**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:** *Scurtă descriere a temei lucrării de licenţă și datele inițiale*
2. **Conţinutul lucrării:** *(enumerarea părţilor componente) Exemplu: Pagina de prezentare, aprecierile coordonatorului de lucrare, titlul capitolului 1, titlul capitolului 2,… titlul capitolului n, bibliografie, anexe.*
3. **Locul documentării**: *Exemplu*: Universitatea Tehnică din Cluj-Napoca, Departamentul Calculatoare
4. **Consultanţi**:
5. **Data emiterii temei:** 1 noiembrie 2013
6. **Data predării:** 28 Iunie 2014 (*se va completa data predării*)

|  |  |
| --- | --- |
| Absolvent: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  |  |
| Coordonator ştiinţific: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

**Declaraţie pe proprie răspundere privind**

**autenticitatea lucrării de licenţă**

Subsemnatul(a)**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**, legitimat(ă) cu \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ seria \_\_\_\_\_\_\_ nr. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_   
CNP \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_, 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  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  | Nume, Prenume  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  |  |  |
|  |  | Semnătura |

**De citit înainte** (această pagină se va elimina din versiunea finală):

1. Cele trei pagini anterioare (foaie de capăt, foaie sumar, declaraţie) se vor lista pe foi separate (nu faţă-verso), fiind incluse în lucrarea listată. Foaia de sumar (a doua) necesită semnătura absolventului, respectiv a coordonatorului. Pe declaraţie se trece data când se predă lucrarea la secretarii de comisie.
2. Pe foaia de capăt, se va trece corect titulatura cadrului didactic îndrumător (consultaţi pagina de unde aţi descărcat acest document pentru lista cadrelor didactice cu titulaturile lor).
3. Documentul curent a fost creat în **MS Office 2007.** Dacă folosiţi alte versiuni e posibil sa fie mici diferenţe de formatare, care se corectează (textul conţine descrieri privind fonturi, dimensiuni etc.).
4. **Cuprinsul** începe pe pagina nouă, impară (dacă se face listare faţă-verso), prima pagina din capitolul **Introducere** tot aşa, fiind numerotată cu 1. Pentru actualizarea cuprinsului, click dreapta pe cuprins (zona cuprinsului va apare cu gri), Update field->Update entire table.
5. Vizualizaţi (recomandabil și în timpul editării) acest document după ce activaţi vizualizarea simbolurilor ascunse de formatare (apăsaţi simbolul **π** din *Home/Paragraph*).
6. Fiecare capitol începe pe pagină nouă, datorită simbolului ascuns Section Break (Next Page) care este deja introdus la capitolul precedent. Dacă ştergeţi din greşeală simbolul, se reintroduce (*Page Layout -> Breaks*).
7. Folosiţi stilurile predefinite (Headings, Figura, Tabel, Normal, etc.)
8. Marginile la pagini nu se modifică (Office 2003 default).
9. Respectaţi restul instrucţiunilor din fiecare capitol.

Cuprins

[Capitolul 1. Introducere – Contextul proiectului (Heading 1 style) 10](#_Toc453704189)

[1.1. Contextul proiectului (Heading 2 style) 10](#_Toc453704190)

[1.1.1. (Heading 3 style) 11](#_Toc453704191)

[Capitolul 2. Obiectivele Proiectului 12](#_Toc453704192)

[Capitolul 3. Studiu Bibliografic 13](#_Toc453704193)

[3.1. Dezvoltarea aplicatiilor web 13](#_Toc453704194)

[3.2. Comunicare in timp real 13](#_Toc453704195)

[3.3. WebRTC 15](#_Toc453704196)

[3.3.1. Arhitectura WebRTC 15](#_Toc453704197)

[3.3.2. Media Path 17](#_Toc453704198)

[3.4. WebRTC vs WebSockets 18](#_Toc453704199)

[3.5. Sisteme similare 19](#_Toc453704200)

[3.5.1. Apizee 20](#_Toc453704201)

[3.5.2. Client Bee 20](#_Toc453704202)

[3.5.3. eFace2Face 20](#_Toc453704203)

[3.5.4. FACEmeeting 20](#_Toc453704204)

[3.5.5. Gearcloud Labs - Mixology 21](#_Toc453704205)

[3.5.6. Net Medical Xpress – RTC Conference Switch 21](#_Toc453704206)

[3.5.7. Proplogic Software Inc - Tawk 21](#_Toc453704207)

[Capitolul 4. Analiză și Fundamentare Teoretică 23](#_Toc453704208)

[4.1. Tehnologii și unelte utilizate pentru dezvoltarea aplicației web 23](#_Toc453704209)

[4.1.1. PHP 23](#_Toc453704210)

[4.1.2. JavaScript 24](#_Toc453704211)

[4.1.3. HTML 24](#_Toc453704212)

[4.1.4. CSS 25](#_Toc453704213)

[4.1.5. GitHub 25](#_Toc453704214)

[4.1.6. Apache Server 26](#_Toc453704215)

[4.1.7. XAMPP 26](#_Toc453704216)

[4.2. Cerințele sistemului 27](#_Toc453704217)

[4.2.1. Cerinte functionale 27](#_Toc453704218)

[4.2.2. Cerinte functionale 28](#_Toc453704219)

[4.3. Cazuri de utilizare 29](#_Toc453704220)

[Capitolul 5. Proiectare de Detaliu și Implementare 34](#_Toc453704221)

[5.1. Arhitectura de layer a sistemului 34](#_Toc453704222)

[5.1.1. Nivelul interfata utilizator 35](#_Toc453704223)

[5.1.2. Nivelul de aplicatie 35](#_Toc453704224)

[5.1.3. Nivelul de business logic 35](#_Toc453704225)

[5.1.4. Nivelul de acces de date 35](#_Toc453704226)

[5.1.5. Nivelul de baza de date 35](#_Toc453704227)

[5.2. Arhitectura MVC 36](#_Toc453704228)

[5.3. Patternul Front Controller 38](#_Toc453704229)

[5.4. Componentele aplicatiei 40](#_Toc453704230)

[5.4.1. Componenta de server 42](#_Toc453704231)

[5.4.2. Componenta de client 46](#_Toc453704232)

[5.5. WebRTC API 50](#_Toc453704233)

[5.5.1. Negocierea informatiilor de retea 50](#_Toc453704234)

[5.5.2. Stream API 54](#_Toc453704235)

[5.5.3. PeerConnection API 55](#_Toc453704236)

[5.5.4. Data Channel API 55](#_Toc453704237)

[5.5.5. Stabilirea conexiunii intre doi utilizatori 56](#_Toc453704238)

[5.5.6. Crearea ofertei 58](#_Toc453704239)

[5.5.7. Asteptarea ofertei 60](#_Toc453704240)

[5.5.8. Asteptarea raspunsului 60](#_Toc453704241)

[5.5.9. Asteptarea candidatilor 61](#_Toc453704242)

[5.5.10. Trimiterea datelor prin DataChannel 61](#_Toc453704243)

[5.5.11. Primirea datelor de pe DataChannel 62](#_Toc453704244)

[5.5.12. Vizualizarea istoricului 63](#_Toc453704245)

[5.6. Proiectarae bazei de date 63](#_Toc453704246)

[5.7. Concluzie 64](#_Toc453704247)

[Capitolul 6. Testare şi Validare 65](#_Toc453704248)

[Capitolul 7. Manual de Instalare și Utilizare 66](#_Toc453704249)

[7.1. Instalare si rulare 66](#_Toc453704250)

[7.1.1. Instalare Git 66](#_Toc453704251)

[7.1.2. Instalare XAMPP 67](#_Toc453704252)

[7.2. Utilizarea aplicatiei 70](#_Toc453704253)

[Capitolul 1. Concluzii 71](#_Toc453704254)

[Bibliografie 72](#_Toc453704255)

[Anexa 1 (dacă este necesar) 74](#_Toc453704256)

**Cuprins**

**Capitolul 1. Introducere – Contextul proiectului (Heading 1 style)**

1.1. Contextul proiectului (Heading 2 style)

1.1.1. (Heading 3 style)

**Capitolul 2. Obiectivele proiectului**

**Capitolul 3. Studiu bibliografic**

**Capitolul 4. Analiză și fundamentare teoretică**

**Capitolul 5. Proiectare de detaliu și implementare**

**Capitolul 6. Testare și validare**

**Capitolul 7. Manual de instalare și utilizare**

**Capitolul 8. Concluzii**

**Bibliografie**

[**Anexa 1 (dacă este necesar)**](#h.lnxbz9)

# Introducere – Contextul proiectului (Heading 1 style)

Titlul capitolului se bazează pe Heading 1 style, numerotat cu o cifra (x. Nume capitol), font Times New Roman de 14, Bold.

Ce se scrie aici:

* Contextul
* Conturarea domeniului exact al temei
* reprezintă cca. 5% din lucrare

## Contextul proiectului (Heading 2 style)

Fontul folosit implicit în acest document este Times New Roman, dimensiune de 12, conform *Normal style*, cu spaţiere la 1 rând (Paragraph, Line spacing de 1.0) şi *Justify*.

Pentru prima linie din fiecare paragraf se foloseşte indentare (implicit in *Normal Style*), iar între paragrafe succesive nu se lasă distanţă suplimentară.

### (Heading 3 style)

Fiecare tabel introdus în lucrare este numerotat astfel: Tabel x.y, unde x reprezintă numărul capitolului iar y numărul tabelului din capitol. Se lasă un rând liber între tabel și paragraful anterior, respectiv posterior.

Tabel 1.1 (Insert caption->Tabel)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Times new roman ( 12) | xxxx | xxxx | xxxx |  |
|  |  |  |  |  |

Fiecare figură introdusă în text este citată (de ex: în figura x.y este prezentată ... ) şi numerotată. Numerotarea se face astfel Figura x.y unde x reprezintă numărul capitolului iar y numărul figurii în acel capitol. Folosiţi (Insert caption->Figura).

