



UNIVERSITATEA "ALEXANDRU IOAN CUZA" DIN IAȘI

**FACULTATEA DE INFORMATICĂ**



LUCRARE DE LICENȚĂ

**Biblioteca Uaic**

propusă de

***Țifui Vali Andrei***

**Sesiunea:** *iulie, 2017*

**Coordonator științific**

**Lector Dr. Alex Moruz**

**UNIVERSITATEA “ALEXANDRU IOAN CUZA” DIN IAȘI**

**FACULTATEA DE INFORMATICĂ**

# **Biblioteca UAIC**

***Țifui Vali-Andrei***

**Sesiunea:** *iulie, 2017*

**Coordonator științific**

***Lector Dr. Alex Moruz***

## DECLARAȚIE PRIVIND ORIGINALITATE ȘI RESPECTAREA DREPTURILOR DE AUTOR

Prin prezenta declar că Lucrarea de licență cu titlul „*Biblioteca UAIC*” este scrisă de mine și nu a mai fost prezentată niciodată la o altă facultate sau instituție de învățământ superior din țară sau din străinătate. De asemenea, declar că toate sursele utilizate, inclusiv cele preluate de pe Internet, sunt indicate în lucrare, cu respectarea regulilor de evitare a plagiatului:

- toate fragmentele de text reproduse exact, chiar și în traducere proprie din altă limbă, sunt scrise între ghilimele și dețin referința precisă a sursei;
- reformularea în cuvinte proprii a textelor scrise de către alți autori deține referința precisă;
- codul sursă, imaginile etc. preluate din proiecte *open-source* sau alte surse sunt utilizate cu respectarea drepturilor de autor și dețin referințe precise;
- rezumarea ideilor altor autori precizează referința precisă la textul original.

Iași,

Absolvent *Prenume Nume*

---

(semnătura în original)

## DECLARAȚIE DE CONSIMȚĂMÂNT

Prin prezenta declar că sunt de acord ca Lucrarea de licență cu titlul „*Titlul complet al lucrării*”, 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.

Iași,

*Absolvent Prenume Nume*

---

(semnătura în original)

## ACORD PRIVIND PROPRIETATEA DREPTULUI DE AUTOR

Facultatea de Informatică este de acord ca drepturile de autor asupra programelor-calculator, în format executabil și sursă, să aparțină autorului prezentei lucrări, *Vali-Andrei Țifui*

Încheierea acestui acord este necesară din următoarele motive:

Iași,

Decan *Prenume Nume*

---

(semnătura în original)

Absolvent *Prenume Nume*

---

(semnătura în original)

# Cuprins

Capitolul I .....	<b>Error! Bookmark not defined.</b>
Introducere.....	8
Motivație .....	9
Capitolul III: Dezvoltarea Aplicației.....	13
III.1 Tehnologii Utilizate .....	<b>Error! Bookmark not defined.</b>
III.1.1 Arhitectura .NET .....	13
III.1.2 Tehnologia .Net Core .....	14
III.1.1.3 Limbajul C# .....	18
III.1.2 Platforma Angular 2.....	19
III.2 Arhitectură și implementare .....	<b>Error! Bookmark not defined.</b>
III.2.1 Aplicația client web SPA (Angular) .....	24
III.2.2 Server-ul Web API.....	30
III.2.3 Arhitectura bazei de date .....	41
Capitolul IV: Ghid de utilizare .....	42
Încărcarea Aplicației.....	42
Autentificarea utilizatorilor .....	42
Înregistrarea utilizatorilor .....	43
Meniul aplicației.....	44
Schimbarea limbii Aplicației .....	44
Pagina de prezentare a celor mai recente cărți .....	45
Afișarea și utilizarea unei cărți .....	45
Utilizarea și afisarea comentariilor .....	46
Filtrarea și căutarea cărților .....	46
Afișarea și editarea profilului utilizatorului.....	46
Vizualizarea cărților .....	47
Administrarea Bibliotecii .....	48

# Introducere

Nevoia omului de a avea parte la educație a existat încă din timpul antichității. În vremurile îndepărtate în lipsa existenței scrisului, învățăturile erau transmise prin viu grai de la o generație la alta, multe dintre aceste învățături fiind transmise eronat sau puteau fi uitate. Odată cu apariția scrisului educația a căpătat un alt sens. Acest înțeles al educației este prezent și în perioada istorică actuală. Astăzi fără lipsa scrisului nu am putea vorbi despre învățătură , învățământ , profesor și alți termeni aflați în strânsă legătură cu educația.

Știm cu toții că educația stă la baza evoluției rasei umane, dar totodată cunoaștem că fără existența cărților educația nu ar putea fi realizabilă. De-a lungul istoriei au fost scrise milioane de cărți în întreaga lume, în diverse domenii, științifice sau neștiințifice. Toate aceste cărți au schimbat lumea în care trăim din epoca bronzului în epoca contemporană, după lumi total diferite. Cărțile au proprietatea unor memorii nevolatice, astfel că cunoștințele pe care strămoșii noștri le aveau acum o mie de ani au putut fi citite, învățate și folosite de către populația actuală. Această caracteristică realizează cel mai important proces al omenirii, evoluția. Așa cum în lipsa educației nu putem vorbi despre evoluție și civilizație , iar în lipsa cărților educația nu ar fi fost posibilă fără tehnicile actuale, deducem că cărțile au constituit singura opțiune de dezvoltare a omenirii.

Chiar și astăzi după mii de ani de la apariția primei cărți, principalul obiect utilizat în învățământ este cartea , fie că vorbim despre o carte electronică sau o carte în format fizic, după cum spune celebrul autor Barbara Tuchman , “Cărțile sunt cărașii civilizației. Fără cărți, istoria e mută, literatura nu are glas, știința paralizată, iar gândirea și meditația suspendate”.

Deoarece numărul cărților a fost și este în continuare în continuă creștere a fost nevoie de adoptarea unor instituții care să centralizeze și să facă managementul acestor cărți. Aceste instituții poartă și astăzi numele de bibliotecă. Pentru a putea ajunge mai rapid la o carte, până acum era obligatoriu să trecem pragul unei biblioteci, de unde cartea poate fi cumpărată sau împrumutată.

Astăzi dezvoltarea tehnologică atinge culmi pe care nici un alt domeniu nu le-a atins vreodată. Trecem de la o eră în care tehnologia este abia la început la o eră complet tehnologizată. În zilele noastre când suntem în căutarea unei informații prima susă este internetul, apoi celelalte surse alternative cum ar fi o carte. Deoarece utilizarea unei biblioteci online crește gradul de comoditate și confort în comparație cu vizitarea unei biblioteci tradiționale , se observă o creștere tot mai mare a numărului de biblioteci online. În cadrul unei biblioteci online cărțile pot fi căutate aproape instant specificând nenumărate filtre de căutare , autori, anul publicării, categorii chiar și după anumite informații din acea carte. Pentru a citi o carte dintr-o astfel de bibliotecă nu mai este nevoie să ne deplasăm pentru a împrumuta sau cumpăra acea carte, fiind



necesar doar descărcarea acestora pe propriul dispozitiv de unde putem începe lectura. Dacă până acum câțiva ani telefonul mobil nu avea access la internet, astăzi acesta este mult mai performant decât majoritatea calculatoarelor personale din acea perioadă.

Evoluția telefoanelor mobile spre ceea ce astăzi numim “smartphone”, aduce un avantaj și mai mare în utilizarea bibliotecilor online. Dacă în trecut cărțile în format electronic puteau fi descărcate și citite doar pe calculatorul personal, acesta fiind nepортabil, acum cărțile în format electronic pot fi descărcate și citite de oriunde atâta timp cât avem la dispoziție un smartphone. Am decis să dezvolt această aplicație pentru universitate, deoarece în momentul de față nu există o astfel de platformă care să ajute studenții în cautarea și găsirea informațiilor. Ca și student al universității mi-aș fi dorit o astfel de platformă de-a lungul celor 3 ani de studiu care să ofere o centralitate a tuturor cărților. De asemenea în perioada studenției, comunicarea și socializarea sunt foarte importante și ne ajută să ne dezvoltăm. Din acest motiv aplicația de față oferă și o caracteristică social prin adăugarea comentariilor, a aprecierilor sau a deaprecierilor asupra unei cărți.

Analizând care ar fi cerințele pe care un posibil utilizator al acestei aplicații le-ar putea avea, biblioteca online este construită în așa fel încât să acopere majoritatea necesităților. În același timp o altă caracteristică foarte importantă ce se dorește de la o astfel de aplicație este eficiența și viteza de răspuns. Această caracteristică fiind asigurată prin analiza și utilizarea celor mai bune tehnologii, precum și gândirea unei arhitecturi ce eficientizează rularea aplicației.

## Motivație

Principalul motiv în dezvoltarea bibliotecii online UAIC este lipsa implementării unui astfel de sistem. Încă din primul an de studiu în cadrul acestei universități mi-aș fi dorit să existe un sistem centralizat online în care studenții pot căuta într-un timp foarte rapid o carte, pot împărtăși păreri în legătură cu acea carte și cel mai important lucru își pot crea propriile notițe pentru cartea respectivă. Pornind de la această dorință, în contextul finalizării ciclului de licență, am decis ca această idee să devină tema propriei lucrări de licență. Un alt motiv ce privește alegerea dezvoltării unei aplicații web îl reprezintă pasiunea și acumularea de bune cunoștințe în acest domeniu.

# Capitolul I: Aplicații Similare

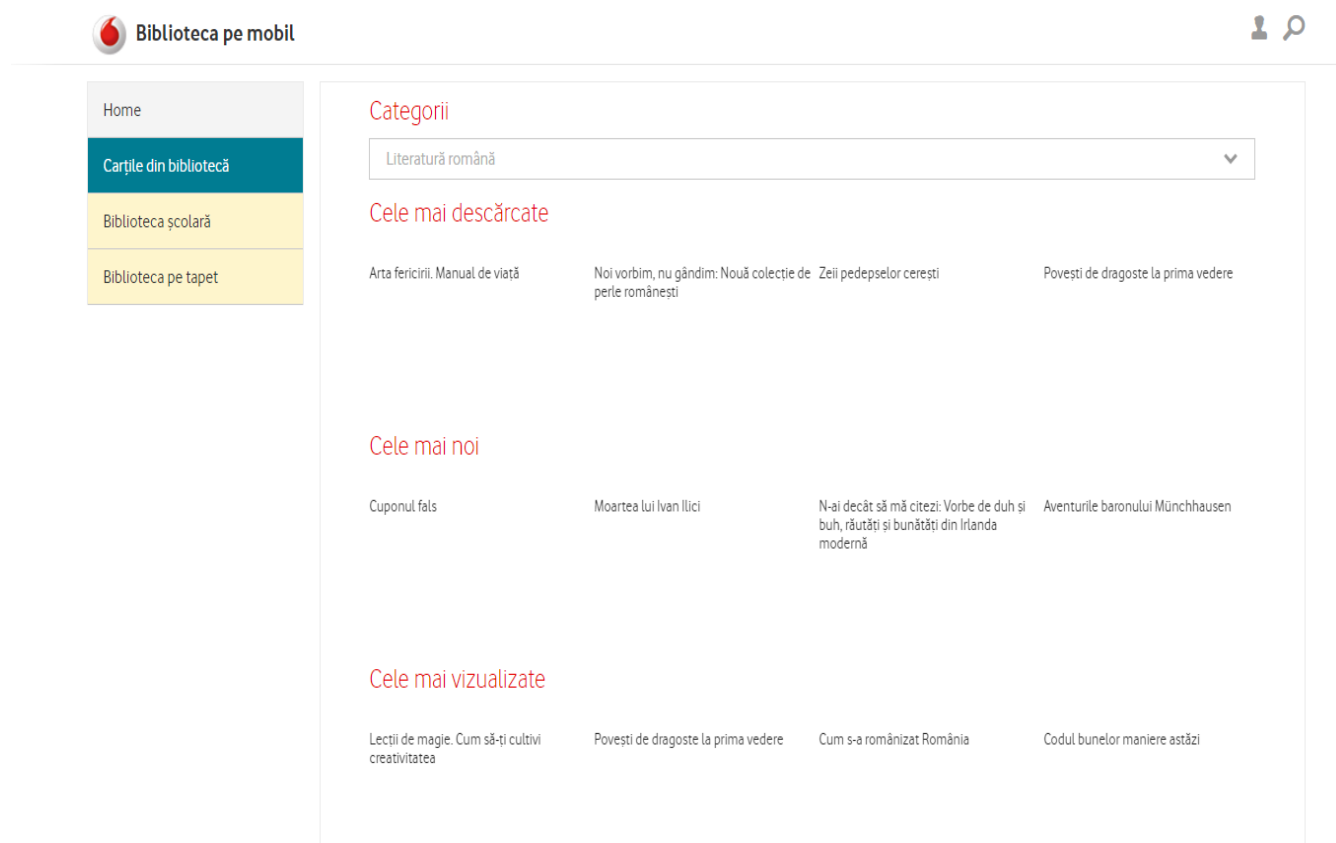
În secolul 21 internetul cunoaște o extindere uimitoare. În fiecare zi apar zeci aplicații și site-uri noi. Din acest motiv este foarte dificil să poți dezvolta o aplicație toată nouă, ideea acesteia ne mai fiind întâlnită vreodată. Deoarece bibliotecile există de sute de ani, ideea de bibliotecă online nu este ceva nou. Această idee reprezintă în cele mai multe cazuri o adaptare a bibliotecii clasice în mediu online.

Ca și surse de inspirație pentru biblioteca UAIC, putem enumera câteva implementări de biblioteci online pe care personal le-am accesat de câteva ori.

## I.1 Biblioteca scolară

Disponibilă la adresa: <https://scoala.bibliotecapemobil.ro/biblioteca.php>

Este un proiect dezvoltat pentru a veni în ajutorul școlarilor. Acest proiect a fost dezvoltat de către un operator de telefonie ce este prezent și în țara noastră. Pentru a putea utiliza această site, trebuie să fii înregistrat. Caracteristicile bibliotecii sunt simple. Există câteva categorii de cărți. În fiecare categorie se găsesc cărți relativ puține.



