        matches.append({"dblp": pub1, "scholar": pub2, "match": True})
                        scholar_keys_to_ignore.add(key2)
                        matched = True
                        break

            if not matched:
                pub1['source'] = 'dblp'
                matches.append({"dblp": pub1})

    if isinstance(scholar_data, dict):
        for key2, pub2 in scholar_data.items():
            if key2 in scholar_keys_to_ignore or not isinstance(pub2, dict):
                continue

            pub2['source'] = 'scholar'
            matches.append({"scholar": pub2})

    return matches

```

Toate șirurile de caractere se normalizează mai întâi, transformând toate literele în minuscule, și eliminând caracterele speciale.

Similaritatea titlurilor se calculează cu SequenceMatcher din modulul difflib. Aceasta calculează raportul de similaritate între două șiruri de caractere. Acest raport trebuie să fie mai mare decât 0.7 pentru a fi considerat un duplicat.

Pentru a calcula similaritatea autorilor, se normalizează stringurile, apoi se extrage pentru fiecare doar numele de familie. Obținând o listă cu nume de familie. Se face acest lucru deoarece pe Google Scholar apărea doar inițiala prenumelui și numele de familie. După ce se calculează listele de nume de familie, raportul de similaritate a autorilor se calculează astfel:

```
def match_authors(authors1, authors2):
    # Extract last names for comparison
    set1 = extract_last_names(authors1)
    set2 = extract_last_names(authors2)

    # Calculate the intersection and union
    intersection = set1.intersection(set2)
    union = set1.union(set2)
    return len(intersection) / len(union) > 0.5 # Adjust threshold as needed
```

Raportul este lungimea intersecției împărțită la lungimea uniunii celor două seturi de nume de familie.

Acest raport trebuie să fie mai mare decât 0.5. Dar acest număr poate fi modificat.

Anul trebuie să fie același la ambele publicații.

#### 5.6.4 Refresh Optim

La fiecare refresh se extrag toate datele din DBLP și Scholar, dar se trece prin fiecare și se adaugă doar publicațiile și citările noi.

Această metodă este optimă deoarece nu se pierde date adăugate de utilizator între timp, date care s-ar pierde dacă la fiecare refresh s-ar înlocui toate datele, dar se economisește și spațiu, care ar fi prea plin dacă s-ar adăuga mereu citările și publicațiile fără a fi verificate.

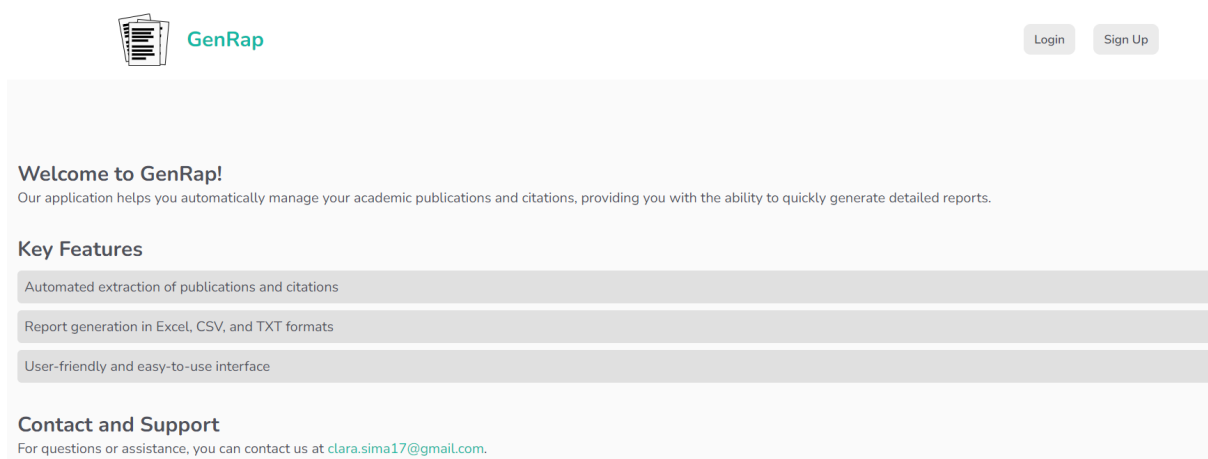
#### 5.6.5 Generarea Rapoartelor

Aplicația poate genera rapoarte în trei formate diferite: TXT, Excel și CSV, încercând să se adapteze la nevoile diferite ale utilizatorilor. De asemenea, în urma fiecărui raport se salvează preferințele câmpurilor utilizatorului.

# Capitolul 6

## Interfața VueJS

### 6.1 Pagina de Întâmpinare pentru Persoanele Neautentificate



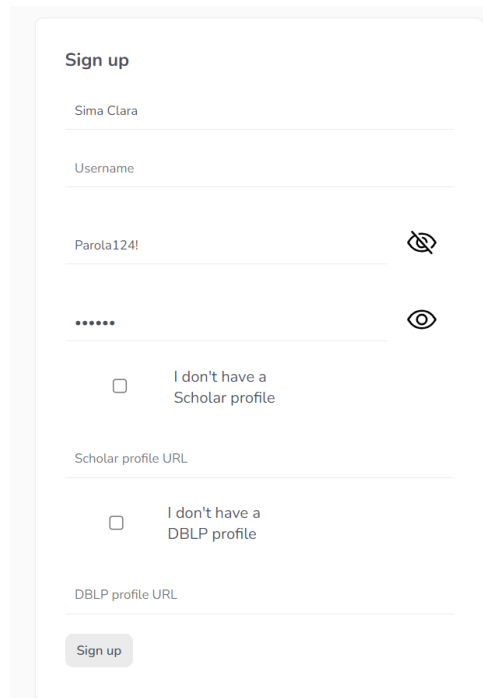
Prima pagină accesată de orice nou utilizator ori utilizator neautentificat în ultima zi, va fi o pagină în care este prezentată aplicația și singurele variante din meniu pe care le va avea vor fi:

- Login (Autentificare)
- Sign Up (Înregistrare)

Dacă în momentul în care un utilizator accesează aplicația web, acesta are salvat în localStorage un token de autentificare și token-ul este valid el va fi direct autentificat. Așadar, în acest caz utilizatorul nu va vedea această pagină.

## 6.2 Înregistrare

Aici utilizatorul poate să își aleagă numele de utilizator care este unic (o eroare va apărea după ce apasă sign up dacă numele este deja luat), o parolă și poate să introducă linkurile către profilele sale de Google Scholar și/sau DBLP. El poate de asemenea să bifeze două checkbox-uri în caz că nu deține unul dintre cele două profile.

A screenshot of a 'Sign up' form. It contains input fields for 'Name' (filled with 'Sima Clara'), 'Username', 'Password' (filled with 'Parola124!'), and a confirmation password field (filled with '\*\*\*\*\*'). There are two checkboxes: 'I don't have a Scholar profile' and 'I don't have a DBLP profile'. Below each checkbox is a corresponding 'Scholar profile URL' or 'DBLP profile URL' input field. A 'Sign up' button is at the bottom.

În caz că a fost o problemă cu informațiile introduse după ce se apasă Sign Up va apărea o listă cu toate problemele existente. În imagine se poate vedea ce se afișează dacă cineva apasă Sign Up fără să completeze nimic.

Field 'password' - Password must contain at least one special character @\$!%\*?&+#+^(){}]

Dacă numele de utilizator a fost utilizat deja de altă persoană atunci următorul mesaj va apărea:

csima has been used before. Please try another username.

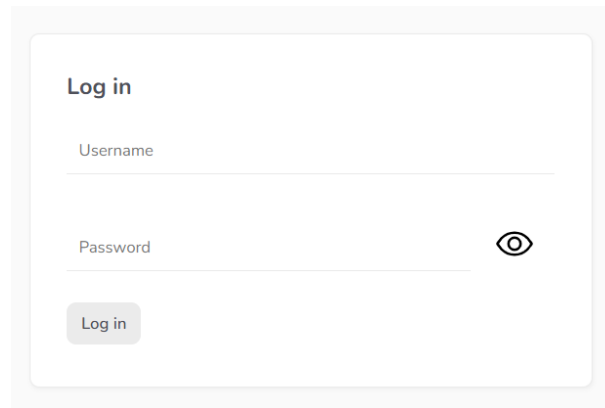
Dacă contul a fost creat cu succes atunci va apărea un mesaj de confirmare:

You signed up successfully, Clara. You will be redirected to Login Page in 3 seconds

[Go to Login now.](#)

## 6.3 Autentificare

Utilizatorii se pot autentifica în aplicație folosind un nume de utilizator și o parolă.



Dacă lipsește username-ul sau parola sau datele introduse nu sunt corecte se va afișa:

Incorrect login credentials

Pentru autorizare se face o cerere GET la endpoint-ul `"/login"` care conține un header de autorizare `"Authorization"` cu textul `"Basic"` ceea ce este tipul autorizării, împreună cu numele de utilizator și parola. Așa arată cererea făcută din interfață:

```
// Basic Auth: base64 encode the 'username:password' string
const authHeader = 'Basic ' + btoa(`${username}:${password}`)

try {
  const response = await axios.get(url, {
    headers: {
      Authorization: authHeader
    }
  })
}
```

## 6.4 Meniul principal din interfață

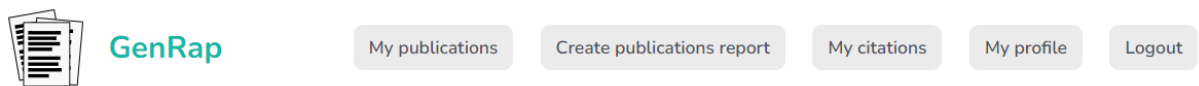
Logat fiind userul are 5 opțiuni în meniul de sus:

- My publications



- Create publications report
- My citations
- My profile
- Logout

Acest meniu este afișat mereu în partea de sus a paginii, la orice rută a aplicației te-ai afla. Funcționalitatea acestui meniu este realizată prin intermediul unui router, care este responsabil pentru navigarea între diferitele pagini și secțiuni ale aplicației. Acesta utilizează biblioteca vue-router pentru a crea rute și a gestiona schimbările dintre pagini.



Înainte de fiecare schimbare a rutei se verifică dacă există un token valid în caz că ruta la care ne ducem are nevoie de autorizare, dacă token-ul nu există sau a expirat utilizatorul este redirecționat la pagina de Log In.