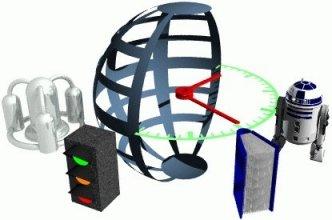


Figura 1.1 Numele figurii (insert->reference->caption->Figura)

Fiecare capitol începe pe pagină nouă.

# Obiectivele Proiectului

În acest capitol se prezintă tema propriu zisă (sub forma unei teme de proiectare/cercetare, formulată exact, cu obiective clare – 2-3 pagini și eventuale figuri explicative).

Reprezintă cca. 10% din lucrare.

# Studiu Bibliografic

În acest capitol se realizează o analiză și evaluare atat a comunicarii in timp real, cat și a unui set de aplicatii care folosesc tehnologii de comunicara in timp real. Se va descrie ce inseamna o aplicatie web, și se vor diferentia solutiile posibile pentru a realiza comnuicare in timp real.

## Dezvoltarea aplicatiilor web

In dezvoltarea a unei aplicatii web, scopurile și functionalitatile aplicatiei sunt separate de mediul in care sunt rulate. In cazul ideal, codul aplicatiilor web nu ar trebuie sa fie dependenta de platforma pe care este rulat, dar din cauza interpretarilor browserelor, pot aparea diferente in functionare, care trebuie tratate separat. O aplicatie web nu este altceva decat un mediu de prezentare și de acces la resurse, din care fac parte atat paginile aplicatiei web cat și resursele de tip imagine, audio sau video. Pentru accesarea acestor resurse se face cu ajutorul unei adrese URI, prin protocolul HTTP. Acest protocol este utilizat pentru a sustine cererile de acces la resursele identificate prin URI.

O aplicatie web reprezinta o colecție interconectată de pagini web cu conținut generat dinamic, menită să ofere utilizatorilor o funcționalitate specifică. Interacțiunea dintre aplicație și utilizatori are loc printr-o interfață web. Orice aplicatie web are la baza arhitectura Client- Server, iar maparea peste aceste componente se face extrem de simplu dacă considerăm că orice browser prin care dorim sa obținem accesul la resursele expuse de aplicație reprezintă Clientul, iar Serverul este cel care răspunde interogărilor pe care clientul le trimite.

Pentru transmiterea interogarilor si a mesajelor, browserul deschide o conexiune catre server prin care ii va trimite un mesaj in care se descrie operatia care se doreste a fi efectuata. In urma trimiterii acestui mesaj, serverul descrie operatia care se doreste a fi efectuata. In urma trimiterii acestui mesaj, serverul verifica daca resursa este disponibila si daca poate sa fie oferita spre afisare. Chiar daca pe server se gaseste resursa respectiva sau nu, clientul va primi un mesaj, ori de eroare, ori de success, impreuna cu continutul resursei accesate de catre utilizator.

Cererile importante si folosite cel mai des in protocolul HTTP sunt GET si POST. GET este adesea utilizat pentru a obține resursele daca acestea sunt disponibile, iar POST oferă posibilitatea introducerii de date in conținutul 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.

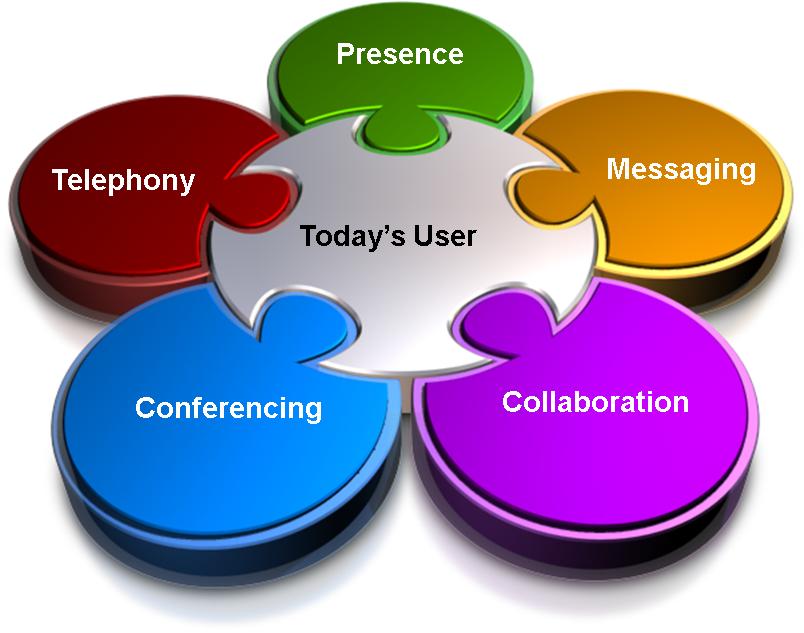
In figura 3.1, se poate observa elementele care construiesc comunicarea in timp real, si cum sunt conectate la utilizator.

figura 3.1

Comunicarea reala se poate imparti in doua moduri: comunicare half-duplex si comunicare full duplex. In comunicare half-duplex, informatiile pot fi transmise in ambele directii, dar nu in acelasi timp, in schimb comunicarea full-duplex, informatiile se pot transmite in ambele directii, in acelasi timp. In general cand vorbim despre RTC, atunci intelegem comunicare peer-to-peer, nu broadcast sau multicast.

In comunicare in timp real, exista intotdeauna o cale directa intre sursa si destinatie, fara ca informatiile sa fie stocate undeva pe durata transmiterii informatiilor. In comparatie cu comunicarile de tip *timeshifting*, informatiile sunt intotdeauna stocate undeva pe un server. In aceasta categorie de timeshifting intra comunicarile prin mail sau comunicare prin commenturi ale retelei de socializare Facebook, unde informatiile sunt salvate pe server, iar intre transmiterea si receptia informatiilor exista o anumita intarziere. In categoria comunicarii in timp real intra urmatoarele: comunicare prin telefon, prin mesaje instante, VoIP, conferinte audio/video, si multe altele.

## WebRTC

In acest subcapitol se va descrie ce inseamna tehnologia WebRTC, si cum s-a ajuns sa existe comunicare in timp real pe web, fara a folosi librarii externe sau pluginuri.

WebRTC este un proiect utilizabil de toata lumea ce asigura comunicare in timp real prin browsere de web si aplicatii mobile. Misiunea de la inceput a fost de a oferi functionalitati de calitate RTC, si care sa se poata integra in diferite platforme mobile sau web, care permite comunicare prin un set de protocoale. WebRTC este sustinut de diferite platforme cum ar fi Chrome, Mozilla sau Opera, si este intretinut de echipele de la Google Chrome.

In Mai, 2011, Google a pornit un proiect open-source, pentru RTC pe web numit WebRTC.[2] Dupa acest proiect, s-a pornit si o serie de lucruri pentru standardizarea protocoalerol relevante in IETF[3] si pentru API-ul respectiv W3C.[4] API-ul a fost facut dupa un proiect anterior facut de WHATWG[6] si a fost referit ca si ConnectionPeer API, iar implementarea conceptului pre-sandardelor a fost facuta la Ericsson Labs.[7]

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

Tehnologia vine cu avantaje mari, cand este vorba de RTC. In primul rand, functionalitatile oferite sunt usor folosite si implementate, si nu este nevoie folosirea unei liberarii externe sau pluginuri de browsere. Tot ce este nevoie este folosirea unui protocol de instantiere a semnalelor, ce are rolul de a transmite informatii de conexiune de la un client la altul. Al doilea avantaj este ca forteaza programatorii de web sa implementeze propria lor securitate si incriptarea datelor.

Avantajele acestei tehnologii se raspandesc nu numai in afara companiilor, ci si in interiorul lor. Reduce costul telefoanelor interne prin inlocuirea lor cu aplicatie WebRTC, si poate oferi comunicare bogata si neintrerupta intre clienti si angajati.

### Arhitectura WebRTC

Arhitectura WebRTC-ului este una complexa, dar folosirea functionalitatilor oferite de aceasta tehnologia este usoara. Arhitectura a fost creata astfel, incat programatorii sa poata folosi si integra cu usurinta functionalitatile WebRTC-ului, iar utilizatorii sa poata folosi aplicatiile web fara descarcarea a unei librarii externe sau pluginuri pentru browseri.

Arhitectura totala se poate observa in figura urmatoare:

.



Fig

La prima vedere se pot observa doua layeri distincte:

1. Programatorii de browseri sunt interesati in API-ul de WebRTC C++
2. Programatorii de aplicatii web sunt interesanti de API-ul de Web

Aplicatiile web vor folosi cabalititatiile de comunicare in timp real oferite de API-ul de Web, oferit de tehnologia WebRTC.

Multe componente sunt abstractizate, oferind creatorilor de aplicatii web sa implementeze propriile lor modalitati de gestiune a datelor cum ar fi componenta de transport is sesiuni folosit pentru diferite protocoale de transport, componenta de STUN/ICE care este folosit pentru stabilirea conexiunii sau componenta de gestionare a sesiunilor.

### Media Path

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

**Peer-to-peer**

****

Figura x

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

**Multi-Unicast**



Figura x

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 x

Mixerul RTP este un punct centralizat care are rolul unui client RTC, dar care are si controlul asupra auido/video-ului folosit in conferinte. Acest client are responsabilitatea de a optimiza toate streamurile si latimea de banda, creand un singur stream ce va fi transmis in continuare.

## WebRTC vs WebSockets

Pentru a implementa o aplicatie web, exista mai multe tehnologii, care sa ofere posibilitatea de a comunica intr-o retea, sau in afara retelei, in timp real. Aceste tehnologii se pot folosi atat separat, cat si combinat, si din cauza aceasta se pot folosi avantajele unor tehnologii fara a se gandi la dezavantajele oferite de celelalte tehnologii.