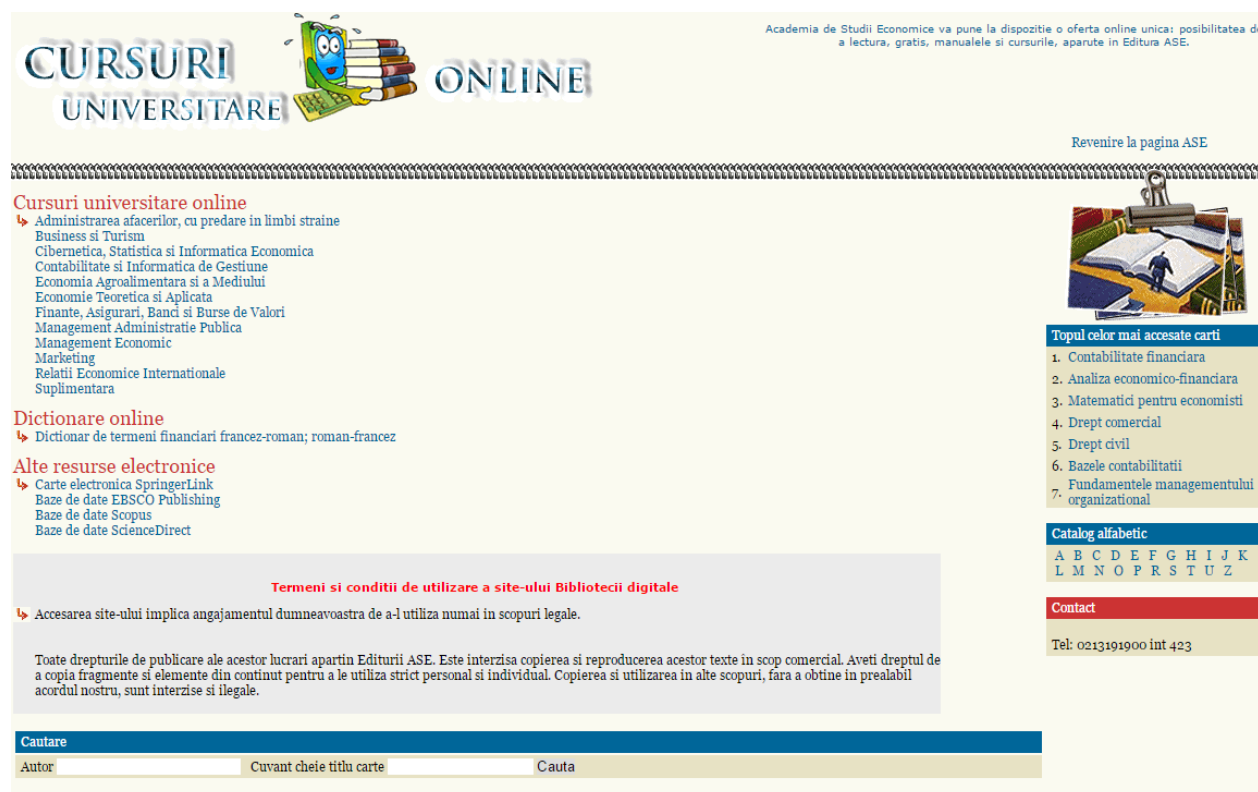
Această aplicație oferă o interfață prietenoasă și ușor de utilizat. A fost concepută pentru a rula în special pe telefoane mobile, acest lucru fiind specificat și în titlul aplicației "Biblioteca pe mobil".

Diferența dintre această aplicație și aplicația biblioteca UAIC este gradul de complexitate.

Biblioteca Uaic prezintă un mecanism mult mai bine pus la punct de filtrare și căutare a cărților, introduce ideea de socializarea asupra cărților precum și mecanism de anotare a unei cărți. Chiar dacă scopul aplicației Biblioteca scolară este asemenea cu cel al aplicației Biblioteca Uaic, scopul fiind acela de a ajuta la dezvoltarea tinerilor, modul de implementare și funcționalitățile sunt în proporție de 80% total diferite.

## I.2 Biblioteca digitală ASE

Ideea acestui site este foarte apropiată de scopul aplicației dezvoltate în cadrul acestei lucrări. Cele două împărtășesc ideea dezvoltării unei platforme destinate studenților pentru eficientizarea și face mult mai utilă utilizarea documentației și a cărților din cadrul universității. Cu toate acestea caracteristicile celor două aplicații sunt diferite începând de la prezentare până la utilizare. Ceea ce mi-aș fi dorit să găsesc la această aplicație ar fi o categorisire a cărților, o interfață mai prietenoasă și totodată un mecanism mai performant de filtrare a cărților, lucru pe care aplicația biblioteca UAIC îl realizează.



Academia de Studii Economice va pune la dispozitie o oferta online unica: posibilitatea de a lectura, gratis, manualele si cursurile, aparute in Editura ASE.

[Revenire la pagina ASE](#)

### Cursuri universitare online

- Administrarea afacerilor, cu predare in limbi straine
- Business si Turism
- Cibernetica, Statistica si Informatica Economica
- Contabilitate si Informatica de Gestiune
- Economia Agroalimentara si a Mediului
- Economie Teoretica si Aplicata
- Finante, Asigurari, Banci si Burse de Valori
- Management Administratie Publica
- Management Economic
- Marketing
- Relatii Economice Internationale
- Suplimentara

### Dictionare online

- Dictionar de termeni financiari francez-roman; roman-francez

### Alte resurse electronice

- Carte electronica SpringerLink
- Baze de date EBSCO Publishing
- Baze de date Scopus
- Baze de date ScienceDirect

### Topul celor mai accesate carti

1. Contabilitate financiara
2. Analiza economico-financiara
3. Matematici pentru economisti
4. Drept comercial
5. Drept civil
6. Bazele contabilitatii
7. Fundamentele managementului organizational

### Catalog alfabetic

A B C D E F G H I J K  
L M N O P R S T U Z

### Contact

Tel: 0213191900 int 423

**Termeni si conditii de utilizare a site-ului Bibliotecii digitale**

Accesarea site-ului implica angajamentul dumneavoastra de a-l utiliza numai in scopuri legale.

Toate drepturile de publicare ale acestor lucrari apartin Editurii ASE. Este interzisa copierea si reproducerea acestor texte in scop comercial. Aveti dreptul de a copia fragmente si elemente din continut pentru a le utiliza strict personal si individual. Copierea si utilizarea in alte scopuri, fara a obtine in prealabil acordul nostru, sunt interzise si ilegale.

**Cautare**

Autor  Cuvant cheie titlu carte  Cauta

Modul de afișare a cărșilor diferă de la un tip de de afișare tabelar la un tip de afișare listat. Bineînțeles ca și utilizator prefer modul de afișare listat, fiind și cel utilizat în cadrul aplicației dezvoltate, dar desigur, aceasta este doar o părere personală.

Asemenea aplicației prezentate anterior nici aceeașta din urmă nu atinge domeniul socializării, singurele operații ce pot fi făcute în cadrul acestei aplicații cautearea simplă a unei cărți și posibilitatea vizualizării unei cărți. Aceste acțiuni sunt suficiente dar nu îndeajuns pentru o bibliotecă online din perioada actuală.

### I.3 Facebook

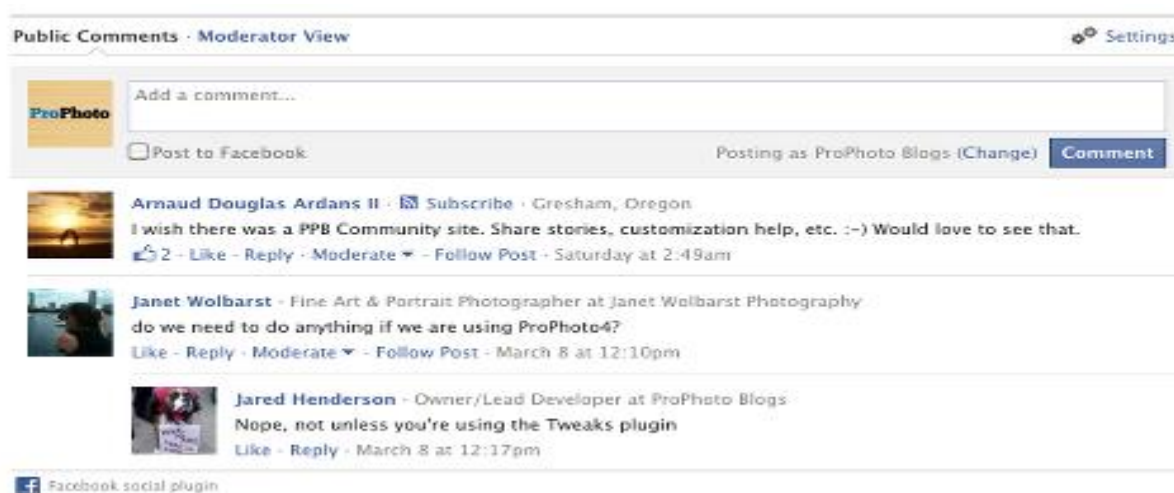
Facebook este cea mai mare rețea e socializate la nivel global. Fiind cea mai utilizată aplicație de pe întreaga planetă.

Modul în care existența Facebook a influențat dezvoltarea acestui proiect este introducerea unui mecanism de socializare asupra cărților asemănător cu cel existent la Facebook. In cazul aplicației de socializare Facebook , socializarea se realizează asupra evenimentelor și acțiunilor publicate de către aceștia. Această idee a fost adaptată și introdusă în cadrul bibliotecii, permițând adăugarea de comentarii , aprecieri ,deaprecieri pentru o carte.

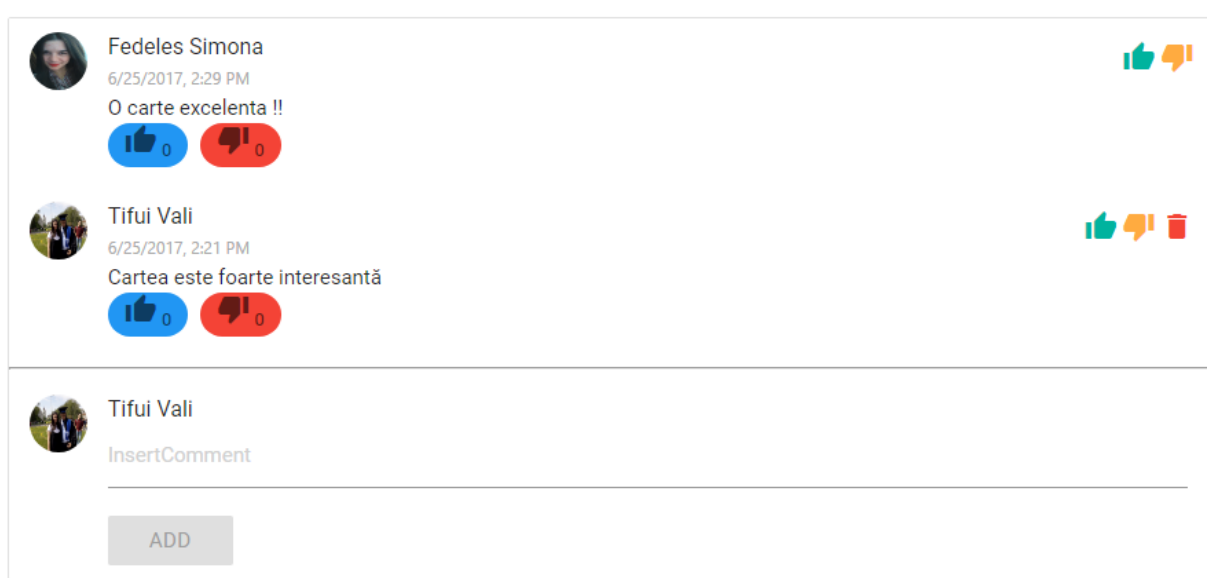
Una dintre cele mai mari diferențe dintre sistemul implementat în cazul propriei aplicații și aplicația Facebook este lipsa unui sistem de notificări , sistem ce este prezent și foarte complex în cadrul aplicației facebook. Dar avand în vedere scopul diferit al celor două platforme, neimplementarea acestui sistem nu afectează biblioteca online într-un mod critic.

Asemănările și diferențele în ceea ce privește managementul comentariilor dintre cele două aplicații poate pot fi observate în imaginile ce urmează.

Comentarii Facebook:



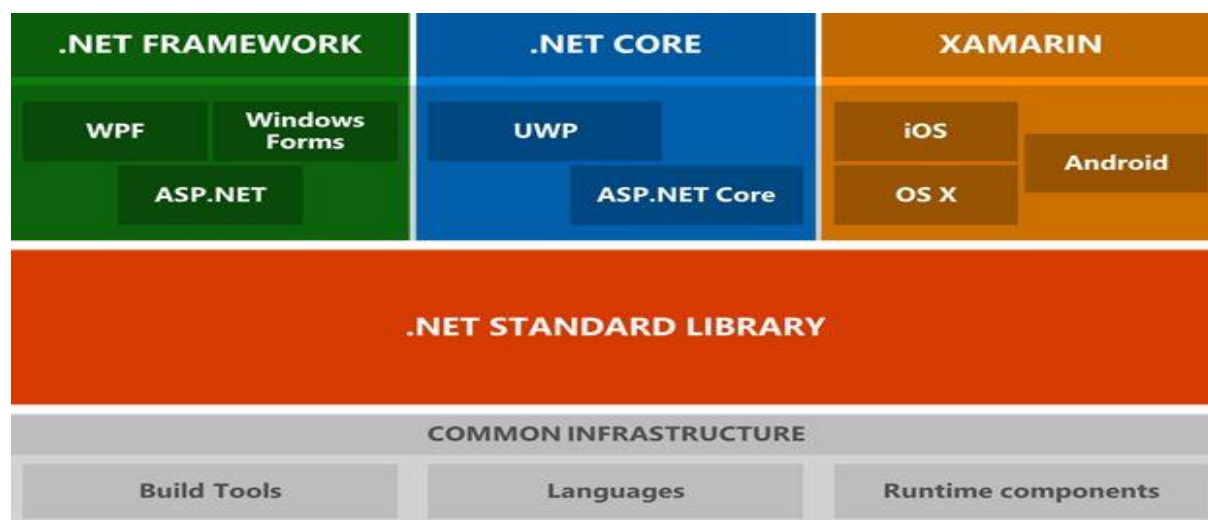
Comentarii Biblioteca Uaic:



## Capitolul II: Tehnologii utilizate

În dezvoltarea acestei aplicații s-au utilizat numeroase tehnologii relativ noi, tehnologii ce vin din zona Microsoft din pachetul .Net, precum și tehnologii destinate dezvoltării de aplicații web SPA, aici fiind vorba de platforma Angular. Din pachetul .Net se utilizează cea mai nouă tehnologie și anume tehnologia .NET Core dezvoltată în anul 2016, fiind un proiect al comunității open source.

### III.1 Arhitectura .NET



Arhitectura.Net conține o colecție foarte largă de limbaje de programare, tehnologii și framework-uri, fiind divizat în 3 mari platforme construite peste o librărie standard.

Prima platformă dezvoltată a fost .Net Framework, o platformă ce viza dezvoltarea de aplicații web și desktop, aplicații ce pot rula doar pe mașini cu sistem de operare Windows. Din această tehnologie fac parte cunoscutele framework-uri .Net, și anume WCF, ASP.NET, Windows Forms, WPF.

În ultimii ani Microsoft a dezvoltat o tentință tot mai mare către oferirea de tehnologii independente de platformă, tehnologii din ce în ce mai variate și diverse. Una dintre aceste tehnologii este tehnologia .Net Core, tehnologie ce expune două framework-uri, ASP.NET CORE framework ce copie funcționalitățile ASP.NET având ca obiectiv principal independența de platformă și îmbunătățirea vitezei de execuție.

Un alt framework oferit de .NetCore este UWP destinat dezvoltatorilor de aplicații Desktop atât linie comandă cât și aplicații cu interfață grafică, și acesta din urmă oferind independență față de platformă.

Tot din pachetul .Net face parte și tehnologia Xamarin tehnologie destinată dezvoltării de aplicații mobile, IOS, Android și Windows Phone. Aceste aplicații pot împărtăși același cod pentru logică, acest cod fiind interpretat de către o mașină de execuție virtuală, dar codul pentru crearea interfeței grafice a aplicației diferă de la un sistem de operare la altul.