```
router.beforeEach((to, from, next) => {  
  const token = getToken();  
  const publicPages = ['login', 'signup', 'home'];  
  if (token || publicPages.includes(to.name)) {  
    next();  
  } else {  
    next({ name: 'login' });  
  }  
});
```

## 6.5 Gestionarea Publicațiilor (My Publications)

GenRap

My publications Create publications report My citations My profile Logout

Total Publications 57 Scholar 48 DBLP 32

Update data from dblp and scholar Add publication

**New** Proof-carrying parameters in certified symbolic execution

Authors: A Arusoaie, D Lucanu  
Year: 2024  
Source: Scholar  
Edit

**✓** AIM-RL: A New Framework Supporting Reinforcement Learning Experiments

Authors: Ionut Cristian Pistol, Andrei Arusoaie  
Year: 2023  
Type: conference  
Indexed: BDI  
Category: B  
Source: dblp  
Edit

**🔗** Identifying Vulnerabilities in Smart Contracts using Interval Analysis

Authors: Stefan-Claudiu Susan, Andrei Arusoaie  
Year: 2023  
Source: dblp  
Edit

### 6.5.1 Butonul Update data form dblp and scholars

Acesta face un POST la `/publications/user/<user_id>/refresh` pentru a actualiza toate publicațiile din baza de date.

### 6.5.2 Tag-ul NEW!

Publicațiile care au fost adăugate la ultimul refresh vor avea tag-ul New!

**New** Proof-carrying parameters in certified symbolic execution

Authors: A Arusoaie, D Lucanu  
Year: 2024  
Source: Scholar  
Edit

### 6.5.3 Prima accesare a unui utilizator nou

În momentul în care un nou utilizator accesează pagina `My Publications` acesta va vedea:

The screenshot shows the GenRap application header with navigation links: "My publications", "Create publications report", "My citations", "My profile", and "Logout". Below the header, three summary cards are displayed: "Total Publications" with a value of 0, "Scholar" with a value of 0, and "DBLP" with a value of 0. Below these cards, there is a button labeled "Update data from dblp and scholar". A red error message states: "No documents found for this user in the database. Try refreshing the data." At the bottom, there is a button labeled "Add publication".

El va trebui să apese pe butonul `Update data from dblp and scholars` pentru a declanșa extragerea informațiilor.

#### 6.5.4 Afișarea publicațiilor deja extrase

Dacă utilizatorul a apăsat vreodată înainte butonul `Update data from dblp and scholars` atunci următoarea dată când acesta vizitează această pagină, publicațiile obținute în urma refreshului trecut vor apărea direct pe pagină.

#### 6.5.5 Status cu numărul de publicații obținute

The screenshot shows the same GenRap application header. The summary cards now display non-zero values: "Total Publications" is 57, "Scholar" is 48, and "DBLP" is 32.

Numărul total este calculat așa:

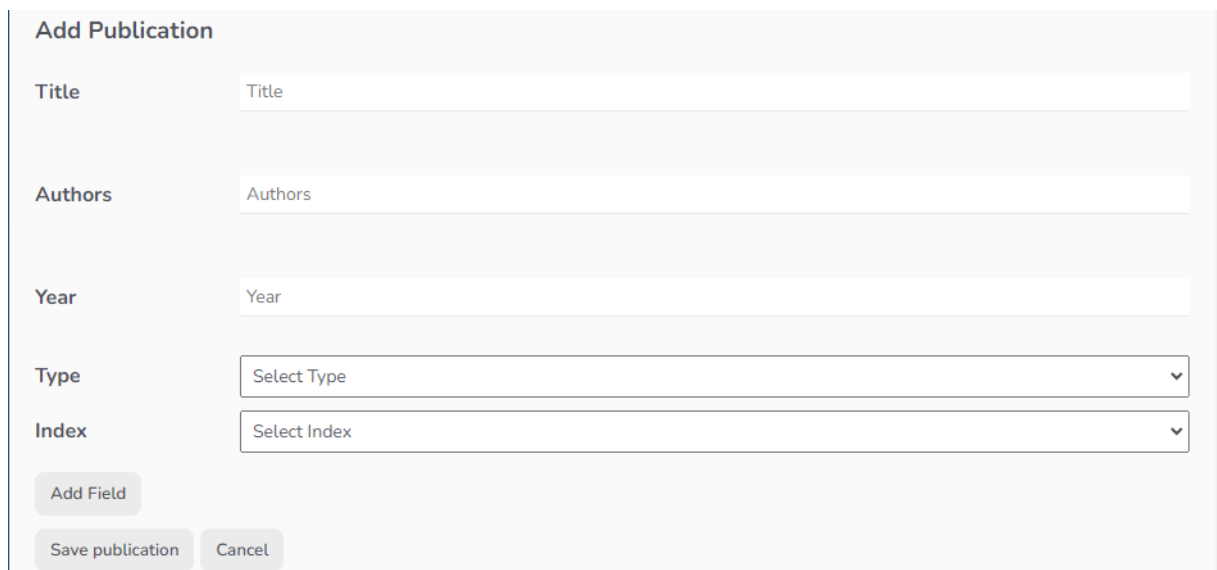
$$(S \cap D) \cup (S \setminus D) \cup (D \setminus S)$$

unde:

- $S$  reprezintă mulțimea publicațiilor din Scholar,
- $D$  reprezintă mulțimea publicațiilor din DBLP,
- $\cap$  reprezintă intersecția,
- $\cup$  reprezintă reuniunea,
- $\setminus$  reprezintă diferența dintre mulțimi.

### 6.5.6 Butonul Add Publication

Utilizatorul are posibilitatea de adăuga o publicație nouă. Acesta este formularul care trebuie completat pentru acest lucru:



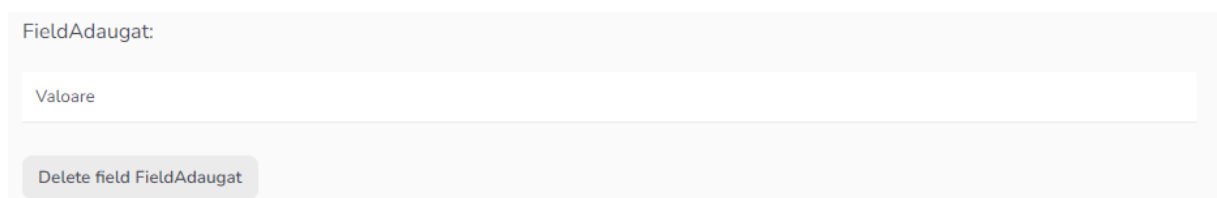
The 'Add Publication' form is a light gray rectangular box. At the top left, the title 'Add Publication' is displayed in a bold, dark font. Below the title, there are five input fields arranged vertically. The first three are text inputs: 'Title', 'Authors', and 'Year', each with its label to the left and a light gray input box to the right. The fourth and fifth are dropdown menus: 'Type' and 'Index', each with its label to the left and a light gray box containing a placeholder text ('Select Type' and 'Select Index' respectively) and a downward arrow icon on the right. At the bottom left of the form, there is a small 'Add Field' button. At the bottom right, there are two buttons: 'Save publication' and 'Cancel'.

Aici există niște câmpuri by default care pot fi completate, dar apare și opțiunea Add Field.



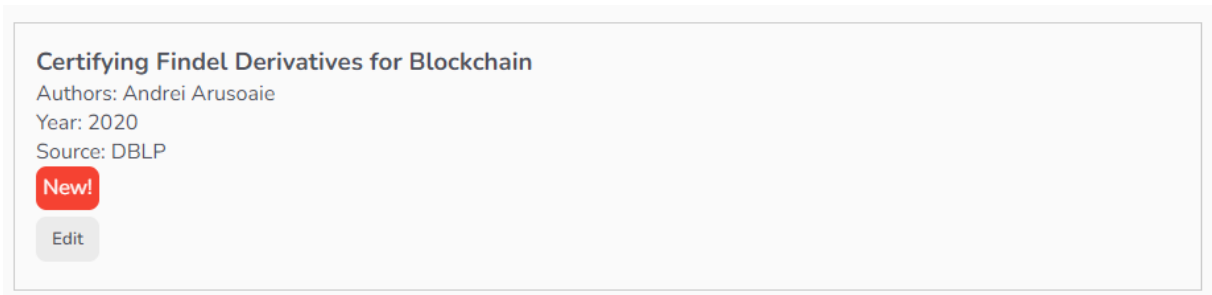
The 'Add Field' dialog is a light gray rectangular box. At the top left, there is a 'Cancel Add Field' button. Below it, there are two text input fields: 'Field Name' and 'Field Value', each with its label to the left and a light gray input box to the right. At the bottom left, there is a 'Confirm Adding Field' button.

După ce au fost adăugate niște câmpuri în plus, acestea pot fi și șterse dacă utilizatorul se răzgândește. El trebuie să apese pe butonul `Delete field`