Una dintre aceste tehnologii este WebSockets. WebSockets este o tehnologie web care ofera comunicare bidirectionala prin protocolul TCP. Este folosita pentru comunicare client-server si server-client, la aplicatii web. Protocolul WebSocket a fost standardizat prima data de IETF, dupa care a fost standardizat API-ul de WebSocket de catre W3C. [x]

Conexiunea este stabilita folosind porturile 80 si 443 in mod implicit. WebSockets fiin bazat pe protocolul TCP, relatia intre protocolul WebSocket si HTTP e de tip handshake, iar potrivit standarului RFC6455, protocolul WebSocket este alcatuit din 2 parti: partea de handshake si transferul de date.[x]

Ca si functionalitate, prima data trebuie stabilita conexiunea intre client si server. Acest lucru se face prin trimiterea mesajelor de handshake intre client si server. Daca handshakeul dintre client si server s-a stabilit cu succes, atunci atat clientul, cat si serverul, poate sa trimita date sau mesaje, deoarece conexiunea este bidirectionala. Dupa ce s-au trimis datele si mesajele, conexiunea intre client si server si poate inchide.

In urma acestor informatii, s-a facut o comparatie intre tehnologia de WebRTC si WebSockets, iar rezultatul se poate vedea in urmatorul tabel:

|  |  |
| --- | --- |
| 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 imagini de video. |
| Comunicarea este bidirecta intre client-client;  Are nevoie de un serviciu extern, numit signalling, prin care se transmit informatii de retea pentru stabilirea conexiunii.  Pentru implementarea unui serviciu de signalling se poate folosi WebSockets, sau diferite librarii care au la baza lor implementare de socketuri (ex. Socket.io) | Comunicarea este bidirecta intre client-server-client, reducand astfel viteza de a trimite date de la un client la altul |
| Suport pentru WebRTC este oferit numai de ultimele versiuni a majoritatea browserelor. Ex: Chrome, Firefox, Opera | Suport pentru socketuri este oferit de toate browserele, fara exceptie. |
| Este browser-browser, iar “camerele” pentru  conversatii sunt create in procesul de signaling | Foloseste handshakeuri HTTP compatibile si porturi default pentru combinarea mai usoara a infrastructurii web existente si a proxyurilor/firewall-urilor |
| Layerul de transport este configurabil | Un API de browser mai usor folosit si implementat |
| Majoritatea aplicatiilor care foloseste implementeaza functionalitatiile de WebRTC folosesc protocolul UDP | Aplicatiile care folosesc socketuri folosesc protocolul TCP |
| Folosind functionalitatea de DataChannel, se pot trimite date neprelucrate de la un client la altul, fara a folosi vreun server de comunicare | Clientii trebuie sa foloseasca neaparat ajutorul unui server pentru a putea comunica intre ei. |
| Datele nu sunt transmise pe cai de “incredere” | Datele sunt transmise pe cai de “incredere” |
| Foloseste servere STUN si TURN pentru a obtine conexiuni cu client care au IP-uri private (clasele A,B,C) | Pentru comunicarea intre clienti nu este necesar sa fie folosit servere de STUN si TURN, deoarece serverul de web decide unde vor ajunge datele de la un client la altul. |
| Pentru a se putea face broadcast (comunicare intre mai mult de 2 clienti in aceasi timp), solutia este aceasi la ambele tehnologii: este nevoie de un server centralizat. Dezavantajul este ca datele mari ce trebuie transmise pot ajunge mai tarziu la client. | |

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

## Sisteme similare

Produsele alternative existente pe piata, sunt sisteme care se concentreaza pe calitatea oferita pentru realizarea comunicarii in timp real, dar exista si alte aplicatii care folosesc tehnologia WebRTC pentru alte scopuri, nu numai comunicare. Aplicatiile se concentreaza atat pe calitate, cat si pentru experienta de utilizator. Aceste aplicatii trebuie sa fie atat usor folosibile, cat si sa ofere calitate. In urmatoarele subcapitole vor fi prezentate unele aplicatii care folosesc pentru comunicare intre utilizatori tehnologia WebRTC.

### Apizee

Este o platforma Saas (Software as a Service) pentru comunicare de web si mobil, in timp real. Solutia oferita de ei ajuta la discutiile de afaceri si imbunatateste colaborarea intrea clientii digitali.

In spatele acestei aplicatii este apirtc-ul, prin care aplicatia realizeaza conectarea si comunicarea intre client. Functionalitatile principale ale aplicatiei sunt: schimb de mesaje de text, audio/video call, colaborare vizuala, multi-dispozitiv, 100% fara pluginuri, integrare si conectare cu alte aplicatii

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

### Client Bee

Aceasta aplicatia are la baza tot technologia webRTC, dar este folosita pentru un total alt scop decat schimbul de mesaje sau conversatiile prin web.

Aceasta aplicatie este folosita pentru: plati online, facturare automata, inregistrare video, video in-browser, reminder automat. Pentru mai multe informatii se poate accesa linkul <https://clientbee.com/>

### eFace2Face

Aceasta aplicatie ofera solutie pentru vanzarea de video bazate pe web si solutii de e-signing.

Imbunatateste productivitatea si efficient cu converatii video, partajarea informatiilor si semnaturilor electornice. Ofera posibilitatea sa te conectezi la mai multi clienti in aceasi timp si este usor de integrat in infrastructure deja existente online. Pentru mai multe informatii se poate accesa linkul <https://eface2face.com/>

### FACEmeeting

O aplicatie simple de “1 click” pentru intalniri online cu conversatii HD audio/video, schimb de mesaje de text, si partajarea fisierelor. Pentru ca foloseste webRTC, nu trebuie downloadat si instalat niciun plugin, iar informatiile de media si de semnalare intre clienti sunt criptate si encodate.

Aplicatia momentan merge numai pe Google Chrome, dar este gratis, in comparatie cu majoritatea aplicatiilor. Pentru mai multe informatii se poate accesa linkul <https://facemeeting.com/>

### Gearcloud Labs - Mixology

Aceasta companie dezvolta o technologie care lasa oamenii si utiliatorii sa impartaseasca experientele noi si unice legat de grafica si streaming de video. Ei o numesc technologia asta “Mixology” si poate fi folosit pe o varietate de aplicatii ce ofera: distractie, jocuri, evenimente live si multe altele.

Technologiile care le folosesc sunt webRTC pentru solutiile de procesare video si OpenCV pentru imaginile si informatiile grafice. Pentru mai multe informatii se poate accesa linkul [www.gearcloudlabs.com](http://www.gearcloudlabs.com/)

### Net Medical Xpress – RTC Conference Switch

Aceast companie folosesc webRTC-ul pentru dezvoltarea aplicatiei lor interne numit “RTC Conference Switch”, pentru a lasa utilizatorul sa-si creeze propria organizatie, grup, si subgroup pentru conferinte de video. Pentru mai multe informatii se poate accesa linkul <http://www.nmxs.com/>

### Proplogic Software Inc - Tawk

Aceasta companie a creat aplicatia de chat numit “Tawk”.

Functionalitatiile principale ale aplicatiei sunt urmatoarele: integrarea aplicatiei in alte aplicatii folosind doar un simplu snippet de cod, chat audio/video, impartasire de monitor, transfer de date securizat si criptat, impartasirea textelor si a fisierelor, optiune fullscreen, camera de conversatie pana la 4 utliziatori. Pentru mai multe informatii se poate accesa linkul <https://tawk.com/>

Dupa cum se observa fiecare aplicatie are la baza technologia WebRTC, si ca si interfata de UI este folosit HTML5. Diferentele intre aceste aplicatii sunt functionalitatile, si serverele pe care ruleaza aplcatia web. Fiind WebRTC-ul raspandit in lume, exista mult mai multe aplicatii care folosesc aceasta technologie, iar mai sus sunt prezentate doar o parte mica din setul de aplicatii care exista peste tot.

# Analiză și Fundamentare Teoretică

În acest capitol vor fi prezentate tehnologiile și uneltele care au contribuit la implementarea aplicației web. Printre acestea se numără: WebRTC, PHP, JavaScript, HTML, CSS, XAMPP, Github etc. Se descriu detaliile considerate necesare în vederea întelegerii codului sursă, cu referire către literatura de specialitate, care detaliaza tehnologiile enumerate.

## 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 in *PHP: Hypertext Preprocessor*.

Codul PHP poate fi integrat cu codul de HTML și poate fi folosit in differite combinații pentru diferite sisteme web. Codul este procesat de un interpretor ca un modul in partea de server sau ca un executabil CGI. Serverul de web combină rezultatele codului PHP interpretat și executat, incluzând imagini și pagini de web generate. Codul PHP poate fi executat și din CLI, și poate fi folosit și pentru implementarea aplicațiilor grafice singulare.[6]

Sintaxa pentru folosirea codului PHP este *<?php…?>* și poate fi combinat cu HTML-ul în urmatorul 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* și *protected*, *metode*, împreuna cu clase *abstracte și finale*. A fost introdusă și o modalitate standard de a declara *constructori* și *destructori*, similar limbajelor de programare C++.

### 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 și 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, și chiar dacă nu ajută la citirea codului, asigură că paginile web sunt rapide și sunt executate in 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 și argument la o altă funcție, care va fi executat la un moment dat. Executarea funcției poate fi imediată, care se numește callback sincron, sau poate fi întarziată, care se numește callback asincron.*[3]

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

### 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 aplicatii mobile*.[2]

Elementele de HTML construiesc blocurile de HTML pentru paginile web. HTML permite folosirea atât a imaginilor, cât și 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 conținut în pagina de web. Există și alte etichete cum ar fi ***<p>…</p>*** ce poate contine informații de text și care poate să conțina 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 și cu CSS pentru a da formă și culoare elementelor de HTML.