La baza acestei arhitecturi se află mecanismele de bază , mecanisme ce asigură abstractizarea mașinii , a platformei hardware sau chiar cea software (sistem de operare) pentru dezvoltator. Acesta din urmă neavând acces direct cu sistemul de operare. La acest nivel fiind implementată mașina virtuală (CLR – Common Language Runtime) ce interpretează codul scris de programator, transformându-l într-un cod de octeți ce pot fi executați de procesorul mașinii reale. Tot la acest nivel este implementat și mecanismul de Garbage Collector, mecanism ce constă în eliberarea memoriei utilizată de execuția programului, eficientizând gradul de utilizare a memoriei RAM.

### III.2 Tehnologia .Net Core

Făcând o analiză a tehnologiilor enumerate mai sus, luând în calcul contextul și cerințele proiectului am considerat că tipul de tehnologie care se pretează cel mai bine este .Net Core datorită caracteristicii cross-platform, a vitezei de execuție a codului și nu în ultimul rând datorită faptului că este o tehnologie nouă și foarte modernă.

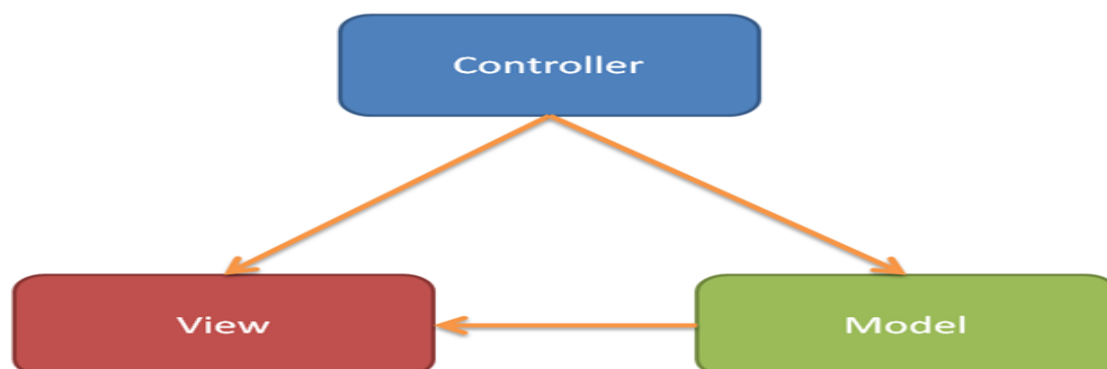
Această tehnologie cuprinde numeroase framework-uri de dezvoltare, cel mai popular fiind framework-ul ASP.NET , dar și alte framework-uri utile cum ar fi Entity Framework Core sau UWP. Aplicațiile și sistemele dezvoltate cu această tehnologie pot rula atât pe sisteme de operare Windows sau Windows Server, pe sisteme de operare Linux, oferind suport pentru numeroase distribuții de linux (Ubuntu, Fedora, Mint, etc) cât și pe mașini ce rulează un sistem de operare Mac Os.

#### III.2.1 ASP.NET CORE

Destinat dezvoltărilor de aplicații și sisteme web independente de platformă cuprinzând o serie de librării și funcționalități necesare implementării aplicației.

Dezvoltarea unei aplicații utilizând această tehnologie se realizează sub pattern-ul de programare MVC (Model-View-Controller).

##### ❖ Model View Controller(MVC)



Este un pattern de programare creațional ce asigură modularitate prin separarea codului aplicației în 3 secțiuni (model , view si controller), oferind astfel o mai bună organizare a codului sursă precum și a logicii aplicației, codul sursă fiind de asemenea mult mai ușor de înțeles.

*Model-ul* este partea ce se ocupă cu modelarea și pregătirea datelor ce urmează a fi prezentate în aplicație. În .Net Core modelul este implementat printr-o clasă ce expune o serie de proprietăți publice.

*View-ul* descrie modul de vizualizare sau prezentare a datelor utilizatorului. În cadrul aplicației curente view-ul constă într-o pagină html ce conține și cod c# necesar manipulării dinamice a modului de prezentare și interpolării datelor (modelului) în cadrul paginii web. În general în .Net, un view este utilizat pentru afișarea modelului în cadrul aplicației.

Deoarece modelul trebuie să fie “transferat” către view, iar scopul view-ului este de a afișa date, reprezentând chiar modelul , această acțiune trebuie realizată de o componentă externă celor două. Această componentă fiind controller-ul.

*Controller-ul* realizează interacțiunea dintre view și controller precum și construirea modelului. În .Net controller-ul se ocupă cu crearea modelului și trimiterea acestuia către view.

#### ❖ *Mecanismul de autentificare JWT*

Mecanismul de autentificare JWT (Java Web Token) este un mecanism de autentificare bazat pe crearea de token-uri JSON de acces cărora le sunt asigurate numeroase drepturi. Spre exemplu server-ul poate crea un token de acces ce are drepturi doar pentru utilizatorii cu rolul de admin.

Autentificarea pe baza acestui mecanism se face în următorii pași:

- User-ul se autentifică în aplicație pe baza unor credențiale.
- Server-ul verifică dacă datele sunt corecte și crează un token asociat respectivului utilizator.
- Se trimite token-ul de acces către client, urmând apoi ca fiecare request făcut din partea client-ului să conțină acest token în header.  
(Ex: `Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXLTJ5Y5CspyHI`)
- Pentru fiecare request se face validarea token-ului de acces pentru a stabili identitatea user-ului.

Structura unui token JWT:

- Header , unde se specifică tipul token-ului și algoritmul de criptare

```
header = '{"alg":"HS256","typ":"JWT"}'
```

- Payload, conține drepturile

```
payload = '{"loggedInAs":"admin","iat":1422779638}'
```

- Semnătura, ce este calculată în felul urmator:

```
key = 'secretkey'  
unsignedToken = encodeBase64(header) + '.' + encodeBase64(payload)  
signature = HMAC-SHA256(key, unsignedToken)
```

Crearea unui token se realizează prin concatenarea header-ului , ce necesită base64 encoding cu rezultatul encodării payload-ului și cu encodarea în format base64 a semnăturii.

```
token = encodeBase64(header) + '.' + encodeBase64(payload) + '.' +  
encodeBase64(signature)
```

Câmpuri standard pentru un token:

- Issuer (iss) – identifică entitatea ce a emis JWT-ul;
- Subject (sub) – identifică subiectul JWT-ului;
- Expiration time (exp) – timpul până la expirarea token-ului;
- Not before (nbf) – data de la care token-ul poate fi acceptat. Dacă token-ul a fost creat la o dată anterioară acestei date , va fi respins;
- Issued at (iat) – data la care token-ul a fost creat;
- JWT ID (jti) – identificator unic al token-ului.

#### ❖ *Autofac Framework*

Autofac este un framework ce expune un serviciu de injectare automată a dependențelor (Dependency Injector)

Această tehnică aplică principiul IOC (Inversion of Control) ce este un patrn de programare foarte cunoscut ce realizează o legătură mult mai slabă între componentele aplicației.

Folosind această tehnică programatorul nu mai este nevoit să fie dependent de implementarea unei componente dar nici de construirea (instanțierea) acesteia manual, framework-ul injectând acea componentă acolo unde este nevoie.

#### ❖ *AutoMapper*

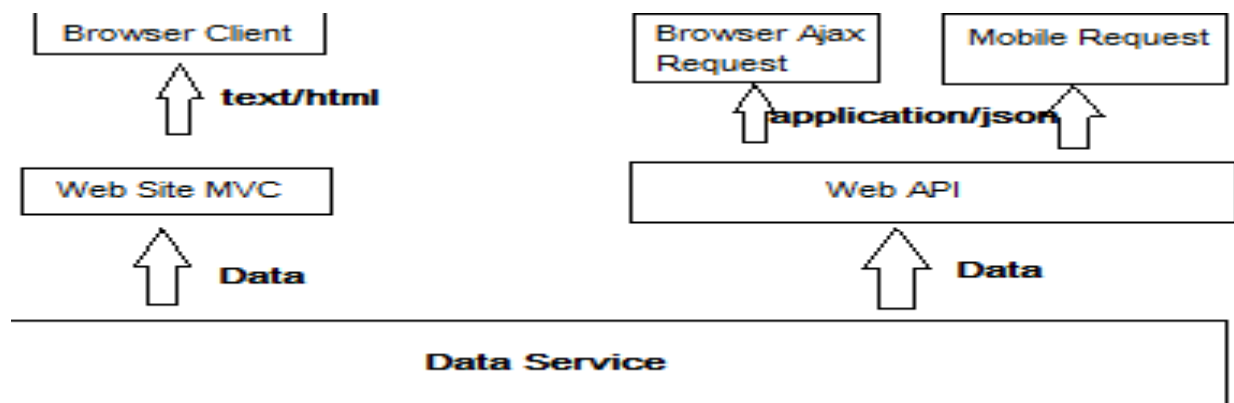
AutoMapper este un framework ce realizează maparea sau covertirea unui obiect la alt obiect, fiind foarte util atunci când în proiect este nevoie de utilizarea unui ORM (Object-relational mapping) pentru accesul la baza de date. Orice ORM necesită crearea unor clase prin care se



modelează entitățile bazei de date, de obicei fiind necesar ca aceste obiecte/entități să fie convertite în obiectele prin care modelăm logica aplicației.

Pentru a configura AutoMapper este necesar să inițializăm clasa statică numită Mapper și să înregistrăm mapările. Pentru a defini o mapare, se specifică tipul obiectului de la care se face maparea, tipul obiectului către care se face maparea și definirea opțiunilor pentru maparea respectivă. Aceste opțiuni pot fi legate de fiecare proprietate a obiectului de la care și pentru care se face maparea respectivă, opțiuni ca ignorarea unei proprietăți sau aplicarea unor acțiuni peste acea proprietate.

#### ❖ ASP NET WEB API



Web API este un framework din cadrul tehnologiei .Net Core ce pune la dispoziția dezvoltatorilor un set de clase și librării pentru dezvoltarea serviciilor web API.

Acest framework utilizează același principiu de programare existent la ASP NET, principiul MVC. Fiind un API aplicația trebuie să expună un end point, acest lucru este realizat prin expunerea de metode publice în clasa controller. Deoarece răspunsul oferit clientului reprezintă un set de date într-un anumit format xml, json sau chiar text, partea de view este neglijată în acest framework, folosindu-se modelul ce este convertit într-un format standardizat.

Pentru construirea modelului este nevoie de a colecta sau primi datele necesare, aceste date fiind stocate de obicei într-o bază de date. Acest framework se poate utiliza cu orice ORM ce suportă tehnologia .Net Core, deci crearea unui model se realizează folosind datele oferite de acest ORM.

### III.2.2 Entity Framework Core

Entity framework este un ORM (Object-relational mappeng) dezvoltat de compania Microsoft făcând parte din platforma .NetCore.

ORM este o tehnică de programare pentru convertirea datelor dintr-un sistem cu tipuri incompatibile (baze de date) în obiecte din programarea orientată obiect. Acesta creează un obiect virtualizat ce descrie o entitate din baza de date. Fiecare tabelă din baza de date este convertită într-un obiect modelat printr-o clasă. Acest obiect conține proprietăți, ce sunt

existente în tabla de legătură ca și coloane. Fiecare înregistrare din tabelă va fi convertită într-un tip orientat obiect.

Din punct de vedere al programării este foarte util să păstrăm codul cât mai inteligibil și clar posibil. Până la apariția acestui framework tehnica folosită pentru accesul la baza de date era ADO.NET. Această tehnologie oferă conectarea și manipularea bazei de date la un nivel mai scăzut fiind necesar scrierea de instrucțiuni specifice sistemelor de gestiune a bazei de date, cod SQL diferit de codul în care este scrisă aplicația. ADO.NET nu oferea o mapare a datelor din baza de date în obiecte din programarea orientată obiect fiind necesară manipularea datelor într-un mod manual. Deoarece nu este plăcut să avem un mix de stiluri și limbaje de programare în cod, în special injectarea de cod SQL în cod din programarea orientată obiect, a fost necesară dezvoltarea acestor ORM-uri ce inovează accesul la baze de date folosind un limbaj de programare orientat obiect.

Entity framework utilizează ADO.NET pentru accesul la baza de date. Conexiunea cu baza de date este manageriată de către un context, context modelat printr-o clasă. Un context conține dbset-uri, colecții de obiecte ce modelează tabelele din baza de date. Fiecare dbset este o colecție de un anumit tip de dată, tip ce modelează înregistrările din acea tabelă.

Pentru a opera cu baza de date nu mai este necesar scrierea de cod SQL, baza de date fiind accesată în manieră orientată obiect. .Net pune la dispoziție o tehnică prin care se poate opera cu colecții și obiecte. Această tehnică se numește LINQ (.Net Language Integrated Query) și poate fi folosită în două moduri:

- Query Syntax – impune scrierea de cod nativ ce seamănă cu limbajul SQL
- Method Syntax – expune metode extinse peste colecții și liste, metode ce pot fi utilizate pentru a selecta, sorta, filtra date din acele colecții.

Folosind aceste tehnici accesul la baza de date se face orientat obiect, bineînțeles cu costul unei performanțe mult mai reduse decât în cazul tradițional.

### **III.3 Limbajul C#**

Limbajul C# a fost dezvoltat în cadrul Microsoft, principalii creatori ai limbajului sunt Anders Hejlsberg, Scott Wiltamuth și Peter Golde. Prima implementare C# larg lansată a fost lansată ca parte a inițiativei .Net în iulie 2000.

Caracteristicile limbajului:

- Este un limbaj de programare simplu, modern, cu utilitate generală. Cu productivitate mare în programare;
- Este un limbaj orientat obiect;
- Permite dezvoltarea de aplicații industriale robuste și durabile;
- Oferă suport complet pentru dezvoltarea de componente software;
- Este un limbaj orientat spre componente, oferind un grad ridicat de modularitate;

- Este un limbaj compilat dar independent de platformă, din compilarea codului rezultând un cod CIL( common intermediate language).

#### ❖ Independența de procesor și de platformă

În momentul în care un program este executat, CLR activează un compilator special, numit JIT (just in time). Acesta preia codul CIL și îl transformă în cod executabil. Codul CIL este independent de platformă deoarece Microsoft a dezvoltat compilatoare JIT ce pot rula pe diferite platforme și sisteme de operare, oferind astfel o interoperabilitate a codului scris în C#.

#### ❖ Managementul automat al memoriei

Alocarea și eliberarea memoriei nu mai este o problemă care trebuie să îi preocupe pe programatori, datorită mecanismului automat de garbage collection.

#### ❖ Securitate

.Net include un mecanism unificat de tratare a excepțiilor, atunci când de pildă se execută o instrucțiune ilegală.

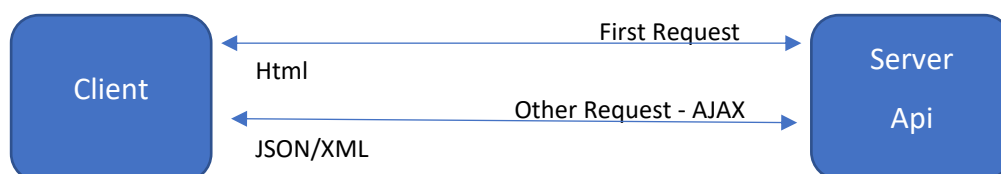
#### ❖ Portabilitate

Un program scris în C# poate fi executat fără nici o modificare pe orice sistem ce are instalat un compilator JIT.

