UNIVERSITATEA “ALEXANDRU IOAN CUZA” DIN IAȘI

**FACULTATEA DE INFORMATICĂ**



LUCRARE DE LICENȚĂ

**Fii Online**

propusă de

***Ovidiu-Ștefan Tudosă***

**Sesiunea**: *iulie, 2018*

**Coordonator științific**

**Decan Conf. dr. Adrian Iftene**

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

**FACULTATEA DE INFORMATICĂ**

**Fii Online**

***Ovidiu-Ștefan Tudosă***

**Sesiunea**: *iulie, 2018*

**Coordonator științific**

***Decan Conf. dr. Adrian Iftene***

DECLARAŢIE PRIVIND ORIGINALITATE ŞI RESPECTAREA DREPTURILOR DE AUTOR

Prin prezenta declar că Lucrarea de licenţă cu titlul „*Fii Online*” este scrisă de mine şi nu a mai fost prezentată niciodată la o altă facultate sau instituţie de învăţământ superior din ţară 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 şi în traducere proprie din altă limbă, sunt scrise între ghilimele şi deţin referinţa precisă a sursei;

- reformularea în cuvinte proprii a textelor scrise de către alţi autori deţine referinţa precisă;

- codul sursă, imaginile etc. preluate din proiecte open-source sau alte surse sunt utilizate cu respectarea drepturilor de autor şi deţin referinţe precise;

- rezumarea ideilor altor autori precizează referinţa precisă la textul original.

Iaşi,

Absolvent *Ovidiu-Ștefan Tudosă*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

DECLARAŢIE DE CONSIMŢĂMÂNT

Prin prezenta declar că sunt de acord ca Lucrarea de licență cu titlul „Fii Online”, codul sursă al programelor şi celelalte conţinuturi (grafice, multimedia, date de test etc.) care însoţesc această lucrare să fie utilizate în cadrul Facultăţii de Informatică.

De asemenea, sunt de acord ca Facultatea de Informatică de la Universitatea „Alexandru Ioan Cuza” din Iași, să utilizeze, modifice, reproducă şi să distribuie în scopuri necomerciale programele-calculator, format executabil şi sursă, realizate de mine în cadrul prezentei lucrări de licenţă.

Iaşi,

Absolvent *Ovidiu-Ștefan Tudosă*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ACORD PRIVIND PROPRIETATEA DREPTULUI DE AUTOR

Facultatea de Informatică este de acord ca drepturile de autor asupra programelor-calculator, în format executabil şi sursă, să aparţină autorului prezentei lucrări, *Ovidiu-Ștefan Tudosă*.

Încheierea acestui acord este necesară din următoarele motive:

Iaşi,

Decan Conf. dr. *AdrianI ftene* Absolvent *Ovidiu-Ștefan Tudosă*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Cuprins

[Introducere 7](#_Toc515635960)

[1. Sisteme de management al învățării 8](#_Toc515635961)

[1.1 Moodle 8](#_Toc515635962)

[1.2 Chamilo 9](#_Toc515635963)

[1.3 SWAD (Shared Workspace At a Distance) 10](#_Toc515635964)

[1.4 Canvas 11](#_Toc515635965)

[2.5 TalentLMS 12](#_Toc515635966)

[1.6 Concluzie 13](#_Toc515635967)

[2. Tehnologii utilizate 14](#_Toc515635968)

[2.1 Microsoft ASP.NET Core 2.0 14](#_Toc515635969)

[2.2 Entity Framework Core 2.0 15](#_Toc515635970)

[2.2.1 Entity Framework Migrations 15](#_Toc515635971)

[2.3 Json Web Tokens 16](#_Toc515635972)

[2.4 Angular 5 18](#_Toc515635973)

[2.4.1 Service Workers 18](#_Toc515635974)

[2.5 Materialize 22](#_Toc515635975)

[3. Fii Online 23](#_Toc515635976)

[3.1 Arhitectură 24](#_Toc515635977)

[3.1.1 Domain Layer 25](#_Toc515635978)

[3.1.2 Repository Layer 26](#_Toc515635979)

[3.1.3 Service Layer 27](#_Toc515635980)

[3.1.4 UI Layer 27](#_Toc515635981)

[3.2 Baza de date 29](#_Toc515635982)

[3.3 Detalii de implementare 30](#_Toc515635983)

[3.3.1 Aplicație ASP.NET Core 30](#_Toc515635984)

[3.3.2 Crearea tabelelor Code First 31](#_Toc515635985)

[3.3.3 Mecanismul de logare 35](#_Toc515635986)

[3.3.4 JWT 39](#_Toc515635987)

[4. Concluzie 42](#_Toc515635988)

[References 43](#_Toc515635989)

# Introducere

Cursurile facultăților sunt în general cursuri bogate în informații. Oricât de atent ar fi un student și oricât de multe notițe ar lua, este inevitabil ca uneori să se afle în situația în care ar trebui să revadă suportul de curs, întrucât e posibil să fi omis anumite informații. În zilele noastre, majoritatea universităților din lume au găsit o soluție de a pune la dispoziție studenților cursurile în format electronic, fie pe pagina facultății, fie utilizând o platformă online care să permită descărcarea cursurilor.

La momentul actual, în cadrul Facultății de Informatică Iași, fiecare profesor are o pagină web personală în care își adaugă cursurile, ceea ce înseamnă ca el trebuie să se ocupe de design-ul paginii și de mecanismul de încărcare a documentelor. Totodată studentul este nevoit să caute mereu pagina profesorului dacă dorește să acceseze cursurile.

Platforma “*Fii Online*” vine în ajutor atât studenților cât și profesorilor Facultății de Informatică din Iași, oferind o modalitate de centralizare a tuturor cursurilor facultății.

Profesorii au la dispoziție o interfață intuitivă prin intermediul căreia pot crea lecții, seminare sau teme pentru cursurile lor, pot adăuga resurse sau observații care să vină în ajutorul studenților și pot în orice moment să editeze sau chiar să șteargă materialele create anterior. La rândul lor, studenții au acces la aceste materiale, doar accesând platforma, fără a mai fi nevoiți să caute pagina web a fiecărui profesor.

În documentația de față găsim următoarele 4 capitole: „Sisteme de management al învățării” care prezintă soluții alternative ale acestei aplicații (alte aplicații aflate pe piață), “Tehnologii utilizate”, un capitol care prezintă pe scurt stack-ul de tehnologii utilizate, “Fii Online” capitolul care prezintă detalii despre arhitectura, baza de date și implementarea aplicației, iar în ultimul capitol se regăsește concluzia.

# 1. Sisteme de management al învățării

## 1.1 Moodle

Este un sistem de management al învățării open source. A fost dezvoltat inițial de către Martin Dougiamas, în 2002, cu scopul de a ajuta profesorii să creeze cursuri online, având ca scop principal facilitarea lucrului în echipă în crearea de cursuri.[1] Astăzi, Moodle este unul dintre liderii mediilor de învățare virtuală folosit de școli, universități, companii și guverne pentru a crea medii virtuale de învățare, punând la dispoziție utilizatorilor, pe lângă posibilitatea accesării de cursuri, diverse tool-uri cum ar fi, posibilitatea de a avea un calendar cu activități sau un sistem de notare.

Aplicația este gratuită pentru grupuri mai mici de 50 de persoane, iar acestea au alocat maxim 200 MB pentru upload de fișiere. Pentru a folosi aplicația la o scară mai mare Moodle pune la dispoziție un număr de pachete care să se plieze pe nevoile clientului. Pe lângă accesul mai multor persoane și o zonă mai mare de stocare, aceste pachete oferă și accesul la diverse plug-in-uri pentru a crește utilitatea aplicației.

Figure 1: Moodle

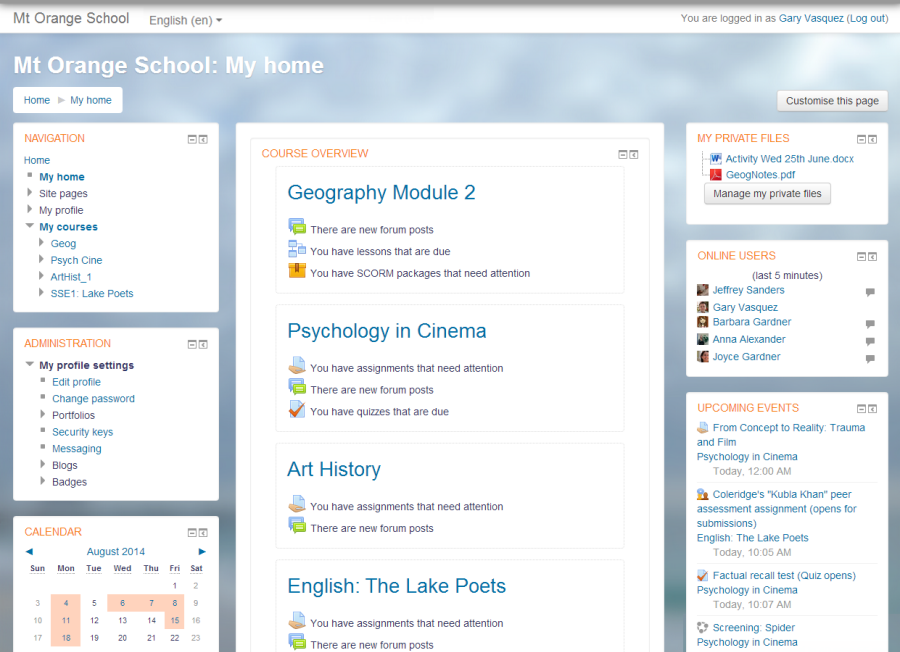
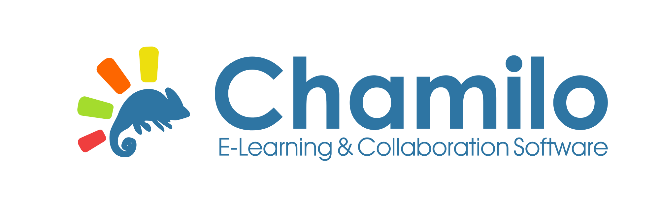


