MINISTERUL EDUCAȚIEI ȘI CERCETĂRII ȘTIINȚIFICE

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
DEPARTAMENTUL CALCULATOARE

Course Streaming – Sistem de tip e-learning pentru educaţia nonformală

LUCRARE DE LICENŢĂ

Absolvent: **Ana-Cristina ANDONIE**

Coordonator ştiinţific: **Prof. as. ing. Cosmina IVAN**

2017

MINISTERUL EDUCAȚIEI ȘI CERCETĂRII ȘTIINȚIFICE

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
DEPARTAMENTUL CALCULATOARE

DECAN,
Prof. dr. ing. Liviu MICLEA

DIRECTOR DEPARTAMENT,
Prof. dr. ing. Rodica POTOLEA

Absolvent: **Ana-Cristina ANDONIE**

TITLUL LUCRĂRII DE LICENŢĂ

1. **Enunţul temei:** *Scopul acestui proiect este de a implementa un sistem software, care să ruleze în browser, capabil să ofere servicii de comunicare în timp real prin concepul de live streaming. Aplicaţia web utilizează tehnologia EasyRTC pentru a realiza comunicarea video, chat şi transferul de fişiere.*

2. **Conţinutul lucrării:** *Introducere, Obiectivele Proiectului, Studiu Bibliografic, Analiză și Fundamentare Teoretică, Proiectare de Detaliu și Implementare, Testare și Validare, Manual de Instalare și Utilizare, Concluzii și Dezvoltări Ulterioare, Bibliografie, Anexe.*

3. **Locul documentării**: Universitatea Tehnică din Cluj-Napoca, Departamentul Calculatoare

4. **Consultanţi**:

5. **Data emiterii temei:** 1 noiembrie 2016

6. **Data predării:** 14 iulie 2017

Absolvent: _____

Coordonator ştiinţific: _____

MINISTERUL EDUCAȚIEI ȘI CERCETĂRII ȘTIINȚIFICE

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
DEPARTAMENTUL CALCULATOARE

Cuprins

Capitolul 1. Introducere 1
1.1. Contextul proiectului 1
1.2. Structura 2
Capitolul 2. Obiectivele Proiectului 4
2.1. Obiectivul principal 4
2.2. Obiective secundare 4
Capitolul 3. Studiu Bibliografic 6
3.1. Concepte 6
3.1.1. Live Streaming 6
3.1.2. Real-Time Communication 7
3.1.3. E-Learning 7
3.2. Sisteme Similare 9
3.2.1. TokBox 9
3.2.2. StreamTorrent 9
3.2.3. XirSys 10
3.2.4. Apizee 11
Capitolul 4. Analiză şi Fundamentare Teoretică 13
4.1. Tehnologii 13
4.1.1. WebRTC 13
4.1.2. EasyRTC 15
4.1.3. Node.js 16
4.1.4. Angular.js 18
4.1.5. Auth0 20
4.1.6. HTML5 21
4.1.7. MySQL 22
4.2. Cerinţe funcţionale 23
4.3. Cerinţe non-funcţionale 25
4.4. Diagrame use-case 26
4.5. Cerinţe de resurse 31
Capitolul 5. Proiectare de Detaliu si Implementare 32
5.1. Structura generală 32
5.2. Descriere componente 33

5.2.1. Structura EasyRTC 34
5.2.2. Structura interfeţei 39
5.2.3. Structura bazei de date 47
Capitolul 6. Testare şi Validare 50
6.1. Black Box Testing 50
6.2. White Box Testing 51
Capitolul 7. Manual de Instalare si Utilizare 53
7.1. Manual de Instalare 53
7.2. Manual de Utilizare 53
Capitolul 8. Concluzii 59
8.1. Dezvoltari ulterioare 59
Bibliografie 61
Anexa 1 - Lista figurilor din lucrare 63
Anexa 2 - Lista tabelelor din lucrare 64
Anexa 3 - Glosar de termeni 65

Capitolul 1. Introducere

1.1. Contextul proiectului

Trăim într-o perioadă de timp în care toată informaţia de care avem nevoie este la câteva click-uri distanţa de noi. Dacă în trecutul îndepartat singura cale de a comunica dintr-un loc în altul era trimiterea de scrisori, apoi avansandu-se la telefonie, acum în timpurile noastre toată informaţia este live, adică nu mai trebuie să aşteptăm deloc pentru a avea acces la informaţie. Majoritatea oamenilor sunt mereu în căutarea de noi informaţii, idei şi ştiri şi nu le place să aştepte prea mult pentru a obţine astfel de date, de aceea tehnologia a avansat atat de mult şi se investeşte în aceasta tocmai cu scopul de eficientiza comunicarea şi modurile de transmitere a informaţiilor. Orice mod de a transmite informaţii poate fi conderat o comunicare. James Lull, un cercetător american defineşte comunicarea astfel: "Comunicarea este terenul de întalnire conceptuală unde se intersectează relaţiile interpersonale şi inovaţiile tehnologice, stimulentele politico-economice şi ambiţiile socio-culturale, divertismentul usor şi informaţia serioasă, mediile ambiante locale şi influenţele globale, forma şi conţinutul, substanţa şi stilul." Orice act iniţiat de un om are într-un sens explicit sau implicit o componentă care este bazată pe comunicare, indiferent că aceasta este între două persoane sau între o persoană şi un dispozitiv cand persoana caută resurse.

Comunicând reuşim să transmitem informaţii noi şi să evoluăm ca omenire. Tehnologia de azi s-a dezvoltat deoarece oamenii au contribuit împreuna la proiectarea şi construirea acesteia. De la dispozitive smart, la tehnologii de comunicare wireless, în ziua de azi e mai usor ca niciodată să comunici indiferent unde te afli pe glob. Modul de a comunica prin Internet nu ţine cont de naţionalitate, etnie, religie sau alte caracteristici care ne diferenţiază sau ne despart fizic de alţi oameni. Internetul este împarţial, şi "oferă" informaţiile tuturor celor care le accesează. Internetul este sursa principală de informaţie, unealta care te poate plasa virtual într-un alt loc, total diferit faţă de cel unde te afli fizic. Toată informaţia expusă este actualizată constant astfel că nu rămâi în urmă cu nici o ştire sau eveniment. De asemenea datele pe care le accesăm pe Internet sunt protejate, majoritatea site-urilor şi companiilor mari oferă securitatea datelor pentru o încredere oferită utilizatorilor. Această caracteristică a securităţii oferă posibilitatea conectării pe Internet cu date personale şi accesarea unor cursuri şi resurse sau pentru accesarea unor site-uri de socializare pentru a comunica.

Un beneficiu major al Internetului este că poate oferi informaţii suplimentare pe lângă cele învăţate la şcoală sau facultate sau poate susţine învăţarea şi descoperirea unor lucruri noi. Aşa s-a întâmplat să apară aceast proiect dedicat expunerii unor informaţii cu ajutorului unui live stream. De ce live stream? Pentru că mai repede de timpul real de a primi informaţia nu se poate. Acest proiect s-a născut în contextul în care Internetul fiind la îndemână oricui, pot fi transmise prin acesta orice fel de informaţii chiar şi un video care este filmat în momentul transmiterii, adică un live stream. Acest video este bineînţeles creat de oameni de specialitate care au noi informaţii de împărtăşit, proiectul fiind bazat pe o metodă de predare-învăţare, care aparţine fenomenului de e-learning.

Aplicaţie web construită se încadrează în domeniul educaţional cu preponderenţă educaţia non-formală cu transmiterea informaţiilor cu ajutorul tehnologiilor de ultima ora. Aplicaţia este simplă şi are rolul de a eficientiza modul de a comunica având că sursă

principala de comunicare un video transmis live şi un chat în care participanţii pot interacţiona. Această aplicaţie este o alternativă la sistemele deja existente pe piaţă, care pune accentul pe dezvoltarea cunoştinţelor pe un anumit domeniu, adică o educaţie care sporeşte cultură generală a individului.

1.2. Structura

În primul capitol este expus contextul proiectului şi scopul pentru care acest proiect a fost creat. Orice aplicaţie are în spate un scop pentru care a fost realizată şi un context în care se va integra şi va fi necesară.

În cel de-al doilea capitol vor fi prezentate obiectivele principale şi secundare ale aplicaţiei dezvoltate. Orice aplicaţie pe lângă obiectivul principal are şi câteva obiective secundare care susţin realizarea obiectivului principal al sistemului. Toate cerinţele funcţionale vor fi prezentate mai detaliat în capitolele care urmează.

În cel de-al treilea capitol este prezentat studiu bibliografic. Acesta conţine conceptele care stau la baza construirii acestui proiect. Studiul bibliografic este important deoarece prin acesta se pot analiză anumite concepte, şi în urmă studiului se pot construi aplicaţii care să satisfacă cerinţele acelor analize. Conceptele studiate pentru acest proiect sunt legate de comunicarea în timp real, de modul în care live streaming-ul susţine realizarea comunicării în timp real şi cum acest mod de comunicare poate fi văzut că şi un tip de educaţie non-formală. Pe lângă concepte, în studiu bibliografic sunt incluse şi analiza unor sistemele similare deja existente pe piaţă şi o comparaţie între acestea pentru a vedea ce aduce nou aplicaţia dezvoltată în acest proiect.

Capitolul patru este despre analiza şi fundamentarea teoretică. Aici intră toate tehnologiile folosite pentru a implementa acest proiect şi motivul pentru care acestea au fost alese. Tehnologia care stă la baza proiectului este EasyRTC, o platforma creată de WebRTC în Node.js. Înregistrarea pe site se face cu ajutorul aplicaţiei Auth0 iar interfaţă este o aplicaţie de tip SPA creată cu Angular.js. Datele sunt păstrate într-o baza de date MySQL. Analiza din acest capitop se bazează pe detalierea cerinţelor funcţionale şi non-funcţionale şi enunţarea cerinţelor hardware si software necesare proiectului. Cerinţele funcţionale sunt împărţite în funcţie de rolul utilizatorului logat în sistem. Cerinţele de resurse se referă la dispozitivele necesare pentru a avea acces la aplicaţia web dezvoltată.

În capitolul cinci este prezentată proiectarea de detaliu şi implementare. Această presupune descrierea arhitecturii sistemului, cum a fost construit şi cum sunt conectate toate tehnologiile folosite pentru a îndeplini cerinţele funcţionale ale sistemului, dar şi pe cele non-funcţionale ale acestuia. Prezentarea pleacă de la structura generală a aplicaţiei şi continuă cu analizarea tuturor componentelor. Componentele principale ale aplicaţiei sunt serverul EasyRTC, baza de date MySQL şi interfaţa utilizator construită ca o aplicaţie SPA în care autentificarea este asigurată de serviciul Auth0. Tot ce cuprind componentele în mod individual este reprezentat şi în diagrama de deployment a sistemului.

Capitolul şase este despre testarea şi validarea aplicaţiei. Pentru testarea sistemului s-au ales câteva cazuri de test care să cuprindă atât verificarea cerinţelor funcţionale cât şi a celor non-funcţionale. Este foarte important ca aceste cerinţe să fie dezvoltate împreună pentru a duce la performanţele cele mai bune şi la un sistem stabil. Testarea s-a realizat folosind două metode: Black Box Testing şi White Box Testing.

În capitolul şapte este prezentat manualul de instalare şi utilizare. Aici va fi prezentată instalarea tuturor tehnologiilor necesare pentru a construi aplicaţia şi manualul de utilizare, adică modul de conectare la aplicaţia web şi modul de gestionare a cerinţelor funcţionale cuprinse în această.

Capitolul opt este legat de concluziile care se desprind în urma dezvoltării şi testării proiectului şi posibile dezvoltări ulterioare. Orice sistem oricât de bine gândit este la început, pe pacursul dezvoltării sale suferă schimbări şi îmbunătăţiri, de aceea este important de a vedea stadiul în care se găseşte aplicaţia într-un anumit moment pentru a putea observa posibilele schimbări care pot să intervină.

Capitolul 2. Obiectivele Proiectului

2.1. Obiectivul principal

Proiectul are ca şi obiectiv principal proiectarea unui sistem care să faciliteze difuzarea în timp real a unui curs tuturor persoanelor care nu pot participa la acesta în mod direct, adică fizic. Comunciarea informaţiilor în timp real folosind doar o conexiune la Internet şi un browser web este cerinţă funcţională principala care se doreşte a fi implementată în această lucrare. Aplicaţia web nu va implica nici o instalare de plugin-uri suplimentare sau diferite programe externe pentru a avea acces la informaţie, ci doar acces la Internet şi un dispozitiv electronic (ex. calculator, laptop, tabletă). Simplitatea aplicaţiei constă în faptul că odată logat un utilizator are acces la toate evenimentele şi cursurile care se desfăşoară pe site. Nu este nevoie de o aprobare din partea administratorului pentru a avea acces la date şi cel mai important, toate cursurile sunt gratuite, această aplicaţie având scop educaţional iar cei care distribuie un curs doresc să împărtăşească acele informaţii pentru a susţine procesul de învaţare şi pentru a aduce un plus de informaţie pe subiectul respectiv. Accentul cade pe a acumula cât mai multe cunoştinţe.

2.2. Obiective secundare

Pe lângă acest obiectiv principal de a construi o aplicaţie care asigura o comunicaţie în timp real, trebuie avute în vedere şi celelalte cerinţe funcţionale ale proiectului pentru a asigura performanţa şi eficienţa acestuia.

Unul dintre obiectivele secundare este posibilitatea utilizatorilor de a se înregistra pe site şi posibilitatea modificării profilului în funcţie de rolul acestuia, adică student sau lector. Trebuie specificat faptul că, în acesta aplicaţie un lector nu trebuie să fie neapărat cadru didactic în cadrul unei facultăţi sau şcoli. El trebuie să aibă cunoştinţele necesare pentru a aborda o anumite temă şi pentru a răspunde la întrebările aduse de către urmăritori. Studentul poate fi orice persoană care fie are studii superioare sau este doritor de a afla mai multe legate de un anumit subiect. O persoană care se înregistrează pe site trebuie să îşi cunoască capacităţile şi să ştie în ce rol se încadrează. Odată ales un rol acesta nu mai poate fi schimbat, deoarece cerinţele pentru cele două roluri sunt diferite. Înregistrarea pe site este permisă folosind un cont pe care utilizatorul deja îl posedă, cum ar fi un cont de Google, Facebook, Linkedin sau Yahoo. Prin acest tip de autentificare, securitatea contului este deja asigurată în cadrul acestor aplicaţii care o au implementată. Dacă un vizitator pe site nu îşi creeaza cont, el nu va putea vedea nici un curs, ci va putea vedea doar informaţii de descriere ale site-ului şi care este rolul acestuia.

Un alt obiectiv important de luat în considerare este asigurarea securităţii datelor şi alte informaţiilor cu caracter personal, dar şi informaţiile care ţin de planificarea unui eveniment şi curs. Dacă accesul în sistem este realizat folosind Google, Facebook, sau Yahoo, utilizatorul are certitudinea că datele personale, notate pentru crearea acelor conturi rămân confidenţiale şi sunt strict secrete chiar şi pentru administratorul site-ului de live stream. Doar email-ul este vizibil pentru a putea fi contactat pe acesta în legătură cu activităţile ce au loc pe site. Apoi securitatea trebuie aplicată şi pe datele ce ţin de construirea unui eveniment, adică trebuie asigurat faptul că nimeni nu poate să modifice

datele unui eveniment şi astfel să impiedice desfaşurarea corectă a acelui live stream. Informaţiile referitoare la evenimente ce urmează să aibă loc sunt păstrate în siguranţă într-o baza de date la care doar administratroul sistemului are acces.

Un alt obiectiv care se doreşte a fi atins constă în crearea de evenimente. Acestea trebuie să fie foarte bine determinate de tema şi data aleasă şi descrierea acestora trebuie să reflecte foarte bine ceea ce live stream-ul va fi. Atunci când un lector vrea să anunţe desfăşurarea unui live stream în care va susţine o anumită temă, acesta trebuie să fie foarte clar în legătură cu tema abordată, limba vorbită şi mai ales data, adică ziua şi ora în care acel live va avea loc. Odată creat un eveniment, persoană care l-a creat trebuie să fie pregătită în ziua şi la ora respectivă pentru a deschide live streaming-ul. Lectorul v-a putea vedea pe tot parcursul streaming-ului numărul de oameni care urmăresc cursul şi întrebările care sunt puse în zona de chat astfel va putea răspunde la acestea când încă este la punctul respectiv. Utilizatorii se pot înscrie la oricâte evenimente doresc, indiferent dacă vor fi prezenţi în urmărirea live-ului sau nu, şi pot de asemenea să şteargă un eveniment din lista lor de evenimente.

Un scop al aplicaţiei este permiterea accesării acesteia de către oricine, aceasta presupunând menţinerea unei interfeţe simple şi uşor de înţeles pentru toţi utilizatorii. Întreg conţinutul trebuie conceput astfel încat să fie înţeles de orice persoană indiferent de vârstă.

Pe lângă cerinţele funcţionale ce reflectă funcţionalităţile oferite de sistem, sunt importante şi cerinţele non-funcţionale ce reflectă calitatea comportamentului sistemului, în atingerea obiectivului final. Am văzut deja că securitatea datelor este foarte importantă în orice sistem, mai ales într-unul în care utilizatorul trebuie să ofere date cu caracter personal. Pe lângă aceasta mai sunt şi alte cerinţe non-funcţionale care se aşteaptă a fi integrate în sistem. Performanţa este una dintre aceste cerinţe. Ea este reflectată de cât de bine sunt implementate cerinţele funcţionale şi care este timpul de răspuns pentru acestea. Dar pentru ca sistemul nu este o aplicaţie desktop ci o aplicaţie web, trebuie luată în considerare şi conexiunea la Internet atunci când se stabileşte timpul de răspuns. Scalabilitatea este şi ea un obiectiv care influenţează modul în care este construit întreg sistemul. Acesta trebuie proiectat si implementat având în vedere şi posibile dezvoltări, adică posibilitatea integrării de noi cerinţe funcţionale dar şi numărul de utilizatori pe care aplicaţia îi va suporta. Modul în care este structurat sistemul va determina cât de eficient este şi cât de stabil rămâne, dacă au loc modificări în cerinţele funcţionale.

Capitolul 3. Studiu Bibliografic

3.1. Concepte

Transmiterea unui video live pe Internet, este o modalitate foarte comună de comunicare în zilele noastre. Ceea ce nu se ştie însă este, care este perspectiva de realizare şi ce tehnologii presupune. De aceea pentru a înţelege cum funcţionează un live stream este importantă definirea conceptelor care îl alcătuiesc.

3.1.1. Live Streaming

Conceptul de live stream are următoarea definiţie, din limba engleză: trimiterea şi acceptarea unui mesaj multimedia, video şi audio prin Internet. Iar definiţia dată în dicţionarul Cambridge are următorul enunţ: "to broadcast video and sound of an event over the Internet as it happens", care tradus spune: a transmite un video cu sunet al unui eveniment, prin Internet, eveniment care are loc în momentul difuzării. Streaming-ul se referă la capabilitatea de a urmări "live" un video pe Internet în loc de a-l descărca. Această difuzare live se realizează având o camera video şi un calculator cu acces la Internet pentru a putea difuza şi transmite live-ul spre potenţiali urmăritori. Pentru ca legătura dintre cel care crează live-ul şi urmăritori este directă, această legătură se realizează prin Peercasting.

Peercasting reprezintă o metodă de distribuţie multicast (one-to-many sau many-to-many) de stream-uri, de obicei audio şi/sau video prin Internet printr-o tehnologie peer-to-peer. O reţea peer-to-peer reprezintă o arhitectură de aplicaţie distribuită în care sunt partiţionate sarcinile de lucru la diferitele staţii (peers) conectate în reţea.

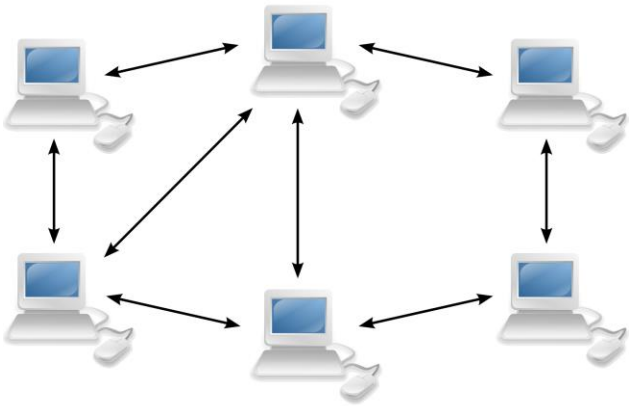

Figura 3.1. Arhitectura unei retele peer-to-peer[1]

Într-o reţea peer-to-peer nevoia de server central care să "servească" este eliminată şi toate staţiile conectate comunică direct. Toate staţiile au aceleaşi privilegii, fiind participanţi echipotenti în aplicaţie, ceea ce înseamnă că în această aplicaţie, un urmăritor poate deveni la rândul sau o persoană care crează şi distribuie un live.

Multicasting-ul reprezintă un grup de comunicare, cu o distribuţie one-to-many sau many-to-many, în care informaţia este transmisă simultan la un grup de calculatoare destinaţie. Comunicarea în grup poate să fie un multicast cu un nivel aplicaţie sau o reţea

asistată, unde distribuţia multicast face posibilă pentru sursă trimiterea eficientă a informaţiei, întregului grup într-o singură transmitere. Nivelul aplicaţie (Application layer), reprezintă o abstracţie care specifică protocoalele împărţite şi metodele de interfaţă folosite de calculatoarele gazdă în comunicarea în reţea. Cu ajutorul acestui nivel abstract se pot ascunde detalii de implementare ale unor seturi particulare de funcţionalităţi, permiţând separarea conceptelor, să faciliteze interoperabilitatea şi independenţa faţă de platforma folosită.

3.1.2. Real-Time Communication

Conform dicţionarului explicativ a comunica înseamnă a face cunoscut, a da de ştire, a înştiinţa, a spune, a transmite ceva cuiva verbal sau scris. Comunicarea în timp real reprezintă cel mai folosit mod de a transmite informaţii, şi cu ajutorul tehnologiei existente, această comunicare se poate realiza chiar şi la o distanţă foarte mare, permiţând transmiterea de fişiere multimedia sau audio în timp real.