HTML5 este a 5-a si, momentan, ultima versiune a limbajului de marcaj HTML. A fost publicată de World Wide Web Consortium ca să îmbunătățească aplicațiile multimedia de ultima generație. HTML5 introduce un API pentru aplicații web complexe, îmbunătățind și 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>*.**

### CSS

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

CSS a fost creeat in primul rand pentru a permite separarea continutului si prezentarea acestuia (aspect, colori, pozitionare). Un avantaj al acestuia ar fi accesibilitatea continutului, deoarece ofera mai multa flexibilitate si control asupra caracteristicilor de prezentare. De asemenea, mai multe pagini HTML pot utiliza acelasi fisier CSS pentru a reduce complexitatea si repetitia in ceea ce priveste stilurile intalnite in aplicatie. Daca paginile HTML necesita elemente diferite de stilizare, acestea pot fi declarate in fisiere .css care contin doar informatiile relevante acelor pagini. De exemplu, paginile de Home si de Inregistrare pot contine declararea stilurilor in fisiere CSS diferite.

### GitHub

GitHub este o aplicație 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 și funcționalități noi. [3] Proiectele pot fi accesate și manipulate folosind linii de comandă standard. Spre deosebire de Git care este o aplicatie folosită strict din linia de comandă, GitHub oferă atât o interfață grafică web, cât și o interfață grafica desktop. O altă funcționalitate întalnită ar fi integrarea cu aplicațiile mobile.

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

### Apache Server

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

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

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

Apache suporta o varietate larga de caracteristici, multe implementari care sa extinda functionalitatea 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 in acest sens fac parte Perl, Python, Tcl si PHP.

Virtual hosting permite instalarea Apache pentru deservirea mai multor aplicatii 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 in mod gratuit, orice programator poate adapta server pentru nevoile proiectului.

### XAMPP

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

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

Acesta vine impreuna cu o serie de module, incluzand OpenSSL, phpMyAdmin, MediaWiki si altele. Mai multe instante ale XAMPP pot exista pe o singura masina, iar orice instanta poate fi copiata de pe un calculator pe altul. XAMPP poate fi folosit atat in versiunea standard (cea mica) cat si in versiunea completa.

Initial, designerii XAMPP-ului au avut intentia de a-l folosi numai ca un instrument de dezvoltare care permite dezoltatorilor si programatorilor site-urilor sa isi testeze programele fara a fi nevoiti sa aiba acces la internet. Pentru a facilita munca acestora, multe dintre caracteristicile XAMPP-ului sunt dezactivate in mod implicit. De asemenea, XAMPP are capacitatea de a deservi pagini web pe World Wide Web [11]. O caracteristica speciala ar fi protejarea anumitor parti ale pachetului cu o parola. O data ce XAMPP este instalat, este posibil sa tratam localhost-ul ca o gazda, prin conectarea folosind un client FTP.

## Cerințele sistemului

Cerințele unui sistem pot fi clasificate între cerințe funcționale și cerințe non-funcționale. Cerințele funcționale prezintă o descriere completă a funcționalităților pe care sistemul trebuie să le îndeplinească, ce să se poată realiza utilizând sistemul. Cerințele non-funcționale dictează proprietăți și constrângeri asupra sistemului și sunt specificate atribute de calitate pe care sistemul trebuie să le dețină.

### Cerinte functionale

Capacitățile funcționale definesc ce acțiuni specifice poate sistemul să ofere. 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

În orice sistem informatic, cerințele non-funcționale sunt identificatorii de calitate ai sistemului. Dacă în subcapitolul precedent am vazut ce funcționalități trebuie să pună la dispoziție sistemul, în următoarele paragrafe vom decide și analliza care sunt principalale calități și constrângeri impuse sistemului.

**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.

**Deploiare**

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 au menirea să ofere o perspectivă globală asupra comportamentului și funcționalităților puse la dispoziție utilizatorilor. Cerințele funcționale ale sistemului nu reprezintă altceva decât cazuri de utilizare posibile, însă nu toate cerințele funcționale trebuie tratate ca și cazuri de utilizare.

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

Use Case

**Fig. 4.1 Cazuri de utilizare pentru user**

**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

In acest capitol se va descrie proiectarea sistemului pentru componentele web, si se va prezenta diagramele de arhitectura folosite, diagrame de secvente si 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 randul lui interactionea cu nivelul de aplicatie si asa mai departe. Aceasta arhitectura este o arhitectura standard folosita pentru creearea aplicatiilor web datorita implementarii a acesteia. Arhitectura fiind pe nivele, se poate adauga si scoate functionalitati fara modificari majore. De exemplu se poate schimba nivelul de acces de date fara a schimba ceva in interfata utilizator.

In figura X se poate observa nivelele de arhitectura folosita in aplicatia web.

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

Figura X

Cum era descris si mai devreme, comunicarea este posibila numai intre nivelele consecutive, creand un cuplaj slab intre componente, astfel aducand avantajul de a aduce schimbari cu usurinta in componentele aplicatiei.

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

### Nivelul interfata utilizator

Acest nivel contine fisierele de .php folosite pentru afisarea paginilor web, fiecare continand diferite elemente de HTML. Fiecare pagina este definit prin fisierul de interfata utliziator si controllerul lui corespunzator. Daca se doreste introducerea unei noi pagini, atunci trebuie creat un fisier separate pentru acea pagina, si un controller ce va contine logica in spatele paginii respective.

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

### Nivelul de aplicatie

Nivelul de aplicatie contine logica din spatele paginilor web. Acest nivel contine functiile 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 si controllerul corespunzator. Daca se doreste schimbarea unei parti de logica, atunci acesta se face cu usurinta prin schimbarea controllerului de la cai, si adaugarea unui nou controller pentru noua logica.

Schimbarile mentionate de mai sus trebuie facute in urmatoarele fisiere: Routes.php pentru schimbarea sau adaugarea cailor folosite de aplicatie, si 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 si accesul la date. Acest nivel preia responsabilitatea de la controller, deoarece controllerul ar fi supraincarcat si 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 contine toate interogarile ce se face catre baza de date. Aceste interogari vin direct de la controller si 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

Arhitectura de layer a aplicatiei este combinat cu o alta arhitectura de design, si anume Model-View-Controller (MVC).

MVC este o arhitectura de software folosit pentru implementari de interfata utilizator. Separa software-ul in trei componente interconectate, ca sa separe reprezentarea interna a informatiilor de afisarea informatiilor catre utilizator[1][2]. Mai demult era folosit pentru interfete grafice de utilizator, dar acest pattern s-a raspandit si pe partea de design de software a aplicatiilor web.

MVC, dupa cum spune si numele de Model-View-Controller, are trei componente principale, iar in figura X se poate observa aceste trei componente de Model, View si Controller si relatiile lor intre ele.



Figura X

Modelul gestioneaza direct datele, logica si regurile aplicatiei.

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

Controllerul receptioneaza intrarile si actiunile utilizatorului si transmite mai departe catre modelul aplicatiei. Rezultatul venit de la model nu trebuie sa treaca neaparat prin controller, ci poate sa actualizeze direct componenta de view a aplicatiei.

In aceasta aplicatie View-ul este reprezentat de paginile PHP. Fiecare pagina PHP are asignat cate o functie din clasa de controller, care are exact aceasi nume ca si numele paginii, si care este configurat in clasa de Routes a aplicatiei. Controllerul are legatura directa cu modelul de date a aplicatiei, iar raspunsul de la model trece tot prin controller, ajungand la pagina, si astfel actualizand pagina respectiva.

Un exemplu de functionare a acestei arhitecturi se poate observa in diagrama de secventa din figura X.

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

Figura X

## Patternul Front Controller

Aplicatia foloseste un design pattern important si necesar aplicatiilor web, si anume patternul Front Controller.

Front Controller este un software design pattern desscris in mai multe cataloage relationand la creearea aplicatiilor web. Patternul ofera un punct de intrare centralizat pentru tratarea request-urilor.[x]

Front Controllerul este adesea folosit in aplicatii web pentru implementarea fluxului de lucru. Desi nu este necesar folosirea acestui pattern, este mult mai usor controlarea navigatiei unui set de pagini printr-un controller unic, decat 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 X se poate observa in detaliu tratarea unui request.

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

Figura X

Dupa cum se observa toate requesturile de la user trec prin scriptul de index.php care va initializa clasa App si care la urma va afisa pagina respectiva. Avantajul acestui pattern este ca nu exista riscul pentru duplicarea datelor si duplicarea tratarea requesturilor, deoarece exista doar un singur punct de intrare care are responsabilitatea de a trata toate requesturile.

## Componentele aplicatiei

In figura X se poate observa cele 3 componente principale ale aplicatiei:

C:\Users\Arctigor\Downloads\module.pngFigura X

Aceste 3 componente principale sunt urmatoarele: *Client Browser*, *Application Server*, *Database Server.*

Utilizatorul are acces doar la partea de client, prin paginile oferite de aplicatie. Pentru folosirea aplicatiei paginile web ofera elemente de html prin care utilizatorul poate interactiona cu aplicatie, si in urma caruia se poate conecta la un alt utilizator.

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

Figura X descrie mai in detaliu arhitectura acestei aplicatii web, si interactionarea concreta intre componentele aplicatiei.

Figura X

Dupa cum se poate observa aceasta figura contine cele 3 componente principale mentionate de mai sus, dar cu mai multe detalii.