Figura 1: Moodle

## 1.2 Chamilo

Chamilo este o platformă de e-learning și un sistem de administrare a conținutului menit să îmbunătățească accesul la educație și cunoștințe la nivel mondial. Este susținut de Chamilo Association care au ca scop promovarea software-ului și colaborarea cu alți dezvoltatori pentru a putea oferi o rețea de servicii prin intermediul platformei.[2]

Platforma pune la dispoziție profesorilor posibilitatea de a crea lecții în care pot adăuga diverse resurse (fișiere audio, video) , pot crea teste și pot urmări evoluția studenților prin intermediul unor statistici bazate pe rezultatele acestora la testele susținute. Totodată prin intermediul platformei profesorii pot face anunțuri pentru studenții care urmăresc cursul sau pot să comunice direct cu studenții prin intermediul canalului de comunicare pus la dispoziție de către aplicație.

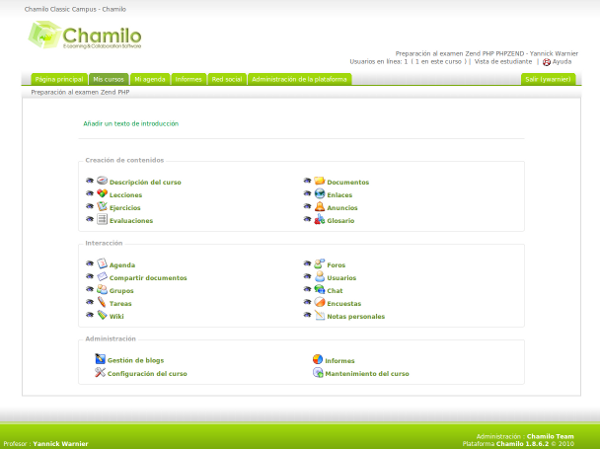


Figure 2: Chamilo

Figura 2: Chamilo

## 1.3 SWAD (Shared Workspace At a Distance)

 SWAD este un software gratuit și rapid, un sistem de management al învățării eficient pentru învățarea și predarea la distanță. A fost dezvoltat și folosit la Universitatea din Granada (Spania) din 1999. Din anul 2012 este folosit în cadrul mai multor universități și a fost mutat în cloud.

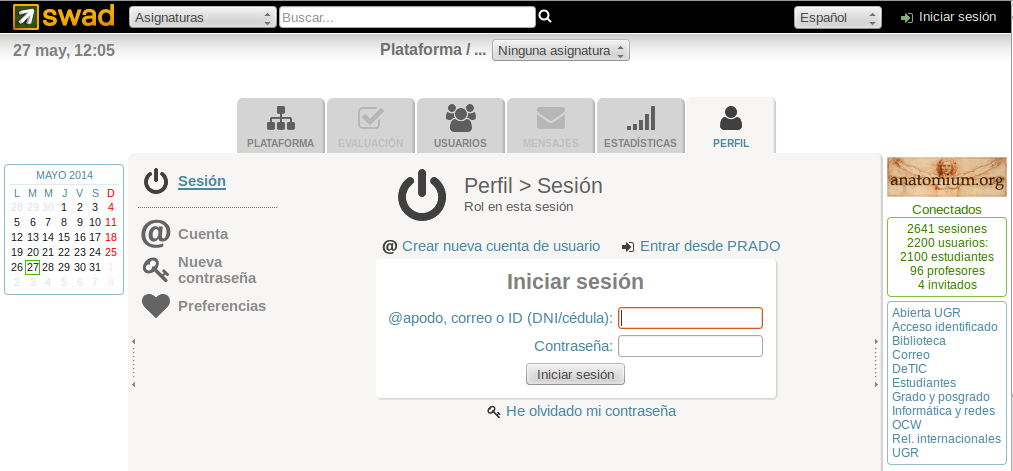
 SWAD integrează într-un tool web mai multe funcționalități pentru a ajuta cu învățarea, predarea și managementul datelor studenților și profesorilor. Printre acestea se numără, accesul la cursuri, zona de stocare a documentelor (notițe, prezentări, teme), liste de studenți și profesori, forum-uri, teste interactive și carnetul de note al fiecărui student. O parte din funcționalitatea platformei este disponibilă si prin intermediul unei aplicații mobile.[3]

Figure 3: SWAD

Figura 3: SWAD

## 1.4 Canvas



Canvas este un alt LMS (Learning Management System) care se remarcă prin faptul că tool-urile pe care le pune la dispoziție utilizatorilor sunt tool-urile cu adevărat folositoare, lucru ce face aplicația sa fie adoptată mult mai ușor de către clienți, spre deosebire de aplicațiile competitoare. Canvas oferă profesorilor o funcționalitatea numită “SpeedGrader”, care permite înregistrarea audio/video a profesorului în timp ce corectează temele studenților, ascunde numele studenților astfel încât corectura să fie una obiectivă, profesorul notează studentul selectând criteriile pe care studentul le-a respectat (criterii dintr-o lista de criterii de notare prestabilite de profesor) si totodată oferă posibilitatea ca profesorul să ofere feedback studentului în legătură cu lucrarea sa.

Canvas este un sistem open source și permite profesorilor să creeze lecții fără nici un cost. Cu toate acestea, percepe anumite taxe pentru controlul administrativ, suport și training-uri și uneori pentru suportul pentru integrare, de care este nevoie în sistemele complexe ale colegiilor și ale universităților.

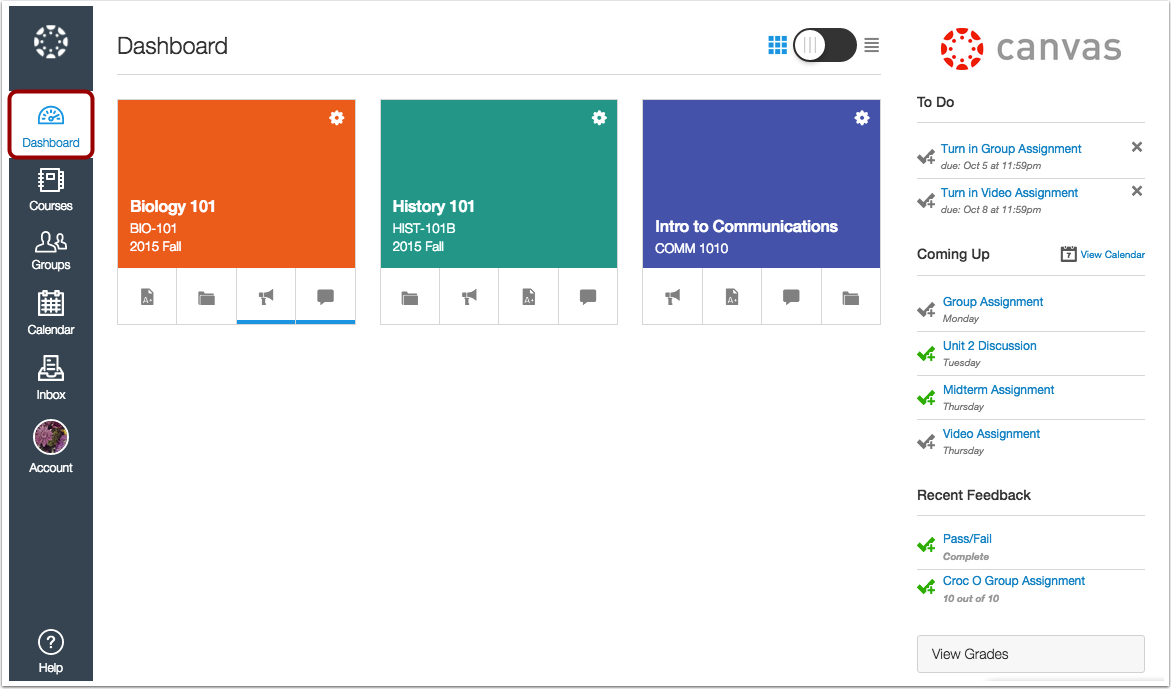
 Această platformă se mai remarcă prin faptul că are o viteză și o scalabilitate foarte bună întrucât este hostată pe serverele AWS (Amazon Web Services). [4]

Figure 4: Canvas

Figura 4: Canvas

## 2.5 TalentLMS



TalentLMS este un sistem de management al învățării foarte ușor de folosit, gândit încă de la început cu un design responsive pentru a putea fi folosit atât de pe desktop/laptop cât și de pe device-uri mobile. Această platformă se caracterizează prin faptul că utilizatorii pot crea cursuri în câteva minute, pot adăuga video-uri sau prezentări mai vechi sau pot adăuga diverse materiale existente întrucât au integrare cu numeroase platforme multimedia ca: Youtube, Vimeo, Scribd, LinkedIn etc. Platforma este foarte customizabilă, oferă utilizatorilor posibilitatea de a adăuga propriul lor logo, de a crea o temă proprie sau să folosesască domeniul propriu pentru a accesa platforma. Pe lângă aceste lucruri, aplicația oferă posibilitatea video-conferințelor, vânzării de cursuri și activării notificărilor. [5]