Trăim într-o perioada în care ne aşteptăm să găsim informaţia unde dorim, în orice moment şi să primim răspunsurile când avem nevoie. Autoservirea, social media, live chat, conferinţele video şi o mulţime de alte feluri de comunicare sunt toate posibile în ziua de azi datorită analizării şi dezvoltării comunicaţiilor în timp real.

Comunicarea în timp real presupune distribuirea informaţiilor şi interactionarea cu alţi oameni printr-o conexiune pe reţea, ca şi cum toate acestea s-ar întâmplă faţă în faţă. Aplicaţiile care folosesc acest concept pot fi găsite în orice industrie virtualizată: centre de contact, servicii financiare, firme juridice, sănătate, educaţie etc.

Există câteva concepte care contribuie la creşterea numărului de aplicaţii în comunicaţii în timp real [2]. WebRTC sau Web-based Real-Time Communications este o tehnologie HTML5 care livrează conţinut cu garanţia implementării unor comunicaţii în timp real: se folosesc conexiuni peer-to-peer pentru a permite comunicarea fără plugin-uri prin intermediul unui browser.

O altă noţiune importantă în dezvoltarea acestui concept este mobilitatea: avem acces la aceeaşi informaţie, funcţionalitate cu aceeaşi caracteristică şi de pe dispozitivele mobile cât şi de pe calculatoarele desktop. Capabilitatea de a completa task-uri şi de a comunica de oriunde în orice moment, a adus o creştere majoră relaţiei de comunicare în timp real peste Internet. De aceea este important că cei care asigură serviciile de Internet să ştie cum să transforme reţelele IP în termeni de scalabilitate, securitate şi eficienţă pe măsură ce acest tip de comunicaţie se măreşte, şi să ştie cum să facă serviciile disponibile pe orice dispozitiv şi pe reţele atât publice cât şi private.

Creşterea mobilităţii a ajutat la dezvoltarea unui alt concept: serviciul over-the-top (OTT). Serviciile OTT rulează peste o reţea şi nu sunt oferite de operatorul de reţea (ex. Hulu, Netflix, Skype, WhatsApp). Aceste aplicaţii profită de avantajele comunicării prin Internet şi de capabilităţile pe care dispozitivele din ziua de azi le oferă.

3.1.3. E-Learning

Prin definiţia sa, e-Learning reprezintă o tehnică de predare-învăţare, construită pentru a furniza materiale de studiu într-o manieră potrivită pentru a fi asimilate toate informaţiile importante de către studenţi.

Conceptul de e-learning a apărut în urma dorinţei de a avea acces la o informaţie şi de a studia afară din sala de clasă. Acest mod de învăţare a ajuns să fie forma

principala de educaţie la distanţă [3]. Un model conceptual de nivel înalt al unui sistem e-learning este prezentat in figura următoare:

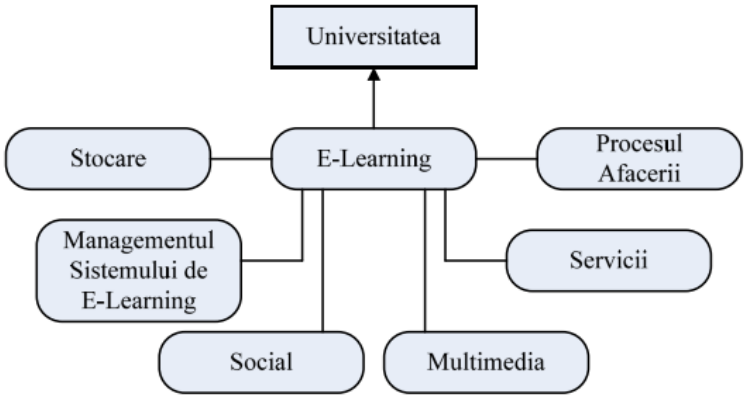

Figura 3.2 Model conceptual pentru un sistem e-learning [4]

Educaţia sau învăţarea reprezintă un ansamblu de măsuri aplicate în mod regulat cu scopul de a forma şi dezvolta însuşiri intelectuale sau morale, pentru a continua educarea membrilor societăţii. Acest fenomen social de transmitere a cunoştinţelor de la o generaţie la alta şi de a studia, poate fi de mai multe tipuri [5]:

Educaţia formală este acea educaţie metodică, intenţionată şi predată de specialişti pe anumite domenii instituţionale, sociale, juridice (lectori, profesori, învăţători etc.). Acest tip de predare-învăţare este elaborat în cadrul unor instituţii specializate în vederea formării unor cunoştinţe care duc la formarea unei profesii sau meserii. Educaţia formală se formează pornind de la şcoală primară şi până la universitate, şi în plus faţă de aceste studii academice, include o diferită gamă de programe de specializare pentru o pregătire profesională. Această formare este importantă deoarece ea uşurează accesul la valorile culturii, ştiinţei, literaturii şi la experienţă social-umană.

Educaţia nonformală are la baza acţiunile organizate în afară sistemului formal fiind considerată complementară cu educaţia formală în ceea ce priveşte conţinutul şi modalitatea de realizare. Acest tip de educaţie este adaptiv nevoilor individului şi situaţiilor speciale pentru acesta, cu scopul de a maximiza învăţarea şi cunoaşterea. Educaţia nonformală este centrată mai mult pe procesul de învăţare decât pe cel de predare, iar activităţile luate în considerare sunt flexibile şi variate în funcţie de interesele participanţilor, dar şi de aptitudinile şi aspiraţiile lor. Actualizarea informaţiilor este rapidă menţindu-se constant interesul publicului din diferite specializări. Această educaţie accentuează importanţa educaţiei care se petrece dincolo de spaţiul formal din sistemul de învăţământ.

Educaţia informală cuprinde totalitatea informaţiilor pe care o persoană le dobândeşte în mod neintenţionat, incidental din multitudinea de activităţi zilnice. Trăsătură caracteristică a acestei educaţii este spontaneintea modului în care un context poate avea un impact relevant şi semnificativ pentru subiect.

Această aplicaţie a fost creată cu scopul de a se putea oferi o platforma de a transmite informaţii cu scopul de a educa sau de a stimula procesul de învăţare. Prin cursurile desfăşurate pe acest site, participanţii vor avea şansa să descopere subiecte noi

pe care în trecut nu le considerau interesante sau poate nu ar fi avut altfel ocazia să le descopere. Subiectele abordate sunt complexe şi extinse pe o multitudine de domenii, toate având scopul de a expune detalii noi şi interesante legate de acele subiecte.

Această metodă de educaţie este nonformală, deoarece modul de predare-învăţare nu este supus nici unui standard de organizare si structurare, găsit în cadrul instituţiilor de învăţământ. Cursurile sunt create cu scopul de a informa şi de a ajută la dezvoltarea individului şi la construirea unei vieţi de calitate pe plan personal, profesional şi social. Educaţia nonformală răspunde nevoilor de învăţare difersificată pentru îmbunătăţirea cunoştinţelor.

3.2. Sisteme Similare

Live Streaming-ul şi comunicarea în timp real sunt două concepte foarte cunoscute şi implementate folosind o mulţime de alte tehnologii, diferite faţă de cea aleasă în acest proiect, fiind folosite în diferite domenii (ştiri, educaţional, medical, tehnologic, sport). Deşi există atât de multe sisteme care se bazează pe aceste concepte, când vine vorba de cerinţele funcţionale implementate, acestea sunt diferite. Printre sistemele existente, cele similare cu aplicaţia de faţă sunt:

3.2.1. TokBox

TokBox [6] este un sistem interactiv de broadcast pentru emisiuni TV, care permite o conexiune directă între cei care urmăresc emisiunea sau programul de ştiri cu prezentatorii. În această aplicaţie există un moderator care face legătură dintre urmăritori şi prezentator, dar legătură dintre aceştia se poate face doar video, nu există zona de chat.

Avantajele aplicatiei TokBox din punctul de vedere al comunicării sunt:

- Înlătură bariera dintre prezentatori şi cei care urmăresc ştirile;
- Se poate include foarte uşor în diferite website-uri sau aplicaţii;
- Ofera o interfaţă modelabilă;
- Inregistrările dupa terminarea transmiterii live pot fi distribuite mai departe.

Dezavantajele acestei aplicaţii sunt:

- Numai un participant poate să intre în direct o dată;
- Nu există mijloc scris (chat), doar live;
- Implementat numai pentru broadcast-uri de ştiri.

3.2.2. StreamTorrent

Un alt sistem asemănător celui dorit să fie implementat în această lucrare este StreamTorrent [7], o aplicaţie bazată pe o arhitectură peer-to-peer, care implementează de asemenea un live stream. StreamTorrent se face diferit prin faptul că asigură robusteţe, eficienţă şi scalabilitate combinând diferite strategii. În tabelul următor sunt prezentate diferenţele majore între aplicaţia în dezvoltare şi StreamTorrent. Prin această comparaţie se poate observă că există mai multe posibilităţi de a implementa un sistem care să aibă ca sursă principală de comunicare un live stream. Acest sistem este cel mai apropiat ca şi mod de implementare, de aplicaţie dezvoltată în acest proiect.

Tabel 3.1 Comparaţie StreamTorrent şi CourseStream

	StreamTorrent	CourseStream
Protocoale	Comunicare peer-to-peer bazată pe protocoale pull-based şi file sharing.	WebRTC şi EasyRTC
Implementare	Java	Node.js, Angular.js
Cerinţe funcţionale	Utilizatorii pot comunica doar audio şi video. Există un al trei-lea utilizator pentru playback.	Dupa logare utilizatorul poate crea un live, poate urmari unul, poate interacţiona direct sau în scris cu cel care ţine live-ul.
Cerinţe non-funcţionale	Robustete, eficientă, scalabilitate.	Securitate, accesibilitate, performanţă, portabilitate.
Descriere	StreamTorrent Player este o aplicaţie media, avand stream-urile puse în fisiere XML şi gazduite pe orice server HTTP.	CourseStream este o aplicaţie de chat live care se bazează pe comunicare video şi audio, dar şi prin mesaje scrise.

3.2.3. XirSys

XirSys [8] este o platformă care asigură infrastructura necesară pentru a produce aplicaţii bazate pe WebRTC într-o reţea self-scaling, reduntantă, şi globală. XirSys asigură:

- Servicii WebRTC .
- Implementare cu webSocket, construită pe serverele STUN/TURN bazate pe WebRTC.
- Framework de JavaScript pentru WebRTC şi acceptarea unor API-uri simple pentru video, audio şi conexiuni prin canale de date si peste.

Comunicarea în XirSys are loc prin protocolul ICE (Interactive Connectivity Establishment). ICE reprezintă un protocol pentru conectarea unei perechi de staţii (peers), cum ar fi doi clienţi care vor să comunice printr-un video chat. ICE încearcă să conecteze cei doi peers direct unul cu celălalt la cea mai joasă frecvenţa prin UDP. Dacă această încercare cu STUN (Session Traversal Utilities for NAT) eşuează din cauza unor restricţii NAT (Network Address Translator) sau firewall, ICE foloseşte serverul intermediar TURN (Traversal Using Relay NAT). Cu aceste cerinţe din partea serverului este critic pentru unele servicii şi aplicaţii să se desfăşoare pe o instrastructura de server

WebRTC care garantează performanţă, fiabilitate, securitate şi scalabilitate, dar XirSys asigura exact aceste lucruri.

3.2.4. Apizee

Apizee [9] este o platformă SaaS (Software as a Service) care permite comunicaţii în timp real atât pe web cât şi mobile. Soluţiile aduse de această ajută întreprinzătorii şi dezvoltatorii de aplicaţii să îşi îmbunătăţească colaborarea de afaceri şi să dezvolte interacţiunile cu clienţii. Această platformă permite comunicarea prin chat şi video chat, dar permite şi transferul de diferite fişiere şi chiar şi distribuirea ecranului de la calculator. La fel ca şi aplicaţia în dezvoltare, Apizee este o tehnologie care rulează în browser şi este bazată pe WebRTC.

Pentru a vede mai bine asemănările şi diferenţele între toate sistemele descrise, între modurile acestora de a permite comunicarea, s-a realizat un tabel de comparaţie:

Tabel 3.2 Comparaţie sisteme similare

	CourseStream	TokBox	XirSys	StreamTorrent	Apizee
Comunicare video + audio	✓	✓	✓	✓	✓
Comunicare audio	X	X	✓	X	X
Chat	✓	X	X	X	✓

Pe lângă aceste aplicaţii care sunt asemănătoare cu aplicaţie care se doreşte a fi dezvoltată în această lucrare, la nivel mondial există câteva sisteme care sunt mult mai avansate în ceea ce priveşte comunicarea şi sunt deja platforme foarte cunoscute şi folosite. Printre acestea putem aminti platformele: Twilio, MIT OpenCourseWare, Coursera şi Udacity. Ultimele trei sunt platforme pe care sunt publicate diferite cursuri predate la anumite universităţi şi instituţii din America.

Twilio [10] este o platforma de comunicare în cloud, un serviciu dezvoltat în San Francisco, California. Aplicaţia permite dezvoltatorilor de software să primescă şi să efectueze apeluri telefonice şi mesaje text folosind API-uri de servicii web. Serviciile oferite de Twilio sunt accesate prin HTTP şi sunt plătite după folosinţă.

Recent această aplicaţie şi-a unit forţele cu compania de telefonie mobilă T-Mobile pentru a ajută dezvoltatorii să construiască aplicaţii care folosesc date celulare. Conceptul numit Twilio Programmable Wireless [11], doreşte permiterea noilor generaţii de comunicări software să folosească cartelele SIM din dispozitivele mobile. În acest fel flexibilitatea aplicaţiilor creşte deoarece nu va mai exista dependenţa de Wi-Fi sau Bluetooth, sau arii de reţele mici.

MIT OpenCourseWare [12] este o platformă în care sunt publicate toate cursurile predate în cadrul institutului MIT (Massachusetts Institute of Technology), cu scopul de a fi accesibile pentru toată lumea. Prin această platformă, lectorii îşi pot îmbunătăţii cursurile, studenţii îşi găsesc resurse adiţionale, iar cei care sunt auto didacti îşi îmbunătăţesc cunoştinţele. Cursurile predate au subiecte vaste, de la Business, Artă şi Educaţie până la Medicină, Stiinţe, Matematică şi Inginerie.

Coursera [13] reprezintă o platformă care asigură acces universal la cursuri de specialitate, de ştiinţă, calculatoare, business chiar şi cursuri despre dezvoltarea personală. Fiecare curs din Coursera este predat de către oameni de specialitate de la mai multe universităţi şi instituţii din America. Cursurile includ o înregistrare video, notare şi teme de revizuit existând şi forum-uri pentru discuţii. La terminarea unui curs întreg se poate obtine si un certificat de specializare.

Udacity [14] este o organizaţie educaţională în care există cursuri diferite mai ales de programare şi computer science, concepută pentru a fi accesată de un număr nelimtat de persoane. Cursurile sunt gratuite, trebuie plătit doar atunci cand o persoană doreşte să fie ajutată şi să aibe un îndrumator. Această platformă este concentrată pe cursuri ce ţin de tehnologii, iar cele mai multe dintre cursuri durează luni sau chiar ani pentru a completa meterialele de studiu în profunzime.

Capitolul 4. Analiză şi Fundamentare Teoretică

4.1. Tehnologii

Tehnologiile reprezinte un ansamblu de procese, metode şi funcţii create cu scopul de a obţine un anumit produs. În această lucrare produsul care se doreşte a fi implementat este o aplicaţie web care să permită transmiterea unui live stream. Conceptul care stă la baza acestei aplicaţii este comunicarea în timp real, iar tehnologia prin care se implementează acest concept este WebRTC.

4.1.1. WebRTC

WebRTC (Web Real-Time Communication) [15] reprezintă o colecţie de protocoale de comunicare şi API- uri (Application Programming Interface) care permit comunicarea în timp real peste conexiuni peer-to-peer. Această tehnologie include aplicaţii de tip video conferinţă, transfer de fişiere, chat sau partajare desktop fără nevoia de a adaugă plugin-uri. WebRTC este un software Open Source, standardizat de către World Wide Web Consortium (W3C) şi Internet Engineering Task Force (IETF).

Din punct de vedere arhitectural, WebRTC este strucurat astfel:

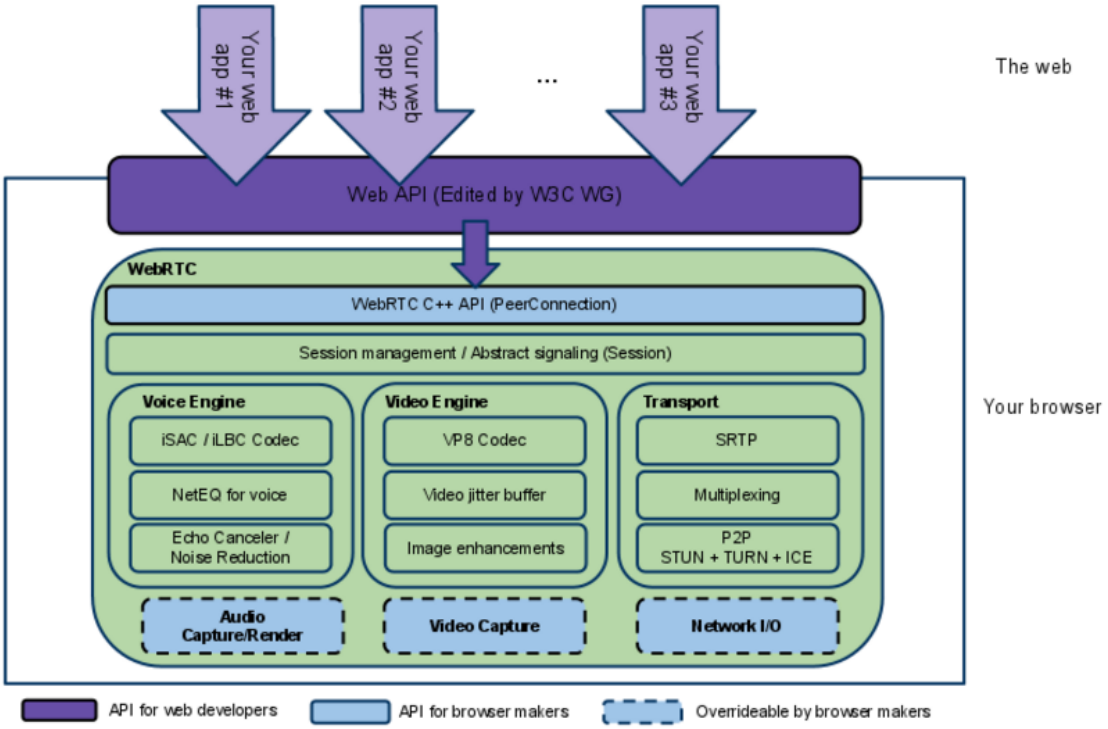

Figura 4.1. Arhitectura WebRTC [16]

Arhitecura este împarţită în 3 mari categorii :

A. Browser API: Pentru a asigura dezvoltatorilor abilitatea de a trimite şi de a primi stream-uri audio şi video direct din browser, browser-ele trebuie să

fie îmbunătăţite cu capabilităţi de a controla dispozitivele audio şi video locale în dispozitivul de calcul în care rulează browserul. Aceste capabilităţi sunt expuse către dezvoltatori printr-o interfaţă de programare bine definită (API).

B. Web application (Aplicaţia web): Modul tipic de a rula o aplicaţie web pentru un utilizator este să descarce un Javascript din serverul web. Acest script va rula apoi local în sistemul utilizatorului dar va interacţiona cu serverul web pentru a executa logica aplicaţiei. Serverul web poate instrui acel script pentru a execute anumite acţiuni pentru ca mai apoi script-ul să poată trimite un mesaj de informare (feedback) către serverul web.

C. Serverul web: Serverul asigură script-urile necesare pentru utilizatori şi execută logica aplicaţiei.

WebRTC foloseşte Real-Time Protocol pentru a transfera audio şi video. Real-Time Transfer Protocol este un protocol de transport în timp real prin intermediul căruia se pot transmite informaţii de tip media (sunete, imagini, text) printr-o reţea de telecomunicaţii. Protocolul RTP are la baza ideile propuse de Klark şi Tenenhauzen şi are ca şi scop transmiterea datelor în timp real. Faţă de acesta se precizează tipul câmpului de date, se numerotează pachetele şi se înregistrează reperul de timp şi se monitorizează transmiterea datelor. Aplicaţiile folosesc de obicei RTP implementat peste UDP (User Datagram Protocol), pentru ca să se poată folosi de posibilitatea de multiplexare şi controlul checksum. RTP permite transmiterea concomitentă pe adrese diferite, dacă multicast-ul este posibil la nivel de reţea.

Scopul construirii platformei WebRTC este de a permite dezvoltatorilor să scrie aplicaţii multimedia în timp real pe web, fără alte plugin-uri, descărcări şi alte instalări de programe, aplicaţii care să meargă pe platforme multiple şi în mai multe tipuri de browser. De reţinut faptul că o comunicare peer-to-peer nu înseamnă că nu există un server ci doar faptul că datele nu trec prin el. Un server este necesar pentru ca doi utilizatori să facă schimb de informaţie (pentru a ştii unde se află pe Internet), astfel încât să se poată stabili conexiunea peer-to-peer. Informaţia folosită pentru a stabili legătură peer-to-peer se cheamă signaling, iar serverul se numeşte signaling server. Deşi WebRTC nu specifice ce fel de signaling server sau protocol să fie utilizat, WebSockets este protocolul cel mai folosit împreună cu această tehnologie (pe lângă polling, sau email).

WebSocket reprezintă un protocol de comunicare, care asigura o legătură full-duplex peste o singură conexiune TCP. Protocolul WebSocket este proiectat pentru a fi folosit în browsere web şi servere web şi poate fi folosit de orice aplicaţie client sau server. Protocolul are la bază protocolul TCP (Transmission Control Protocol) dar este independent de acesta. Acest protocol face posibilă o interacţiune între un browser şi un server, existând un transfer în timp real mult mai rapid de la şi pentru server. Acest lucru este posibil asigurând o cale standard pentru server pentru a trimite conţinut la browser fără a fi solicitat de către client, şi permiţând ca mesajele să fie trimise înainte şi înapoi în timp ce conexiunea este menţinută deschisă. În acest fel o cale bi-direcţională de comunicare poate să aibă loc între browser şi server.

