



UNIVERSITATEA DE VEST DIN TIMIȘOARA  
FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ  
PROGRAMUL DE STUDII DE LICENȚĂ: Informatică

# LUCRARE DE LICENȚĂ

**COORDONATOR:**  
Conf. Dr. Mîndruță Cristina

**ABSOLVENT:**  
Ovidiu Bachmațchi

TIMIȘOARA  
2023

UNIVERSITATEA DE VEST DIN TIMIȘOARA  
FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ  
PROGRAMUL DE STUDII DE LICENȚĂ: Informatică

## Utilizare WebRTC API. Studiu de caz

**COORDONATOR:**  
Conf. Dr. Mîndruță Cristina

**ABSOLVENT:**  
Ovidiu Bachmațchi

TIMIȘOARA  
2023

# Abstract

This study dives into the application of the WebRTC API for the creation of a web-based solution to facilitate live interviews between tech job candidates and software professionals. The principal objective is to create a more streamlined, efficient interview process where both interviewees and interviewers can write and run code simultaneously, without reliance on a continuous connection to an external server. Another notable advantage of this application is its potential for limitless scalability, as it bypasses central server data transmission in favor of direct user-to-user data transfer through WebRTC peer-to-peer technology.

The research strategy comprises a thorough exploration of the WebRTC API, scrutinizing its potential pros and cons for real-time web applications, as well as the practical implementation of the technology. Data was gathered via API documentation analysis, hands-on trials, and functional application testing.

This WebRTC application can notably enhance the interviewing process by cutting down latency and eradicating dependency on external servers, thereby uplifting the user experience. Since there's no user limit, the system showcases immense scalability. Further exploration is suggested to evaluate the long-term effects of WebRTC usage in such an application and to delve deeper into opportunities for technology optimization and extension.

# Abstract

Această analiză se axează pe implementarea API-ului WebRTC pentru a genera o soluție care permite desfășurarea eficace a interviurilor live între candidatii la posturi în IT și experții în dezvoltare software. Obiectivul este construirea unui sistem de interviu mai intuitiv și mai productiv, unde atât candidatii interviului, cât și interviewatorii pot scrie și rula cod în același timp, scăpând de necesitatea unei conexiuni permanente la un server extern. O altă caracteristică notabilă a acestei aplicații este capacitatea sa de scalabilitate, deoarece evită trimiterea de date printr-un server central, promovând transferul direct de date între utilizatori prin tehnologia peer-to-peer WebRTC.

Abordarea de cercetare implică o examinaremeticuoasă a API-ului WebRTC, evaluând potențialele plusuri și minusuri pentru aplicațiile web în timp real, alături de punerea în practică a tehnologiei. Datele au fost obținute prin studierea documentației API-ului și prin teste aplicative.

Aplicația WebRTC are capacitatea de a îmbunătăți considerabil procesul de interviu prin scăderea latenței și eliminarea necesității de servere externe, sporind în acest fel experiența utilizatorului. Lipsa unei limite de utilizatori face ca sistemul să ofere o scalabilitate imensă. Se propune continuarea cercetărilor pentru a evalua efectele pe termen lung ale folosirii WebRTC într-o aplicație de acest gen și pentru a examina oportunitățile de optimizare și dezvoltare a tehnologiei.

# Cuprins

<b>1</b>	<b>Introducere</b>	<b>6</b>
1	Definiția Problemei . . . . .	6
2	Delimitări și Limitări . . . . .	7
3	Scopul studiului . . . . .	7
<b>2</b>	<b>Soluții similare</b>	<b>8</b>
1	Prezentare soluții similare . . . . .	8
2	Asemănări dintre soluțiile similare și aplicația mea . . . . .	8
3	Deosebiri dintre soluțiile similare și aplicația mea . . . . .	9
<b>3</b>	<b>Despre WebRTC</b>	<b>10</b>
1	Arhitectura WebRTC . . . . .	10
2	WebRTC Protocol Stack . . . . .	11
3	Conexiunea intre doi useri . . . . .	12
<b>4</b>	<b>Descrierea aplicație</b>	<b>14</b>
1	Arhitectură Aplicație . . . . .	14
2	Facilități Aplicație . . . . .	15
3	Cazuri de utilizare aplicație . . . . .	17
<b>5</b>	<b>Detalii de implementare</b>	<b>25</b>
1	Conexiunea dintre doi useri . . . . .	25
2	Tehnologii utilizate în dezvoltarea client-side . . . . .	27
3	Tehnologii utilizate în dezvoltarea server-side . . . . .	29
<b>6</b>	<b>Exemple de utilizare</b>	<b>32</b>
1	Interviu Tehnic la Distanță . . . . .	32
2	Colaborare în Echipă (Pair Programming) . . . . .	33
<b>7</b>	<b>Concluzii și Directii Viitoare de Dezvoltare</b>	<b>34</b>
1	Îmbunătățirea Interfeței Utilizator . . . . .	34
2	Extinderea Funcționalităților . . . . .	34
3	Implementarea unui Sistem de Evaluare Automată . . . . .	35
4	Integrarea cu Sisteme de Recrutare și Gestionare a Talentelor . . . . .	35
5	Extinderea Aplicației pe Platforma Mobilă și Hosting . . . . .	35

# 1 Introducere

Dezvoltarea software nu se rezumă doar la cod; este un ecosistem complex compus din codul sursă, documentație și date de configurare necesare pentru a funcționa optim. În cadrul unei companii de dezvoltare software, interviurile sunt de o importanță crucială, având rolul de a evalua competențele și cunoștințele candidaților. În această lumină, s-a creat o aplicație web menită să faciliteze acest proces în timp real, evitând nevoia unui server intermediar, ceea ce constituie un progres important în eficientizarea procesului de interviu.

Aceasta aplicația web le permite utilizatorilor să redacteze și să ruleze cod în timp real fară o conexiune constantă la un server extern pentru transferul de date, ceea ce face ca aplicația să poată fi scalată la nesfârșit.

Această aplicație nu numai că simplifică procesul de interviu, dar creează și un cadru colaborativ prin care inginerii software pot lucra împreună la diferite proiecte, toate prin intermediul unei conexiuni sigure peer-to-peer. În plus, aplicația încurajează crearea de probleme sau exerciții personalizate, adaptate la cerințele specifice ale interviului.

La baza sa, această aplicație web vizează îmbunătățirea procesului de interviu în companiile de dezvoltare software, facilitând cooperarea între inginerii software și oferind candidaților o experiență superioară. Prin eliminarea necesității unui server intermediar, aplicația nu doar că reduce în mod considerabil costurile asociate, dar și accelerează și eficientizează procesul de interviu, contribuind astfel la un proces de angajare mai productiv și mai eficient, neîngrădit de răspunsuri teoretice sau rezolvarea de probleme care nu sunt relevante pentru activitatea lor cotidiană.

## 1.1 Definiția Problemei

Într-o lume tot mai digitalizată, procesul de recrutare și interviu în domeniul dezvoltării software poate adesea să fie dificil și consumator de timp. Problema principală rezidă în dificultatea evaluării eficiente a candidaților, având în vedere natura complexă a abilităților de programare și a problemelor de rezolvat. De asemenea, limitările tehnologice, cum ar fi dependența de un server extern pentru a facilita comunicarea în timpul interviurilor, pot încetini procesul și pot adăuga costuri suplimentare.

Un alt aspect al problemei este reprezentat de lipsa unui mediu de colaborare în timp real pentru dezvoltatorii de software. Aceasta poate fi o provocare atât pentru candidați, care au nevoie să demonstreze eficient abilitățile lor de rezolvare a problemelor și de programare, cât și pentru interviewatori, care doresc să înțeleagă gândirea

și abordarea candidaților în timp real.

## 1.2 Delimitări și Limitări

Această lucrare de licență se axează pe utilizarea tehnologiei WebRTC pentru a dezvolta o aplicație web ce facilitează interviurile în timp real și oferă un mediu colaborativ pentru dezvoltatori. Totuși, există o serie de delimitări și limitări care trebuie luate în considerare.

### Delimitări:

- Această lucrare se concentrează pe dezvoltarea unei aplicații desktop, nu pe crearea unei versiuni pentru dispozitivele mobile sau alte platforme. Chiar dacă se poate folosi și pe alte dispozitive prin intermediul unui browser, experiența de utilizare este mai slabă pe dispozitive mobile deoarece tastatura ocupa un spațiu prea mare din ecran, ca o estimare aproximativă, tastatura de pe ecranul unui smartphone ocupă, de obicei, aproximativ 30-50% din ecran atunci când este în modul portret (orientare verticală) și aproximativ 30-40% atunci când este în modul peisaj (orientare orizontală).
- Lucrarea nu include aspecte legate de securitatea și criptarea datelor în cadrul tehnologiei WebRTC. Deși acestea sunt probleme importante, ele nu intră în scopul acestei lucrări.

### Limitări:

- Deși tehnologia WebRTC permite o scalabilitate semnificativă, performanța aplicației poate fi influențată de viteza și stabilitatea conexiunii la internet a utilizatorilor.
- În timp ce tehnologia WebRTC elimină necesitatea unui server intermediu pentru schimbul de date, utilizatorii vor avea nevoie de o conexiune inițială la un server pentru a stabili conexiunea peer-to-peer.
- Capacitatea de a rula cod în timp real este dependență de limbajul de programare selectat și de disponibilitatea unui interpreter sau compilator corespunzător pe mașina utilizatorului. Aplicatia folosește un API gratuit cu o capacitate limitată de trimiteri a codului.

## 1.3 Scopul studiului

Scopul acestui studiu este de a identifica informațiile și detaliile relevante pentru documentarea și crearea unei aplicații eficiente ce folosesc API-ului WebRTC.

## 2 Soluții similare

### 2.1 Prezentare soluții similare

Există mai multe aplicații similare, care oferă o modalitate de a realiza interviuri de programare sau de a colabora pe proiecte de software. Iată câteva exemple de aplicații publice:

- CoderPad - o aplicație web care permite interviewatorilor să creeze probleme de programare și să le ofere candidaților pentru a le rezolva în timp real, folosind diferite limbi de programare.
- CodePen - o platformă online care permite utilizatorilor să creeze, să colaboreze și să împărtășească proiecte de web design și de programare.
- Repl.it - o aplicație web care oferă o platformă de programare în browser, cu suport pentru mai multe limbi de programare.
- GitLab - o platformă de colaborare pentru proiecte de software, care oferă instrumente de versionare, testare și livrare continuă.
- HackerRank - o aplicație web care oferă probleme de programare pentru a fi rezolvate de către utilizatori, precum și o platformă de interviuri pentru angajații.