### Componenta de server



Figura X

Aceasta componenta contine logica aplicatiei si redirectionarea paginilor la interactionarea utilizatorului cu paginile web. Prin fisierul *App.php* se creeaza o instanta a aplicatiei, la fiecare schimbare de pagina. Obiectul care este instantiat este chiar clasa App care se regaseste in fisierul acesta. Acest obiect la urma lui instantieaza obiectul Controller care se regaseste in fisierul Controller.php. Obiectul App are si o instanta de Route unde sunt predefinite caile aplicatiei impreuna cu fisierul de interfata si metoda de controller corespunzator. La accesarea unei pagini web, obiectul App cauta in rutele predefinita numele paginii web pe care utilizatorul vrea sa acceseze, iar in cazul in care nu se gaseste, se arunca un mesaj de eroare.

Obiectul de *Controller* poate fi accesat doar prin obiectul de App, si contine toate functiile la care utilizatorul poate accesa in mod direct sau indirect. Acesare in mod direct inseamna ca daca utilizatorul ajunge pe o pagina a aplicatiei, atunci este direct apelata functia corespunzatoare din controller, cum ar fi de exemplu login-ul. Acesarea in mod indirect este de exemplu cand utilizatorul doreste sa se conecteze la un alt utilizator, iar transmiterea de informatii se intampla automat, fara ca utilizatorul sa interactioneze mai mult cu interfata. De exemplu ar fi cand utilizatorul cere conexiunea la un alt utilizator, iar butonul de acceptare sau refuzare apare pe pagina al celuilalt utilizator, fara ca acesta sa apese vreun buton.

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

Figura X ne arata diagrama de clasa a partii de server, unde se poate observa cu exactitate clasele si conexiunile intre ele.

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

Figura X

Baza de date este pe un alt server, decat serverul de aplicatii, si este nevoie de o interfata intre cele servere pentru acesarea datelor din baza de date. Cu ajutorul PHP-ului conexiunea este facuta cu usurinta prin apelarea functiei mysqli\_connect("IP", "username", "password", "baza de date");. Dupa este apelata functia respectiva, se pot face interogari la baza de date. Daca functia nu este apelata, se va arunca o exceptie, de care utilizatorul aplicatiei nu ar trebui sa stie, deoarece este responsabilitatea serverului si nu a clientului sa gestioneze erorile de server si de baza de date. Aceasta functie poate fi acesat in obiectul Database, prin functia 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();

}

}

Aceasta functie *render* nu face altceva decat urmatoarele lucruri: verifica daca exista metoda in clasa de controller, si daca exista atunci va apela acea metoda. Daca functia din controller returneaza un obiect de tip JSON atunci nu incarca pagina de interfata, deoarece acest raspuns vine in urma unui apelare AJAX, si nu vrem reincarcarea paginii in acest caz. Daca raspunsul de la functia controllerului nu este de tip JSON, atunci se va incarca pagina respectiva. In toate cazurile contrare utilizatorul va fi redirectionat la o pagina de eroare, cu informatia ca pagina accesata nu exista.

### Componenta de client



Figura X

Componenta de client contine diferite tipuri de fisier: de PHP, JavaScript si CSS.

Fisierele 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 fisierelor PHP sunt accesate indirect de catre utilizator, in urma unor evenimente sau interactionare cu interfata utilizator ce porneste o reactie de apelare de functii. Paginile de JS de care utilizatorul are nevoie pentru folosirea aplicatiei sunt urmatoarele*: jquery-2.2.1.min.js, facebook.js, login\_with\_fb.js, webrtc.js.* Pentru a da forma sa culoare aplicatiei web, sunt folosite fisiere CSS*: header.css, form.css si welcome.css*.

In urmatoarea diagrama din figura X 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 X

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

Acest header va fi intotdeauna prezent, cu mici modificari in meniu, depinzand pe ce pagina se afla utilizatorul. De exemplu daca ii pe pagina principala, atunci apar in meniu linkurile pentru logare si inregsitrare, iar daca utilizatorul este logat in aplicatie, va aparea doar butonul de logout din acest meniu.

Pagina *home*.*php* nu contine altceva decat titlul aplicatiei si meniul din header.

Pagina *login*.*php* contine elementele de logare a utilizatorului si anume: numele utilizatorului, parola si butonul de login. Daca utilizatorul introduce date gresite si apasa pe butonul de login, atunci aplicatia va arunca un mesaj de eroare, ca userul sau parola ii gresita, dupa care utilizatorul va putea reintroduce datele. Daca utilizatorul introduce corect datele, atunci este redirectionat catre pagina de welcome, unde va putea folosi functionalitatile principale oferite de aplicatie. In urma loginului cu success se creeaza sesiunea cu informatiile despre utilizator.

Pagina *loginWithFb*.*php* este goala deoarece foloseste o interfata adusa de API-ul de Facebook. In cazul in care un cont de Facebook nu este asociat cu un cont deja inregistrat, atunci se va arunca un mesaj de eroare. Daca utilizatorul nu este logat pe Facebook, atunci i se va cere sa se logeze, iar daca contul de facebook este deja asociat unui cont existent, atunci utilizatorul va fi redirectionat pe pagina de welcome.

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

Pentru a folosi API-ul de Facebook, trebuie sa creat o pagina de JS, care se numeste in aplicatia aceasta *facebook*.*js*, si 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'));

Daca acest script este incarcat, aplicatia poate folosi functionalitatile oferite de Facebook, cum ar fi accesarea numeui de utilizator, id-ul contului, poza de profil si multe alte informatii. Pentru folosirea acestor functionalitati, trebuie creata mai intai o aplicatie pe Facebook, care va genera un id, si care trebuei sa fie pus la apelarea functiei *FB*.*init* la atributul *appId*.

Pentru login se folosesc functiile *FB.getLoginStatus* pentru a verifica daca este deja un utilizator logat pe Facebook, si *FB.login* pentru logare pe Facebook. Fiecare functie va returna un raspuns de tip JSON.

Pagina *logout*.*php* nu va contine nimic, ci o sa redirectioneze utilizatorul deja logat in aplicatie, pe pagina principala. In urma logout-ului, aplicatia va sterge datele de sesiune si de cookie.

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

Pentru o inregistrare terminata cu success, utilizatorul trebuie sa tina cont de urmatoarele lucruri:

* Toate campurile trebuie completate, fara exceptie
* Numele utilizatorului nu trebuie sa existe deja in baza de date
* Numele utilizatorului nu trebuie sa contina caractere de tip space (“ ”).
* Parola si confirmarea parolei trebuie sa contina aceasi caractere

Daca utilizatorul respecta regurile de mai sus, atunci este redirectionat pe pagina principala, cu un mesaj de success, in caz contrar, va ramane pe acelasi pagina, urmat de un mesaj de eroare coresponzator regurilor. Dupa redirectionarea utilizatorului, acesta se va putea loga in aplicatie si sa foloseasca functionalitatile principale oferite de aplicatie.

Pagina *history*.*php* poate fi accesata numai daca utilizatorul este logat deja in aplicatie, altfel aplicatia va arunca un mesaj de eroare. Aceasta pagina va contine istoria conversatiei intre 2 utilizatori, pe baza id-urilor acestora, si poate fi accesata numai din pagina de welcome. Informatiile de conversatie vor fi afisate in formatul urmator:

*[data] nume de utilizator: mesaj*

Pagina *welcome*.*php* contine toate elementele principale ce poate folosi utilizatorul in aplicatie: conectare la un alt utilizator, aduagarea unui prieten, afisarea istoriei de conversatii, conectare la un cont de Facebook, chat text si audio/video si transmiterea fisierelor.

La incarcarea acestei pagini se incarca si scripturile *webrtc*.*js* si *jquery-2.2.1.min.js.* Folosirea librariei jquery este necesara, deoarece aplicatia foloseste ajax call-uri pentru a accesa functiile din server, mai exact din controller, fara ca pagina sa fie reincarcata.

Scriptul *webrtc*.*js* are acces la toate functionalitatile oferite de WebRTC, deoarece, dupa cum se observa in diagrama de arhitectura de design a aplicatie, scripturile de JS ruleaza pe browseri, iar in browser este implementat deja tehnologia WebRTC ce ofera functionalitatile pentru conexiune peer-to-peer. Acest script contine si tratarea evenimentelor de apasare de butoane, dar si a evenimentelor trimise de WebRTC. Tratarea evenimentelor si conexiunea propriu-zisa va fi prezentat in subcapitolul unde se va descrie in detaliu functionalitatile oferite de WebRTC si cum sunt acestea integrate in aplicatia de web.

## WebRTC API

WebRTC este tehnologia oferita de browserele de web, prin care se pot interconecta 2 si pot vorbi peer-to-peer prin mesaje de text sau audio/video, fara ca aceste informatii sa treaca printr-un server de aplicatii.

**Principiu de functionare**

* In principu, pentru conectarea a 2 utilizatori, trebuie facuti urmatorii pasi:
* Crearea obiectului de conexiune WebRTC
* Trimiterea ofertei de la un utilizator la altul cu informatiile de SDP, prin serviciul de semnalare.
* Trimiterea raspunsului de la al 2-lea utilizator cu informatiile de SDP, tot prin serviciul de semnalare.
* Cautarea si adaugarea candidatiilor ICE si trimiterea acestor candidati la celalalt utilizator prin serviciul de semnalare
* Tratarea evenimentelor pentru deschiderea/inchiderea conexiunilor

In aceasta aplicatie, acest principiu este transformat intr-o implementare proprie, si anume in scriptul webrtc.js.

In urmatoarele subcapitole vor fi prezentate elementele principale de WebRTC iar dupa o sa urmeze integrarea acestor elemente in aplicatia de web.

WebRTC: Comunicare in timp real intre browseri, defineste un API de JavaScript, care poate fi folosita in aplicatii web pentru a folosi functionalitatile oferite de WebRTC pentru comunicare peer-to-peer.