### III.4 Platforma Angular 2

Este o platformă de programare a aplicațiilor web dezvoltată și menținută de către gigantul IT Google, utilizată în dezvoltarea aplicațiilor web de tipul single page application (SPA). Ultima versiune de Angular suportă limbajul de programare TypeScript, un limbaj de programare ce extinde limbajul JavaScript, incluzând aspecte legate de programarea orientată obiect, devenind astfel de la un limbaj netipizat așa cum este JS la un limbaj tipizat.

#### III.4.1 Single Page Application



Este o aplicație web, un site web ce consistă într-o singură pagină. O aplicație de acest tip este construită din componente, fiecare pagină fiind o componentă. Există o ierarhie de componente care interacționează între ele, transferând date sau apelând metode.

O caracteristică importantă a acestei tehnici este faptul că toate acțiunile din aplicație au loc în mod asincron, nefiind necesară o reîncărcare a întregii paginii web după executarea unei anumite acțiuni, ci doar reîmprospătare componentei actuale. Bineînțeles lucrul acesta aduce un plus important de viteză aplicației.

Datorită împărțirii aplicației în componente, se introduce o tehnică de reutilizare a codului, respectiv a componentei prin crearea de componente generice. Astfel se economisește timp util în dezvoltarea aplicației. Totodată această divizare face ca aplicația să fie mult mai modulară, ușor de înțeles și întreținut.

Comunicarea cu server-ul într-o astfel de aplicație se realizează prin apeluri AJAX, apeluri ce sunt realizate în mod asincron, neblocați interfața cu utilizatorul.

Ajax (Asynchronous JavaScript and XML) este o tehnică de programare a aplicațiilor web interactive. Intenția acestei tehnici este de a face paginile web să devină cât mai rapide, prin schimbul în fundal al unor cantități mici de date cu server-ul, astfel încât să nu fie nevoie să se reîncarce pagina la fiecare acțiune a utilizatorului.

În cadrul unei pagini web apelurile AJAX sunt realizate de browser, prin intermediul unui API specializat.

#### III.4.2 XMLHttpRequest:

Versiunile actuale de web browsere expun un API de comunicare cu server-ul prin intermediul obiectului **XMLHttpRequest (XHR)**. Acest obiect expune câteva metode pentru realizarea comunicării client – server.

Metode XMLHttpRequest:

- **open( Method, URL, Asynchronous, Username Password )**, inițializează cererea ce va fi trimisă către server specificând câțiva parametri:
  - *Method* – specifică metoda HTTP (Get, Post, Delete, Patch);
  - *Url* – specifică url-ul către care se realizează requestul;
  - *Asynchronous* – indică dacă cererea este executată în mod asincron;
  - *Username, password* – pot fi specificate dacă cererea necesită autentificare;
- **setRequestHeader( Name, Value )** – setează un câmp din header specificând numele câmpului și valoarea;
- **send( Data )** – trimite cererea către server incluzând în corpul cererii datele specificate;
- **abort()** – oprește execuția unei cereri.

### III.4.3 JSON (Java Script Object Notation)

Json este un format standardizat de memorare a datelor , usor de citit și înțeles de om folosit frecvent în transmiterea de date serializabile în rețea. Fișierele ce au date stocate în acest format vor avea extensia .json și va conține informații stocate în pereche cheie : valoare.

Datele convertite în JSON pot fi de cinci tipuri:

- Numeric – utilizat pentru stocarea numerelor fie întregi fie numere în virgule mobile;
- Boolean – se pot stoca valorile true și false;
- String – stocarea șirurilor de caractere;
- Array – utilizat pentru memorarea colecțiilor;
- Null.

Reprezentarea unui obiect în formatul JSON se realizează printr-un set de proprietăți cheie : valoare pusă între acolade, proprietăți ce descriu obiectul.

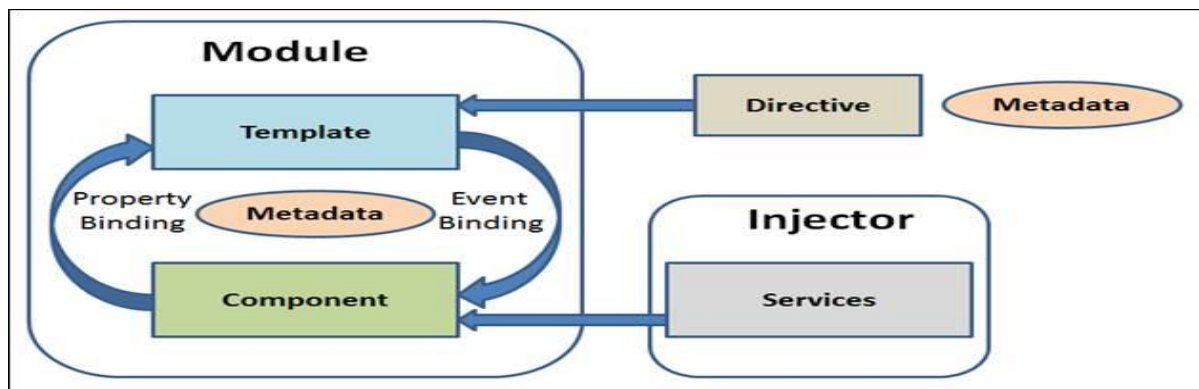
```
{  
  "streetAddress": "21 2nd Street",  
  "city": "New York",  
  "state": "NY",  
  "postalCode": "10021-3100"  
}
```

Orice atribut sau cheie poate stoca la rândul său să stocheze un alt obiect sau o colecție de obiecte, realizându-se o ierarhie de obiecte ușor de citit și înțeles.

Scopul principal al acestui format este trimiterea datelor în rețea. Fiind un format standardizat asigură portabilitate și este ușor de integrat cu orice alt sistem existent.

### III.4.4 Caracteristici Angular 2

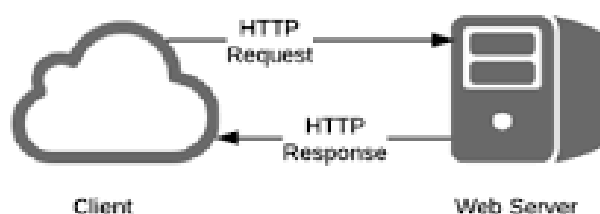
Angular 2 vine cu un set de funcționalități și caracteristici noi față de versiunea anterioară , aici putem menționa limbajul TypeScript , un mecanism de injecție a dependențelor automat, îmbunătățirea definirii modului de rutare a componentelor și altele.



Angular 2 funcționează în maniera MVC. Putem compara Angular cu ASP .NET existând câteva caracteristici asemănătoare doar că discutăm de un concept puțin diferit. La fel ca în modelul MVC de la ASP.NET unde aveam un cotroller , aici întâlnim conceptul de componentă. O componentă este descrisă printr-o clasă ce conține metode și constructori necesari modelării comportamentului componentei. Fiecare componentă are nevoie de un template. Un template este descrierea html a interfeței acelei componente. După cum am văzut la ASP.NET putem compara acest template cu un view obișnuit de acolo. Ceea ce ne lipsește pentru a avea o arhitectură MVC este modelul bineînțeles. Dacă la ASP.NET modelul este reprezentat printr-o simplă clasă ce descrie datele ce se doresc a fi prezentate, aici lucrurile sunt oarecum asemănătoare. Singura diferență este faptul că în loc să utilizăm o conexiune la baza de date pentru a genera acest model , vom utiliza un serviciu care construiește acel model pentru noi. După cum observăm în cazul Angular am putea numi modelul CTS și anume Component-Template-Service. Componenta fiind care utilizează serviciul pentru a recepționa datele de la server date ce reprezintă modelul, apoi acest model este prezentat utilizatorului prin afișarea template-ului ca și componentă html în pagina web. Fiecare acțiune din pagina web respectiv template poate fi tratată ca și metodă a componentei. Acest lucru se realizează prin abonarea la diverse evenimente din cadrul elementelor html , evenimente ce sunt tratate ca și metode în componentă.

Angular dispune de un mecanism integrat de injectare a serviciilor , servicii ce pot fi folosite în interiorul unei componente fără a fi necesar inițializarea acestuia manual.

Angular oferă o diversitate de componente și servicii foarte utile în dezvoltarea aplicației. Unul dintre acestea fiind modulul de rutare o componentă ce ușurează crearea și manipularea rutelor. Fiecare rută este descrisă de o cale și o componentă ce va fi afișată când utilizatorul introduce în bara de adrese această cale.



Un alt modul important este Http care oferă un API foarte util pentru a primi și trimite date către server. Acest serviciu este implementat prin utilizarea apelurilor AJAX de către browser-ul de unde aplicația este utilizată. Metodele care se găsesc la acest serviciu fiind asemănătoare cu cele ale obiectului XMLHttpRequest, metode pentru trimiterea de cereri către server și metode pentru setarea header-ului cererii.

Toate aceste module definite de aplicație trebuie să fie specificate în modulul principal al aplicației ce este definit în fișierul app.module.ts.

Acest fișier conține 3 secțiuni , o listă cu modulele definite în aplicație , o listă cu componentele aplicației și lista cu providerii de servicii utilizați de aplicație.

Odată cu dezvoltarea aplicației aceste liste pot avea dimensiuni din ce în ce mai mari. Dacă toate aceste liste s-ar defini în cadrul modulului principal acesta va deveni foarte greu de înțeles. De aceea este recomandat ca aceste liste să fie stocate în fișiere separate care mai apoi vor fi importate și utilizate în modulul principal.

#### ❖ Crearea unei componente angular

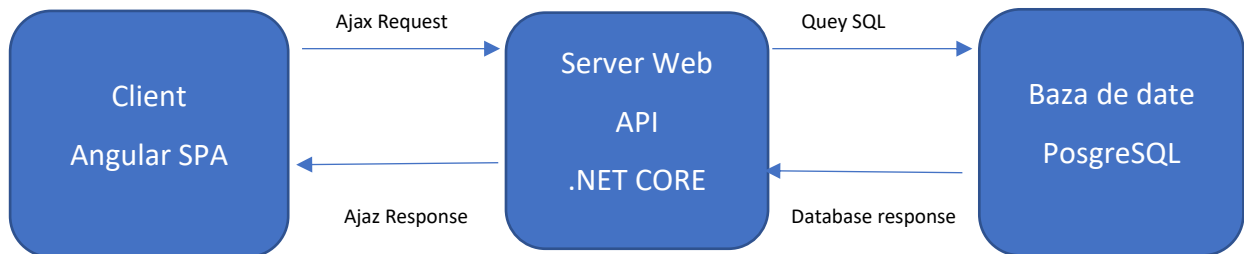
Pentru a crea o componentă angular, se creează clasa unde se definește comportamentul acelei componente , clasă ce trebuie să conțină în mod obligatoriu secțiunea @Component(). În această secțiune se definesc elementele necesare afișării componenteii cum ar fi fișierul de template, selectorul și stiluri.

În cadrul clasei se definesc metode necesare tratării evenimentelor și metode pentru logica aplicației. Pentru a trata un eveniment din cadrul unui element html din template , se va preciza numele evenimentului între paranteze rotunde urmat de egal și expresia ce va fi executată când evenimentul are loc. Expresia este e cel mai multe ori un apel de metodă din cadrul clasei ce definește componenta. Atributele unui element html sau a unei componente angular pot fi setate în mod dinamic specificând acel atribut între paranteze pătrate urmat de egal și valoarea variabilei din cadrul clasei unde sunt memorate informațiile dorite.

Un altă tehnică foarte utilizată în construirea unei componente angular este tehnica de interpolare tehnică utilizată pentru afișarea modelului în cadrul template-ului. Utilizarea acestei tehnici este simplă fiind necesar doar precizarea variabilei sau a expresiei ce va fi evaluată și afișată în cadrul paginii , această expresie fiind poziționată între două perechi de acolade.

# Capitolul III: Arhitectură și implementare

Aplicația este împărțită în trei componente principale: clientul sau aplicația web, serverul web API și baza de date. Fiecare componentă are un rol și un scop bine definit comunicând și transferând date între ele.



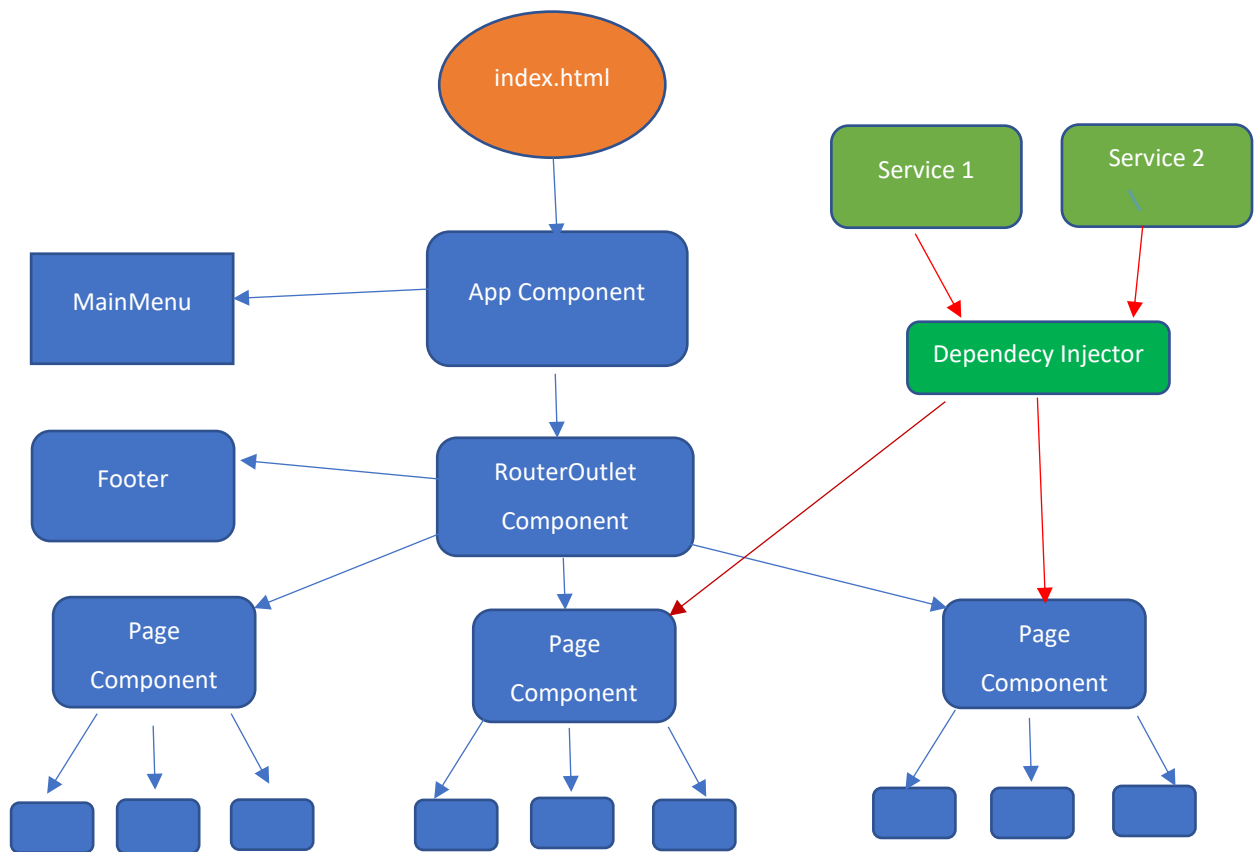
## III.1 Aplicația client web SPA (Angular)

Această aplicație este utilizată pentru crearea interfeței și interacțiunii cu utilizatorul. Datorită ideii aplicației de a dezvolta o bibliotecă care pe lângă afișarea unor liste de cărți , oferă și funcționalități mai complexe cum ar fi citirea , anotarea , salvarea paginii curente , adăugarea de comentarii asupra unei cărți, managementul resurselor precum și a altor funcționalități ce necesită o bună sincronizare cu server-ul de unde se preiau datele aplicației. Deoarece reîncărcarea paginilor la fiecare acțiune a utilizatorului nu este cea mai bună soluție pentru acest tip de aplicație , SPA (Single Page Application) rezolvă această problemă aducând optimizare și un plus de eficiență. Angular este unul dintre cele mai utilizate framework-uri pentru dezvoltarea aplicațiilor SPA, remarcându-se prin gradul de modularitate și reutilizare pe care îl oferă utilizarea componentelor.