### 2.2 Asemănări dintre soluțiile similare și aplicația mea

Asemănările dintre aplicația mea și alte soluții similare:

- Toate aceste aplicații permit realizarea interviurilor de programare sau colaborarea pe proiecte de software prin intermediul unei conexiuni securizate la internet.
- Toate aceste aplicații oferă o modalitate de a scrie și de a execuționa codul în timp real, fără a fi necesară conexiunea la un server extern.
- Toate aceste aplicații au o interfață intuitivă, care este ușor de utilizat atât pentru interviewatori, cât și pentru candidați.
- Toate aceste aplicații permit personalizarea întrebărilor sau a problemelor de programare în funcție de necesitățile fiecărei companii sau proiecte.

- Toate aceste aplicații pot fi accesate de pe orice dispozitiv conectat la internet, oferind o flexibilitate mare.
- Toate aceste aplicații oferă o experiență mai bună pentru candidați, deoarece le permit să demonstreze abilitățile lor de programare într-un mod mai natural.

## 2.3 Deosebiri dintre soluțiile similare și aplicația mea

Deosebirile dintre aplicația mea și alte soluții similare:

- Aceasta aplicație oferă un nivel mai mare de confidențialitate, deoarece nu are nevoie de conexiunea la un server intermediar pentru a funcționa, astfel transferul de date/audio/video este mai sigur. Alte aplicații implică transferul de date prin intermediul unui server, ceea ce poate ridica probleme privind confidențialitatea
- De asemenea, din cauza faptului că aplicația mea nu are nevoie de un server intermediar pentru funcționare, acesta oferă mai multă fiabilitate în cazul în care un server extern ar fi supraîncarcat. Astfel, interviurile sau colaborarea pe proiecte de software pot continua fără întrerupere, chiar dacă serverul extern are probleme.
- În plus, fără nevoie de conectarea la un server intermediar, aplicația mea poate oferi o scalabilitate mai bună. Astfel, aplicația poate fi folosită de mai mulți utilizatori fără a se compromite performanță sau fără a fi nevoie de adăugarea de resurse suplimentare. aplicație
- Aceasta aplicație oferă o experiență mai bună pentru interviewatori și candidați, deoarece oferă funcționalități suplimentare, precum apeluri video și audio, care pot face interviul mai interactiv și mai personal.
- Aceasta aplicație oferă instrumente avansate pentru realizarea interviurilor, precum posibilitatea de a testa codul candidaților în mai multe limbi de programare și de a vedea rezultatele în timp real. Alte aplicații au funcționalități mai limitate în acest sens.

# 3 Despre WebRTC

WebRTC (Web Real-Time Communication) este o tehnologie open-source care permite comunicarea în timp real (RTC) între browsere chiar și dispozitive mobile, fără a necesita plugin-uri sau aplicații adiționale. În această secțiune se vor explica principalele componente ale WebRTC-ului, mecanismele de semnalizare, protocoalele de rețea utilizate și aspecte legate de securitate și confidențialitate. De asemenea, se discută provocările și perspectivele de viitor în dezvoltarea acestei tehnologii în domeniul comunicațiilor pe internet.

În ultimul deceniu, comunicarea în timp real pe internet a devenit o parte esențială a vieții noastre de zi cu zi. WebRTC este o tehnologie care a transformat modul în care interacționăm în mediul online prin browsere, permitându-ne să comunicăm prin video, audio și să partajăm date, în timp real, direct din browser.

## 3.1 Arhitectura WebRTC

În esență, arhitectura WebRTC poate fi împărțită în două straturi distincte care interacționează pentru a permite comunicarea în timp real. Primul strat este destinat dezvoltatorilor de browsere web și implică API-ul nativ C++ WebRTC și hook-urile de captură/redare. Al doilea strat este destinat dezvoltatorilor de aplicații web și implică API-ul Web.

În figura 3.1, se vede cum aceste componente se aliniază în cadrul arhitecturii generale WebRTC și cum interacționează pentru a permite funcționalitatea de comunicații în timp real. Fiecare componentă joacă un rol esențial în gestionarea și optimizarea fluxului de date multimedia, fie că este vorba de audio sau video, de la sursă la destinație și înapoi.

### Stratul dezvoltatorilor de aplicații web:

Web API: Este un API destinat dezvoltatorilor terți pentru a dezvolta aplicații web asemănătoare cu videochat-ul. Prin intermediul acestui API, dezvoltatorii pot accesa și controla funcționalitățile de comunicații în timp real oferite de WebRTC.

### Stratul dezvoltatorilor de browser:

WebRTC Native C++ API: Aceasta este un strat API care permite dezvoltatorilor de browsere să implementeze cu ușurință API-ul WebRTC. Oferă funcționalități esențiale pentru gestionarea comunicațiilor în timp real.

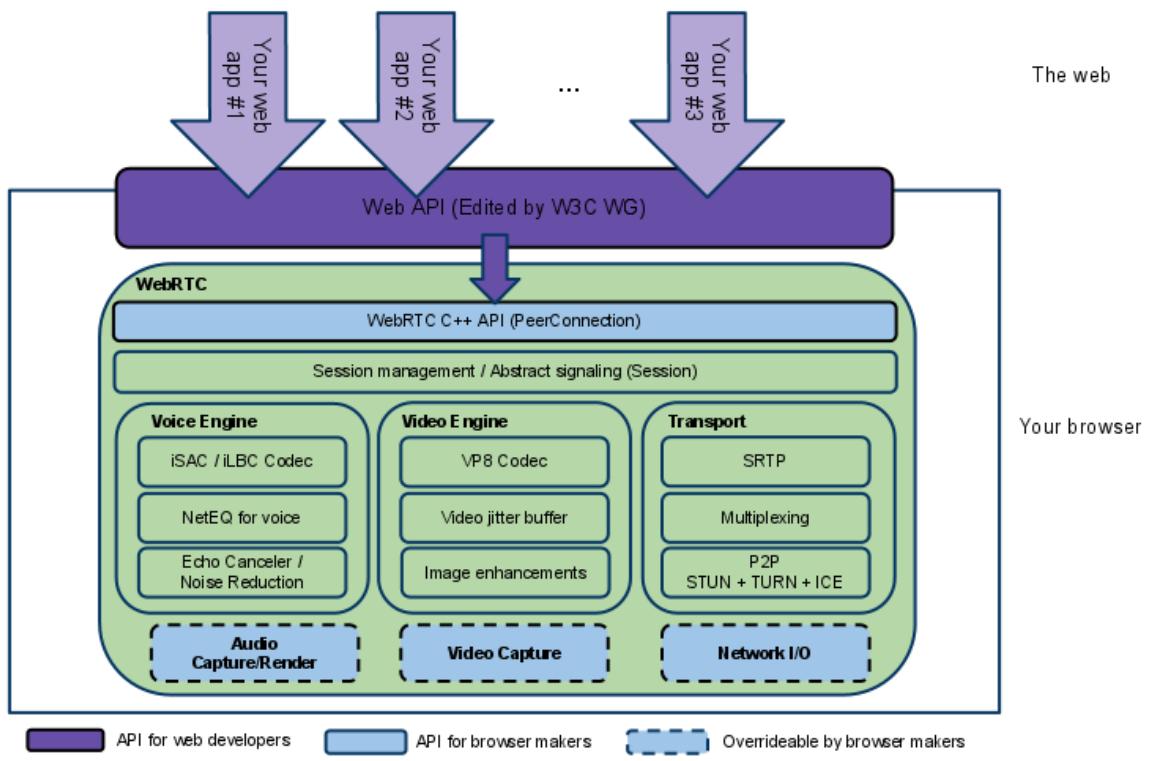


Figura 3.1: Arhitectura WebRTC, conform (9)

## 3.2 WebRTC Protocol Stack

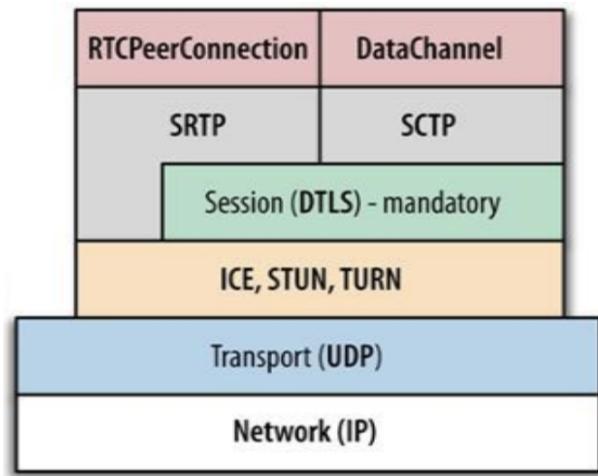


Figura 3.2: WebRTC protocol stack, conform (4)

Iată o explicație a fiecărei părți din stiva de protocoale WebRTC, din figura 3.2

**RTCPeerConnection:** RTCPeerConnection este interfață principală a API-ului WebRTC care se ocupă de stabilirea și gestionarea conexiunilor peer-to-peer. Aceasta gestionează negocierile de sesiune, stabilirea conexiunii și schimbul de date între perechi. RTCPeerConnection se bazează pe protocoale precum ICE, STUN și TURN pentru a descoperi adresele IP ale perechilor și pentru a stabili o conexiune chiar și în cazul în care există obstacole precum NAT și firewall-uri.

**RTCDataChannel:** RTCDataChannel este o interfață a API-ului WebRTC care permite transmiterea bidirectională de date între perechi pe o conexiune peer-to-peer stabilită prin RTCPeerConnection. RTCDataChannel suportă diferite tipuri de date, precum text, binar și tipuri de date personalizate, și este utilizat pentru a transmite date care nu sunt parte a fluxurilor media, cum ar fi mesaje text și fișiere.

**SRTP (Secure Real-time Transport Protocol):** SRTP este o extensie a protocolului RTP (Real-time Transport Protocol) care adaugă criptare, autentificare și integritate pentru a proteja datele media în tranzit. SRTP este utilizat în WebRTC pentru a asigura securitatea datelor media (audio și video) între perechi și pentru a preveni interceptarea sau alterarea acestora.

**SCTP (Stream Control Transmission Protocol):** SCTP este un protocol de transport care oferă un mecanism fiabil și ordonat pentru transmiterea datelor între perechi. În contextul WebRTC, SCTP este utilizat pentru a transmite date prin RTCDataChannel, asigurându-se că datele sunt livrate în ordinea corectă și fără pierderi sau corupție. SCTP este transportat peste DTLS (Datagram Transport Layer Security) pentru a asigura criptarea și integritatea datelor.

Aceste componente ale stivei de protocoale WebRTC lucrează împreună pentru a asigura comunicații sigure, eficiente și în timp real între perechi, permitând realizarea apelurilor video și audio, precum și partajarea de date direct din browser, fără a necesita plugin-uri sau aplicații adiționale.

### 3.3 Conexiunea între doi useri

În WebRTC, stabilirea unei conexiuni între doi utilizatori implică un proces numit "semnalizare" (signaling). Semnalizarea este utilizată pentru a facilita comunicarea între utilizatori înainte de stabilirea unei conexiuni peer-to-peer directe. Un server de semnalizare este un server intermediar care ajută la coordonarea și schimbul de informații între perechi în timpul procesului de semnalizare.