TalentLMS este o aplicație open source gratuită, cu toate acestea percepe anumite taxe în funcție de dimensiunea grupurilor de persoane. Pentru un grup mai mare de 1000 de persoane (cum este cazul facultăților) aceasta percepe o taxă lunară de 349$. [6]

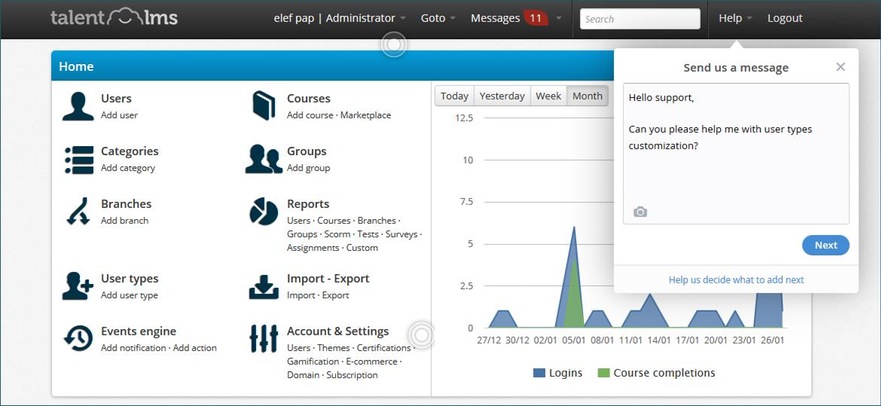


Figure 5: TalenLMS

Figura 5: TalentLMS

## 1.6 Concluzie

În momentul de față există foarte multe aplicații web sau cloud care oferă soluții pentru crearea unui mediu virtual de învățare, cu ajutorul numeroaselor tool-uri pe care le pun la dispoziția clienților. Deși majoritatea aplicațiilor reușesc să vină în ajutorul profesorilor și studenților, acest ajutor nu vine gratuit, și de cele mai multe ori universitățile sunt nevoite să plătească sume considerabile pentru a beneficia de aceste servicii.

Platforma “Fii Online” propune o soluție minimalistă, dar construită după nevoile Facultății de Informatică Iași, pentru a ușura munca profesorilor și a facilita experiența studenților. Deși platforma are mai puține funcționalități decât celelalte aplicații existente, trebuie conștientizat faptul ca multe din tool-urile oferite de acestea nu sunt cu adevărat necesare (în contextul Facultății de Informatică) și că având prea multe funcționalități uneori poate fi în dezavantajul utilizatorului. Totodată “Fii Online” oferă facultății o soluție open source, cu adevărat gratuită care păstrează în design-ul ei identitatea instituției în ajutorul căreia vine.

# 2. Tehnologii utilizate

## 2.1 Microsoft ASP.NET Core 2.0

ASP.NET Core 2.0 este un framework cross-platform, open source, high-performance pentru dezvoltarea de aplicații moderne, cloud-based, internet-connected, dezvoltat de Microsoft și comunitatea .NET. Factorul principal care îl deosebește de .NET Framework este faptul că este cross-platform. Acest lucru înseamnă că este suportat de mai multe sisteme de opera precum: Windows, macOs sau Linux și poate fi folosit de pe orice device. [7]

ASP.NET Core 2.0 poate fi folosit atât pentru aplicații .NET Framework cât și pentru aplicații .NET Core. .NET Framework este folosit pentru a crea aplicații desktop pentru Windows și aplicații enterpirse de mari dimensiuni. Este preferat în cazurile în care dezvoltatorii folosesc tehnologii, librării third-party sau pachete NuGet care nu sunt încă accesibile în .NET Core sau când platforma nu suportă .NET Core (deși este cross-platform nu toate platformele Microsoft third-party suportă .NET Core, de exemplu unele servicii Azure). [8]

.NET Core este un framework open-source, care a căpătat popularitate în ultimii ani fiind cross-platform (Visual Studio Code și CLI pot fi folosite pe Windows, Linux si macOS ). Este preferat atunci când se lucrează cu Microservicii (.NET Core este considerat alternativa modernă pentru NodeJS pentru lucrul cu Microservicii), când se folosesc containere Docker (containerele și Microserviciile sunt folosite în general împreună), când se dorește performanță crescută și scalabilitate sau când se dorește control de la linia de comandă (CLI, unii dezvoltatori preferă să lucreze cu editoare lightweight și cu linia de comandă). [8]

Razor Pages este una dintre cele mai mari funcționalități aduse de ASP.NET Core 2.0 acestea implementând o noua pardigmă, MVVM (Model-View-ViewModel). O pagină Razor este similară componentei view din ASP.NET MVC, diferența fiind că codul pentru model și controller sunt incluse în aceeași pagină. Acest lucru oferă posibilitatea de two-way data binding și o experiență de devlopment simplificată. [9]

## 2.2 Entity Framework Core 2.0

Entity Framework este un ORM (object-relational mapper) framework open source, menit să permită utilizarea datelor relaționale prin maparea lor la obiecte specifice din domeniu. Astfel se elimină o mare parte din codul necesar accesării bazei de date pentru a aduce datele.

Entity Framework poate fi folosit în trei moduri:

* Baza de date relațională este deja existentă, caz în care Entity Framework va genera clasele corespunzătoare tabelelor
* Code First, clasele entităților sunt create prima dată, apoi Entity Framework va genera baza de date și tabelele
* Schema bazei de date este creată cu ajutorul unui designer vizual, iar EF va crea atât clasele entităților cât și baza de date cu tabele. [10]

Entity Framework Core 2.0 targhetează .NET Standard 2.0, ceea ce înseamnă că poate lucra cu .NET Core 2.0, .NET Framework 4.6.1 sau cu alte biblioteci implementate de .NET Standard 2.0.

## 2.2.1 Entity Framework Migrations

În timpul dezvoltării unei aplicații pot apărea situații (omiterea unui detaliu de implementare, nevoia adăugării unei noi funcționalități) care impun schimbarea modelului, ceea ce înseamnă că și baza de date trebuie modificată.

Migrations oferă o modalitate de a modifica baza de date pentru a fi sincronizată cu modelul, în timp ce datele existente sunt prezervate. Pentru a folosi EF Migrations este necesară rularea a doua comenzi:

* Add-Migration <Nume> - generează un fișier cu comenzi pentru următoarea migrare
* Update DataBase – aplică modificările din migrare peste baza de date.

## 2.3 Json Web Tokens

Un Json Web Token (JWT) este un open standard care permite transmiterea de date între părțile impicate în format JSON, într-un mod compact și securizat. Sunt folosite în general pentru autentificare și pentru schimbarea de informații, întrucât datele transmise între sursă și receptor sunt semnate digital și pot fi verificate ușor. [11]

JWT-urile sunt structurate în trei secțiuni:

* Header: un obiect JSON care conține metadate despre JWT și algoritmul de hashing folosit pentru criptarea datelor.

Figura 6: JWT Header [20]

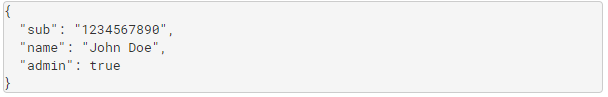
*  Payload: un alt obiect JSON care conține datele propriu zise, acestea sunt scrise în claim-uri care reprezintă trăsături, de obicei ale utilizatorului.

Figura 7: JWT Payload [20]

* Semnătură: acestă secțiune permite verificarea integrității datelor, ea fiind semnătura digitală bazată pe secțiunile anterioare. [11]

Figura 8: JWT Signiture [20]

Cele 3 secțiuni sunt combinate într-un șir (în baza 64), separate prin “.” pentru ca datele să poată fi trimise ușor prin mediile bazate pe HTTP. Când este folosit în autentificare, tehnologia JWT permite clienților sa stocheze datele de sesiune și sa trimită token-ul către sever atunci când încearcă să acceseze resurse protejate. De obicei, token-ul este trimis în header-ul HTTP Authorization, folosind schema Bearer și conține toate informațiile de care este nevoie pentru a permite sau nu accesul la o resursă 

Figure 6: JWT Token [20]

Figura 9: JWT Token [20]

## 2.4 Angular 5

Angular este un framework JavaScript dezvoltat și menținut de Google, folosit frecvent pentru aplicații web, mobile sau desktop. Angular este o platformă care combină template-uri decarative, dependecy injection, end to end tooling și integrated best practices pentru a rezolva provocările din development. [12]

Angular este o librărie complexă de aceea este preferată în cazul proiectelor non-triviale care se bazează pe formulare sau pe lucrul cu date. Totodată este preferat în cazul aplicațiilor care trebuie sa ruleze în mai multe medii de development. Pentru aplicații care trebuie să ruleze atât pe Windows cât și pe macOS se poate folosi Angular cu Electron, iar în cazul aplicațiilor care trebui să ruleze si pe web și pe mobile se pot folosi tehnologii ca NativeScript. [13]

Angular 5 a fost anunțat în noiembrie 2017 și are scopul de a face Angular mai mic și mai rapid față de versiunile anterioare. Angular 5 aduce în plus: standardizarea pipe-urilor number, date și currency, îmbunătățirea suportului pentru decorator, optimizarea build-ului, un compilator mai rapid, mai multe opțiuni în validarea form-urilor și un suport mai bun pentru Service Workers pentru realizarea de PWA (Progresive-Web-Apps). [12]

### 2.4.1 Service Workers

Văzut cât mai simplu, un service worker este un script care rulează în browser, separat de pagina web, oferind suport pentru funcționalități care nu au nevoie de o pagină web sau interacțiune cu utilizatorul. În prezent, service worker-ii includ funcționalități ca: push notification sau background sync. În viitor, aceștia ar putea suporta funcționalități ca periodic sync sau geofencing. [14]

Funcționalitatea principală pe care am folosit-o în cadrul aplicației “ Fii Online” este interceptarea și gestionarea cererilor de pe rețea.

