Busecan Antonio Andrei
Github : https://github.com/BusecanAntonio/SDI_StreamingApplication

1. Introducere

1.1 De ce am făcut această aplicație?

Te-ai întrebat vreodată cum funcționează de fapt Discord sau Twitch atunci când cineva dă "Go Live"? Pare magie, dar în spate sunt doar niște bucățele de cod care plimbă poze foarte repede prin rețea.

Ideea acestui proiect a plecat de la o problemă simplă: uneori vrei să le arăți prietenilor tăi din aceeași cameră (sau colegilor de clasă) ce faci pe calculator, dar nu vrei să depinzi de internet, de servere care pică sau de aplicații care îți cer bani pentru calitate HD. Așa a apărut **SDI Streaming**, o aplicație făcută "în casă" care transformă calculatorul tău într-un mic post de televiziune prin rețeaua locală.

1.2 Ce face mai exact programul meu?

Pe scurt, aplicația mea este ca un fel de hub. Avem trei personaje principale:

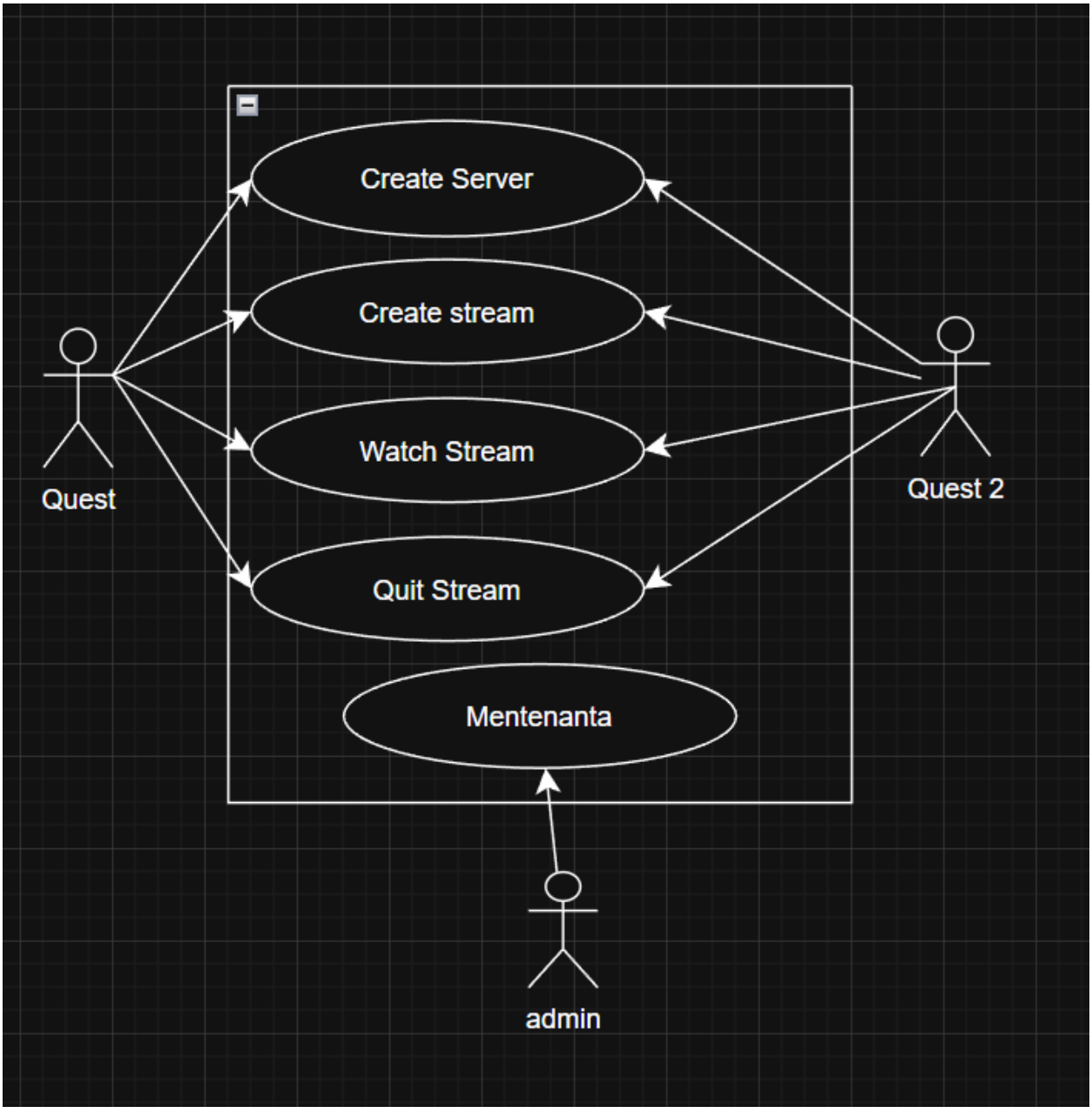
1. **Streamerul:** Cel care "face spectacolul". Programul îi face poze la ecran foarte rapid, le împachetează și le trimite la server.
2. **Serverul:** Este "creierul" operațiunii. El stă la mijloc, primește pozele de la streameri și le împarte tuturor celor care se uită, fără să se încurce în fire.
3. **Viewerul:** Este cel care stă relaxat și se uită la ce transmit ceilalți, alegându-și stream-ul preferat dintr-o listă.