Iată o descriere pas-cu-pas a modului în care se stabilește conexiunea între doi utilizatori cu ajutorul unui server de semnalizare:

Ambii utilizatori se conectează la serverul de semnalizare: Utilizatorii se conectează la serverul de semnalizare prin intermediul unui protocol de transport, cum ar fi WebSocket sau HTTP. Acest server va facilita comunicarea între utilizatori în timpul procesului de semnalizare.

Schimbarea informațiilor despre sesiune: Utilizatorii schimbă informații despre sesiune, cum ar fi tipurile de codec-uri acceptate, constrângerile de lățime de bandă și

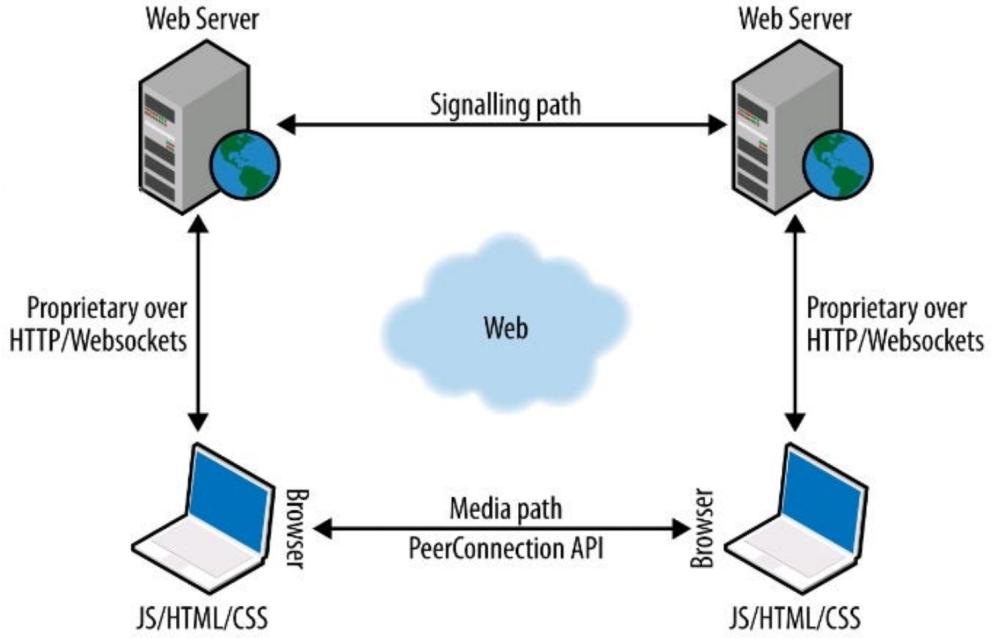


Figura 3.3: WebRTC signaling server, conform (4)

adresele de rețea. Aceste informații sunt transmise într-un format standard numit SDP (Session Description Protocol).

Descoperirea adreselor de rețea și stabilirea unei conexiuni: Utilizatorii descoperă adresele IP ale perechilor și stabilesc o conexiune peer-to-peer, chiar și în cazul în care există NAT-uri și firewall-uri. Acest proces implică utilizarea unor protocoale precum ICE (Interactive Connectivity Establishment), STUN (Session Traversal Utilities for NAT) și TURN (Traversal Using Relays around NAT).

Negocierea conexiunii: După ce adresele de rețea sunt descoperite, utilizatorii negociază și stabilesc o conexiune peer-to-peer. Acest proces implică schimbul de informații criptografice pentru a asigura securitatea comunicației și configurarea protocolului de transport (de exemplu, SRTP pentru datele media și SCTP pentru datele de aplicație).

Începerea comunicației: Odată ce conexiunea peer-to-peer este stabilită, utilizatorii pot începe să comunice unul cu celălalt prin trimiterea de date media (audio și video), conform (5), și de aplicație (de exemplu, prin RTCDATAChannel).

Este important de menționat că serverul de semnalizare nu este implicat în comunicarea directă între utilizatori odată ce conexiunea peer-to-peer este stabilită. Rolul său principal este să ajute la coordonarea și schimbul de informații în timpul procesului de semnalizare pentru a facilita stabilirea conexiunii între perechi.

# 4

# Descrierea aplicație

Se vor analiza în această secțiune trei aspecte ale aplicației:

Arhitectura Aplicației Simplificate: Se prezinta o imagine de ansamblu a modului în care funcționează aplicația și cum interacționează componentele sale.

Facilitățile Generale ale Aplicației: Trecem prin fiecare funcționalitate principală a aplicației, de la scrierea codului la compilarea în timp real al lui.

Cazurile de Utilizare ale Aplicației: Se explorează fiecare scenariu specific în care aplicația poate fi utilizată, și se trece în detaliu despre cum funcționează fiecare caz de utilizare

## 4.1 Arhitectură Aplicație

În această secțiune, vom prezenta o versiune simplificată a arhitecturii aplicației pentru a oferi o înțelegere de bază a modului în care funcționează aceasta

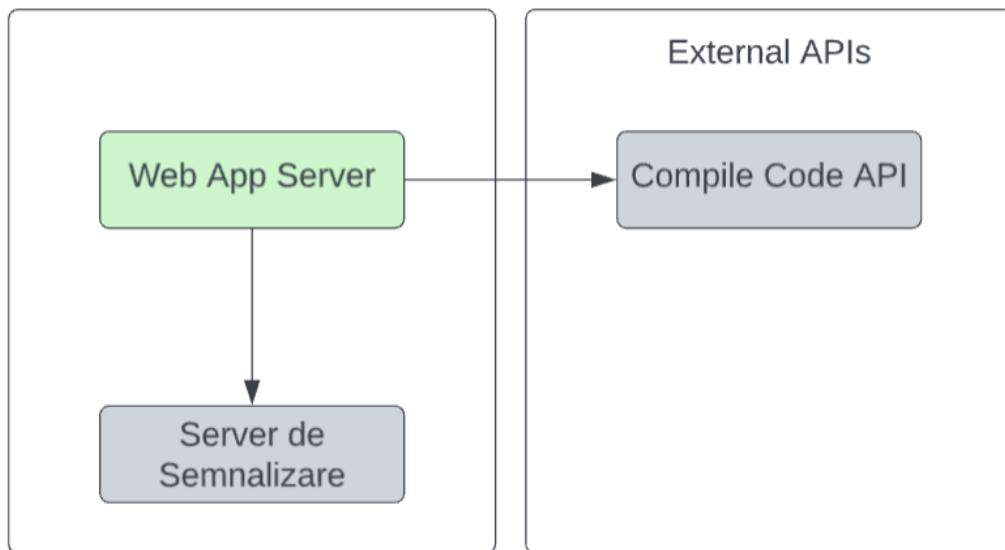


Figura 4.1: Arhitectura Aplicației

**Web App Server:** Această este componenta principală a aplicației, este partea unde utilizatorii interacționează cu front-endul. De asemenea, folosește un API ex-

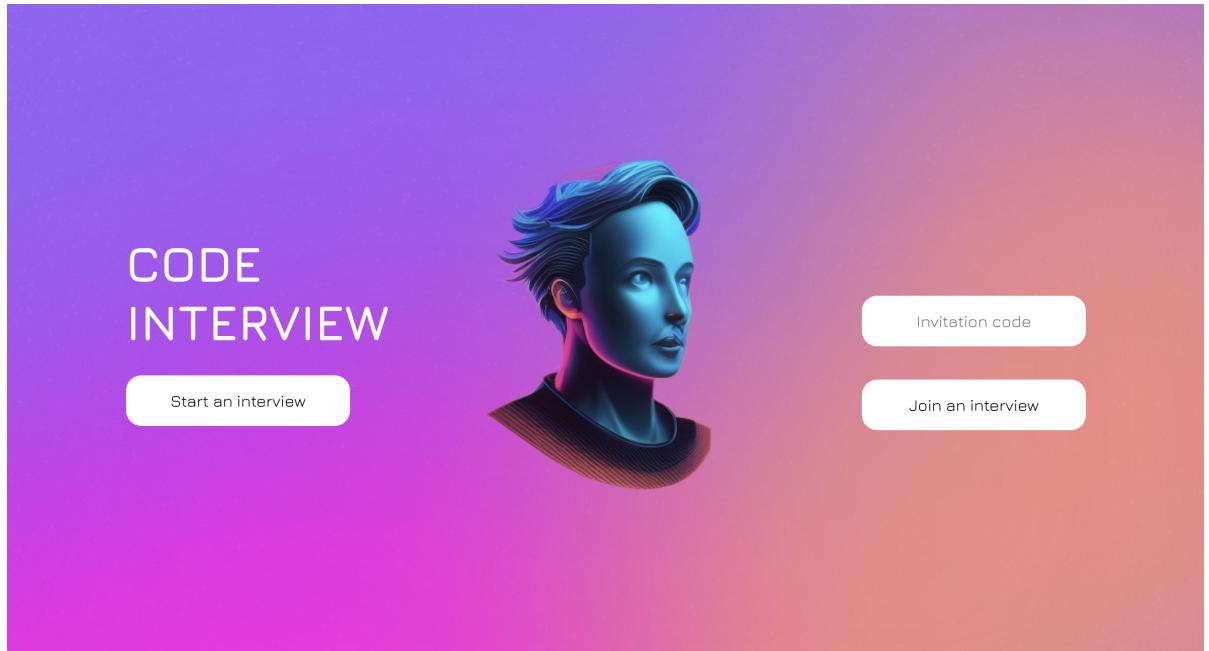
tern, preluat din (3) pentru a compila codul sursa scris. Asta înseamnă că poate face request-uri fără a pune prea multă presiune pe server. Mai mult, face ca lucrurile să fie ușoare pentru utilizatori, permitându-le să scrie, să modifice și să distribuie cod prin conexiunea peer-to-peer.

**Serverul de Semnalizare:** Acest server are o sarcină importantă - ajută la crearea conexiunii între utilizatori. Folosește WebRTC pentru a permite schimbul de informații între utilizatori, astfel încât să poată colabora.