O aplicaţie WebRTC implementează în general următorii paşi:

- Se obţine acces la camera şi microfonul locale în forma unui stream media.

- Se stabileşte o conexiune la un signaling server.
- Se initializează un apel cu o persoană din alt browser
- Se conectează cu stream-urile video prin etichetele video.

Folosind framework-ul EasyRTC, unii din aceşti paşi pot să fie legaţi într-un singur apel, simplificând munca dezvoltatorilor.

4.1.2. EasyRTC

EasyRTC [17] este un framework pentru comunicarea în timp real audio, video şi transfer de date direct între browserele web.

EasyRTC reprezintă o tehnologie open source care este alcătuită din două compomente principale:

- O librarie de browser client, scrisă în JavaScript. Acest client gestionează semnalizarea (signaling) şi în mare masură, izolează aplicaţiile de modificările în curs de desfăşurare în API-urile WebRTC.
- Un server de semnalare bazat pe Node.js.

Cu ajutorul acestor componente se poate realiza o aplicaţie simplă de video conferinţă cu chat integrat, sau o aplicaţie de distribuire a fişierelor.

Serverele EasyRTC pot fi utilizate în propria infrastructură sau pot fi incluse în alte aplicaţii. Serverele pot fi instalate chiar şi în cloud, oferind organizaţiilor control asupra centrelor de date utilizate. EasyRTC asigură confidenţialitatea transferului de mesaje şi de fisiere folosind modul de comunicare peer-to-peer.

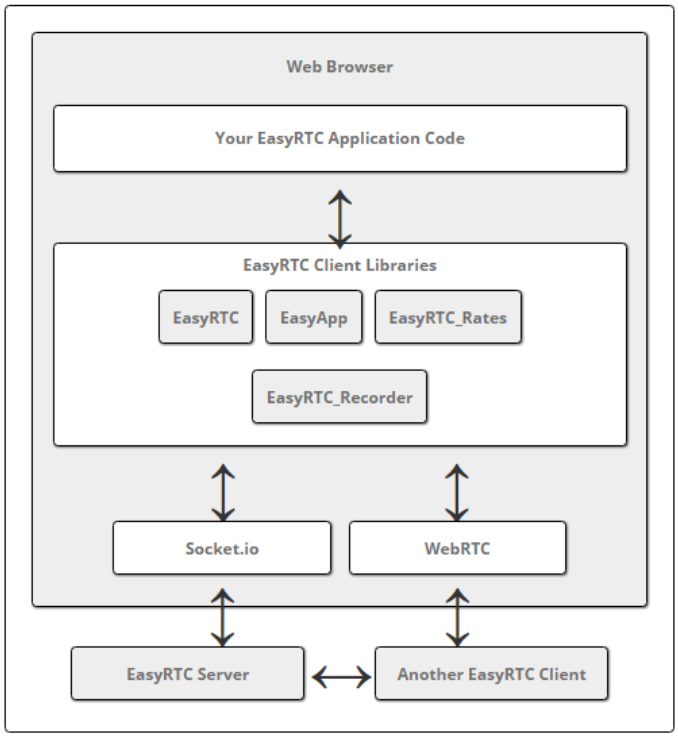

Figura. 4.2 Arhitectura EasyRTC

Terminologia folosita:

- **Video track** - obiect care reprezintă rezultatul de ieşire al unei camere
- **Audio track** - obiect care reprezintă rezultatul de ieşire al unui microfon
- **Media Stream** - obiect care serveşte ca un pachet pentru a ţine unul sau mai multe video track-uri şi/sau audio.
- **Data channel** - obiect folosit pentru a trimite string-uri sau date binare de la un peer la altul.
- **Peer connection** - obiect care serveşte ca şi portal între 2 peer-uri permitând streamuri media şi data channels sa fie distribuite în reţea.
- **Easyrtcid** - id pentru un peer care este unic în serverul EasyRTC.
- **Call** - ca şi verb, reprezintă acţiunea de a stabili o conexiune peer to peer astfel incât media stream-urile să poată fi trimise. Ca şi substantiv reprezintă momentul în care un peer comunică cu un alt peer.

EasyRTC poate fi intregrat atât în aplicaţii web cât şi sisteme care necesită o construcţie hardware pentru funcţionare. Un exemplu este construirea unui robot cu o placă Intel Edison, având ca sursă principală de comunicare, tehnologia EasyRTC. În proiectarea aplicaţiei se foloseşte Node.js, placa de calcul Intel Edison cu kit, două motoare cu drivere pentru roţi şi un suport pentru telefonul mobil.

Un alt sistem în care tehnologia WebRTC s-a integrat este PrioPhone [18] creat de Priologic Software. Această aplicaţie reprezintă un softphone de generaţie următoare, care utilizează tehnologia WebRTC, care este un standard în curs de dezvoltare pentru comunicaţiile video, audio şi transfer de date. Concepută pentru a extinde comunicarea prin reţeaua de telefonie mobilă, PrioPhone este o aplicaţie web care se foloseşte de cele mai noi versiuni de Chrome şi Firefox şi nu necesită instalare de plugin-uri. Aplicaţia asigura o platforma sigură şi scalabilă între modurile de comunicare prin Internet şi telefonia mobilă. Folosind Internetul că sursă principală de transmitere a informaţiilor se reduc costurile pentru minutele care erau vorbite prin telefon. Pentru a avea acces la aplicaţie trebuie doar să existe o conexiune cu WebRTC Session Controller de la Oracle. Prin acest controller se pot activă funcţii de comunicare pentru pagini web şi aplicaţii mobile.

4.1.3. Node.js

Node.js [19] este o platformă de dezvoltare a aplicaţiilor pe partea de server construită în JavaScript Engine de la Google Chrome, dezvoltat de Ryan Dahl in 2009. Definiţia din documentaţia oficială [20] spune următoarele:

Node.js este o platformă construită în Chrome JavaScript runtime, pentru a crea simplu şi rapid aplicaţii în reţea. Node.js foloseşte un model bazat pe evenimente, fără să blocheze funcţiile de I/O, care permite aplicaţiilor care rulează pe diferite dispozitive distribuite să fie eficiente şi uşoare, perfecte pentru dată intensiv solicitată în timp real.

Node.js = Runtime Environment + JavaScript Library

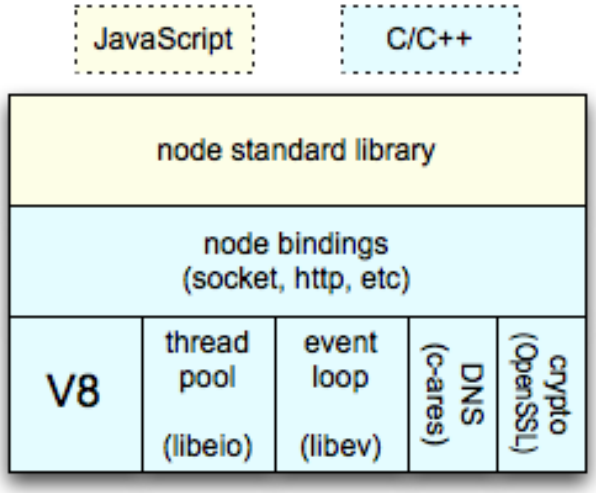

Figura 4.3. Arhitectura Node.js [21]

Cele mai importante caracteristici pentru care Node.js este o alegere pentru proiectanţii software sunt:

- Asincron şi bazat pe evenimente [22] - toate API-urile sunt asincrone deci neblocante. Această înseamnă că tot serverul de Node.js nu aşteaptă niciodată pentru un API să returneze date. Serverul trece la următorul API după ce îl apelează şi un mecanism de notificare de evenimente ajută serverul să primească un răspuns de la apelul unui API precedent.
- Foarte rapid - Fiind construit pe motorul V8 JavaScript de la Google Chrome, librăria Node.js este foarte rapidă în executarea codului. V8 compilează codul de javascript direct în codul maşinii native înainte de a-l execută.
- Un singur Thread dar foarte scalabil - Node.js foloseşte un model cu un singur thread pentru rutarea evenimentelor. Mecanismul bazat pe evenimente ajută serverul să răspundă într-un mod neblocant şi face ca serverul să fie foarte scalabil spre deosebire de serverele tradiţionale care crează un număr de threaduri limitate pentru a răspunde la cereri. Acelaşi program care foloseşte un singur thread poate să asigure un serviciu cu un număr mai mare de cereri.
- Licenţiat - Node.js este eliberat sub licenţa celor de MIT.

Node.js este tehnologia de implementare a părţii de server din această aplicaţie, deoarece platforma EasyRTC este construită pe baza acestei tehnologii. Framework-ul în care este construită partea din aplicaţie cu Node.js este Express.

4.1.3.1. Express

Express este un framework flexibil pentru aplicaţii Node.js care asigură robusteţea unui set de caracteristici pentru aplicaţii web sau mobile. Express [23] se poziţionează deasupra funcţionalităţii serverului Node.js şi simplifică operaţiile cu diferite API-uri şi adaugă noi caracteristici ajutătoare. Express facilitează exacutarea dinamică a paginilor HTML; defineşte un standard extensibil şi uşor de implementat.

Express aduce două mari caracteristici pentru serverul Node.js:

- Adaugă un număr mare de avantaje serverului Node.js de HTTP, abstractizând o mare parte din complexitate.
- Permite împarţirea unei funcţii monolitice care se ocupă cu cererile (requests), în mai multe cereri mai mici care manipulează numai anumite cerinţe. Aceasta împarţire este mult mai uşor de susţinut şi mai modulară.

În figura următoare este prezentat fluxul unei cereri în Express. Ce este scris în cerc reprezinta codul de scris iar, ce este in patrate nu necesita implementarea de catre dezvoltator.

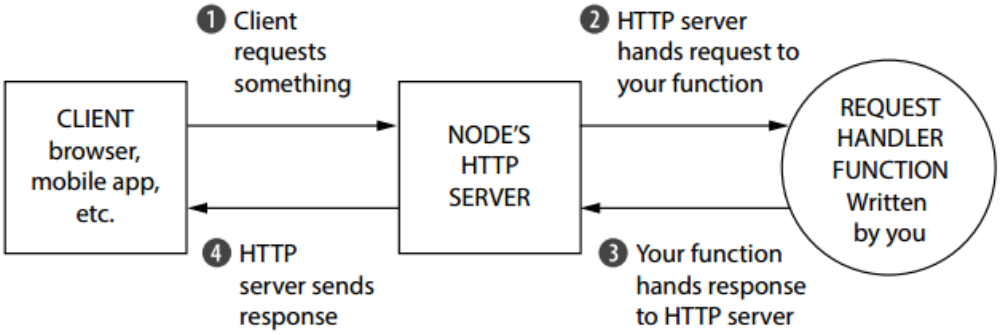

Figura 4.4 Fluxul unei cereri in Express.

4.1.4. Angular.js

Angular.js [24] este un framework structural pentru aplicaţii web dinamice, care permite folosirea limbajului HTML ca şi template şi permite extinderea sintaxei limbajului pentru a exprima componentele aplicaţiei clar şi succint. Integrarea datelor şi injectarea dependenţelor elimină mult din codul care altfel ar trebui scris. Angular.js simplifică dezvoltarea aplicaţiei printr-o abstractizare la un nivel mai înalt prezentată dezvoltatorului. Şi ca orice abstractizare vine cu un cost al flexibilităţii, adică nu toate aplicaţiile sunt construite să se potrivească cu Angular.js, acesta fiind construit bazat pe operaţiile de CRUD (Create, Read, Update, Delete).

AngularJS asigură o soluţie completă pentru partea de interfaţă, dar nu reprezintă o singură piesă în proiectarea părţii client dintr-o aplicaţie web, el fiind piesa care uneşte componente ca DOM (Document Object Model) sau AJAX şi le pune într-o structură bine definită. Acesta este construit pe ideea că un cod declarativ este mai bun decât unul imperativ când vine vorba de construirea unei UI (User Interface) şi scrierea componentelor software împreună, în timp ce un cod imperativ este excelent pentru a exprimă logică de business. Avantajele acestei decizii sunt următoarele:

- Este o idee foarte bună să detaşezi manipularea DOM de aplicaţia logică. Acest lucru îmbunătăţeşte testabilitatea codului.
- Este o idee foarte bună de a privi testarea ca fiind la fel de importantă că şi scrierea codului. Dificultatea testării este afectată de felul în care codul este structurat.
- Detaşarea părţii client de partea de server permite dezvoltarea să progreseze în paralel, şi ambele părţi să poată să fie refolosite separat.

- Este foarte important că framework-ul ales să ghideze dezvoltatorul printr-o călătorie în dezvoltarea aplicaţiei: de la proiectarea interfeţei utilizatori, până la scrierea logicii de business şi testare.
- Întotdeauna este bine ca cerinţele comune să fie efectuate simplu iar cerinţele dificile să fie făcute posibile.

AngularJS eliberează dezvoltatorul de următoarele dificultăţi:

- Înregistrarea apelurilor: această cerinţă îngreunează codul, de aceea înlăturarea ei reduce necesarul de cod javascript care trebuie scris şi face mult mai uşor de văzut ce face aplicaţia.
- Manipularea DOM-urilor HTML programatic: această abordare este piatra de rezistenţă a aplicaţiilor AJAX, dar este greoaie şi predispusă la erori. Prin descrierea declarativă cum interfaţă utilizator ar trebui să se schimbe cum se schimbă şi starea aplicaţiei, se eliberează necesitatea de nivel scăzut din manipularea DOM.
- Manipularea datelor de la şi pentru UI: operaţiile CRUD compensează majoritatea cerinţelor din aplicaţiile AJAX. Fluxul de a manevra data de la server la un obiect intern la un form HTML, permiţând utilizatorului să modifice acel form, validând form-ul, afişând erorile de validare, returnând un model intern şi apoi înapoi la server, crează un şablon pentru scrierea codului. AngularJS elimină aproape tot acest şablon cu detaliile de implementare, lăsând doar codul care descrie fluxul general al aplicaţie.
- Scrierea codului de iniţializare doar pentru a porni aplicaţia: tipic e necesară scrierea unui cod de configurare doar pentru a crea o simplă aplicaţie cu AJAX. Cu AngularJS se poate implementa o aplicaţie mai uşor folosind servicii, care sunt injectate în aplicaţie într-un mod bazat pe injectarea dependenţelor. Acest lucru permite dezvoltarea aplicaţiei mult mai rapid. Iar ca un bonus dezvoltatorul are control total asupra procesul de initializare în testele automate.

Angular.js este potrivit pentru aplicaţia dezvoltată deoarece oferă un mod de administrare şi încarcare dinamică a paginilor HTML, prin conceptul de single page application.

4.1.4.1. Single Page Application

Single Page Applications (SPAs) [25] sunt aplicaţii web care încarcă o singură pagină HTML şi apoi în mod dinamic actualizează pagina în funcţie de cum interacţionează utilizatorul cu aplicaţia. SPA foloseşte AJAX şi HTML5 pentru a crea aplicaţii web fluide şi sensibile la răspuns, fără ca pagina să fie constant reîncărcată. Acest lucru înseamnă că mai multă muncă are loc în partea de client, în JavaScript. După ce prima pagină se încarcă, toată interacţiunea cu serverul are loc prin apeluri AJAX. Aceste apeluri returnează o dată, de obicei în format JSON. Aplicaţia foloseşte data JSON pentru a actualiza dinamic pagina, fără a o reîncarcă. Într-o aplicaţie web tradiţională de fiecare dată când se apelează serverul, acesta trimite o pagină HTML, iar acest lucru declanşează o reîncarcare a paginii în browser.

În figura următoare este prezentată diferenţa dintre fluxul unei pagini tradiţionale şi fluxul unei pagini SPA în relaţie cu serverul.

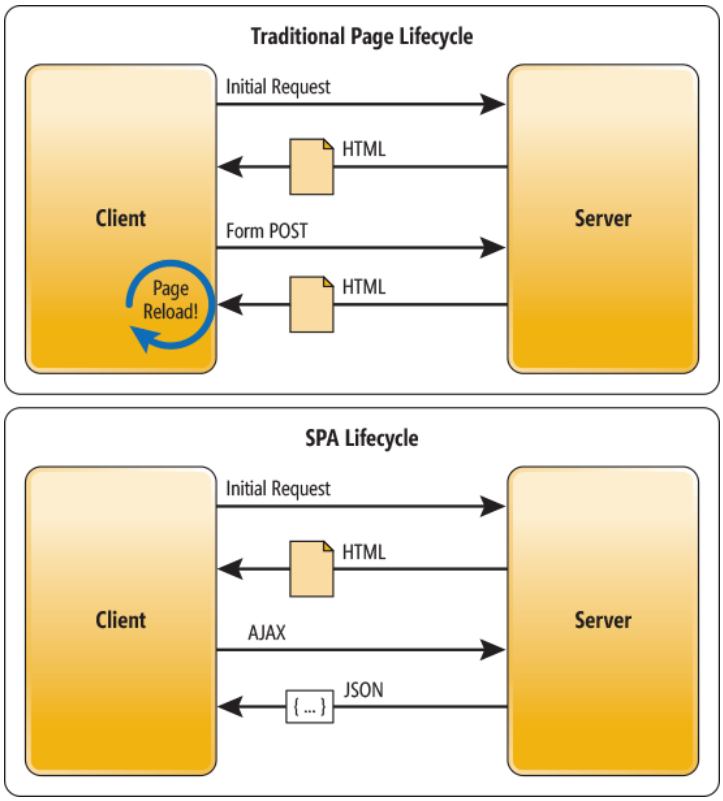

Figura 4.5 Diferenţa între fluxul unei pagini traditionale HTML şi o pagină SPA [25]

Un avantaj evident al acestei abordări este faptul că aplicaţiile răspund mult mai rapid şi uşor fără nevoia de reîncărcare şi retransmitere a paginii. Un alt avantaj este legat de arhitectura aplicaţiei web. Trimiterea unei date ca tip JSON crează o separare între prezentare (HTML markup) şi logica aplicaţiei. Această separare face dezvoltarea pe nivele mult mai flexibilă. Cu această arhitectură partea de client este separată faţă de partea de server, acest lucru făcând posibilă schimbarea uneia dintre părţi fără a o afecta pe cealaltă.

4.1.5. Auth0

Auth0 [26] este un serviciu care abstractizează modul în care utilizatorii se autentifică în aplicaţii. Avantajul acestui serviciu este că aplicaţia dezvoltată poate fi implementată în orice limbaj de programare, deoarece Auth0 are construit un template pentru oricare. Un alt avantaj este că datele de înregistrare ale utilizatorilor sunt confideţiale şi se asigură păstrarea lor în siguranţă.

Authorization Code Grant este un proces în care browser-ul primeşte un cod de autorizare (Authorization Code) de la Auth0 şi îl trimite la aplicaţia web. Aplicaţia va interacţiona cu Auth0 şi va schimbă codul de autorizare pentru un access_token, şi opţional pentru un id_token şi un refresh_token. Aplicaţia va putea apoi să folosească acel access_token pentru a apela API-ul în locul utilizatorului.

Arhitectura sistemului în care autentificarea cu Auth0 se mapează pe un proiect este următoarea:

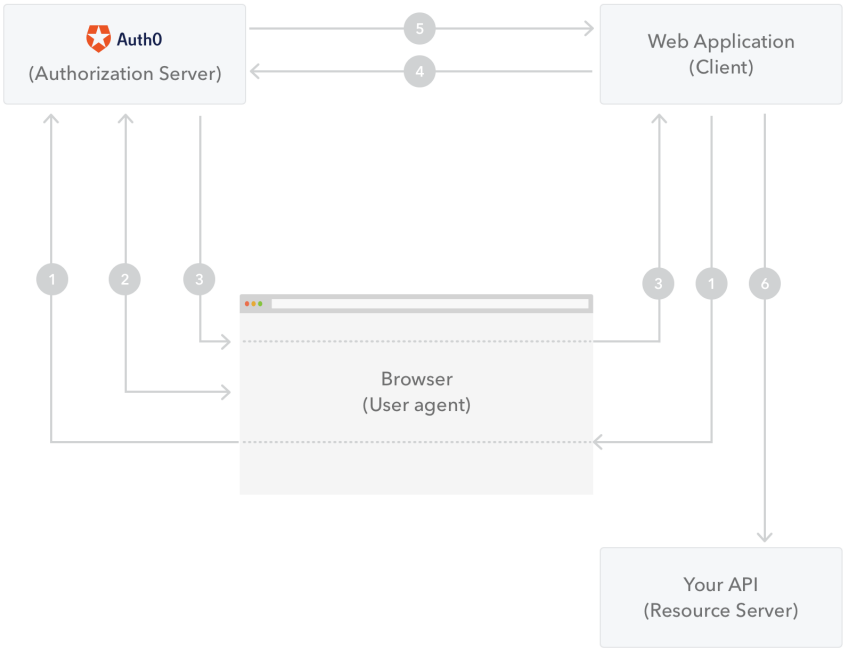

Figura 4.6 Legătura dintre Auth0 şi un API [27]

1) Aplicaţia web iniţiază procesul şi redirectează browser-ul către Auth0, pentru ca utilizatorul să se poată autentifica.
2) Auth0 autentifcă utilizatorul. Prima dată când utilizatorul trece prin această faza, îi va apărea o pagină în care vor fi listate toate permisiunile care îi sunt date ca şi Client.
3) Auth0 redirectează utilizatorul către aplicaţia web cu codul de autorizaţie în querystring (code).
4) Aplicaţia web trimite codul de autorizaţie la Auth0 pentru a primi în schimb access_token-ul.
5) Auth0 autentifică aplicaţia web, validează codul de autorizaţie şi răspunde înapoi cu access_token-ul.
6) Aplicaţia web poate folosi acel access_token pentru a apela API-ul în numele utilizatorului.