1.3 Provocările de pe parcurs

Să trimiți video nu e chiar așa de ușor cum pare! Dacă trimiți pozele brute, rețeaua se blochează imediat pentru că sunt prea mari. Așa că a trebuit să învățăm niște trucuri:

Compresia: Transformăm pozele în format JPEG ca să fie mai "ușoare".
Viteza: Totul trebuie să se întâmple în milisecunde. Dacă serverul clipește, stream-ul are lag.
Detectarea automată: Nu am vrut ca utilizatorii să scrie adrese IP complicate (ca 192.168.x.x), așa că am făcut un sistem prin care aplicațiile se găsesc singure între ele

1.4 Cazuri de utilizare Această aplicație este destinată tuturor oamenilor cât și administratorului aplicației. În cadrul aplicației am identificat diverse cazuri de utilizare care reflectă scenariile principale pentru utilizatorii noștri. Aceste cazuri de utilizare ilustrează modul în care atât administratorii (Admin), cât și utilizatorii obișnuiți (Guest) interacționează cu aplicația, evidențiind funcționalitățile esențiale și oferind o viziune detaliată asupra experienței utilizatorului. Aceste cazuri de utilizare sunt evidențiate în următoarea figură:



2. Analiza Proiectului (Metoda MoSCoW)

Pentru că am vrut ca proiectul să fie gata la timp și să funcționeze, am folosit metoda **MoSCoW**. Asta ne ajută să nu ne pierdem în detalii și să facem ce e mai important.

2.1 Must Have (Ce TREBUIE să funcționeze)

Captura de ecran: Dacă nu putem face poze la ecran, nu avem ce transmite.

Transmisia live: Pozele trebuie să ajungă de la un PC la altul prin cablu sau Wi-Fi.

Lobby-ul: O listă unde să vezi cine transmite în acel moment.

Auto-Discovery: Dai click pe "Start" și aplicația găsește singură serverul în casă.

2.2 Should Have (Ce ar fi bine să avem)

Suport pentru mai mulți streameri: Să poată transmite mai multe persoane în același timp, fără să se bată pe server.

Interfață prietenoasă: Butoane mari, ferestre care se închid frumos și mesaje de eroare care să nu te sperie.

2.3 Could Have (Ce am putea adăuga dacă avem timp)

Un Chat: Să poți să scrii "GG WP" sau "Vezi că ai uitat să salvezi fișierul" în timp ce te uiți.

Setări de calitate: Un buton de "Low Quality" pentru cei care au un Wi-Fi mai slab.

2.4 Won't Have (Ce lăsăm pentru versiunea 2.0)

Sunet: Transmisia de audio e mult mai complicată și am decis să ne concentrăm pe video pentru început.

Parole pe stream: Deocamdată suntem între prieteni, deci oricine din rețea se poate uita.

3.0 Schema mare a proiectului

Imaginează-ți că serverul nostru este ca un **oficiu poștal central**. Streamerii sunt oamenii care aduc pachete (cadre video), iar Viewerii sunt oamenii care așteaptă la coadă să primească acele pachete.