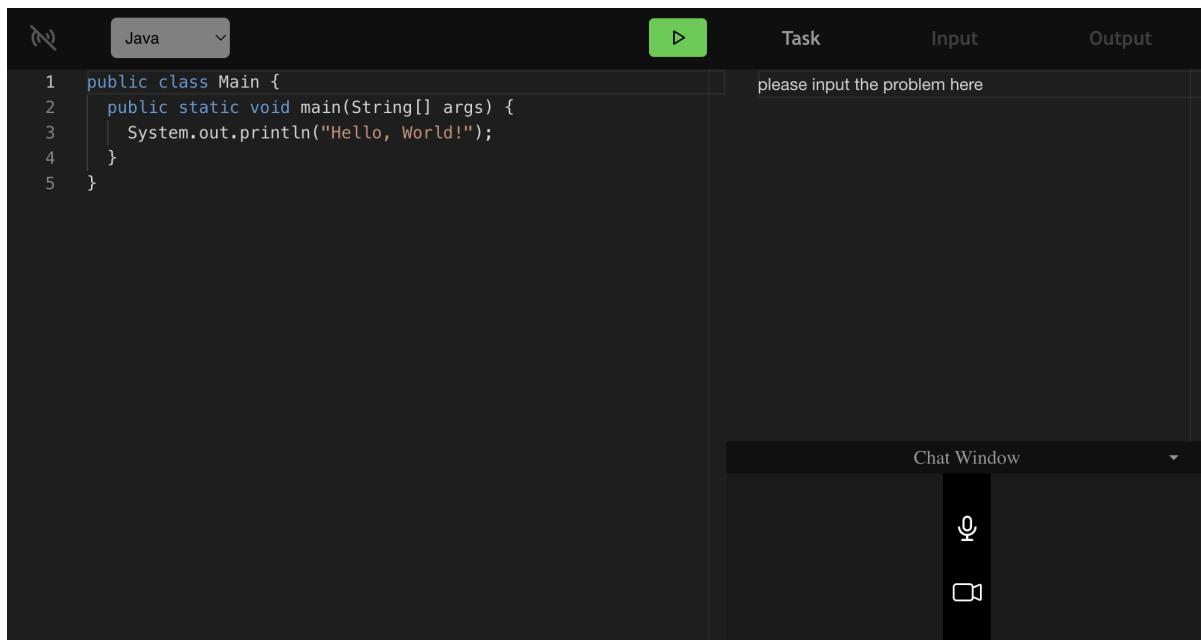
**API-ul Extern de Compilare a Codului:** Acest API este o resursă exterioară aplicației. Ajută aplicația să poată compila cod în diferite limbi, fără a avea nevoie de resurse mari pe propriul server. Datorită acestui API, aplicația poate oferi mai multe opțiuni de scriere de cod, utilizatorilor. Acest lucru înseamnă că utilizatorii nu sunt forțați să lucreze într-un anumit limbaj de programare și pot alege ce li se potrivește cel mai bine.

## 4.2 Facilități Aplicație

În cadrul aplicației dezvoltate, s-a implementat o serie de facilități care îmbunătățesc experiența utilizatorilor și facilitează procesul de interviu și colaborare în timp real. Aceste facilități sunt menite să ofere un mediu interactiv și eficient pentru intrevievatori și candidați. Mai jos sunt prezentate în detaliu fiecare facilitate a aplicației alături de o poză cu cele două ecrane ale aplicației:



Interfata utilizatorului în pagina de landing



Interfata utilizatorului în pagina de interviu

#### 4.2.1 Scriere și Execuție Cod în Timp Real

Aplicația oferă posibilitatea utilizatorilor de a scrie și execuția codului în timp real, fără a fi necesară o conexiune constantă la un server extern. Această funcționalitate facilitează interacțiunea în timp real între interviewatori și candidați, permitându-le să colaboreze în timpul interviului sau să rezolve probleme de programare împreună. Utilizatorii pot selecta limbajul de programare dorit și pot începe să scrie codul direct în editorul de text integrat. Codul poate fi apoi rulat în cadrul aplicației, iar rezultatele vor fi afișate în consola aplicației. Această funcționalitate permite interviewatorilor să evaluateze abilitățile de programare ale candidaților în timp real și să ofere feedback imediat.

#### 4.2.2 Colaborare în Timp Real

Aplicația oferă un mediu de colaborare în timp real, prin intermediul unei conexiuni securizate peer-to-peer. Utilizatorii pot lucra împreună la aceeași problemă sau proiect de programare, permitându-le să editeze și să partajeze cod în timp real. Schimbările efectuate de un utilizator sunt reflectate imediat pe ecranul celuilalt utilizator, facilitând colaborarea și comunicarea eficientă între interviewatori și candidați.

#### 4.2.3 Partajare de Probleme și Întrebări de Programare

Aplicația permite interviewatorilor să creeze și să partajeze probleme sau întrebări de programare către candidați. Acestea pot fi personalizate în funcție de nevoile specifice ale interviului și pot fi scrise într-un limbaj de programare specific. Candidații pot accesa aceste probleme în timp real și pot răspunde la ele prin scrierea și executarea codului corespunzător. Această funcționalitate facilitează procesul de evaluare a abilităților de programare ale candidaților și permite interviewatorilor să evaluateze și să compare rezultatele obținute.

#### **4.2.4 Apeluri Video și Audio**

Aplicația oferă posibilitatea utilizatorilor de a efectua apeluri video și audio în timpul interviului sau colaborării. Această facilitate facilitează comunicarea între intervievaitori și candidați, permitându-le să se vadă și să se audă în timp real. Utilizatorii pot activa sau dezactiva microfonul și camera video în funcție de preferințele lor. Aceasta adaugă un element de interactivitate și personalizare în procesul de interviu și facilitează comunicarea verbală în timpul rezolvării problemelor de programare sau a colaborării pe proiecte de software.

Protocolul WebRTC utilizează un sistem de criptare end-to-end pentru a asigura confidențialitatea și securitatea datelor transmise între dispozitive. În mod normal, transmisia de date multimedia prin WebRTC este permisă numai atunci când conexiunea este securizată prin HTTPS.

Prin urmare, pentru a asigura funcționarea corectă a transmisiei video-audio, este necesară implementarea HTTPS securizat pentru atât partea de client-side, cât și partea de server-side a aplicației. Aceasta implică obținerea și implementarea unui certificat SSL (Secure Sockets Layer) pentru a securiza conexiunea. Certificatul SSL garantează autenticitatea și integritatea datelor transmise și asigură că conexiunea este criptată în mod corespunzător.

### **4.3 Cazuri de utilizare aplicație**

#### **4.3.1 Diagrama cazurilor de utilizare**

În această variantă, intervievatorul și candidatul sunt reprezentați că fiind două entități diferite

În cazul aplicației, intervievatorul și candidatul au aceleasi acțiuni (cum ar fi accesarea întrebărilor sau problemele de programare, scrierea de cod și transmiterea răspunsurilor în timp real), dar au roluri diferite (intrevievatorul este cel care face întrebările, în timp ce candidatul este cel care le răspunde).

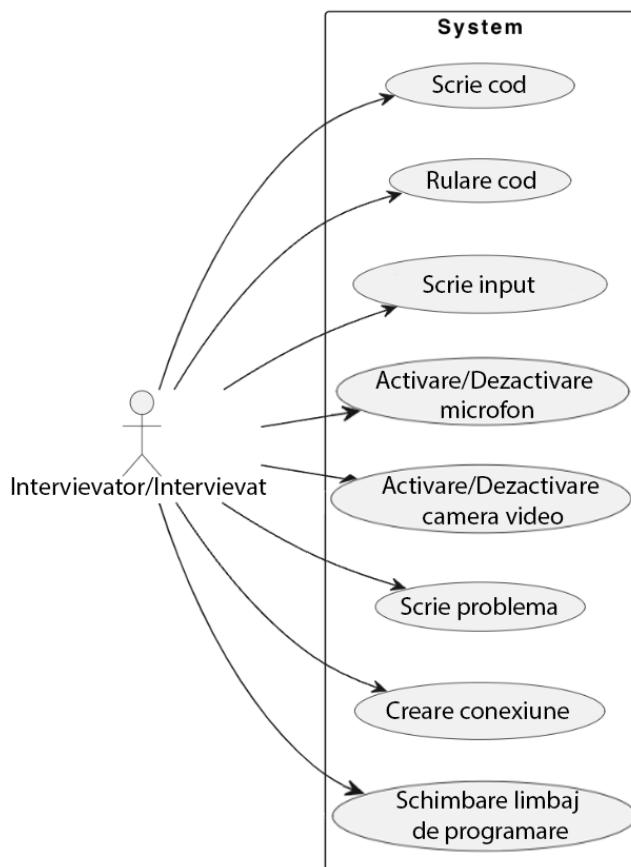


Figura 3.1

#### 4.3.2 Detalii caz de utilizare ”Scrie cod”

Nume UC: Scrie cod

Actor: Interviewator, Candidat

Descriere: Actorul are posibilitatea să scrie și să partajeze codul către celălalt actor conectat

Precondiții: Utilizatorul trebuie să fie conectat cu celălalt utilizator pentru a putea partajara codul

Postconditii: Sistemul partajeaza codul scris de actor

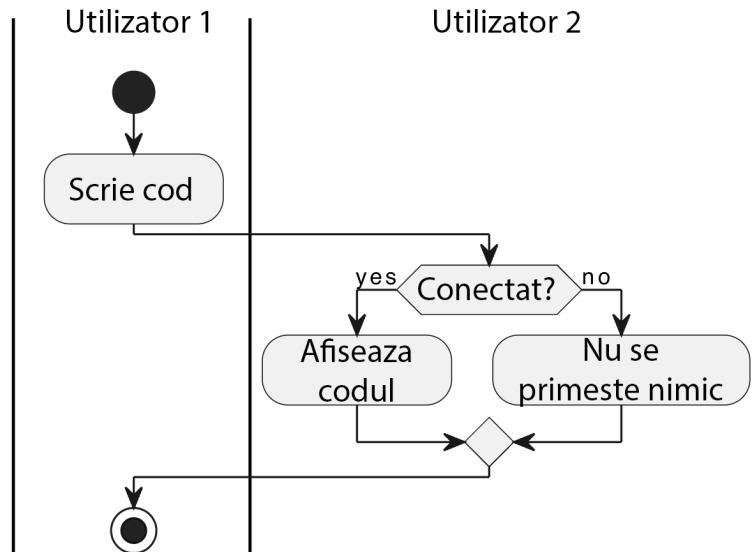


Figura 3.2

#### 4.3.3 Detalii caz de utilizare "Creare conexiune"

Nume UC: Creare conexiune

Actor: Intervievator, Candidat

Descriere: Actorul crează o conexiune automat cu celălalt actor

Precondiții: Ambii actori trebuie să fie pe aceleași link de camera

Postconditii: O conexiune se realizează între cei doi actori fără un server în spate să mențină conexiunea

