

UNIVERSITATEA "ALEXANDRU-IOAN CUZA" DIN IASI

FACULTATEA DE INFORMATICA



LUCRARE DE LICENTA

AndroidGameStreamer

propusă de

Ştefan Crăciun

Sesiunea: iulie, 2020

Coordonator științific

Conf. Dr. Gațu Cristian

Cuprins

Motivație	3
Introducere	4
1 Aplicatii similare	7
1.1 Google Stadia	7
1.2 GeForce Now	9
2 Detalii de implementare	12
2.1 Tehnologii folosite	12
2.1.1 Java	12
2.1.2 Python	14
2.1.3 C	14
2.1.4 SQLite	15
2.1.5 Protocolul TCP	15
2.1.6 Protocolul UDP	16
2.2 Arhitectură și implementare	16
2.2.1 CentralServer	16
2.2.2 WindowsClient	17
2.2.3 AndroidClient	18

2.2.4	StreamingModule: Compresia și transmiterea fluxului video	19
2.2.5	ListenerModule: transmisia și interpretarea input-ului efectuat de utilizator	21
3	Scenarii de utilizare	24
3.1	Aplicația pentru Windows	24
3.2	Aplicația pentru Android	25
4	Direcții de dezvoltare	29
4.1	Suport pentru controller	29
4.2	Fluxul video	30
4.3	Fluxul audio	32
4.4	Setări în aplicația Android	32
	Bibliografie	34

Motivătie

Am simțit nevoia unei aplicații de game streaming încă de acum câțiva ani, când încă eram la liceu.

Mai ales în timpul vacantelor de vară, se întâmpla să fiu plecat de acasă câteva zile la rând. Cum la vremea respectivă nu dețineam laptop, ci doar un desktop pe care puteam instala jocuri de Windows, nu puteam juca nimic dacă nu eram acasă.

Fiind vorba, cu aproximativă, de anul 2014, majoritatea programelor care astăzi rezolvă această problemă erau încă în dezvoltare, sau, în unele cazuri, încă inexistente.

În prezent, beneficiem de numeroase aplicații de game streaming ce oferă o experiență de calitate în ceea ce privește accesarea de la distanță a jocurilor ce rulează pe o mașină puternică, folosind un alt calculator cu specificații modeste. Totuși, nu multe au și versiuni de Android, iar acelea care au sunt mai mult axate pe cloud gaming. Pentru acestea este nevoie uneori să cumpărăm din nou jocul pentru a-l putea juca pe platforma respectivă, chiar dacă deținem deja versiunea de PC (cazul Google Stadia).

De aceea, am considerat că este nevoie de o aplicație prin care să putem rula jocurile instalate pe calculatorul personal și când nu ne aflăm în fața acestuia. Luând în considerare că, în aceasta situație, este mult mai probabil să avem la noi un telefon decât un laptop, am decis să încerc dezvoltarea unei aplicații pentru Android.

Aceasta încercare a primit doar de curând un nume: `AndroidGameStreamer`.

Introducere

Serviciile de Game Streaming și Cloud Gaming au atras în ultimii ani un deosebit interes din partea mai multor categorii de persoane, printre care se numără pasionații de jocuri video, companiile care le dezvoltă și nu în ultimul rând, producătorii de echipamente folosite în gaming (console, telefoane inteligente, sisteme de calcul și componente pentru acestea). Fiecare categorie amintită mai sus este interesată de aceste servicii din motive diferite.

În primul rând, **gamerilor** li se pun în prezent la dispoziție jocuri video care reușesc să transpună jucătorul într-o lume virtuală de un realism care cu 10 sau 15 ani în urmă era aproape inimaginabil.

Este evident că pentru a fi posibilă dezvoltarea unor jocuri ce oferă experiențe din ce în ce mai bogate a fost nevoie de o creștere susținută a performanțelor echipamentelor folosite pentru rularea acestora. Prin urmare, în prezent, chiar și cele mai performante sisteme de gaming rar pot face față cerințelor celor mai noi jocuri la mai mult de 5 ani de la achiziționarea sau asamblarea lor.

Cum înlocuirea sau îmbunătățirea lor este destul de costisitoare, pasionații jocurilor video au căutat întotdeauna alternative la upgrade-urile de hardware atunci când noile jocuri au cerințe atât de mari încât experiența are de suferit (atunci când devine necesar un compromis între calitatea setărilor grafice și framerate).

Pentru perspectivă, este nevoie de aproximativ 1000 de euro pentru a acoperi costul componentelor necesare construirii unui PC de gaming decent care va fi în stare să ruleze jocuri la calitate Full HD (1080p) și setări grafice maxime pentru 2 sau 3 ani. Prețul poate urca până la 3000 de euro dacă cumpărăm un sistem gata asamblat în loc să achiziționăm fiecare piesă separat, dacă dorim să rulăm jocuri la o rezoluție

superioară (1440p sau 4K), sau dacă odată cu sistemul avem nevoie și de periferice (mouse, monitoare de gaming, etc.). La acestea se adaugă și costul jocurilor în sine care nu este nici el de neglijat în seamă.