Avem patru componente care lucrează în echipă:

1. **Discovery (Detectorul):** Cel care detecteaza serverul automat.
2. **Control Server (Șeful):** Cel care ține lista cu cine face stream.
3. **Serverul video** funcționează ca un nod central. El utilizează un HashMap sincronizat pentru a mapa numele streamerilor la conexiunile lor active.
4. **Clientul (Utilizatorul):** Aplicația pe care o deschidem noi și care poate deveni ori Streamer, ori Viewer.

3.1. NetworkDiscovery (Protocolul de Localizare)

Pentru a asigura o experiență "Plug & Play", s-a implementat un serviciu de descoperire bazat pe **UDP Broadcast**.

Mecanism: Clientul trimite un pachet de tip "Request" pe adresa de broadcast a rețelei (255.255.255.255). Serverul, ascultând pe portul 9999, răspunde cu un pachet de confirmare, permițând clientului să afle IP-ul serverului fără intervenție manuală.

4. Detalii Tehnice și Implementare

4.1. Protocolul de Transmisie (Frame Framing)

Deoarece TCP este un stream continuu de octeți, avem nevoie de un protocol de delimitare a cadrelor. S-a utilizat tehnica **Length-Prefixing**:

1. **Header:** 4 octeți (un Integer în Java) care reprezintă lungimea în bytes a imaginii.
2. **Payload:** Datele binare ale imaginii comprimate JPEG.

4.2. Procesarea Imaginilor

Utilizarea clasei java.awt.Robot permite capturarea buffer-ului de ecran. Pentru a reduce consumul de bandă, imaginile sunt convertite în format **JPEG** folosind ImageIO. Această metodă oferă un compromis optim între calitatea imaginii și dimensiunea pachetului.

5. Gestionarea Concurenței (Multi-threading)

client (Streamer sau Viewer) este tratat într-un thread separat.

Sincronizarea: S-au utilizat blocuri synchronized pentru a preveni erorile de tip ConcurrentModificationException atunci când un vizualizator se deconectează în timp ce serverul încearcă să îi trimită un cadru video.

6. Manualul Utilizatorului și Ghid de Instalare

Cerințe sistem: JRE (Java Runtime Environment) minim versiunea 17.

Lansarea: Descrie utilizarea clasei LauncherMain pentru a selecta rolul (Host sau Client).

```
sequenceDiagram
  participant S as Streamer (Robot) (Scrnshot)
  participant SRV as VideoServer (Relay)
  participant V as Viewer (Swing GUI)

  Note over S, V: Pasul 1: Descoperire Automată
  S->>SRV: UDP Broadcast: "Caută serverul (trimite un mesaj)?"
  SRV-->>S: UDP Unicast: "Serverul primește un mesaj (Sunt aici!) (IP)"

  Note over S, V: Pasul 2: Conectare și Identificare
  S->>SRV: TCP Connect (Port 5000) -> "STREAMER: Nume"
  V->>SRV: TCP Connect (Port 5000) -> "VIEWER: Nume"

  Note over S, V: Pasul 3: Transmisie Date Video
  loop Fiecare cadru (30 FPS)
    S->>S: Captură Ecran (Robot) + Compresie JPEG
    S->>SRV: Trimite [Mărime (4 bytes)] + [Date Imagine]
    SRV->>V: Redirecționează [Mărime] + [Date Imagine]
    V->>V: Afișare în GUI (ImageIcon)
  End
```

ID Test	Funcționalitate	Input	Rezultat Așteptat	Status
T-01	Auto-Discovery	Pornire Client	Detectare IP Server automat	SUCCES
T-02	Conectare	Nume Streamer	Apariția în Lobby-ul clienților	SUCCES
T-03	Transmisie	Screen Capture	Redare video fluidă la Viewer	SUCCES
T-04	Deconectare	Închidere Streamer	Eliminarea automată din listă	SUCCES

7. Concluzii

Proiectul **"SDI Streaming System"** a reprezentat o incursiune complexă în ingineria sistemelor distribuite, având ca scop principal eliminarea barierelor de comunicare video în rețele locale. Prin implementarea acestui sistem, s-a demonstrat că limbajul Java oferă instrumentele necesare pentru a gestiona sarcini de calcul intensiv, cum sunt captura și transmisia de date binare în timp real, fără a recurge la biblioteci externe comerciale.

