**TITLUL LUCRĂRII DE LICENȚĂ**

LUCRARE DE LICENŢĂ

|  |  |  |
| --- | --- | --- |
|  | Absolvent: | **Szabolcs BENE** |
|  |  |  |
|  | Coordonator ştiinţific: | **Assist. Prof. Eng. Cosmina IVAN** |

**2016**

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | |  |
| DECAN, |  | | DIRECTOR DEPARTAMENT, |
| **Prof. dr. ing. Liviu MICLEA** |  | **Prof. dr. ing. Rodica POTOLEA** | |

Absolvent: **Szabolcs BENE**

**TITLUL LUCRĂRII DE LICENŢĂ**

1. **Enunţul temei:**Proiectul propus rezolvă problema comunicării în timp real printr-o aplicație web, fără folosirea pluginurilor sau a programelor externe. Acest proiect oferă posibilitatea utilizatorului să se conecteze peer-to-peer la alți utilizatori, fără folosirea unui server intermediar, care ar avea rolul să transmită datele și informațiile de la un utilizator la altul. Avantajul oferit de acest proiect este reducerea timpului de creare a conexiunii și a datelor transmise de utilizator, iar interfața grafică este ușor utilizabilă
2. **Conţinutul lucrării:** *Pagină de prezentare, Introducere, Obiectivele Proiectului, Studiu Bibliografic, Analiză şi Fundamentare Teoretică, Proiectare de Detaliu şi Implementare, Testare şi Validare, Manual de Instalare şi Utilizare, Concluzii, Bibliografie.*
3. **Locul documentării**: Universitatea Tehnică din Cluj-Napoca, Departamentul Calculatoare
4. **Consultanţi**: Assist. Prof. Eng. Cosmina IVAN
5. **Data emiterii temei:** 1 noiembrie 2015
6. **Data predării:** 28 Iunie 2016

|  |  |
| --- | --- |
| Absolvent: | Bene Szabolcs |
|  |  |
| Coordonator ştiinţific: | Assist. Prof. Eng. Cosmina IVAN |

**Declaraţie pe proprie răspundere privind**

**autenticitatea lucrării de licenţă**

Subsemnatul(a) Szabolcs BENE, legitimat(ă) cu Carte de Identitate seria KX nr. 721039   
CNP 1911225124605, autorul lucrării, \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_elaborată în vederea susţinerii examenului de finalizare a studiilor de licență la Facultatea de Automatică și Calculatoare, Specializarea \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ din cadrul Universităţii Tehnice din Cluj-Napoca, sesiunea \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ a anului universitar \_\_\_\_\_\_\_\_\_\_, declar pe proprie răspundere, că această lucrare este rezultatul propriei activităţi intelectuale, pe baza cercetărilor mele şi pe baza informaţiilor obţinute din surse care au fost citate, în textul lucrării, și în bibliografie.

Declar, că această lucrare nu conţine porţiuni plagiate, iar sursele bibliografice au fost folosite cu respectarea legislaţiei române și a convenţiilor internaţionale privind drepturile de autor.

Declar, de asemenea, că această lucrare nu a mai fost prezentată în faţa unei alte comisii de examen de licenţă.

In cazul constatării ulterioare a unor declaraţii false, voi suporta sancţiunile administrative, respectiv, *anularea examenului de licenţă*.

|  |  |  |
| --- | --- | --- |
| Data: 4 iulie 2016 |  | Bene Szabolcs |
|  |  | Semnătura |
|  |  |  |

**Cuprins**

[Capitolul 1. Introducere 7](#_Toc454222624)

[1.1. Contextul proiectului 7](#_Toc454222625)

[1.2. Motivația 7](#_Toc454222626)

[1.3. Continutul lucrarii 8](#_Toc454222627)

[Capitolul 2. Obiectivele Proiectului 9](#_Toc454222628)

[2.1. Obiectivul principal 9](#_Toc454222629)

[2.2. Obiective secundare 9](#_Toc454222630)

[Capitolul 3. Studiu Bibliografic 11](#_Toc454222631)

[3.1. Dezvoltarea aplicatiilor web 11](#_Toc454222632)

[3.2. Comunicare in timp real 11](#_Toc454222633)

[3.3. WebRTC 13](#_Toc454222634)

[3.3.1. Arhitectura WebRTC 13](#_Toc454222635)

[3.3.2. Media Path 15](#_Toc454222636)

[3.4. WebRTC vs WebSockets 16](#_Toc454222637)

[3.5. WebRTC API 18](#_Toc454222638)

[3.5.1. Negocierea informatiilor de reţea 19](#_Toc454222639)

[3.5.2. Stream API 23](#_Toc454222640)

[3.5.3. PeerConnection API 24](#_Toc454222641)

[3.5.4. Data Channel API 24](#_Toc454222642)

[3.6. Sisteme similare 25](#_Toc454222643)

[Capitolul 4. Analiză și Fundamentare Teoretică 28](#_Toc454222644)

[4.1. Tehnologii și unelte utilizate pentru dezvoltarea aplicației web 28](#_Toc454222645)

[4.1.1. PHP 28](#_Toc454222646)

[4.1.2. JavaScript 29](#_Toc454222647)

[4.1.3. HTML 29](#_Toc454222648)

[4.1.4. CSS 30](#_Toc454222649)

[4.1.5. GitHub 31](#_Toc454222650)

[4.1.6. Apache Server 31](#_Toc454222651)

[4.1.7. XAMPP 32](#_Toc454222652)

[4.2. Cerințele sistemului 32](#_Toc454222653)

[4.2.1. Cerinte functionale 33](#_Toc454222654)

[4.2.2. Cerinte functionale 34](#_Toc454222655)

[4.3. Cazuri de utilizare 35](#_Toc454222656)

[Capitolul 5. Proiectare de Detaliu și Implementare 39](#_Toc454222657)

[5.1. Arhitectura de layer a sistemului 39](#_Toc454222658)

[5.1.1. Nivelul interfata utilizator 40](#_Toc454222659)

[5.1.2. Nivelul de aplicatie 40](#_Toc454222660)

[5.1.3. Nivelul de business logic 40](#_Toc454222661)

[5.1.4. Nivelul de acces de date 40](#_Toc454222662)

[5.1.5. Nivelul de baza de date 41](#_Toc454222663)

[5.2. Arhitectura MVC 41](#_Toc454222664)

[5.3. Patternul Front Controller 43](#_Toc454222665)

[5.4. Componentele aplicatiei 45](#_Toc454222666)

[5.4.1. Componenta de server 47](#_Toc454222667)

[5.4.2. Componenta de client 51](#_Toc454222668)

[5.4.3. Stabilirea conexiunii intre doi utilizatori 55](#_Toc454222669)

[5.4.4. Crearea ofertei 56](#_Toc454222670)

[5.4.5. Asteptarea ofertei 58](#_Toc454222671)

[5.4.6. Asteptarea raspunsului 58](#_Toc454222672)

[5.4.7. Asteptarea candidatilor 59](#_Toc454222673)

[5.4.8. Trimiterea datelor prin DataChannel 59](#_Toc454222674)

[5.4.9. Primirea datelor de pe DataChannel 60](#_Toc454222675)

[5.4.10. Vizualizarea istoricului 61](#_Toc454222676)

[5.5. Proiectarae bazei de date 61](#_Toc454222677)

[5.6. Concluzie 62](#_Toc454222678)

[Capitolul 6. Testare şi Validare 63](#_Toc454222679)

[Capitolul 7. Manual de Instalare și Utilizare 67](#_Toc454222680)

[7.1. Instalare si rulare 67](#_Toc454222681)

[7.1.1. Instalare Git 67](#_Toc454222682)

[7.1.2. Instalare XAMPP 68](#_Toc454222683)

[7.2. Utilizarea aplicatiei 71](#_Toc454222684)

[Capitolul 8. Concluzii 73](#_Toc454222685)

[8.1. Realizarea obiectivelor propuse 73](#_Toc454222686)

[8.2. Dezvoltari ulterioare 73](#_Toc454222687)

[Bibliografie............................. 75](#_Toc454222688)

# Introducere

În momentul de faţă, în orice domeniu, fie IT, marketing, construcţii sau multe altele, este nevoie de comunicare rapidă în timp real, prin folosirea internetului şi a browserelor web, fără instalarea programelor sau toolurilor, şi care să meargă pe orice fel de platformă. Reducerea timpului de transmitere a datelor şi eficacitatea comunicaţiilor în timp real este ceea ce se urmăreşte a fi realizat prin simplificarea activităţilor în spaţiul web.

## Contextul proiectului

Într-o lume tot mai aglomerată de tehnologie, care joacă un rol din ce în ce mai important, fiecare doreşte să fie informat într-un timp cât mai scurt de activităţile de care este interesat şi să îi fie oferit accesul cât mai rapid la informaţiile necesare.

Sistemul propus se încadrează în acest context de comunicație în timp real, combinând elementele de socializare din lumea modernă, cum ar fi Facebook-ul sau Skype-ul cu elementele propuse spre a fi implementate de acest proiect. Sistemul nu a fost creeat specific industriei de IT, deoarece este nevoie de comunicare internă sau externă în timp real, în orice industrie.

Sistemul dorește cu ajutorul actorilor, şi anume a utilizatorilor, principali şi singulari, să utilizeze funcţionalităţile oferite de aplicaţia web. Utilizatorii beneficiează de simplitatea şi eficienţa realizării unei conexiuni între ei, şi de comunicare în timp real prin text, audio său video.

Sistemul înglobează toate nevoile utilizatorului şi urmăreşte eficientizarea procesului de comunicare între aceşti utilizatori, prîntr-o aplicaţie web, care rulează pe ultimele versiuni ale browserului. Folosind acest sistem, utilizatorul nu are nevoie să instaleze plugin-uri sau alte programe, pentru a putea realiza o comunicare directă cu un alt utilizator în timp real.

## Motivația

Ţinând cont de timpul necesar conexiunii online prin video său text, sau de transmiterea datelor în timp eficient, a apărut ideea implementării unui sistem care oferă toate funcţionalităţile necesare de care are nevoie un client pentru a realiza o comunicare directă şi bidirecţională în timp real, prin folosirea tehnologiei WebRTC oferite de browserele web. WebRTC este o tehnologie folosită în browserele web, care oferă posibilitatea utilizatorilor să se conecteze peer-to-peer prin aceste browsere și să transmită mesaje de tip text sau date de audio și video. S-a constatat faptul că pentru a comunica cu cineva în timp real, de multe ori este nevoie de un server, care să transmită datele necesare de la un client la altul. Prin folosirea tehnologiei WebRTC, s-a evitat folosirea unui server, iar utilizatorii se pot conecta unul cu celălalt, numai prin Internet, iar riscul de pierdere al conexiunii la server sau al pierderilor de date este minimizat.

Avantajul acestei aplicaţii web este că elimină librăriile externe şi pluginurile de care era necesar să se realizeze o conexiune între browsere, şi creeând conexiuni peer-to-peer, se măreşte viteza de trimitere a datelor şi a fişierelor între utilizatori.

## Continutul lucrarii

În următoarele paragrafe se va prezenta structura lucrării, capitolele și conținutul acestora.

**Capitolul 1 – Introducere** – Acest capitol prezintă contextul problemei care a fost rezolvată prin aplicaţia propusă

**Capitolul 2 – Obiectivele Proiectului** – Este prezentat obiectivului principal al proiectului, şi a obiectivelor secundare care au fost propuse spre îndeplinirea unei comunicaţii bidirectionale în timp real.

**Capitolul 3 – Studiu Bibliografic** – Acest capitol înglobează studiul necesar dezvoltării aplicațiilor web, cât și studiul necesar pentru folosirea tehnologiei WebRTC. Va fi prezentat un studiu relevant cu privire la tehnologia WebRTC si functionarea sa. Totodată, acest capitol prezintă asemănările și deosebirile între funcționalitățile aplicației și a aplicațiilor similare disponibile.

**Capitolul 4 – Analiză și Fundamentare Teoretică** – În acest capitol sunt prezentate tehnologiile folosite in aplicație si motivele pentru care s-au ales acele tehnologii. Mai sunt si prezentate cerințele funcționale si nonfuncționale ale aplicației web.

**Capitolul 5 – Proiectare de Detaliu și Implementare** – Acest capitol prezintă arhitecturile si design patternurile folosite în proiect, si prezentarea în detaliu a fiecărei componente. Diagrama de arhitectura a sistemului, diagramele de clase, diagramele de navigare precum și diagrama de secvente sunt prezentate tot în acest capitol. Componentele importante ale sistemului, serverul si clientul, sunt explicate în detaliu. La finalul acestui capitol este prezentată și arhitectura bazei de date.

**Capitolul 6 – Testarea, Validarea și Evaluarea** – Acest capitol prezintă metodele de testare ale aplicației si pașii care s-au urmat pentru îndeplinirea validarea sistemului.

**Capitolul 7 – Manual de Instalare și Utilizare** – Acest capitol prezintă resursele necesare ce trebuie să le aibă un utilizator pentru rularea si instalarea aplicației. Este prezentat și manualul de utilizare, dar si pașii de instalare a programelor necesare.

**Capitolul 8 – Concluzii** – Acest capitol prezintă obiectivele care s-au realizat și dezvoltările ulterioare ce pot fi implementate în viitor.

# Obiectivele Proiectului

În acest capitol vor fi prezentate obiectivele propuse spre a fi realizate. Sistemul are scopul de a eficientiza comunicarea online în timp real, cu ajutorul browserelor, care nu se rezumă doar la comunicare în text ci la aproape tot ce înseamnă comunicație în timp real, incluzând mecanismul de transmitere de fişierelor într-un mod eficient.

## Obiectivul principal

Obiectivul principal al acestui proiect reprezinta proiectarea, definirea si construirea unui sistem care să ofere functionalitati de comunicație online in timp real, fără a fi nevoită folosirea unor pluginuri sau a unor programe externe, obiectiv care se poate atinge prin realizarea obiectivelor secundare care urmeaza a fi prezentate.

## Obiective secundare

**Obiective generale propuse**

Primul dintre obiectivele secundare pe care sistemul doreşte să le atingă este eficienţa în comunicare în timp real, care se concretizează prin folosirea interfeţei grafice. Deoarece interfaţa grafică de utilizator este simplă şi uşor utilizabilă, utilizatorul poate realiza o comunicare directă cu un alt utilizator într-un timp foarte scurt.

Un alt obiectiv ar fi ca sistemul sa poată fi dezvoltat în continuare, și sa fie posibile să se extindă funcționalitățile.

Sistemul trebuie să le ofere utilizatorilor posibilitatea de autentificare simplă, cât şi autentificare cu conturi din reţele de socializare, un exemplu fiind Facebook-ul. În lumea modernă se doreste că toate aplicaţiile web să interacţioneze cu reţelele de socializare, pentru eficientizarea logarii în aplicaţie şi căutarea sau adăugarea unor prieteni noi. Obiectivul propus este că utilizatorii să se poată conecta la conturile lor private din reţelele de socializare de pe conturile aplicaţiei, şi astfel să se poate loga folosind contul de Facebook, fără a mai introduce numele de utilizator şi parola. Dacă utilizatorul foloseşte autentificarea simplă oferită de aplicaţie, şi introduce numele de utilizator sau o parolă greşită, atunci logarea va eşua, iar utilizatorul va fi notificat cu un mesaj de eroare.

Sistemul trebuie să ofere şi înregistrarea conturilor noi. Obiectivul propus este că un utilizator nou să-şi poată creea un cont nou din aplicaţie, pe baza unui set de reguli. Dacă regurile sunt respectate, atunci utilizatorul îşi va putea creea propriul său cont în aplicaţie. În cazul în care regurile nu sunt respectate, cum ar fi de exemplu introducerea unui nume de utilizator deja existent, sau introducerea unui cont de email în format invalid, atunci utilizatorul va fi notificat cu un mesaj de eroare, mesajul conţinând motivul pentru care utilizatorul nu a putut să-şi creeze un cont nou.

Sistemul trebuie să ofere adăugarea prietenilor noi. Obiectivul propus este că un utilizator să poată să adauge în lista de prieteni goală, sau deja existentă, prieteni noi, prin introducerea numelui de utilizator. În cazul în care nu există numele de utilizator, atunci utilizatorul va fi notificat prîntr-un mesaj de eroare. Un alt obiectiv propus legat de lista de prieteni este actualizarea automată a listei. Dacă utilizatorul a fost adăugat de un alt utilizator, atunci numele acestui utilizator trebuie să apară în lista cu prieteni. Totodată lista de prieteni trebuie să afişeze care utilizator este online, şi care nu.