Prin acest API se stabileste o conexiune intre doua puncte finale, iar fiecarui punct final este asignat cate un rol: offerer si answerer. Cel care porneste conexiunea si negocierea de date este offerer, iar cel care raspunde la aceasta negociere este answerer. Aplicatia trateaza aceste roluri, astfel incat ca fiecare punct final poate sa inceapa conexiunea.

### Negocierea informatiilor de retea

Negocierea datelor se face prin serviciul de semnalare. In acest caz serviciul de semnalare este folosirea bazei de date. Aplicatia va scrie in baza de date informatiile de conexiune oricand un utilizator vrea sa inceapa o conexiune cu un alt utilizator. Este necesara implementarea proprie a serviciului de semnalare, deoarece API-ul de WebRTC nu vine cu mecanismul de semnalare proprie. Informatiile ce trebuie transmise prin serviciul de semnalare sunt acestea: Session Description Protocol, si ICE candidati.

SDP-ul este un format ce descrie parametrii de initializare a streamingului de media. Acest format este destinat pentru descrierea sesiunilor de comunicare multimedia, pentru scopul de invitatii de sesiune sau negocierea parametrilor. SDP-ul nu livreaza instantele de media, ci este folosit pentru negocierea tipului de media,formatul si proprietatile asociate acestora, intre cele doua puncte finale.[x]

Candidatii ICE sunt transmise cu informatiile de ip prin NAT(Network Address Translator). ICE vine de la Interactive Connectivity Establishment, si este folosit pentru crearea comunicatie pentru VOIP, peer-to-peer, mesaje instante si alte interactivitati de media. Formatul este urmatorul:

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

Aici UDP-ul ne zice ce fel de protocol este folosit, typ host specifica ce tip de ICE candidat este, iar *host* ne zice ca candidatii sunt generate inauntrul firewall-ului. Un alt tuo de candidat este tipul relay, care sunt folosite pentru comunicarile in exteriorul firewall-ului.

Aceste informatii se pot trimite in orice forma de la un peer la altul prin orice format si protocol, atata timp cat fiecare peer intelege acel protocol.



Figura X

Diagrama din figura X arata arhitectura, unde semnalarea intre punctele finale de WebRTC traverseaza serverul de web, in timp ce conexiunea de WebRTC se stabileste intre cele doua puncte finale de WebRTC.

Pentru a colecta candidatii de ICE, punctele de conexiune trebuie configurate astfel incat sa aiba informatii despre serverele de STUN si TURN, acesta, in cazul in care aplicatia este rulata in afara retelei private. Momentan, nu exista mecanism pentru descoperirea acestor servere, si de aceea trebuie configurate in scripturile de JavaScript. Pentru folosirea serverelor, trebuie introduse intr-un sir adresele acestora. Mai jos se poate vedea un exemplu cum se configureaza aceste servere de STUN si 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.iptel.org'},*

*{'urls':'stun:stun.rixtelecom.se'},*

*{'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>'}*

*]*

*}*

Functionarea acestor servere impreuna cu candidatii ICE este complexa, si nu este vizibil utilizatorului. Trebuie mentionat ca un candidat te tip host contine adreasa de IP local si portul clientului ICE. Acesta reprezinta adresa hostului, chiar daca nu se foloseste ICE.

Un candidat de server reflexiv contine o adresa de IP si portul clientului ICE, vazut de serverul de STUN. Daca clientul ICE este locat dupa un NAT, acest candidat va contine adresa publica ce a fost asignat de NAT, clientului ICE. Clientul ICE obtine candidatul prin trimiterea unei cereri folosind protocolul de STUN pentru NAT. Serverul de STUN detecteaza sursa adresei si trimite inapoi la client. In caz ca nu exista NAT intre clientii de ICE si serverul de STUN, atunci adresa candidatului de server reflexiv va fi identic cu al candidatului de host.

Este important sa se stie ca conexiunea intre clientii ICE nu va traversa serverul STUN. Serverul exista pentru tratarea requesturilor STUN si pentru a oferi candidatii de server reflexive clientilor ICE.

WebRTC-ul nu defineste un mecanism pentru gasirea serverelor STUN si TURN de catre clientii ICE, de aceea trebuie configurate manual aceste servere.

In diagrama de secvanta din figura X, potrivit surselor[xx], se poate observa interschimbarea informatiilor intre cei doi utilizatori folosind servere de STUN si TURN, iar in figura X de la sursa [xxx] se poate observa crearea conexiunii dintre cei doi clienti, impreuna cu starea actuala a obiectului PeerConnection.



Figura X



Figura X

### Stream API

Obiectul de MediaStream trateaza streamurile de media (audio si video). Fiecare MediaStream are un obiect MediaStreamTrack care poate fi trimis la utilizatorul remote, prin obiectul PeerConnecion. Fiecare MediaStream poate sa contina cateva canale, cum ar fi de exemplu canalul pentru video si canalul pentro stereo sau sunet 5.1. Cu un singur MediaStream se poate controla toate trackurile de media, inclusiv datele ce se pot trimite la utilizatorul remote.

Obiectul de MediaStream poate fi creat apeland functia *getUserMedia()*. In acest caz va lua de la utilizatorul local camera de web si microfonul ca si data de input. Pentru afisarea acestui stream intr-o pagina web, se poate face prin crearea unui URL prin functia createObjectURL(MediaStream stream) prin care se pune ca si sursa intr-un tag de HTML, si anume *<video></video>.*

### PeerConnection API

In mijlocul implementarii WebRTC se afla obiectul PeerConnection[x]. Obiectul PeerConnection permite comunicare directa intre browseri. Inainte sa se stabileasca o conexiune directa, obiectul PeerConnection are nevoie de un canal pentru tratarea negocierilor si a semnalarii.

Prima parte a negocierii este schimbarea de adrese IP intre cei doi utilizatori. Deoarece utilizatorii se afla in spatele unui NAT (Network Address Translation) sau un firewall, exista un mecanism deja implementat in WebRTC, prin care se poate afla adresa reala a utilizatorlor. Cand se creeaza obiectul PeerConnection, primul parametru este o adresa catre un server STUN sau TURN. Al doilea paramtru este o functie care este chemata cand PeerConnection vrea sa trimita ceva semnale de daet pentru celalalt utilizator. Datele de semnale sunt procesate de catre functia *processSignalinMessage(message)* implementata in obiectul PeerConnection.

Cand conexiunea de PeerConnection este stabilita, exista posibilitatea de a adauga streamuri de media. Cand un stream media este adaugat, obiectul PeerConnection va negocia codec-uri in loc de canale de semnalare. Dupa ce negocierea este pornita, se trimit evenimente catre cealalta parte a conexiunii, indicand ca streamul de media a fost adaugat, si apoi se poate decide ce se va face cu acel stream media.

### Data Channel API

Canalul de date WebRTC este realizat folosind doua streamuri SCTP unidirectionale, fiecare avand aceasi valoare de ID de stream. Protocolul SCTP care ofera streamuri de SCTP, sunt incapsulate peste DTLS. Urmatoarea figura X ne arata structura de protocoale pentru realizarea canalurilor de date in WebRTC.



Figura X

Canalul de date nu defineste un layer peste layerul de SCTP, dar cele doua streamuri SCTP din canalul de date si datele ce trebuie transportate prin canalul de date este adaugat la layerul de SCTP ca si data de user SCTP.

Folosind un set de extensii SCTP, se pot oferi diferite proprietati la canalul de date. Datele pot fi transmise intr-o anumita ordina sau random, depinzand de nevoile utilizatorului.

Mecanismul canalului de date nu limiteaza numarul de canaluri de date ce pot fi deschise simultan, iar numarul maxim de streamuri este definit de layerul de SCTP, care este 65535, iar fiecare limita este dependenta de date particulare si de implementarile canalului de date ale punctulilor finale ale aplicatiei.

### Stabilirea conexiunii intre doi utilizatori

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



Figura X

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 in scriptul de JavaScript webrtc.js, impreuna cu trimterea fisierelor si a chatului audio/video.

Pentru crearea conexiunii trebuie initializate urmatoarele clase: RTCPeerConnection, RTCIceCandidate si RTCSessionDescription. Aceste clase vor fi instantiate in urmatorul mod, depinzand de browserul in care este rulata aplicatia.

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;

Dupa cum se vede, pentru crearea canalului de date se apeleaza functia implementata in obiectul de PeerConnection deja existent, si anume createDataChannel. Se initializeaza si tratarile de evenimente pentru fiecare tip de eveniment:

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

Cand se seteaza toate acestea mentionate mai sus, se apeleaza metoda de crearea de oferta, *createOffer*, dupa 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);

dupa care se scrie in baza de date urmatoarele informatii: id-ul userului care trimite cererea de oferta, id-ul userului catre care se trimite cererea de oferta, tipul ofertei – offer, si informatia de SDP local, de care o sa aiba nevoie celalalt utilizator pentru crearea conexiunii.

Daca in baza de date exista deja o cerere de oferta intre cei doi utilizatori, atunci nu se va scrie o noua cerere in baza de date, ci se va suprascrie cererea veche cu informatiile noi de SDP.

Scrierea in baza de date se face cu apelari ajax folosind libraria jquery, in felul urmator:

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

$.ajax({

data : offerJSON,

type : "post",

url : url,

});

}

Parametrul offerJSON este un obiect JSON care contine informatiile necesare pentru crearea ofertei, iar parametrul url este calea catre template-ul si functia de controller care se ocupa cu scrierea ofertei in 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 catre clientul respectiv. Daca exista o oferta in baza de date, acesta este citita de utilizator, dupa care poate alege sa accepte sau sa refuze oferta. In cazul in care utilizatorul refuza oferta, conexiunea se va inchide, iar in baza de date se va finaliza oferta cu tipul de oferta decline.