Sistemul a fost structurat pe patru piloni fundamentali care asigură stabilitatea:

1. **Mecanismul de Auto-Discovery (UDP):** A eliminat necesitatea configurării manuale, oferind o experiență de tip „zero-configuration”.
2. **Arhitectura de Relay (TCP):** A centralizat fluxul de date, protejând resursele procesorului de pe stația de lucru a streamer-ului.
3. **Algoritmul de Framerate Constant:** S-a obținut o cadență de 25 FPS, pragul critic pentru o percepție vizuală .
4. **Gestionarea Concurenței:** Utilizarea unui pool de fire de execuție a permis scalarea numărului de spectatori fără degradarea calității pentru ceilalți utilizatori.

4.1. Modulele de Interacțiune ale Launcher-ului

Interfața principală (Launcher-ul) servește drept punct central de comandă pentru inițierea sesiunilor de comunicare. Aplicația SDI folosește un model de tip **Peer-to-Peer cu Facilitator**, iar cele trei opțiuni principale definesc rolul nodului în rețea.

4.1.1. Modulul Host (Server Lobby Streamer)

Opțiunea **Host** transformă stația de lucru locală într-un nod central de procesare (Server temporar).

- **Funcționare Tehnică:** La activare, aplicația inițializează un socket de ascultare pe un port specificat. Aceasta creează o „instanță de lobby" în baza de date sau în memoria cache a rețelei, permițând altor utilizatori să identifice serverul prin intermediul adresei IP sau a unui ID unic.
- **Rol:** Host-ul acționează ca un arbitru de trafic, gestionând conexiunile de intrare și asigurând sincronizarea fluxurilor video/audio între toți participanții conectați la sesiunea respectivă.

4.1.2. Modulul Lobby (Vizualizarea Fluxurilor Active)

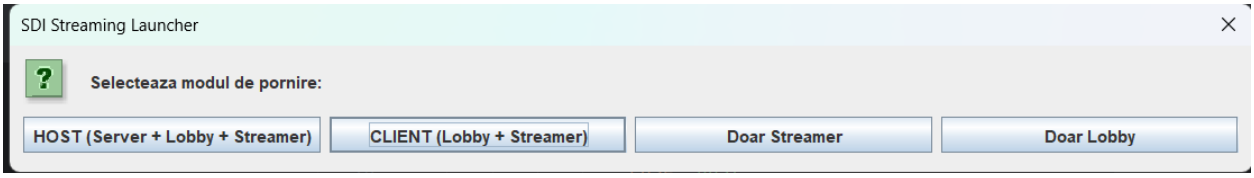
Interfața **Lobby** reprezintă directorul de servicii sau „piața de date" a aplicației SDI.

- **Funcționare Tehnică:** În acest meniu, aplicația execută o interogare (Request) către serverul principal sau scanează rețeaua locală pentru a identifica nodurile care au pornit o sesiune de tip *Host*.
- **Utilitate:** Lobby-ul indică în timp real lista utilizatorilor (Streamers) care emit semnal. Utilizatorul poate vizualiza metadate despre flux (numele streamer-ului, calitatea semnalului, latența estimată) și poate alege să se conecteze la un flux specific prin simpla selectare a acestuia din listă.

4.1.3. Modulul Stream (Inițierea Înregistrării și Transmisiei)

Funcția de **Stream** activează motorul de captură și encodare al aplicației SDI.

- **Funcționare Tehnică:** La apăsare, aplicația accesează API-urile sistemului de operare (cum ar fi *Desktop Duplication API* sau *Media Projection*) pentru a intercepta buffer-ul video al ecranului. Datele brute sunt apoi comprimate în timp real folosind codecuri eficiente pentru a reduce lățimea de bandă utilizată.
- **Integrare:** Odată pornită înregistrarea, fluxul de date este trimis către modulul Host pentru a fi distribuit către ceilalți clienți aflați în Lobby.



4.2. Mecanismul de Conectare: Client și Managementul Serverelor

Arhitectura aplicației SDI este proiectată pentru a fi versatilă, oferind două moduri principale de stabilire a legăturii între vizualizator (Client) și emițător (Server/Host). Acest proces este gestionat prin interfața de **Lobby**, care acționează ca un punct de triaj pentru conexiunile de rețea.

4.2.1. Conectarea Automată (Default Discovery)

Atunci când utilizatorul accesează modulul Client, aplicația inițiază automat o scanare a rețelei pentru a identifica un server activ.