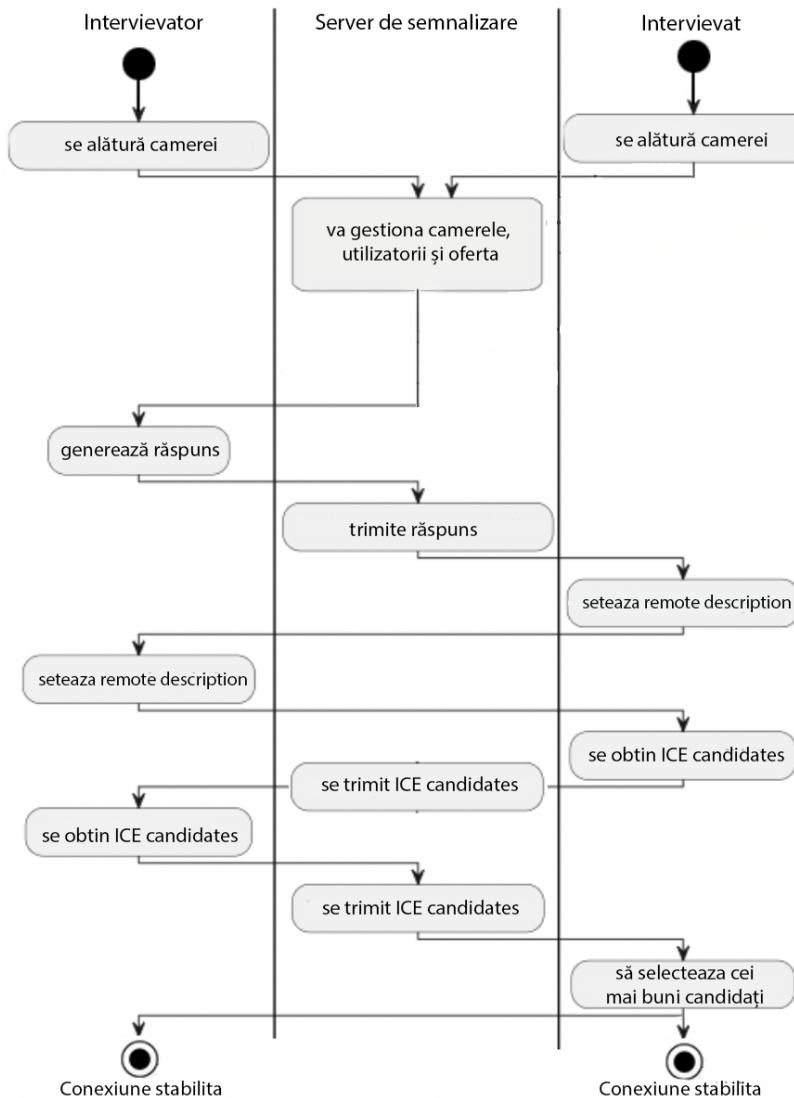


Figura 3.3

#### 4.3.4 Detalii caz de utilizare "Scrie problema"

Nume UC: Scrisie problema

Actor: Interviewator, Candidat

Descriere: Interviewatorul are posibilitatea să scrie și să partajeze o problema către candidatul conectat

Precondiții: Utilizatorul trebuie să fie conectat cu celălalt utilizator pentru a putea partajara problema

Postconditii: Sistemul partajeaza problema scrisa de interviewator

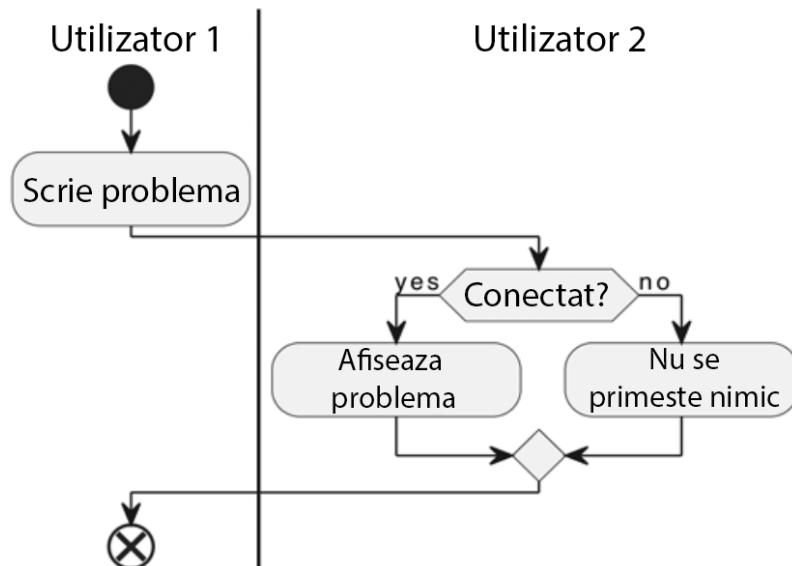


Figura 3.4

#### 4.3.5 Detalii caz de utilizare "Rulare cod"

Nume UC: Rulare cod

Actor: Intervievator, Candidat

Descriere: Actorul are posibilitatea ruleze codul scris sau primit în timp real

Precondiții: Actorul trebuie aiba limbajul în care scrie codul selectat

Postconditii: Sistemul afiseaza în consola rezultatul codului rulat

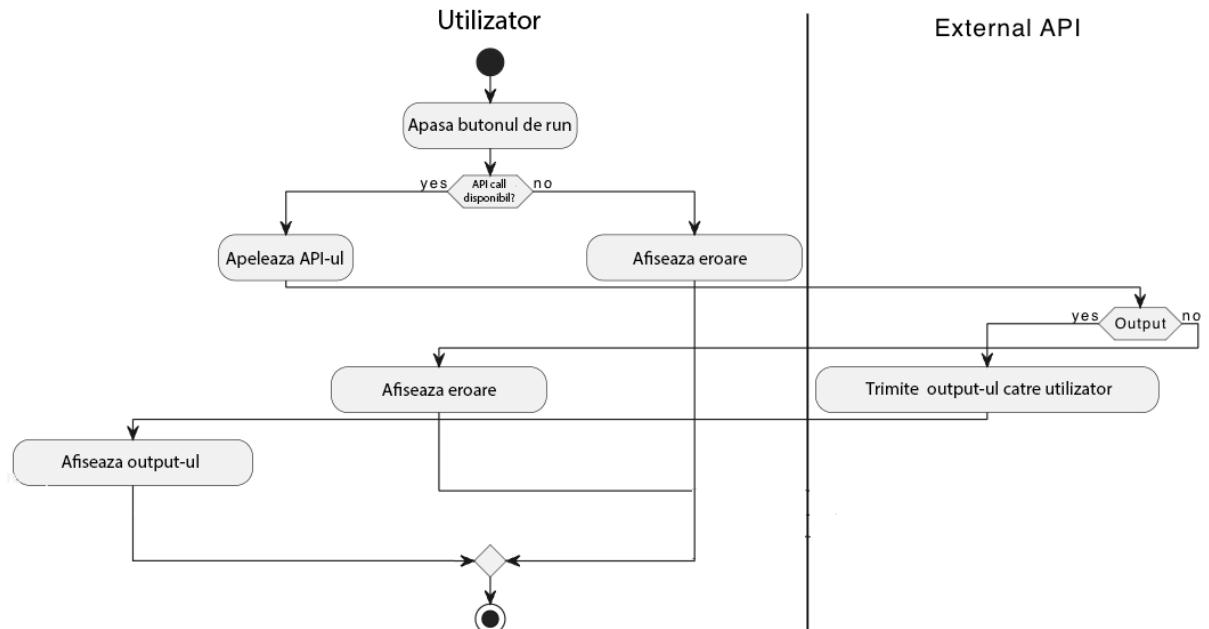


Figura 3.5

#### 4.3.6 Detalii caz de utilizare "Scrie input"

Nume UC: Scrie input

Actor: Intervievator, Candidat

Descriere: User-ul are posibilitatea să scrie și să partajeze o input-ul către candidatul conectat

Precondiții: Utilizatorul trebuie să fie conectat cu celălat utilizator pentru a putea partajara inputul

Postconditii: Sistemul partajeaza inputul pentru a fi ulterior folosit cand se ruleaza codul

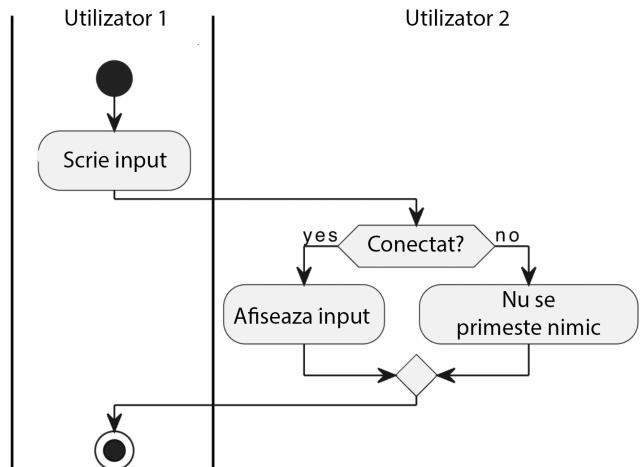


Figura 3.6

#### 4.3.7 Detalii caz de utilizare ”Activare/Dezactivare microfon”

Nume UC: Dezactivare/Activare microfon

Actor: Interviewator, Candidat

Descriere: Actorul are posibilitatea de a dezactiva microfonul pentru a opri transmisia audio

Precondiții: Microfonul actorului trebuie să fie activ

Postconditii: Microfonul actorului este dezactivat/activat, oprirea transmisiei audio

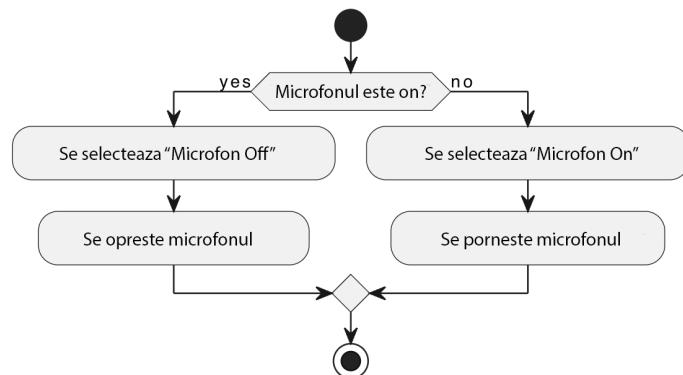


Figura 3.7

#### 4.3.8 Detalii caz de utilizare ”Activare/Dezactivare camera video”

Nume UC: Activare/Dezactivare camera video

Actor: Intervievator, Candidat

Descriere: Actorul are posibilitatea de a închide/deschide camera video pentru a opri/porni transmisia video

Precondiții: Camera video a actorului trebuie să fie activă

Postconditii: Camera video a actorului este închisă, oprirea transmisiei video

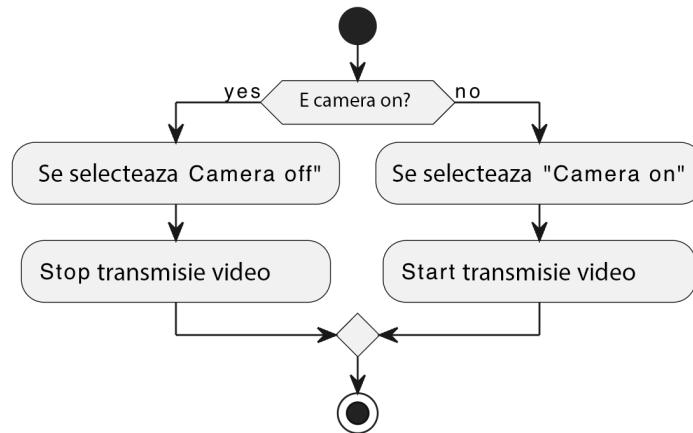


Figura 3.8

#### 4.3.9 Detalii caz de utilizare ”Schimbare limbaj de programare”

Nume UC: Schimbare limbaj de programare

Actor: Intervievator, Candidat

Descriere: Actorul are posibilitatea de a schimba limbajul de programare utilizat pentru a scrie cod