4.1.6. HTML5

HTML5 [28] este noua versiune a limbajului HTML, cu elemente, atribute şi comportamente noi reprezentând un set larg de tehnologii care permit conectarea la site-uri şi aplicaţii web mai puternice din punctul de vedere al suportului multimedia. Acest set a fost dezvoltat pentru a fi folosit de toţi dezvoltatorii Open Web, având conceptele clasificate în grupuri bazate pe funcţia acestora.

a) Semantica – permite descrierea mai precisă a conţinutului, având noi elemente pentru descrierea structurii. Pentru a folosi HTML5 se adaugă în prima linie din fişier tag-ul: <!DOCTYPE html> şi se adaugă în <head> setul de caractere folosit <meta charset=”UTF-8”>.Cu HTML5

se restrictionează seturile de caractere la cele compatibile cu codul ASCII şi cele care folosesc cel puţin 8 biti. În acest fel se întăreşte securitatea. Alte elemente noi de secţionare a paginii sunt tag-urile: <article>, <section>, <nav> şi <aside>.

b) Conectivitatea – permite comunicarea cu server-ul în noi şi inovative moduri. WebRTC este o tehnologie care oferă posibilitatea aplicaţiilor web să transmită materiale audio şi video şi să facă schimb de date fără a solicita un intermediar.

c) Offline-ul şi stocarea – permite paginilor web să stocheze date în partea de client local şi să opereze în modul offline mult mai eficient. Prin mecanismul de `application caching`, aplicaţiile web pot să funcţioneze şi offline.

d) Multimedia: conţinutul video şi audio devin o prioritate în Open Web. Tag-ul <video> permite integrearea foarte uşor a unui fişier video folosindu-se atributul `src` care conţine calea spre acel video.

e) Grafica 2D/3D şi efecte - permite o gamă mult mai diversă de opţiuni de prezentare prin adaugarea tag-ului <canvas>.

f) Performanţa şi integrare - asigură o optimizare a vitezei şi o utilizare mai bună a componentelor hardware.

g) Accesul la dispozitiv – permite utilizarea unei game largi de dispozitive de intrare şi iesire.

4.1.7. MySQL

În orice aplicaţie este important ca datele să fie stocate într-o bază de date bine organizată şi bine protejată de orice atac. O bază de date [29] reprezintă un instrument pentru colectarea şi organizarea datelor pentru o perioadă nedeterminată de timp. O bază de date computerizată este un container de obiecte care pot fi organizate în tabele care sunt legate între ele prin diferite relaţii.

În bibliografia de specialitate există o multitudine de baze de date, fiecare organizate şi create diferit pentru a susţine diferite domenii de activitate. Baza de date alesa pentru a stoca datele în acest proiect este MySQL.

MySQL este un sistem relaţional de gestionare a bazelor de date (RDBMS). Un sistem de management al bazelor de date relaţionale (RDBMS – Relaţional Database Management System), reprezintă un sistem de administrare care se bazează pe un model relaţional creat de Edgar F. Codd. Bazele de date permit modelarea lumii reale prin entităţi şi relaţii şi înregistrează date din lumea reală ce posedă o semnificaţie intrinsecă.

Abrevierea SQL vine de la Structured Query Language şi reprezintă un limbaj de programare specific pentru manipularea datelor în sistemele de gestionare a bazelor de date relaţionale. Acesta are ca scop inserarea datelor, actualizarea şi ştergerea lor, modificarea şi crearea schemelor şi controlul accesului la date.

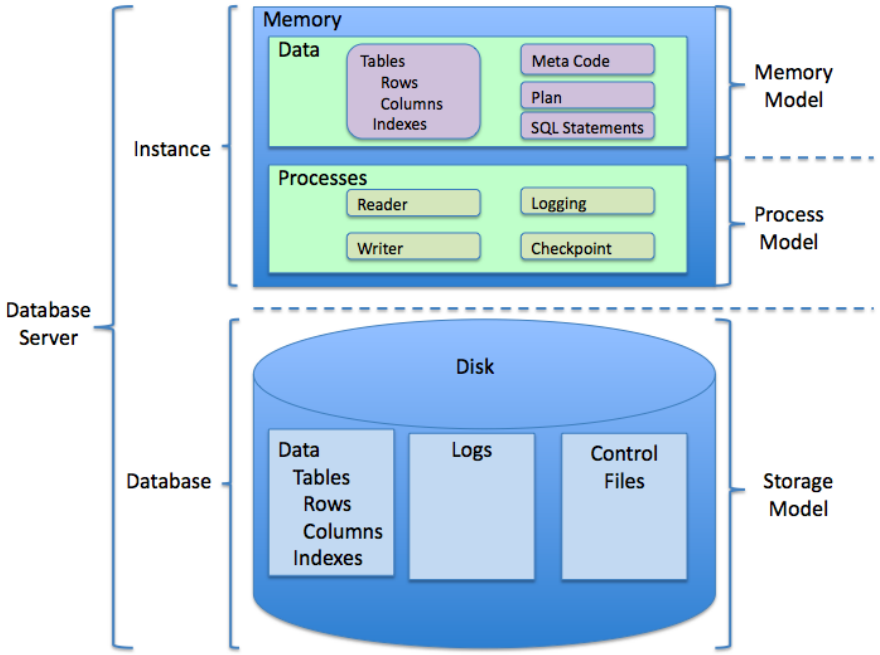

Figura 4.7 Structura generala a unei baze de date [31]

Elementele limbajului SQL sunt:

- Clauza – componenta a instrucţiunilor şi interogărilor.
- Expresia – are efectul de a produce valori scalare sau tabele.
- Predicate - specifică condiţiile care sunt evaluate de SQL conform unei logici, cu scopul de a elimita efectul instrucţiunilor.
- Interogarea – are scopul de a regăsi date după anumite criterii specifice.
- Instrucţiunea – poate avea efect persistent asupra datelor, sau poate controla tranzacţiile, conexiunile sau cursul programului.

4.2. Cerinţe funcţionale

Cerinţele funcţionale sunt baza pe care se construieşte aplicaţia, rolurile fundamentale pe care aplicaţia le va avea, şi modul prin care aplicaţie e unică şi diferită faţă de alte aplicaţii cu aceeaşi temă. Iar dacă unele cerinţele funcţionale sunt posibile a fi la fel cu ale altor aplicaţii atunci implementarea acestora va face diferenţa. Pricipalele cerinţe funcţionale ale sistemului dezvoltat sunt trecute în următorul tabel:

Tabel 4.1. Cerinţe funcţionale

CERINTE FUNCTIONALE	
CF1	Gestiune utilizatori
CF1.1	Înregistrare utilizator
CF1.2	Logare utilizator
CF2	Personalizare profil

CF3	Vizualizare cursuri
CF3.1	Vizualizare cursuri live
CF3.2	Vizualizare evenimente pentru cursuri urmatoare
CF4	Căutare cursuri
CF5	Inscriere la eveniment
CF6	Creare eveniment
CF7	Creare live stream
CF7.1	Pornirea înregistrarii video
CF7.2	Distribuirea live-ului pe Internet
CF8	Comunicarea între utilizatori
CF8.1	Chat pentru comunicarea cu lectorul
CF8.2	Video-chat între lector şi un urmaritor
CF8.3	Trimiterea unui fişier

Prima cerinţă funcţională se referă la posibilitatea utilizatorului de a se autentifica pentru a avea acces la resursele din aplicaţie dar şi pentru a avea permisiunile impuse de sistem. La înregistrarea o persoană are posibilitatea autentificării cu un alt cont creat pe un alt site (ex. Google, Facebook, Yahoo), astfel având toate informaţiile personale gata completate. Înregistrarea şi logarea se face cu ajutorul serviciului oferit de Auth0.

După autentificare şi logare, utilizatorului i se mai cer câteva informartii privind profesia acestuia, adică dacă acesta este un student sau un lector. În funcţie de această alegere sunt departajati utilizatorii pe roluri. Studenţii vor putea doar să urmărească cursuri sau să se înscrie la diferite evenimente, iar lectorii sunt cei care vor putea să creeze video-urile live şi să creeze evenimente pentru acestea. Studentul în acest context reprezintă o persoană care este dornică de a învaţă lucruri noi sau de a le aprofunda pe cele deja ştiute dar nu există pentru el o alternativă de a participa fizic la un astfel de curs. Pe lângă această informaţie utilizatorii vor putea să mai completeze diferite câmpuri privind facultatea din care fac parte, materiile studiate sau diferite subiecte de care sunt interesaţi să afle mai multe, şi limbile vorbite.

Vizualizarea cursurilor este multiplă sau singulară, adică un utilizator (atât studenţii şi profesorii pot urmări orice cursuri – în acest caz nu există nici o diferenţiere), poate să vadă toate cursurile care sunt live în acel moment sau poate să caute după o anumită preferinţă (limba, subiect, autor) un curs care este în desfăşurare şi să între să îl urmărească. Pe lângă cursurile care sunt în desfăşurare, un utilizator poate să caute evenimente care anunţă un anumit curs când va avea loc. Şi pentru evenimente există un filtru de căutare după limba vorbită de autor, subiectul dezbătut şi chiar şi după autorul cursului.

După ce un utilizator s-a decis să participe la o următoare sesiune de live streaming văzând evenimentul creat pentru acesta, el se poate înscrie la curs prin acel eveniment. Odată înscris, evenimentul va fi adăugat în lista de evenimente pe profilul utilizatorului. Dacă până la data evenimentului, un utilizator se hotăreşte să nu mai participe atunci el poate cu uşurinţă să şteargă evenimentul din lista sa.

Crearea unui eveniment presupune completarea unui template cu câmpuri determinate şi salvarea acestui eveniment în baza de date pentru a putea fi găsit şi afişat la secţiunea de evenimente. Câmpurile necesare pentru înregistrarea unui eveniment sunt: numele profesorului care va ţine cursul, data şi ora la care acesta va avea loc, subiectul abordat de lector şi limba în care se va desfăşura cursul. Odată creat evenimentul, profesorul poate să vadă numărul de utilizatori care s-au înscris la acesta. În cazul în care cursul se anulează din anumite motive, şi evenimentul legat de curs se şterge, utilizatorii vor fi înştiinţaţi de acest lucru.

Un curs live va putea fi creat doar de către un utilizator cu rol de lector. Acesta va putea fi creat bazat pe un eveniment care a anunţat acest curs sau la decizia lectorului va putea fi creat direct. Dacă live stream-ul este creat direct, cel care îl crează trebuie să completeze unele informaţii despre acesta pentru a putea fi încadrat într-o anumită categorie la căutare. Distribuirea acestuia pe Internet se face automat, după salvarea acestuia în baza de date. La încheierea cursului, acesta nu rămâne salvat.

Pe parcursul cursului transmis live, utilizatorii care urmăresc acel curs vor avea oportunitatea de a comunica cu lectorul printr-un chat ataşat live-ului. În acest chat studenţii vor putea să adreseze diferite întrebări legate de subietul abordat, lectorului. Acesta va putea răspunde la întrebări, atunci când apar în chat, sau la finalul predării cursului. Pe lângă acest chat, utilizatorii vor putea comunica direct cu lectorul printr-un video chat. Pentru a fi selectat să participe la video chat, urmăritorii vor trebui să se înscrie la acesta, iar la final lectorul va selecta un participant să între într-o discuţie directă cu acesta. De reţinut că nu toţi cei care vor dori să îşi adreseze direct întrebarea vor avea acesta şansă, totul depinde de câte întrebări este dispus lectorul să răspundă.

La finalul cursului, lectorul poate să trimită un fişier, care să conţină ideile principale prezentate în curs. Transferul se face prin mecanismul ‚drag&drop' iar studentul poate să decidă, dacă acceptă fişierul sau nu. Dacă îl acceptă, acel fişier va fi salvat în calculatorul său propriu.

4.3. Cerinţe non-funcţionale

Cerinţele non-funcţionale sunt cerinţele care stabilesc constrângerile sub care trebuie să opereze sistemul. Atributele non-funcţionale reflectă calitatea, comportamentul produsului la execuţie, structura şi organizarea software. Aceste cerinţe trebuie stabilite de la început, pentru a putea fi impus un standard aplicaţiei dezvoltate.

Cerinţele non-funcţionale de baza în această aplicaţie sunt:

a) **Securitatea** reprezintă gradul de rezistenţă şi protecţie necesar împotriva oricărei vulnerabilităţi sau posibile ameninţări. Pentru a asigura securitatea, cel care construieşte aplicaţia trebuie să ia toate măsurile de precauţie posibile pentru a asigura utilizatorii că datele lor nu vor fi compromise şi rămân în orice situaţie confidenţiale. Iar datorită autentificării pe site, utilizatorii pot avea siguranţa că personele

autentificate sub un anumit nume sunt persoanele care pretind a fi. Securitatea trebuie oferită şi în cadrul datelor legate de evenimentele ce au loc pe site. Aceasta trebuie să asigure faptul că informaţiile cu privire la un eveniment nu sunt schimbate.

b) **Performanţa** unui sistem reprezintă analizarea parametrilor care pot îmbunătăţii sistemul (ex timpul de răspuns, rată de procesare, modul de utilizare al resurselor calculatorului, disponibilitatea sistemului, lăţimea de bandă pentru transfer etc). Starea unui sistemul poate fi susceptibilă, adică aceasta se poate schimba şi poate modifica parametrii în mod negativ scăzând performanţa. Performanţa aplicaţiei trebuie să fie asigurată în orice situaţie, adică indiferent de numărul de utilizatori care folosesc aplicaţia în acelaşi timp, aceasta trebuie să ruleze la cele mai bune standarde de timp.

c) **Eficienţa –** se referă la cât de economic se utilizează resursele calculatorului. Eficienţa determină şi afectează performanţa. Această aplicaţie necesită navigarea pe un browser şi o conexiune la Internet, deci pentru un utilizator care doar urmăreşte un curs, resursele folosite sunt puţine. În plus de acestea un utilizator care distribuie un video live, mai este necesară activarea camerei şi a microfonului pentru a transmite informaţiile.

d) **Disponibilitatea sau accesibilitatea** constă în însuşirea de a fi disponibil, convenabil şi uşor de înţeles. Un proiect accesibil este mai uşor de modificat mai târziu. Disponibilitatea măsoară timpul în care aplicaţia este operaţională şi gata de folosit. Această aplicaţie este construită cu scopul de a fi operaţională 24 de ore din 24, adică nu există nici un moment în care acesta să nu fie disponibilă, doar în unele cazuri excepţionale în care este impusă oprirea acesteia.

e) **Portabilitatea –** Aplicaţie face abstracţie de sistemul de operare în care este deschis browser-ul în care acesta rulează. În schimb unele funcţionalităţi nu sunt disponibile în unele browsere datorită tehnologiei folosite care nu este implementată momentan pentru acestea.

f) **Flexibilitatea -** Modificarea, adăugarea sau ştergerea unei funcţionalităţi a sistemului se poate realiza destul de uşor deoarece fiecare funcţionalitate are modulul său de implementare aşa cum s-au construit iniţial pe tehnologia aleasă. De exemplu distribuirea video-ului live este independenţa de chat-ul ataşat adică poate să funcţioneze şi fără acesta.

g) **Fiabilitatea** – probabilitatea ca un sistem software să funcţioneze fără erori pentru o perioada de timp precizată, într-un mediu dat. Această cerinţă este totuşi relativă, în contextul în care dacă există o eroare dar ea nu are consecinţe serioase, aplicaţia să poată totuşi fi fiabilă. O aplicaţie care are procese fiabile înseamana că, va conduce în mod consistent la obţinerea de produse de înaltă calitate [30].

4.4. Diagrame use-case

În această aplicaţie, cele două roluri principale ale utilizatorilor sunt cea de stundent şi cea de lector. Pe baza acestor roluri s-au construit diagramele de use-case.

Prima diagrama use-case este pentru utilizatorul cu rol de lector.

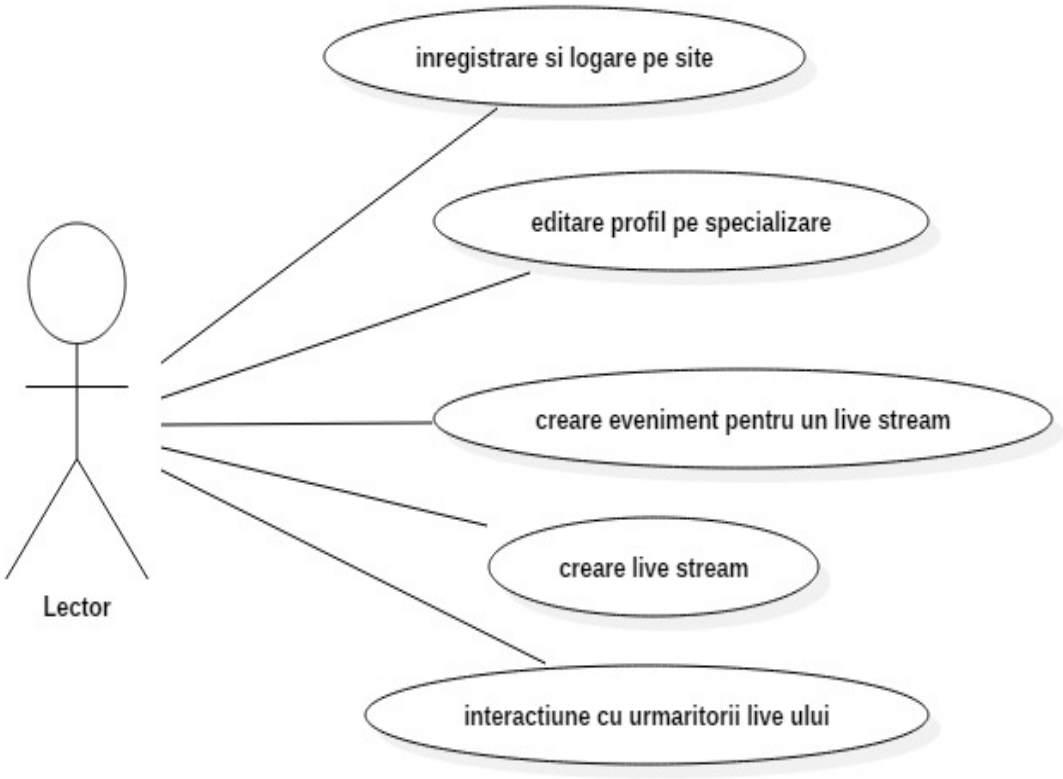

Figura 4.8. Diagrama use-case pentru utilizatorul lector

Tabel 4.2 Use-case-urile pentru utilizatorul lector

Use-case	Descriere
1. Înregistrarea pe site	Descriere: se înregistrează numele , prenumele, email, username, password, telefon, profesie, specializare. Precondiţii: utilizatorul nu are cont pe site Postcondiţii: utilizatorul are cont pe site Scenariu de succes: înregistrarea se efectuează cu succes, iar lectorul se va putea loga în contul sau. Scenarii de eşec: din cauza căderii serverului sau a conexiunii înregistrarea poate să nu aibă loc, utilizatorul fiind nevoit să încerce să se înregistreze din nou, altă dată.
2. Logarea pe site	Descriere: utilizatorul introduce email-ul şi parolă. Precondiţii: utilizatorul nu este deja logat Postcondiţii: utilizatorul este logat în contul sau. Scenariu de succes: logarea se efectuează cu succes, iar lectorul va avea acces la profilul sau. Scenarii de eşec: din cauza conexiunii utilizatorul nu se poate loga în cont şi din această cauza ar putea fi anulat un live care urmă să fie ţinut de lectorul respectiv.

3. Editare profil	Descriere: lectorul îşi adaugă specializare pentru a ştii cei care îl urmăresc ce subiecte abordează, dar şi limba în care va preda cursurile. Precondiţii: profilul nu este completat Postcondiţii: toate datele sunt actualizate. Scenariu de succes: editarea se efectuează cu succes, adică date modificate sunt salvate. Scenarii de eşec: datele nu se actualizează. Extensii: un lector poate avea mai multe specializări.
4. Creare eveniment	Descriere: lectorul decide ce subiect va preda şi va crea un eveniment cu ziua şi ora în care va ţine acel curs. Precondiţii: lectorul are detaliile despre cursul pe care doreşte să îl ţine. Postcondiţii: se pun detaliile în eveniment. Scenariu de succes: evenimentul se creează cu succes şi este vizibil pentru toată lumea. Scenarii de eşec: evenimentul nu se poate crea din cauza unor probleme tehnice.
5. Creare live stream	Descriere: lectorul începe live stream-ul cursului anunţat sau nu. Precondiţii: lectorul are acces la Internet şi este logat în contul său. Scenariu de succes: live-ul se difuzează fără întreruperi sau blocaje, într-o calitate bună video şi audio. Scenarii de eşec: din cauza reţelei live-ul poate fi întrerupt sau anulat. Lectorul poate să anuleze cursul din motive personale.
6. Interactiune cu urmaritorii	Descriere: lectorul poate să răspundă la întrebările puse pe chat de către utilizatori sau poate să intre într-o conversaţie directă cu un student. Precondiţii: lectorul trebuie să iniţieze un live stream. Scenariu de succes: comunicarea dintre cei doi decurge fără probleme şi întreruperi. Scenarii de eşec: comunicare se poate întrerupe, lectorul neavând posibilitatea să răspundă studentului.

A doua diagramă use-case este pentru utilizatorul cu rol de student.