**Obiective de comunicare între utilizatori**

Sistemul trebuie să ofere o posibilitate pentru utilizatori de a se conecta şi de a comunica cu alţi utilizatori. Obiectivul propus este că utilizatorul să poată alege din lista de prieteni, şi prîntr-o simplă apăsare de buton să ceară permisiunea să se conecteze la cel de-al doilea utilizator. Un alt obiectiv legat de cererea de conexiune este posibilitatea de a accepta sau a refuza cererile de conexiune. Dacă conexiunea este refuzată de un utilizator, atunci cererea este terminată automat. Dacă utilizatorul acceptă conexiunea, atunci conexiunea va trece la paşii următori în mod automat, fără a mai introduce date suplimentare, iar după conectare cererea de conexiune se realizează cu succes.

Sistemul trebuie să ofere diferite metode de comunicare între utilizatori. Obiectivul propus este că cei doi utilizatori deja conectaţi să poată comunica între ei cu ajutorul mesajelor de tip text, sau cu ajutorul sunetului şi a afişării video.

Sistemul trebuie să ofere utilizatorului posibilitatea de a comunica cu mai mulţi utilizatori în aceaşi timp. Obiectivul propus este că la selectarea unui prieten din lista de prieteni, să apară o fereastră nouă, special pentru cel de-al doilea utilizator, astfel fiecare utilizator din lista de prieteni având fereastra de comunicare respectivă. Dacă se schimbă focusul ferestrei, se schimbă şi sursele video, iar dacă se închide fereastra atunci se va închide şi conexiunea dintre utilizatori.

Sistemul trebuie să ofere utilizatorului posibilitatea de a vizualiza conversaţiile anterioare cu alţi utilizatori. Obiectivul propus este că la afişarea ferestrei de comunicare, se afişează şi un link, care va duce utilizatorul pe o pagină nouă, unde utilizatorul va putea vizualiza conversaţia cu celălalt utilizator. Precondiţia acestei funcţionalităţi este că utilizatorul să fie logat în aplicaţie.

Sistemul trebuie să ofere utilizatorului posibilitatea de a transmite fişiere către celălalt utilizator în timp real. Obiectivul proupus este că utilizatorul să poată alege fişierul dorit la o apăsare de buton, şi la apăsarea butonului "Send", să trimită fişierul selectat. La primirea fişierului, cel de-al doilea utilizator va putea accesa şi descarcă fişierul transmis prin apăsarea unui link care va fi generat la transmiterea fişierului.

# Studiu Bibliografic

În acest capitol se realizează analiza şi evaluarea atât a comunicării în timp real, cât şi a unui set de aplicaţii care folosesc tehnologii de comunicare în timp real. Se va descrie ce înseamnă o aplicaţie web, şi se vor diferenţia soluţiile posibile pentru a realiza comnuicarea în timp real.

## Dezvoltarea aplicatiilor web

Când se dezvoltă o aplicație web, functionalitatile aplicației sunt separate de mediul în care sunt rulate. Pentru accesarea resurselor, pe care le acceseaza utilizatorul prin aplicația web, se face prin adrese URI, folosind protocolul HTTP.

Aplicațiile web au ca și arhitectură Client-Server, unde Clientul este cel cu care interacționează utilizatorul, iar Serverul este cel care răspunde la interogările utilizatorului.

Cererile importante în protocolul HTTP sunt GET şi POST. GET este foarte mult utilizat pentru a obţine resursele, iar prin POST se introduc date în conţinutul mesajului transmis serverului.

## Comunicare in timp real

Comunicarea in timp inseamna o telecomunicatie in orice mod, in care utilizatorii pot interschimba orice fel de informatii, text, audio sau video, instant sau cu intarzieri foarte mici.

În figura 3.1, se pot observa elementele care construiesc comunicarea în timp real, şi cum sunt conectate la utilizator.



Figura 3.1 Elementele de comunicare in timp real

Comunicarea reală se poate împărţi în două moduri: comunicare half-duplex şi comunicare full duplex. În comunicarea half-duplex, informaţiile pot fi transmise în ambele direcţii, dar nu în acelaşi timp, în schimb în comunicarea full-duplex, informaţiile se pot transmite în ambele direcţii, în acelaşi timp. În general când vorbim despre RTC, atunci înţelegem comunicare peer-to-peer, nu broadcast sau multicast.

În comunicarea în timp real, există întotdeauna o cale directă între sursă şi destinaţie, fără că informaţiile să fie stocatepe durata transmiterii informaţiilor. În comparaţie cu comunicările de tip timeshifting, informaţiile sunt întotdeauna stocate undeva pe un server. În această categorie de timeshifting intră comunicările prin mail sau comunicări prin comentarii în interiorul reţelei de socializare Facebook, unde informaţiile sunt salvate pe server, iar între transmiterea şi recepţia informaţiilor există o anumită întârziere. În categoria comunicării în timp real intră următoarele: comunicare prin telefon, prin mesaje instanţe, VoIP și conferinţe audio/video..

## WebRTC

În acest subcapitol se va descrie ce înseamnă tehnologia WebRTC, şi cum s-a ajuns să existe comunicare în timp real pe web, fără a folosi librării externe sau pluginuri.

WebRTC este un proiect utilizabilce asigura comunicare în timp real prin browsere web şi aplicaţii mobile. Misiunea a fost, de la început, de a oferi funcţionalităţi de calitate RTC(Real Time Communication), şi care să se poată integra în diferite platforme mobile sau web, care permit comunicarea prîntr-un set de protocoale. WebRTC este susţinut de diferite platforme cum ar fi Chrome, Mozilla sau Opera, şi este întreţinut de echipele Google Chrome.

În Mai, 2011, Google a pornit un proiect open-source pe web, pentru RTC, numit WebRTC.[10] După acest proiect, s-au pornit şi o serie de implementări pentru standardizarea protocoalelor relevante în IETF[9] şi pentru API-ul respectiv W3C.[12] API-ul a fost realizat după un proiect anterior realizat de WHATWG[11] şi a fost denumit că şi ConnectionPeer API, iar implementarea conceptului pre-standardelor a fost realizată la Ericsson Labs.[13]

WebRTC vine cu o gama mare de funcţionalităţi ce permite comunicarea de tip text, audio şi video, doar prin folosirea browserului. WebRTC-ul permite şi partajarea fişierelor sau a ecranului utilizatorului, dar şi multe altele.

Tehnologia vine cu avantaje mari, când este vorba de RTC. În primul rând, funcţionalităţile oferite sunt uşor folosite şi implementate, şi nu este nevoie de folosirea unei librării externe sau pluginuri pentru browsere. Tot ce este nevoie este folosirea unui protocol de instanţiere a semnalelor, ce are rolul de a transmite informaţii de conexiune de la un client la altul. Al doilea avantaj este că forţează programatorii web să implementeze propria lor modalitate pentru securitate şi incriptarea datelor.

Avantajele acestei tehnologii se răspândesc nu numai în afara companiilor, ci şi în interiorul lor. Se reduce costul telefoanelor interne prin înlocuirea lor cu aplicaţie WebRTC, şi se poate oferi comunicare bogată şi neîntreruptă între clienţi şi angajaţi.

### Arhitectura WebRTC

Arhitectura WebRTC-ului este una complexă, dar folosirea functionalitatilor, oferite de această tehnologia, sunt uşor utilizabile. Arhitectura a fost creată astfel încât programatorii să poată folosi şi integra cu uşurinţă funcţionalităţile WebRTC-ului, iar utilizatorii să poată folosi aplicaţiile web fără descărcarea unei librării externe sau pluginuri pentru browsere.

Arhitectura totală se poate observa în figura 3.2:

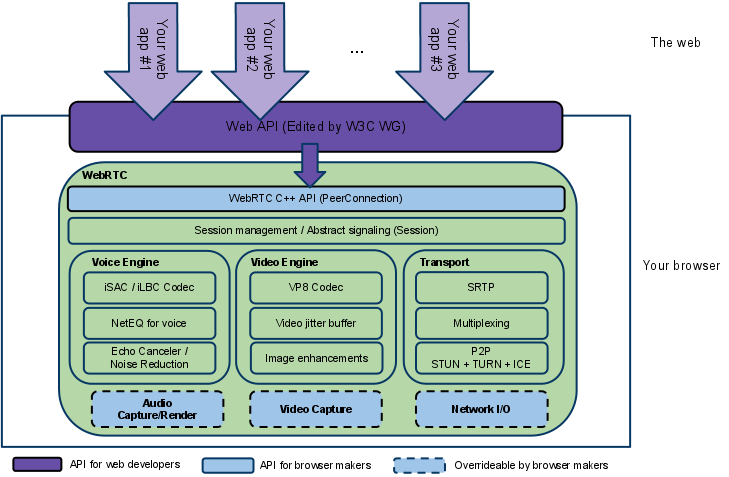


Figura 3.2 Arhitectura WebRTC

.

La prima vedere se pot observa două layere distincte:

1. Programatorii browserelor sunt interesaţi în API-ul WebRTC C++
2. Programatorii de aplicaţii web sunt interesanţi de API-ul Web

Aplicaţiile web vor folosi capabilitaţile de comunicare în timp real oferite de API-ul Web, oferit de tehnologia WebRTC.

Multe componente sunt abstractizate, oferind creatorilor de aplicaţii web să implementeze propriile lor modalităţi de gestiune a datelor cum ar fi componenta de transport şi sesiuni folosite pentru diferite protocoale de transport, componenta de STUN/ICE care este folosită pentru stabilirea conexiunii sau componentă de gestionare a sesiunilor.

### Media Path

Pentru calea de media se foloseste Real Time Protocol (RTP). Conform schițelor de specificatii [5], se consideră urmatoarele topologii, ce se pot implementa cu WebRTC.

**Peer-to-peer**

****

Figura 3.3 peer-to-peer

Este cazul standard de utilizare pentru RTC, o conexiune unica point-to-point cu clienții respectivi implementând WebRTC.

**Multi-Unicast**



Figura 3.4 Multicast

In aceasta topologie sunt stabilite mai multe conexiuni de WebRTC. Dezavantajul acestei topologii este ca creste cererea de bandwidth si de retea la cresterea utilizatorilor, care intra in aceasta topologie.

**Mixer RTP cu doar un Unicast**



Figura 3.5 Mixer RTP

Mixerul RTP este un punct centralizat care are rolul unui client RTC, dar care are şi controlul asupra audio/video-ului folosit în conferinţe. Acest client are responsabilitatea de a optimiza toate streamurile şi lăţimea de banda, creând un singur stream ce va fi transmis în continuare.

## WebRTC vs WebSockets

Pentru a implementa o aplicaţie web, există mai multe tehnologii, care sa ofere posibilitatea de a comunica într-o reţea, sau în afara reţelei, în timp real. Aceste tehnologii se pot folosi atât separat, cât şi combinat, şi din cauza aceasta se pot folosi avantajele unor tehnologii fără a se lua in considerere dezavantajele oferite de celelalte tehnologii.

Una dintre aceste tehnologii este WebSockets. WebSockets este o tehnologie web care oferă comunicare bidirectionala prin protocolul TCP. Este folosita pentru comunicare client-server şi server-client, la aplicaţii web. Protocolul WebSocket a fost standardizat prima dată de IETF, după care a fost standardizat API-ul WebSocket de către W3C. [19]

Conexiunea este stabilită folosind porturile 80 şi 443 în mod implicit. WebSockets fiind bazat pe protocolul TCP, relaţia între protocolul WebSocket şi HTTP e de tip handshake, iar potrivit standarului RFC6455, protocolul WebSocket este alcatuit din 2 parti: partea de handshake şi transferul de date.[20]

Ca şi functionalitate, prima dată trebuie stabilită conexiunea între client şi server. Acest lucru se face prin trimiterea mesajelor de handshake între client şi server. Dacă handshakeul dintre client şi server s-a stabilit cu succes, atunci atât clientul, cât şi serverul, pot să trimită date sau mesaje, deoarece conexiunea este bidirectionala. După ce s-au trimis datele şi mesajele, conexiunea între client şi server şi poate închide.

În urma acestor informaţii, s-a realizat o comparaţie între tehnologia WebRTC şi WebSockets, iar rezultatul se poate vedea în următorul tabel:

Tabel 3.1 WebRTC vs WebSockets

|  |  |
| --- | --- |
| WebRTC | WebScockets |
| Peer-to-Peer  Acesta este principalul avantaj folosind WebRTC, deoarece nu este nevoie de un webserver pentru transmiterea datelor | WebSockets are nevoie de un webserver centralizat pentru a trimite datele, mesajele sau imaginile video. |
| Comunicarea este bidirecta între client-client;  Are nevoie de un serviciu extern, numit signalling, prin care se transmit informaţii de reţea pentru stabilirea conexiunii.  Pentru implementarea unui serviciu de signalling se poate folosi WebSockets, sau diferite librarii care au la baza lor implementarea socketurilor (ex. Socket.io) | Comunicarea este bidirectionala între client-server-client, reducând astfel viteza de trimitere a datelor de la un client la altul |
| Suportul pentru WebRTC este oferit numai de ultimele versiuni ale majorităţii browserelor. Ex: Chrome, Firefox, Opera | Suportul pentru socketuri este oferit de toate browserele, fără excepţie. |
| Este browser-browser, iar “camerele” pentru  conversaţii sunt create în procesul de signaling | Foloseşte handshakeuri HTTP compatibile şi porturi default pentru combinarea mai uşoară a infrastructurii web existente şi a proxyurilor/firewall-urilor |
| Layerul de transport este configurabil | Un API de browser mai usor folosit şi implementat |
| Majoritatea aplicatiilor implementează funcţionalităţile WebRTC folosind protocolul UDP | Aplicaţiile care folosesc socketuri folosesc protocolul TCP |
| Folosind funcţionalitatea de DataChannel, se pot trimite date neprelucrate de la un client la altul, fără a folosi vreun server de comunicare | Clienţii trebuie să folosească neaparat ajutorul unui server pentru a putea comunica între ei. |
| Datele nu sunt transmise pe căi de “incredere” | Datele sunt transmise pe căi de “incredere” |
| Foloseşte servere STUN şi TURN pentru a obţine conexiuni cu client care au IP-uri private (clasele A,B,C) | Pentru comunicarea între clienţi nu sunt necesare să fie folosite servere de STUN şi TURN, deoarece serverul web decide unde vor ajunge datele de la un client la altul. |
| Pentru a se putea face broadcast (comunicare între mai mult de 2 clienţi în aceasi timp), soluţia este aceeasi la ambele tehnologii: este nevoie de un server centralizat. Dezavantajul este că datele mari ce trebuie transmise pot ajunge mai tarziu la client. | |

În urma acestei comparaţii realizate, s-a ajuns la decizia folosirii tehnologiei WebRTC, din cauza solutiei şi avantajelor oferite de aceasta. Avantajele principale ale aceste tehnologii sunt urmatoarele: comunicare peer-to-peer fără folosirea pluginurilor sau altor librarii externe, schimb de date text/audio/video fără ajutorul unui server intermediar, combinarea cu alte tehnologii pentru procesul de signalling.

## WebRTC API

WebRTC este tehnologia oferită de browserele web, prin care se pot interconecta 2 şi pot vorbi peer-to-peer prin mesaje de tip text sau audio/video, fără ca aceste informaţii sa treaca prîntr-un server de aplicaţii.

**Principiu de functionare**

În principiu, pentru conectarea a 2 utilizatori, trebuie facuti urmatorii paşi:

* Crearea obiectului de conexiune WebRTC.
* Trimiterea ofertei de la un utilizator la altul cu informaţiile SDP(Session Description Protocol), prin serviciul de semnalare.
* Trimiterea răspunsului de la al doilea utilizator cu informaţiile SDP, tot prin serviciul de semnalare.
* Căutarea şi adăugarea candidatiilor ICE şi trimiterea acestor candidaţi la celalalt utilizator prin serviciul de semnalare.
* Tratarea evenimentelor pentru deschiderea/închiderea conexiunilor

În această aplicaţie, acest principiu este transformat într-o implementare proprie, şi anume în scriptul webrtc.js.

În urmatoarele subcapitole vor fi prezentate elementele principale WebRTC, iar după vor urma integrarea acestor elemente în aplicaţia web.

WebRTC: Comunicarea în timp real între browsere, defineste un API de JavaScript, care poate fi folosită în aplicaţii web pentru a utiliza funcţionalităţile oferite de WebRTC pentru comunicare peer-to-peer.

Prin acest API se stabileşte o conexiune între două puncte finale, iar fiecarui punct final îi este asignat cate un rol: offerer şi answerer. Cel care porneste conexiunea şi negocierea de date este offerer, iar cel care răspunde la această negociere este answerer. Aplicaţia tratează aceste roluri, astfel încât fiecare punct final poate să înceapă conexiunea.