Daca utilizatorul accepta oferta, atunci se creeaza o alta instanta de RTCPeerConnection pentru utilizatorul remote, si se initializeaza, ca si la utilizatorul local, streamurile de media si canalul de date.

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

createDataChannel();

setLocalMedia();

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

remotePeerConnection.setRemoteDescription(remoteSDP);

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

remotePeerConnection.setLocalDescription(sessionDescriptionProtocol);

}

Dupa setarea acestor valori se va actualiza in baza de date cererea de oferta cu informatiile de SDP remote impreuna cu tipul de oferta answered.

### Asteptarea raspunsului

Asteptarea raspunsului se face similar cu asteptarea ofertei, insemnand ca se va citi periodic din baza de date, oferta care are tipul answered. Daca se va gasi astfel de oferta, atunci creearea conexiunii va merge catre pasul urmator si anume adunarea candidatii de ICE. Dar inainte sa se adune acesti candidati, se va seta SDP-ul clientului remote la conexiunea locala de RTCPeerConnection, prin apelarea metodei *setRemoteDescription*. Cand se seteaza SDP-ul remote, automat se va trimite un eveniment ce va cauta candidatii de ICE, evenimentul fiind numit *onicecandidate*.

Dupa adunarea acestor candidati ICE, se va actualiza in baza de date cu candidatii actuali si 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);

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

Dupa cum se observa, exista multe asteptari de oferte si raspunsul, iar toate acestea se fac prin apelarea functiilor ajax de la jquery. Un exemplu de asteptare de raspuns ar fi urmatorul.

**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 fisierelor prin canalul de date, utliziatorii trebuie prima data sa se conecteze. Atat pentru transmiterea de fisiere, cat si transmiterea mesajelor de text, se foloseste functia *send*() din DataChannel. Un exemplu de trimitere de mesaj se face in felul urmator:

**function** sendData() {

**var** data = dataChannelSend.value;

**var** username = getMyUsername();

**var** date = getCurrentTime();

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

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

}

Dupa cum se observa, functia sendData() face urmatoarele lucruri:

* Se ia valoarea de text din TextArea-ul in care a scris utilizatorul mesajul lui
* Se ia numele utilizatorului care trimite mesajul
* Se calculeaza timpul local
* Se afiseaza in TextArea-ul de mesaje primite data, username-ul si mesajul
* Se trimite prin canalul de date mesajul impreuna 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 afisarea mesajului.

Pentru trimiterea fisierelor se incarca un fisier de pe calculatorul utilizatorului, dupa care fisierul este trimis pe bucati la celalalt utilizator, byte cu byte. In timp ce utilizatorul incarca fisierul de pe calculatorul lui, numele fisierului va fi scris in baza de date pentru schimbul de informatii.

### Primirea datelor de pe DataChannel

La primirea mesajelor de text se face prin tratarea eventului de onmessage. La primirea mesajelor de text se face decodarea mesajului pentru a afla numele utilizatorului de la care a venit mesajul, si se afisesaza in TextArea-ul utilizatorului, la fel cand se afiseaza trimiterea mesajului. Dupa afisarea mesajului se salveaza mesajul in baza de date, pentru a putea vizualiza in viitor conversatia intre cei doi clienti. Salvarea se face printr-o simpla scriere in baza de date a urmatoarelor informatii: id-ul clientului local, id-ul clientului remote, mesajul primit, si data in care a primit mesajul.

La primirea fisierelor se creeaza prima data 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);

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

Deoarece mesajele si fisierele sunt transmise in acelasi fel, la primire trebuie diferentiate cele doua tipuri de mesaje. Acesta se face printr-un bloc de try/catch.

**try** {

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

...

} **catch** (err) {

**var** receiveBuffer = [];

...

}

Dupa cum se observa, se verifica daca poate sa faca decodarea mesajului primit. Daca acesta nu arunca o exceptie, atunci inseamnca ca mesajul primit este de tip text. In caz contrar aceasta exceptie este prinsa, si este tratata ca mesajul fiind fisier.

Mesajele de la un utilizator la altul se pot transmite in 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. Functia care face acest lucru este urmatoarea

**function** viewHistory(friend) {

**var** peerId = friend.id;

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

}

Functia *window.open* este deja implementata in browser-ul clientului, iar in scriptul de JacaScript, tot ce trebuie facut este sa fie apelata functia respectiva.

## Proiectarae bazei de date

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

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

Figura X

Dupa cum se observa, tabelul user, are relatie m-n cu toate celelalte tabele, cheile primare fiind *id-ul* si *username-ul*. Aceste chei sunt atat primare cat si unice.

## Concluzie

Aplicatia de web implementeaza tehnologia WebRTC oferita de browserele de web folosind JavaScript si PHP, iar mecanismul de semnalare implementat este scrierea si citirea din baza de date. Avantajul acestui mecanism de semnalare este ca nu avem nevoie de un server separat pentru gestionarea negocierilor si a transmiterilor de informatii de retea, ci doar de aplicatia propriu-zisa, care are conexiune catre baza de date.

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

Baza de date fiind MySql, se poate integra si folosi cu usurinta in codul de PHP, astfel creand simplitate in codul aplicatiei.

# Testare şi Validare

Aproximativ 5% din total.

# Manual de Instalare și Utilizare

Acest capitol conține sub formă de tutoriale pașii de urmat pentru realizarea cu succes a instalării componentelor sistemului pe o mașină locală și eventualul deployment într-o rețea locală. Tot în acest capitol vor fi prezentate resursele software și hardware necesare, dar și un scurt manual de utilizare.

## Instalare si rulare

Pentru instalarea aplicatiei pe o retea locala avem nevoie de urmatoarele aplicatii instalate pe clientul local: Git, XAMPP si un browser de web (Chrome, Opera), iar sistemul de operare trebuie sa fie Windows 7,8 sau 10.

### 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.

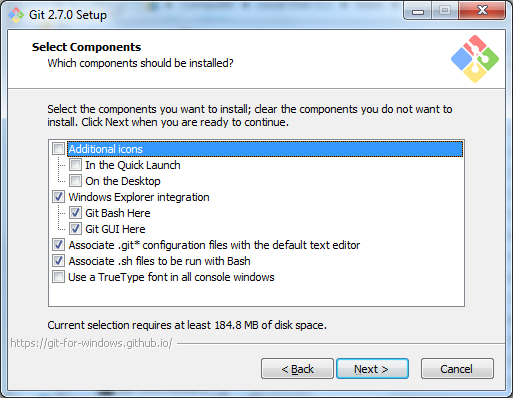


Figura X

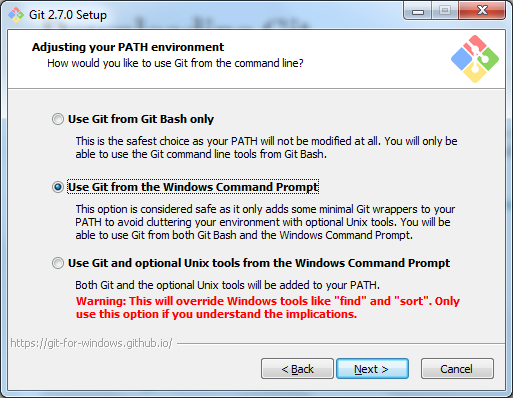


Figura X

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

### Instalare XAMPP

Pentru instalarea programului XAMPP, utilizatorul trebuie sa intre pe pagina

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

si sa descarce ultima versiune a programului. Dupa ce se termina descarcarea se poate porni executabilul de instalare si cu instalarea programului in sine.

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

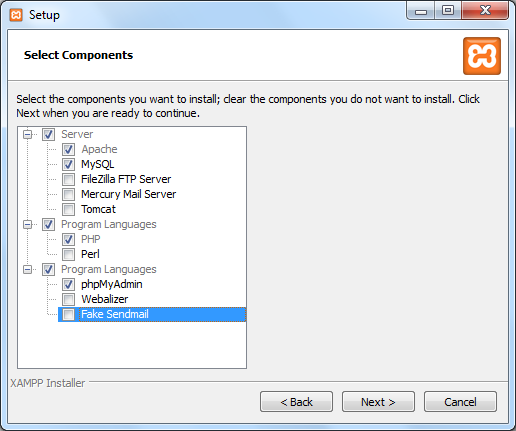


Figura X

Fiind programele Git si XAMPP instalate, se poate descarca aplicatia de pe siteul github. Utilizatorul trebuie sa intre din Command Line in 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)

Dupa ce se ruleaza aceasta comand, aplicatia va fi descarcata in folderul htdocs sub numele de webrtc. Urmatorul pas este sa intre in folderul de webrtc\server-conf si se copieaza cele doua fisiere in folderele de configuratii a serverului de apache, care se afla pe urmatoarele cai.

*...\xampp\apache\conf*

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

Dupa aceste configurari, utilizatorul trebuie sa modifice fisierul de hosts din C:\Windows\System32\drivers\etc\hosts cu urmatoarele informatii.

*<IP local> szabi.dev*

Un exemplu ar fi 192.168.1.103 szabi.dev.

Dupa aceste modificari se porneste Control Panelul de la XAMPP si se porneste serverul de Apache si MySQL, dupa cum se vede in figura X.