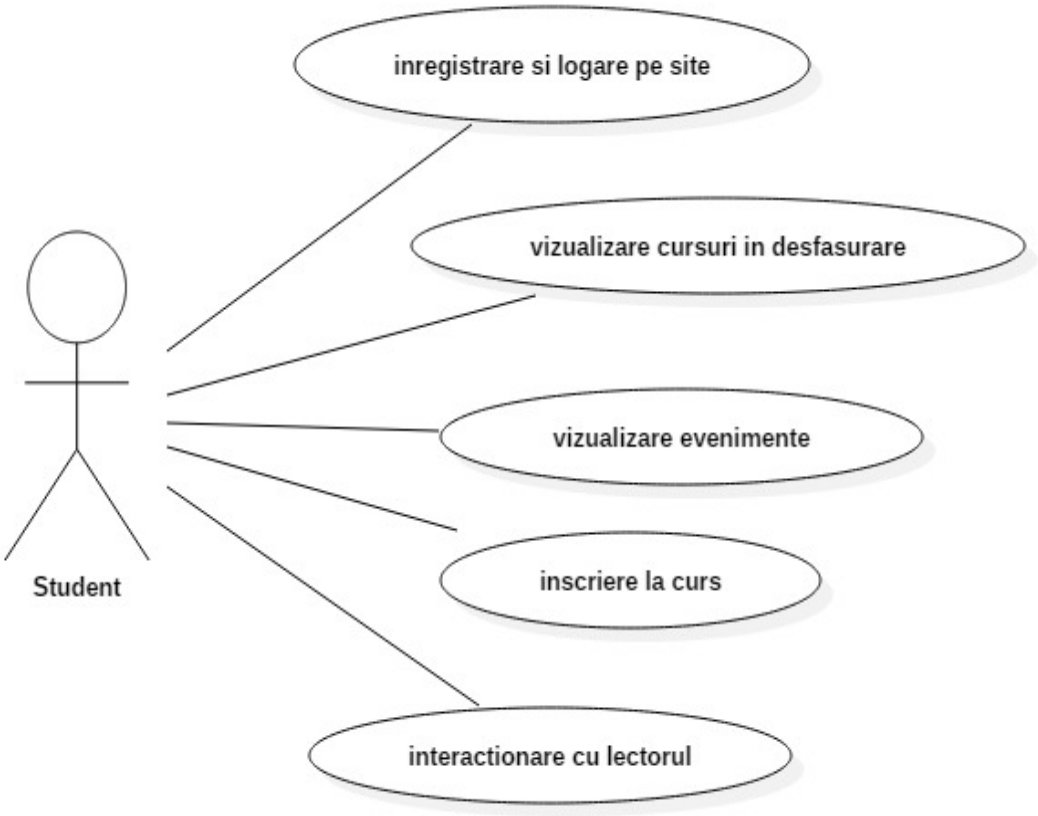

Figura. 4.9 Diagrama use-case pentru utilizatorul student

Tabel 4.3 Use-case-uri pentru utilizatorul student

Use-case	Descriere
1. Înregistrarea pe site	Descriere: se înregistrează numele , prenumele, email, username, password, profesie, specializare. Precondiţii: utilizatorul nu are cont pe site Postcondiţii: utilizatorul are cont pe site Scenariu de succes: înregistrarea se efectuează cu succes, iar studentul se va putea loga în contul său. Scenarii de eşec: din cauza căderii serverului sau a conexiunii înregistrarea poate să nu aibă loc, utilizatorul fiind nevoit să încerce să se înregistreze din nou, altă dată.
2. Logarea pe site	Descriere: utilizatorul introduce email-ul şi parola. Precondiţii: utilizatorul nu este deja logat Postcondiţii: utilizatorul este logat în contul său. Scenariu de succes: logarea se efectuează cu succes, iar studentul va avea acces la profilul său. Scenarii de eşec: din cauza conexiunii utilizatorul nu se poate loga în cont.

3. Vizualizare cursuri in desfasurare	Descriere: studentul poate vizualiza un curs care este în desfăşurare fără să se fi înscris la un eveniment creat pentru cursul respectiv. Precondiţii: studentul trebuie să fie logat în contul sau. Postcondiţii: studentul poate intra şi urmării unul din cursuri. Scenariu de succes: cursul nu este cu înscriere astfel că studentul poate participa. Scenarii de eşec: conexiunea se poate întrerupe sau studentului nu i se permite accesul deoarece acel curs este bazat pe înscriere.
4. Vizualizare evenimente	Descriere: studentul poate cauta un eveniment după anumite specificaţii şi se poate înscrie la câte evenimente doreşte. Precondiţii: studentul trebuie să fie logat în contul sau. Postcondiţii: studentul este înscrie la un anumit curs. Scenariu de succes: evenimentele la care studentul se înscrie nu sunt în acelaşi timp, el putând să se înscrie la toate cele pe care le doreşte. Scenarii de eşec: din cauza conexiunii studentul nu se poate înscrie la eveniment, riscând să nu mai poată participa la acesta.
5. Înscriere la curs (prin eveniment)	Descriere: aplicarea efectivă la un eveniment Precondiţii: studentul trebuie să fie logat în contul sau. Postcondiţii: studentul este înscrie la un anumit curs. Scenariu de succes: înscrierea are loc cu succes Scenarii de eşec: evenimentul poate fi condiţionat de un număr maxim de utilizatori care să îl urmărească sau poate fi contra cost.
6. Interacţiune cu lectorul	Descriere: studentul poate să interacţioneze cu cel care ţine cursul printr-un chat care se desfăşoară pe tot parcursul live-ului sau poate intra în legătură directă cu lectorul la terminarea predării cursului, adresându-şi astfel întrebările. Precondiţii: studentul trebuie să fie logat în contul sau. Postcondiţii: studentul a reuşit să pună întrebări Scenariu de succes: comunicarea dintre cei doi se desfăşoară în condiţii optime fără întreruperi. Scenarii de eşec: comunicarea dintre student şi lector nu se poate efectua din cauza conexiunii sau fiind un număr mare de utilizatori la care lectorul trebuie să răspundă, nu mai ajunge să răspundă şi studentului în cauză.

Administratorul gestionează autentificarea utilizatorii prin serviciul Auth0 şi administrează baza de date prin programul WorkBench. Acesta nu trebuie să aibă cont în aplicaţie pentru a supraveghea ce se întâmplă în baza de date sau pentru a efectua operaţii CRUD pe tabelele din aceasta. În cazul în care se produce o modificare, administratorul poate să trimită un mail persoanei afectate.

4.5. Cerinţe de resurse

Aplicaţia construită este o aplicaţie web, care poate fi deschisă în browser fără nevoia instalării de plugin-uri. Pentru a avea acces la un browser, utilizatorul are nevoie de un dispozitiv electronic, conectat la Internet (ex. calculator, telefon, tabletă). Tehnologia folosită, EasyRTC nu permite folosirea oricărui browser, dar este funcţională pe cele mai principale cum sunt: Google Chrome, Mozilla Firefox şi Opera.

Sistemul este dezvoltat ca şi o aplicaţie web deoarece browser-ul reprezintă platforma principală de aplicare a majorităţii cerinţelor când acestea necesită Internet pentru funcţionare. În timp ce pe dispozitivele mobile se crează diferite aplicaţii care înlocuiesc browser-ului, pe calculatoarele desktop el rămâne sursă primară de accesare a Internetului, indiferent de activitatea desfăşurată.

Capitolul 5. Proiectare de Detaliu si Implementare

În acest capitol este descrisă structura sistemului dezvoltat împreună cu modul de implementare şi dezvoltare al componentelor sistemului.

5.1. Structura generală

Arhitectura generală a sistemului reprezintă modul în care sunt conectate componentele pentru a se construi un sistem capabil să implementeze cerinţele funcţionale impuse de aplicaţie.

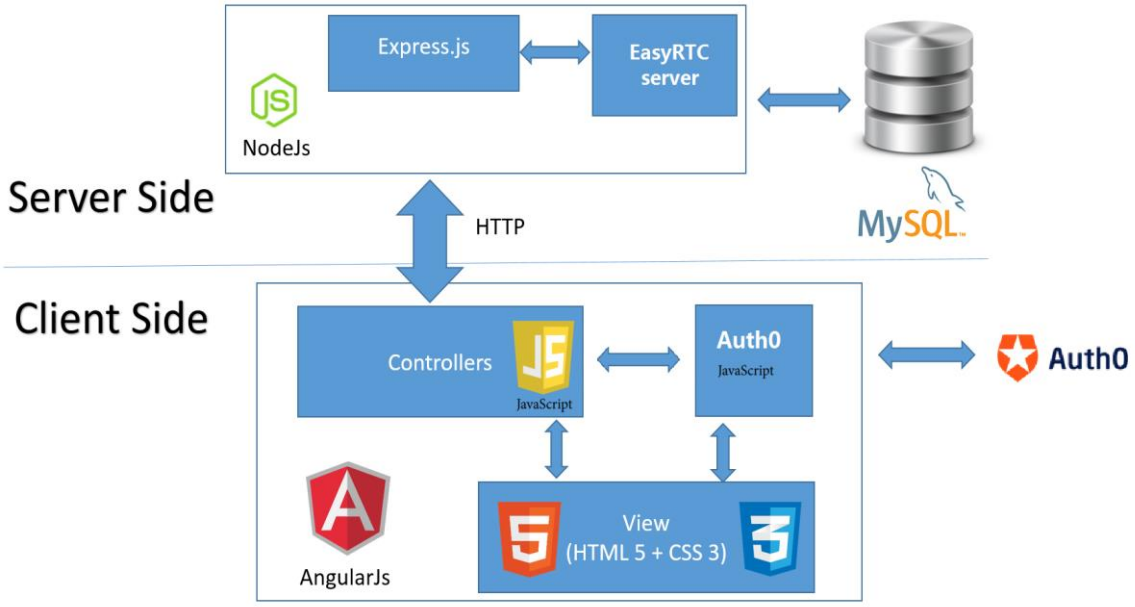

Figura 5.1. Arhitectura generală a sistemului

Structura generală a sistemului este reprezentată de o arhitectură client-server. Această arhitectură împarte aplicaţia în trei componente principale: clientul, infrastructura reţelei (modul de transmitere a datelor) şi serverul, fiind o arhitectură stratificată. Componenta client reprezintă interfaţa prin care un utilizator comunică cu aplicaţia şi partea din care se fac apeluri către server pentru a primi date. Logica aplicaţiei se referă la partea din program care decide ce acţiuni trebuie să se execute în diferite situaţii. În funcţie de tipul arhitecturii client-server logica aplicaţiei poate să fie plasată în totalitate în partea de server sau poate fi împărţită între client şi server. În acest sistem logică de business ce conţine operaţiile cu baza de date este implementată în partea de server iar logică de autentificare a utilizatorului este implementată în partea de client. Server-ul are rolul de a furniza servicii şi resurse utilizatorilor. Funcţia principală a serverului este responsabilitatea administrării accesului la baza de date: sortarea datelor, selectarea resurselor cerute de utilizator, asigurarea că datele nu sunt modificate de utilizatori care nu ar trebui să aibă acces şi persistenţa datelor. Din punct de vedere hardware, componenta server este un calculator puternic cu servere specializate bazate pe microprocesoare Intel, cu procese dedicate, pregătite să administreze: unităţile de disc (file server), imprimante (print server), traficul pe reţea (network service).

Componentele sistemului comunica prin intermediul protocolului HTTP, prin metodele GET şi POST. GET este cea mai folosită metodă, fiind utilizată atunci când se cere o resursă de la server. POST este folosită pentru a trimite date către server. Server-ul preia cererile de la client şi pe baza lor trimite interogări la baza de date pentru răspunsuri, pe care le trimite înapoi la client.

Între utilizatorii aplicaţiei, în procesul de comunicare, există o relaţie peer-to-peer. Această relaţie permite comunicarea directă între utilizatori în momentul desfăşurării live streaming-ului şi în zona de chat ataşată acelui live. Utilizatorii comunică cu serverul doar în momentul în care au nevoie de informaţiile stocate în baza de date.

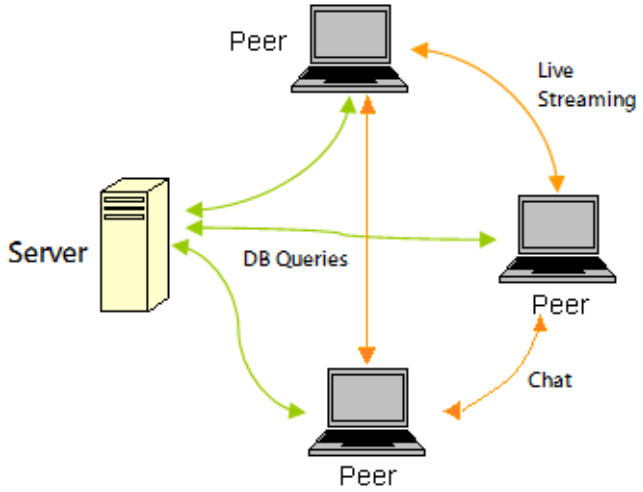

Figura 5.2 Diagrama de comunicare

In diagrama de comunicare s-a notat cu portocaliu conexiunea între utilizatori pentru comunicare directă şi cu verde conectarea staţiilor cu serverul, implicit cu baza de date.

5.2. Descriere componente

Componentele sistemului sunt: server-ul EasyRTC, baza de date MySQL şi interfaţa utilizator care este un model SPA care este conectată cu server-ul de Auth0 pentru autentificarea utilizatorilor.

Diagrama de deployment reprezintă o structură care descrie arhitectura sistemului cu artefactelor software şi specifică modul de implementare al componentelor. În construirea diagramei pentru acest sistem, componentele principale sunt: server-ul, aplicaţia cu interfaţa utilizator şi baza de date. În aplicaţie intră şi serviciul de autentificare Auth0.

Aplicaţie web comunică cu server-ul prin intermediul protocolului HTTP, prin metodele GET şi POST. Din interfaţă sunt iniţiate cererile, care sunt trimise către server şi pentru fiecare se aşteaptă un răspuns înapoi.

Server-ul comunică cu baza de date prin intermendiul interogărilor care se pot trimite odată ce s-a realizat comunicarea între server şi baza de date. Conexiunea se verifică prin funcţia `*mysql.createConnection*` în care este introdusă baza de date cu care se comunică.

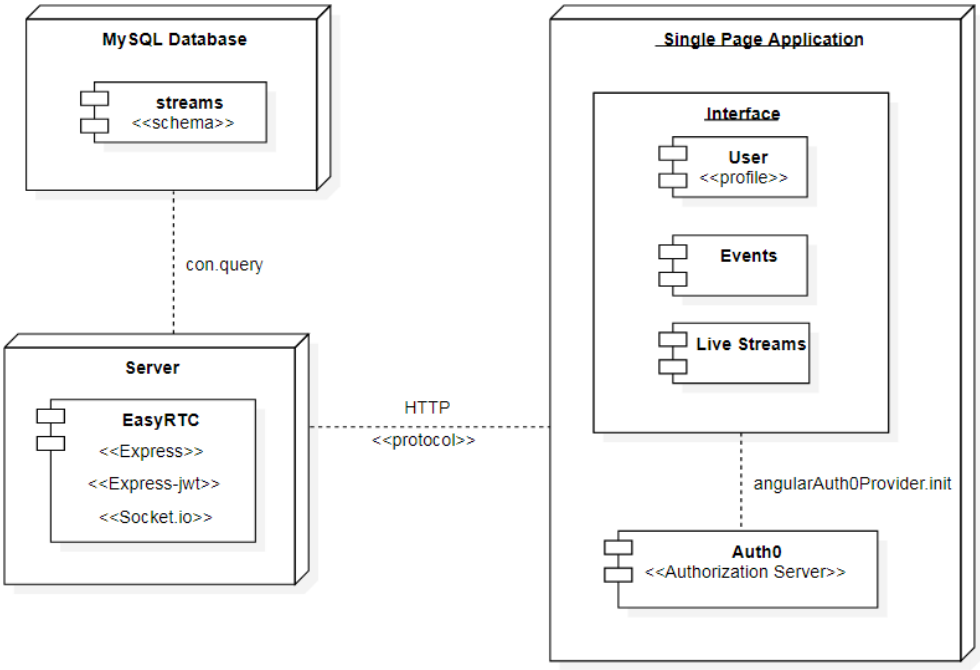

Figura 5.3 Diagrama de deployment

5.2.1. Structura EasyRTC

EasyRTC este o tehnologie care are implementată partea de comunicare video şi prin mesaje între doi sau mai mulţi utilizatori. Din toate modelele pe care această tehnologie le implementează, în aplicaţia dezvoltată vor fi integrate comunicarea printr-un video live, comunicarea prin mesaje şi trimiterea de fişiere.

Pentru a avea acces la funcţiile din EasyRTC şi pentru a porni serverul trebuie instalate dependenţele pentru http, express, socket.io, express-jwt, jwks-rsa, cors şi mysql. După instalarea dependenţelor se încarcă modulele specifice acestora în program.

```
var http    = require("http");
var express = require("express);
var socketIo = require("socket.io");
var easyrtc = require("../");
var jwt = require('express-jwt');
var jwks = require('jwks-rsa');
var cors = require('cors');
var bodyParser = require('body-parser');
var mysql = require('mysql');
```

In `*app*` se configurează server-ul Express http, după care server-ul poate fi deschis pe portul 8081.

```
var app = express();
var webServer = http.createServer(app).listen(8081);
```

După configurarea server-ului de Express http se ataşează la acesta Socket.io prin `*socketIo*`. Socket.io este un modul care permite comunicarea bidirecţională bazată pe eveniment, pentru aplicaţii web care livrează conţinut în timp real.

```
var socketServer = socketIo.listen(webServer, {"log level":1});
```

Cu ajutorul server-ului Express şi socket.io se poate porni server-ul EasyRTC.

```
var rtc = easyrtc.listen(app, socketServer, null, function(err,
rtcRef) {
    console.log("Initiated");
    rtcRef.events.on("roomCreate", function(appObj,
creatorConnectionObj, roomName, roomOptions, callback) {
        console.log("roomCreate fired! Trying to create: " +
roomName);
        appObj.events.defaultListeners.roomCreate(appObj,
creatorConnectionObj, roomName, roomOptions, callback);
    });
});
```

Pentru a asigura faptul că resursele folosite şi operaţiile efectuate în interfaţă sunt executate de către o persoană autentificată în sistem, este nevoie de JWT. JWT (JSON Web Tokens) reprezintă un standard care asigură un mod de autentificare pentru cererile care au loc între aplicaţia front end şi API-ul de backend.

```
var authCheck = jwt({
    secret: jwks.expressJwtSecret({
      cache: true,
      rateLimit: true,
      jwksRequestsPerMinute: 5,
      jwksUri: "https://anacristina.eu.auth0.com/.well-
known/jwks.json"
    }),
    audience: 'https://' + AUTH0_DOMAIN + '/userinfo',
    issuer: "https://anacristina.eu.auth0.com/",
    algorithms: ['RS256'],
    getToken: function fromHeaderOrQuerystring (req) {
        if (req.headers.authorization &&
req.headers.authorization.split(' ')[0] === 'Bearer') {
           return req.headers.authorization.split(' ')[1];
        } else if (req.query && req.query.token) {
          return req.query.token;
        }
        return null;
      } });
```

În jwt se foloseşte informaţia din API-ul creat în Auth0 (domain, audience, issuer). Jwt este folosit pentru a proteja resursele şi pentru a verifica că utilizatorul care încearcă să acceseze aceste resurse este autentificat prin serviciul Auth0. Pentru această verificare se aplică `*authCheck*` acelor rute care conţin informaţii protejate prin metodele `*app.get*` şi `*app.post*`. Aceste metode se conectează cu cererile de HTTP care au fost trimise din interfaţă.

De exemplu atunci când o persoană doreşte să vadă lista de evenimente la care s-a înscris, în momentul în care apasă pe butonul `My Events` din meniu, se face automat un apel ($http.get) la baza de date prin care se caută evenimentele la care este înscrisă persoana respectivă. În partea de server această cerere este conectată cu o metodă `*app.get*` care are ca şi parametru variabila `*authCheck*` prin care se verifică dacă cererea trimisă din interfaţă este realizată de un utilizator autentificat.

```
app.get('/my/events', authCkeck, function(req, res) {
    con.query('SELECT `event`.`id`, `event`.`title`,
`user`.`name`, `event`.`startDate`, `event`.`category`,
`event`.`language`, `event`.`description`, `event`.`status` '
+ ' FROM event INNER JOIN user ON `user`.`id`=`event`.`creator`'
    + ' WHERE `event`.`creator`=?',[req.body.id], function(err,
rows) {
        if(err) throw err;
        res.send(rows);
    });
});
```

Dacă un utilizator neautentificat reuşeşte să apeleze link-ul care conţine `*/my/events*`in coadă în speranţa că va avea acces la informaţia din baza de date, prin verificarea cu `*authCheck*` se va observa că nu a fost un utilizator autentificat care a făcut cererea, pentru că nu există nici un *accessToken* înregistrat, drept urmare nu va primi datele din baza de date ci o eroare.

Dacă pentru datele cerute din baza de date nu este nevoie de autentificare, cum este cererea pentru afişarea tuturor evenimentelor, atunci authCheck nu trebuie trimis ca şi paramentru pentru verificarea autentificării.

```
 app.get('/all/events', function(req, res) {
    con.query('SELECT `event`.`id`, `event`.`title`,
`user`.`name`, `event`.`startDate`, `event`.`category`,
`event`.`language`, `event`.`description`, `event`.`status`  '
    + ' FROM event INNER JOIN user ON
`user`.`id`=`event`.`creator`', function(err, rows) {
        if(err) throw err;
        res.send(rows);
    });
});
```

Modurile de comunicare din această aplicaţie sunt: prin video şi audio, printr-un chat şi prin transfer de fişiere.

Comunicarea video şi audio din această aplicaţie presupune difuzarea unui live stream de către o persoană şi urmărirea acestuia de alţi utilizatori. În momentul în care un lector deschide pagina de difuzare a live stream-ului, camera web şi microfonul trebuie să fie conectate la calculator, stream-ul live fiind conectat automat.

Această conectare automată se realizează cu ajutorul funcţiei *connect()* care este introdusa în tag-ul <body> al paginii de live stream cu elementul *onload*. *Onload* permite executarea codului de javaScript imediat după încarcarea paginii HTML. În funcţia *connect()* sunt integrate toate funcţiile care asigura comunicarea între lector şi cei care urmăresc cursul.

```
function connect() {
```

```
    easyrtc.setVideoDims(640,480);
    easyrtc.setRoomOccupantListener(convertListToButtons);
    easyrtc.easyApp("easyrtc.audioVideoSimple", "selfVideo",
loginSuccess, loginFailure);
     easyrtc.enableDebug(false);
     easyrtc.enableDataChannels(true);
     easyrtc.enableVideo(false);
     easyrtc.enableAudio(false);
     easyrtc.enableVideoReceive(false);
     easyrtc.enableAudioReceive(false);
     easyrtc.setDataChannelOpenListener(openListener);
     easyrtc.setDataChannelCloseListener(closeListener);
     easyrtc.setPeerListener(addToConversation);
     easyrtc.connect("easyrtc.dataMessaging", loginSuccess,
loginFailure);
     easyrtc.setAcceptChecker(function(easyrtcid, responsefn) {
         responsefn(true);
         document.getElementById("connectbutton_" +
easyrtcid).style.visibility = "hidden";
     });
     easyrtc.setDataChannelOpenListener(function(easyrtcid,
usesPeer) {
        var obj =
document.getElementById(buildDragNDropName(easyrtcid));
        if (!obj) {
            console.log("no such object ");
        }
        jQuery(obj).addClass("connected");
        jQuery(obj).removeClass("notConnected");
    });
    easyrtc.setDataChannelCloseListener(function(easyrtcid) {
  jQuery(buildDragNDropName(easyrtcid)).addClass("notConnected");
jQuery(buildDragNDropName(easyrtcid)).removeClass("connected");
      });
}
```

Pentru partea de live streaming sunt necesare funcţiile:

- *setVideoDims(640, 480);* - se setează dimensiunile zonei pentru camera web.
- *setRoomOccupantListener(convertListToButtons);* - se setează un callback care va fi invocat atunci când lista de oameni prezenţi în live stream se schimbă.
- *easyApp("easyrtc.audioVideoSimple", "selfVideo", loginSuccess, loginFailure);* - se atribuie stream-ului media local obiectul video identificat de `*monitorVideoId*`, stream-ului video remote obiectul identificat de *videoIds*, apoi se apelează funcţia *onReady*. Primul parametru reprezintă numele aplicaţiei. Al doilea parametru reprezintă id-ul obiectului video prin care se accesează camera locală. Al treilea parametru reprezintă funcţia care se apelează la conectarea cu succes iar parametrul al patrulea reprezintă funcţia care se apelează la eşec.