### Negocierea informatiilor de reţea

Negocierea datelor se face prin serviciul de semnalare. În acest caz serviciul de semnalare constă în folosirea bazei de date. Aplicaţia va scrie în baza de date informaţiile de conexiune oricând un utilizator vrea să înceapă o conexiune cu un alt utilizator. Este necesara implementarea proprie a serviciului de semnalare, deoarece API-ul WebRTC nu vine cu mecanismul de semnalare proprie. Informaţiile ce trebuie transmise prin serviciul de semnalare sunt acestea: SDP (Session Description Protocol), şi ICE candidaţi (Interactive Connectivity Establishment).

SDP-ul este un format ce descrie parametrii de iniţializare a streamingului de media. Acest format este destinat pentru descrierea sesiunilor de comunicare multimedia, pentru scopul de invitaţii de sesiune sau negocierea parametrilor. SDP-ul nu livrează instanţele media, ci este folosit pentru negocierea tipului de media, formatul şi proprietatile asociate acestora, între cele două puncte finale.[21]

Candidaţii ICE sunt transmişi cu informaţiile de IP prin NAT (Network Address Translator). ICE este folosit pentru crearea comunicării pentru VOIP, peer-to-peer, mesaje instanţe şi alte interactiuni media. Formatul este următorul:

a=candidate:1 1 **UDP** 2130706431 **192.168.1.102 1816** typ **host**

Aici UDP-ul ne arată ce fel de protocol este folosit, typ host specifică ce tip de ICE candidat este, iar *host* ne zice că candidaţii sunt generate înăuntrul firewall-ului. Un alt tip de candidat este tipul relay, care este folosit pentru comunicările în exteriorul firewall-ului.

Aceste informaţii se pot trimite în orice formă de la un peer la altul prin orice format şi protocol, atata timp cât fiecare peer intelege acel protocol.



Figura 3.6 Comunicare dintre offerer si answerer

Diagrama din figura 3.6 arată arhitectura, unde semnalarea între punctele finale WebRTC traversează serverul web, în timp ce conexiunea WebRTC se stabileşte între cele două puncte finale WebRTC.

Pentru a colecta candidaţii de ICE, punctele de conexiune trebuie configurate astfel încât sa aiba informaţii despre serverele STUN şi TURN, acesta în cazul în care aplicaţia este rulata în afara reţelei private. Momentan, nu există mecanism pentru descoperirea acestor servere, şi de aceea trebuie configurate în scripturile de JavaScript. Pentru folosirea serverelor, trebuie introduse într-un sir adresele acestora. Mai jos se poate vedea un exemplu cum se configureaza aceste servere de STUN şi TURN.

*var* ***serverConfig*** *= {*

*'iceServers': [*

*{'urls':'stun:stun1.l.google.com:19302'},*

*{'urls':'stun:stun2.l.google.com:19302'},*

*{'urls':'stun:stun3.l.google.com:19302'},*

*{'urls':'stun:stun4.l.google.com:19302'},*

*{'urls':'stun:stun01.sipphone.com'},*

*{'urls':'stun:stun.ekiga.net'},*

*{'urls':'stun:stun.fwdnet.net'},*

*{'urls':'stun:stun.ideasip.com'},*

*{'urls':'stun:stun.schlund.de'},*

*{'urls':'stun:stunserver.org'},*

*{'urls':'stun:stun.softjoys.com'},*

*{'urls':'stun:stun.voiparound.com'},*

*{'urls':'stun:stun.voipbuster.com'},*

*{'urls':'stun:stun.voipstunt.com'},*

*{'urls':'stun:stun.voxgratia.org'},*

*{'urls':'stun:stun.xten.com'},{'urls':'turn:numb.viagenie.ca:3478','credential':'<*

*password>','username':'<username>'}*

*]*

*}*

Funcţionarea acestor servere împreună cu candidaţii ICE este complexa, şi nu este vizibilă utilizatorului. Trebuie mentionat ca un candidat de tip host conţine adresa de IP local şi portul clientului ICE. Acesta reprezintă adresa hostului, chiar dacă nu se foloseşte ICE.

Un candidat de serverconţine o adresă de IP şi portul clientului ICE, vazut de serverul de STUN. Dacă clientul ICE este localizat după un NAT, acest candidat va conţine adresa publică ce a fost asignat de NAT, clientului ICE. Clientul ICE obţine candidatul prin trimiterea unei cereri folosind protocolul STUN pentru NAT. Serverul STUN detectează sursa adresei şi trimite inapoi la client. În caz că nu există NAT între clienţii de ICE şi serverul STUN, atunci adresa candidatului de server reflexiv va fi identic cu al candidatului de host.

Este important sa se ştie că conexiunea între clienţii ICE nu va traversa serverul STUN. Serverul există pentru tratarea requesturilor STUN şi pentru a oferi candidaţi de server,clienţilor ICE.

WebRTC-ul nu defineste un mecanism pentru găsirea serverelor STUN şi TURN de către clienţii ICE, de aceea trebuie configurate manual aceste servere.

În diagrama de secvenţă din figura 3.7, potrivit surselor[22], se poate observa interschimbarea informatiilor între cei doi utilizatori folosind servere STUN şi TURN, iar în figura 3.8 de la sursa [23] se poate observa crearea conexiunii dintre cei doi clienţi, împreună cu starea actuala a obiectului PeerConnection.



Figura 3.7 Conectare prin servere STUN si TURN



Figura 3.8 Crearea conexiunii dintre 2 clienti

### Stream API

Obiectul de MediaStream tratează streamurile media (audio şi video). Fiecare MediaStream are un obiect MediaStreamTrack care poate fi trimis la utilizatorul remote, prin obiectul PeerConnecion. Fiecare MediaStream poate să conţină cateva canale, cum ar fi de exemplu canalul video şi canalul stereo său sunet „5.1”. Cu un singur MediaStream se pot controla toate trackurile de media, inclusiv datele ce se pot trimite la utilizatorul remote.

Obiectul MediaStream poate fi creat apelând funcția getUserMedia(). În acest caz, se va lua de la utilizatorul local camera web şi microfonul ca şi date de input. Pentru afişarea acestui stream într-o pagina web, se poate face prin crearea unui URL prin funcția createObjectURL(MediaStream stream) prin care se pune ca şi sursa într-un tag de HTML, şi anume <video></video>.

### PeerConnection API

Obiectul PeerConnection reprezintă principalul obiect în implementarea WebRTC[14]. Obiectul PeerConnection permite comunicarea directa între browsere. Inainte sa se stabileasca o conexiune directa, obiectul PeerConnection are nevoie de un canal pentru tratarea negocierilor şi a semnalarii.

Prima parte a negocierii este schimbarea de adrese IP între cei doi utilizatori. Deoarece utilizatorii se află în spatele unui NAT (Network Address Translation) sau un firewall, există un mecanism deja implementat în WebRTC, prin care se poate află adresa reala a utilizatorilor. Când se creeaza obiectul PeerConnection, primul parametru este o adresă către un server STUN sau TURN. Al doilea paramtru este o funcție care este apelată când PeerConnection vrea să trimită ceva semnale de date pentru celalalt utilizator. Semnalele de date sunt procesate de către funcția processSignalinMessage(message) implementată în obiectul PeerConnection.

Când conexiunea de PeerConnection este stabilită, există posibilitatea de a adauga streamuri de media. Când un stream media este adăugat, obiectul PeerConnection va negocia codec-uri în loc de canale de semnalare. După ce negocierea este pornita, se trimit evenimente către cealalta parte a conexiunii, indicand ca streamul de media a fost adăugat, şi apoi se poate decide ce se va face cu acel stream media.

### Data Channel API

Canalul de date WebRTC este realizat folosind două streamuri SCTP unidirectionale, fiecare având aceasi valoare de ID de stream. Protocolul SCTP care oferă streamuri de SCTP, sunt încapsulate peste DTLS. Urmatoarea figura X ne arată structura de protocoale pentru realizarea canalurilor de date în WebRTC.



Figura 3.9 Structura protocoalelor

Canalul de date nu defineste un layer peste layerul de SCTP, dar cele două streamuri SCTP din canalul de date şi datele ce trebuie transportate prin canalul de date sunt adaugate la layerul de SCTP ca şi datele de user SCTP.

Folosind un set de extensii SCTP, se pot oferi diferite proprietati la canalul de date. Datele pot fi transmise într-o anumita ordine sau in mod aleator, depinzand de nevoile utilizatorului.

Mecanismul canalului de date nu limitează numarul de canale de date ce pot fi deschise simultan, iar numarul maxim de streamuri este definit de layerul de SCTP, care este 65535, iar fiecare limită este dependentă de date particulare şi de implementarile canalului de date ale punctelor finale ale aplicaţiei.

## Sisteme similare

Produsele alternative existente pe piaţă, sunt sisteme care se concentrează pe calitatea oferită pentru realizarea comunicării în timp real, dar există şi alte aplicaţii care folosesc tehnologia WebRTC pentru alte scopuri, nu numai pentru comunicare. Aplicaţiile se concentrează atât pe calitate, cât şi pentru experienta de utilizator. Aceste aplicaţii trebuie să fie atât usor folosibile, cât şi sa ofere calitate. În urmatoarele subcapitole vor fi prezentate unele aplicaţii care folosesc pentru comunicarea dintre utilizatori tehnologia WebRTC.

**Apizee**

Este o platforma Saas (Software as a Service) pentru comunicare web şi mobil, în timp real. Soluţia oferită de aceştia ajuta la discutiile de afaceri şi îmbunătăţeşte colaborarea intre clienţii digitali.

În spatele acestei aplicaţii este apirtc-ul, prin care aplicaţia realizează conectarea şi comunicarea între clienţi. Funcţionalităţile principale ale aplicaţiei sunt: schimb de mesaje de tip text, audio/video call, colaborare vizuala, multi-dispozitiv, 100% fără pluginuri, integrare şi conectare cu alte aplicaţii

Aplicaţia a castigat mai multe premii la diverse conferinţe cum ar fi “Stratégie Clients”, “Oscars of economy Côtes d’Armor” şi altele. Pentru mai multe informaţii se poate accesa linkul <https://apizee.com/>

**Client Bee**

Această aplicaţie are la baza tot tehnologia webRTC, dar este folosită pentru un total alt scop decât schimbul de mesaje sau conversaţiile prin web.

Această aplicaţie este folosită pentru: plăţi online, facturare automată, înregistrare video, video în-browser, reminder automat. Pentru mai multe informaţii se poate accesa linkul <https://clientbee.com/>

**eFace2Face**

Această aplicaţie oferă soluţii pentru vizualizari video bazate pe web şi soluţii de e-signing.

Îmbunătăţeşte productivitatea şi eficient cu converatii video, partajarea informatiilor şi semnaturilor electornice. Oferă posibilitatea sa te conectezi la mai multi clienţi în aceasi timp şi este usor de integrat în infrastructure deja existente online. Pentru mai multe informaţii se poate accesa linkul <https://eface2face.com/>

**FACEmeeting**

O aplicaţie simpla de “1 click” pentru întâlniri online cu conversaţii HD audio/video, schimb de mesaje de tip text, şi partajarea fişierelor. Pentru ca foloseşte webRTC, nu trebuie downloadat şi instalat niciun plugin, iar informaţiile media şi de semnalare între clienţi sunt criptate şi encodate.

Aplicaţia momentan funcţionează numai pe Google Chrome, dar este gratis, în comparaţie cu majoritatea aplicaţiilor. Pentru mai multe informaţii se poate accesa linkul <https://facemeeting.com/>

**Gearcloud Labs - Mixology**

Această companie dezvoltă o tehnologie care lasa oamenii şi utiliatorii sa împărtăşească experientele noi şi unice legat de grafică şi streaming de video. Ei o numesc tehnologia aceasta “Mixology”, şi poate fi folosită pe o varietate de aplicaţii ce oferă: distractie, jocuri, evenimente live şi multe altele.

Tehnologiile care se folosesc sunt webRTC pentru soluţiile de procesare video şi OpenCV pentru imaginile şi informaţiile grafice. Pentru mai multe informaţii se poate accesa linkul [www.gearcloudlabs.com](http://www.gearcloudlabs.com/)

**Net Medical Xpress – RTC Conference Switch**

Această companie foloseste webRTC-ul pentru dezvoltarea aplicaţiei lor interne numită “RTC Conference Switch”, pentru a lăsa utilizatorul sa-şi creeze propria organizaţie, grup, şi subgrup pentru conferinţe de video. Pentru mai multe informaţii se poate accesa linkul <http://www.nmxs.com/>

**Proplogic Software Inc - Tawk**

Această companie a creat aplicaţia de chat numit “Tawk”.

Funcţionalităţile principale ale aplicaţiei sunt urmatoarele: integrarea aplicaţiei în alte aplicaţii folosind doar un simplu fragment de cod, chat audio/video, împărtăşire de monitor, transfer de date securizat şi criptat, împărtăşirea textelor şi a fişierelor, opţiune fullscreen, camera de conversaţie pana la 4 utliziatori. Pentru mai multe informaţii se poate accesa linkul <https://tawk.com/>

După cum se observă fiecare aplicaţie are la baza tehnologia WebRTC, şi ca şi interfata de UI este folosit HTML5. Diferenţele între aceste aplicaţii sunt funcţionalităţile, şi serverele pe care rulează aplcatia web. Tehnologia WebRTC poate fi folosită în mai multe contexte, cum ar fi comunicare, împartașire a datelor, înregistrare video sau plăți online, și datorită faptului că oferă posibilități în mai multe domenii, există multe aplicații la baza cărora se află această tehnologie.

# Analiză și Fundamentare Teoretică

În acest capitol vor fi prezentate tehnologiile și uneltele folosite pentru implementarea aplicației web. Printre acestea se numără: WebRTC, PHP, JavaScript, HTML, CSS, XAMPP, Github, și așa mai departe.

## Tehnologii și unelte utilizate pentru dezvoltarea aplicației web

### PHP

PHP este un limbaj de programare de tip script, folosit pe partea de server a aplicațiilor web. Numele original provenit de la *Personal Home Page* a fost ulterior schimbat în *PHP: Hypertext Preprocessor*.

Codul PHP poate fi integrat cu codul HTML şi poate fi folosit în diferite combinații pentru diferite sisteme web. Codul este procesat de un interpretor ca un modul în partea de server sau ca un executabil CGI. Serverul web combină rezultatele codului PHP interpretat şi executat, incluzând imagini şi pagini web generate. Codul PHP poate fi executat și din CLI, şi poate fi folosit și pentru implementarea aplicațiilor grafice singulare.[15]

Sintaxa pentru folosirea codului PHP este *<?php…?>* și poate fi combinat cu HTML-ul în următorul mod.

*<body>*

*<?php echo ‘<p>Hello World</p>’;?>*

*</body>*

Deși se poate combina cu HTML-ul, nu exista nicio regulă să combini aceste două limbaje, iar dacă fișierul conține numai cod PHP, atunci este indicat sa omitem etichetele de închidere.

*<? ’Hello world’;*

Variabilele trebuie anotate cu caracterul “*$”,* iar tipul variabilei nu trebuie specificat. Expresiile trebuie să se termine cu caracterul “*;”*, altfel codul PHP nu poate fi interpretat corect și ne va da mesaje de eroare. PHP-ul este similar cu limbajele de programare C/C++/Java, avand atât sintaxe și cuvinte cheie similare cât și logica expresiilor *if* sau *for*.

De la versiuna PHP5, sunt introduse variabile *private* şi *protected*, *metode*, împreuna cu clase *abstracte şi finale*. A fost introdusă și o modalitate standard de a declara *constructori* şi *destructori*, similar limbajelor de programare C++.

Motivul folosirii PHP este că este rapid de configurat, şi usor de deployat pe server. Fiind un limbaj de programare OOP, se poate implementa patternurile de arhitectura şi design din celelalte limbaje de porgramare OOP, cum ar fi de exemplu Java.

### JavaScript

JavaScript este un limbaj de programare interpretabil, netipizat, dinamic și de nivel înalt.[1] A fost standardizat în specificația de limbaje de la ECMAScript.[2] Este una dintre tehnologiile de bază pentru producția aplicațiilor web. Marea majoritate a websiturilor folosesc această tehnologie și este întretinută de toate browserele moderne.[1] Prefixul Java sugerează incorect că are legatură cu limbajul de programare Java, iar sufixul Script sugerează ca nu este un limbaj de programare “real”.

JavaScript-ul rulează pe partea de client, de obicei pe browser, și datorită acestui lucru poate să raspundă rapid la interacțiunile utilizatorului, și chiar dacă are defectele lui de design şi arhitectură, este foarte puternic și popular, fiind întretinut de toate browserele.