The 'Delete field' dialog is a light gray rectangular box. At the top left, the text 'FieldAdaugat:' is displayed. Below it, there is a text input field with the label 'Valoare' to its left. At the bottom left, there is a 'Delete field FieldAdaugat' button.

## 6.5.7 Butonul Edit: Editare Manuală

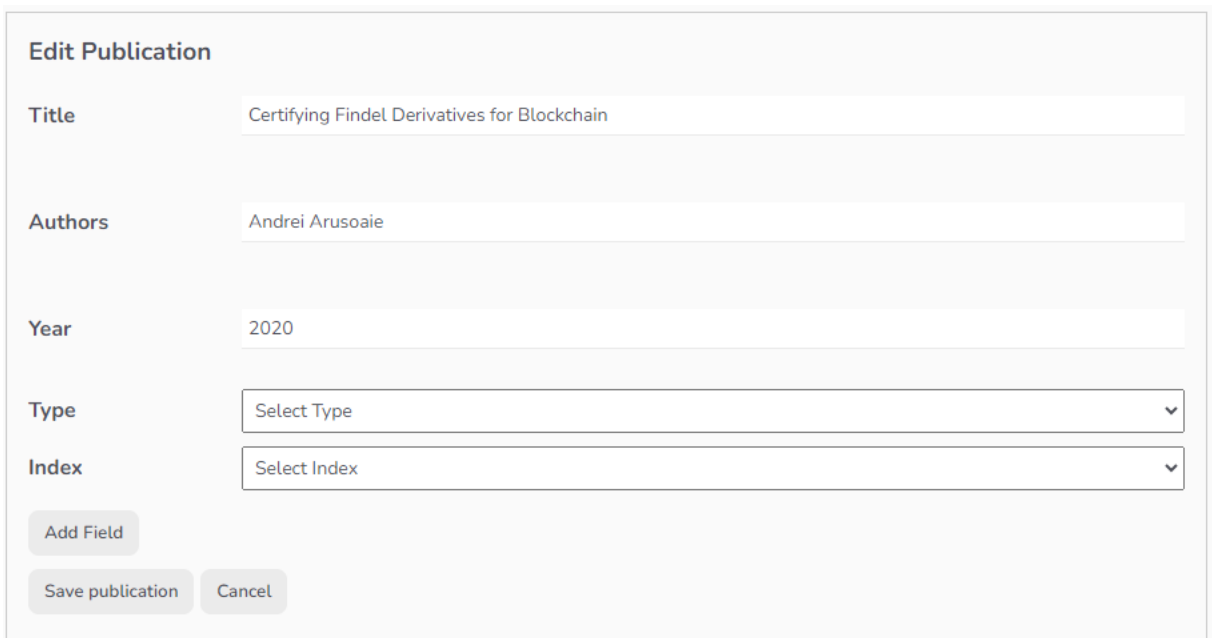


**Certifying Findel Derivatives for Blockchain**  
Authors: Andrei Arusoaie  
Year: 2020  
Source: DBLP

New!

Edit

Utilizatorii au dreptul să editeze o aplicație. Ei pot apăsa butonul Edit. Astfel ei pot să editeze valorile deja existente, să completeze câmpurile necompletate sau să adauge câmpuri noi.



**Edit Publication**

Title: Certifying Findel Derivatives for Blockchain

Authors: Andrei Arusoaie

Year: 2020

Type: Select Type

Index: Select Index

Add Field

Save publication Cancel

Câmpurile Category și Impact Factor apar în funcție de ce a fost selectat la Index și Category.

Câmpurile deja completate sunt: Title, Authors și Year. Apoi mai apar două câmpuri din care ai opțiuni de selectare. Câmpurile sunt Type: Conference, Journal, Workshop, Book și Index: BDI, ISI, Unranked. Dacă se selectează tipul Conference sau Workshop, mai apare un câmp numit Category cu patru variante: A, B, C, D. Dacă se selectează tip Workshop sau se selectează Journal, alături de Index ISI apare încă un câmp Impact Factor care are un input în care se poate introduce un număr float.

Pentru salvare modificărilor se va apăsa pe butonul Save publication, altfel modificările se vor anula pe butonul Cancel.

După ce a fost editată o publicație aceasta va avea tagul Edited.

Edited ✓

**Proof-carrying parameters in certified symbolic execution**  
 Authors: A Arusoaie, D Lucanu  
 Year: 2024  
 Type: conference  
 Indexed: BDI  
 Category: B  
 Source: scholar

Edit

## 6.5.8 Publicații asemănătoare

Dacă s-au detectat două publicații asemănătoare, adică în backend s-a considerat că o publicație de pe DBLP și una de pe Scholar sunt una și aceeași. Sunt afișate amândouă și utilizatorul trebuie să aleagă dintre "Yes, this is a match" și "No, separate publications". Dacă se alege "Yes, this is a match", mai apar două butoane: "Keep DBLP" și "Keep Scholar", dintre care utilizatorul poate alege una, în funcție de răspuns. Dacă se alege separate, două publicații apar în locul celei care era înainte.

New

**Decentralized Application for Rating Internet Resources**  
 Authors: Andreea Buterchi, Andrei Arusoaie  
 Year: 2021  
 Source: DBLP

**Decentralized Application for Rating Internet Resources.**  
 Authors: A Buterchi, A Arusoaie  
 Year: 2021  
 Source: Scholar

Yes, this is a match

No, separate publications

New

**Decentralized Application for Rating Internet Resources**  
 Authors: Andreea Buterchi, Andrei Arusoaie  
 Year: 2021  
 Source: DBLP

**Decentralized Application for Rating Internet Resources.**  
 Authors: A Buterchi, A Arusoaie  
 Year: 2021  
 Source: Scholar

Keep DBLP

Keep Scholar

După ce o asemănare a fost confirmată aceasta va avea tag-ul Confirmed Match.

**Identifying Vulnerabilities in Smart Contracts using Interval Analysis**  
 Authors: Stefan-Claudiu Susan, Andrei Arusoaie  
 Year: 2023  
 Source: dblp

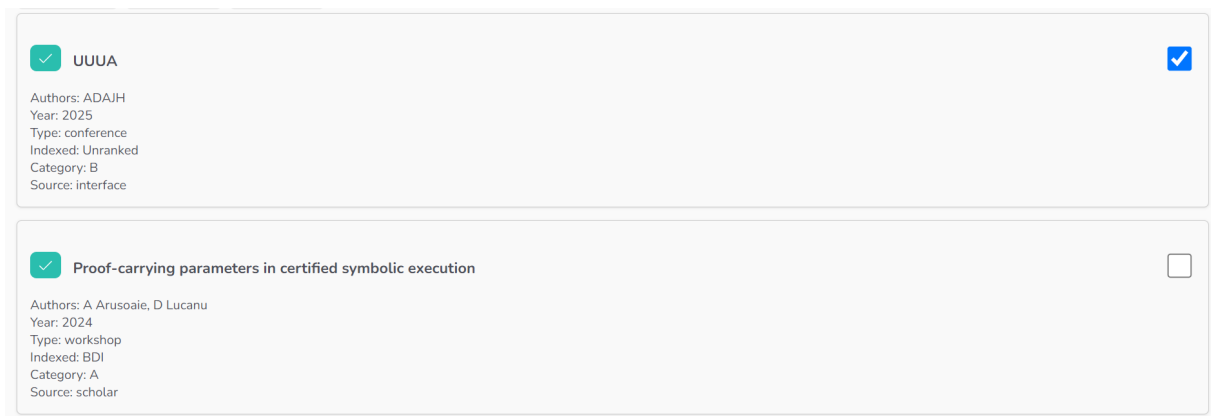
Edit

### 6.5.9 See citations button

La publicațiile care sunt de pe Scholar apare un buton "See citations" care te duce la o pagină cu toate citările acelei publicații.

## 6.6 Generarea Rapoartelor: Create publications report

Dacă se selectează din meniul de sus *Create publications report*, va fi afișată o listă doar cu publicațiile care au fost editate și salvate de către utilizator. La fiecare publicație va apărea un checkbox care poate fi bifat sau debifat.



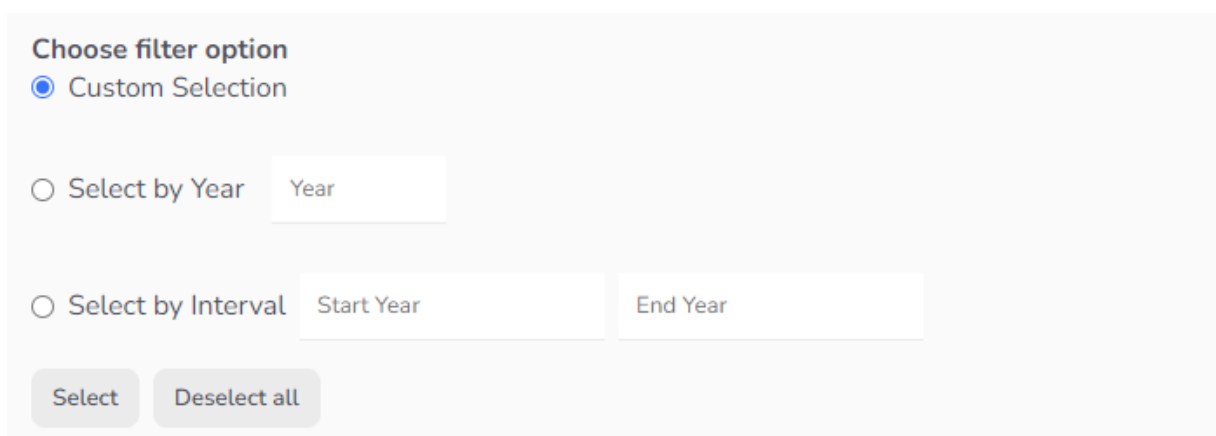
The screenshot displays a list of publications in a table-like format. Each row represents a publication and includes a green checkmark icon in a box on the left, the publication title, and a blue checkmark icon in a box on the right. Below the title, the following metadata is listed: Authors, Year, Type, Indexed status, Category, and Source.

Publication Title	Authors	Year	Type	Indexed	Category	Source
UUUA	ADAJH	2025	conference	Unranked	B	interface
Proof-carrying parameters in certified symbolic execution	A Arusoale, D Lucanu	2024	workshop	BDI	A	scholar

Deasupra tuturor publicațiilor editate se află un formular cu trei variante:

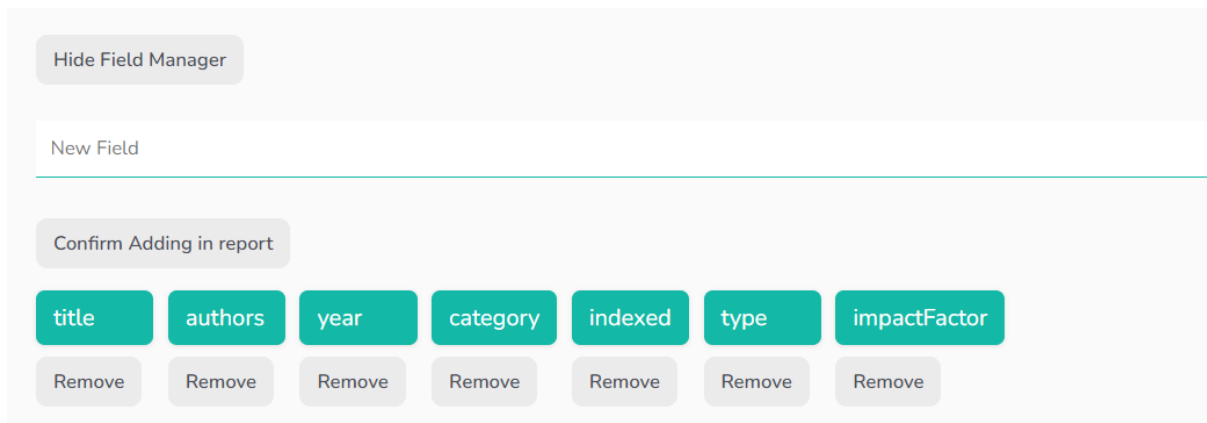
1. *Custom Selection*
2. *Select by Year* - există un câmp care trebuie completat cu un an
3. *Select by Interval* - există două câmpuri care trebuie completate cu un *start year* și celălalt cu *end year*