### III.1.1 Arhitectura aplicației Client

Aplicația client este alcătuită din componente și servicii ce se ocupă cu managementul datelor. Cea mai importantă componentă este componenta principală, app component, componentă ce va conține toate celelalte componente. Fiecare componentă ce are nevoie să recepționeze sau să trimită date către server va utiliza un serviciu pentru acest scop.





#### ❖ Pagina Principală (index page)

Pagina principală este prima pagină a aplicație afișată de browser în timpul încărcării aplicației și conține componenta de bază app component. În cadrul acestei pagini sunt încărcate foile de stil declarate global precum și a librăriilor javascript.

#### ❖ Componenta Principală (App Component)

Această componentă este creierul aplicației, ea fiind cea care controlează toate celelalte sub componente. Această componentă include 3 copii, componenta NavBar, componenta Footer și componenta RouterOutlet

#### ❖ Componenta NavBar (NavBar Component)

Este utilizată pentru afișarea și controlarea barei de navigație prezente în partea de sus a aplicației. Componenta LanguageList este un copil al acestei componente având ca și rol afișarea unui element de tip dropdown list în cadrul barei de navigare și controlează schimbarea limbei aplicației. Această componentă utilizează serviciul AuthService pentru a determina autenticitatea utilizatorului , astfel încât opțiunile din meniul afișat să poată fi filtrate în funcție de rolul utilizatorului.

#### ❖ Componenta Footer

Prezintă o scurtă descrierea a bibliotecii precum și a câtorva linkuri utile. La fel ca și componenta NavBar este o componentă fixată ce se întâlnește pe toate paginile aplicației.

#### ❖ Configurarea rutelor

Pe următorul nivel în cadrul ierarhiei de componente se află componenta router outlet, o componentă necesară pentru afișarea paginilor din cadrul aplicației. Fiecare pagină este în esență o componenta angular. Componenta router outlet este copilul componenteii app și controlează care este componenta ce va fi afișată în funcție de calea ce a fost introdusă în bara de adrese.

Rute existente în aplicație:

Cale	Componentă	Descriere
/home	HomeComponent	Pagina principală
/signup	SignUpComponent	Pagina de înregistrare a unui utilizator
/login	LoginComponent	Pagina de autentificare
/studentaccount	StudentProfileComponent	Pagina de vizualizare a profilului studentului
/books	BookPageComponent	Pagina de vizualizare a cărților existente în bibliotecă
/viewbook/:bookId	BookViewerComponent	Pagina de vizualizare a unei cărți
/administration	AdministrationPageComponent	Pagină destinată utilizatorului cu rolul de admin pentru administrarea bibliotecii
/editBookId	AdministrationPageComponent	Pagina de editare a unei cărți
/editAuthor/:authorId	AdministrationBookComponent	Pagina de editare a unui autor
/editFaculty/:facultyId	AdministrationBookComponent	Pagina de editare a unei facultăți
/latestBooks	LatestBooksComponent	Vizualizarea noutăților în materie de cărți
/authors	AuthorsComponent	Pagina de vizualizare a autorilor
/faculties	FacultiesComponent	Pagina de vizualizare a facultăților
Altă cale	PageNotFoundComponent	Pagină de notificare a utilizatorului în legătură cu inexistența căii introduse

Toate routele specificate mai sus, în aplicație sunt definite în fișierul routes , fișier importat în modulul principal specificând rutarea căilor aplicației.

### III.1.2 Componente tip pagină

La baza fiecărei pagini din aplicație există o componentă angular ce deservește acea pagină. Fiecare componentă de acest tip poate conține alte componente copil necesare vizualizării conținutului paginii respective. Fiecare componentă pagină are nevoie de un template, template ce descrie interfața grafică a paginii, și un serviciu pentru managerierea datelor dacă este nevoie.

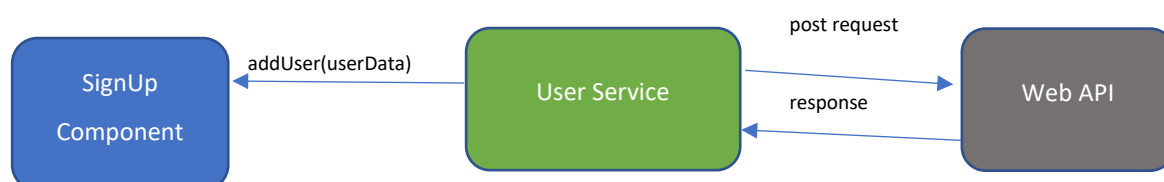
#### ❖ Pagina Principală (Home page)

Constă în afișarea conținutului salvat de către administratorul bibliotecii. Ca și mod de funcționare se preiau datele de la server prin intermediul serviciului AdministrationService serviciu ce implementează o metodă publică ce returnează conținutul paginii principale setate de administrator. După apelarea acestei metode conținutul este afișat în pagină.



#### ❖ Pagina de înregistrare (SignUp Page)

Prezintă utilizatorului un formular de înregistrare. Acest formular cuprinde informații legate de autenticitatea utilizatorului, facultatea de care acesta aparține, adresa de email ce va fi utilizată ca și credențial de autentificare precum și o parolă. Toate aceste date urmează a fi “împachetate” într-un singur obiect apoi trimise către server pentru a fi verificate și mai apoi salvate în baza de date. Pentru aceste operații componenta utilizează serviciul UserService, serviciu ce oferă un end point pentru manipularea acestor informații.



#### ❖ Pagina de autentificare (SignUp Page)

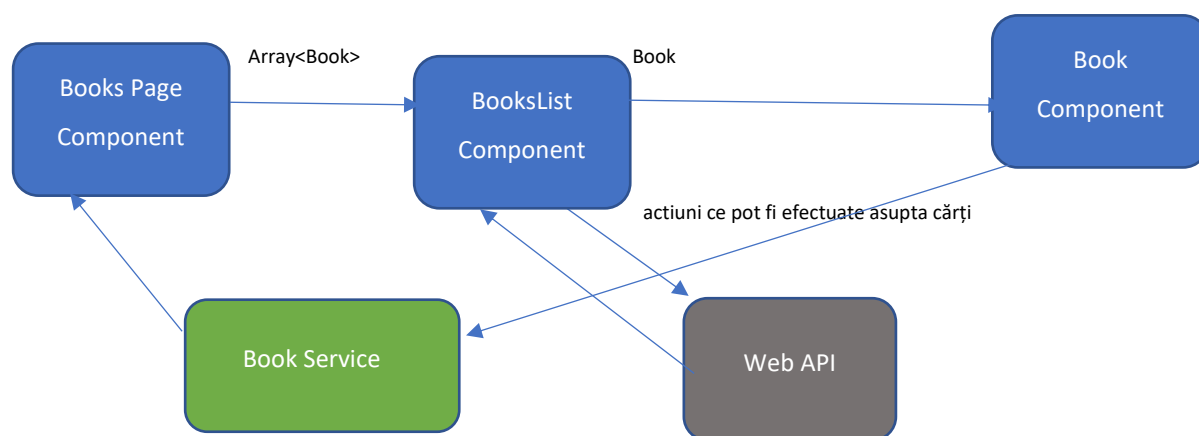
Utilizată pentru a autentifica un utilizator. Orice utilizator fie el admin este recunoscut după o adresa de email cu care s-a înregistrat și o parolă. Informațiile introduse de utilizator sunt verificate folosind metoda `verifyUser(accountData)` din serviciul `AuthService`.

#### ❖ Pagina Noutăți

Pentru accesarea acestei pagini nu este necesar ca utilizatorul să fie autentificat. Componenta ce stă la baza acestei pagini este `LatestBooksComponent`. Această componentă utilizează o altă componentă responsabilă cu afișarea unei liste de cărți. În cadrul acestei pagini vor fi afișate ultimele 10 cărți adăugate în bibliotecă. Preluarea datelor se realizează folosind serviciul `BooksService`, serviciu ce expune pentru această operațiune metoda `getLatestBooks()`

#### ❖ Pagina de vizualizare a cărților

Dispune de o secțiune de filtrare a cărților, fiind posibilă filtrarea după titlul cărții, categorie, autori și facultăți. Procesul de filtrare constă în crearea unui obiect `BookFilter` ce conține informațiile precizate mai sus, obiect ce este transmis către server folosind metoda `getFilteredBooks()` din cadrul serviciului `BookService`. Rezultatele primite ca o listă de cărți vor fi folosite în cadrul componentei `BookList` componentă copil a componentei `BooksPageComponent`. Mai departe fiecare carte va fi afișată folosind componenta `Book`, componentă ce se ocupă cu prezentarea informațiilor și a acțiunilor ce pot fi efectuate asupra unei cărți.



#### ❖ Pagina de profil al utilizatorului

Accesând această pagină utilizatorul poate vizualiza și edita datele aferente profilului său. Aceste date sunt cerute de la server prin intermediul serviciului `UserService`, prin apelarea metodei `getUser()` expusă de acest serviciu. Pentru schimbarea imaginii de profil al utilizatorului se folosește o componentă copil, componenta `SingleFileUploader`. Această componentă este o componentă reutilizabilă folosită pentru încărcarea de fișiere.

În cadrul acestei componente mai sunt de asemenea referențiate și ale serviciilor, pentru afișarea diverselor informații despre acțiunile utilizatorului. Unul dintre aceste servicii este și `BookService`, utilizat pentru crearea de statistici privind accesarea cărților. Aceste statistici sunt prezentate utilizatorului sub formă de tabele.

#### ❖ Pagina autorilor

Asemenea componentei de afișare a cărților, scopul acestei componente este de afișare a listei de autori prezenți în aplicație. Deoarece în pagină există și o componentă ce se ocupă de paginare , procedeul de request al listei de autori trebuie să suporte paginare. Paginarea este efectuată prin trimiterea obiectului AutorFilter către server obiect ce este derivat din PageFilter ce are două proprietăți necesare paginării: pagina și dimensiunea paginii.

### III.1.3 Alte Componente Angular

#### ❖ Book Component

Utilizată pentru prezentarea cărții ca și element al unei liste, BookComponent ocupă un rol foarte important în cadrul aplicației fiind componenta ce răspunde de toate acțiunile pe care un utilizator le poate efectua asupra unei cărți. Componenta primește cartea prin setarea atributului [book] . La efectuarea unei acțiuni asupra cărții componenta comunică cu serverul prin intermediul serviciului BookService. Unele acțiuni precum vizualizarea cărții sau descărcarea acesteia nu necesită comunicare explicită cu serverul ci doar o redirectionare către o altă componentă din aplicație.

#### ❖ Single File Uploader

Este o componentă reutilizabilă folosită pentru încărcarea de fișiere către server. Această componentă este utilizată pentru încărcarea efectivă a conținutului cărților, încărcarea de imagini aferente lor, precum și în pagina de profil, pentru schimbarea imaginii utilizatorului. Încărcarea unei imagini se realizează prin folosirea unui serviciu extern, serviciu ce diferă de la context la context. Pentru încărcarea fișierelor pentru o carte, serviciul utilizat este BookService, în timp ce serviciul utilizat pentru schimbarea imaginii de profil al utilizatorului este UserService. Acest serviciu trimite conținutul fișierului către server și tratează răspunsul oferit de acesta , furnizând calea de pe server către acel fișier.

#### ❖ Dropdown Component

Deoarece în aplicație este nevoie în diverse locuri de o componentă de tip dropdown utilizată pentru selectarea unei opțiuni dintr-o listă, s-a dezvoltat această componentă generică care se ocupă de acest aspect. Opțiunile ce pot fi selectate sunt trimise componentei prin intermediul atributului [items], aceste opțiuni fiind de tipul DropdownItem. Un obiect dropdown item este descris de id , nume și valoare. Pentru a seta care este elementul curent selectat , se utilizează atributul [selected], precizând elementul ce se dorește a fi selectat.

Principalul eveniment ce se desprinde din cadrul acestei componente este evenimentul (onChange) eveniment ce are loc. Atunci când utilizatorul selectează sau schimbă opțiunea curentă.

#### ❖ MultiSelect Component

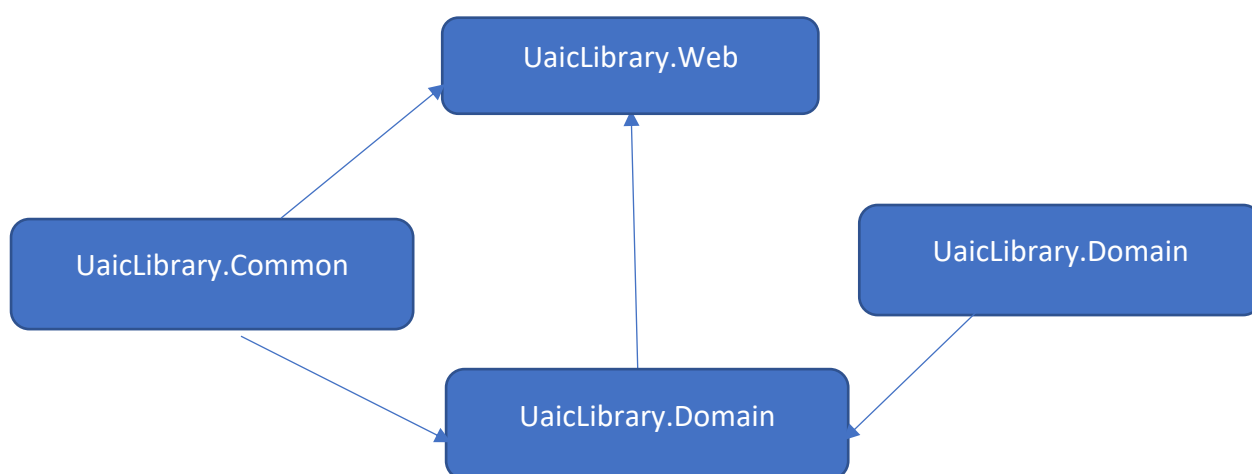
Asemănătoare componenteii dropdown , aceasta din urmă se desprinde din necesitatea de a putea selecta opțiuni multiple. De data aceasta opțiunile sunt de tipul MultiSelectItem , obiecte descrise de un id, nume, valoare , dar și de un câmp de tip boolean ce determină dacă opțiunea este selectată sau nu.