Comunicarea prin mesaje se realizează în timpul difuzării unui live stream. Pe tot parcursul live streaming-ului cei care îl urmăresc au posibilitatea a a-i pune întrebări lectorului. Pentru comunicarea prin mesaje sunt necesare funcţiile:

- easyrtc.enableDebug(false); - se activează sau dezactivează legătura cu consola.
- *easyrtc.enableDataChannels(true);* - se setează atunci când canalele de date WebRTC sunt folosite pentru a trimite mesaje între clienţi, mesajele pe care clienţii doresc să le trimită si nu mesajele de semnalizare de la WebRTC.
- *easyrtc.enableVideo(false);* - se setează dacă este un videoclip de transmis. Este setat la valoare `false` pentru că între utilizatori sunt trimise numai mesaje scrise.
- *easyrtc.enableAudio(false);* - se setează dacă există sunet de trimis între utilizatori. La fel ca şi pentru video este setat la valoarea `false`.
- *easyrtc.enableVideoReceive(false);* - se controlează cererile video pe care un client le poate primi de la un alt peer în timpul unei conexiuni. Este setat la valoarea `false` deoarece între utilizatori sunt transmise numai mesaje text.
- *easyrtc.enableAudioReceive(false);* - se controleaza cererile audio pe care un client le poate primi de la un alt utilizator. Este setat la valoarea `false`, implicit fiind setat la valoarea `true`, pentru a nu permite comunicarea audio în zona destinată mesajelor text.
- *easyrtc.setDataChannelOpenListener(openListener);* - se setează un callback care este apelat în momentul în care este deschis un canal de date şi este pregatit să trimită date. Parametrul este un element de tip easyrtcid.
- *easyrtc.setDataChannelCloseListener(closeListener);* - se setează un callback care se apelează când un canal de date deschis anterior se închide.
- *easyrtc.setPeerListener(addToConversation);* - se setează un listener pentru datele trimise de alt client (peer-to-peer sau prin websockets).
- *easyrtc.connect("easyrtc.dataMessaging", loginSuccess, loginFailure);* - conectarea la serverul EasyRTC. Primul parametru reprezintă numele aplicaţiei astfel încât diferite aplicaţii să poată avea liste diferite cu clienţi. Al doilea parametru este o funcţie care este apelată atunci când conexiunea se realizează cu succes. Al treilea parametru este o funcţie care se apelează atunci când conexiunea nu se realizează.

Pentru transferul de fişiere se adaugă şi funcţiile

- *easyrtc.setAcceptChecker(function(easyrtcid, responsefn));* se setează un callback prin care se decide dacă se acceptă sau se respinge un anumit transfer. Parametrul este o funcţie `*acceptCheck*` care trebuie apelată fie cu o valoare `true` pentru a accepta transferul, fie cu o valoare `false` pentru a respinge transferul.
- *easyrtc.setDataChannelOpenListener(function(easyrtcid, usesPeer));* - se setează un callback care este apelat în momentul în care este deschis un canal de date şi este pregatit să trimită date.
- *easyrtc.setDataChannelCloseListener(function(easyrtcid));* - se setează un callback care se apelează când un canal de date deschis anterior se închide.
- *easyrtc.connect("easyrtc.dataFileTransfer", loginSuccess, loginFailure);* - conectarea la serverul EasyRTC. Primul parametru reprezintă numele

aplicaţiei. Al doilea parametru este o funcţie care este apelată atunci când conexiunea se realizează cu succes. Al treilea parametru este o funcţie care se apelează atunci când conexiunea nu se realizează.

5.2.2. Structura interfeţei

Interfaţa utilizator reprezintă o aplicaţie web dezvoltată pe modelul SPA folosind Angular.js şi HTML5. Fişierul index.html este deschis la lansarea aplicaţie în browser, integrând cu directiva *ng-view* elementele din pagina de home din fişierul home.html.

Directivele din Angular.js folosite în acest proiect sunt:

a) ng-app – declară un element ca fiind elementul rădăcină (root) al aplicaţiei permiţând modificarea comportamentului prin tag-uri HTML.
b) ng-view – directiva de bază responsabilă cu gestionarea rutelor care rezolva JSON înainte de redarea şabloanelor din controller-ele specificate.
c) ng-model – permite o legătură bidirecţională între pagina de interfaţă view şi componenta scope din controller.
d) ng-if – directiva care verifică o anumită funcţie înainte de a permite gestionarea altor variabile.
e) ng-repeat – instanţiază un element pe rând dintr-o colecţie.
f) ng-click – permite specificare unui comportament distinct când se apasă pe un element.

Directivele sunt marcatoare pentru elementele DOM (Document Object Model), care transmit compilatorului de HTML din Angular.js ($compile) să atribuie un comportament specific acelui element. Directivele creează o conexiune între paginile HTML şi controller-ele din javascript care conţin funcţionalitatea elementelor din interfaţă.

În componenta client există câteva fişiere care au rolul de a gestiona şi ruta toate celelalte fişiere pentru a se conecta împreună.

În fişierul `*app.js*` se găseşte funcţia de configurare în care sunt create legăturile între paginile HTML şi controller-e (config). Rutarea se realizează cu ajutorul elementului *$stateRouter* şi funcţia *state* care permite alocarea de nume rutelor.

Toate rutele care sunt construite pentru evenimentele din această aplicaţie, adică crearea, editarea şi vizualizarea evenimentelor sunt următoarele:

```
$stateProvider.state('myEvents', {
    url: '/myEvents',
    controller: 'MyEventsController',
    templateUrl: 'components/events/myEvents.html',
    controllerAs: 'myevents'
     });

  $stateProvider.state('editEvent', {
    url: '/editEvent/:id',
    controller: 'EditEventController',
    templateUrl:'components/events/editEvent.html',
    controllerAs: 'editevent'
  });
```

```
$stateProvider.state('addEvent', {
  url: '/addEvent',
  controller: 'AddEventController',
  templateUrl:'components/events/createEvent.html',
  controllerAs: 'addevent'
});

$stateProvider.state('allEvents', {
 url: '/allEvents',
 controller: 'AllEventsController',
 templateUrl: 'components/events/allEvents.html',
 controllerAs: 'allevents'
});
```

În funcţia *state* se adaugă numele, url-ul, templateUrl care face trimitere la pagina de HTML, numele controller-ului şi numele variabilei principale folosite în controller şi trimisă la view. Cu acest template construit nu mai este necesară folosirea adreselor directe şi nici schimbarea atributului href de fiecare dată când se schimbă ruta. Cu state trebuie doar trecut numele în atributul ui-sref.

Pe lângă funcţia de configurare a rutelor în *app.js* se mai găseşte şi metodă de initializare (init) prin care se face legătură cu serverul de Auth0 care va permite autentificarea utilizatorilor.

```
angularAuth0Provider.init({
      clientID: AUTH0_CLIENT_ID,
      domain: AUTH0_DOMAIN,
      responseType: 'token id_token',
      redirectUri: AUTH0_CALLBACK_URL,
      audience: 'https://' + AUTH0_DOMAIN + '/userinfo',
      authService: {
          params: {
             scope: 'openid email'
              }
          }
    });
```

Initializarea conţine variabilele care sunt definite în *auth0.variable.js*, variabile care sunt disponibile odată ce administratorului aplicaţiei a creat un cont pe auth0.com şi a creat un client şi un API pentru acesta. Variabilele folosite pentru conectare sunt:

```
var AUTH0_CLIENT_ID='iAYssqt03c0myXXvQopYSZgQ08gkXf54';
var AUTH0_DOMAIN='anacristina.eu.auth0.com';
var AUTH0_CALLBACK_URL=window.location.href;
var AUTH0_API_AUDIENCE = 'coursestream';
```

Variabilă AUTH0_CLIENT_ID reprezintă un id care este trimis la Auth0 în momentul în care cineva se autentifică pe site pentru a face legătură cu clientul creat pe site , iar variabila AUTH0_API_AUDIENCE reprezintă numele API-ului creat pentru acel client. Diferenţa dintre client şi API este acea că un API reprezintă o entitate prin care se creează o legătură cu o sursă externă, reprezentând doar o parte din aplicaţie iar clientul reprezintă întreagă aplicaţie construită cu serviciul oferit de Auth0.

Pe lângă variabilele din Auth0 pentru initializare mai este necesar elementul de *responseType* prin care se trimite de la server-ul Auth0 idToken-ul utilizatorului autentificat.

Fişierul `*auth.service.js*` este un service de autentificare care este responsabil cu autentificarea utilizatorilor şi cu administrarea stării acestora. În funcţia principală authService se află:

- funcţia de login care redirecţionează apelul utilizatorului de a se autentifica către Auth0 şi va asigura detaliile de autentificare.

```
function login() {
  angularAuth0.authorize();
}
```

- funcţia de parsare a codului hash care vine de la Auth0 după autentificare pentru a extrage idToken-ul şi accessToken-ul pentru utilizator.

```
function handleParseHash() {
   angularAuth0.parseHash(
     { _idTokenVerification: false },
     function(err, authResult) {
     if (err) {
       console.log(err);
     }
     if (authResult && authResult.idToken) {
       setUser(authResult);
     }
   });
 }
```

- funcţia prin care se se iau datele utilizatorului autentificat în aplicaţie. Librăria auth0.js conţine o metodă numită *client.userInfo* care apelează link-ul ce conţine */userInfo* în coadă şi va prelua informaţiile utilizatorului. Token-ul de acces *accessToken* al utilizatorului trebuie sa fie primul parametru al metodei.

```
function getProfile(cb) {
     var accessToken = localStorage.getItem('access_token');
     if (!accessToken) {
       throw new Error('Access token must exist to fetch
profile');
     }
     angularAuth0.client.userInfo(accessToken, function (err,
profile) {
        if (profile) {
         setUserProfile(profile);
       }
        cb(err, profile);
      });
     }
```

- funcţia de logout care va distruge idToken-ul şi accessToken-ul utilizatorului când acesta iasă din cont .

```
function logout() {
```

```
    localStorage.removeItem('access_token');
    localStorage.removeItem('id_token');
  }
```

- funcţia care stochează datele de la user adică idToken şi accessToken local.

```
function setUser(authResult) {
    localStorage.setItem('access_token',
       authResult.accessToken);
    localStorage.setItem('id_token',
          authResult.idToken);
  }
```

- funcţia care verifică dacă un utilizator este logat verificând dacă există salvate local un idToken şi un accessToken.

```
    function isAuthenticated() {
          return authManager.isAuthenticated();
    }
```

Controller-ele prin care se conectează interfaţa cu partea de server sunt împarţite în funcţie de cerinţele funcţionale ale utilizatorului.

1) Home

Pagina de Home este pagina de întâmpinare a utilizatorilor care intră pe site-ul aplicaţiei. Aceasta conţine mesaje diferite pentru utilizatorii autentificaţi şi pentru cei neautentificaţi.

În partea stânga e ecranului se află meniul de navigare. Acesta este introdus în pagină index.html, meniul fiind vizibil indiferent de ce pagină este accesată de utilizator. Conţinutul acestuia este diferit pentru un utilizator înregistrat faţă de un utilizator neautetificat.

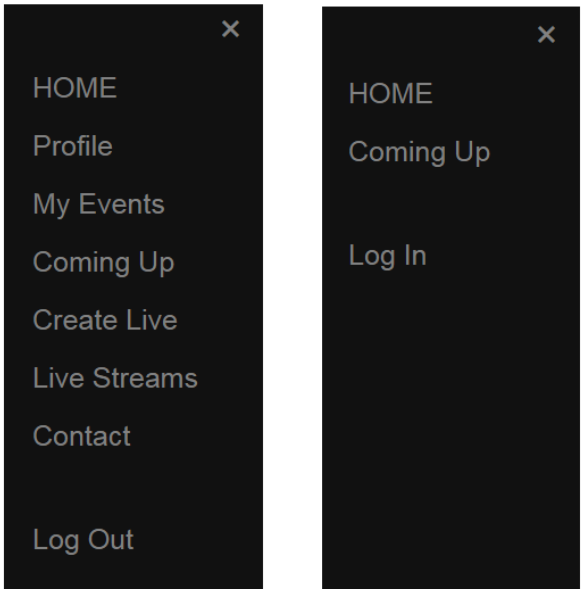

Figura5.4 Meniu de navigare pentru persoane autentificate şi neautentificate

În cod, acest mod de a face vizibile doar anumite elemente din meniu pentru o persoană neautentificată şi toate pentru o persoană autentificată este posibil cu ajutor funcţiei create în service-ul auth.service.js, *auth.isAuthenticated()*:

```
<div ng-app="app">
    <div id="mySidenav" class="sidenav">
      <a href="javascript:void(0)" class="closebtn"
      onclick="closeNav()">&times;</a>
        <a class="active" ui-sref="home"
      onclick="closeNav()">HOME</a>
        <a class="active" ng-if="auth.isAuthenticated()"
        ui-sref="profile" onclick="closeNav()">Profile</a>
        <a class="active" ng-if="auth.isAuthenticated()"
        ui-sref="myEvents" onclick="closeNav()">My Events</a>
        <a class="active" ui-sref="allEvents"
      onclick="closeNav()">Coming Up</a>
        <a class="active" ng-if="auth.isAuthenticated()"
        ui-sref="liveCourse" onclick="closeNav()">Create Live</a>
        <a class="active" ng-if="auth.isAuthenticated()"
        ui-sref="allCourses" onclick="closeNav()">
            Live Streams</a>
        <a class="active" ng-if="auth.isAuthenticated()"
        ui-sref="contact" onclick="closeNav()">Contact</a>
        <a class="active" ng-if="!auth.isAuthenticated()"
      ng-click="auth.login()">Log In</a>
        <a class="active" ng-if="auth.isAuthenticated()"
        ng-click="auth.logout()">Log Out</a>
    </div>
    <span onclick="openNav()">&#9776;MENU</span>
  <div ui-view></div>
</div>
```

Prin directiva *ng-if* care analizează rezultatul funcţiei *auth.isAuthenticated()* se verifică dacă utilizatorul este logat şi astfel vor fi vizibile în meniu toate elementele. Directiva ng-if aşteaptă un rezultat pozitiv pentru a face vizibil un element. Dacă există vreun element care trebuie să fie vizibil doar pentru un utilizator neautentificat (ex. butonul de Login) atunci în directiva ng-if se adaugă semnul `!` care înseamnă negare şi care împreună cu rezultatul funcţiei auth.isAuthenticated() care va fi fals în cazul acesta va produce un rezultat pozitiv pentru directiva ng-if. Toate elementele din meniu conţin un *ui-sref* care face trimitere la o rută dintr-o anumită pagină configurată cu *$stateProvider*.

2) Profile

Profilul unui utilizator logat în aplicaţie conţine următoarele caracteristici: nume, email, status (student / lector), şcoala, specializarea, locaţia (ţară, oraş). Acest profil poate fi editat cu excepţia email-ului pe care l-a folosit pentru autentificare şi statusul care o dată stabilit nu va mai putea fi schimbat. Profilul utilizatorului este adus din baza de date unde se caută după email-ul persoanei logate. Această înregistrare în baza de date se efectuează atunci când o persoană se înregistrează pe site pentru prima dată. Pentru a accesa profilul utilizatorului autentificat, pentru a salva datele în baza de date, se apelează

funcţia `*getCachedProfile*` din auth.service.js şi se pastrează profilul într-o variabilă *prof.profile*.

```
if (authService.getCachedProfile()) {
          prof.profile = authService.getCachedProfile();
        } else {
          authService.getProfile(function(err, profile) {
            prof.profile = profile;
          });
        }
```

Prima dată cand se autentifică o persoană profilul acesteia va trebui completat în toate câmpurile sale, acestea fiind goale. Din autentificare se reţine numele, email-ul şi poza pe care utilizatorul o are la profilul email-ului cu care s-a conectat.

Aceste două operaţii de afişare şi editare creează două controllere ProfileController şi EditProfileController. În controller-ul de afişare se execută o operaţie de get prin HTTP pentru a aduce datele din baza de date.

```
$http.get('http://localhost:8081/user/profile').
then(function(response) {
        prof.user = response.data;
    }).catch(function(error){
        prof.user = "Error getting the data from DB.";
     });
```

In controller-ul de editare se execută o operaţie de post prin HTTP prin care se trimite noile datele la baza de date pentru a fi schimbate.

```
$http.post('http://localhost:8081/edit/profile/',
        "name=" + name + "&school=" + school + "&specialization="
+ specialization + "&location=" + location,
        {headers   :   {'Content-Type':   'application/x-www-form-
urlencoded' }}
         ).then( function(response) {
           editprof.message = response.data;
         } ).catch(function(error) {
        editprof.message = "Error in editing you profile!"; }) };
```

3) Events

Operaţiile pe evenimente sunt împărţite în funcţie de rolul pe care îl are utilizatorul autentificat în aplicaţie. Un student poate să vadă toate evenimentele de pe site, să caute după o anumită categorie sau limbă şi să se aboneze la evenimentele care conţin un subiect despre care acesta este interesat. Toate evenimentele la care s-a abonat sunt înregistrate în secţiunea „My Events” şi de acolo utilizatorul le poate şterge pe cele l-a care nu va mai dori să participe. Un lector pe lângă aceste posibilităţi, poate să creeze un eveniment, să îl modifice sau chiar să îl şteargă.

În funcţie de operaţiile pe evenimente există 4 controller-e: MyEventsController, EditEventController, AddEventController şi AllEventsController.

Pentru a aduce din baza de date evenimentele la care s-a abonat o persoană se foloseşte metoda de get:

```
$http.get('http://localhost:8081/subscribe/events').
then(function(response) {
```

```
                myevents.subs = response.data;
                }).catch(function(error) {
                myevents.message = "Unable to get my subscribed events
from DB!";
                });
```

Pe aceeaşi pagină cu evenimentele la care s-a abonat un profesor sunt puse şi evenimentele pe care acesta le-a creat.

```
$http.get('http://localhost:8081/my/events').
then(function(response) {
         myevents.events = response.data;
       }).catch(function(error) {
         myevents.message = "Unable to get my events from DB!";
      });
```

Pentru a adauga un eveniment, un profesor trebuie să adauge detaliile despre eveniment în câmpurile sugerate, acele detalii fiind apoi trimise la baza de date pentru salvarea lor. Toată informaţia se trimite la apăsarea butonului „Add Event” din interfaţă care are ataşată funcţia `*addEvent()*`:

```
addevent.addEvent = function() {
     var title = addevent.title;
     var date = addevent.date;
     var cat = addevent.cat;
     var desc = addevent.desc;
     var lang = addevent.lang;

     $http.post('http://localhost:8081/add/event/',
     "title=" + title + "&date=" + date + "&cat=" + cat +
"&desc=" + desc + "&lang=" + lang,
      {headers : {'Content-Type': 'application/x-www-form-
urlencoded' }}
      ).then( function(response) {
         addevent.message = response.data;
         } ).catch(function(error) {
         addevent.message = "Error in adding event!"; })
        }
    }
```

Pentru a edita un eveniment, se alege evenimentul din listă, iar după ce au fost modificate câmpurile, prin apăsarea butonului `*Edit Event*` datele sunt trimise la baza de date pentru a fi modificate câmpurile respective.

Toate evenimentele din baza de date sunt vizibile în pagina `Coming Up`. Acestea sunt aduse din baza de date printr-o metodă get HTTP şi sunt afisate într-un tabel.

Căutarea nu mai face trimitere la baza de date pentru a se cauta după o anumită specializare sau limba ci se foloseşte un filtru de la Angular.js care permite filtrarea după introducerea unor caractere din categorie sau limbă. Acest filtru va selecta doar acele evenimente ale căror categorie sau limba se potriveşte cu caracterele introduse. Filtrul se adaugă în dreptul directivei ng-repeat fiind despărţite de o bară verticală ‚|’si fiind conectat cu elementul după care se filtrează evenimentele prin directiva ng-model.

Ştergerea unui eveniment nu presupune eliminarea acestuia din baza de date decât după ce a trecut dată la care trebuia să se desfăşoare live streaming-ul. Când un lector

şterge un eveniment, acesta va apărea în lista de evenimente că fiind anulat iar studenţii nu se vor mai putea abona la acesta. Motivul pentru care s-a ales implementarea aceasta este pentru a nu crea confuzii în rândul studenţilor crezând că numai din lista lor de evenimente a dispărut acel eveniment. Fiind anulat, dar nu şters lectorul are ocazia de a redeschide evenimentul. În baza de date în tabelul event există un câmp `*status*` care atunci când este true sau 1 înseamnă că evenimentul va fi susţinut iar atunci când este 0 înseamnă că evenimentul a fost anulat.