Acest formular este pentru selectarea datelor după filtrul selectat. După ce se selectează unul și se scriu anii, trebuie apăsat pe butonul *select*.



The screenshot shows a form titled "Choose filter option". It has three radio button options: "Custom Selection" (which is selected), "Select by Year", and "Select by Interval". Below "Select by Year" is a text input field labeled "Year". Below "Select by Interval" are two text input fields labeled "Start Year" and "End Year". At the bottom of the form are two buttons: "Select" and "Deselect all".

Există de asemenea posibilitatea ca acele câmpuri care vor apărea în rapoarte să fie selectate. Există deja unele by default care sunt afișate sub formularul de filtrare (*title, authors, year, conference, indexed, impactFactor*). Acestea sunt predefinite, dar fiecare are un buton de *Remove* lângă el care poate scoate câmpul corespunzător.

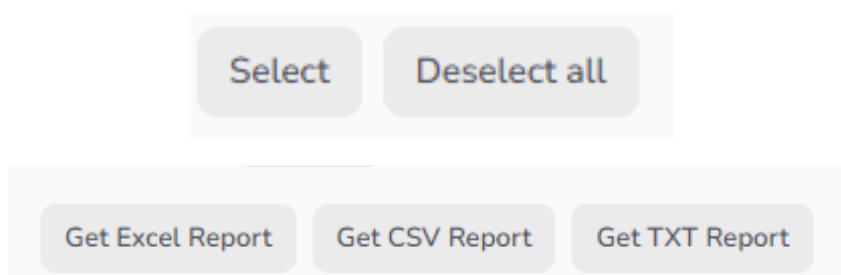


Există de asemenea un input cu buton *Add field* lângă. Dacă este scris un câmp în acel input și apoi este apăsat butonul de *Add field*, acest nou câmp va apărea în rapoarte cu toate valorile corespunzătoare.

În momentul în care se va genera raportul în backend, se salvează și preferința utilizatorului pentru câmpurile raportului și vor apărea exact aceleași data viitoare. Nu va mai fi nevoie să fie introdus sau șters.

Există 5 butoane:

1. *Select* - care trebuie apăsat după alegerea filtrului
2. *Deselect* - deselectează tot ce e selectat
3. *Get Excel Report*
4. *Get CSV Report*
5. *Get TXT Report*



După ce se apasă 3, 4, 5, se va genera un document în formatul specificat cu toate publicațiile selectate. Acesta se va descărca instant pe calculator.



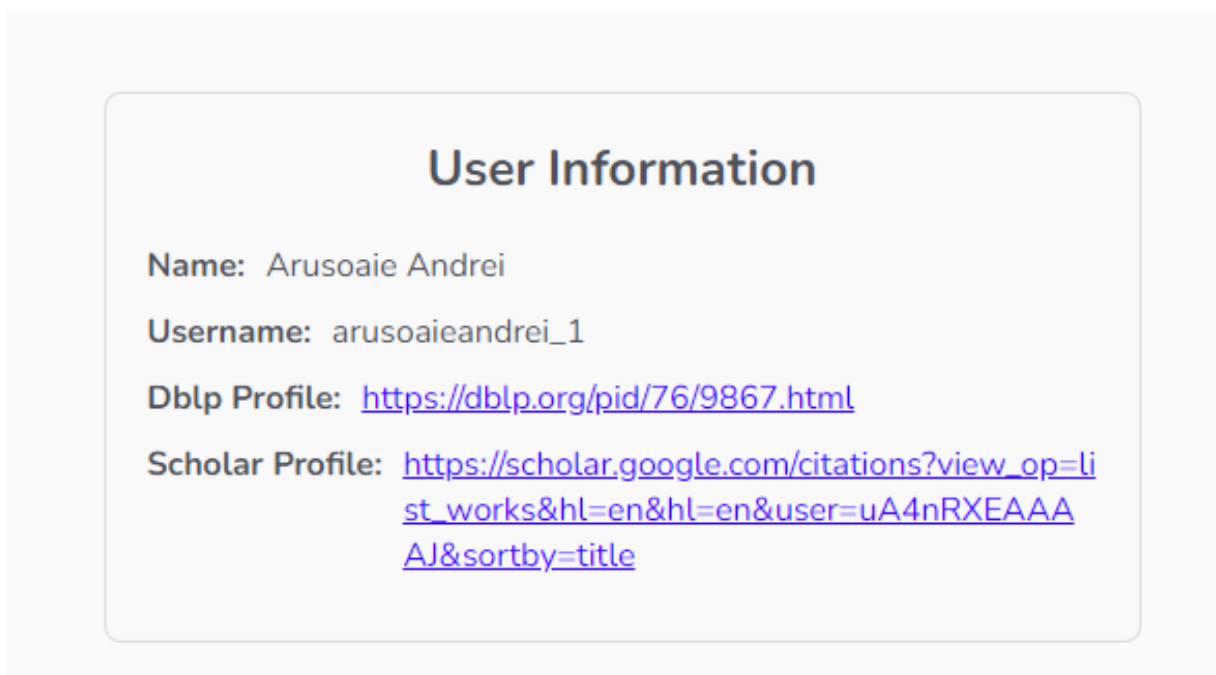
## 6.7 Publicațiile cu citări: My Citations

Dacă este selectat My citations, atunci va apărea o listă cu titlurile tuturor publicațiilor care au citări cu un buton See citations în dreptul fiecăruia. Acele butoane te vor duce la pagina de citări ale unei publicații.



## 6.8 Profilul utilizatorului: My Profile

Dacă este selectat din meniul de sus My profile, vor apărea informațiile utilizatorului activ:

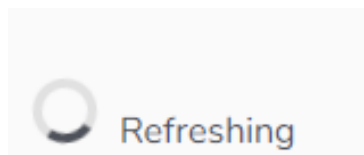


## 6.9 Profilul utilizatorului: Log out

Dacă se selectează Logout, utilizatorul va fi delogat și se va șterge token-ul din localStorage.

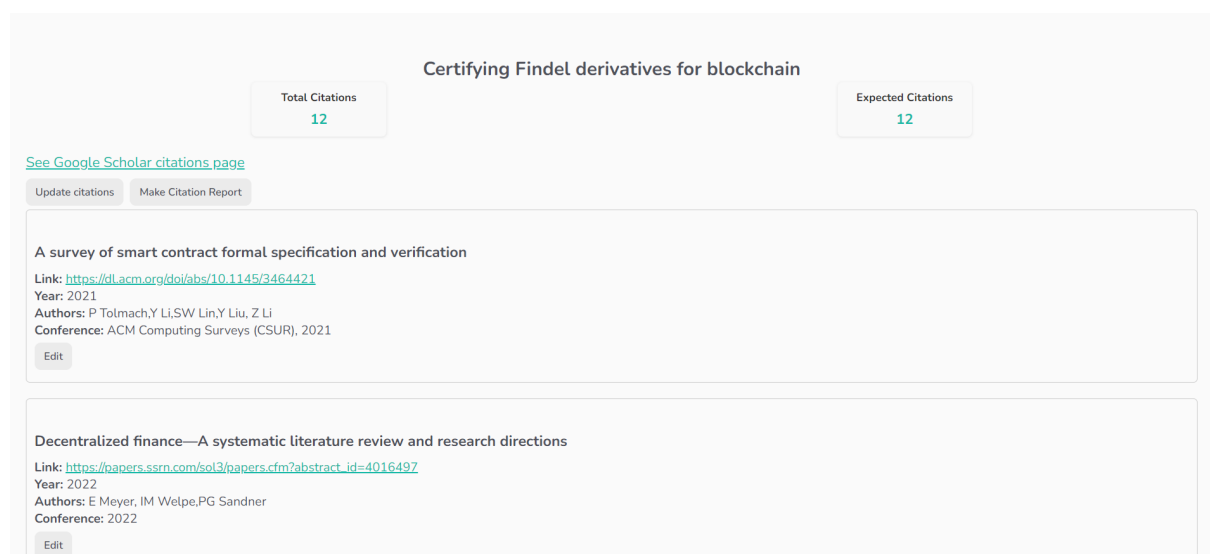
## 6.10 Vizualizarea grafică a unei acțiuni în curs

Atunci când utilizatorul a făcut o acțiune și cererea pentru aceasta a fost trimisă și în momentul actual se așteaptă un răspuns se afișează o roțiță care se învâрте alături de un mesaj care spune ce se întâmplă:



## 6.11 Pagina de Citări ale unei Publicații

Prin apăsarea butonului *See citations* din dreptul unei publicații se va ajunge la pagina de citări ale unei publicații. Inițial, apar citările deja existente în baza de date. Utilizatorul poate apăsa *Update citations* pentru a actualiza datele. Există un link "*See scholar citations page*" către pagina Scholar cu toate citările.



Apoi, apare lista de citări. Fiecare citare are câmpurile completate: *Title*, *Author*, *Year*, *Conference* și *Link*. Fiecare citare are un buton *edit*. După ce se apasă butonul *edit*, apare un formular cu două câmpuri necomplete: *Index* și *Impact Factor*, care au aceleași valori descrise mai sus la publicații.

Utilizatorul poate edita și câmpurile deja completate. După ce totul e completat și editat, utilizatorul poate apăsa *Save citation* sau *Cancel*.

### Edit Citation

Title	Decentralized finance—A systematic literature review and research directions
Authors	E Meyer, IM Welp,PG Sandner
Conference	2022
Year	2022
Index	Select Index ▼
Impact Factor	Impact Factor

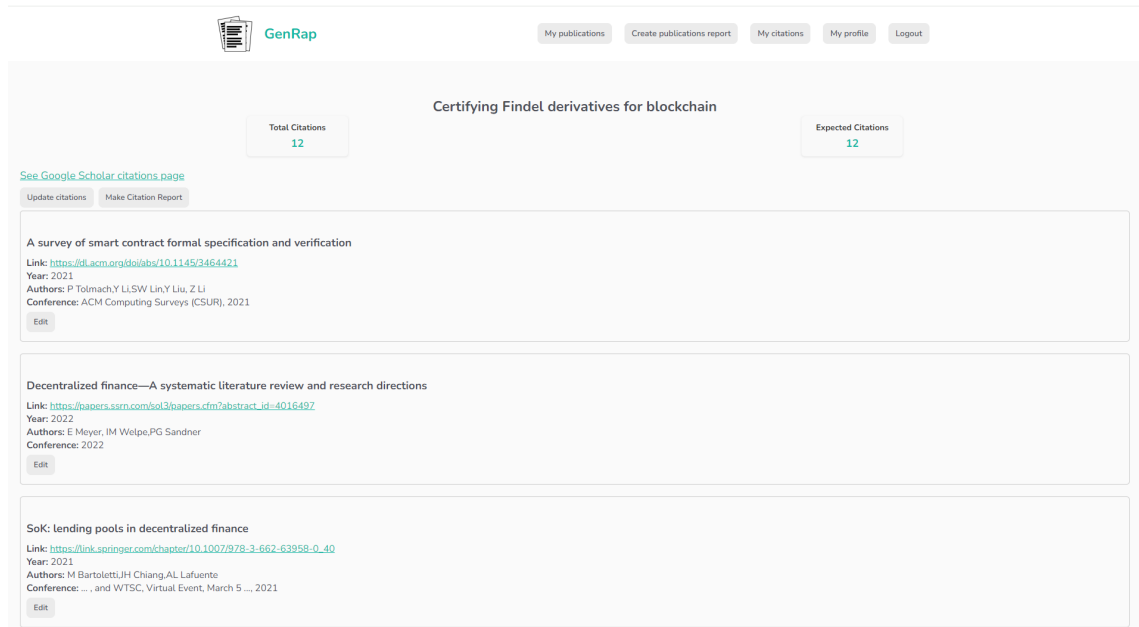
Save citation

Cancel

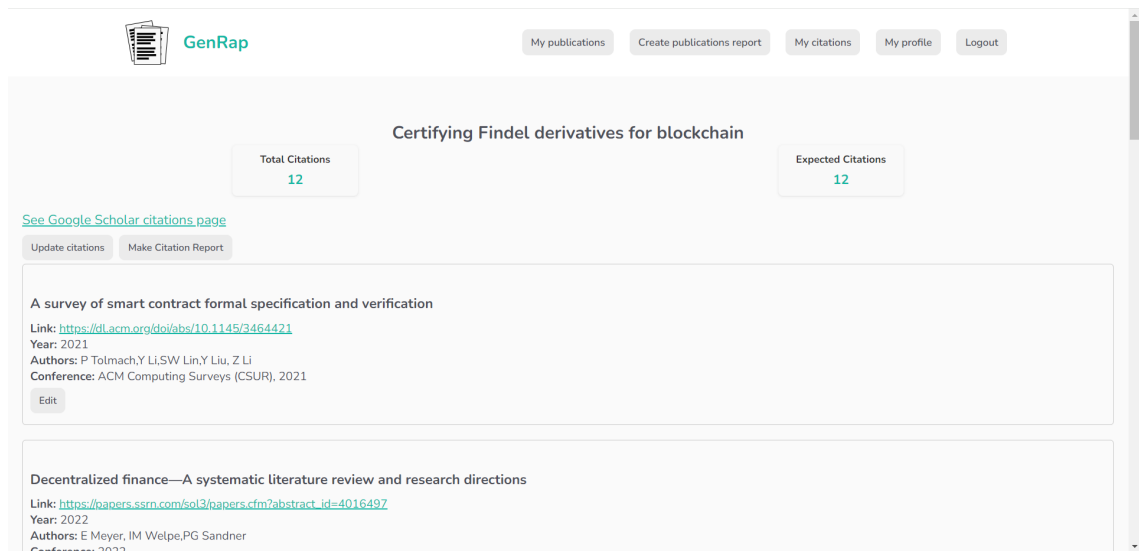
## 6.12 Înfățișare adaptabilă (Reactive)

Aplicația este adaptabilă, oferind o experiență optimă pe ecrane de dimensiuni variate.


- Ecran mare:



- Ecran mediu:



- Ecran mic:



My publications

Create publications report

My citations

My profile

Logout

## Certifying Findel derivatives for blockchain

Total Citations  
**12**

Expected Citations  
**12**

[See Google Scholar citations page](#)

Update citations

Make Citation Report

**A survey of smart contract formal specification and verification**

Link: <https://dl.acm.org/doi/abs/10.1145/3464421>

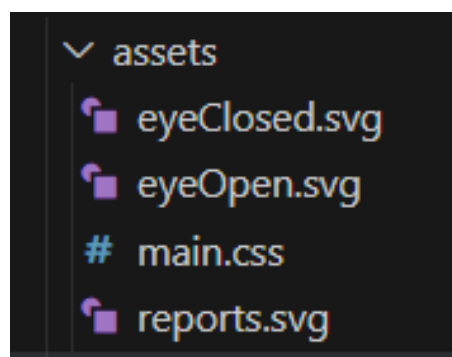
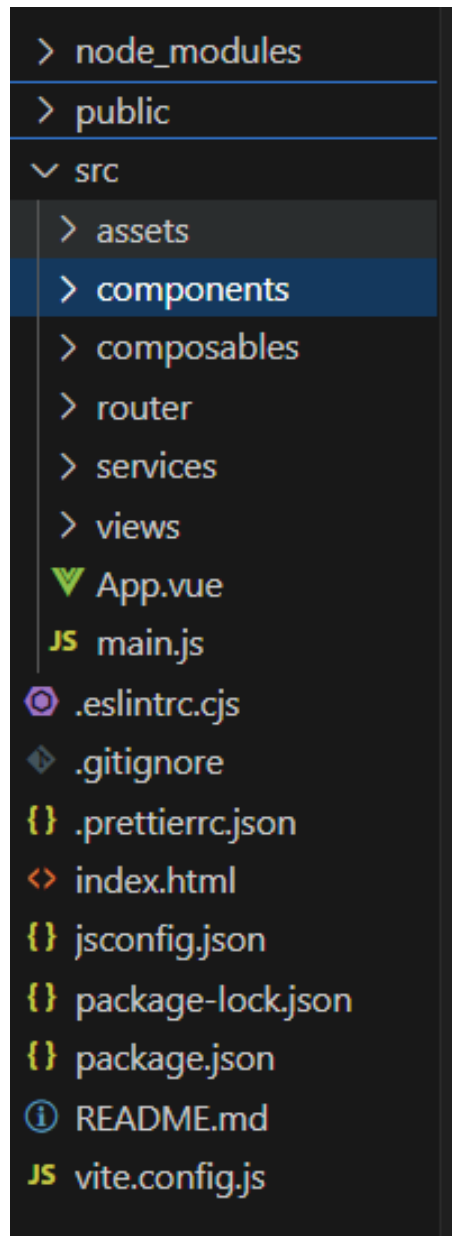
Year: 2021

Authors: P Tolmach, Y Li, SW Lin, Y Liu, Z Li

Conference: ACM Computing Surveys (CSUR), 2021

Edit

## 6.13 Structura codului