La selectarea unei opțiuni componenta tratează evenimentul, emițând mai departe către componenta părinte evenimentul dataChange. Pentru abonarea la acest eveniment se folosește sintaxa angular “banan in the box” sintaxă ce permite setarea unui atribut dar și abonarea la evenimentu emis de acel atribut. Acest procedeu poartă numele de two way data binding.

Pentru setarea opțiunilor seletate se va specifica o listă cu opțiuni având proprietatea isSelected true, pentru opțiunile ce se doresc a fi setate.

### III.2 Server-ul Web API

Dezvoltat sub umbrela tehnologiei .NET core , utilizând framework-ul de dezvoltare a aplicațiilor web, ASP.NET CORE, acest proiect este partea de bază a întregii aplicații , aici având loc logica aplicației precum și prelucrarea de date. La dezvoltarea acestei soluții s-a utilizat o arhitectură onion, fiind compus din 4 proiecte , fiecare având un rol și o logică aparte.



## Integrarea cu aplicația client SPA

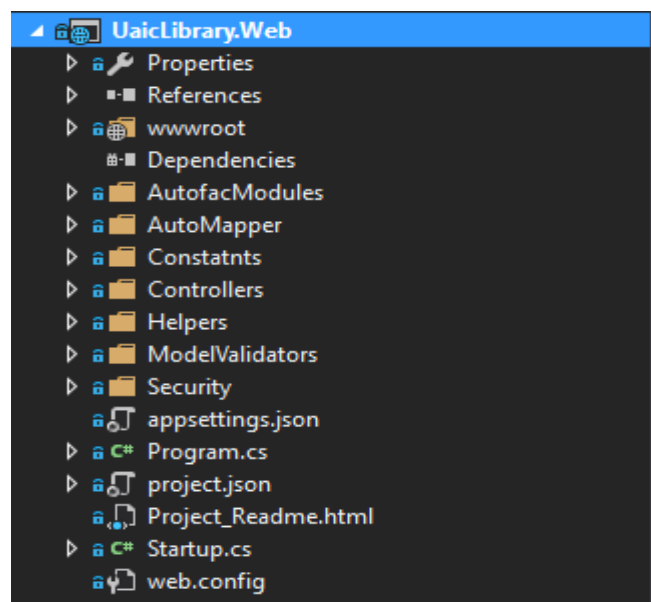
Fiecare cerere primită de la aplicația client , este tratată în cadrul proiectului Web, proiect ce expune endpoint-ul de conexiune cu aplicația client. Dacă cererea necesită date sau prelucrare de date , atunci se va apela la un o funcționalitate existentă în cadrul proiectului Domain. Dacă cererea necesită preluarea de date din baza de date , atunci se va utiliza funcționalități și servicii prezente în proiectul destinat accesului la baza de date a aplicației.

Fiecare cerere primită de la aplicația client va fi autentificată prin verificarea token-ului. Dacă acesta este valid atunci cerea trece mai departe în controller pentru a fi tratată. Deoarece aplicația client și server-ul API nu rulează sub același domeniu, UaicLibrary web utilizează framework-ul CORS (Cross-Origin -Resource-Sharing) , un middleware ce se impune între controller și preluarea cererii care stabilește dacă originea cererii este acceptată. Această acceptare se realizează prin specificarea domeniului și a metodelor http acceptate. În cadrul acestui API este acceptat doar domeni-ul clientului SPA, cu toate metodele http.

### III.2.1 UaicLibrary.Web

Acest proiect își propune implementarea efectivă a back-end-ului întregii aplicații. Orice request din aplicația client este tratat prima oară în cadrul acestui proiect prin expunerea unui API (Application Programming Interface), ușor de utilizat și înțeleș.

Ca și structură a proiectului, acesta include diferite framework-uri utile în tratarea requestului, precum și o serie de configurări.



#### ❖ Configurare Autentificare

Se utilizează autentificare JWT , autentificarea bazată pe generarea și utilizarea de token-uri. Implementarea acestei autentificări constă în specificarea unui midleware situat între controller

și punctul de recepționare a cererii. Acest middleware verifică dacă în cadrul header-ului requestului este setat un token valid.

Pentru generarea unui token valid, s-a creat controller-ul Connect ce expune metoda Post care primește ca și argumente informațiile utilizatorului, informații dobândite prin parsarea query string-ului din request. Aceste informații sunt verificate printr-un apel la baza de date. Dacă informațiile sunt valide are loc generarea token-ului de acces folosind opțiunile de generare din fișierul JwtIssuerOptions.

Opțiuni generare token:

Opțiune	Valoare
ValidFor	TimeSpan.FromMinutes(240)
NotBefore	DateTime.Now
IssuedAt	DateTime.Now
Expiration	IssuedAt.Add(ValidFor)

#### ❖ Configurare și utilizare Autofac

În folder-ul AutofacModules există patru module principale ce sunt înregistrate în autofac. Primul modul DatabaseModule înregistrează interfețele și clasele utilizate pentru comunicarea cu baza de date. Cel de-al doilea modul RepositoriesModule înregistrează repozitoriurile din cadrul proiectului Domain. Ultimele module fiind destinate înregistrării validatorilor (ValidatorModule) și a componentelor web ce trebuiesc injectate (WebModule).

Toate aceste referințe înregistrate în cadrul builder-ului vor fi injectate unde acestea sunt utilizate. Majoritatea utilizărilor au loc în constructorii de clasă.

#### ❖ Configurare și utilizare AutoMapper

Conversia între obiecte de tipuri diferite este realizată utilizând acest framework. Pentru realizarea unei converșii este necesar specificarea tipurilor între care are loc conversia și a opțiunilor de conversie. Definirea mapărilor se realizează în clase ce sunt derivate din clasa Profile, în corpul constructorului.

Pentru proiectul UaicLibrary.Web mapările sunt definite într-o singură clasă GeneralWebAutoMappings.

Utilizarea mapărilor este simplă, folosind clasa statică Mapper ce are două metode importante Map() și Map<>().

#### ❖ Utilizare MVC



Fiind o aplicație ASP NET CORE întreaga structură este bazată pe utilizarea acestui pattern.

Fiecare entitate (Model, View sau Controller) este definită printr-o clasă. Clasele ce descriu modele sunt simple și conțin doar proprietăți ce pot fi citite și setate. Aceste modele sunt trimise ca și rezultat al cererii, reprezentând informațiile cerute.

Modele existente în aplicație:

Clasa Model	Proprietăți
Author	Id, Nume, ImageUrl, DetailsPageUrl
BookCategory	Id, Name, Details
SelectedItemModel	Id, Name, Value, IsSelected
Book	Id, Isbn, BookAuthors, Description, Name, NumberOfPages, Comments, BookCategory, PublishDate, ImageUrl, Likes, Dislikes, Faculty, BookUrl, Views, Reads, PageAnnotations
Faculty	Id, Name, Address
Comment	Id, User, Text, Likes, Dislikes
MatricolNumber	Matricol
Setting	Key, Value
Student	Id, MatricolNumber, Email, FirstName, LastName, Password, About, FullName, ReadedBooks, OpennedBooks, ImageUrl, EditedBooks, Year, Faculties
User	Id, MatricolNumber, Email, Password, FirstName, LastName, Role, About, ImageUrl, Faculties

Deoarece aplicația expune un web API, componentele de tip view sunt inexistente, datele fiind trimise către client prin convertirea modelului la un format standard, formatul JSON (JavaScript Object Notation).

Cea mai importantă componentă a acestui pattern este controller-ul. Controller-ul este de asemenea o clasă ce expune metode, reprezentând un punct de acces ( o cale) în aplicație. În constructorul unui controller sunt injectate repository-uri utilizate pentru logica aplicației. Controller-ul este cel care preia cererea , validează datele primite folosind un validator din domeniu, dacă este cazul apoi preia datele necesare prin apelarea unei metode din repository și trimite rezultatul cererii.

Fiecare clasă controller este derivată din clasa `BaseApiController` clasă ce este la rândul ei derivată din clasa `Controller`. `BaseApiController` implementează câteva metode utile pentru trimiterea răspunsului către client. Una dintre aceste metode este `ValidationResult`, metodă ce primește un obiect de tip `Result` și crează un răspuns de tipul `Ok (200)` dacă rezultatul nu conține erori, respectiv un răspuns de tipul `BadRequest(400)` dacă rezultatul conține erori. Tot în cadrul acestei clase sunt definite proprietăți despre utilizator (`Email`, `Username`, `UserId`), informații ce pot fi accesate din orice controller.

Controller-ul ocupă un rol foarte important în cadrul definirii rutelor în aplicație. Fiecare end point este reprezentat de o metodă din controller. Un end point este definit de numele controller-ului eliminând terminația `“Controller”`, urmat de numele metodei din controller.

#### ❖ Rutele expuse de aplicația WEB API

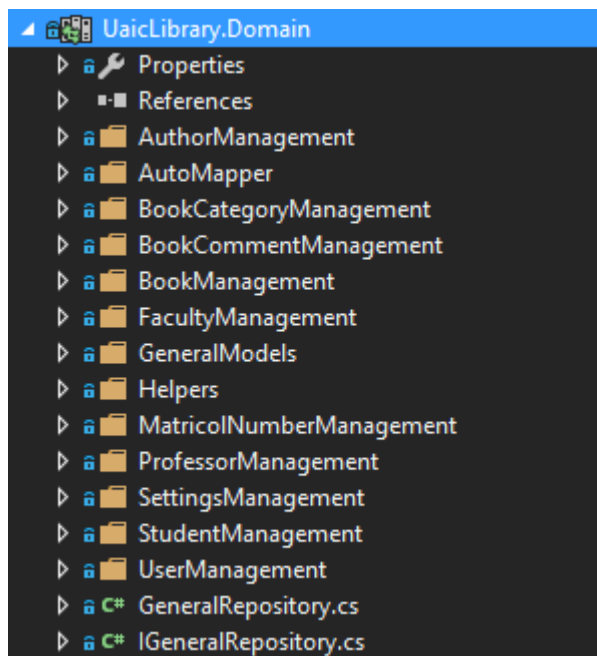
Cale	Metoda HTTP	Controller	Metod ă Controller	Descriere
/connect	Post	Connect	Post	Se obține token-ul de access în aplicație
/book/get	Get	Book	Get	Returnează lista tuturor cărților din bibliotecă
/book/getLatest	Get	Book	GetLatest	Returnează lista cu ultimele zece cărți adăugate în bibliotecă
/book/getFiltered	Get	Book	GetFiltered	Returnează lista de cărți filtrată. Se vor seta proprietății filter-ului în query string
/book/getBookById	Get	Book	GetBookById	Returnează o carte specificând id-ul.
/book/like	Post	Book	Like	Incrementează numărul de

				like-uri a cărții specificate pentru utilizatorul curent.
/book/getBookUserLikes	Get	Book	GetBookUserLikes	Returnează lista cu utilizatori care au dat like cărții specificate în request.
/book/Download	Get	Book	Download	Descarcă conținutul cărții specificate
/book/MarkAsRead	Post	Book	MarkAsRead	Marchează cartea specificată ca și citită
/book/getBookAnnotationContent	Get	Book	GetBookAnnotationContent	Returnează conținutul anotării pentru cartea și pagina specificată
/book/saveBook	Post	Book	SaveBook	Salvează o carte în baza de date
/book/saveBookAnnotation	Post	Book	SaveBookAnnotation	Salvează anotarea specificată în baza de date
/student/get	Get	Student	Get	Returnează informațiile despre studentul specificat
/student/updateDescription	Post	Student	UpdateDescription	Actualizează descrierea studentului specificat

/faculty/getFaculty	Get	Faculty	GetFaculty	Returnează facultatea după id-ul specificat în request
/faculty/saveFaculty	Post	Faculty	SaveFaculty	Salvează sau actualizează facultatea specificată
/user/post	Post	User	Post	Salvează informațiile utilizatorului
/user/uplaodAvatar	Post	User	UploadAvatar	Încarcă imaginea utilizatorului
/user/getCurrentUser	Get	User	getCurrentUser	Returnează utilizatorul autentificat curent
/user/getLatestOpenedBooks	Get	User	GetLatestOppenedBooks	Returnează lista cu cele mai recente cărți deschise de utilizatorul specificat
/bookComents/get	Get	Book	Get	Returnează comentariul după id-ul specificat
/bookComents/addComment	Post	Book	AddComment	Adaugă un nou comentariu pentru cartea specificată și utilizatorul conectat

### III.2.2 UaicLibrary.Domain

În cadrul acestui proiect este dezvoltată întreaga logică aplicației. Pentru fiecare entitate a aplicației este creat un director Entitate plus sufixul management, director ce conține clasele și interfețele acestora necesare pentru a realiza managementul entității respective.



În cadrul fiecărui director se găsește modelul ce descrie entitatea utilizată, o clasă repository ce face managementul efectiv și un proful automapper ce descrie mărirele dintre modelul logic și modelul bazei de date (clasa POCO).

Fiecare repository implementat în acest proiect este utilizat în proiectul web prin injectarea sa în constructorul fiecărui controller ce necesită acel repository.

Fiecare repository are injectat un context al bazei de date prin intermediul căreia se efectuează operații CRUD (Create Read Update Delete) în baza de date.