Una din motivele pentru care JavaScript-ul este atât de popular, este implementarea execuțiilor de cod asincron, şi chiar dacă nu ajută la citirea codului, asigură că paginile web sunt rapide şi sunt executate în timp minim. Pentru executarea codului în mod asincron trebuie folosite funcții de callback. O funcție callback este o bucată de cod executabilă, care poate fi plasată ca şi argument la o altă funcție, care va fi executat la un moment dat. Executarea funcției poate fi imediată, care se numeste căllback sincron, sau poate fi întarziată, care se numeste căllback asincron.[3]

JavaScript-ul este folosit și în alte medii, ce nu au legatură cu aplicaţii web, cum ar fi documente PDF, widgeturi, jocuri de calculator şi aplicații mobile. Existp implementpri şi pe parte de server, nu numai client, care ruleazpă în medii cum ar fi Node.js.

Motivul folosirii acestui limbaj de script, este că prin acest limbaj putem ajunge la implementarea şi folosirea tehnologiei WebRTC, care este implementat în browserul web. Deoarece comunicarea cu browserul şi cu API-urile oferite de acesta se face prin JavaScript, acesta este cel mai usor mod de a folosi funcţionalităţile oferite de WebRTC API.

### HTML

HyperText Markup Language, sau prescurtat HTML, este limbajul de marcare standard pentru crearea paginilor web. *Pe lângă CSS și JS, HTML-ul este una dintre tehnologiile necesare pentru a creea pagini web, dar și pentru a creea interfețe de utilizare pentru aplicaţii mobile*.[6]

Elementele de HTML construiesc blocurile de HTML pentru paginile web. HTML permite folosirea atât a imaginilor, cât şi a formelor interactive, și asigură că documentele create să fie structurate. Elementele de HTML sunt folosite prin etichete cum are fi ***<img />*** sau ***<input />*** introducând astfel direct conţinut în pagina web. Există şi alte etichete cum ar fi ***<p>…</p>*** ce poate conţine informaţii de tip text şi care poate să conţină sub-elemente. Browsere nu afișeaza etichetele de HTML ci le folosesc ca să interpreteze conținutul paginii web.

HTML-ul poate încorpora scripturi cum ar fi JavaScript, care poate afecta comportamentul paginilor. Se mai poate combina şi cu CSS pentru a da formă şi culoare elementelor de HTML.

HTML5 este a 5-a şi, momentan, ultima versiune a limbajului de marcaj HTML. A fost publicată de World Wide Web Consortium ca să îmbunătățească aplicaţiile multimedia de ultima generație. HTML5 introduce un API pentru aplicații web complexe, îmbunătățind şi extinzând astfel funcționalitățile deja existente.

HTML5 introduce funcționalități noi precum folosirea conținutului de tip multimedia și elementelor grafice. Exemple de etichete multimedia și etichete grafice sunt urmatoarele: ***<video>, <audio>, <canvas>*.**

Motivul folosirii HTML şi HTML5 pentru interfata grafica este că HTML-ul este usor de implementat şi usor de combinat cu restul limbajelor de scripting, JavaScript şi CSS.

### CSS

CSS (Cascading Style Sheets) este un limbaj folosit pentru descrierea şi stilizarea unui document scris într-un limbaj de marcare. Acesta descrie cum sunt pozitionate şi stilizate elementele HTML pe un ecran. De asemenea, este folosit pentru a controla aspectul mai multor pagini web în acelasi timp. Acelasi fişier CSS poate fi folosit pentru mai multe pagini HTML. De exemplu, paginile de Înregistrare şi de Logare în aplicaţie pot folosi acelasi fişier de stilizare a elementelor HTML. Acest principiu a fost aplicat şi pentru stilizarea paginilor de înregistrare şi logare a sistemului de comunicaţie bazat pe tehnologii WebRTC. Desi cel mai des este folosit pentru setarea stilurilor vizuale ale paginilor web împreună cu HTML şi XHTML, aceasta poate fi folosit inclusiv cu XML sau SVG. Impreuna cu HTML şi JavaScript, CSS-ul este o tehnologie folosita de cele mai multe site-uri web pentru creearea paginilor , dar poate fi intalnita şi pentru stilizarea aplicatiilor mobile.

CSS a fost creat în primul rând pentru a permite separarea continutului şi prezentarea acestuia (aspect, colori, pozitionare). Un avantaj al acestuia ar fi accesibilitatea continutului, deoarece oferă mai multa flexibilitate şi control asupra caracteristicilor de prezentare. De asemenea, mai multe pagini HTML pot utiliza acelasi fişier CSS pentru a reduce complexitatea şi repetitia în ceea ce priveste stilurile intalnite în aplicaţie. Dacă paginile HTML necesita elemente diferite de stilizare, acestea pot fi declarate în fişiere .css care contin doar informaţiile relevante acelor pagini. De exemplu, paginile de Home şi de Înregistrare pot conţine declararea stilurilor în fişiere CSS diferite.

### GitHub

GitHub este o aplicaţie web folosită pentru salvarea și stocarea aplicațiilor prin servicii Git. Precum Git, GitHub oferă funcționalităti cum ar fi controlul versiunilor și gestionarea codului sursă, însă putem întâlni şi funcționalități noi. [4] Proiectele pot fi accesate și manipulate folosind linii de comandă standard. Spre deosebire de Git care este o aplicaţie folosită strict din linia de comandă, GitHub oferă atât o interfaţă grafică web, cât și o interfaţă grafica desktop. O altă funcționalitate întalnită ar fi integrarea cu aplicaţiile mobile.

Utilizatorii aplicațiilor GitHub au posibilitatea de a se înregistra pentru a discuta despre proiecte şi pentru a creea depozite. Utilizatorii neînregistrați au posibilitatea de a vizualiza şi de a salva proiectele, însă nu pot creea şi modifica proiectele existente. Alte funcționalități ar fi vizualizarea bug-urilor, ale funcționalităților existente, dar şi gestionarea noilor task-uri.

Motivul folosirii acestei aplicaţie împreună cu Git, care se foloseşte prin linii de comenzi, este faptul ca este nevoie de controlul sursei de cod şi de administrarea defectelor. Folosind funcţionalităţile oferite de Github, atât controlul cât şi accesul resurselor este realizat cu uşurinţă.

### Apache Server

Apache HTTP Server, numit Apache este cel mai folosit software de server web din lume. Initial bazat pe serverul NCSA HTTPd, dezvoltarea pentru Apache a inceput în anul 1995. Software-ul este disponibil pentru o gama larga de sisteme de operare pe langa Unix, inclusiv eComStation, Microsoft Windows, şi altele. Eliberat sub licenta Apache, Apache este un software gratuit şi open-source.

Pentru proiectul Apache HTTP Server a fost depus un efort de dezvoltare software care vizeaza creearea unui server HTTP bogat în caracteristici, disponibil în mod liber celor care doresc sa il utilizeze.

Apache a fost dezvoltat şi intretinut de către o comunitate deschisa de dezvoltatori. Proiectul este gestionat de un grup de voluntari situati în intreaga lume, folosind Internetul pentru a comunica, impartasi planuri, dar şi sa dezvolte serverul şi documentația acestuia. Acest proiect face parte din Apache Software Foundation. În plus, sute de utilizatori au contribuit cu idei, cod şi documentatie pentru intretinerea şi îmbunătăţirea proiectului.

Apache suporta o varietate larga de caracteristici, multe implementari care sa extinda funcţionalitatea de baza a proiectului initial. Dintre acestea ar putea fi suportul pentru autentificare oferit de partea de server. Dintre interfetele de limbaj comune care sunt sprijinite în acest sens fac parte Perl, Python, Tcl şi PHP.

Virtual hosting permite instalarea Apache pentru deservirea mai multor aplicaţii web diferite. O masina cu un singur server Apache poate deservi simultan "www.videocall.com", 'www.videocall.org'. O alta caracteristica ar fi suportul pentru autentificarea cu parola. Deoarece codul sursa este disponibil în mod gratuit, orice programator poate adapta server pentru nevoile proiectului.

### XAMPP

XAMPP este un pachet de server integrat în Apache, MySQL, PHP şi Perl. Totul este pre-configurat şi pregatit pentru rulare doar prin dezarhivarea sau instalarea acestuia. Este o distributie Apache care face ca dezvoltatorii sa aiba o munca simplificata în ceea ce priveste creearea unui web server local folosit pentru testare şi implementare.Tot ceea ce este necesar pentru configurarea unui server web este inclus într-un fişier extractabil: aplicaţia server (Apache), baza de date (MariaDB), şi limbajul de scripting PHP [7].

XAMPP are nevoie doar de un singur fişier zip, tar, 7z sau exe pentru a putea fi descarcat şi rulat. Configurari ale diferitelor componente care alcatuiesc web serverul pot fi necesare, dar nu intotdeauna.

Acesta vine împreună cu o serie de module, incluzand OpenSSL, phpMyAdmin, MediaWiki şi altele. Mai multe instanţe ale XAMPP pot exista pe o singura masina, iar orice instanţa poate fi copiata de pe un calculator pe altul. XAMPP poate fi folosit atât în versiunea standard (cea mică) cât şi în versiunea completa.

Initial, designerii XAMPP-ului au avut intentia de a-l folosi numai ca un instrument de dezvoltare care permite dezoltatorilor şi programatorilor site-urilor sa isi testeze programele fără a fi nevoiti sa aiba acces la internet. Pentru a facilita munca acestora, multe dintre caracteristicile XAMPP-ului sunt dezactivate în mod implicit. De asemenea, XAMPP are capacitatea de a deservi pagini web pe World Wide Web [8]. O caracteristica speciala ar fi protejarea anumitor parti ale pachetului cu o parola. Odată ce XAMPP este instalat, este posibil sa tratam localhost-ul ca o gazda, prin conectarea folosind un client FTP.

Motivele pentru folosirii acestui program este deoarece vine împreună cu serverul de MySQL, şi de Apache. Deoarece aplicaţie de server este scrisa de PHP, este nevoie de serverul de aplicaţii Apache pentru a rula aplicaţia web. Programul XAMPP mai vine şi cu un tool de administrare a bazei de date numit phpmyadmin, prin care se poate gestiona cu uşurinţă datele din baza de date.

## Cerințele sistemului

Cerințele unui sistem pot fi cerințe funcționale și cerințe non-funcționale. Cerințele funcționale prezintă o descriere completă a funcționalităților, iar cerințele non-funcționale sunt reguli și constrângeri ale sistemului.

### Cerinte functionale

Cerințele funcționale sunt funcționalități ce poate sa ofere sistemul. Aceste funcționalități sunt urmatoarele: login, logout, register, integrare cu rețele de socializare, adăugare prieteni, chat în formă de text, chat audio/video, trimiterea fișierelor între utilizatori, salvarea conversațiilor.

**Login/Logout/Register**

Utilizatorul va putea sa se înregistreze cu un mail valid și să-și creeze un nume de utilizator unic. După toate acestea utilizatorul va putea să intre în aplicație introducând numele de utilizator și parola setată, și odată intrat in aplicație, va putea ieși din aplicație apasând butonul de logout.

**Integrare cu rețele de socializare**

Utilizatorul isi va putea sincroniza contul de chat cu contul de Facebook, dupa care va putea sa intre aplicatie folosind acel cont de socializare.

**Adăugare prieteni**

Utilizatorul va putea să-și adauge în lista de prieteni alți utilizatori, introducând numele de utilizator al prietenului.

**Chat in formă de text**

Utilizatorul va putea sa comunice cu un alt utilizator din lista de prieteni prin mesaje de text.

**Chat audio/video**

Utilizatorul va putea sa comunice cu un alt utilizator din lista de prieteni prin convorbiri audio și video.

**Trimiterea fișierelor între clienți**

Utilizatorul va putea sa trimită fișiere cu o mărime maxima setată la un alt utilizator din lista de prieteni.

**Salvarea conversațiilor**

Aplicatia va salva conversațiile de mesaje între utilizatori, ca aceștia să poată vedea conversțtiile mai vechi cu alți utilizatori.

### Cerinte functionale

Cerințele non-funcționale definesc calitatea sistemului, iar in următoarele paragrafe vor fi prezentate cerințele majore non-funcționale.

**Accesibilitate**

Aplicația web este ușor accesibilă dintr-un web browser. Utilizatorul trebuie să introducă URL-ul de baza a aplicației, după care va putea naviga între paginile aplicației. Linkurile pentru înregistrare și logare vor fi accesibile în pagina de bază a aplicației și utilizatorul va fi redirecționat întotdeauna către paginile dorite, fără a mai da clickuri suplimentare.

**Deployment**

Instalarea aplicatiei pe server este ușor realizată, prin copierea codului sursă pe serverul de aplicații. Pentru a vedea modificări pe partea de client, tot ce trebuie sa facă utlizatorul este să reîncarce pagina web, iar modificarile de server se vor putea vedea instant, fără reîncarcarea paginilor.

**Extindere**

Extindarea aplicației este usor realizabilă datorită structurii codului sursă, care contine două părți majore: partea de server și partea de client. Partea de server are o arhitectura MVC, care ușureaza adăugarea noilor pagini web. Logica din spatele paginilor web se poate schimba cu ușurință deoarece interfața de utilizator este decuplat de controllerul ei corespunzător.

**Performanță**

Performanța aplicației este ridicată, deoarece serverul este folosit pentru semnalarea și transmiterea informatiilor de conexiune de la un peer la altul. Transmiterea mesajelor de chat și a frame-urilor video se face rapid, deoarece utilizatorii sunt conectați direct între ei.

**Compatibilitate**

Aplicația este compatibilă atât pe windows, cât și pe linux sau alte sisteme de operare, deoarece este o aplicație web. Majoritatea browserelor majore, Opera, Chrome, Firefox, chiar și ultima versiune de Internet Explorer, au implementată tehnologia WebRTC, și din cauza aceasta aplicația web poate fi folosită pe oricare din browserele menționate mai sus.

**Securitate**

Pentru folosirea aplicației și a funcționalităților majore oferite de aplicație, serverul de aplicație trebuie sa fie securizat cu SSL. Astfel informațiile trimise de la server-client și client-server vor fi încriptate. Pentru a naviga în aplicație folosind *https,* serverul de aplicații trebuie să conțină certificatul de încredere împreună cu cheile de încriptare, iar browserul trebuie să accepte aceste certificate, și să adauge aplicația în lista de aplicații acceptate.

## Cazuri de utilizare

Cazurile de utilizare sunt cazurile ce sunt prezentate de cerințele funcționale.