Precondiții: Actorul trebuie să aibă un limbaj de programare selectat

Postconditii: Sistemul își schimbă limbajul de programare pe baza selecției actorului

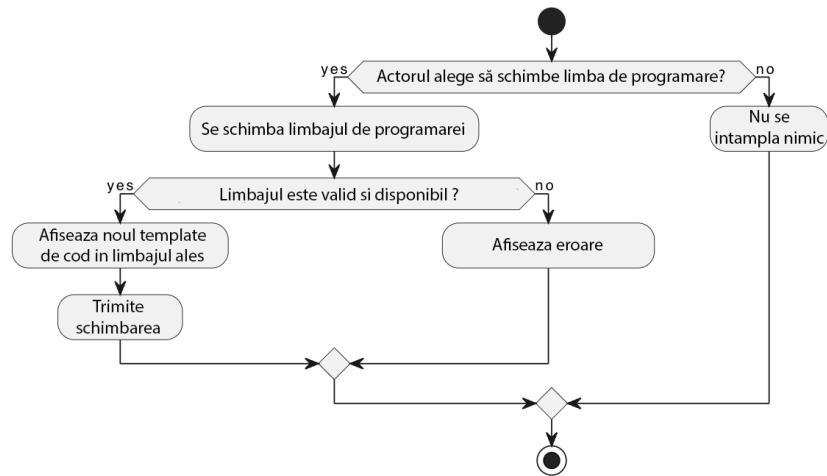


Figura 3.9

# 5 Detalii de implementare

## 5.1 Conexiunea dintre doi useri

Conform figurii 5.1, conexiunea dintre doi utilizatori se face cu ajutorul unui server de semnalizare, procesul are loc și în partea serverului dar și în partea front-end-ului.

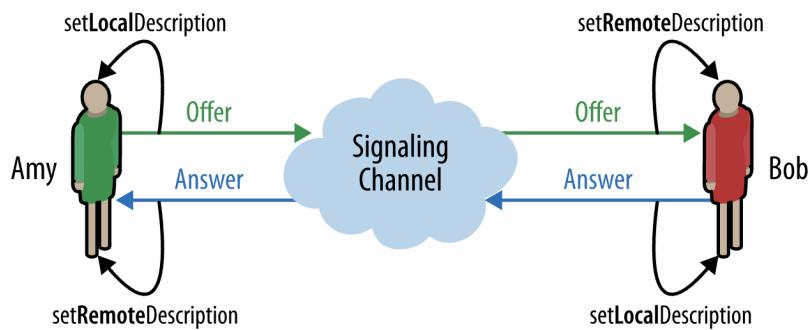


Figura 5.1: WebRTC signaling, conform (4)

### 5.1.1 Server side

**Inițializarea serverului:** Serverul de semnalizare este inițializat pentru a asculta conexiuni pe portul 8080. Se creează un obiect Map pentru a urmări camerele și clienții conectați în camere.

**Conecțarea unui client:** Când un client se conectează la server, se verifică dacă camera în care dorește să intre are deja doi clienți. Dacă sunt prea mulți clienți în cameră, conexiunea în plus va fi închisă, altfel clientul este adăugat în camera respectivă.

**Procesul de semnalizare:** Serverul de semnalizare gestionează mesajele între clienții care doresc să stabilească conexiuni peer-to-peer. Acest proces implică mesaje de tip "join", "offer", "answer" și "candidate", care sunt retransmise între clienții din aceeași cameră, pentru a facilita stabilirea conexiunii peer-to-peer.

1. Clientul inițial se alătură unei camere și așteaptă ca un al doilea client să se alăture.

2. Al doilea client se alătură camerei și serverul de semnalizare trimite un mesaj de tip "initiate\_offer" către initiator.
3. Initiatorul creează și trimite o ofertă SDP serverului de semnalizare, care apoi o retransmite către celălalt client.
4. Celălalt client răspunde cu un răspuns SDP, pe care serverul de semnalizare îl retransmite către initiator.
5. Ambii clienți schimbă candidați ICE prin intermediul serverului de semnalizare, pentru a descoperi adresele IP ale perechilor și a stabili o conexiune peer-to-peer.

**Stabilirea conexiunii peer-to-peer:** Odată ce procesul de semnalizare este complet, clienții pot comunica direct între ei prin conexiunea peer-to-peer stabilită.

**Deconectarea unui client:** Atunci când un client se deconectează, serverul de semnalizare îl elimină din cameră și resetează starea clientului rămas pentru a aștepta un nou utilizator să se alăture camerei.

### 5.1.2 Client side

**Extrage ID-ul camerei:** Se extrage ID-ul camerei al utilizatorui din URL-ul paginii pentru a se asigura că se conectează la camera corectă.

**Conecțarea la serverul de semnalizare:** Utilizatorul se conectează la serverul de semnalizare folosind WebSocket și trimite un mesaj de tip "join" pentru a se alătura camerei.

**Crearea conexiunii RTC:** Se creează un obiect RTCPeerConnection fiecarui utilizator, conform (1) și (2), care va fi folosit pentru a stabili conexiunea peer-to-peer cu celălalt client. De asemenea, se creează un canal de date pentru a comunica informațiile între clienți.

**Inițierea procesului de semnalizare:** În funcție de mesajele primite de la serverul de semnalizare, utilizatorul va iniția procesul de semnalizare pentru a stabili conexiunea peer-to-peer. Aceasta implică trimiterea unei oferte SDP, primirea și trimiterea răspunsului SDP și schimbul de candidați ICE.

**Stabilirea conexiunii peer-to-peer:** Odată ce procesul de semnalizare este complet, conexiunea peer-to-peer este stabilită, iar clienții pot comunica direct între ei prin canalul de date.

**Procesarea mesajelor primite:** Utilizatorul procesează mesajele primite de la serverul de semnalizare și actualizează starea locală a aplicației în funcție de mesajele primite. De exemplu, dacă primesc un mesaj de tip "offer", "answer" sau "candidate", vor procesa și aplica informațiile corespunzătoare în obiectul RTCPeerConnection.

**Trimiterea și primirea datelor prin canalul de date:** După ce conexiunea peer-to-peer este stabilită, utilizatorii pot comunica între ei prin canalul de date. Aplicația ascultă mesajele primite pe canalul de date și actualizează starea aplicației în consecință.

**Deconectarea:** Când un utilizator se deconectează, serverul de semnalizare se ocupă de deconectare și actualizează starea celorlalți utilizatori care așteaptă un nou utilizator să se alăture camerei.

## 5.2 Tehnologii utilizate în dezvoltarea client-side

### 5.2.1 React

In partea de dezvoltare a interfeței s-a folosit React, din (8). React este o bibliotecă JavaScript dezvoltată și întreținută de Facebook, care permite crearea de interfețe utilizator scalabile și eficiente pentru aplicații web și mobile. Se va documenta arhitectura React, componentele sale, paradigma de programare declarativă, sistemul de reconciliere și performanța. De asemenea, se discută avantajele utilizării React în dezvoltarea frontend și modul în care acesta se integrează cu alte tehnologii și biblioteci.

**Arhitectura React** Arhitectura React este bazată pe conceptul de "componente". O componentă este o unitate modulară și reutilizabilă de cod care reprezintă o parte a interfeței utilizator. React utilizează un sistem de "arbore de componente" pentru a organiza și a gestiona interacțiunea dintre aceste componente într-o aplicație.

**Componentele React** Componentele React pot fi definite ca funcții sau clase și reprezintă elemente de bază ale interfeței utilizator. Fiecare componentă are un anumit "stat" (state) și "proprietăți" (props). State-ul unei componente reprezintă datele interne pe care componenta le gestionează, în timp ce proprietățile sunt datele primite de la componentele părinte.

**Componente funcționale:** Componentele funcționale sunt cele mai simple tipuri de componente React și sunt definite ca funcții JavaScript. Acestea sunt de obicei utilizate pentru a construi componente "fără stare" (stateless) care depind exclusiv de proprietăți pentru a-și afișa conținutul.

**Componente de clasă:** Componentele de clasă sunt definite ca clase JavaScript și permit utilizarea unui state intern și a unor metode de ciclu de viață mai complexe. Aceste componente sunt utilizate pentru a construi componente "cu stare" (stateful), care au nevoie să gestioneze date interne și să răspundă la evenimente din ciclul de viață al componentei.

**Paradigma de programare declarativă** React utilizează o paradigma de programare declarativă, în care dezvoltatorul descrie "ce" trebuie să se întâmple în loc să specifică "cum" să se întâmple. Acest lucru îmbunătățește lizibilitatea și predictibilitatea codului, deoarece componentele React descriu aspectul și comportamentul

interfeței utilizator, în loc să se concentreze pe manipularea directă a DOM-ului (Document Object Model).

**Reconcilierea și performanța** Unul dintre aspectele cheie ale React este sistemul său de reconciliere, care se ocupă de actualizarea eficientă a interfeței utilizator în urma modificărilor de stare sau proprietăți ale componentelor. React utilizează un algoritm diferențial pentru a compara ”arborele virtual” al componentelor cu arborele DOM real și a determina modificările necesare în interfața utilizator. Această abordare, numită ”DOM virtual”, reduce numărul de operațiuni costisitoare de manipulare a DOM-ului și îmbunătățește performanța aplicației.

**Avantajele utilizării React** Utilizarea React în dezvoltarea frontend oferă mai multe avantaje, printre care:

**Reutilizarea componentelor:** Abordarea modulară a React permite dezvoltatorilor să creeze componente reutilizabile, reducând astfel duplicarea codului și îmbunătățind productivitatea.

**Performanță îmbunătățită:** Utilizarea DOM-ului virtual și a algoritmului de reconciliere asigură o performanță excelentă, chiar și în cazul aplicațiilor cu un număr mare de componente și actualizări frecvente ale interfeței utilizator.

**Comunitatea largă și ecosistemul:** React se bucură de o comunitate vastă de dezvoltatori și un ecosistem bogat de biblioteci și instrumente suplimentare, care facilitează dezvoltarea și îmbunătățirea aplicațiilor.