Un service worker are un ciclu de viață separat de cel al paginii web. Pentru a instala un service worker pe un website, acesta trebuie înregistrat, lucru care se face în pagina de JavaScript. Înregistrarea unui service worker va face ca browser-ul să pornească un pas de instalare în background. [14]

În general în timpul instalării, dorim să adăugăm în cache o parte din resursele statice. Dacă toate resursele sunt adăugate în cache cu succes, atunci service worker-ul este instalat, altfel pasul de instalare eșuează și service worker-ul nu va fi activ. Dacă instalarea a eșuat, se va încerca din nou la următoarea accesare a site-ului web. [14]

După pasul de activare, service worker-ul va controla paginile pentru care utilizatorul face request. O dată ce un service worker este în control, acesta poate fi în una din două stări, fie va fi în starea “terminated” pentru a elibera memorie, fie va gestiona requesturile de pe rețea. [14]

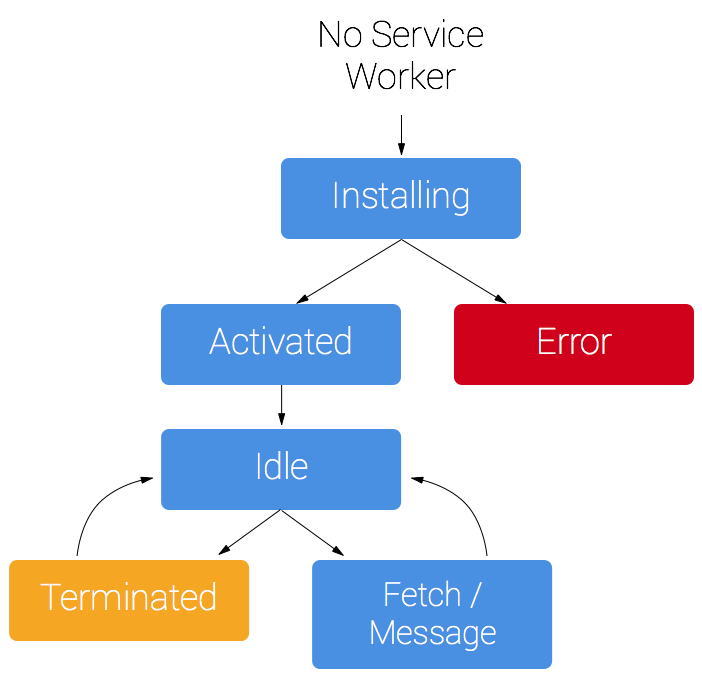


Figura 10: Ciclul de viață a unui service worker [14]

Aplicațiile Angular, fiind one-page-applications, beneficiază în mode deosebit de avantajele pe care un service worker le oferă. Începând cu versiunea 5.0.0, Angular vine cu o implementare pentru service workeri. Developer-ii care folosesc Angular pot profita de avantajele pe care un service worker le aduce, fără a fi nevoiți să folosească API-uri low-level. [15]

Service Worker-ul creat de Angular are scopul de a optimiza experiența utilizatorilor când rețeaua nu funcționează în parametri normali (sau chiar deloc) și în același timp minimizează riscul de a servi conținut învechit. [15]

Service worker-ul creat de Angular urmărește următoarele lucruri:

* Caching-ul unei aplicații să fie similar ca instalarea unei aplicații native, aplicația fiind cach-uită unitar, iar fișierele reactualizate odată
* O aplicație continuă să ruleze cu aceeași versiune de fișiere, nu va primi deodată fișiere memorate (cached) de la o versiune mai nouă, întrucât e posibil să fie incompatibile
* Când un utilizator dă refresh aplicației, acesta va primi ultima versiune memorată complet
* Update-urile au loc în background, după ce schimbările sunt publicate. Ultima versiune este servită până când update-ul este instalat
* Service worker-ul conservă bandwidth-ul când este posibil, resursele fiind descărcate doar dacă au suferit modificări. [15]

Pentru ca lucrurile menționate să aibă loc, service worker-ul încarcă un fișier manifest de la server. Acest fișier descrie care sunt fișierele care trebuie memorate și conține hash-uri pentru conținutul fiecărui fișier. Când un update este publicat, conținutul mainifestului se schimbă, informând service worker-ul că a apărut o nouă versiune a aplicației. Manifestul este generat de un fisier numit ngsw-config.json. [15]

|  |
| --- |
| {  **"index"**: **"/index.html"**,  **"assetGroups"**: [{  **"name"**: **"app"**,  **"installMode"**: **"prefetch"**,  **"resources"**: {  **"files"**: [  **"/favicon.ico"**,  **"/index.html"** ],  **"versionedFiles"**: [  **"/\*.bundle.css"**,  **"/\*.bundle.js"**,  **"/\*.chunk.js"** ]  }  }, {  **"name"**: **"assets"**,  **"installMode"**: **"lazy"**,  **"updateMode"**: **"prefetch"**,  **"resources"**: {  **"files"**: [  **"/assets/\*\*"** ],  **"urls"**: [**"http://fonts.googleapis,com./\*\*"**]  }  }] |

Figura 11: Fișierul ngsw-config.json din FiiOnline

Fișierul de configurare prezentat mai sus reprezintă un JSON (Java Script Object Notation). Pe primul nivel al obiectului se găsesc două atribute, index care indică spre fișierul index.html (entry point-ul aplicației) și assetGroup care conține obiectul de configurare pentru memorarea resurselor statice.

AssetGroup este alcătuit din două obiecte, app și assets. Obiectul app are modul de instalare „prefetch” ceea ce înseamnă că servie worker-ul va adăuga în cache toate elementele menționate în resources. Al doilea mod de instalare („lazy”) se regăsește în obiectul assets, acesta indicând service worker-ului să adauge resursele menționate în „resources” doar adunci când se face request pentru ele.

Putem extinde configurarea și să memorăm resurse de la api-uri externe. Pe același nivel cu assetGroups trebuie adăugat un nou obiect numit dataGroup.

|  |
| --- |
| **"dataGroups"**: [  {  **"name"**: **"fii-online-api"**,  **"urls"**:[**"http://localhost:63944/\*\*"**],  **"cacheConfig"**: {  **"strategy"**: **"freshness"**,  **"maxSize"**: 100,  **"maxAge"**: **"365d"**,  **"timeout"**: **"5s"** }  },  {  **"name"**: **"users-pictures"**,  **"urls"**:[**"https://firebasestorage.googleapis.com/\*\*"**],  **"cacheConfig"**: {  **"strategy"**: **"freshness"**,  **"maxSize"**: 20,  **"maxAge"**: **"7d"**,  **"timeout"**: **"5s"** }  } ] |

La rândul său obiectul dataGroup conține alte două obiecte, fii-online-api și users-pictures. Printre atributele obiectelor se regăsesc „urls” care reprezintă lista de pattern-uri de url-uri al căror răspuns dorim sa îl memorăm (pentru primul obiect avem o expresie regulată care spune ca dorim să memorăm răspunsul de la toate url-urile care încep cu „http://localhost:63994/”).

CacheConfig este și el un obiect de configurare, atributul strategy specifică metoda de caching care poate fi de două tipuri:

* Performance – această metodă specifică service worker-ului ca atunci când primește un request să caute mai întâi în cache răspunsul și doar dacă nu îl găsește în cache să apeleze la rețea. Această strategie este ideală pentru resursele care nu se schimbă frecvent, întrucât scade considerabil timpul de încărcare a paginii.
* Freshenss – prin această metodă specificăm service worker-ului să ia răspunsul de pe rețea și dacă primește timeout de la rețea să caute în cache răspunsul. Această strategie este folosită pentru resursele care se schimbă frecvent.

Pe lângă „strategy” cacheConfig mai conține „maxSize” care menționează câte răspunsuri trebuie ținute în cache (în momentul în care maxSize este depășit se șterge cel mai vechi răspuns memorat), „maxAge” face referire la perioada de timp în care o resursă este considerată validă (după ce expiră această perioadă de timp, resursa nu mai este considerată validă și este evacuată ) și „timeout” care indică service worker-ului cât timp să aștepte răspunsul de la rețea până când să folosească cache-ul în locul ei.

## 2.5 Materialize

Materialize este o bibliotecă pentru UI (User-Interface) creată cu CSS și JavaScript. Componentele din Materialize ajută la dezvoltarea de aplicații web atractive, consistente și funcționale în timp ce aderă la principii de web design moderne ca: portabilitatea între browsere, independența între dispozitive, etc. Materialize este inspirat de Google Material Design și ajută la crearea rapidă de website-uri responsive și unitare.