```

  components
  Citations
    CitationItem.vue
    EditCitation.vue
  Publications
    AddField.vue
    EditPublication.vue
    PublicationBasic.vue
    PublicationMatch.vue
    PublicationsItem.vue
    PublicationsList.vue
  NavBarComp.vue

```

```

  composables
    JS getCitations.js
    JS getUser.js
    JS tokenManagement...
    JS useLogin.js
    JS useLogout.js
    JS usePublications.js
    JS useSignup.js

```

```

  router
    JS index.js
  services
    JS apiService.js

```

```

  views
  |
  |   auth
  |   |   LoginView.vue
  |   |   SignupView.vue
  |   |
  |   reports
  |   |   CitationsView.vue
  |   |   CreateReport.vue
  |   |   PublicationsView.vue
  |   |   HomeView.vue
  |   |   PubCitationsView.vue
  |   |   UserProfile.vue
  |   App.vue
  |   JS main.js

```



# Capitolul 7

## Ghid de Configurare

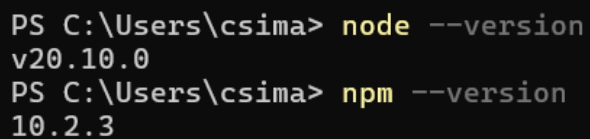
### 7.1 Configurarea Aplicației Vue.js

#### 7.1.1 Precondiții

- Instalați Node.js pe sistemul dumneavoastră de pe <https://nodejs.org/>. După ce urmați procesul de instalare, puteți verifica dacă instalarea a fost făcută cu succes astfel:

```
node --version  
npm ---version
```

- Acestea sunt versiunile cu care a fost făcut proiectul:



```
PS C:\Users\csima> node --version  
v20.10.0  
PS C:\Users\csima> npm --version  
10.2.3
```

#### 7.1.2 Pași

1. Clonați repository-ul proiectului:

```
https://github.com/clarasima/GenRap.git
```

2. Accesați directorul interfeței.