### 5.2.2 Componente externe utilizate

În realizarea aplicației, s-a utilizat o componentă externă numită ”Monaco Editor” preluată din (6). Monaco Editor este un editor de cod open-source dezvoltat de Microsoft care oferă funcționalități de editare avansate, cum ar fi evidențierea sintaxei, auto-completare, navigare rapidă, și multe altele. A fost utilizat pentru a îmbunătăți experiența utilizatorului în ceea ce privește editarea și vizualizarea codului.

Monaco Editor a fost integrat în patru componente principale ale interfeței:

**Editorul Principal:** Aceasta este locul unde candidatul își scrie codul pentru a rezolva problema propusă. Funcțiile avansate de editare oferite de Monaco Editor ajută la scrierea eficientă și precisă a codului.

**Editorul Inputului:** Aceasta este locul unde utilizatorul introduce un input care va fi preluat pentru de codul sursa scris pentru compilare.

**Editorul Outputului:** Aici este afișat outputul generat de codul compilat. Monaco Editor ajută la prezentarea acestui output într-un mod ușor de citit și de înțeles.

**Editorul Task-ului:** În această secțiune, interviewatorul poate introduce problema

pe care candidatul trebuie să o rezolve. Monaco Editor asigură o scriere și o citire ușoară a textului.

## 5.3 Tehnologii utilizate în dezvoltarea server-side

### Node.js

Conform informațiilor oferite, NodeJS (7), este o platformă de rulare a codului JavaScript în afara browserului, bazată pe motorul JavaScript V8 al Google Chrome. Această platformă îmi permite dezvoltarea server-side și rețea al aplicației, oferind o performanță excelentă și o scalabilitate ridicată. Se va discuta despre arhitectura Node.js, modelul său de execuție bazat pe evenimente, facilitățile oferite de modulul său de gestionare a pachetelor, NPM, și integrarea cu alte tehnologii și biblioteci.

**Arhitectura Node.js** Node.js se bazează pe motorul JavaScript V8 și pe librăria libuv, care oferă facilități de intrare/ieșire non-blocantă și o buclă de evenimente (event loop) eficientă. Această arhitectură permite dezvoltarea de aplicații care pot procesa un număr mare de conexiuni simultane, fără a avea nevoie de resurse semnificative de sistem.

**Modelul de execuție bazat pe evenimente** În Node.js, execuția codului este organizată în jurul unei bucle de evenimente, care procesează și trimit evenimente într-o manieră non-blocantă. Acest model asincron permite aplicațiilor să gestioneze o cantitate mare de operațiuni de intrare/ieșire în paralel, fără a bloca execuția sau a aștepta finalizarea acestora.

**Call Stack:** Call stack-ul în Node.js este un mecanism care ține evidența funcțiilor care se execută într-un moment dat. Când o funcție se termină, aceasta este eliminată din call stack, iar execuția continuă cu funcția care a apelat-o.

**Event Queue:** Event queue în Node.js este o coadă de evenimente care așteaptă să fie procesate de bucla de evenimente. Când un eveniment este emis, acesta este adăugat în coada de evenimente și așteaptă să fie procesat de bucla de evenimente.

**Event Loop:** Event loop în Node.js este un mecanism care se ocupă de procesarea evenimentelor din coada de evenimente și de coordonarea execuției între call stack și event queue. În fiecare iterare a buclei de evenimente, acesta verifică dacă există evenimente în coadă și le procesează în ordinea în care au fost adăugate, asigurându-se că execuția nu este blocată.

**NPM - Node Package Manager** Node.js include un sistem de gestionare a pachetelor, numit NPM (Node Package Manager), care permite dezvoltatorilor să instaleze, să actualizeze și să gestioneze biblioteci și module externe pentru aplicațiile lor. NPM facilitează distribuirea și reutilizarea codului și sprijină comunitatea Node.js în dezvoltarea și împărtășirea

În codul pentru serverul de semnalizare scris în Node.js, se utilizează modulul WebSocket pentru a crea un server WebSocket care gestionează conexiunile și comunicarea peer-to-peer.

## WebSocket

WebSocket este un protocol de comunicație care furnizează o conexiune bidirecțională și full-duplex între un client și un server prin intermediul unei conexiuni TCP. Modulul WebSocket importat în acest cod permite crearea unui server WebSocket care poate accepta conexiuni de la clientii WebSocket și gestionarea comunicării între aceștia.

**Ce este TCP?:** TCP este un protocol de transport utilizat pentru comunicația între calculatoare în cadrul rețelelor de calculatoare, cum ar fi internetul. Protocolul asigură o livrare fiabilă, ordonată și controlată a datelor între aplicații care rulează pe sisteme găzădă diferite. TCP este orientat spre conexiune, ceea ce înseamnă că se stabilește o conexiune între două puncte terminale și menține ca datele să fie transferate.

**De ce ce TCP?:** TCP (Transmission Control Protocol) este utilizat pentru stabilirea conexiunii inițiale între dispozitive în cadrul WebRTC din mai multe motive:

**Fiabilitatea:** TCP oferă un mecanism integrat de confirmare a primirii pachetelor și de retransmitere a acestora în cazul pierderii sau deteriorării lor. Acest aspect este important în faza inițială a stabilirii conexiunii, deoarece asigură că toate pachetele de control necesare pentru inițierea sesiunii sunt livrate corect și în ordine.

**Controlul fluxului:** TCP gestionează controlul fluxului prin intermediul algoritmului de control al congestiei și mecanismelor de alocare a resurselor. Acest lucru permite adaptarea ratei de transfer la capacitatea rețelei și evitarea suprasolicitării acesteia în timpul stabilirii conexiunii.

**Securitate:** TCP oferă funcționalități de securitate mai avansate comparativ cu UDP, inclusiv suport pentru criptare și autentificare. În timpul etapei de inițiere a sesiunii, se pot aplica măsuri suplimentare de securitate pentru a proteja comunicarea.

**Compatibilitatea cu infrastructura existentă:** TCP este un protocol larg utilizat și este acceptat de majoritatea dispozitivelor și infrastructurilor de rețea. Prin utilizarea TCP pentru stabilirea conexiunii inițiale, se asigură interoperabilitatea între dispozitivele care rulează WebRTC și alte aplicații și servicii care utilizează TCP.

După ce conexiunea este stabilită cu succes, WebRTC trece la utilizarea UDP pentru transferul efectiv al datelor în timp real, deoarece UDP oferă performanță superioară în ceea ce privește latența și eficiența transferului de date în timp real.

**Suprasolicitarea rețelei:** TCP utilizează și un algoritm de control al congestiei pentru a preveni suprasolicitarea rețelei și pentru a asigura o utilizare eficientă a lățimii de bandă disponibile. Acest algoritm ajustează dinamic dimensiunea ferestrei de transmisie a expeditorului pentru a se adapta la condițiile din rețea, precum întârzierea și pierderea de pachete.

**Numerele de secvență:** Un alt aspect important al TCP este numărul de secvență și numărul de confirmare (ACK). Numerele de secvență sunt atribuite fiecarui octet de date trimise, permîțând receptorului să reasambleze datele în ordinea corectă și să detecteze eventualele pierderi de pachete. Numerele de confirmare sunt utilizate pentru a indica expeditorului că datele au fost primite cu succes, iar în cazul în care nu se primește o confirmare, expeditorul poate retransmite datele.

În sinteză, TCP este un protocol de transport esențial în rețelele de calculatoare, care asigură o comunicare fiabilă, ordonată și controlată a datelor între aplicații. Acesta oferă control al fluxului, mecanisme de confirmare și retransmitere, precum și gestionarea congestiei pentru a asigura o utilizare eficientă a resurselor rețelei și o livrare de încredere a datelor între punctele terminale.

**Crearea serverului WebSocket** Un server WebSocket este creat folosind construcțorul `WebSocket.Server`. În acest caz, serverul este creat să asculte pe portul 8080. O instanță a serverului WebSocket este stocată în variabila `wss`.

**Gestionarea conexiunilor** Serverul WebSocket folosește evenimentul `connection` pentru a gestiona conexiunile clientilor. Când un client se conectează la server, o funcție de gestionare a conexiunilor este apelată, care primește obiectul `WebSocket` asociat clientului (denumit `ws` în acest caz) și o cerere (denumită `req`).

**Urmărirea clientilor conectați** Pentru a urmări clientii conectați și să mențină o listă a clientilor aflați în aceeași cameră, se utilizează o instanță Map denumită `clientsInRoom`. Această mapă stochează perechi cheie-valoare, unde cheia reprezintă ID-ul camerei, iar valoarea este un vector care conține obiectele `WebSocket` ale clientilor din camera respectivă.

**Gestionarea mesajelor** Când serverul `WebSocket` primește un mesaj de la un client, el declanșează evenimentul `message`. În acest caz, o funcție de gestionare a mesajelor este definită pentru a procesa mesajele primite de la clienti. Această funcție parsează mesajul primit în format JSON pentru a determina tipul mesajului și, în funcție de tip, procesează mesajul în consecință.

**Gestionarea deconectărilor** Când un client se deconectează de la server, evenimentul `close` este declanșat. O funcție de gestionare a deconectărilor este definită pentru a elimina clientul deconectat din lista clientilor din camera respectivă și pentru a reseta proprietatea `isInitiator` a clientului rămas în cameră, dacă este cazul.

**Transmiterea mesajelor** Funcția `broadcast` implementată este definită pentru a transmite un mesaj tuturor clientilor dintr-o anumită cameră, cu excepția celui care a trimis mesajul. Această funcție este folosită pentru a transmite mesaje precum oferte, răspunsuri și candidații între clienți pentru a stabili conexiunea.

# 6 Exemple de utilizare

În cadrul aplicației dezvoltate, există numeroase exemple de utilizare care pot fi aplicate în contextul unui interviu tehnic la distanță și al colaborării în echipă. Aceste exemple de utilizare demonstrează versatilitatea și utilitatea aplicației în diferite scenarii și contexte de lucru. Mai jos sunt prezentate în detaliu câteva exemple relevante:

## 6.1 Interviu Tehnic la Distanță

Aplicația oferă o platformă ideală pentru desfășurarea unui interviu tehnic la distanță. Interviewatorul și candidatul pot se conecta prin intermediul aplicației și pot interacționa în timp real pentru rezolvarea problemelor de programare și evaluarea abilităților tehnice. Exemple de utilizare includ:

- **Interviu de codare în timp real:** Interviewatorul poate scrie întrebări sau probleme de programare, iar candidatul poate răspunde prin scrierea și rularea codului în cadrul aplicației. Interviewatorul poate observa abilitățile de programare ale candidatului în timp real și poate oferi feedback imediat.
- **Analiză și debriefing:** După rezolvarea unei probleme de programare, interviewatorul și candidatul pot discuta rezultatele, abordarea și posibile îmbunătățiri. Aceasta facilitează evaluarea tehnică și permite interviewatorului să evaluateze gândirea critică și abilitățile de rezolvare a problemelor ale candidatului.
- **Testarea abilităților practice:** Interviewatorul poate cere candidatului să demonstreze abilități practice în timp real, cum ar fi implementarea unei funcționalități specifice sau rezolvarea unei probleme complexe. Aceasta oferă un mediu autentic pentru evaluarea abilităților tehnice ale candidatului.
- **Colaborare în timp real:** Interviewatorul și candidatul pot colabora pentru a rezolva o problemă de programare sau pentru a implementa o funcționalitate. Aceasta testează abilitățile de colaborare și capacitatea candidatului de a lucra într-o echipă.
- **Evaluarea performanțelor:** Aplicația poate fi utilizată pentru a evalua performanțele candidatului în timpul unui interviu tehnic. Interviewatorul poate înregistra timpul de rezolvare a unei probleme, precizia și eficiența codului scris și rezultatele obținute. Aceasta facilitează o evaluare obiectivă și comparativă a candidaților.