# 3. Fii Online

Proiectul constă în implementarea unui PWA (Progresive-Web-App) destinat atât personalului Facultății de Informatică Iași (profesori, administratori) cât și studenților facultății. Aplicația a fost dezvoltată după nevoile facultății și păstrează în design-ul ei identitatea acesteia. Încă de pe prima pagină accesibilă (pagina de log-are) utilizatorul poate identifica cu ușurință ce instituție reprezintă aplicația.

Aplicația suportă trei tipuri de utilizatori: administrator, profesor și student fiecare având permisiuni diferite. Administratorul poate să creeze și să editeze cursuri (acesta fiind container-ul în care profesorii atribuiți cursului pot adăuga materiale), poate crea conturi pentru profesori sau poate adăuga, modifica sau șterge anunțuri.

Profesorul poate crea conținutul cursurilor, el poate adăuga, modifica sau șterge lecții, seminare sau teme, poate vedea numărul de studenți care urmăresc cursurile lui, poate adăuga anunțuri legate de cursurile sale sau poate sa își editeze profilul.

Studentul își poate crea cont plecând de la pagina de log in. Pentru crearea contului acesta trebuie sa folosească adresa de e-mail primită în cadrul facultății pentru a putea confirma ulterior contul. În cazul în care și-a uitat parola acesta are posibilitatea resetării parolei printr-un formular pe care îl va primi pe email. După ce este autentificat studentul are acces la cursurile din cadrul facultății (acesta are posibilitatea de a alege cursurile care îl interesează în mod deosebit, iar acestea vor apărea pe o pagină separată), el poate deschide cursurile online sau le poate descărca, poate vedea anunțurile de la administrație sau de la profesorii ale căror materii le urmărește și poate să își editeze profilul.

## 3.1 Arhitectură

Arhitectura tradițională folosită pentru aplicațiile web este MVC (Model-View-Controler), una din cele mai folosite și mai bine văzute arhitecturi în industrie. Problema cu această arhitectură este faptul că se ajunge la cuplaj mare între UI (User-Interface) și business logic sau între business logic și database logic. [16]

Pentru a evita cuplajul prea mare am ales pentru această aplicație arhitectura Onion. Această arhitectură încearcă să rezolve problemele întâlnite în arhitecturile tradiționale cum ar fi “separation of concerns” sau “coupling”. [16]

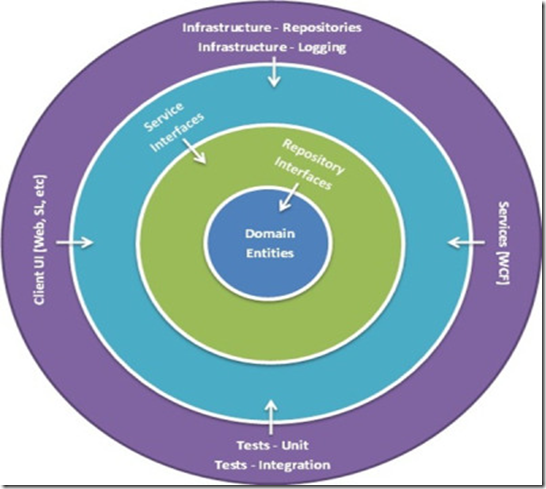
 Arhitectura Onion are layer-ele definite de la core spre infrastructură. Ideea principală este de a controla cuplajul, de aceea această arhitectură impune cuplajul să fie realizat din exterior în spre centru. În centru se află modelul, în jurul său se află layere care oferă comportament acestor obiecte (repositories, services), iar pe layerul din exterior se află lucrurile care au probabilitate mare de a fi schimbate, cum ar fi UI-ul, astfel modificările făcute pe ultimul layer nu impune modificări și pe celelalte layere. [16]

Figura 12: Arhitectura Onion [16]

### 3.1.1 Domain Layer

Domain layer-ul se află în centrul schemei arhitecturale, acesta conținând toate obiectele domeniului. La acest nivel se regăsesc doar definițiile și proprietățile entităților(pe acest layer nu se vor găsi bucăți de cod care să comunice cu baza de date sau care să aibă “business function”). În afara obiectelor domeniului aici se mai pot regăsi interfețele domeniului, dar doar interfețe, fără nici un fel de implementare.

Figura 13: Domain Layer FiiOnline

|  |
| --- |
| public class Week  {  private Week() { }  public Guid Id { get; private set; }  public string Title { get; private set; }  public DateTime Date { get; private set; }  public int WeekNr { get; private set; }  public string Description { get; private set; }  public ICollection<Resource> Resources { get; private set; }  [ForeignKey("Courses")]  public Guid CourseId { get; private set; }  public Course Course { get; private set; }  public static Week Create(string title, DateTime date, int weekNr, string description, Guid courseId, Course course)  {  var instace = new Week {Id = Guid.NewGuid()};  instace.Update(title, date, weekNr, description, courseId, course);  return instace;  }  public void Update(string title, DateTime date, int weekNr, string description)  {  Title = title;  Date = date;  WeekNr = weekNr;  Description = description;  }  public void Update(string title, DateTime date, int weekNr, string description, Guid courseId, Course course)  {  Title = title;  Date = date;  WeekNr = weekNr;  Description = description;  CourseId = courseId;  Course = course;  }  } |

### 3.1.2 Repository Layer

|  |
| --- |
| public class WeeksRepository: ACrudRepository<Week, Guid>, IWeeksRepository  {  public WeeksRepository(IDatabaseContext databaseContext) : base(databaseContext)  {}  public void Create(WeekCreatingModel entity)  {  var course = \_databaseContext.Courses.FirstOrDefault(c => c.Id.Equals(entity.courseId));  var week = Week.Create(entity.Title, entity.Date, entity.WeekNr, entity.Description, entity.courseId,  course);  course.AddWeek(week);  \_databaseContext.Courses.Update(course);  \_databaseContext.SaveChanges();  }  public override Week GetById(Guid Id) => \_databaseContext.Weeks.Include(w => w.Resources).FirstOrDefault(w => w.Id.Equals(Id));  public void Update(WeekCreatingModel entity, Guid id)  {  var week =\_databaseContext.Weeks.FirstOrDefault(w => w.Id.Equals(id));  week.Update(entity.Title, entity.Date, entity.WeekNr, entity.Description);  \_databaseContext.Weeks.Update(week);  \_databaseContext.SaveChanges();  }  public string GetCourseName(Guid id) => \_databaseContext.Courses.FirstOrDefault(c => c.Id.Equals(id)).Name;  } |

Repository layer este un layer de abstractizare între entitățile domeniului și Business Logic. Este un pattern care oferă o abordare cu un cuplaj redus pentru accesul datelor. În repository interogăm baza de date, mapăm datele de la sursă într-o entitate business și persistăm în baza de date schimbările făcute în business layer. [17]

Figura 14:Repository Layer FiiOnline

### 3.1.3 Service Layer

Acest layer conține interfețe folosite pentru comunicarea între UI Layer și Repository Layer. Conține business logic pentru o entitate, de aceea este numit, împreună cu Repository Layer, Business Layer. Atât la Repository Layer cât și la Service Layer se poate observa că intefețele sunt separate de implementări, ceea ce arată cuplajul redus și “separation of concerns”. [16]

|  |
| --- |
| public class WeeksService: IWeeksService  {  private readonly IWeeksRepository \_weeksRepository;  private readonly IResourcesService \_resourcesService;  public WeeksService(IWeeksRepository weeksRepository, IResourcesService resourcesService)  {  \_weeksRepository = weeksRepository;  \_resourcesService = resourcesService;  }  public void Create(WeekCreatingModel entity)  {  \_weeksRepository.Create(entity);  }  public void Update(WeekCreatingModel entity, Guid id)  {  \_weeksRepository.Update(entity, id);  }  public IEnumerable<WeekDTO> GetAll()  {  List<WeekDTO> weekDtos = new List<WeekDTO>();  var weeks = \_weeksRepository.GetAll();  foreach (var week in weeks)  {  weekDtos.Add(GetById(week.Id));  }  return weekDtos;  } |

### 3.1.4 UI Layer

Figura 15: Service Layer FiiOnline

Figura 16: UI Layer FiiOnline

Este layerul din exterior, acesta poate reprezenta o aplicație web, un Web API sau un proiect de Unit Teste. UI Layer comunică cu layer-ele interne prin interfețe. În aplicația “ Fii Online” pe layer-ul UI se găsește Proiectul FiiOnline, care este un Web API și conține controller-ele aplicației.

|  |
| --- |
| [Produces("application/json")]  [Route("Weeks")]  public class WeeksController : Controller  {  private readonly IWeeksService \_weeksService;  public WeeksController(IWeeksService weeksService)  {  \_weeksService = weeksService;  }  [Authorize(Roles = "Professor")]  [HttpPost]  public IActionResult PostCourse([FromBody] WeekCreatingModel weekModel)  {  try  {  \_weeksService.Create(weekModel);  return StatusCode((int)HttpStatusCode.Created);  }  catch (Exception e)  {  return BadRequest(e.Message);  }  }  [Authorize]  [HttpGet]  public IActionResult GetWeeks()  {  var weeks = \_weeksService.GetAll();  if (weeks == null)  {  return NotFound("There are no users");  }  return Ok(weeks);  } |

## 3.2 Baza de date

Pentru stocarea informațiilor aplicației am folosit baza de date SQL Server Express. Tabelele bazei de date au fost create folosind Entity Framework Core, abordarea folosită fiind Code First (menționată în capitolul 2.2.). Toate informațiile sunt stocate în baza de date cu excepția resurselor adăugate de profesori (cursuri, teme, seminarii) acestea sunt salvate pe server-ul de back-end , numai calea lor este salvată în baza de date. Același lucru se întâmplă și cu pozele de profil ale utiliazatorilor, doar că acestea sunt salvate în storage-ul de pe Firebase.