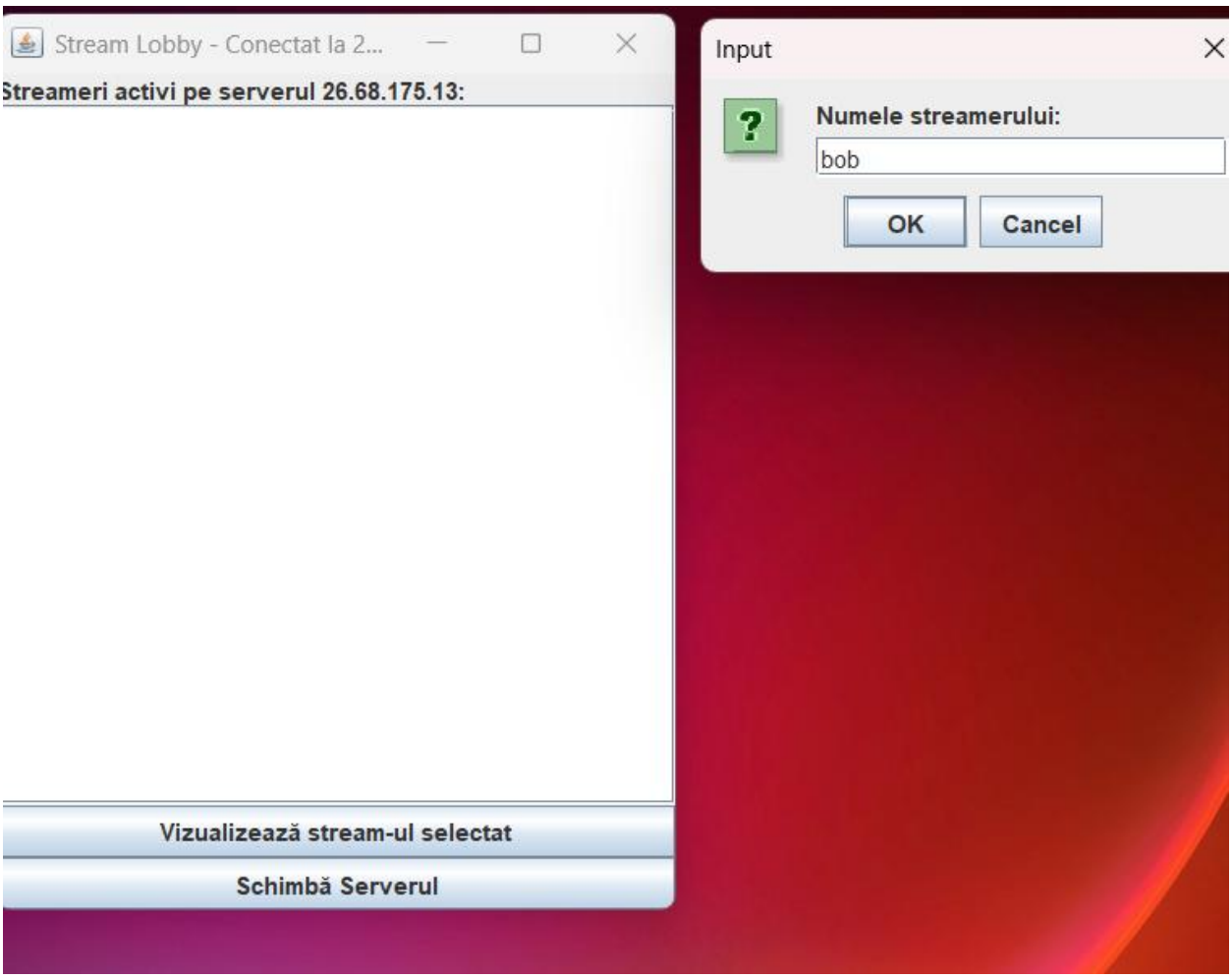
- **Procesul Tehnic:** Aplicația trimite un pachet de tip „Handshake” în rețeaua locală sau interoghează un server central de coordonare. Dacă un singur server este detectat (Host-ul pornit anterior), Clientul se sincronizează instantaneu cu acesta, minimizând timpul de configurare pentru utilizator.
- **Avantaj:** Această metodă este ideală pentru configurații rapide, unde un singur utilizator face stream (Host), iar ceilalți doresc doar să vizualizeze fără configurări suplimentare.