```
cd GenRap
cd interfata
```

### 3. Instalați dependențele:

```
npm install
```

### 4. Porniți aplicația:

```
npm run dev
```

5. Deschideți browserul și navigați la url-ul specificat (`http://localhost:abcd`) în linia de comandă pentru a vedea aplicația funcționând.

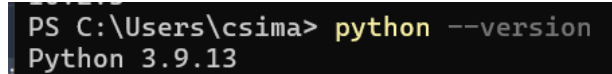
## 7.2 Configurarea API-ului

### 7.2.1 Precondiții

- Instalați Python de pe <https://www.python.org/>. Puteți verifica dacă instalarea a fost făcută cu succes astfel:

```
python --version
```

- În caz că o librărie este depreciată ("deprecated") pentru versiunea ta curentă de python, terminalul te va anunța în momentul în care vei încerca să o instalezi sau să o rulezi și îți va da alte opțiuni. Urmează instrucțiunile date.
- Aceasta este versiunea de python cu care a fost făcut proiectul:



```
PS C:\Users\csima> python --version
Python 3.9.13
```

## 7.2.2 Pași

1. Clonați repository-ul proiectului:

```
https://github.com/clarasima/GenRap.git
```

2. Accesați directorul API-ului.

```
cd GenRap  
cd api
```

3. Instalați dependențele:

```
pip install -r requirements.txt
```

4. Setați variabilele de mediu: Pentru Windows:

```
$env:SCRAPER_API_KEY="cheie_scraper_api"  
$env:DB_URI="mongodb_uri"  
$env:SECRET_KEY="secret_key_for_generating_JWT_tokens"
```

5. Rulați aplicația Flask:

```
python run.py
```

6. API-ul va fi accesibil la `http://localhost:5000`.

## 7.3 Testarea API-ului

### 7.3.1 Testarea în Terminal

Pentru a verifica dacă variabila de mediu este setată corect, utilizați următoarea comandă:

```
Get-Item Env:DB_URI  
Get-Item Env:SCRAPER_API_KEY  
Get-Item Env:SECRET_KEY
```

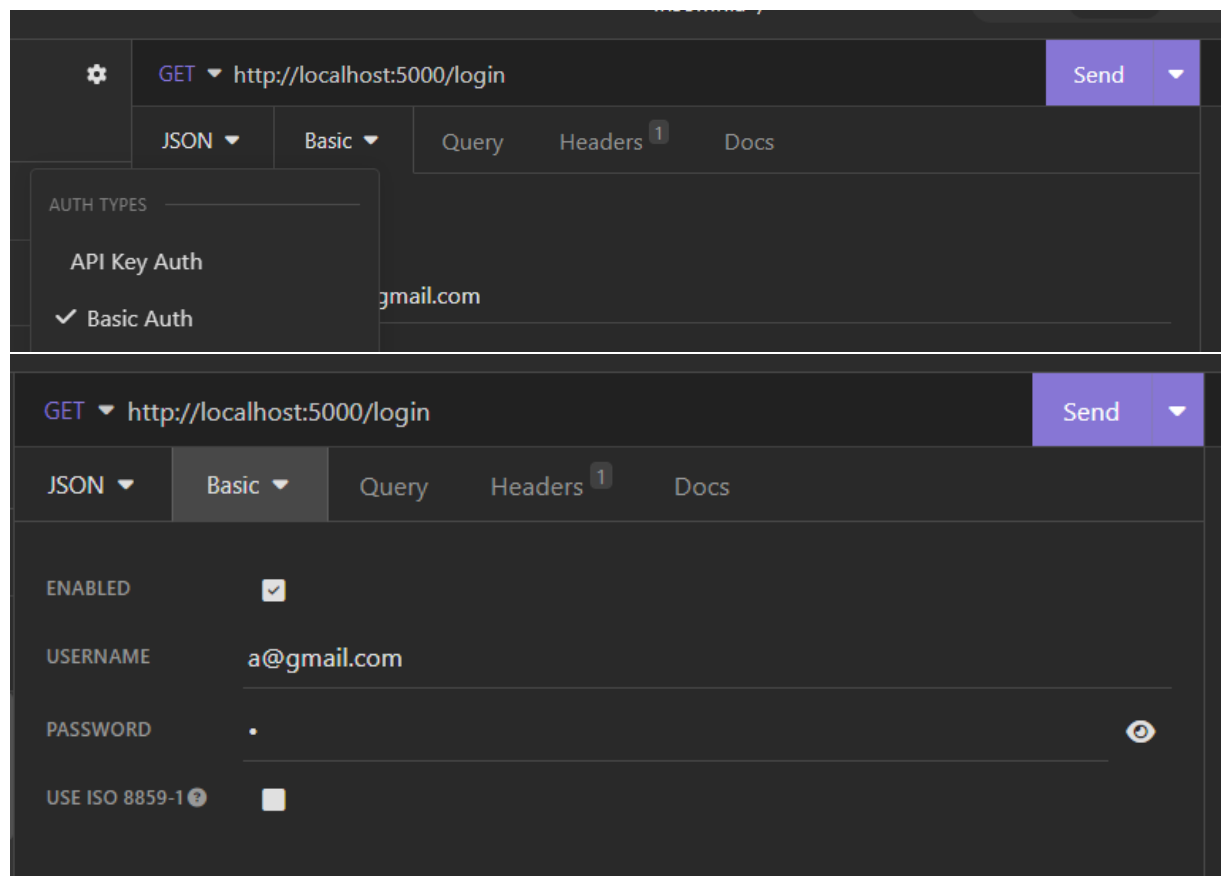
### 7.3.2 Testarea cu Insomnia

Insomnia este un client REST API care permite testarea API-urilor. Pentru a testa API-ul proiectului cu Insomnia, urmați pașii de mai jos:

1. Descărcați și instalați Insomnia de la <https://insomnia.rest/download>.
2. Deschideți Insomnia și creați un nou Workspace.
3. Adăugați o nouă cerere (Request):
  - (a) Selectați tipul cererii (GET, POST, PUT, DELETE).
  - (b) Introduceți URL-ul cererii (de exemplu, `http://localhost:5000/login`).
  - (c) Dacă cererea necesită autentificare, adăugați header-ul 'x-access-token' cu token-ul de acces.
4. Pentru a testa autentificarea, creați o cerere GET pentru endpoint-ul de login și selectați Basic Auth, apoi completați username și password:

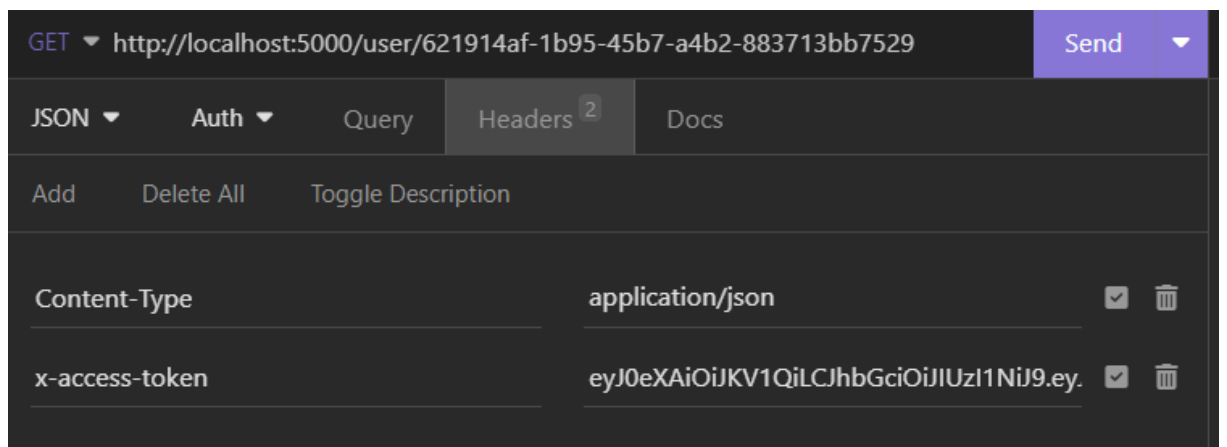
URL: `http://localhost:5000/login`

Method: GET



5. Trimiteți cererea și verificați răspunsul. Ar trebui să primiți un token de autentificare dacă datele de autentificare sunt corecte.
6. Utilizați token-ul primit pentru a testa celelalte endpoint-uri. De exemplu, pentru a obține informațiile despre un utilizator, creați o cerere GET:

```
URL: http://localhost:5000/user/{user_id}
Method: GET
Headers:
    x-access-token: <token>
```



7. Repetați procesul pentru celelalte endpoint-uri ale API-ului, utilizând metodele corespunzătoare și anteturile de autentificare necesare.

## Capitolul 8

# Analiză Comparativă cu alte Proiecte Similare

În lumea academică, administrarea eficientă a publicațiilor și citărilor este fundamentală. Există diverse instrumente disponibile pentru acest scop, fiecare având propriile beneficii și capacități. Printre cele mai cunoscute sunt Zotero, Google Scholar Profiles și Scopus. Scopul proiectului meu este să ofere o interfață prietenoasă, care să se diferențieze de celelalte programe existente prin diverse modalități, aducând valoare suplimentară.

### 8.1 Zotero

Zotero ajută la colectarea, administrarea și organizarea publicațiilor, permițând de asemenea utilizatorilor să citeze. Cu toate că Zotero se evidențiază prin modul său de gestionare a publicațiilor, acesta nu oferă posibilitatea de extragere automată a acestora sau a citărilor corespondente, lucru pe care aplicația îl oferă. În Zotero, oamenii trebuie să introducă manual fiecare citare și publicație, având mai târziu posibilitatea să utilizeze acele date adunate. Proiectul meu se evidențiază prin colectarea datelor din diverse website-uri, precum Google Scholar și DBLP, automatizând procesul de extragere a datelor și actualizându-le cu ușurință.

## 8.2 Google Scholar

Google Scholar pune la dispoziție cercetătorilor un loc în care își pot vedea lista de publicații și pot să urmărească citările corespunzătoare acestora. Baza lor de date se actualizează automat. Totuși, așa cum am menționat și la începutul lucrării, pe Google Scholar nu apar toate publicațiile cercetătorilor, având unele lipsuri. De exemplu, urmărind publicațiile de pe DBLP, se pot găsi publicații care lipsesc pe Google Scholar.

Așadar, proiectul meu își lărgeste baza de date preluând informații și de pe DBLP. În cadrul proiectului, se pot face editări manuale și se pot adăuga noi câmpuri. Astfel, utilizatorii au un control mai mare asupra datelor și o flexibilitate crescută în gestionarea acestora.

## 8.3 Scopus

Scopus are o bază de date amplă, cuprinzând un număr foarte mare de jurnale, conferințe și lucrări academice. De asemenea, oferă detalii despre citări și lucrări academice. Totuși, accesul la Scopus este limitat, fiind necesar un cont creat prin instituții acreditate, ceea ce face ca puține persoane să aibă acces la toate datele disponibile.

Proiectul meu vine cu îmbunătățiri față de Scopus. Diferențele principale dintre proiectul meu și Scopus sunt următoarele:

- Posibilitatea de a extrage automat publicațiile și citările din mai multe surse precum Google Scholar și DBLP
- Posibilitatea de a administra și edita manual informațiile existente.
- Posibilitatea de adăugare a unor câmpuri manual
- Adăugarea automată a câmpurilor: Type, Category, Index și Impact Factor
- Generarea de rapoarte detaliate

## **8.4 Funcționalități adiționale ale aplicației propuse**

Aplicația mea combină punctele forte ale programelor deja existente, dar aduce și îmbunătățiri prin reducerea limitărilor acestora. Ceea ce este adus în plus aplicației descrisă este posibilitatea de a extrage automat informațiile din mai multe surse și de a crea rapoarte personalizate pentru fiecare cercetător. Aceste îmbunătățiri ajută profesorii să își gestioneze publicațiile și citările și să își creeze rapoartele cu ușurință, economisind astfel timp valoros.



# Capitolul 9

## Concluzii

### 9.1 Planuri de îmbunătățire

Făcând această aplicație am început să am și diferite idei de cum aș putea să o îmbunătățesc, următoarele sunt câteva dintre ideile mele:

- Am observat că <https://www.sciencedirect.com/> are o bază de date cu foarte multe publicații. Un plan măreț ar fi să fie extrase toate datele de acolo și folosite pentru această aplicație.
- Ar mai fi o îmbunătățire dacă s-ar mai putea detecta automat unele câmpuri precum indexul.
- Publicațiile ar putea fi filtrate după mai multe câmpuri pentru a genera rapoartele, nu doar manual sau după an. De exemplu s-ar putea genera un raport cu toate publicațiile indexate ISI.
- Utilizatorul ar putea avea acces la toate rapoartele generate în trecut când își accesează profilul.
- O coadă cu lista tuturor request-urilor ar fi o îmbunătățire. Aceasta ar putea ajuta la identificarea unor posibile bug-uri.
- Folosirea ScraperApi și pentru extragerea publicațiilor pentru evitarea posibilității de a fi blocate requesturile din cauza excesului. Acest lucru ajută la scalabilitatea aplicației, dar implică mai multe costuri.
- Completarea principiului CRUD cu partea de DELETE.

## 9.2 Concluzie finală

În această lucrare am adresat problema profesorilor care investesc mult timp în crearea rapoartelor periodice cu publicațiile și citările academice, deoarece trebuie să analizeze și să extragă informații din mai multe surse. Soluția a fost să creez o aplicație web care automatizează extragerea informațiilor de pe site-urile DBLP și Google Scholar, oferind posibilitatea profesorilor să editeze informațiile. Apoi, aceștia au posibilitatea de a genera rapoarte în 3 formate diferite (Excel, CSV și Text), alegând manual unele publicații și citări pentru rapoarte sau folosind filtre bazate pe intervalul temporal în care acestea au apărut, putând de asemenea să modifice câmpurile care apar în rapoarte.

# Bibliografie

- <https://www.mongodb.com/> - MongoDB: Documentația oficială MongoDB.
- <https://flask.palletsprojects.com/> - Flask: Documentația oficială Flask.
- <https://scraperapi.com/> - ScraperAPI: Un serviciu pentru extragerea datelor din paginile web.
- <https://scholar.google.com/> - Google Scholar: Motor de căutare pentru publicații academice.
- <https://dblp.org/>: DBLP - Baza de date digitală pentru publicații în domeniul informaticii.
- [dbdiagram.io](https://dbdiagram.io) - Un instrument pentru crearea diagramelor bazei de date.
- <https://app.diagrams.net/> Draw.io: Un instrument pentru crearea diagramelor și schemelor logice.
- <https://phosphoricons.com/> SVG Icons: Utilizat pt ochii de langa parole
- <https://tailwindcss.com/docs/customizing-colors> TailWindsCss: utilizat pentru culori
- <https://udemy.com> Build Web Apps with Vue JS 3 and Firebase: Curs Udemy despre web aplicatii in VueJs
- <https://github.com/iamshaunjp/Vue-3-Firebase> Github files curs Build Web Apps with Vue JS 3 and Firebase - Am folosit resurse de aici pentru css si organizare interfata.
- <https://www.youtube.com/watch?v=J5bIPtEbS0Q> Explicatii despre autentificarea unui Flask API cu JWT