Cazurile de utilizator se poate observa in figura de mai jos:

Use Case

Figura 4. 1 Cazuri de utilizare

**CU1. Titlu:** Login

**Descriere**: utilizatorul simplu acceseaza aplicatia și se autentifica

**Actor primar**: utilizator simplu

**Precondiţii**: utilizatorul are un cont care consta intr-un nume de utilizator și o parola

**Postconditii**: utilizatorul vizualizează cu success pagina principala

**Principiu**: utilizatorul selectează pagina de autentificare

**Scenariu de success**: Sistemul afişează pagina dorită

**Scenariu de eşec**:

a. utilizatorul introduce un nume de utilizator incorect

b. utilizatorul introduce o parola incorecta

**CU2. Titlu**: Login folosind Facebook

**Descriere**: utilizatorul simplu acceseaza aplicatia și se autentifica folosind Facebook

**Actor** **primar**: utilizator simplu

**Precondiţii**: utilizatorul are un cont pe reteaua de socializare Facebook care consta intr-un nume de utilizator și o parola

**Postconditii**: utilizatorul vizualizează cu success pagina principala

**Principiu**: utilizatorul selectează pagina de autentificare folosind Facebook

**Scenariu** **de** **success**: Sistemul afişează pagina dorită

**Scenariu** **de** **eşec**:

a. utilizatorul introduce un nume de utilizator incorect

b. utilizatorul introduce o parola incorecta

**CU3**. **Titlu**: Logout

**Descriere**: utilizatorul simplu acceseaza aplicatia și se deconecteaza

**Actor** **primar**: utilizator simplu

**Precondiţii**: utilizatorul este inregistrat in aplicatie cu un nume de utilizator și o parola

**Postconditii**: utilizatorul vizualizează pagina de start

**Principiu**: utilizatorul selectează optiunea de deconectare

**Scenariu** **de** **success**: Sistemul afişează pagina de start

**Scenariu** **de** **eşec**:

a. utilizatorul nu este redirectionat catre pagina de start a aplicatiei

b. numele de utilizator și parola utilizatorului nu sunt resetate

**CU4**. **Titlu**: Inregistrare

**Descriere**: utilizatorul simplu acceseaza aplicatia și se inregistreaza

**Actor** **primar**: utilizator simplu

**Precondiţii**: utilizatorul nu este inregistrat in aplicatie

**Postconditii**: utilizatorul vizualizează pagina de start

**Principiu**: utilizatorul selectează optiunea de inregistrare

**Scenariu** **de** **success**: sistemul creeaza un cont de utilizator și afişează pagina de start

**Scenariu** **de** **eşec**:

a. numele de utilizator exista in baza de date, iar contul nu este creat

b. utilizatorul nu este redirectionat catre pagina de start a aplicatiei

**CU5**. **Titlu**: Conversatie de tip text

**Descriere**: utilizatorul simplu initiaza o conversatie de tip text cu un alt utilizator

**Actor** **primar**: utilizator simplu

**Precondiţii**: utilizatorul este autentificat in aplicatie și conexiunea cu un alt utilizator s-a efectuat

**Postconditii**: utilizatorul trimite mesaje de tip text catre un alt utilizator

**Principiu**: utilizatorul initiaza o conversatie de tip text

**Scenariu** **de** **success**: sistemul trimite mesaje de tip text catre un alt utilizator

**Scenariu** **de** **eşec**:

a. conexiunea cu cel de-al doilea utilizator a esuat

b. mesajul de tip tip text nu este transmis

c. mesajul de tip tip text nu este transmis correct sau in totalitate

**CU6**. **Titlu**: Conversatie audio/video

**Descriere**: utilizatorul simplu initiaza o conversatie audio/video cu un alt utilizator

**Actor** **primar**: utilizator simplu

**Precondiţii**: utilizatorul este autentificat in aplicatie și conexiunea cu un alt utilizator s-a efectuat

**Postconditii**: utilizatorul converseaza prin camera web și microfon

**Principiu**: utilizatorul initiaza o conversatie audio/video

**Scenariu** **de** **success**: utilizatorii inregistrati in conversatia audio/video poarta o conversatie folosind microfonul și camera web

**Scenariu** **de** **eşec**:

a. conexiunea cu cel de-al doilea utilizator a esuat

b. imaginea utilizatorului folosind camera web nu este afisata

c. sunetul nu se aude

**CU7**. **Titlu**: Transfer de fisiere

**Descriere**: utilizatorul simplu transmite fisiere unui alt utilizator

**Actor** **primar**: utilizator simplu

**Precondiţii**: utilizatorul este autentificat in aplicatie și conexiunea cu un alt utilizator s-a efectuat

**Postconditii**: utilizatorul transfera fisiere cu succes

**Principiu**: utilizatorul transmite fisiere unui alt utilizator

**Scenariu** **de** **success**: utilizatorii conectati pot face schimb de fisiere

**Scenariu** **de** **eşec**:

a. conexiunea cu cel de-al doilea utilizator a esuat

b. fisierul nu este transmis

c. fisierul nu este transmis in totalitate

**CU8**. **Titlu**: Istoria conversatiilor de tip text

**Descriere**: utilizatorul simplu vizualizeaza istoria conversatiilor

**Actor** **primar**: utilizator simplu

**Precondiţii**: utilizatorul este autentificat in aplicatie

**Postconditii**: utilizatorul vizualizeaza istoria unei conversatii

**Principiu**: utilizatorul deshide și vizualizeaza istoria unei conversatii

**Scenariu** **de** **success**: utilizatorul poate vizualiza cu succes istoria conversatiilor de tip text

**Scenariu** **de** **eşec**:

a. conexiunea cu cel de-al doilea utilizator a esuat

b. istoria unei conversatii nu poate fi vizualizata

# Proiectare de Detaliu și Implementare

În acest capitol se va descrie proiectarea sistemului pentru componentele web, şi se va prezenta diagramele de arhitectura folosite, diagrame de secvente şi structura tabelelor din baza de date.

## Arhitectura de layer a sistemului

Arhitectura componentei web a sistemului respectă modelul arhitecturii pe nivele. Clientul interactioneaza cu nivelol de prezentare, care la rândul lui interactionea cu nivelul de aplicaţie şi asa mai departe. Această arhitectura este o arhitectura standard folosita pentru creearea aplicatiilor web datorita implementarii a acesteia. Arhitectura fiind pe nivele, se poate adauga şi scoate funcţionalităţi fără modificari majore. De exemplu se poate schimba nivelul de acces de date fără a schimba ceva în interfata utilizator.

În figura 5.1 se poate observa nivelele de arhitectura folosita în aplicaţia web.

C:\Users\Arctigor\Downloads\layer.png

Figura 5.1 Arhitectura de niveluri a aplicației

Cum era descris şi mai devreme, comunicarea este posibila numai între nivelele consecutive, creând un cuplaj slab între componente, astfel aducand avantajul de a aduce schimbari cu uşurinţă în componentele aplicaţiei.

Nivelele de arhitectura a aplicaţie sunt urmatoarele: *User Interface, Application, Business, Data Access, Database.*

### Nivelul interfata utilizator

Acest nivel conţine fişierele de .php folosite pentru afişarea paginilor web, fiecare continand diferite elemente de HTML. Fiecare pagina este definit prin fisierul de interfata utliziator şi controllerul lui corespunzător. Dacă se doreste introducerea unei noi pagini, atunci trebuie creat un fişier separate pentru acea pagina, şi un controller ce va conţine logica în spatele paginii respective.

Prin acest nivel se comunica cu tehnologia WebRTC, desigur, fără ca utilizatorul sa ştie sa sa vada acest lucru.

### Nivelul de aplicatie

Nivelul de aplicaţie conţine logica din spatele paginilor web. Acest nivel conţine funcțiile de controller pentru fiecare pagina. Configurarea cailor se face tot pe acest nivel. Prin cale se intelege URL-ul la care este asociat fisierul de interfata utilizator şi controllerul corespunzător. Dacă se doreste schimbarea unei parti de logica, atunci acesta se face cu uşurinţă prin schimbarea controllerului de la căi, şi adăugarea unui nou controller pentru noua logica.

Schimbarile mentionate de mai sus trebuie facute în urmatoarele fişiere: Routes.php pentru schimbarea sau adăugarea cailor folosite de aplicaţie, şi Controller.php pentru schimbari de logica sau aduagarea functionalitatilor noi.

### Nivelul de business logic

Acest nivel are rol de mediator, de a prelua comunicarea dintre controller şi accesul la date. Acest nivel preia responsabilitatea de la controller, deoarece controllerul ar fi supraincarcat şi astfel fiecare apelare de la controller la nivelul de acces de date se face prin acest nivel intermediar.

### Nivelul de acces de date

Acest nivel conţine toate interogarile ce se face către baza de date. Aceste interogari vin direct de la controller şi de la nivelul de business.

### Nivelul de baza de date

Acest nivel contine doar conexiunea la baza de date, pe care folosesc functiile din nivelul de acces de date.

## Arhitectura MVC

MVC este o arhitectura de software folosită pentru implementari de interfata utilizator. Separa software-ul în trei componente interconectate, ca sa separe reprezentarea interna a informatiilor de afişarea informatiilor către utilizator[16][17]. Mai demult era folosit pentru interfete grafice de utilizator, dar acest pattern s-a raspandit şi pe partea de design de software a aplicatiilor web.

MVC, după cum spune şi numele de Model-View-Controller, are trei componente principale, iar în figura 5.2 se pot observa aceste trei componente de Model, View şi Controller şi relaţiile lor între ele.



Figura 5. 2 Arhitectura MVC

Modelul gestioneaza direct datele, logica şi regurile aplicaţiei.

Viewul poate insemna orice reprezentare grafica a informatiei, cum ar fi diagrame, grafice sau tabele. Mai multe view-uri poate să conţină aceasi informatie, depinzand de nevoile utilizatorului.

Controllerul receptioneaza intrarile şi actiunile utilizatorului şi transmite mai departe către modelul aplicaţiei. Rezultatul venit de la model nu trebuie să treaca neaparat prin controller, ci poate să actualizeze direct componenta de view a aplicaţiei.

În această aplicaţie View-ul este reprezentat de paginile PHP. Fiecare pagina PHP are asignat cate o funcție din clasa de controller, care are exact aceasi nume ca şi numele paginii, şi care este configurat în clasa de Routes a aplicaţiei. Controllerul are legatura directa cu modelul de date a aplicaţiei, iar răspunsul de la model trece tot prin controller, ajungand la pagina, şi astfel actualizand pagina respectivă.

Această arhitectură se îmbină cu arhitectura de layeri a aplicației. Clasele de View ar fi echivalente claselor din User Interface, clasele de Controller ar fi echivalente claselor din Application si Business layer, iar clasele de Model ar fi echivalente claselor din Data Access și Database layer.

Un exemplu de funcţionare a acestei arhitecturi se poate observa în diagrama de secvenţă din figura 5.3.

C:\Users\Arctigor\Desktop\seq1_v2.png

Figura 5.3 Diagrama de secventa a unui exemplu de MVC

## Patternul Front Controller

Aplicaţia foloseşte un design pattern important şi necesar aplicatiilor web, şi anume patternul Front Controller.

Front Controller este un software design pattern utilizat pentru creearea aplicatiilor web. Patternul oferă un punct de intrare centralizat pentru tratarea request-urilor.[18]

Front Controllerul este adesea folosit în aplicaţii web pentru implementarea fluxului de lucru. Desi nu este necesar folosirea acestui pattern, este mult mai usor controlarea navigatiei unui set de pagini prîntr-un controller unic, decât sa lase responsabilitatea paginilor existente. Alternativa acestui pattern ar fi implementarea navigarii la fiecare pagina PHP, cu riscul de a duplica datele si obiectele comune pentru fiecare pagina.

In cazul acestei aplicatii, scriptul index.php are rolul de Front Controller. Acesta va instantia clasa de baza App dupa care este lasat pe seama acestei clase tratarea requesturilor.

In diagrama din figura 5.4 se poate observa in detaliu tratarea unui request.

C:\Users\Arctigor\Downloads\seq2.png

Figura 5.4 Diagrama de secventa a Front Controller-ului

După cum se observă toate requesturile de la user trec prin scriptul de index.php care va initializa clasa App şi care la urma va afisa pagina respectivă. Avantajul acestui pattern este că nu există riscul pentru duplicarea datelor şi duplicarea tratarea requesturilor, deoarece există doar un singur punct de intrare care are responsabilitatea de a trata toate requesturile.

## Componentele aplicatiei

Aplicația conține trei componente principale: *Client Browser*, *Application Server*, *Database Server*, iar în figura 5.5 se poate observa aceste componente.

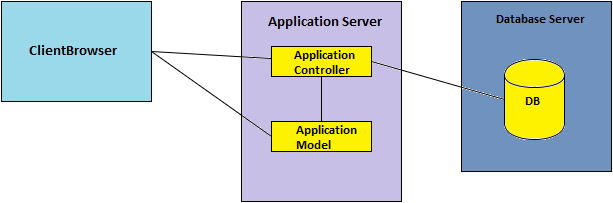


Figura 5.5 Arhitectura de componente

Utilizatorul are acces doar la partea de client, prin paginile oferite de aplicaţie. Pentru folosirea aplicaţiei paginile web oferă elemente de html prin care utilizatorul poate interactiona cu aplicaţia, şi în urma caruia se poate conecta la un alt utilizator.

Gestionarea şi interogarea bazei de date se face prin serverul de aplicaţie, iar rezultatul acestor interogari este responsabilitatea acestei componente. Dacă utilizatorul introduce date valide sau interactioneaza cu paginile web unde nu are permisiune, aplicaţia are responsabilitatea sa arunce mesaje de eroare respective. Conexiunea la baza de date se poate face numai prin controllerul aplicaţiei.

Figura 5.6 descrie mai în detaliu arhitectura acestei aplicaţii web, şi interactionarea concreta între componentele aplicaţiei.



Figura 5.6 Arhitectura detaliata a aplicatiei

După cum se poate observa această figura conţine cele 3 componente principale mentionate de mai sus, dar cu mai multe detalii.

### Componenta de server



Figura 5.7 Componenta de server

Această componenta conţine logica aplicaţiei şi redirecţionarea paginilor la interactionarea utilizatorului cu paginile web. Prin fisierul *App.php* se creeaza o instanţa a aplicaţiei, la fiecare schimbare de pagina. Obiectul care este instantiat este chiar clasa App care se regaseste în fisierul acesta. Acest obiect la urma lui instantieaza obiectul Controller care se regaseste în fisierul Controller.php. Obiectul App are şi o instanţa de Route unde sunt predefinite caile aplicaţiei împreună cu fisierul de interfata şi metoda de controller corespunzător. La accesarea unei pagini web, obiectul App caută în rutele predefinite numele paginii web pe care utilizatorul vrea să acceseze, iar în cazul în care nu se găseşte, se aruncă un mesaj de eroare.

Obiectul de *Controller* poate fi accesat doar prin obiectul de App, şi conţine toate funcțiile la care utilizatorul poate accesa în mod direct sau indirect. Acesare în mod direct inseamna ca dacă utilizatorul ajunge pe o pagina a aplicaţiei, atunci este direct apelată funcția corespunzatoare din controller, cum ar fi de exemplu login-ul. Acesarea în mod indirect este de exemplu când utilizatorul doreste sa se conecteze la un alt utilizator, iar transmiterea de informaţii se intampla automat, fără ca utilizatorul sa interactioneze mai mult cu interfata. De exemplu ar fi când utilizatorul cere conexiunea la un alt utilizator, iar butonul de acceptare sau refuzare apare pe pagina al celuilalt utilizator, fără ca acesta sa apese vreun buton.

Cererile de informaţii din baza de date se face tot prin acest controller. Exista funcții ce returneaza niste obiecte de tip array, în format de *JSON*, pentru acesarea mai uşoară a informatiilor. Acest obiect JSON la urma lui va fi returnat în funcția din JavaScript, de unde s-a pornit cererea de informaţii. Aceste informaţii sunt returnate din baza de date, fără ca controllerul sa faca modificari la ele.

Figura 5.8 ne arata diagrama de clasa a partii de server, unde se pot observa clasele si conexiunile intre ele.

C:\Users\Arctigor\Desktop\uml.png

Figura 5.8 Diagrama de clase

Baza de date este pe un alt server, decât serverul de aplicaţii, şi este nevoie de o interfata între cele servere pentru acesarea datelor din baza de date. Cu ajutorul PHP-ului conexiunea este realizată cu uşurinţă prin apelarea funcţiei mysqli\_connect("IP", "username", "password", "baza de date");. După este apelată funcția respectivă, se pot face interogari la baza de date. Dacă funcția nu este apelată, se va aruncă o excepţie, de care utilizatorul aplicaţiei nu ar trebui sa ştie, deoarece este responsabilitatea serverului şi nu a clientului sa gestioneze erorile de server şi de baza de date. Această funcție poate fi acesat în obiectul Database, prin funcția statica connect();

Mai jos se poate vedea cateva exemple de functii din clasa Controller.

**public function** logout(){

$this->unsetCookie('id');

$this->unsetCookie('username');

session\_destroy();

header("Location: /");

}

Functia *logout* nu va returna niciun obiect de tip JSON, dar in schimb va redirectiona utilizatorul la pagina principala a aplicatiei. Aceasta functie va distruge sesiunea cu informatiile de utilizator si va curata informatiile de cookie.

**public function** getFileName(){

$data = $\_POST;

$offererId = $data['myId'];

$answererId = $data['peerId'];

$connection = $this->getConnection();

$getOfferIdSql = "SELECT \* FROM `offers` WHERE (offererid="."'" . $offererId .

"' AND " . "answererid='".$answererId.

"') OR (offererid="."'" . $answererId .

"' AND " . "answererid='".$offererId.

"') AND status='complete' ORDER BY updatedtime DESC LIMIT 1";

$getOfferIdResult = $connection->query($getOfferIdSql);

$offer = $getOfferIdResult->fetch\_object();

**if**($offer){

$arrayFile = **array**();

$arrayFile["name"] = $offer->filename;

**return new** JsonResponse($arrayFile);

}

**return array**();

Functia *getFileName* in schimb va returna un Obiect de tip JSON la apelarea de ajax facuta de utilizator, daca s-a gasit informatia necesara din baza de data, iar in caz contrar va returna un array gol.

Partea de randare a paginilor web se face in clasa App, prin urmatoarea functie.