4) Live Stream

Înainte de a crea live streaming-ul, un lector trebuie să introducă câteva date despre acesta pentru ca să fie găsit în lista de cursuri live. După introducerea acelor informaţii se deschide pagina în care se conectează automat camera web şi microfonul iar lectorul îşi va putea susţine cursul.

```
livecourse.addCourse = function() {

            var title = livecourse.title;
            var cat = livecourse.cat;
            var desc = livecourse.desc;
            var lang = livecourse.lang;
            $http.post('http://localhost:8081/add/course/',
              "title=" + title +  "&cat=" + cat + "&desc=" + desc
      + "&lang=" + lang,
              {headers : {'Content-Type': 'application/x-www-
form-urlencoded' }}
             ).then( function(response) {
                  livecourse.message = response.data;
               } ).catch(function(error) {
                   livecourse.message = "Error in adding a live
course!"; });
                  } }
```

La încheierea live streaming-ului, cursul este şters din baza de date şi comunicarea dintre utilizatori se incheie.

Pentru a vedea ce cursuri sunt live sunt aduse din baza de date, toate înregistrările. În baza de date sunt prezente numai cursurile care încep sau sunt în desfăşurare în momentul căutării.

```
  $http.get('http://localhost:8081/all/courses').
then(function(response) {
        allcourses.courses = response.data;
        }).catch(function(error) {
        allcourses.message  =  "Unable  to  get  the  courses  from
DB!";
         });
```

Căutarea unui curs se face după categoria din care acesta face parte sau după limba în care este predat cursul.

5.2.3. Structura bazei de date

Baza de date din acest sistem este o baza de date relaţională, în care datele văzute ca şi atribute ale entităţilor reale, sunt stocate în tabele între care există relaţii. O relaţie reprezintă o asociere între două entităţi ale tabelelor. Structura bazei de date este prezentată în figura următoare.

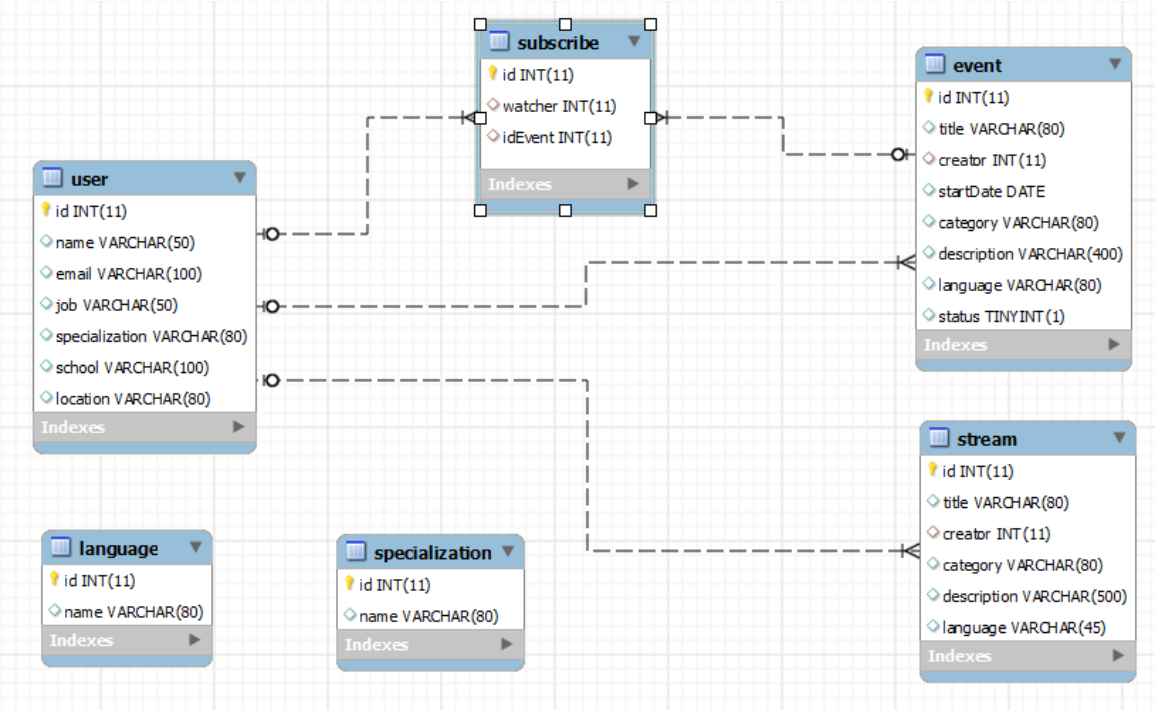

Figura 5.5 Structura bazei de date

Tabelele din baza de date sunt:

A. Tabelul *User* conţine câmpurile:

- id – cheie primară cu auto increment,
- name – numele şi prenumele,
- email – email-ul cu care s-a autentificat în aplicaţie,
- job – student sau lector,
- specialization – daca este specializat într-un domeniu,
- school – ultima şcoală terminată sau care este începută,
- location – ţara şi oraşul în care locuieşte.

B. Tabelul *Event* conţine câmpurile:

- id – cheie primară cu auto increment,
- title – titlu sugestiv pentru eveniment
- creator – cheie straină – acest câmp este conectat la tabelul User prin cheia primară a acestuia reprezentând lectorul care a creat acest eveniment. Legatura este de tip one-to-many (1:M), un lector poate să creeze mai multe evenimente.
- startDate – data la care este programat live streaming-ul.

- category – categoria din care face parte evenimentul
- description – scurta descrie pentru a capta interesul studentilor.
- language – în ce limbă se va desfăşura live streaming-ul
- status – reprezintă starea evenimentului – dacă are valoarea 1 (true) înseamnă că live streaming-ul se va desfăşura iar dacă are valoarea 0 (false) înseamnă că evenimentul a fost amanat.

C. Tabelul *Stream* conţine câmpurile

- id – cheie primară cu auto increment
- title – titlu sugestiv
- creator – cheie straină – acest câmp este conectat la tabelul User prin cheia primară a acestuia reprezentând lectorul care va transmite live streaming-ul. Legatura este de tip one-to-many (1:M), un lector poate crea mai multe live streaming-uri.
- category – categoria din care face parte
- description – scurtă descriere în care este prezentat subiectul abordat.
- language – limba în care lectorul prezintă cursul.

D. Tabelul *Subscribe* contine campurile:

- id - cheie primară cu auto increment
- watcher – student care s-a înscrie la eveniment – acest câmp este conectat la tabelul User prin cheia primară a acestuia. Legatura este de tip one-to-many (1:M), un student se poate înscrie la mai multe evenimente.
- idEvent – id-ul evenimentului la care s-a înscris studentul – acest câmp este conectat la tabelul Event prin cheia primară a acestuia. Legatura dintre aceste tabele este de tip one-to-many (1:M), adică la un eveniment se pot înscrie mai multi studenţi.

E. Tabelul *Specialization* conţine câmpurile:

- id - cheie primară cu auto increment
- name – numele specializării

Valorile introduse în acest tabel sunt legate de specializările care sunt introduse la crearea de evenimente şi cursuri live.

F. Tabelul *Language* conţine câmpurile:

- id – cheie primară cu auto increment,
- name – limba propri-zisă.

Valorile introduse în acest tabel sunt valori ale limbilor în care se vor desfaşura cursurile live.

Aplicaţia web are o interfaţă în care totul este scris în limba engleză în contextul în care această limbă este cea mai vorbită. Pe lângă aceasta, lectorii vor avea posibilitatea să îşi predea cursul în orice limbă dar este de aşteptat ca aceasta sa fie notată în tabel.

Normalizarea reprezintă descompunerea sau reorganizarea relaţiilor astfel încât acestea să ajungă într-o formă relaţională corectă, fără pierdere de informaţie şi cu evitarea redundanţei. Prin acest proces formal se identifică asocierile între schemele de relaţii şi dependenţele funcţionale între atribute. Normalizarea dă posibilitatea celui care

proiectează baza de date, să efectueze o seria de teste pe baza de date, toate aceste teste ducând la prevenirea posibilităţii de a apărea anomalii. Anomaliile pot fi evitate dacă se înlocuiesc schemele de relaţie cu scheme echivalente pentru care dependenţele satisfac anumite restricţii. Schemele de relaţii echivalente reprezintă forme normale pentru setul de relaţii.

Relaţiile din baza de date a sistemului sunt în a treia formă normală FN3. O relaţie R este în a treia formă normală FN3, dacă este în FN2 şi dacă sunt eliminate toate dependenţele funcţionale tranzitive (dacă nu există nici o dependenţă funcţională între atributele non-prime). O relaţie R este în FN2, dacă este în FN1 şi toate dependenţele între cheia primară a lui R şi celelalte atribute a lui R sunt elementare (atributele nu depind de o parte din cheie). O relaţie R este în FN1 atunci când atributele sale nu pot fi descompuse.

Capitolul 6. Testare şi Validare

Testarea unei aplicaţii presupune verificarea îndeplinirii cerinţelor funcţionale şi non-funcţionale ale sistemului. Testarea este realizată printr-o percepţie obiectivă verificându-se faptul că produsul este construit în concordanţă cu cerinţele specificate la începutul dezvoltării. Prin validare se asigura că produsul este utlizabil şi poate fi lansat pe piaţă.

Aplicaţia poate fi testată din mai multe puncte de vedere. Pentru această aplicaţie s-a folosit testarea manuală de tip: Black Box Testing şi White Box Testing.

6.1. Black Box Testing

Testarea de tip Black Box (cutia neagră) presupune testarea funcţionalităţii aplicaţiei şi nu include deloc cod sau nimic ce ţine de procesele interne ale sistemului. Această testare se aplică pe interfaţa utilizator şi se verifică dacă cerinţele specificate au fost implementate. Prin acesta metodă se testează interfaţa, API-urile şi comunicarea cu baza de date.

Un scenariul de test ales pentru testare este deschiderea unui live stream. Paşii parcurşi pentru a implementa această cerinţă funcţională si ce anume se verifică la fiecare pas este descris astfel:

- deschiderea aplicaţiei web într-un browser - *se verifică conexiunea la Internet şi faptul că pe calculator există instalat un browser.*
- autentificarea pe site cu un cont creat pe platformele: Facebook, Google, Yahoo sau Linkedin. Dacă un utilizator nu are cont creat acesta va trebui să se înregistreze mai întai în secţiunea `Sign Up`. – *se verifică conexiunea la serverul Auth0.*
- se deschide meniul şi se alege “Create Live”. Aceste element din meniu este vizibil numai unui utilizator care are rolul de lector. În caz contrar acest element nu apare. – *se verifică faptul că numai un lector are acest element in meniu.*
- se completează câmpurile pentru a oferi detalii despre curs - *se verifică câmpurile să corespundă cu câmpurile din tabelul `stream` din baza de date.*
- se apasă butonul `Create Live` care va trimite detaliile introduse despre curs la baza de date pentru a fi stocate şi afişate apoi utilizatorilor care caută cursurile live şi se deschide pagina în care este integrat live stream-ul cu zona de mesaje şi transferul de fişiere – *se verifică dacă a fost introdus cursul în baza de date şi dacă conexiunile cu camera web şi microfonul sunt create şi dacă în zona alocată videoclipului se poate vedea ce se capturează cu camera.*
- la finalul cursului se apasă butonul `End Course` moment în care lectorul este direcţionat către pagina sa de profil şi se şterge din baza de date rândul care conţine cursul încheiat – *se verifică ştergerea cursului din baza de date şi redirectarea către pagina lectorului.*

În paralel cu acest scenariu s-au testa paşii pe care ii parcurge un utilizator student pentru a vedea un curs live. Paşii pentru acest scenariu sunt:

- deschiderea aplicaţiei web într-un browser - *se verifică conexiunea la Internet şi faptul că pe calculator există instalat un browser.*
- autentificarea pe site cu un cont creat pe platformele: Facebook, Google, Yahoo sau Linkedin. Dacă un utilizator nu are cont creat acesta va trebui să se înregistreze mai întai în secţiunea `Sign Up`. – *se verifică conexiunea la serverul Auth0* .
- se deschide meniul şi se alege "Live Streams", dupa care se alege un curs pentru a intra şi a-l urmari în direct – *se verifică cursurile care sunt aduse din baza de date şi se verifică dacă studentul poate urmari cursul şi poate să trimită întrebari.*
- se apasă butonul 'Leave Course' pentru a opri vizionarea cursului – *se verifică redirecţionarea către pagina de cursuri live.*

Un alt scenariu de test care a fost considerat util a fi verificat este crearea unui eveniment pentru un curs. Pasii parcurşi pentru acest scenariu sunt următorii:

- deschiderea aplicaţiei web într-un browser - *se verifică conexiunea la Internet şi faptul că pe calculator există instalat un browser.*
- autentificarea pe site cu un cont creat pe platformele: Facebook, Google, Yahoo sau Linkedin. Dacă un utilizator nu are cont creat acesta va trebui să se înregistreze mai întai în secţiunea `Sign Up`. – *se verifică conexiunea la serverul Auth0* .
- se deschide meniul şi se alege `My Events`, dupa care pentru un lector va fi vizibil butonul `Create Event`. – *se verifică faptul că butonul este vizibil.*
- se apasă pe buton pentru a deschide pagina de creare a unui eveniment şi se completează câmpurile corespunzatoare – *se verifică direcţionarea către pagina de creare de evenimente şi câmpurile de completat sunt cele din baza de date din tabelul `event`.*
- se apasă butonul `Create Event` pentru a trimite datele la baza de date – *se verifică înregistrarea evenimentului in baza de date.*
- lectorul poate să verifice dacă adăugarea evenimentului s-a efectuat cu succes deschizând încă o dată pagina 'My Events'. Sub tabelul cu evenimentele la care un lector s-a abonat, există un tabel în care sunt trecute toate evenimentele create de acel lector. Dacă noul eveniment este prezent în tabel înseamnă că adăugarea s-a efectuat cu succes şi acesta este vizibil pentru utilizatori pentru a se înscrie la el.

Fiecare scenariu de test conţine o cerinţă funcţională şi este testat separat pentru a verifica dacă este implementată corect acea cerinţă, iar la adăugarea unei funcţionalităţi noi se testează întreaga aplicaţie pentru a avea siguranţa că nouă funcţionalitate nu modifică sau distruge modul de funcţionare al aplicaţiei.

6.2. White Box Testing

Testarea prin metoda White Box (cutia albă) reprezintă testarea prin care se verifică structura internă a aplicaţiei şi cum sunt prelucrate datele. Pentru acest tip de testare, testărul are acces la codul sursă al aplicaţiei. El poate să introducă elemente de

testare prin care poate să determine dacă funcţiile sunt corect implementate, evaluând fiecare parametru şi funcţie. Analiza se realizează din punctul de vedere al corectitudinii codului scris dar şi al modului de organizare al acestuia.

Cea mai importantă parte de verificat este comunicarea dintre componenta client şi server. Între aceste componente se comunică prin metodele GET şi POST din protocolul HTTP. Cererile HTTP trimise de client la server se pot verifica cu ajutorul unui tool numit “Postman”. Acest tool permite simularea cererilor HTTP şi verificarea parametrilor trimişi şi a header-ului.

Prin verificarea şi compilarea codului se pot descoperi erori care pot fi rezolvate înainte de deshiderea aplicaţiei.

Capitolul 7. Manual de Instalare si Utilizare

7.1. Manual de Instalare

În acest subcapitol sunt prezentate resursele hardware şi software necesare pentru instalarea şi rularea aplicaţiei.

Un utilizator student are nevoie de un dispozitiv electronic cu o conexiune la Internet. Un utilizator lector are nevoie pe lângă un dispozitiv electronic conectat la Internet, o camera web şi un microfon. Componenta software necesară este un browser.

Pentru accesarea locală a aplicaţiei trebuie efectuate următoarele instalări şi descărcări:

- Se descarcă proiectul de pe BitBucket.
- În directorul unde se afla fişerul *server.js* se deschide Command Prompt şi se introduce comanda *npm install*. Pentru a fi recunoscută comanda *npm* trebuie instalat mai intai Node.js. Node.js se poate instala de pe pagina oficială a sa. (https://nodejs.org/en/)
- Pentru a deschide aplicaţia server se introduce în Command Prompt sau în linia de comandă, comanda *node server*.

Pentru a avea acces la aplicaţia client:

- Se deschide Command Prompt în directorul unde se găsesc fişierele *app.js, app.run.js* si *auth0-variables.js* şi se instalează dependenţele prin comanda *bower install*.
- Pentru a integra serviciul Auth0 în aplicaţie trebuie instalată librăria auth0.js prin comanda: *npm install --save auth0-js*.
- Apoi se introduce comanda *npm install –g http-sever* pentru a instala un server de web.
- Pentru a deschide aplicaţia web se introduce comanda *http-server* în Command Prompt sau în linia de comandă,
- Se accesează din browser link-ul: *http://localhost:8080*.
- Pentru conectarea cu serverul se instalează şi în directorul cu aplicaţia client, dependenţele prin *npm install*.

Baza de date a aplicaţiei se poate deshide în programul Workbench. După deschiderea lui se face conectarea la bazele de date locale. Se caută baza de date `*streams*` şi apăsând double click pe aceasta vor fi afişate tabelele pe care le conţine.

7.2. Manual de Utilizare

În acest subcapitol este explicat modul de utilizare al aplicaţiei web în funcţie de utilizatorul care este întregistrat, dar şi ce permisiuni are un utilizator care nu este înregistrat pe site.

Aplicaţie are o pagină principală la lansare, în care este prezentat pe scurt scopul pentru care a fost dezvoltată. Această pagină se schimbă în momentul în care utilizatorul se autentifică pe site.

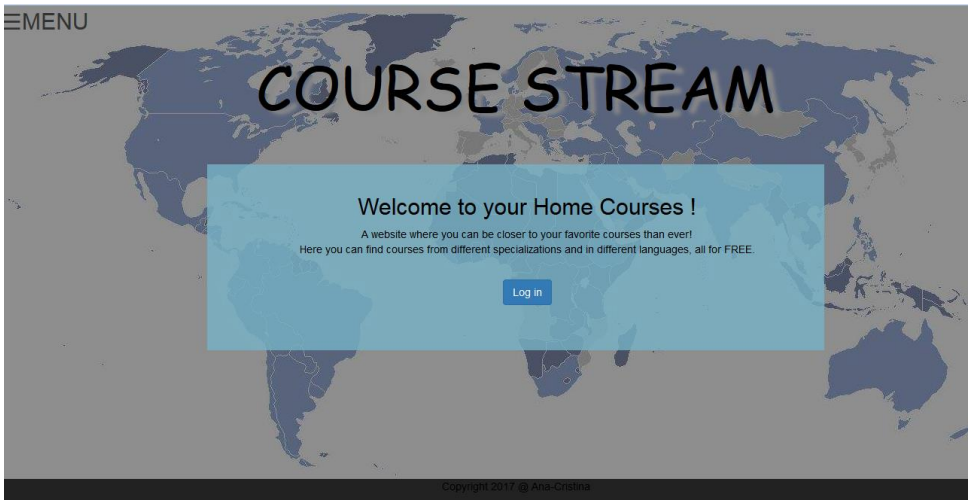

Figura.7.1 Pagina de Home pentru o persoana neautentificata

În momentul în care un utilizator doreşte să se autentifice în aplicaţie, acesta va fi direcţionat către pagină de autentificare din Auth0. Dacă persoana se autentifică pentru prima dată, ea va trebui să se înregistreze la *Sign Up*, pentru a-i fi salvat contul în serviciul de Auth0 şi la urmatoarele autentificari să fie recunoscut. În această pagină de *Log In* sau *Sign Up* o persoană are posibilitatea să se inregistreze folosind unul dintre următoarele conturi: Facebook, Google, Yahoo sau Linkedin.

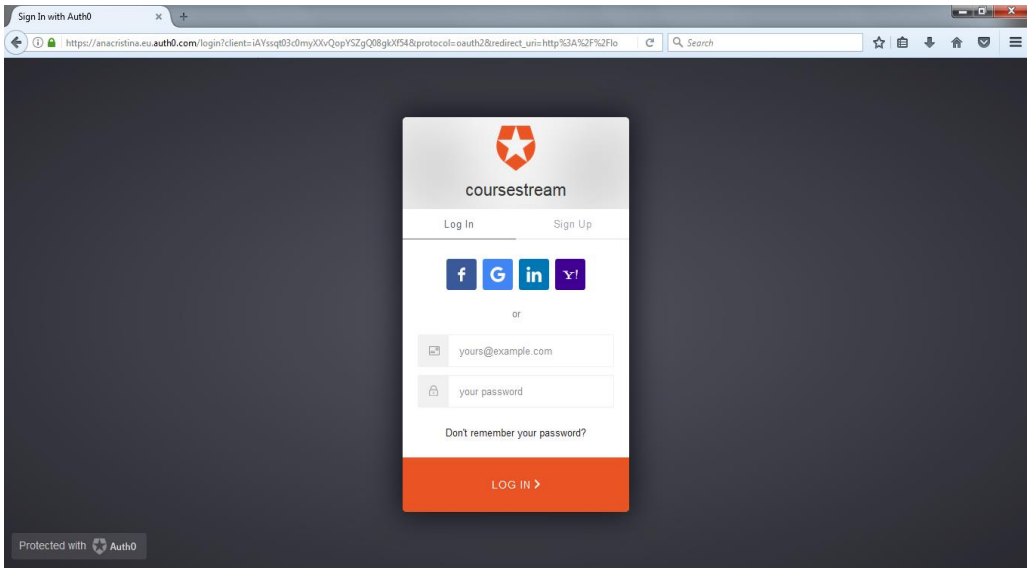

Figura 7.2 Pagina de autentificare

După înregistrare, utilizatorul va fi redirecţionat pe pagină de Home a aplicaţiei şi va putea beneficia de cursurile disponibile pe site. La prima autentificare un utilizator este rugat să îşi selecteze rolul pe acest site, student sau lector. Apasând pe butonul ‚*Register your Profile*', utilizatorul va fi automat adăugat în baza de date cu rolul pe care şi l-a ales. Acest rol nu va putea fi schimbat.

Figura 7.3 Înregistrarea profilului pentru prima dată

Dacă un utilizator nu se înregistrează el va putea să vizualizeze evenimentele care urmează să se desfăşoare, dar nu se va putea înscrie la acestea. Cursuri care sunt transmise live sunt disponibile numai pentru utilizatorii care sunt înregistraţi în aplicaţie.