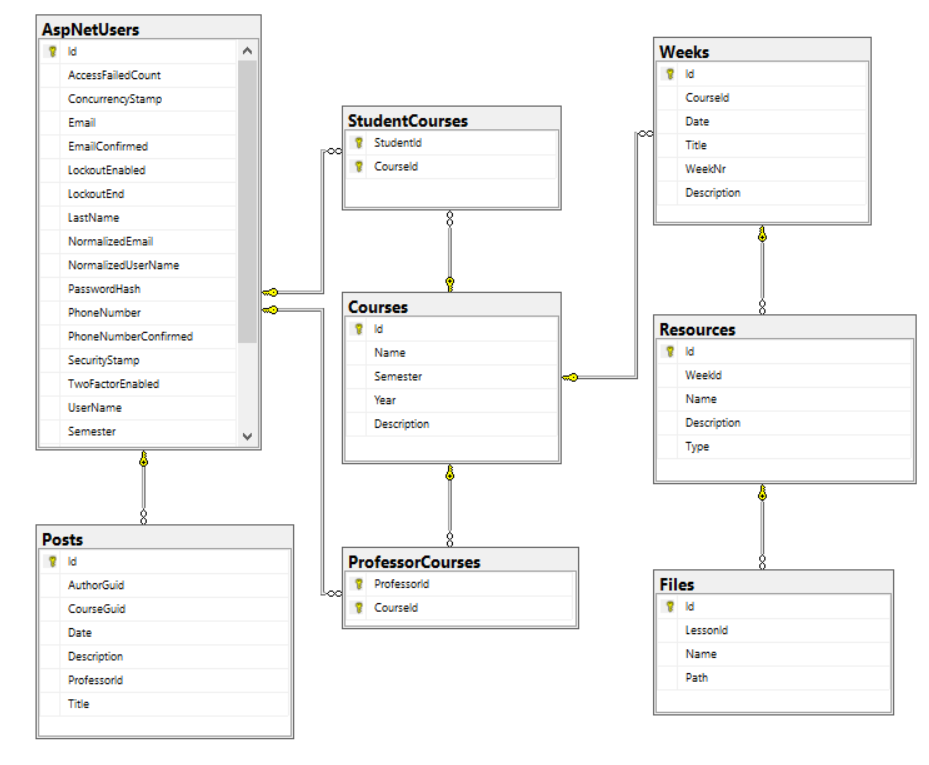


Figura 17: Arhitectura bazei de date FiiOnline

## 3.3 Detalii de implementare

### 3.3.1 Aplicație ASP.NET Core

|  |
| --- |
| public class Program  {  public static void Main(string[] args)  {  var host = BuildWebHost(args);  using (var scope = host.Services.CreateScope())  {  var dbContext = scope.ServiceProvider.GetService<DatabaseContext>();  var roleManager = scope.ServiceProvider.GetService<RoleManager<IdentityRole>>();  var userManager = scope.ServiceProvider.GetService<UserManager<User>>();  DbSeeder.Seed(dbContext, roleManager, userManager);  }  host.Run();  }  public static IWebHost BuildWebHost(string[] args) =>  WebHost.CreateDefaultBuilder(args)  .UseStartup<Startup>()  .Build();  } |

O aplicație ASP.Net Core este un console app care creează un web server în metoda Main.

Metoda Main invocă WebHost.CreateDefaultBuilder, care urmează pattern-ul builder pentru a crea un host al aplicației web. Builder-ul are metode care definesc server-ul web și clasa Startup.

Clasa Startup.cs conține două metode: ConfigureServices și Configure care vor fi apelate când pornește aplicația. Metoda ConfigureServices definește serviciile folosite de aplicație, iar metoda Configure este folosită pentru a specifica aplicației cum să răspundă la requesturi HTTP.

### 3.3.2 Crearea tabelelor Code First

Clasa DatabaseContext este responsabilă pentru comunicarea cu baza de date. Primul pas în crearea tabelelor este conectarea la baza de date. Acest lucru se realizează prin înregistrarea DatabaseContext-ului ca un serviciu în ConfigureServices din Startup.

|  |
| --- |
| services.AddTransient<IDatabaseContext, DatabaseContext>();  services.AddDbContext<DatabaseContext>(options => options.UseSqlServer(Configuration.GetConnectionString("DefaultConnection"))); |

Metoda AddTransient este folosită pentru maparea tipurilor abstracte la servicii concrete care vor fi instanțiate separat pentru fiecare obiect care are nevoie de ele (metoda AddTransient este folosită pentru mecanismul de dependecy injection).

Al doilea pas este crearea de modele, acestea se află în domain layer. Modelele conțin detalii despre entitate, fiecare atribut din model va reprezenta o coloană din tabelul ce urmează sa fie generat.

|  |
| --- |
| public class Week  {  private Week() { }  public Guid Id { get; private set; }  public string Title { get; private set; }  public DateTime Date { get; private set; }  public int WeekNr { get; private set; }  public string Description { get; private set; }  public ICollection<Resource> Resources { get; private set; }  [ForeignKey("Courses")]  public Guid CourseId { get; private set; }  public Course Course { get; private set; } |

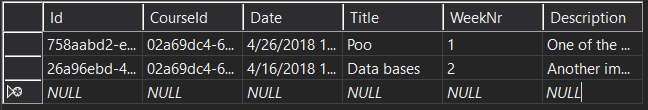


Figura 18: Tabelul Weeks

Pasul trei este să mapăm entitățile la tabele, acest lucru se face în DatabaseContext:

|  |
| --- |
| public sealed class DatabaseContext : IdentityDbContext<User>, IDatabaseContext  {  public DatabaseContext(DbContextOptions<DatabaseContext> options) : base(options)  {  Database.EnsureCreated();  }  public new DbSet<User> Users { get; set; }  public DbSet<Student> Students { get; set; }  public DbSet<Professor> Professors { get; set; }  public DbSet<Course> Courses { get; set; }  public DbSet<Week> Weeks { get; set; }  public DbSet<Resource> Resources { get; set; }  public DbSet<ProfessorCourse> ProfessorCourses { get; set; }  public DbSet<AppFile> Files { get; set; }  public DbSet<Post> Posts { get; set; }  public DbSet<StudentCourse> StudentCourses { get; set; }  } |

Tot în DatabaseContext se fac configurațiile pentru relațiile dintre entități (tabele). Dacă am dori să configurăm o relație many-to-many am avea nevoie de trei entități, două dintre ele reprezintă entitățile propriu-zise iar a treia entitate va fi tabelul de legătură între cele două entități.

O relație many-to-many între profesori și cursuri arată în felul următor:

Pe lângă atributele sale entitatea professor va conține un array de ProfessorCourses (entitatea de legătură)

|  |
| --- |
| public class Professor: User  {  private Professor()  {  }  public ICollection<ProfessorCourse> ProfessorCourses { get; private set; }  } |

Similiar și pentru entitatea courses:

|  |
| --- |
| public class Course  {  private Course() { }  public Guid Id { get; private set; }  public string Name { get; private set; }  public string Year { get; private set; }  public int Semester { get; private set; }  public string Description { get; private set; }  public ICollection<ProfessorCourse> UserCourses { get; private set; }  } |

Entitatea ProfessorCourses va conține id-urile celor două entități între care face legătura și entitățile propriu-zise.

|  |
| --- |
| public class ProfessorCourse  {  private ProfessorCourse() { }  public string ProfessorId { get; private set; }  public Professor Professor { get; private set; }  public Guid CourseId { get; private set; }  public Course Course { get; private set; }  } |

În DatabaseContext această relație se mapează în felul următor:

|  |
| --- |
| protected override void OnModelCreating(ModelBuilder modelBuilder)  {  base.OnModelCreating(modelBuilder);  modelBuilder.Entity<ProfessorCourse>()  .HasKey(uc => new {uc.ProfessorId, uc.CourseId});  modelBuilder.Entity<ProfessorCourse>()  .HasOne(uc => uc.Professor)  .WithMany(p => p.UserCourses)  .HasForeignKey(uc => uc.ProfessorId);  modelBuilder.Entity<ProfessorCourse>()  .HasOne(uc => uc.Course)  .WithMany(c => c.UserCourses)  .HasForeignKey(uc => uc.CourseId);  } |

Configurația de mai sus mapează două relații one-to-many, una între Professor și ProfessorCourse, iar cealaltă între Course și ProfessorCourse.

Ultimul pas în crearea tabelelor este să folosim migrations pentru generarea lor. În package manager console, selectăm proiectul „Data.Persistance” și folosim următoarele două comenzi:

* Add-Migration <Nume>
* Update-Database

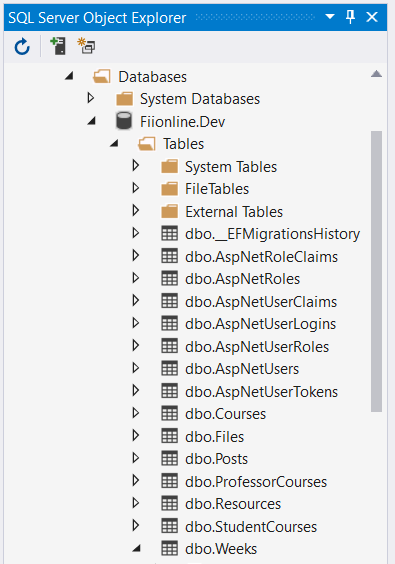
La sfârșit putem merge în Visual Studio 2017 -> Edit -> SQL Server Object Explorer iar acolo ar trebui să găsim baza de date cu tabelele corespunzătoare.