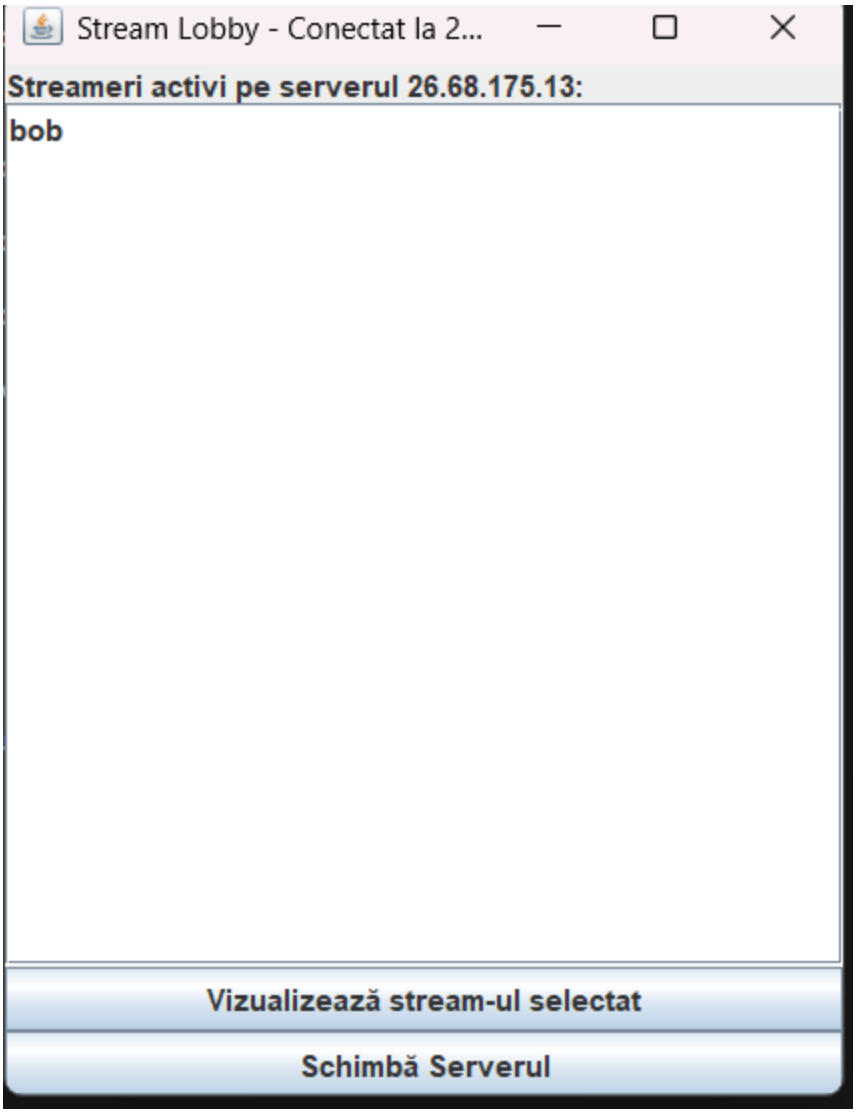
4.2.2. Selecția Manuală a Serverului (Manual Server Selection)