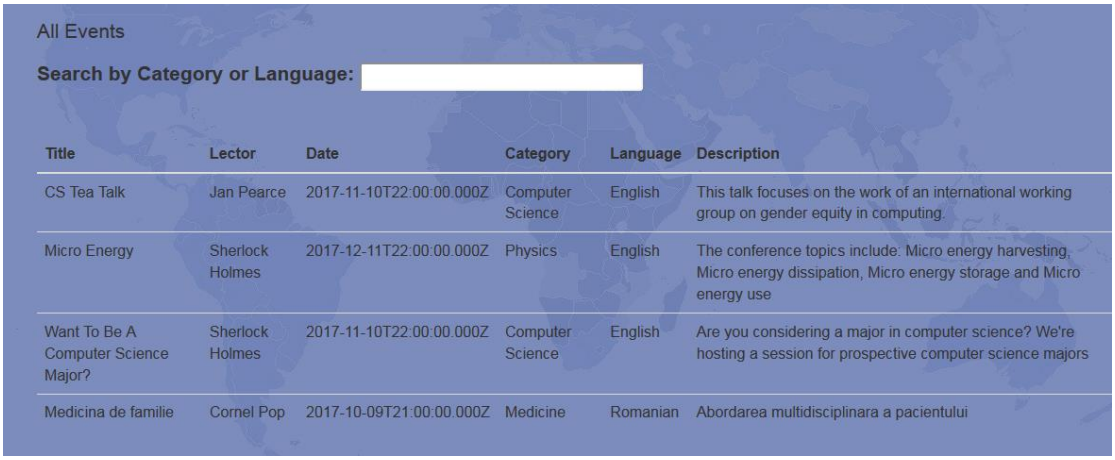

Figura 7.4 Pagina de evenimente pentru o persoană neautentificată

Orice persoană autentificată îşi poate vedea şi edita profilul din submeniul ‚Profile'.

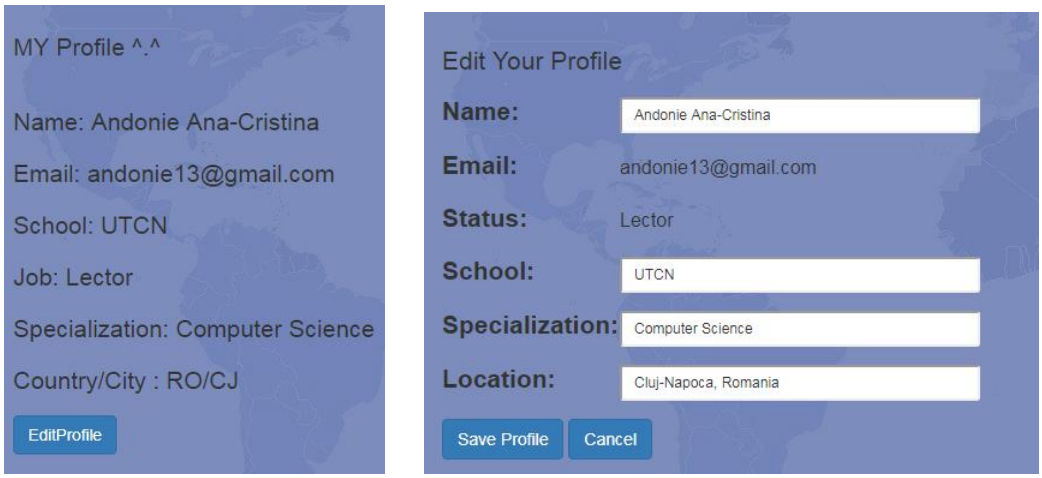

Figura 7.5 Profilul unui utilizator vizualizat si editat

Un student poate să vadă din submeniul ‚My Events' doar evenimentele la care s-a înscris. Un lector poate să vadă si evenimentele pe care le-a creat.

Figura7.6 Evenimentele l-a care s-a înscris o persoană

Din meniul de la elementul ‚Coming Up' un utilizator poate să vadă toate evenimentele care anunţă desfaşurarea unui viitor live streaming. Apăsând butonul de ‚Subscribe', utilizatorul se poate înscrie la eveniment.

Figura 7.7 Pagina de evenimente

Dacâ în dreptul unui eveniment este semnul `Canceled`, înseamnă că lectorul a anulat acel live stream, de accea nu se mai pot face înscrieri la evenimentul respectiv.

În căsuţa `*Search by Category or Language*` se poate căuta un eveniment după categoria din care face parte şi după limba în care se va preda cursul.

Figura 7.8 Căutare evenimente după categorie

Un lector poate să adauge un eveniment şi să transmită un live stream, iar la finalul acestuia poate să trimită un fişier care să conţine prezentarea cursului.

După ce au fost adăugate toate detaliile despre cursul care urmează a fi difuzat şi lectorul apasă butonul `*Create live*`, acesta va fi direcţionat către pagina în care se va deshide fereastra cu conţinutul video, camera web şi microfonul conectandu-se automat. Sub această fereastră este zona de chat în care cei care urmăresc cursul vor putea pune întrebări sau vor putea oferi un feedback lectorului în legătură cu modul de predare. Lectorul poate să răspundă şi în interiorul zonei de mesaje dar poate şi să citească mesajele şi să răspundă live la acestea.

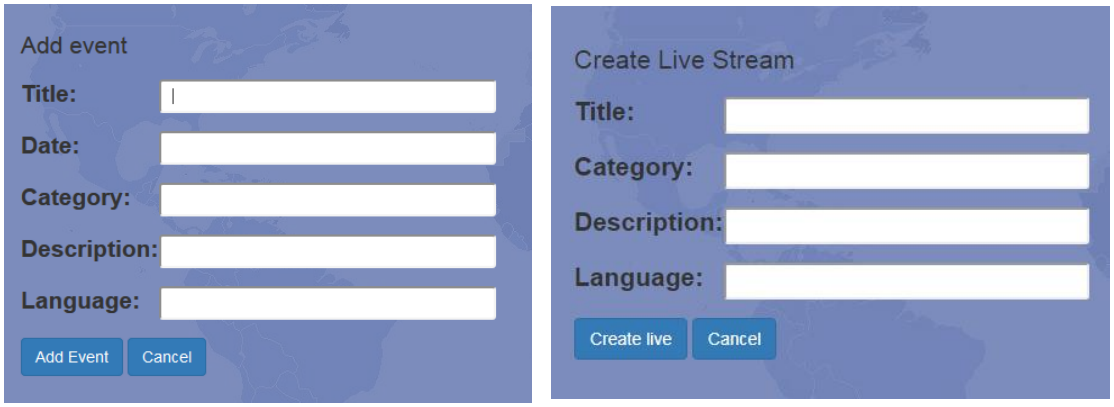

Figura 7.9 Adaugare eveniment si live stream

În pagina în care se transmite live stream-ul sunt integrate toate modurile de comunicare din acestă aplicaţie. Live stream-ul este inclus într-o zonă cu dimensiuni stabilite.

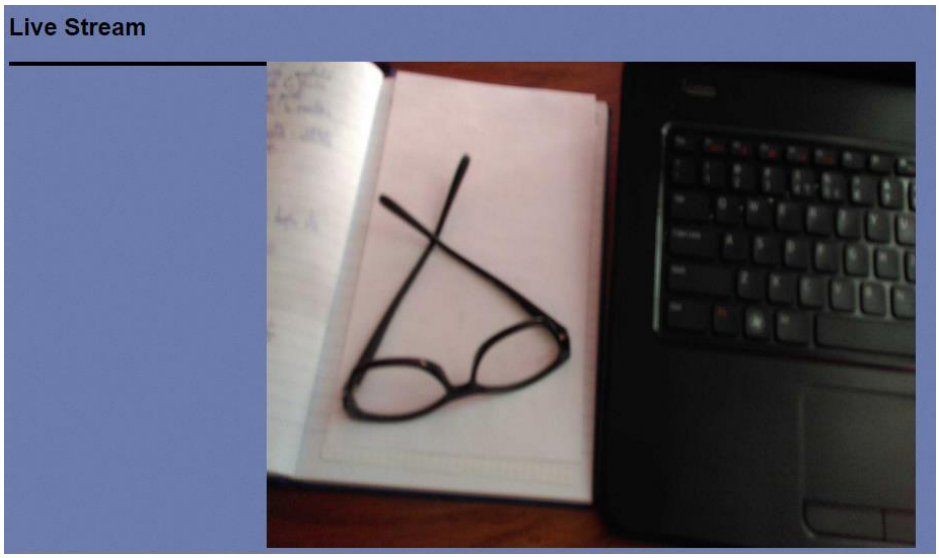

Figura 7.10 Live streaming

Comunicarea prin mesaje este amplasată sub zona de video. În stânga este zona în care un utilizator poate să scrie un mesaj, iar în dreapta este zona de chat în care sunt vizibile toate mesajele trimise în acel live stream. Pentru a trimite mesajul, utilizatorul trebuie sa apese butonul `*Send Message*`. Lectorul este conectat cu toţi utilizatorii şi poate să trimită mesaje private pentru a le raspunde întrebărilor.

Figura 7.11 Comunicarea prin mesaje

Sub zona de mesaje se află zona de transfer de fişiere. Lectorul poate să trimită un fişier numai după ce utilizatorul s-a conectat în această zonă de trimitere a fişiere.

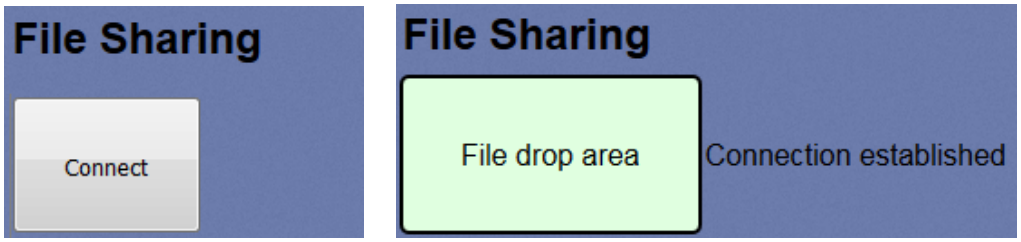

Figura 7.12 Acceptarea fişierelor de la lector

Lectorul trimite fişierul prin mecanismul `drag and drop` adică trage fişierul în zona `File drop area` pentru a fi trimis.

Figura 7.13 Trimiterea unui fişier şi primirea acestuia

Dacă utilizatorul acceptă fişierul i se va deschide o fereastră în care va putea alege unde să salveze fişierul în calculatorul propriu.

Capitolul 8. Concluzii

Cunoaşterea a fost şi este o cheie importantă în societate. Cu cât ştii mai multe cu atât ai şanse mai mari la a avea o viaţă împlinită pe plan personal dar şi profesional. Dorinţa de învăţare se naşte în contextul în care beneficiile ei durează o viaţă întreagă. Învăţarea nu are buton de oprire pentru că nu există niciodată un moment în care să ai toate cunoştintele.

Aplicaţia web dezvoltată în acest proiect este destinată persoanelor care doresc să îşi îmbunătăţească cunoştinţele într-un domeniu sau chiar mai multe. Interfaţa aplicaţiei este simplă pe înţelesul tuturor, iar ceea ce oferă este accesul la informaţia din confortul propriei case.

Obiectivul principal al aplicaţiei a fost construirea unui sistem care să permită transmiterea unui video live prin Internet cu scopul de a fi vizualizat de oricine, iar pe marginea lui, participanţii să poate să adreseze întrebări lectorului, în cazul în care ceva nu este clar pentru ei. Acest video live poate fi transmis dintr-o sala de curs unde sunt prezenţi oameni pentru a lua parte la curs, sau se poate transmite din biroul unei persoane, în acest caz toate persoanele care urmăresc cursul sunt online.Nu există nici o regulă cu privire la modul de predare atâta timp cât informaţia este prezentată în mod inteligibil si aduce un plus de cunoştinte legate de acel subiect. Aplicaţia dezvoltată poate fi considerată încă un mod de a comunica cu ajutorul tehnologiei.

8.1. Dezvoltari ulterioare

Sistemul dezvoltat livrează o aplicaţie uşor de utilizat care le oferă utilizatorilor un context în care au acces la informaţii prin urmărirea unui live stream. Acest sistem poate fi dezvoltat şi se pot adaugă mai multe cerinţe funcţionale.

Una dintre dezvoltările care poate avea loc este salvarea video-ului la încheierea acestuia. Acest lucru aduce un beneficiu major pentru cei care au dorit să urmărească cursul dar nu au reuşit, sau pentru cei care deşi l-au urmărit nu au reuşit să înţeleagă tot sau să îşi noteze. Salvarea conţinutului poate să se facă în aplicaţie sau cu ajutorul contului cu care s-a logat lectorul, într-o altă aplicaţie, fiind stocat pe o altă platformă. Dacă este stocat într-o aplicaţie, lectorul poate să decidă cât timp să rămână stocat şi după o perioada poate să îl şteargă. O îmbunătăţire adusă video-ul live care rămâne stocat este posibilitatea adăugării unei subtitrări pentru a putea fi vizualizat şi de persoane care nu cunosc limba vorbită de lector dar sunt interesaţi de subiect. Subtitrarea ar putea fi adăugată de către lector sau acesta poate să îi ofere accesul unei persoane pentru a încarcă subtitrarea.

O altă dezvoltare ar fi trecerea aplicaţie web la o aplicaţie mobilă. Dispozitivele mobile sunt obiectele cele mai la îndemână atunci când vine vorba de comunicarea la distanţă sau atunci când o persoană caută informaţii pe Internet. O aplicaţie mobilă ar fi mult mai practică de folosit pe un smartphone şi tabletă decât browser-ul. Utilizatorul ar putea interacţiona mai uşor prin aplicaţie iar transmiterea live streaming-ului s-ar putea realiza şi în mişcare.

O dezvoltare care ar putea îmbunătăţii comunicarea dintre doi oameni care doresc să discute în privat ar fi implementarea unui chat prin care doi utilizatori ar putea să discute direct, nu prin intermediul unui live stream. Profilurile utilizatorilor să fie vizibile

către alţi utilizatori astfel dezvoltându-se mici comunităţi de oameni care altfel nu s-ar fi întâlnit şi nu ar fi schimbat idei. Această dezvoltare ar putea aduce aplicaţia aproape de ideea de a fi o reţea de socializare dar care să rămână la bază o aplicaţie de predare-învăţare.

O dezvoltare a aplicaţiei în ceea ce priveşte tehnologia integrată în sistem, adică EasyRTC, este implementarea comunicării cu EasyRTC Enterprise un produs care urmează să fie lansat pe piaţă care include mai multe servicii de comunicare pentru clienţi. În EasyRTC Enterprise se implementează o parte din comunicare şi în server, folosind micro-servicii. Un beneficiu al acestei abordări este că o nouă funcţionalitate poate fi adăugată la un micro-serviciu fără a modifică alt micro-serviciu. Un alt avantaj este că dezvoltatorul nu mai trebuie să înţeleagă tot sistemul doar serviciul pe care doreşte să îl modifice.

Bibliografie

[1] Arhitectura peer-to-peer, Available: https://andrewmagdurulan.wordpress.com/networking/

[2] Rachel Ramsey, “What is Real Time Communications?”, Real Time Communication, Aprilie 2014, Available: http://www.realtimecommunicationsworld.com/topics/realtimecommunicationsworld/articles/376916-what-real-time-communications.htm

[3] John Sener, "Uodated E-Learning Definitions", OLC Insights, iulie 2015 Available: https://onlinelearningconsortium.org/updated-e-learning-definitions-2/

[4] Iuliana Dobre, "Studiu critic al actualelor sisteme de e-learning", Bucuresti, 2010 Available https://www.scribd.com/document/68626294/e-Learning

[5] Mariana Iacob, "Educatie formala, nonformala si informala", ianuarie 2014, Available: http://trainermarianaiacob.weebly.com/blog/educatia-formala-nonformala-si-informala

[6] TokBox, Available: https://tokbox.com/accelerator-pack/interactive-broadcast#spotlight-how-it-works

[7] Remo Meier, Master Thesis Project, Peer-to-Peer Live Streaming „StreamTorrent”, Swiss Federal Institute of Technology Zurich, 2006.

[8] XirSys, Available: https://xirsys.com/

[9] Apizee, Available: https://apizee.com/

[10] Twilio, Available: https://en.wikipedia.org/wiki/Twilio

[11] Ken Yeung, „Twilio teams up with T-Mobile to help developers build apps that use cellular data”, VentureBeat , Mai 2016, Available: https://venturebeat.com/2016/05/24/twilio- teams- with-t-mobile-so-developers-can-build-apps-using-cellular-data/

[12] MIT OpenCourseWare Available: https://ocw.mit.edu/about/

[13] Coursera Available: https://blog.coursera.org/about/

[14] Udacity Available: https://www.udacity.com/us

[15] Definiţie WebRTC, Available: https://en.wikipedia.org/wiki/WebRTC

[16] Arhitectura WebRTC, "WebRTC: Why and How?", FRAFOS Available: http://www.frafos.com/wp-content/uploads/2014/11/FRAFOS_WebRTC_Deployment.pdf

[17] EasyRTC, Available: https://easyrtc.com/docs/easyrtc_gettingStarted.php#what-is-easyrtc-opensource

[18] "PrioPhone - A next generation softphone that utilizes WebRTC technology", februarie 2014, Available: https://easyrtc.com/blog/2014-02-25-priophone-a-next-generation-softphone-that-utilizes-webrtc-technology/

[19] Node.js Tutorial, Available: https://www.tutorialspoint.com/nodejs/nodejs_tutorial.pdf

[20] Node.js Pagina Oficială, Available: https://nodejs.org/en/

[21] Arhitectura Node.js Available: https://blog.zenika.com/2011/04/10/nodejs/

[22] Mike Cantelon, Marc Harter, TJ. Holowaychuk, Nathan Rajlich, “Node.js in Action”, Shelter Island NY 2014.

[23] Evan M. Hahn, “Express in Action”, Shelter Island NY 2016

[24] Angular.js, Available: https://docs.angularjs.org/guide/introduction

[25] Mike Wasson "Single-Page Applications: Build Modern, Responsive Web Apps with ASP.NET", MSDN Magazine, noiembrie 2013, Available: https://msdn.microsoft.com/en-us/magazine/dn463786.aspx

[26] Auth0, Available: https://auth0.com/docs/quickstart/spa/angularjs

[27] "Calling APIs from Server-side Web Apps", Available: https://auth0.com/docs/api-auth/grant/authorization-code#overview-of-the-flow

[28] HTML5, Available: https://developer.mozilla.org/en/docs/Web/Guide/HTML/HTML5

[29] "Notiuni elementare despre bazele de date", Available: https://support.office.com/ro-ro/article/No%C8%9Biuni-elementare-despre-bazele-de-date-a849ac16-07c7-4a31-9948-3c8c94a7c204

[30] Curs Sisteme de Calitate în Tehnologia Informaţiei, anul IV, semestrul II, profesor Iulia Costin, cursul 1.

[31] Structura generală a unei baze de date relaţionale, Available: https://en.wikipedia.org/wiki/Relational_database_management_system

Anexa 1 - Lista figurilor din lucrare

Figura 3.1. Arhitectura unei retele peer-to-peer..... 6
Figura 3.2 Model conceptual pentru un sistem e-learning 8
Figura .4.1. Arhitectura WebRTC 14
Figura. 4.2 Arhitectura EasyRTC 16
Figura 4.3. Arhitectura Node.js 17
Figura 4.4 Fluxul unei cereri in Express. 18
Figura 4.5 Diferenţa între fluxul unei pagini traditionale HTML şi o pagina SPA 20
Figura 4.6 Legătura dintre Auth0 şi un API 21
Figura 4.7 Structura generală a unei baze de date 23
Figura 4.8. Diagrama use-case pentru utilizatorul lector 27
Figura 4.9 Diagrama use-case pentru utilizatorul student 29
Figura 5.1. Arhitectura generală a sistemului 32
Figura 5.2 Diagrama relaţiei peer-to-peer 33
Figura 5.3 Diagrama de deployment 34
Figura 5.4 Meniu de navigare pentru persoane autentificate şi neautentificate 42
Figura 5.5 Structura bazei de date 46
Figura.7.1 Pagina de Home pentru o persoană neautentificată 53
Figura 7.2 Pagina de autentificare 53
Figura 7.3 Înregistrarea profilului pentru prima dată.....55
Figura 7.4 Pagina de evenimente pentru o persoană neautentificată 55
Figura 7.5 Profilul unui utilizator vizualizat si editat 55
Figura7.6 Evenimentele l-a care s-a înscris o persoană 56
Figura 7.7 Pagina de evenimente 56
Figura 7.8 Căutare evenimente dupa categorie 56
Figura 7.9 Adaugare eveniment şi live stream 57
Figura 7.10 Live streaming 57
Figura 7.11 Comunicarea prin mesaje 58
Figura 7.12 Acceptarea fişierelor de la lector 58
Figura 7.13 Trimiterea unui fişier şi primirea acestuia 58

Anexa 2 - Lista tabelelor din lucrare

Tabel 3.1 Comparaţie StreamTorrent şi CourseStream 10
Tabel 3.2 Comparaţie sisteme similare 11
Tabel 4.1. Cerinţe funcţionale 24
Tabel 4.2 Use-case-uri pentru utilizatorul lector 27
Tabel 4.3 Use-case-uri pentru utilizatorul student 29

Anexa 3 - Glosar de termeni

AJAX – Asynchronous JavaScript and XML
API – Application Programming Interface
CORS–Cross-Origin Resource Sharin
CRUD – Create Read Update Delete
DOM – Document Object Model
DNS – Domain Name Server
HTML – HyperText Markup Language
HTTP – Hypertext Transfer Protocol
ICE - Interactive Connectivity Establishment
IETF - Internet Engineering Task Force
ILBC – Internet Low Bitrate Codec
ISAC – Internet Speech Audio Codec
JSON – JavaScript Object Notation
JWKS-RSA – JSON Web Key Set - RSA
JWT – JSON Web Token
NAT - Network Address Translation
OTT – Over The Top
RDBMS – Relational database management system
RTP – Real-time Transport Protocol
SPA – Single Page Application
SQL – Srtuctured Query Language
SRTP – Secure Real-time Transport Protocol
SSL – Secure Socket Layer
STUN – Session Traversal Utilities for NAT
TCP – Transmission Control Protocol
TURN - Traversal Using Relay NAT
UDP – User Datagram Protocol
UI – User Interface
WebRTC – Web Real Time Communication
XML – eXtensible Markup Language