Figura 19: Tabelele generate cu Entity Framework Core

### 3.3.3 Mecanismul de logare

ASP.NET Core Identity este un sistem de membership care permite adăugarea funcționalității de login în aplicații. Utilizatorii utilizatorii își pot crea conturi sau pot folosi provideri externi ca: Facebook, Google, Twitter, etc. [18]

Pentru proiectul „Fii Online” am configurat ASP.NET Core Identity să folosească SQL Server Express pentru a stoca utilizatorii, parolele lor și datele despre profilul lor.

Pentru a folosi Identity trebuie să adăugăm serviciile sale în Startup.cs în metoda ConfiureServices:

|  |
| --- |
| services.AddIdentity<User, IdentityRole>(  opts =>  {  opts.Password.RequireDigit = true;  opts.Password.RequireLowercase = true;  opts.Password.RequireUppercase = true;  opts.Password.RequireNonAlphanumeric = false;  opts.Password.RequiredLength = 7;  opts.SignIn.RequireConfirmedEmail = true;  })  .AddEntityFrameworkStores<DatabaseContext>()  .AddDefaultTokenProviders(); |

Primele 5 opțiuni folosite mai sus fac referire la forma pe care trebuie sa o aibă parola utilizatorului. Ultima opțiune aduce o nouă constrângere pentru utilizator, și anume, contul lui trebuie să fie validat înainte de a avea acces.

Serviciile menționate sunt accesibile prin dependency injection. Identity va fi accesibil aplicației prin metoda UseAuthentication() din Configure:

|  |
| --- |
| public void Configure(IApplicationBuilder app, IHostingEnvironment env)  {  app.UseAuthentication();  } |

Pentru ca aplicația să poată folosi Identity, după configurările din Startup.cs, trebuie să ne asigurăm că Entitatea User implementează interfața IdentityUser, iar DatabaseContext implementează interfața IdentityDbContext<User>

După aceste setări, când rulăm aplicația ar trebui să găsim în SQL Server Object Explorer următoarele tabele:

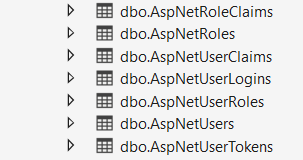


Figura 20: Tabele generate de Identity

|  |
| --- |
| [HttpPost("register")]  public async Task<IActionResult> Register([FromBody] UserCreatingModel userModel)  {  var userNameNumber = \_usersService.GetNumberOfSimilarNames(userModel.FirstName);  userModel.Username = String.Format("{0}{1}", userModel.FirstName, (userNameNumber+1).ToString());  try  {  var result = await \_usersService.CreateAsync(userModel);  if (result.Succeeded)  {  var user = \_usersService.GetByUserName(userModel.Username);  var code = await \_userManager.GenerateEmailConfirmationTokenAsync(user);  var callbackUrl = String.Format("http://localhost:4200/#/account-confirmation?userId={0}&code={1}", user.Id,  code);  await \_emailSender.SendEmailAsync(userModel.Email, "Confirm your account",  $"Please confirm your account by clicking here: <a href='{callbackUrl}'>link</a>");  return StatusCode((int) (HttpStatusCode.Created));  }  return BadRequest("Could not create user");  }  catch (Exception e)  {  return BadRequest(e.Message);  }  } |

Acum că Identity este configurat trebuie create controllerele de de register și login.

Controller-ul de mai sus încearcă să creeze un nou utilizator. Dacă utilizatorul a fost creat cu succes, atunci utilizatorul v-a primi un email care v-a conține un querry string. Querry string-ul respectiv conține la rândul său id-ul utilizatorului și un token generat de \_userManager. Atunci când utilizatorul accesează link-ul primit in e-mail, un nou end-point va valida contul acestuia pe baza informațiilor din querry string.

|  |
| --- |
| [AllowAnonymous]  [HttpGet]  [Route("ConfirmAccount", Name = "ConfirmAccountRoute")]  public async Task<IActionResult> ConfirmEmail(string userId = "", string code = "")  {  try  {  var user = \_userManager.FindByIdAsync(userId).Result;  var result = await \_userManager.ConfirmEmailAsync(user, code);  return Ok(result);  }  catch (Exception e)  {  return BadRequest(e.Message);  }  } |

Dacă verificarea userului decurge cu succes, atunci câmpul EmailConfirmed din tabelul AspNetUsers, de pe linia utilizatorului va fi modificat din false în true, iar acesta se va putea loga.

|  |
| --- |
| [AllowAnonymous]  [HttpGet("auth")]  public async Task<object> Login(string email="", string password ="")  {  var appUser = \_userManager.Users.SingleOrDefault(r => r.Email == email);  var result = await \_signInManager.PasswordSignInAsync(appUser.UserName, password, false, false);  if (result.Succeeded)  {  return Json(new {token = \_generator.GenerateJwtToken(email, appUser), appUser});  }  throw new ApplicationException("INVALID\_LOGIN\_ATTEMPT");  } |

Figura 21: Controller logare utilizator

În cazul în care un utilizator și-a uitat parola și dorește schimbarea ei, acesta trebuie să își introducă email-ul în formularul de recuperare a parolei, iar dacă emailul este valid, acesta va primi un e-mail care va conține, ca și în cazul confirmării contului, un querry string alcătuit din id-ul user-ului și un token unic generat de userManager.

|  |
| --- |
| [AllowAnonymous]  [HttpPost("forgotpassword")]  public async Task<IActionResult> ForgotPassword([FromBody] LoginCreatingModel model)  {  var user = await \_userManager.FindByEmailAsync(model.Email);  if (user == null || !(await \_userManager.IsEmailConfirmedAsync(user)))  {  // Don't reveal that the user does not exist or is not confirmed  return BadRequest((int) HttpStatusCode.Forbidden);  }  var code = await \_userManager.GeneratePasswordResetTokenAsync(user);  var callbackUrl = String.Format("http://localhost:4200/#/reset-password?userId={0}&code={1}", user.Id,  code);  await \_emailSender.SendEmailAsync(model.Email, "Reset Password",  $"Please reset your password by clicking here: <a href='{callbackUrl}'>link</a>");  return StatusCode((int) HttpStatusCode.OK);  } |

Doar accesând link-ul din email, utilizatorul își poate schimba parola, faptul că are acces la email confirmă faptul că cel care intenționează să modifice parola este întradevăr deținătorul contului. O dată accesat, utilizatorul este redirecționat către o nouă pagină unde poate completa un formular cu noua sa parolă. Un nou controller va verifica datele din querry string pentru a valida autenticitatea utilizatorului, iar dacă datele sunt corecte userManager-ul va modifica parola utilizatorului.

|  |
| --- |
| public async Task<IActionResult> ResetPassword([FromBody] LoginCreatingModel model, string userId = "", string code = "")  {  try  {  var user = \_userManager.FindByIdAsync(userId);  await \_userManager.ResetPasswordAsync(user.Result, code, model.Password);  return StatusCode((int) HttpStatusCode.OK);  }  catch (Exception e)  {  return BadRequest(e.Message);  }  } |

### 3.3.4 JWT

Pentru a asigura faptul că doar utilizatorii aplicației pot trece de login și că doar ei pot face request-uri, mai mult, doar utilizatorii cu anumite roluri pot face anumite request-uri am folosit JWT (menționate în capitolul 2.3).

Pentru a folosi JWT trebuie mai întâi făcute configurările în Startup.cs, ConfigureServices:

|  |
| --- |
| services.AddAuthentication(opts =>  {  opts.DefaultScheme = JwtBearerDefaults.AuthenticationScheme;  opts.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;  opts.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;  })  // Add Jwt token support  .AddJwtBearer(cfg =>  {  cfg.RequireHttpsMetadata = false;  cfg.SaveToken = true;  cfg.TokenValidationParameters = new TokenValidationParameters  {  // standard configuration  ValidIssuer = Configuration["Auth:Jwt:Issuer"],  IssuerSigningKey = new SymmetricSecurityKey(  Encoding.UTF8.GetBytes(Configuration["Auth:Jwt:Key"])),  ValidAudience = Configuration["Auth:Jwt:Audience"],  ClockSkew = TimeSpan.Zero,  // security switches  RequireExpirationTime = true,  ValidateIssuer = true,  ValidateIssuerSigningKey = true,  ValidateAudience = true  };  cfg.IncludeErrorDetails = true;  }); |

În codul de mai sus înregistrăm schema autentificării JWT folosind metoda AddAuthentication, specificând JwtBearerDefaults.AuthenticationScheme. După aceasta configurăm schema de autentificare cu opțiuni pentru JWT bearer. În mod special specificăm care parametri trebuie luați în considerare pentru că token-ul să fie considerat valid. În cod am specificat că vom considera un token valid dacă:

* Serverul care a emis token-ul este validat (“ValidateIssuer = true”)
* Cel care trebuie să primească token-ul este autorizat sa îl primească (“ValidateAudience = true)
* Token-ul nu este expirat și semnătura emițătorului este validă (ValidateLifetime = true)
* Dacă cheia folosită pentru semnarea toke-ului este parte dintr-o listă de chei de încredere(ValidateIssuerSigningKey = true) [11]

În plus adăugăm valori pentru emițător (Issuer), audiență și cheia semnăturii în appsettings.json

|  |
| --- |
| "Auth": {  "Jwt": {  "Issuer": "http://localhost:63944/",  "Audience": "http://localhost:8080/",  "Key": "Ay2Hs5%Sc\_$3d!f?=h.",  "ExpireDays": 1  }  } |

Acum că JWT este configurat avem nevoie de o metodă care să genereze aceste token-uri. În codul din figura 21, în cazul în care utilizatorul era autentificat cu succes controller-ul returna:

|  |
| --- |
| return Json(new {token = \_generator.GenerateJwtToken(email, appUser) , appUser}) |

GenerateJwtToken este metoda care generează token-ul.

|  |
| --- |
| public object GenerateJwtToken(string email, User user)  {  var claims = new List<Claim>  {  new Claim(JwtRegisteredClaimNames.Sub, email),  new Claim(JwtRegisteredClaimNames.Jti, Guid.NewGuid().ToString()),  new Claim(ClaimTypes.NameIdentifier, user.Id),  new Claim(ClaimTypes.Role, user.Role)  };  var key = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(\_configuration["Auth:Jwt:Key"]));  var creds = new SigningCredentials(key, SecurityAlgorithms.HmacSha256);  var token = new JwtSecurityToken(  \_configuration["Auth:Jwt:Issuer"],  \_configuration["Auth:Jwt:Audience"],  claims,  expires: DateTime.Now.AddMinutes(120),  signingCredentials: creds  );  return new JwtSecurityTokenHandler().WriteToken(token);  } |

Anterior, în capitolul Json Web Tokens (2.3) am menționat că aceste token-uri pot conține anumite date, numite “claims”. Acestea sunt în general informații despre utilizator, de exemplu e-mail, id, rol, sau orice altă informație care ar putea sa îl diferențieze de alt utilizator. Putem adăuga claim-uri pentru a putea verifica și autoriza un utilizator când cere o resursă

În codul de mai sus, în metoda GenerateJwtToken, primul lucru pe care îl fac este să adaug clam-uri. Rolul va fi de folos în mod deosebit atunci când trebuie sa decid daca un utilizator are dreptul să acceseze sau să modifice o anumită resursă.

După ce stabilesc caim-urile, îmi generez cheia, credențialele și în final token-ul folosind clasa JwtSecurityToken. La sfârșit returnez token-ul ca un string, facând această conversie cu metoda WriteToken din clasa JwtSecurityTokenHandler.

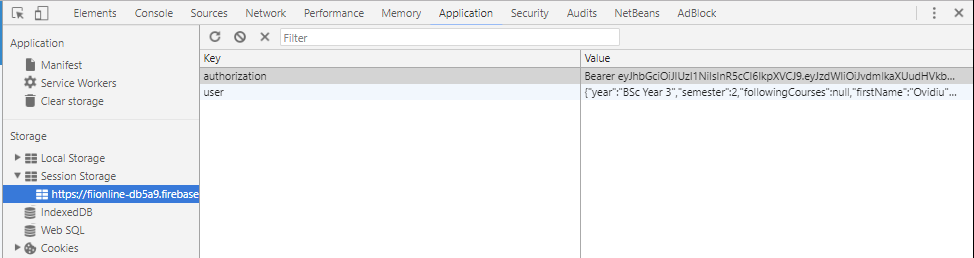
În final, după ce un utilizator s-a logat cu succes, pe partea de client vor fi slavate în session storage detalii despre el și token-ul cu care va avea acces la numite resurse.

Figura 22: Session storage după logare

# 4. Concluzie

În ultimii ani au apărut pe piață tot mai multe aplicații lms (Learning Management System), fiecare aducând noi funcționalități. Deși soluțiile par să arate și să se comporte diferit față de aplicațiile inițiale, acestea au la bază același lucru, accesul la materiale de studiu.

Aplicația “Fii Online” apare ca o alternativă pentru aceste aplicații, dar în mod special pentru soluția actuală de a oferi studenților acces la cursuri în cadrul Facultății de Informatică. Această platformă este o alternative a soluției actuale întrucât reușește să centralizeze toate cursurile într-un singur loc, oferă siguranță profesorilor că nimeni din afara facultății nu va avea acces la cursurile lor, totodată le oferă acestora o variantă de a adăuga cursuri într-un timp foarte scurt, fără niciun efort.

Totodată “Fii Online” reprezintă, în contextul Facultății de Informatică, o alternativă pentru aplicațiile deja existente pe piață, întrucât este o aplicație cu adevărat gratuită, indiferent de numărul utilizatorilor și mai mult, la momentul actual oferă funcționalitățile strict necesare, spre deosebire de majoritatea aplicațiilor competitoare care au ajuns să integreze un număr foarte mare de funcționalități ce fac aplicația să fie dificil de utilizat și menținut.

Cu toate acestea există o serie de funcționalități care pot fi aduse platformei astfel încât aplicația să joace un rol și mai important în planul facultății. Una din funcționalități ar fi un canal de comunicare între studenți și profesori, deși posibilitatea de a adăuga anunțuri scade deja utilitatea web-mail-ului, consider că având un canal de comunicare, am putea elimina necesitatea lui. Un alt lucru care ar aduce valoare aplicației ar fi integrarea cu eSims și cu orarul facultății, astfel, “Fii Online” ar centraliza toate serviciile pe care facultatea le folosește la momentul actual.

# 6. Referințe

|  |  |
| --- | --- |
| [1] | „Moodle,” [Interactiv]. Available: https://www.wikiwand.com/en/Moodle. |
| [2] | „Chamilo,” [Interactiv]. Available: https://chamilo.org/. [Accesat 15 Mai 2018]. |
| [3] | „SWAD,” [Interactiv]. Available: https://alternativeto.net/software/swad/. [Accesat 15 Mai 2018]. |
| [4] | „Canvas,” [Interactiv]. Available: https://reviews.financesonline.com/p/canvas-lms/. [Accesat 15 Mai 2019]. |
| [5] | „TalentLMS,” [Interactiv]. Available: https://reviews.financesonline.com/p/talentlms/. [Accesat 15 Mai 2018]. |
| [6] | „TalentLMS2,” [Interactiv]. Available: https://www.talentlms.com/prices. [Accesat 15 Mai 2018]. |
| [7] | R. A. S. L. Daniel Roth, „Microsoft,” [Interactiv]. Available: https://docs.microsoft.com/en-us/aspnet/core/?view=aspnetcore-2.0. [Accesat 22 Mai 2018]. |
| [8] | A. Stringfellow, „DZone,” [Interactiv]. Available: https://dzone.com/articles/net-core-vs-net-framework-how-to-pick-a-net-runtim. [Accesat 22 Mai 2018]. |
| [9] | M. Watson, „stackify,” 16 August 2017. [Interactiv]. Available: https://stackify.com/asp-net-razor-pages-vs-mvc/. [Accesat 22 Mai 2018]. |
| [10] | „TutorialsPoints,” [Interactiv]. Available: https://www.tutorialspoint.com/entity\_framework/entity\_framework\_overview.htm. [Accesat 22 Mai 2018]. |
| [11] | A. Chiarelli, „auth0,” 7 Decembrie 2017. [Interactiv]. Available: https://auth0.com/blog/securing-asp-dot-net-core-2-applications-with-jwts/. [Accesat 22 Mai 2018]. |
| [12] | P. Otemuyiwa, „auth0,” 14 Noiembrie 2017. [Interactiv]. Available: https://auth0.com/blog/whats-new-in-angular5/. [Accesat 25 Mai 2018]. |
| [13] | T. VanToll, „Telerik Developer Network,” 18 Ianuarie 2018. [Interactiv]. Available: https://developer.telerik.com/topics/web-development/what-is-angular/. [Accesat 25 Mai 2018]. |
| [14] | M. Gaunt, „developers.google,” [Interactiv]. Available: https://developers.google.com/web/fundamentals/primers/service-workers/. [Accesat 30 Mai 2018]. |
| [15] | „angular.io,” [Interactiv]. Available: https://angular.io/guide/service-worker-intro. [Accesat 30 Mai 2018]. |
| [16] | C. Vihite, „thedigitalgroup,” 06 Iulie 2015. [Interactiv]. Available: http://blog.thedigitalgroup.com/understanding-onion-architecture. [Accesat 29 Mai 2018]. |
| [17] | S. S. Shekhawat, „c-sharpercorner,” 01 Ianuarie 2017. [Interactiv]. Available: https://www.c-sharpcorner.com/article/onion-architecture-in-asp-net-core-mvc/. [Accesat 30 Mai 2018]. |
| [18] | „Microsoft,” [Interactiv]. Available: https://docs.microsoft.com/en-us/aspnet/core/security/authentication/identity?view=aspnetcore-2.1&tabs=visual-studio%2Caspnetcore2x. [Accesat 1 Iunie 2018]. |
| [19] | „tutorialspoint,” [Interactiv]. Available: https://www.tutorialspoint.com/materialize/index.htm. [Accesat 25 Mai 2018]. |
| [20] | „jwt.io,” [Interactiv]. Available: https://jwt.io/introduction/. [Accesat 22 Mai 2018]. |