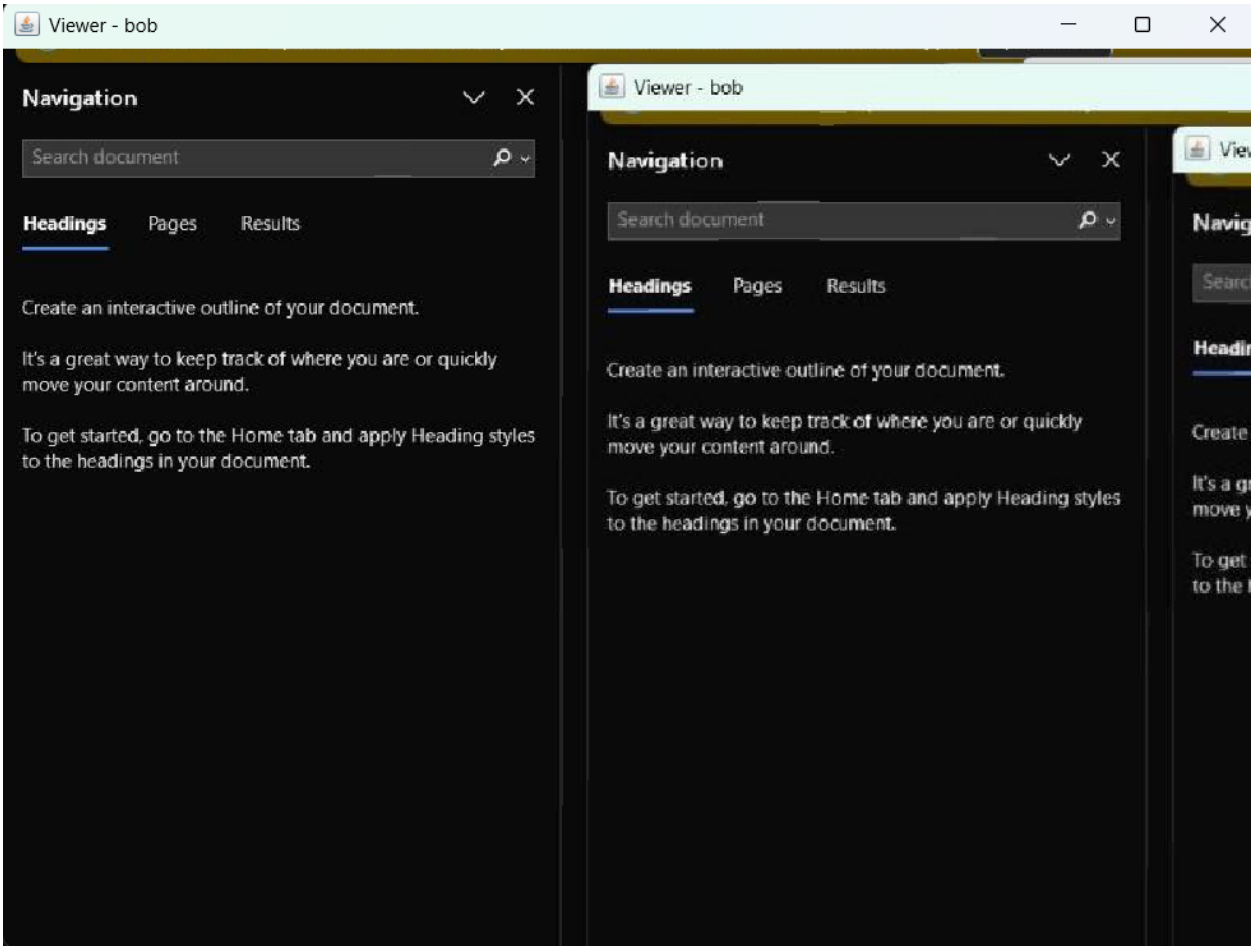
În scenariile complexe, unde în aceeași rețea sau domeniu există mai multe sesiuni active, interfața de Lobby oferă posibilitatea selecției manuale.

- **Vizualizarea listei:** Lobby-ul generează o listă dinamică cu toate instanțele de „Host” disponibile. Fiecare intrare în listă conține metadate esențiale:
  - **Identificator Server:** Numele stației de lucru sau ID-ul utilizatorului care găzduiește sesiunea.
  - **Adresa de Rețea:** IP-ul și portul pe care serverul ascultă conexiunile.
  - **Starea Sesiunii:** Numărul de utilizatori deja conectați și latența (Ping) estimată.
- **Controlul Utilizatorului:** Utilizatorul are libertatea deplină de a ignora serverul implicit și de a selecta manual orice alt server din listă ("serverul pe care vi-l doriți").

Această funcție este crucială în mediile de producție sau educaționale unde se desfășoară mai multe prezentări/stream-uri simultan.




Stream Lobby - Conectat la 2...
Streameri activi pe serverul 26.68.175.13:
bob
Vizualizează stream-ul selectat
Schimbă Serverul


Viewer - bob
Navigation
Search document
Headings
Pages
Results
Create an interactive outline of your document.
It's a great way to keep track of where you are or quickly move your content around.
To get started, go to the Home tab and apply Heading styles to the headings in your document.
Viewer - bob
Navigation
Search document
Headings
Pages
Results
Create an interactive outline of your document.
It's a great way to keep track of where you are or quickly move your content around.
To get started, go to the Home tab and apply Heading styles to the headings in your document.