**private function** render() {

**if** (method\_exists($this->controller, $this->controller\_function)) {

$response = $this->controller->{$this>controller\_function}();

**if** ($response **instanceof** JsonResponse) {

header('Content-Type: application/json');

**echo** $response->jsonOutput;

}

**else** {

$templates = scandir(ROOT . '/templates');

**foreach** ($templates **as** $template) {

**if** ($template == $this->routes[$this->path]['template'] . '.php') {

$template\_name = $this->routes[$this->path]['template'];

$this->render\_page($template\_name, $response);

}

}

}

}**else** {

$this->urlNotFound();

}

}

Această funcție *render* nu face altceva decât urmatoarele lucruri: verifica dacă există metoda în clasa de controller, şi dacă există atunci va apela acea metoda. Dacă funcția din controller returneaza un obiect de tip JSON atunci nu incarca pagina de interfata, deoarece acest răspuns vine în urma unui apelare AJAX, şi nu vrem reincarcarea paginii în acest caz. Dacă răspunsul de la funcția controllerului nu este de tip JSON, atunci se va incarca pagina respectivă. În toate cazurile contrare utilizatorul va fi redirecţionat la o pagina de eroare, cu informaţia ca pagina accesata nu există.

### Componenta de client



Figura 5.9 Componenta de client

Componenta de client conţine diferite tipuri de fişier: de PHP, JavaScript şi CSS.

Fişierele de PHP pe care utilizatorul le poate accesa sunt urmatoarele*: index.php, header.php, home.php, login.php, loginWithFb.php, logout.php, register.php, history.php,* welcome.php. Restul fişierelor PHP sunt accesate indirect de către utilizator, în urma unor evenimente sau interactionare cu interfata utilizator ce porneste o reactie de apelare de funcții. Paginile de JS de care utilizatorul are nevoie pentru folosirea aplicaţiei sunt urmatoarele*: jquery-2.2.1.min.js, facebook.js, login\_with\_fb.js, webrtc.js.* Pentru a da formă sa culoare aplicaţiei web, sunt folosite fişiere CSS*: header.css, form.css şi welcome.css*.

In urmatoarea diagrama din figura 5.10 se poate observa cum sunt conectate paginile de PHP intre ele, si rutele pe care le poate accesa dintr-o pagina.

C:\Users\Arctigor\Desktop\php.png

Figura 5.10 Diagrama de pagini PHP

Pagina *header.php* conţine meniul cu urmatoarele linkuri: *Home*, *Login*, *Login* *with* *Fb* şi *Register*.

Acest header va fi intotdeauna prezent, cu mici modificari în meniu, depinzand pe ce pagina se află utilizatorul. De exemplu dacă îi pe pagina principala, atunci apar în meniu linkurile pentru logare şi inregsitrare, iar dacă utilizatorul este logat în aplicaţie, va aparea doar butonul de logout din acest meniu.

Pagina *home*.*php* nu conţine altceva decât titlul aplicaţiei şi meniul din header.

Pagina *login*.*php* conţine elementele de logare a utilizatorului şi anume: numele utilizatorului, parola şi butonul de login. Dacă utilizatorul introduce date gresite şi apasa pe butonul de login, atunci aplicaţia va aruncă un mesaj de eroare, ca userul sau parola îi gresita, după care utilizatorul va putea reintroduce datele. Dacă utilizatorul introduce corect datele, atunci este redirecţionat către pagina de welcome, unde va putea folosi funcţionalităţile principale oferite de aplicaţie. În urma loginului cu success se creeaza sesiunea cu informaţiile despre utilizator.

Pagina *loginWithFb*.*php* este goala deoarece foloseşte o interfata adusa de API-ul de Facebook. În cazul în care un cont de Facebook nu este asociat cu un cont deja inregistrat, atunci se va aruncă un mesaj de eroare. Dacă utilizatorul nu este logat pe Facebook, atunci i se va cere sa se logeze, iar dacă contul de facebook este deja asociat unui cont existent, atunci utilizatorul va fi redirecţionat pe pagina de welcome.

Această pagina utilizeaza scriptul de *facebook*.*js* şi *login\_with\_fb.js*, pentru a folosi funcţionalităţile oferite de API-ul de Facebook. La incarcarea acestei pagini scripturile vor fi rulate automat.

Pentru a folosi API-ul de Facebook, trebuie să creat o pagina de JS, care se numeste în aplicaţia aceasta *facebook*.*js*, şi trebuie introduse urmatoarele linii de cod:

window.fbAsyncInit = **function**() {

FB.init({

appId : '1740563549524938',

xfbml : **true**,

status : **true**,

cookie : **true**,

version : 'v2.6',

});

};

(**function**(d, s, id){

**var** js, fjs = d.getElementsByTagName(s)[0];

**if** (d.getElementById(id)) {**return**;}

js = d.createElement(s); js.id = id;

js.src = "//connect.facebook.net/en\_US/sdk.js";

fjs.parentNode.insertBefore(js, fjs);

}(document, 'script', 'facebook-jssdk'));

Dacă acest script este încărcat, aplicaţia poate folosi funcţionalităţile oferite de Facebook, cum ar fi accesarea numeui de utilizator, id-ul contului, poza de profil şi multe alte informaţii. Pentru folosirea acestor funcţionalităţi, trebuie creata mai intai o aplicaţie pe Facebook, care va genera un id, şi care trebuie să fie pus la apelarea funcţiei *FB*.*init* la atributul *appId*.

Pentru login se folosesc funcțiile *FB.getLoginStatus* pentru a verifica dacă este deja un utilizator logat pe Facebook, şi *FB.login* pentru logare pe Facebook. Fiecare funcție va returna un răspuns de tip JSON.

Pagina *logout*.*php* nu va conţine nimic, ci o să redirectioneze utilizatorul deja logat în aplicaţie, pe pagina principala. În urma logout-ului, aplicaţia va sterge datele de sesiune şi de cookie.

Pagina *register*.*php* conţine elementele de HTML pentru înregistrarea utilizatorului. Aceste elemente sunt urmatoarele: nume de utilizator, email valid, parola, confirmarea parolei şi butonul de register.

Pentru o înregistrare terminată cu success, utilizatorul trebuie să tina cont de urmatoarele lucruri:

* Toate campurile trebuie completate, fără excepţie
* Numele utilizatorului nu trebuie să existe deja în baza de date
* Numele utilizatorului nu trebuie să conţină caractere de tip space (“ ”).
* Parola şi confirmarea parolei trebuie să conţină aceasi caractere

Dacă utilizatorul respecta regurile de mai sus, atunci este redirecţionat pe pagina principala, cu un mesaj de success, în caz contrar, va ramane pe acelasi pagina, urmat de un mesaj de eroare coresponzator regurilor. După redirecţionarea utilizatorului, acesta se va putea loga în aplicaţie şi sa folosească funcţionalităţile principale oferite de aplicaţie.

Pagina *history*.*php* poate fi accesata numai dacă utilizatorul este logat deja în aplicaţie, altfel aplicaţia va aruncă un mesaj de eroare. Această pagina va conţine istoria conversatiei între 2 utilizatori, pe baza id-urilor acestora, şi poate fi accesata numai din pagina de welcome. Informaţiile de conversaţie vor fi afisate în formatul următor:

*[data] nume de utilizator: mesaj*

Pagina *welcome*.*php* conţine toate elementele principale ce poate folosi utilizatorul în aplicaţie: conectare la un alt utilizator, aduagarea unui prieten, afişarea istoriei de conversaţii, conectare la un cont de Facebook, chat text şi audio/video şi transmiterea fişierelor.

La incarcarea acestei pagini se incarca şi scripturile *webrtc*.*js* şi *jquery-2.2.1.min.js.* Folosirea librariei jquery este necesara, deoarece aplicaţia foloseşte ajax call-uri pentru a accesa funcțiile din server, mai exact din controller, fără ca pagina să fie reincarcata.

Scriptul *webrtc*.*js* are acces la toate funcţionalităţile oferite de WebRTC, deoarece, după cum se observă în diagrama de arhitectura de design a aplicaţie, scripturile de JS ruleaza pe browsere, iar în browser este implementat deja tehnologia WebRTC ce oferă funcţionalităţile pentru conexiune peer-to-peer. Acest script conţine şi tratarea evenimentelor de apasare de butoane, dar şi a evenimentelor trimise de WebRTC. Tratarea evenimentelor şi conexiunea propriu-zisa va fi prezentat în subcapitolul unde se va descrie în detaliu funcţionalităţile oferite de WebRTC şi cum sunt acestea integrate în aplicaţia web.

### Stabilirea conexiunii intre doi utilizatori

In urmatoarea diagrama din figura 5.11 se poate observa implementarea completa pentru crearea conexiunii intre doi utilizatori, folosind ca mecanism de semnalare scriere si citire din baza de date.



Figura 5. 11 Diagrama pentru conectare intre clienti

In subcapitolele urmatoare se va descrie in detaliu toate elementele din aceasta diagrama, ce au rol in crearea conexiunii intre doi clienti.

### Crearea ofertei

Crearea conexiunii se face tot în scriptul de JavaScript webrtc.js, împreună cu trimterea fişierelor şi a chatului audio/video.

Pentru crearea conexiunii trebuie initializate urmatoarele clase: RTCPeerConnection, RTCIceCandidate şi RTCSessionDescription. Aceste clase vor fi instantiate în următorul mod, depinzand de browserul în care este rulata aplicaţia.

RTCPeerConnection=window.mozRTCPeerConnection||window.webkitRTCPeerConnection;

RTCIceCandidate = window.mozRTCIceCandidate || window.RTCIceCandidate;

RTCSessionDescription = window.mozRTCSessionDescription

|| window.RTCSessionDescription;

La pornirea crearii conexiunii se creeaza o instanta de RTCPeerConnection folosind lista de serveri de STUN si de TURN, daca exista.

localPeerConnection = **new** RTCPeerConnection(servers);

Dupa creearea instantei de clase RTCPeerConnection urmeaza setarile pentru canalul de date si media locala. Pentru a seta streamul local, dupa cum era descris in capitolul de Stream API, se apeleaza metoda *getUserMedia*, impreuna cu restrictiile urmaatoare:

video : **true**

audio : **true**

Dupa apelarea acestei metode se seteaza sursa elementului de HTML pentru video cu URL-ul creat cu ajutorul streamului:

localVideo.src = URL.createObjectURL(stream);

Dupa setarea streamului local se initializeaza canalul de date in felul urmator:

dataChannel = localPeerConnection.createDataChannel('myDataChannel');

dataChannel.onmessage = eventDCMessage;

dataChannel.onopen = eventDCOpen;

dataChannel.onclose = eventDCClosed;

dataChannel.onerror = eventDCError;

După cum se vede, pentru crearea canalului de date se apeleaza funcția implementată în obiectul de PeerConnection deja existent, şi anume createDataChannel. Se initializeaza şi tratarile de evenimente pentru fiecare tip de eveniment:

* onmessage: se trimite când un utilizator trimite date de tip text sau binar către celalalt utilizator
* onopen: se trimite când conexiunea este finalizata cu success între cei doi utilizatori
* onclose: se trimite când un utilizator nu mai este conectat la celalalt utilizator
* onerror: se trimite când se trimit date eronate prin canalul de date

Când se seteaza toate acestea mentionate mai sus, se apeleaza metoda de creare de ofertă, *createOffer*, după care se adauga streamul local de media la conexiune.

localPeerConnection.createOffer(createOffer, errorHandler);

localPeerConnection.addStream(localStream);

La apelarea acestei functii se seteaza conexiunea locala de PeerConnection cu SDP-ul userului in urmatorul fel,

localPeerConnection.setLocalDescription(sessionDescriptionProtocol);

după care se scrie în baza de date urmatoarele informaţii: id-ul utilizatorului care trimite cererea de ofertă, id-ul userului către care se trimite cererea de ofertă, tipul ofertei – offer, şi informaţia de SDP local, de care o să aiba nevoie celalalt utilizator pentru crearea conexiunii.

Dacă în baza de date există deja o cerere de ofertă între cei doi utilizatori, atunci nu se va scrie o noua cerere în baza de date, ci se va suprascrie cererea veche cu informaţiile noi de SDP.

Scrierea în baza de date se face cu apelari ajax folosind libraria jquery, în felul următor:

**function** insertDataToDb(offerJSON, url) {

$.ajax({

data : offerJSON,

type : "post",

url : url,

});

}

Parametrul offerJSON este un obiect JSON care conţine informaţiile necesare pentru crearea ofertei, iar parametrul url este călea către template-ul şi funcția de controller care se ocupa cu scrierea ofertei în baza de date.

### Asteptarea ofertei

Asteptarea unei oferte de la un utilizator se face periodic, acesta insemnand ca se citeste periodic din baza de date ofertele create de utilizatori către clientul respectiv. Dacă există o ofertă în baza de date, acesta este citita de utilizator, după care poate alege sa accepte sau sa refuze oferta. În cazul în care utilizatorul refuza oferta, conexiunea se va închide, iar în baza de date se va finaliza oferta cu tipul de ofertă decline.

Dacă utilizatorul accepta oferta, atunci se creeaza o alta instanţa de RTCPeerConnection pentru utilizatorul remote, şi se initializeaza, ca şi la utilizatorul local, streamurile de media şi canalul de date.

remotePeerConnection = **new** RTCPeerConnection(server);

createDataChannel();

setLocalMedia();

În urma setarii acestor elemente de WebRTC, se seteaza SDP-ul primit de la utilizatorul care a creat oferta, şi se apeleaza funcția *createAnswer* a obiectului remote de PeerConnection.

remotePeerConnection.setRemoteDescription(remoteSDP);

remotePeerConnection.createAnswer(**function**(sessionDescriptionProtocol) {

remotePeerConnection.setLocalDescription(sessionDescriptionProtocol);

}

După setarea acestor valori se va actualiza în baza de date cererea de ofertă cu informaţiile de SDP remote împreună cu tipul de ofertă answered.

### Asteptarea raspunsului

Asteptarea răspunsului se face similar cu asteptarea ofertei, insemnand ca se va citi periodic din baza de date, ofertă care are tipul answered. Dacă se va gasi astfel de ofertă, atunci creearea conexiunii va merge către pasul următor şi anume adunarea candidaţii de ICE. Dar inainte sa se adune acesti candidaţi, se va seta SDP-ul clientului remote la conexiunea locala de RTCPeerConnection, prin apelarea metodei setRemoteDescription. Când se seteaza SDP-ul remote, automat se va trimite un eveniment ce va caută candidaţii de ICE, evenimentul fiind numit onicecandidate.

După adunarea acestor candidaţi ICE, se va actualiza în baza de date cu candidaţii actuali şi cu tipul ofertei candidate. Actualizarea bazei de date se face similar cu celelalte actualizari de oferte.

### Asteptarea candidatilor

Asteptarea candidatiilor este ultimul pas in creerea conexiunii dintre cei doi utilizatori. La fel ca si la celelalte asteptari, se citeste periodic baza de date, si daca se gaseste o oferta care are tipul *candidate* atunci se adauga la instanta de RTCPeerConnection candidatii respectivi.

remotePeerConnection.addIceCandidate(iceCandidate);

După adăugarea candidatului ICE, userul trebuie doar sa actualizeze baza de date cu tipul ofertei *complete*, şi restul le lasa sa se ocupe WebRTC-ul, adica creearea de conexiuni şi transmiterea streamurilor media şi a deschiderii canalelor de date. La completarea conexiunii utilizatorii pot trimite mesaje de tip text, sau fişiere, dar pot să şi faca conversaţie audio/video.

După cum se observă, există multe aşteptări de oferte şi răspunsuri, iar toate acestea se fac prin apelarea functiilor ajax de la jquery. Un exemplu de aşteptare de răspuns ar fi următorul.

**function** getOfferFromDb() {

**var** myId = getMyId();

**var** responseJSON = **null**;

myDataJSON = {

myId : myId

};

$.ajax({

data : myDataJSON,

type : "post",

url : "/getOffer",

async : **false**,

success : **function**(response) {

responseJSON = response;

}

});

**return** responseJSON;

}

### Trimiterea datelor prin DataChannel

Pentru trimiterea mesajelor sau a fişierelor prin canalul de date, utilizatorii trebuie prima dată sa se conecteze. Atât pentru transmiterea de fişiere, cât şi transmiterea mesajelor de tip text, se foloseşte funcția *send*() din DataChannel. Un exemplu de trimitere de mesaj se face în felul următor:

**function** sendData() {

**var** data = dataChannelSend.value;

**var** username = getMyUsername();

**var** date = getCurrentTime();

dataChannelReceive.value += "[" + date + "] " + username+": "+ data+"\n";

dataChannel.send(encode(username) + data);

}

După cum se observă, funcția sendData() face urmatoarele lucruri:

* Se ia valoarea de tiptext din TextArea-ul în care a scris utilizatorul mesajul lui
* Se ia numele utilizatorului care trimite mesajul
* Se calculeaza timpul local
* Se afiseaza în TextArea-ul de mesaje primite data, username-ul şi mesajul
* Se trimite prin canalul de date mesajul împreună cu username-ul.
* Se trimite codat username-ul, deoarece nu vrem citiri suplimentare din baza de date, iar la primire de mesaj se decodeaza username-ul pentru afişarea mesajului.