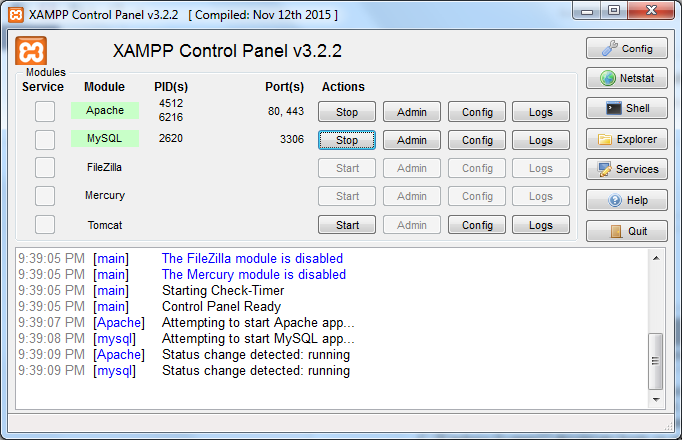
**

Figura X

Dupa ce se pornesc cu succes cele doua module, urmatorul pas este creearea bazei de date cu informatii de baza care vin in mod implicit cu aplicatie. Clientul trebuie sa navigheze pe linkul

<http://localhost/phpmyadmin/>

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

## Utilizarea aplicatiei

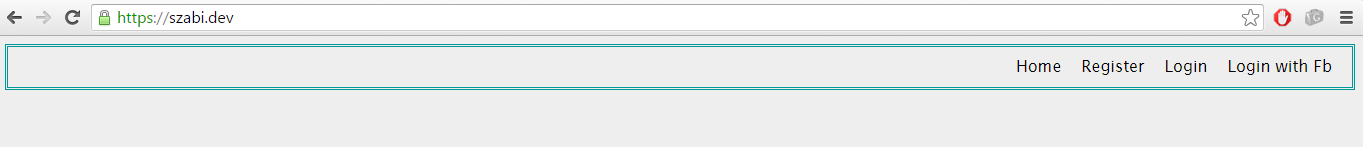
La navigarea aplicatiei la pagina principala, utilizatorul poate avea trei optiuni, dupa cum se vede si in figura X de mai jos: inregistrare, logare simpla, logare cu facebook.

Figura X

La inregistrare utilizatorul trebuie sa introduca datele valide, ca sa se poate inregistra, dupa regurile mentionate in capitolul 5. Dupa ce clientul a creat cu succes un utilizator, poate sa se logeze cu acel utilizator, dupa care va fi redirectionat pe pagina de welcome, unde se poate conecta si comunica cu alti utilizatori.

Pe pagina de welcome poate sa adauge prieteni apasand pe butonul de Add friend, si poate incepe o conversatie de tip text sau audio/video apasand pe unul dintre lista de prieteni si apoi pe butonul de Start.

Daca 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. Daca se accepta conexiunea, atunci trebuie sa astepte pana ce aplicatia va crea conexiunea intre cei doi utilizatori.

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

Daca se creeaza o conexiune dintre cei doi utilizatori, se pot transfera intre ei fisiere de orice tip, prin a da click pe butonul Send file, dupa ce se va alege fisierul dorit.

La iesirea din aplicatie, utilizatorul poate sa dea click pe butonul de Logout aflat in meniul aplicatiei.

O conexiune cu success se poate vedea in figura X de mai jos.

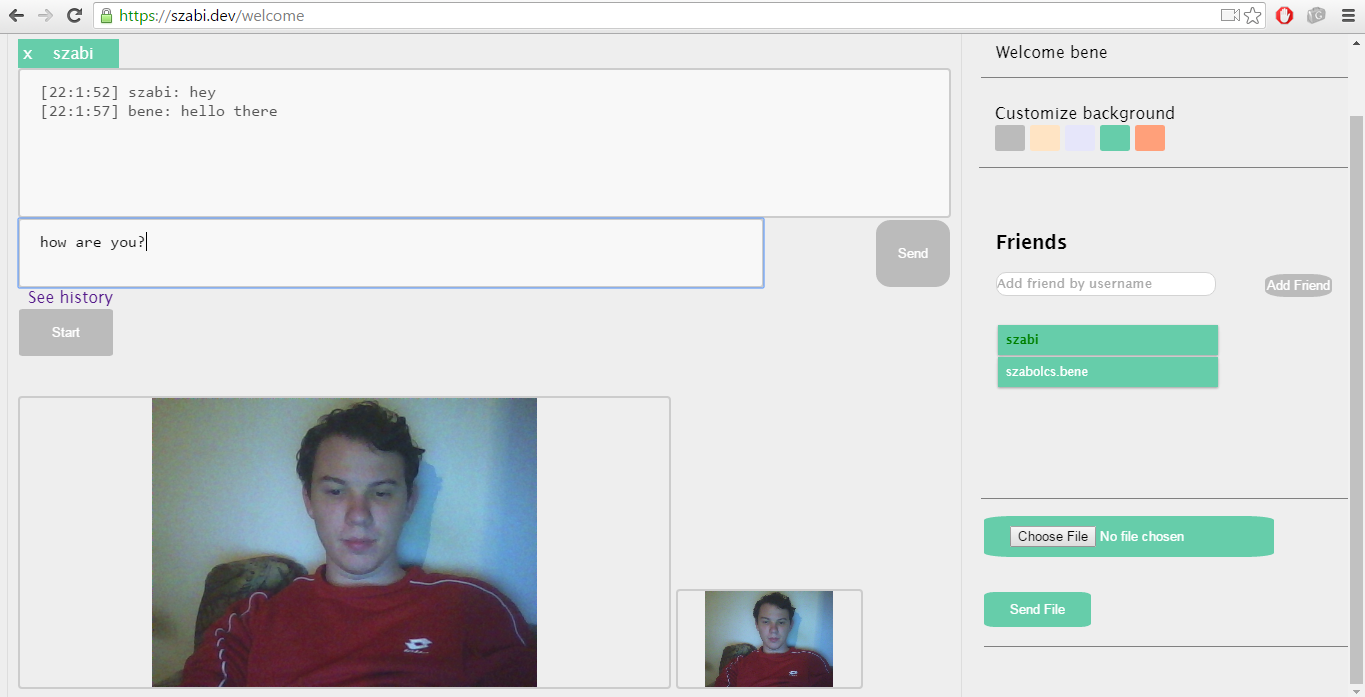


Figura X

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

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

# Concluzii

Cca. 5% din total.

Capitolul ar trebui sa conţină (nu se rezumă neapărat la):

* un rezumat al contribuţiilor voastre
* analiză critică a rezultatelor obţinute
* descriere a posibilelor dezvoltări și îmbunătăţiri ulterioare

# Bibliografie

[1] A. Bak, S. Bouchafa, and D. Aubert, "Detection of independently moving objects through stereo vision and ego-motion extraction," in *IEEE Intelligent Vehicles Symposium (IV)*, San Diego, USA, 2010, pp. 863-870.

[2] A. Chambolle and T. Pock, "A First-Order Primal-Dual Algorithm for Convex Problems with Applications to Imaging," *Journal of Mathematical Imaging and Vision,* vol. 40, pp. 120-145, 2011.

[3] R. C. Gonzalez and R. E. Woods, *Digital Image Processing. Second Edition.*: Addison-Wesley Longman Publishing Co., Inc., 2001.

[4] Ajax Tutorial, <http://www.tutorialspoint.com/ajax/>.

[5] Perkins, C. ; Westerlund, M. ; Ott, J.: Web Real-Time Communication

(WebRTC): Media Transport and Use of RTP / IETF Secretariat. Version:Marz

2012. http://tools.ietf.org/html/draft-ietf-rtcweb-rtp-usage-02. 2012

(draft-ietf-rtcweb-rtp-usage-02). { Internet-Draft

2["Google release of WebRTC source code from Harald Alvestrand on 2011-05-31"](http://lists.w3.org/Archives/Public/public-webrtc/2011May/0022.html). public-webrtc@w3.org. Retrieved 2012-09-12.

3 [Charter of the Real-Time Communication in WEB-browsers (rtcweb) working group](http://datatracker.ietf.org/wg/rtcweb/charter/)

**4**[**Jump up^**](https://en.wikipedia.org/wiki/WebRTC#cite_ref-4) [*"WebRTC 1.0: Real-time Communication Between Browsers"*](http://www.w3.org/TR/webrtc/).

 5[*"Introduction — HTML Standard"*](http://www.whatwg.org/specs/web-apps/current-work/multipage/introduction.html#history-1). Whatwg.org*. Retrieved2012-09-12*.

**6**[**Jump up^**](https://en.wikipedia.org/wiki/WebRTC#cite_ref-7) [*"Beyond HTML5: Peer-to-Peer Conversational Video | Ericsson Labs"*](http://www.ericsson.com/research-blog/context-aware-communication/beyond-html5-peer-peer-conversational-video/).

http://searchunifiedcommunications.techtarget.com/definition/real-time-communications

[heart]W3C: PeerConnection. Working Draft. http://dev.w3.org/2011/webrtc/

editor/webrtc-20111004.html#peerconnection. Version: Oktober 2011

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

<http://www.slideshare.net/cryingnavi/webrtc-20150528>

1. "More deeply, the framework exists to separate the representation of information from user interaction." [The DCI Architecture: A New Vision of Object-Oriented Programming](http://www.artima.com/articles/dci_vision.html)- [Trygve Reenskaug](https://en.wikipedia.org/wiki/Trygve_Reenskaug) and [James Coplien](https://en.wikipedia.org/wiki/James_Coplien) - March 20, 2009.
2. [**Jump up^**](https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller#cite_ref-2) Burbeck (1992): "... the user input, the modeling of the external world, and the visual feedback to the user are explicitly separated and handled by three types of object."
3. Alur, Deepak; John Crup; Dan Malks (2003).Core J2EE Patterns, Best Practices and Design Strategies, 2nd Ed. Sun Microsystems Press. pp. 650pp.[*ISBN*](https://en.wikipedia.org/wiki/International_Standard_Book_Number) [*0-13-142246-4*](https://en.wikipedia.org/wiki/Special:BookSources/0-13-142246-4).

# Anexa 1 (dacă este necesar)

…

Secţiuni relevante din cod

…

Alte informaţii relevante (demonstraţii etc.)

…

Lucrări publicate (dacă există)

etc.