#### ❖ Managementul autorilor

Interfața `IAuthorRepository` expune cinci metode prin care se realizează managementul autorilor din aplicație. Clasa ce implementează această interfață este `AuthorManagement`. Metodele publice expuse sunt:

- `IList<Author> GetAllAuthors` – returnează lista tuturor autorilor salvați în baza de date a aplicației
- `Result<Author> SaveAutor (Author autor)` – salvează sau actualizează informațiile unui autor în baza de date. Această metodă are ca și rezultat un obiect de tip `Result`, obiect ce descrie dacă operația a fost realizată cu succes sau au apărut erori în timpul execuției celei operații.
- `Result<Author> GetAutorById(int id)` – returnează rezultatul căutării autorului în baza de date după id-ul specificat
- `Result<Author> RemoveAuthorById(int id)` – returnează rezultatul ștergerii autorului după id-ul specificat
- `Result<AuthorData> GetFilterd(AuthorFilter filter)` – returnează lista autorilor filtrată în funcție de filter-ul aplicat

## ❖ Managementul facultăților

Se realizează prin utilizarea interfeței *IfacultyRepository* , interfață implementată de clasa *FacultyRepository* ce conține următoarele metode:

- *IList<Faculty> GetAllFaculties()* – returnează lista facultăților înscrise în aplicație
- *Result<Faculty> GetFacultyById( int facultyId )* - returnează facultatea în funcție de id-ul facultății ce este precizat ca și parametru
- *Result<Faculty> SaveFaculty(Faculty faculty)* – adaugă sau actualizează facultatea în baza de date
- *Result RemoveFacultyById( int facultyId)* – ștege facultatea cu id-ul facultyId din baza de date
- 

## ❖ Managementul cărților și a acțiunilor asupra unei cărți

Pentru managementul cărților și acțiunilor asociate unei cărți se utilizează repository-ul *BookRepository*, clasă ce implementează interfața *IbookRepository* ce expune următoarele metode:

- *Ilist<Book> GetAll()* – returnează lista tuturor cărților existente în aplicație
- *Result<Book> GetBookById* – rezultatul căutării cărții în funcție de id-ul precizat
- *Ilist<Book> GetLatestBooks(int count)* – returnează lista utimilor cărți citite. Numărul de rezultate returnate este specificat de variabila count.
- *Ilist<BookCategory> GetBookCategories()* – lista tuturor categoriilor de cărți ce există în aplicație
- *BookData GetFiltered(BookFilter filter)* – returnează lista de cărți filtrată după aplicarea filter-ului primit ca și parametru.
- *Result Like(int bookId, int userId)* – adaugă o apreciere a utilizatorului cu id-ul userId pentru cartea cu id-ul bookId.
- *Result Dislike(int bookId, int userId)* – adaugă o deapreciere a utilizatorului cu id-ul userId pentru cartea cu id-ul bookId
- *IList<User> GetBookUserLikes ( int id, bool isLike)* – returnează lista cu utilizatorii care au apreciat sau deapreciat cartea cu id-ul precizat ca parametru. Parametrul isLike specifică dacă se dorește interogarea aprecierilor sau a deaprecierilor.
- *IList<User> GetBookOpenners(int bookId)* – returnează lista tuturor utilizatorilor ce au deschis cartea cu id-ul bookId.
- *IList<User> GetBookReaders(int bookId)* – returnează lista utilizatorilor ce au citit cartea cu id-ul bookId.
- *Result MarkAsRead(int bookId, int userId)* – marchează cartea cu id-ul bookId ca și citită de utilizatorul cu id-ul userId.
- *Bool IsBookMarkesAsRead(int bookId, int userId)* – verifică dacă cartea cu id-ul bookId este marcată ca și citită de către utilizatorul userId.
- *Result<BookPageMark> GetLastSavedPageMark(int userId, int bookId)* – returnează ultimul semn de carte specificat de utilizatorul cu id-ul userId cărții cu id-ul bookId.

- *Result SavePageMark(BookPageMark bookPageMark)* – salvează un semn de carte
- *Result<BookPageAnnotation> GetBookPageAnnotation( int bookId, int userId, int page)* – returnează anotarea realizată de utilizatorul cu id-ul userId , asupra cărții cu id-ul bookId la pagina page.
- *Result<Book> SaveBook(BookAdministrationModel model)* – adaugă sau actualizează o carte în aplicație
- *Result RemoveBookById(int id)* - sterge cartea din baza de date

#### ❖ Managementul utilizatorilor

Pentru a realiza managementul utilizatorilor se utilizează atât repository-ul UserRepository cât și provider-ul UserProvider.

Metode ale interfeței IUserRepository:

- *Result<User> CreateUser(User user)* – această metodă este utilizată pentru înregistrarea noilor utilizator. În cadrul acestei metode sunt efectuate o serie de validări ale datelor.
- *IList<Book> GetLatestOpenedBooks(int count, int userId)* – returnează lista celor mai recente cărți citite de utilizatorul cu id-ul userId. Numărul de cărți returnate este egal cu valoarea variabilei count primită ca și parametru.
- *IList<BookAnnotatedModel> getLastAnnotatedBooks(int count, int userId)* – returnează lista anotărilor efectuate de utilizatorul cu id-ul userId. Numărul lor depinde de variabila count.

#### ❖ Clasa Result

Result este o clasă helper dezvoltată pentru reprezentarea rezultatului efectuării a unei operații în cadrul aplicației. Deoarece logica aplicației este bazată pe accesul intens la baza de date, această clasă vine în ajutorul descrierii rezultatului acestor tipuri de operații. Orice operație fie ea cât de simplă poate întâlni erori în timpul execuției. Acest fapt implică adaptarea unui procedeu de notificare a utilizatorului despre problemele apărute. Fiecare metodă din aplicație ce este predispusă la apariția erorilor va returna un obiect de acest tip.

Proprietățile clasei:

- *bool IsSuccess* – flag ce indică dacă operația s-a efectuat cu succes
- *bool IsFailure* – flag ce indică dacă operația a întâmpinat probleme în timpul execuției
- *IEnumerable<Error>* - listă ce conține erorile întâlnite în timpul execuției operației. Această este vidă atunci când operația s-a efectuat cu succes. Fiecare eroare este reprezentată de un cod de eroare și o serie de parametri

- *T Value* – reprezintă obiectul rezultat în urma execuției operației cu succes. Această proprietate este disponibilă doar în cadrul clasei *Result<T>* , clasă generică.

Metodele clasei:

- *Result Ok()* – este o metodă statică din clasa *Result* ce crează un obiect de tipul *Result* ce desemnează un rezultat efectuat cu succes.
- *Result Ok(T value)* – returnează un obiect *Result* cu proprietatea *IsSuccess* setată și proprietatea *value* egală cu valoarea primită ca și argument.
- *Result Fail(string code, object[] paramms)* – inițializează și returnează un obiect *result* cu proprietatea *IsFailure* setată , adăugând eroarea din parametru în lista de erori.
- *Result Combine(Result[] results)* – returnează un obiect de tip *result* , rezultat prin combinarea listei de obiecte *result*, primite ca și parametru.

### III.2.3 UaicLibrary.Database

Acest proiect este destinat accesului aplicației la baza de date. Ca și framework ORM de acces la baza de date este utilizat *EntityFrameworkCore*. După modul de funcționare și utilizare a acestui framework este necesar construirea unui context al bazei de date , context ce cuprinde descrierea tuturor tabelor și a relațiilor existente în baza de date. Fiecare entitate este descrisă printr-o clasă *POCO* , clasă formată din proprietăți publice ce descriu coloanele entității din baza de date. Unul dintre avantajele acestui framework eliminarea codului dependent de sistemul de gestiune al bazei de date. Baza de date este construită folosind tehnica *code first*. Această tehnică implică inițial realizarea contextului și crearea claselor *POCO* ce descriu entitățile utilizate ,urmând apoi generarea bazei de date folosind aceste modele.

În cadrul acestui proiect există trei directoare importante. Primul director este directorul ce stochează migrările ce s-au efectuat asupra bazei de date. Migrarea este o clasă ce descrie modificările ce s-au efectuat de la ultima versiune. Cel mai important director din cadrul acestui proiect este directorul *Models*. Aici sunt salvate toate clasele model ce descriu entitățile bazei de date.

### III.2.4 UaicLibrary.Common

Proiect destinat împărțirii de date și setări între toate proiectele existente în cadrul aplicației web API. Principalul scop al acestui proiect este centralizarea setărilor. Framework-uri precum *Automapper*, *AutoFac*, *EntityFramework* necesită anumite setări. Aceste framework-uri sunt utilizate în toate celelalte proiecte. Pentru a nu fi nevoie să rescriem setările pentru fiecare proiect

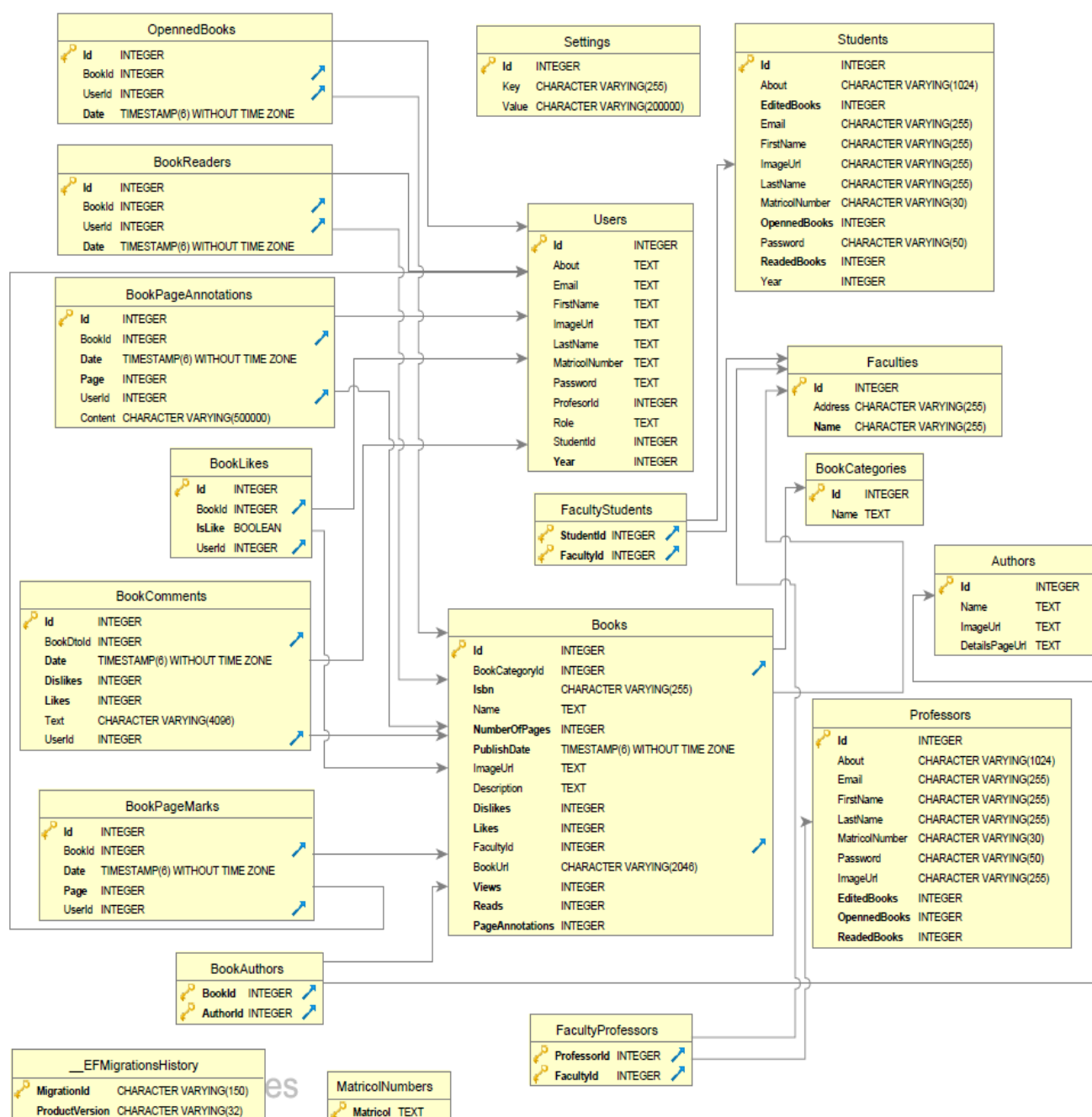


în parte , a fost dezvoltat acest proiect pentru centralizarea setărilor și a datelor partajate de proiectele componente.

### III.3 Arhitectura bazei de date

Ca și sistem de gestiune a bazei de date se utilizează PostgreSQL. Acest sistem este disponibil gratuit sub o licență open source de tip BSD. Cea mai importantă caracteristică a acestui sistem este caracterizată de suportarea a diverselor platforme hardware și a multiplelor sisteme de operare. Alegerea acestui sistem de gestiune de baze de date pentru biblioteca online fiind legată de această caracteristică. Deoarece fiecare parte din proiect prezentată până la acest moment poate rula pe orice sistem de oprere sau platformă hardware, pentru păstrarea portabilității este necesar ca și în cazul bazei de date să avem această carcateristică. De aceea PosgreSql este o soluție bună atunci cand vine vorba de baza de date , pentru această aplicație.

#### ❖ Structura bazei de date



## Capitolul IV: Ghid de utilizare

Aplicația UaicLibrary este disponibilă online prin accesarea unei adrese web obișnuite. La accesarea acestei adrese aplicația va fi încărcată în browser-ul utilizatorului. În timpul încărcării utilizatorul este întâmpinat prin afișarea paginii de încărcare a aplicației.

### Încărcarea Aplicației



După finalizarea încărcării aplicației, ca și utilizator al aplicației vei fi redirecționat către pagina principală (Home Page). Această pagină necesită ca utilizatorul să fie autentificat. Dacă aceasta este prima accesare a aplicației, fără o autentificare în trecut astfel încât browser-ul pe care îl utilizezi să conțină în memorie credențialele utilizate anterior, vei fi redirecționat către pagina de autentificare.

### Autentificarea utilizatorilor

Pagina de autentificare conține două câmpuri obligatorii (email și parolă). Aceste informații sunt informațiile completate la înregistrarea unui utilizator în aplicație. Dacă nu sunteți utilizator al acestei aplicații puteți să va înregistrați accesând pagina de înregistrare.

## Autentificare

Adresa de email

example@uaic.ro

Parolă

••••••••

TRIMITE

După completarea câmpurilor și apăsarea butonului “Trimite”, utilizatorul va fi notificat dacă operația s-a efectuat cu succes sau au apărut erori în timpul execuției. Dacă autentificare s-a realizat cu succes utilizatorul va fi redirecționat către pagina de prezentare a cărților (Books Page).

## Înregistrarea utilizatorilor

Pentru înregistrarea unui utilizator nou este necesară introducerea codului matricol al studentului care înregistrează acest cont. Numărul matricol poate fi găsit în carnetul de student. Acest lucru restrânge utilizarea bibliotecii doar în cadrul facultății. Nu se pot înregistra două cont-uri folosind același număr matricol.

Pagina de Înregistrare

### Înregistrare

☐ student  
☐ profesor

Nume

Prenume

Anul de facultate

0

Număr matricol

Adresa de email

Facultate

Alege facultățile de care aparții

Parolă

ConfirmPassword

TRIMITE

Se pot înregistra două tipuri de utilizatori: studenți și profesori

Câmpuri obligatorii:

- Nume
- Prenume
- Anul de facultate
- Numărul matricol
- Adresa de email
- Parola
- Confirmarea parolei

De asemenea se poate specifica și facultatea sau facultățile la care este înscris utilizatorul prin selectarea facultăților dintr-o listă a facultăților înregistrate în bibliotecă.

## Meniul aplicației

Meniul aplicației este prezent în partea de sus a aplicației. Opțiunile prezente în acest meniu diferă în funcție de contextul autentificării.

Atunci când utilizatorul nu este autentificat , meniu este simplu prezentând câteva opțiuni ce pot fi accesate.



Descrierea opțiunilor din meniu:

- Acasă – accesează pagina principală a aplicației
- Noutăți – deschide pagina celor mai recente cărți
- Înregistrare – accesează pagina de înregistrare
- Autentificare – accesează pagina de autentificare
- Limbă – prezintă o listă de limbi ce pot fi utilizate pentru a schimba limba aplicației

Dacă utilizatorul este autentificat în aplicație , în meniu sunt adăugate opțiuni ce pot fi accesate doar în contextul utilizatorului autentificat. Următoarele opțiuni sunt prezente în meniu doar dacă utilizatorul este autentificat iar acesta nu este administrator:

- Cărți – deschide pagina de prezentare a cărților din bibliotecă
- Autori – accesează pagina autorilor
- Facultăți – accesează pagina facultăților
- Profilul meu – pagina de vizualizare și editare a profilului utilizatorului

Dacă utilizatorul este autentificat ca și utilizator admin, din meniul principal va fi eliminate opțiunea “Profilul meu” fiind adăugată o nouă opțiune “Administrare Bibliotecă” opțiune ce accesează pagina de administrare a bibliotecii.