Pentru trimiterea fişierelor se incarca un fişier de pe calculatorul utilizatorului, după care fisierul este trimis pe bucati la celalalt utilizator, byte cu byte. În timp ce utilizatorul incarca fisierul de pe calculatorul lui, numele fisierului va fi scris în baza de date pentru schimbul de informaţii.

### Primirea datelor de pe DataChannel

La primirea mesajelor de tip text se face prin tratarea eventului de onmessage. La primirea mesajelor de tip text se face decodarea mesajului pentru a află numele utilizatorului de la care a venit mesajul, şi se afisesaza în TextArea-ul utilizatorului, la fel când se afiseaza trimiterea mesajului. După afişarea mesajului se salveaza mesajul în baza de date, pentru a putea vizualiza în viitor conversatia între cei doi clienţi. Salvarea se face prîntr-o simpla scriere în baza de date a urmatoarelor informaţii: id-ul clientului local, id-ul clientului remote, mesajul primit, şi data în care a primit mesajul.

La primirea fişierelor se creeaza prima dată un buffer cu ajutorul caruia se creeaza URL-ul folosit pentru downloadarea fisierului.

**var** receiveBuffer = [];

receiveBuffer.push(event.data);

**var** received = **new** window.Blob(receiveBuffer);

receiveBuffer = [];

downloadAnchor.href = URL.createObjectURL(received);

**var** name = getFileName();

createDownloadLink(name);

După ce se creeaza URL-ul cu fisierul primit, se citeste din baza de date numele fisierului, şi se va creea linkul de descarcare a fisierului.

Deoarece mesajele şi fişierele sunt transmise în acelasi fel, la primire trebuie diferentiate cele două tipuri de mesaje. Acesta se face prîntr-un bloc de try/catch.

**try** {

**var** dataArray = decode(event.data);

...

} **catch** (err) {

**var** receiveBuffer = [];

...

}

După cum se observă, se verifica dacă poate să faca decodarea mesajului primit. Dacă acesta nu aruncă o excepţie, atunci inseamnca ca mesajul primit este de tip text. În caz contrar această excepţie este prinsa, şi este tratata ca mesajul fiind fişier.

Mesajele de la un utilizator la altul se pot transmite în acelasi timp, deoarece canalul de date este bidirectional.

### Vizualizarea istoricului

Pentru a vizualiza istoria conversatiilor, se citeste din baza de date liniile ce tin de cei doi utilizatori. Tot ce trebuie utilizatorul sa faca, este sa se logeze, altfel nu poate vizualiza istoricul. Vizualizarea se face prin deschiderea paginii history.php cu parametrul de id a clientului remote. Funcția care face acest lucru este urmatoarea

**function** viewHistory(friend) {

**var** peerId = friend.id;

window.open("/history/?peerId=" + peerId + "");

}

Funcția *window.open* este deja implementată în browser-ul clientului, iar în scriptul de JacaScript, tot ce trebuie realizat este să fie apelată funcția respectivă.

## Proiectarae bazei de date

In figura 5.12 se observa tabelele din baza de date si relatiile intre ele.

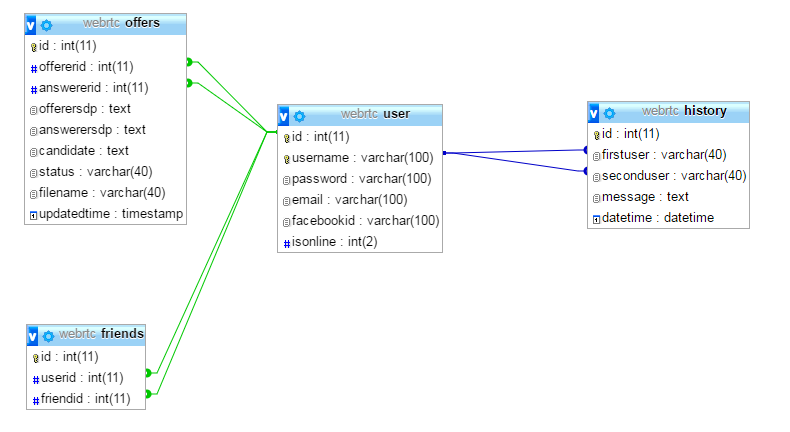


Figura 5.12 Baza de date

După cum se observă, tabelul user, are relatie m-n cu toate celelalte tabele, cheile primare fiind *id-ul* şi *username-ul*. Aceste chei sunt atât primare cât şi unice.

## Concluzie

Aplicaţia web implementează tehnologia WebRTC oferită de browserele web folosind JavaScript şi PHP, iar mecanismul de semnalare implementat este scrierea şi citirea din baza de date. Avantajul acestui mecanism de semnalare este că nu avem nevoie de un server separat pentru gestionarea negocierilor şi a transmiterilor de informaţii de reţea, ci doar de aplicaţia propriu-zisa, care are conexiune către baza de date.

Arhitectura acestei aplicaţii este bazat pe arhitectura de layer, iar fiecare layer are responsabilitatea lui. Această arhitectura este imbinat cu diferite patternuri de design cum ar fi MVC-ul sau Front Controller-ul, avantajele acestor patternuri fiind separarea responsabilitatilor şi centralizarea controlului de date.

Baza de date fiind MySql, se poate integra şi folosi cu uşurinţă în codul de PHP, astfel creând simplitate în codul aplicaţiei.

# Testare şi Validare

În acest capitol se prezintă metodele de testare a aplicației web. Deoarece dezvoltarea aplicaţiei s-a făcut în mod iterativ, iar la finalizarea functionalitatilor s-a făcut şi testare manuală a acestor funcţionalităţi. Testarea a fost începută cu scenarii simple de succes, după care a urmat testarea cu date invalide. Prin introducerea valorilor invalide, s-au găsit diferite probleme mai critice şi mai puţin critice, care s-au rezolvat cu succes.

Întregul proces de testare s-a făcut cu date valide care s-au citit şi s-au scris în baza de date. S-au făcut sesiuni de testare de tip smoke, după fiecare funcţionalitate majoră implementată. La fiecare sesiune de testare s-au şters datele din baza de date pentru a avea un mediu de dezvoltare curat. Aceste sesiuni de testare de tip smoke conţin paşii de testare a functionalitatilor majore a aplicaţiei.

Paşii de testare a functionalitatilor majore s-au făcut după modul următor.

**Înregistrarea utilizatorului**

Tabel 6.1 Inregistrarea utilizatorului

|  |  |
| --- | --- |
| **Acţiunea actuală** | **Rezultatul asteptat** |
| Utilizatorul va apăsa linkul de înregistrare din meniu şi va introduce în câmpurile de text următoarele informaţii: smoke, smoke@szabi.dev, Smoke1Smoke, Smoke1Smoke  După introducerea infromatiilor se apasă butonul de înregistrare | Utilizatorul cu nume smoke şi parolă Smoke1Smoke a fost creat, iar utilizatorul este redirecţionat la pagină de home. |
| Utilizatorul va apăsa linkul de înregistrare din meniu şi va introduce în câmpurile de text următoarele informaţii: smoke2, smoke2@szabi.dev, Smoke2Smoke, Smoke2Smoke  După introducerea infromatiilor se apasă butonul de înregistrare | Utilizatorul cu nume smoke2 şi parolă Smoke2Smoke a fost creat, iar utilizatorul este redirecţionat la pagină de home |
| Utilizatorul va apăsa linkul de înregistrare din meniu şi va introduce în câmpurile de text următoarele informaţii: smoke, [smoke@szabi.dev](mailto:smoke@szabi.dev), Smoke1Smoke, Smoke1Smoke  După introducerea infromatiilor se apasă butonul de înregistrare | Utilizatorul va primi un mesaj de eroare că numele de utilizator există deja în baza de date, şi nu va fi redirecţionat pe pagină de home. |
| Utilizatorul va apăsa linkul de înregistrare din meniu şi va introduce în câmpurile de text următoarele informaţii: smoke, [smoke\_mail](mailto:smoke@szabi.dev), Smoke1Smoke, Smoke1Smoke  După introducerea infromatiilor se apasă butonul de înregistrare | Utilizatorul va primi un mesaj de eroare că formatul de email nu este valid, şi nu va fi redirecţionat pe pagină de home. |
| Utilizatorul va apăsa linkul de înregistrare din meniu şi nu va completă toate câmpurile de text, după care va apasă butonul de înregistrare | Utilizatorul va primi un mesaj de eroare că să completeze toate câmpurile de texit, şi nu va fi redirecţionat pe pagină de home. |

* **Login şi Logout**

Tabel 6.2 Login si Logout

|  |  |
| --- | --- |
| **Acţiunea actuală** | **Rezultatul asteptat** |
| Utilizatorul va apăsa linkul de login din meniu şi va introduce în câmpurile de text următoarele informaţii: smoke1, Smoke1Smoke. După introducerea infromatiilor se apasă butonul de login. | Utilizatorul va primi primi un mesaj de eroare că numele de utilizator sau parolă este greşită şi nu va fi redirecţionat pe pagină de welcome. |
| Utilizatorul va apăsa linkul de login din meniu şi va introduce în câmpurile de text următoarele informaţii: smoke1, Smoke2Smoke. După introducerea infromatiilor se apasă butonul de login. | Utilizatorul va primi un mesaj de eroare că numele de utilizator există deja în baza de date, şi nu va fi redirecţionat pe pagină de home. |
| Utilizatorul va apăsa linkul de login din meniu şi va introduce în câmpurile de text următoarele informaţii: smoke1, Smoke1Smoke. După introducerea infromatiilor se apasă butonul de login. | Utilizatorul va fi redirecţionat pe pagina de welcome. |
| Utilizatorul va apăsa linkul de Logout din meniu. | Utilizatorul va fi redirecţionat pe pagina de home. |

* **Adăugare prieteni**

Tabel 6.3 Adaugare prieteni

|  |  |
| --- | --- |
| **Acţiunea actuală** | **Rezultatul asteptat** |
| Utilizatorul va deschide browserul Chrome şi se va loga in aplicatie cu urmatoarele credentiale: smoke, Smoke1Smoke | Utilizatorul smoke va fi redirecţionat pe pagina de welcome. |
| Utilizatorul va deschide browserul Chrome în mod privat şi se va loga în aplicaţie cu următoarele credentiale: smoke2, Smoke2Smoke | Utilizatorul smoke va fi redirecţionat pe pagina de welcome. |
| Utilizatorul smoke va introduce în câmpul pentru adăugarea prietenilor informaţia: utiliziator şi va apăsa butonul de add friend. | Utilizatorul va primi primi un mesaj de eroare că numele de utilizator nu există |
| Utilizatorul smoke va introduce în câmpul pentru adăugarea prietenilor informaţia: smoke2 şi va apăsa butonul de add friend. | Lista de prieteni a utilizatorului smoke și smoke 2 este actualizat cu noul prieten. |

* **Conectare cu prieteni**

Tabel 6.4 Conectare cu prieteni

|  |  |
| --- | --- |
| **Acţiunea actuală** | **Rezultatul asteptat** |
| Utilizatorul smoke va apăsa linkul de smoke2 din lista de prieteni | La utilizatorul smoke va apărea un nou tab cu numele de smoke2 |
| Utilizatorul va apăsa pe butonul de Start | După 5 secunde de așteptare la utilizatorul smoke2 va apărea cererea de conexiune. |
| Utilizatorul smoke2 va apăsa butonul de Accept | După aproximativ 15 secunde conexiunea se realizează iar în ferestrele de video vor apărea imaginile de la cele 2 surse de video |
| Utilizatorul smoke va trimite un mesaj cu urmatorul text: “Salut smoke2” | Utilizatorul smoke2 va primi mesajul “Salut smoke2”, iar fereasra de primire a mesajelor va fi actualizată cu informația următoare: [data curentă] smoke: Salut smoke2 |

Dacă s-a găsit o problema în aceşti paşi, atunci a devenit prioritar rezolvarea acestei probleme, după care s-a reluat sesiunea de testare smoke. Dacă toţi paşii de testare s-au realizat cu succes, s-a testat şi nouă funcţionalitate implementată.

Prin această metodă de testare, s-a ridicat nivelul de calitate a aplicaţiei şi s-a micşorat riscul de a avea probleme critice de funcţionare.

# Manual de Instalare și Utilizare

Acest capitol conține instrucțiunile de instalare si de utilizare a întregului sistem, și instrucțiunile de instalare a programelor ajutătoare.

## Instalare si rulare

Pentru instalarea aplicaţiei pe o reţea locala avem nevoie de urmatoarele aplicaţii instalate pe clientul local: Git, XAMPP şi un browser web (Chrome, Opera), iar sistemul de operare trebuie să fie Windows 7,8 sau 10.

Configurările necesare minime de hardware sunt următoarele: procesor Intel Core i3, placă video Nvidia GeForce 6100 nForce, 2GB RAM.

### Instalare Git

Pentru descarcarea programului utilizatorul trebuie sa intre pe pagina

<https://git-scm.com/download/win>

iar dupa descararea programului se porneste executabilul. Se urmeaza pasii de la wizardul de instalare dupa figurile urmatoare.

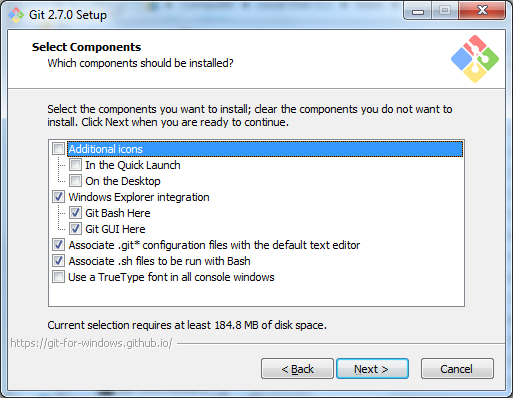


Figura 7.1 Selectarea functiilor Git

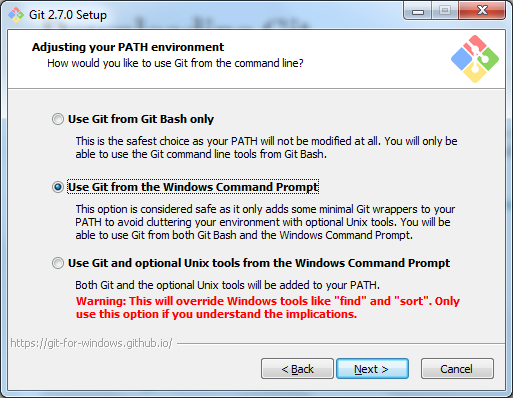


Figura 7.2 Selectarea functiei din cmd

Restul optiunilor poate sa ramana aceasi, iar utilizatorul poate sa dea click pe butonul de Next pana incepe instalarea aplicatiei.

Acest program este folosit atât pentru controlul sursei de cod a aplicației, cât și controlul versiunii a documnetației.

### Instalare XAMPP

Pentru instalarea programului XAMPP, utilizatorul trebuie să între pe pagina

<https://www.apachefriends.org/download.html>

şi sa descarce ultima versiune a programului. După ce se termina descarcarea se poate porni executabilul de instalare şi cu instalarea programului în sine.

Minima configuratie a programului trebuie să fie ca în figura X de mai jos, iar restul optiunilor poate fi lasat pe seama clientului.

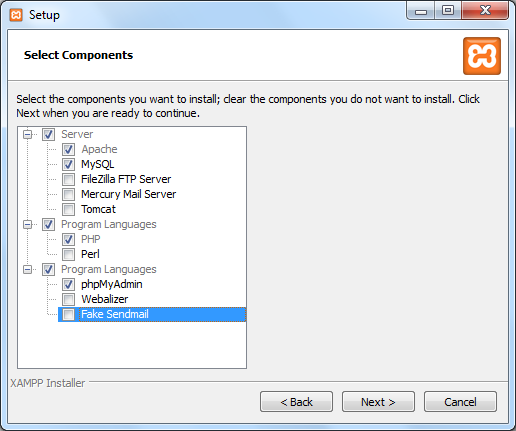


Figura 7.3 Selectarea functiilor XAMPP

Fiind programele Git şi XAMPP instalate, se poate descărca aplicaţia de pe siteul github. Utilizatorul trebuie să între din Command Line în folderul unde a instalat xampp-ul, pe calea urmatoare:

*<Installation folder of xampp>\htdocs*

Dupa ce a intrat in acest folder, trebuie rulat urmatoarea comanda.

[*https://github.com/Arctigor/webrtc.git*](https://github.com/Arctigor/webrtc.git)

După ce se ruleaza această comanda, aplicaţia va fi descarcata în folderul htdocs sub numele de webrtc. Următorul pas este sa între în folderul webrtc\server-conf şi se copieaza cele două fişiere în folderele de configuratii a serverului de apache, care se află pe urmatoarele căi.