## 6.2 Colaborare în Echipă (Pair Programming)

Aplicația poate fi utilizată și în contextul colaborării în echipă, inclusiv Pair Programming. Această metodă implică doi programatori care lucrează împreună la același cod, în același timp și în același mediu de dezvoltare. Exemple de utilizare includ:

- Dezvoltare de funcționalități: Doi programatori pot lucra împreună pentru a implementa sau a rezolva o funcționalitate specifică. Aceasta implică colaborarea în timp real, scrierea și rularea codului în aplicație și rezolvarea problemelor de dezvoltare în mod eficient.
- Depanare și rezolvare de probleme: Atunci când apare o eroare sau o problemă în codul existent, doi programatori pot lucra împreună pentru a depista și a rezolva problema. Aceasta implică analizarea și depanarea codului în timp real, identificarea cauzelor și implementarea soluțiilor.
- Revizuire de cod: Pair Programming poate fi folosit și pentru a efectua revizuirea de cod în timp real. Doi programatori pot analiza și discuta împreună un fragment de cod, identificând posibile probleme, îmbunătățiri sau optimizări. Aceasta facilitează îmbunătățirea calității și a standardelor de cod.
- Mentorat și învățare: Pair Programming poate fi folosit pentru mentorat și învățare între un programator experimentat și un programator mai puțin experimentat. Programatorul experimentat poate ghida și oferi feedback în timp real, facilitând procesul de învățare și dezvoltare a abilităților.

# 7 Concluzii și Direcții Viitoare de Dezvoltare

În această lucrare, se prezinta o aplicație web dezvoltată pentru facilitarea interviurilor tehnice în timp real și colaborarea în echipă în domeniul dezvoltării software. S-a explorat facilitățile aplicației, inclusiv scrierea și executarea codului în timp real, colaborarea în timp real, partajarea de probleme și întrebări de programare, apelurile video și audio, chatul în timp real și gestionarea securității. Aceste facilități oferă un mediu interactiv, eficient și securizat pentru interviewatori și candidați, permitându-le să lucreze împreună în timp real și să evaluateze abilitățile tehnice într-un mod autentic.

În urma implementării și testării aplicației, putem trage concluzia că aceasta aduce multiple beneficii în procesul de interviu tehnic la distanță și colaborare în echipă. Prin eliminarea necesității unui server intermediar și prin folosirea tehnologiei WebRTC, aplicația oferă un mediu scalabil, securizat și performant. Interacțiunea în timp real între interviewatori și candidați facilitează evaluarea abilităților tehnice, permite feedback imediat și optimizează procesul de interviu. Colaborarea în timp real și partajarea de probleme și întrebări de programare permit dezvoltatorilor să lucreze eficient împreună, să rezolve probleme și să îmbunătățească calitatea codului.

Cu toate acestea, aplicația poate fi îmbunătățită și extinsă în direcții viitoare. Mai jos sunt prezentate câteva direcții de dezvoltare posibile:

## 7.1 Îmbunătățirea Interfeței Utilizator

Aplicația ar putea beneficia de o interfață utilizator mai intuitivă și atractivă, care să faciliteze navigarea și utilizarea facilităților oferite. Elementele de design și experiența utilizatorului pot fi îmbunătățite pentru a oferi o experiență mai plăcută și eficientă utilizatorilor.

## 7.2 Extinderea Funcționalităților

Aplicația ar putea fi extinsă prin adăugarea de funcționalități suplimentare, cum ar fi integrarea sistemelor de versionare (Git), instrumente de testare automată sau instrumente de analiză statică a codului. Acestea ar putea contribui la îmbunătățirea procesului de dezvoltare și evaluare a candidaților.

Un aspect important de menționat în cadrul aplicației este faptul că atunci când unul dintre utilizatori dă refresh la pagină, conexiunea între cei doi utilizatori se va pierde temporar. Cu toate acestea, această problemă este rezolvată și remediată în aplicație.

Datorită tehnologiei WebRTC utilizate în aplicație, odată ce conexiunea se pierde, aplicația va detecta automat acest lucru și va încerca să refacă conexiunea între cei doi utilizatori. Astfel, după ce utilizatorul a reîncărcat pagina sau a pierdut temporar conexiunea, aplicația va încerca să reconecteze utilizatorii pentru a restabili interacțiunea în timp real.

Totuși, un aspect important de luat în considerare este faptul că progresul realizat în timpul interacțiunii poate fi pierdut în acest proces. Deoarece nimic nu este salvat într-un server extern, atunci când conexiunea este întreruptă, datele și progresul realizate în timpul sesiunii se vor pierde. Astfel, utilizatorii vor trebui să reia activitățile și progresul de la început.

Pentru a evita acest lucru, o direcție viitoare de dezvoltare ar fi implementarea unei funcționalități care să permită salvarea și recuperarea progresului într-un mod sigur și eficient. Aceasta ar putea include stocarea datelor relevante într-o bază de date sau într-un sistem de fișiere pentru a permite utilizatorilor să continue de unde au rămas în cazul unei întreruperi a conexiunii.

Astfel, deși pierderea temporară a conexiunii poate fi rezolvată prin refacerea conexiunii între utilizatori, recuperarea progresului în cazul întreruperilor de conexiune rămâne o direcție viitoare de dezvoltare pentru a îmbunătăți experiența utilizatorilor și a asigura continuitatea în activitățile lor în cadrul aplicației.

### **7.3 Implementarea unui Sistem de Evaluare Automată**

Un aspect interesant pentru dezvoltare viitoare ar fi implementarea unui sistem de evaluare automată a codului scris de candidați. Acest sistem ar putea analiza și evalua automat calitatea, eficiența și corectitudinea codului, oferind feedback imediat și obiectiv atât interviewatorului, cât și candidatului.

### **7.4 Integrarea cu Sisteme de Recrutare și Gestioneare a Talentelor**

Aplicația ar putea fi integrată cu sisteme de recrutare și gestionare a talentelor existente, facilitând procesul de înregistrare a candidaților, urmărirea progresului și evaluarea acestora. Integrarea cu astfel de sisteme ar simplifica fluxul de lucru și ar aduce un nivel mai mare de eficiență în procesul de recrutare și selecție.

### **7.5 Extinderea Aplicației pe Platforma Mobilă și Hosting**

O direcție viitoare de dezvoltare pentru aplicație ar fi extinderea sa pe platforma mobilă, astfel încât să fie disponibilă și utilizatorilor de dispozitive mobile. Implementarea

unei aplicații mobile ar oferi o experiență mai accesibilă și mai convenabilă utilizatorilor, permitându-le să acceseze facilitățile aplicației de pe telefoanele lor inteligente sau tablete.

Cu toate acestea, trebuie menționat că implementarea și furnizarea unei aplicații mobile implică costuri suplimentare și cerințe tehnice specifice. Pe lângă dezvoltarea aplicației mobile în sine, există și aspecte legate de hosting și disponibilitate. De obicei, o aplicație mobilă necesită o infrastructură de server dedicată pentru a asigura funcționalitatea sa și pentru a permite comunicarea cu utilizatorii prin intermediul dispozitivelor mobile.

Hostingul și mențenanța unei aplicații mobile necesită resurse financiare și cunoștințe tehnice adecvate pentru a gestiona și a menține serverele și infrastructura asociată. Este nevoie de un buget lunar pentru a plăti o firmă de hosting pentru a găzdui atât partea de client-side, cât și partea de server-side a aplicației mobile.

Extinderea aplicației pe platforma mobilă și asigurarea unui hosting adecvat ar permite accesul la aplicație unui număr mai mare de utilizatori și ar crește disponibilitatea și accesibilitatea acesteia. În plus, ar oferi utilizatorilor posibilitatea de a accesa facilitățile aplicației de pe dispozitivele lor preferate, indiferent de locație sau timp.

Astfel, extinderea aplicației pe platforma mobilă și furnizarea unei infrastructuri de hosting adecvate reprezintă o direcție viitoare de dezvoltare care ar aduce beneficii semnificative în ceea ce privește accesibilitatea și utilizabilitatea aplicației pentru un număr mai mare de utilizatori.

# Bibliografie

- [1] Getting started with webrtc, 2023. Accesat la: 10.11.2022. URL: <https://webrtc.org/getting-started/overview>.
- [2] MDN contributors. Webrtc api, 2023. Accesat la: 01.02.2023. Last modified: 2023-06-05. Copyright: 1998-2023 by individual mozilla.org contributors. Content available under a Creative Commons license. Accesat la: 2022-12-15. URL: [https://developer.mozilla.org/en-US/docs/Web/API/WebRTC\\_API](https://developer.mozilla.org/en-US/docs/Web/API/WebRTC_API).
- [3] Herman Zvonimir Došilović and Igor Mekterović. Robust and scalable online code execution system. In *2020 43rd International Convention on Information, Communication and Electronic Technology (MIPRO)*, pages 1627–1632, 2020. Accesat la: 05.04.2023. doi:10.23919/MIPRO48935.2020.9245310.
- [4] Ilya Grigorik. WebRTC BROWSER APIS AND PROTOCOLS, CHAPTER 18, 2013. Accesat la: 20.05.2023. URL: <https://hpbn.co/webrtc/>.
- [5] Cullen Jennings, Florent Castelli, Henrik Boström, Jan-Ivar Bruaroey, Ian Hickson, et al. Webrtc 1.0: Real-time communication between browsers, 2023. Accesat la: 20.01.2023. URL: <https://www.w3.org/TR/webrtc/>.
- [6] Microsoft. Monaco editor, 2023. Autori: Comunitatea de pe Github. Accesat la: 10.01.2023. URL: <https://github.com/react-monaco-editor/react-monaco-editor>.
- [7] Node.js Team. Node.js documentation, 2023. Accesat la: 18.12.2022. URL: <https://nodejs.org/en/docs>.
- [8] React Team. React getting started, 2023. Accesat la: 18.12.2023. URL: <https://legacy.reactjs.org/docs/getting-started.html>.
- [9] WebRTC. Webrtc architecture, 2023. Autori: Comunitatea WebRTC. Accesat la: 10.11.2022. URL: <https://webrtc.github.io/webrtc-org/architecture/>.