## Schimbarea limbii Aplicației

Limba aplicației poate fi schimbată din meniul principal acesând opțiunea “Limbă”. Vor fi afișate lista de limbi disponibile. La selectarea unei limbi aplicația va fi reîncărcată , utilizatorul fiind redirecționat către pagina principală.

## Pagina de prezentare a celor mai recente cărți

Este asemănătoare paginii de prezentare a cărților din bibliotecă, fără existența acțiunilor ce pot fi făcute asupra cărților. Această pagină nu necesită autentificare putând fi vizualizată și de către persoanele ce nu sunt înregistrate în aplicație.

Pagina prezintă o listă cu cele mai recente zece cărți ce au fost introduse în bibliotecă. Asupra cărților nu se poate efectua nici o acțiune, fiind doar prezentate utilizatorului.

## Afișarea și utilizarea unei cărți



### Curs practic de java

Aceasta carte cuprinde informatii utile in invatarea limbajului java. Este un curs cu directii spre partea practica.

Categorie: Java

Autori:  Cristian Frăsinaru

Facultate: Facultatea de Informatica

Număr de pagini: 462

Data publicării: 9/6/2006

 0

 0

1 vizualizări

0 cititori

1 notițe

 VIZUALIZEAZĂ

 DESCARCĂ

 ÎMI PLACE


 NU ÎMI PLACE

 ARATĂ COMENTARIILE

Acțiunile ce pot fi făcute asupra unei cărți se află în partea de jos a elementului de afișare a cărții.

Descrierea acțiunilor:


- Vizualizează – afișează conținutul cărții
- Descarcă – se va descărca un fișier pdf a conținutului cărții
- Îmi place - adaugă o apreciere a utilizatorului curent cărții
- Nu Îmi Place – adaugă o deapreciere a utilizatorului curent cărții
- Arată Comentariile – va afișa lista de comentarii asociate cărții





Fedeles Simona

6/25/2017, 2:29 PM

O carte excelenta !!

 0


 0





Tifui Vali

6/25/2017, 2:21 PM

Cartea este foarte interesantă

 0

 0



Tifui Vali

InsertComment

ADD

## Utilizarea și afisarea comentariilor

Fiecare comentariu poate fi apreciat sau deapreciat. Un comentariu poate fi șters doar de către utilizatorul ce a postat acel comentariu.

Pentru a posta un comentariu pentru o carte, în partea de jos a listei de comentarii se găsește o secțiune destinată acestei acțiuni.

## Filtrarea și căutarea cărților

Filtrează după:

Titlu

Selectează categoriile...

Selectează autori...

Selectează facultăți...

FILTREAZĂ

În pagina de prezentare a cărților în partea superioară a listei de cărți, se observă un bloc ce conține opțiuni de filtrare a listei.

Opțiuni de filtrare a cărților:

Titlu – dacă se specifică un titlu, se va returna cărțile ce conțin în titlu, textul precizat în cadru acestui câmp

- Categori de cărți – rezultatul filtrării va conține cărți ce aparțin doar categoriilor selectate
- Autori – vor fi afișate doar cărți ce au fost scrise de către autorii selectați
- Facultăți – permite filtrarea cărților după facultățile selectate de utilizator

## Afișarea și editarea profilului utilizatorului

Pagina ce se ocupă cu acest aspect este compusă din patru secțiuni. Prima secțiune este Situate în partea superioară a paginii. Această secțiune afișează informații despre utilizator.



Tifui Vali  
Student

0

Cărți citite

1

Cărți deschise

0

Cărți notate



### ❖ Schimbarea imaginii de profil

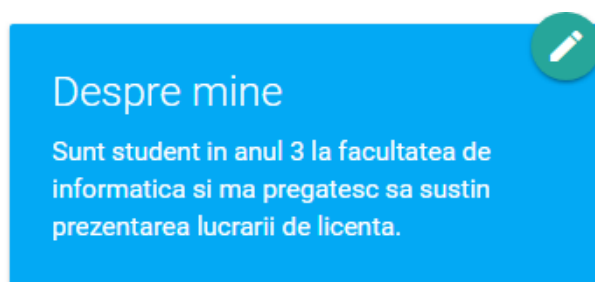
Singura acțiune disponibilă în această secțiune este schimbarea imaginii de profil a utilizatorului.

Facând click pe imagine , va apărea fereastra de încărcarea a imaginii noi.

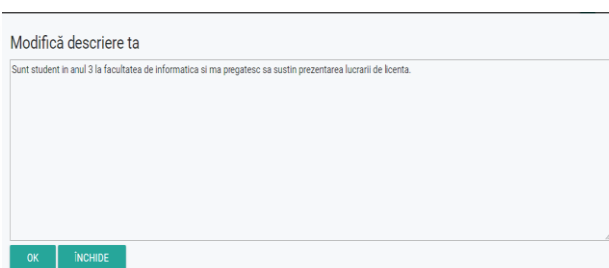
### ❖ Editarea descrierii utilizatorului

O altă secțiune prezentă în cadrul paginii de profil este secțiunea de afisare si editare a propriei descrieri.

FIGURĂ 1



FIGURĂ 2

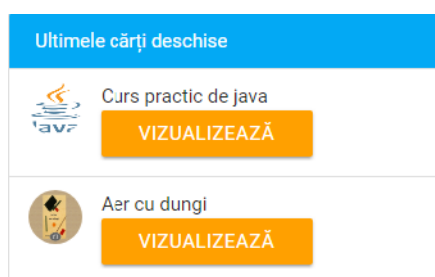


La apăsarea butonului de editare, butonul verde din figura 1, se va deschide fereastra prezentată în figura 2 , fereastră ce permite editarea descrierii.

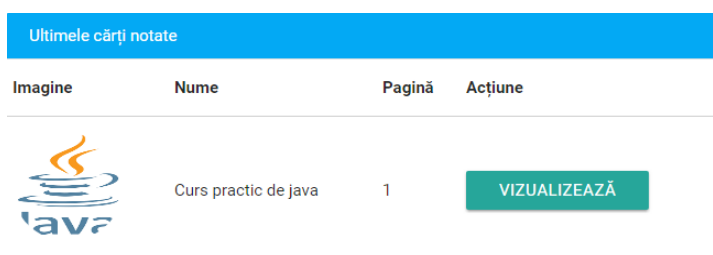
### ❖ Vizualizarea ultimelor acțiuni asupra cărților

Cea de-a treia secțiune a paginii de profil prezintă lista utimelor cărți accesate (Figura 3). Ulima secțiune afișează lista ultimelor annotări efectuate în aplicație (Figura 4).

FIGURĂ 3



FIGURĂ 4



### Vizualizarea cărților

La deschiderea unei cărți, utilizatorul este redirecționat către pagina de vizualizare a unei cărți.

Această pagină conține un element în cadrul căreia este afișat conținutul unei pagini a cărții.

Pagina cărții poate fi schimbată din partea inferioară, prin selectarea paginii dorite din lista de pagini afișate, sau prin introducerea paginii în câmpul "Pagină". Pentru a mări sau a micșora modul de vizualizare a cărții, se va introduce o nouă valoare penru scalarea dorită, completând câmpul "Scală".

Pagina de vizualizare pune la dispoziție câteva opțiuni utile ce pot fi efectuate asupra cărții current vizualizate (Figura 5)

FIGURĂ 5



Opțiunea “Memorează pagina” permite memorarea paginii curente a cărții. Această opțiune permite setarea semnelor de carte.

Opțiunea “Marchează ca și citită” poate fi utilizată de utilizatorul curent pentru a marca o carte ca și citită.

La selectarea “Deblochează notițele pentru click”, atunci cand utilizatorul efectuează click în interiorul conținutului cărții , va fi deschisă fereastra de annotarea a paginii curente.

Opțiunea “Adaugă notițe” deschide fereastra de adaugare a notelor pentru pagina currentă.

### Administrarea Bibliotecii

Pentru a avea access la pagina de administrare a bibliotecii , utilizatorul autentificat trebuie să dețină rolul de administrator. În cadrul bibliotecii se poate face managementul cărților, a categoriilor de carte , a autorilor și a facultăților înscrise în bibliotecă. Pentru fiecare resursă ce necesită management, este creată o pagină separată destinată acestui lucru.

#### ❖ Managementul categoriilor de cărți

CARTE      AUTOR      FACULTATE      CATEGORIE CARTE      SCHIMBĂ CONȚINUTUL PAGINII P...

### Adauga o categorie de carte

Nume

---

Detalii

---

TRIMITE

Prin accesarea tab-ului Categorie Carte, va fi afișat un formular de adaugare a unei noi categorii.

Pentru adaugarea unei categorii se vor completa numele și detaliile categoriei.

După adaugarea categoriei în partea din dreapta a ecranului v-a apărea butonul “Șterge categoria” ce poate fi utilizat pentru a elimina categoria salvată.

#### ❖ Managementul facultăților

Asemenea adăugării unei categorii , adaugarea unei facultăți presupune completarea a două câmpuri specifice unei facultăți: nume și adresa facultății.



Pentru a edita o facultate , se accesează pagina de vizualizare a facultăților unde pentru fiecare facultate există butonul edit. La apăsarea acestui buton ve-ți fi redirectionați către pagina de editare a unei facultăți (Figura 6).

FIGURĂ 6

### Editează facultatea

ȘTERGE FACULTATEA

Nume

Facultatea de Educatie Fizica si Sport

Adresă

Str. Toma Cozma 3, RO-700554 – Iași

TRIMITE

### ❖ Managementul Autorilor

CARTE

AUTOR

FACULTATE

CATEGORIE CARTE

SCHIMBĂ CONȚINUTUL PAGINII ...

#### Adaugă autor

Imaginea autorului



SCHIMBĂ IMAGINEA

Nume

Pagina web a autorului

TRIMITE

Se accesează tabul Autor , prezent în pagina de administrare a bibliotecii. Pentru a adauga un nou autor, sunt necesare completarea următoarelor câmpuri:

- Imaginea Autoului
- Nume
- Pagina web a autorului

In pagina de vizualizare a autorilor în cadrul listei de autori, la acționarea butonului edit ve-ți fi redirectionat către pagina de editare a unui autor.

## ❖ Managementul Cărților

Ca și bibliotecă este necesar o modalitate de a face managementul cărților. Pentru a adăga o carte în bibliotecă trebuie luat în vedere următoarele aspecte:

- Autorul sau autorii cărții sunt înregistrați în cadrul bibliotecii
- Conținutul cărții ce se dorește a fi publicată este un fișier în format pdf
- Categoria din care face parte cartea este existentă în cadrul bibliotecii
- Dacă cartea aparține unei facultăți, este necesar ca aceasta să fie înscrisă în bibliotecă

După asigurarea situațiilor menționate anterior, se vor completa câmpurile necesare adăgării unei noi cărți.

Descrierea câmpurilor necesare adăugării unei cărți:

- Isbn – reprezintă codul unic de identificare al cărții (obligatoriu)
- Nume – numele cărții (obligatoriu)
- Categorie – se va selecta categoria din care face parte cartea (obligatoriu)
- Data Publicării – data publicării cărții de către editură (obligatoriu)
- Autori – se vor selecta autorii cărții (obligatoriu)
- Descriere (opțional)
- Numărul de pagini (obligatoriu)
- Fișierul cărții (obligatoriu) (formatul suportat: pdf)
- Facultate – desemnează facultatea de care aparține cartea (optional)

## Rezumat

Aplicația Biblioteca Uaic este alcătuită din 3 mari componente. Aplicația web SPA, ce prezintă interfața de comunicare cu utilizatorul, urmand apoi aplicația web api, unde are loc logica și procesările de date necesare, și bineînțeles locul stocării pentru datele aplicației și anume baza de date. Rolul esențial al aplicației este de a eficientiza procesul de dezvoltare a studenților din cadrul universității, prin oferirea unei platforme de documentare bine structurată și ușor de utilizat. Un alt rol important al aplicației îl reprezintă crearea unui mediu în care studenții pot împărtăși opinii despre documentele utilizate în procesul de studiu, prin introducerea unui nivel de socializare. Al treilea motiv prin care această aplicație își dovedește utilitatea este introducerea instrumentului de adăugare a notițelor asupra unei cărți.

## Concluziile lucrării

Lucrarea de față are ca scop prezentarea unui aplicații utile în realizarea procesului de dezvoltare a studenților. Deoarece era în care trăim este într-o continuă dezvoltare, iar tehnologia este prezentă peste tot, această aplicație propune schimbarea modului în care utilizăm în acest moment biblioteca din cadrul facultății și a universității. Timpul este prețios

pentru toată lumea. Nimănui nu îi face plăcere să aștepte la coadă sau să se deplaseze pentru a putea avea acces la informație. Acest aspect poate fi rezolvat prin înlocuirea bibliotecii clasice cu o bibliotecă online. Avantajele pe care le oferă o bibliotecă online în comparație cu o bibliotecă clasică sunt foarte multe. Începând cu oferirea accesului din orice colț al lumii și terminând cu viteza de răspuns și căutare a informațiilor, bibliotecile online devin mult mai folosite într-o lume în care timpul este o comoară neprețuită.

Ca și structură a aplicației, aceasta este o aplicație web ce poate fi accesată de orice utilizator ce deține un cont de acces în aplicație. Acest cont de acces este necesar pentru a realiza securitatea aplicației precum și a drepturilor de autor.

În acest moment utilizatorii acestei aplicații dispun de următoarele beneficii:

- Acces la o serie de cărți ce sunt disponibile atât pentru vizualizare cât și pentru descărcare
- Mechanism complex de căutare și căutare a cărților
- Posibilitatea interacțiunii cu ceilalți utilizatori prin intermediul componentei sociale a aplicației
- Posibilitatea adăugării semnelor de carte
- Posibilitatea adăugării de notițe pentru fiecare pagină a unei cărți
- Control asupra propriului profil
- Afișarea de informații cu caracter statistic
- Managementul resurselor de către un administrator

În decursul dezvoltării au fost întâlnite diverse probleme de dezvoltare. Inițial modelul de dezvoltare abordat a fost Test-Driven-Development, un mediu bazat pe dezvoltarea inițială a testelor unitare, apoi implementarea efectivă este realizată pentru a trece testele cu succes. Această metodologie face ca dezvoltarea să dureze mult mai mult, dar calitatea produsului este la un standard înalt. Din cauza timpului limitat a fost nevoie să se renunțe la această metodologie, dezvoltarea aplicației făcându-se în stil classic.

Pe viitor aplicația poate fi extinsă cu noi opțiuni interesante. Opțiuni ce au fost vizate pentru dezvoltare inițial dar nu au fost finalizate. Se poate extinde funcționalitățile utilizatorului profesor pentru a suporta încărcarea de cursuri. De asemenea se poate adăuga un mecanism de împărtășire a notițelor. Bineînțeles se pot face îmbunătățiri legate de interfața grafică precum și optimizări în ceea ce privește managementul resurselor. În acest moment în aplicație există o secțiune de administrare destinată administratorului pentru a face managementul cărților.

Această secțiune poate fi înlocuită cu o aplicație dedicată acestui scop. Pentru a mări gradul de utilitate a aplicației este necesar construirea unui client mobil care să comunice cu web api-ul, oferind astfel o modalitate mult mai portabilă de acces în cadrul aplicației.