*...\xampp\apache\conf*

*...\xampp\apache\conf\extra*

După aceste configurari, utilizatorul trebuie să modifice fisierul de hosts din C:\Windows\System32\drivers\etc\hosts cu urmatoarele informaţii.

*<IP local> szabi.dev*

Un exemplu ar fi 192.168.1.103 szabi.dev.

După aceste modificari se porneste Control Panelul de la XAMPP şi se porneste serverul de Apache şi MySQL, după cum se vede în figura 7.4.

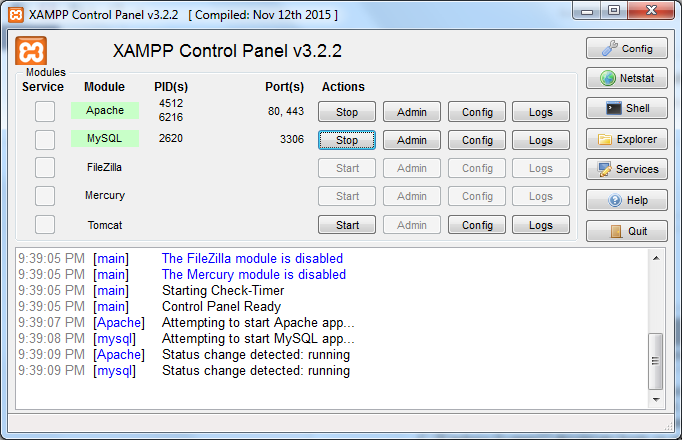
**

Figura 7.4 Pornirea serverelor

După ce se pornesc cu succes cele două module, următorul pas este creearea bazei de date cu informaţii de baza care vin în mod implicit cu aplicaţie. Clientul trebuie să navigheze pe linkul

<http://localhost/phpmyadmin/>

după care trebuie să creeze o noua schema prin a da click pe obiectul *New*, care se află în partea stanga a paginii. După apasarea butonuli New, apare o noua fereastra unde clientul trebuie să introduca numele de webrtc, şi sa creeze schema noua. După crearea schemei se, de click pe webrtc creat, şi se da click pe butonul de import aflat prin taburile paginii. După ce apare pagina urmatoare, clientul trebuie să aleaga fisierul webrtc de pe calea *...\xampp\htdocs\webrtc\mysql\webrtc.sql*, apoi trebuie să dea click pe butonul de OK. Baza de date va fi importat cu success, iar clientul poate să folosească aplicaţia cu succes, navigand pe pagina <https://szabi.dev/>

## Utilizarea aplicatiei

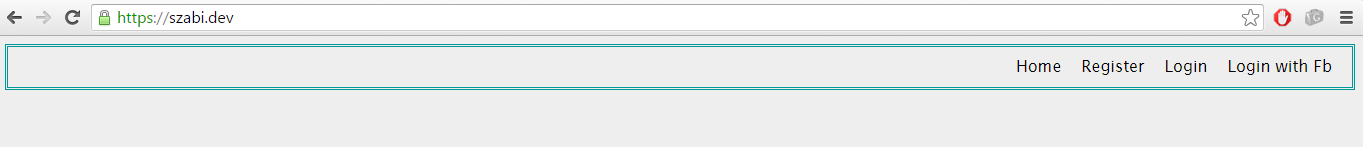
 La navigarea aplicaţiei la pagina principala, utilizatorul poate avea trei opţiuni, după cum se vede şi în figura X de mai jos: înregistrare, logare simpla, logare cu facebook.

Figura 7. 5 Meniul aplicatiei

La înregistrare utilizatorul trebuie să introduca datele valide, ca sa se poate inregistra, după regurile mentionate în capitolul 5. După ce clientul a creat cu succes un utilizator, poate să se logeze cu acel utilizator, după care va fi redirecţionat pe pagina de welcome, unde se poate conecta şi comunica cu alti utilizatori.

Pe pagina de welcome poate să adauge prieteni apasand pe butonul de Add friend, şi poate începe o conversaţie de tip text sau audio/video apasand pe unul dintre lista de prieteni şi apoi pe butonul de Start.

Dacă un alt utilizator cere sa se conecteze la client, atunci acesta va putea alege de a accepta sau a refuza acea conexiune apasand pe butoanele de Accept sau Decline. Dacă se accepta conexiunea, atunci trebuie să astepte pana ce aplicaţia va crea conexiunea între cei doi utilizatori.

Apasand pe butonul de Connect to Fb, utilizatorul poate să-şi conecteze propriul cont de facebook la contul creat. După ce utilizatorul se logeaza pe facebook, conexiunea dintre contul curent şi contul de facebook se face automat, după care utilizatorul poate să se logeze cu contul de Facebook din pagina principala, apasand butonul Login with Fb.

Dacă se creeaza o conexiune dintre cei doi utilizatori, se pot transfera între ei fişiere de orice tip, prin a da click pe butonul Send file, după ce se va alege fisierul dorit.

La iesirea din aplicaţie, utilizatorul poate să dea click pe butonul de Logout aflat în meniul aplicaţiei.

O conexiune cu success se poate vedea în figura 7.6 de mai jos.

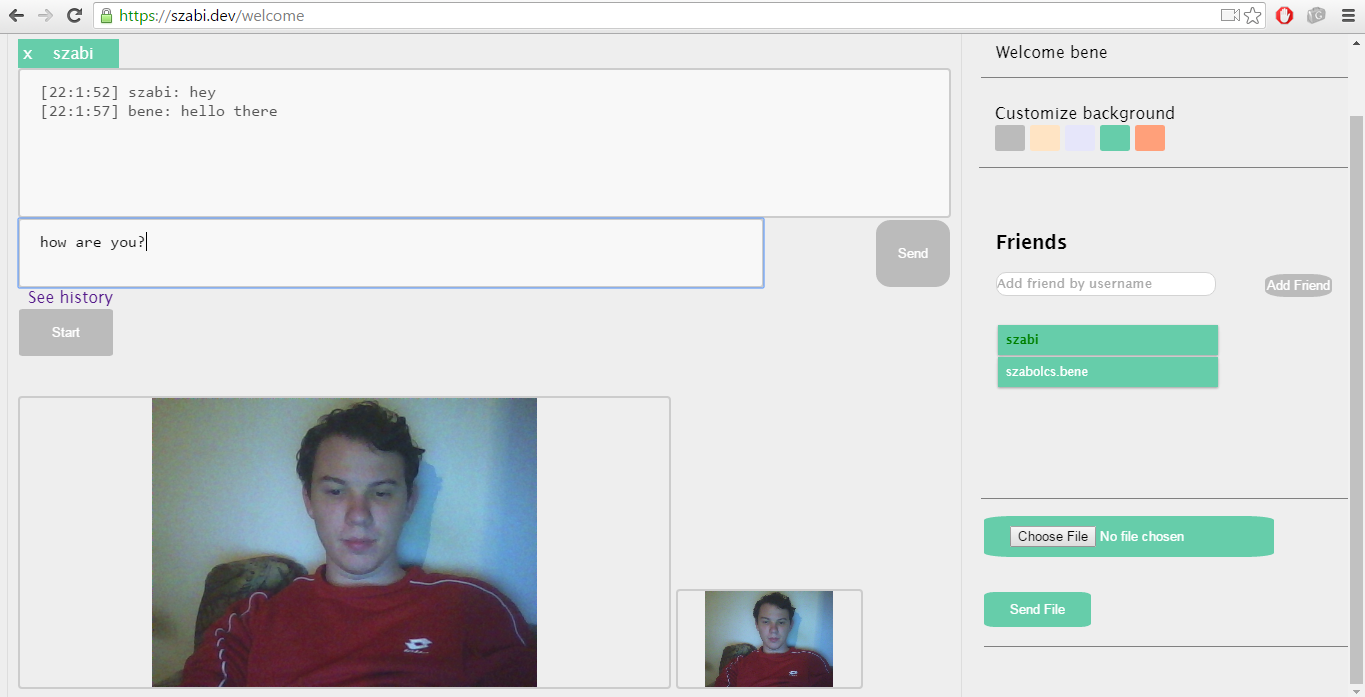


Figura 7.6 Conexiunie realizata cu succes

Pentru a vedea istoricul unei conversaţii, utilizatorul trebuie să dea click pe un prieten din lista de prieteni, şi apoi pe linkul de See history sub cele 2 TextArea-uri.

Utilizatorul se mai poate juca şi cu culorile backgroundului, şi poate să-şi aleaga fiecare ce culoare isi doreste din lista de culori prezentata în figura de mai sus.

# Concluzii

În acest capitol vor fi prezentate pe scurt obiectivele propuse, gradul de realizare și câteva idei pentru dezvoltări ulterioare.

## Realizarea obiectivelor propuse

Sistemul care s-a realizat reuseste sa isi atinga scopul de a putea fi un sistem competitiv cu sistemele similare disponibile.

Obiectivele secundare propuse s-au realizat în totalitate, utilizatorii se pot autentifica în aplicaţie prin nume de utilizator şi parola sau cont de Facebook, şi pot să se conecteze la alti utilizator pentru a realiza convorbiri în mesaje de tip text sau audio/video în timp real. Aplicaţia nu are nevoie de alte pluginuri sau programe externe pentru a realiza conexiunile cu alti utilizatori. Un utilizatori poate adauga mai multi prieteni în lista de prieteni şi poate să se conecteze la mai multi utilizatori în aceasi timp. Utilizatorul mai poate accesa şi conversaţiile anterioare pe care le-a avut cu alti utilizatori, şi poate să trimită fişiere către utilizatorii deja conectati, în timp real.

Componenta de client se ocupa cu afişarea interfetei utilizator şi cu tratarea evenimentelor pornite de către utilizator, şi împreună cu componenta de server poate să realizeze conexiunea dintre utilizatori.

Componenta de server se ocupa cu scrierea şi citirea de informaţii din baza de date, dar şi cu controlul paginilor şi a requesturilor utilizatorului. Conexiunea dintre componenta de client şi componenta de server se face prin requesturi sau apelari de funcții AJAX.

Sistemul reuseste sa livreze un mediu usor de inteles şi de utilizat, lucru sustinut şi de datele obţinute în urma studiului de evaluare. Utilizabilitatea şi lipsa nevoii de o perioada lunga de acomodare cu sistemul este sustinuta tot de rezultatele obţinute în urma efectuarii studiului.

## Dezvoltari ulterioare

În general orice sistem informatic poate să beneficieze de dezvoltări ulterioare și de extinderea funcționalităților.

O dezvoltare ulterioară poate fi creearea conferinţelor şi adăugarea a mai multor prieteni într-o singură conversaţie. Această funcţionalitate oferă posibilitatea de comunicare simultană de tip text sau audio/video, între clienţi, prin sincronizarea canalelor de comunicație.

O altă funcţionalitate poate fi transmiterea a mai multor fişiere în aceaşi timp, prin selectarea acestor fişiere de pe calculatorul personal al clientului. Avantajul acestei funcţionalităţi este că utilizatorul nu trebuie să selecteze şi să trimită separat aceste fişiere. Pe partea de comunicare de tip text, se pot introduce mai multe funcţionalităţi cum ar fi folosirea tipului Rich Text Format şi a smileyurilor. Acesta va oferi posibilitatea clienţilor de a schimbă formatul şi culoarea textului trimis la ceilalţi clienţi. Pe partea de comunicare de tip audio şi video, se poate introduce funcţionalitatea de a modifică volumul sunetului sau de a opri sunetul de tot. Aceste dezvoltări pot să ofere şi schimbarea calităţii videoului transmis, dar şi schimbarea dimensiunea ferestrelor de video.

Partea de interfaţă grafică se poate îmbunătăţi prin alegerea a mai multer teme de interfaţă şi salvarea acestora.

Cu adăugarea informaţiei de email la înregistrarea utilizatorilor, poate să vină şi confirmarea înregistrării, prin trimiterea unui email de confirmare. Dacă utilizatorul nu confirmă înregistrarea, atunci nu va putea să folosească contul creat. Cu ajutorul informaţiei de email, aplicaţia poate să ofere utilizatorului să-şi schimbe sau să-şi recupereze parola.

# Bibliografie

[1] Flanagan,David, "JavaScript:The DefinitiveGuide (5th ed.).  [O'Reilly & Associates](https://en.wikipedia.org/wiki/O%27Reilly_Media). " [ISBN](https://en.wikipedia.org/wiki/International_Standard_Book_Number) [978-0-596-80552-4](https://en.wikipedia.org/wiki/Special:BookSources/978-0-596-80552-4), 2011.

[2] Flanagan,David, "JavaScript:The DefinitiveGuide (6th ed.).  [O'Reilly & Associates](https://en.wikipedia.org/wiki/O%27Reilly_Media). " [ISBN](https://en.wikipedia.org/wiki/International_Standard_Book_Number) [978-0-596-80552-4](https://en.wikipedia.org/wiki/Special:BookSources/978-0-596-80552-4), 2011.

[3] [James Coplien](https://en.wikipedia.org/wiki/James_Coplien" \o "James Coplien), ["Perl Cookbook - 11.4. Taking References to Functions"](http://www.unix.org.ua/orelly/perl/cookbook/ch11_05.htm), 2010

[4] Williams Alex, ["GitHub Pours Energies into Enterprise – Raises $100 Million From Power VC Andreessen Horowitz"](http://techcrunch.com/2012/07/09/github-pours-energies-into-enterprise-raises-100-million-from-power-vc-andreesen-horowitz/), 2012

[5] Perkins, C. ; Westerlund, M. ; Ott, J.: Web Real-Time Communication (WebRTC): Media Transport and Use of RTP / IETF Secretariat. Version:Marz, 2012. <http://tools.ietf.org/html/draft-ietf-rtcweb-rtp-usage-02>

[6] John Crup,  ["World Wide Web Consortium: Conformance - requirements and recommendations"](http://www.w3.org/TR/REC-html40-971218/conform.html" \l "deprecated), 2007.

[7]  Udemy, H,["Downloading and installing XAMPP"](https://blog.udemy.com/xampp-tutorial/),2014

[8] Owusu-Ansah Godfred, ["Where XAMPP is used commonly and what it can do."](https://globalhealthdatamanagement.tghn.org/community/blogs/post/5122/2013/07/xampp/), 2012.

[9] Williams Alex, "[Charter of the Real-Time Communication in WEB-browsers working group](http://datatracker.ietf.org/wg/rtcweb/charter/)", 2013

[10] [Alex G, "WebRTC source code from Harald Alvestrand", 2011](http://lists.w3.org/Archives/Public/public-webrtc/2011May/0022.html).

[11] Whatwg.org, ["Introduction — HTML Standard"](http://www.whatwg.org/specs/web-apps/current-work/multipage/introduction.html#history-1), 2012

[12] ["WebRTC 1.0: Real-time Communication Between Browsers"](http://www.w3.org/TR/webrtc/), 2012

[13] Ericsson Labs, "Beyond HTML5: Peer-to-Peer Conversational Video", 2012

[14] W3C: PeerConnection, "Working Draft. http://dev.w3.org/2011/webrtc/

editor/webrtc-20111004.html#peerconnection", 2011

[15] PHP Manual, ["Introduction: What can PHP do?"](https://php.net/manual/en/intro-whatcando.php), 2009

[16] Moore, Dana et al. Professional Rich Internet Applications: Ajax and Beyond: "Since the origin of MVC", 2007

[17] Burbeck, Steve, "[Applications Programming in Smalltalk-80:How to use Model–View–Controller (MVC)](http://st-www.cs.illinois.edu/users/smarch/st-docs/mvc.html) ", 1992

[18] Alur, Deepak, John Crup, Dan Malks, Core J2EE Patterns, "Best Practices and Design Strategies, 2nd Ed. Sun Microsystems Press", 2003, pp. 650pp

[19] Ian Fette, Alexey Melnikov,  ["Relationship to TCP and HTTP"](https://tools.ietf.org/html/rfc6455#section-1.7), 2011

[20] Peter, Lubbers, ["How Web Sockets Interact With Proxy Servers"](http://www.infoq.com/articles/Web-Sockets-Proxy-Servers), 2010

[21] [Trygve Reenskaug](https://en.wikipedia.org/wiki/Trygve_Reenskaug), [James Coplien](https://en.wikipedia.org/wiki/James_Coplien), "More deeply, the framework exists to separate the representation of information from user interaction.", 2009

[22] WebRTC API –

<https://developer.mozilla.org/en-US/docs/Web/API/WebRTC_API/Connectivity>

[23] WebRTC Connection - http://www.slideshare.net/cryingnavi/webrtc-20150528