Pe lângă reducerea costurilor, un alt avantaj al accesării unui joc care rulează pe un server în cloud este că fișierele jocului nu mai ocupă spațiu pe calculatorul personal. Dacă în urmă cu 10 ani aceasta nu era o problemă majoră, spațiul ocupat de un joc depășind rar 7 GB, în prezent acestea ajung până la 150 GB, ceea ce reprezintă un procent însemnat chiar și din spațiul soluțiilor de stocare de mare capacitate din ziua de astăzi. De asemenea, în cazul jocurilor ce rulează în cloud, actualizările se realizează automat, ceea ce ne scutește și de timpii de așteptare când vine vremea instalării unui update.

Dezvoltatorii de jocuri video sunt atrași în principal de expunerea mai mare a propriului brand în fața consumatorului, fapt care urmează lansării creațiilor lor pe o platformă populară de Cloud Gaming.

Se știe deja că de pe urma implementării unui ecosistem în care se pot cumpăra și juca jocuri de diferite genuri și produse de companii diferite beneficiază atât gamerilor (care găsesc tot ce au nevoie în același loc și beneficiază de reduceri frecvente), cât și companiile dezvoltatoare de jocuri (acestea au parte de vânzări mult mai mari de pe urma magazinelor de jocuri online binecunoscute decât a reclamelor și a livrării în format fizic). Când vorbesc despre succesul magazinelor virtuale de jocuri video, mă refer, desigur, la binecunoscuta platformă de distribuție Steam.

În altă ordine de idei, chiar drepturile de publicare a unei versiuni compatibile cu anumite tipuri de sisteme sunt extrem de costisitoare. De exemplu, aproximativ 30% din profitul realizat în urma vânzării fiecărei copii pentru Xbox sau Playstation a unui joc merge către companiile care dețin drepturile asupra acestor tehnologii (Microsoft, respectiv Sony).

Un sistem de cloud gaming performant ar pune presiune pe cei care dețin aceste drepturi să rămână competitivi pe piață. Prin urmare, beneficiile ar fi atât de partea dezvoltatorilor, ale căror costuri scad, cât și a gamerilor, care nu ar mai avea nevoie să cumpere o nouă copie a aceluiasi joc compatibilă cu fiecare dispozitiv pe care doresc îl ruleze.

Producătorii de echipamente pentru gaming se află într-o poziție deosebit de avantajoasă pentru a lansa un serviciu cloud gaming de succes, deoarece beneficiază deja de hardware-ul necesar construirii unor ferme de servere care să ruleze jocurile.

O astfel de abordare ar fi foarte eficientă din punctul de vedere al costului de implementare, deoarece componentele de care este nevoie și care nu sunt deja produse de compania în cauză vor fi achiziționate în cantități mari (caz în care furnizorul ar putea fi deschis negocierilor). De asemenea, un server performant poate rula mai multe sesiuni de cloud gaming simultan, spre deosebire de hardware-ul disponibil consumatorului de rând, ceea ce ar duce din nou la economii substantiale.

Totuși, implementarea sistemelor de cloud gaming este deosebit de dificilă mai ales la scară mare, din numeroase motive. Cel mai mare problemă este aceea a latenței, care trebuie să fie cât mai mică atât în cazul fluxului video și a celui audio, cât și a input-ului introdus de utilizator. Prin urmare, implementarea la scară largă necesită ca serverele ce rulează jocurile să se afle fizic aproape de utilizator. De aceea, singura soluție este ca acestea să se afle în numeroase centre, ceea ce poate deveni costisitor.

Capitolul 1

Aplicatii similare

1.1 Google Stadia

Numită în timpul dezvoltării Project Stream, Stadia este aplicația de cloud gaming creată de Google și lansată în noiembrie 2019 (s-a aflat în versiunea closed beta încă din octombrie 2018).

Conform Google, este capabil de streaming la o rezoluție de până la 4K și 60 fps. Există versiuni pentru Android (deși în acest moment, doar anumite dispozitive sunt suportate oficial), pentru browserul Google Chrome și pentru smart TV-uri (în acest caz este nevoie de dispozitivul pentru streaming video Chromecast Ultra).

Odată cu lansarea unui controller dedicat pentru Stadia, Google a adăugat unele funcționalități interesante, precum trimiterea input-ului direct la servere prin Wi-Fi, sau un buton pentru Google Assistant, care, apăsat în timpul jocului, vă căuta un clip pe Youtube și momentul din cadrul acestuia care este relevant pentru punctul în care ne aflăm în joc. În acest fel putem primi indicii în cazul în care nu știm cum să progresăm.

Deoarece salvările sunt stocate în cloud, controllerul poate fi deconectat de la dispozitivul pe care ne jucăm și conectat la altul pe care să continuăm exact din punctul în care am rămas.

În ceea ce privește resursele necesare pentru a folosi Stadia, cea mai importantă este conexiunea la Internet, Google recomandând o lățime de banda minima de 20 Mbps pentru gaming în Full HD (1080p) și 35 Mbps pentru 4K. Aceasta face aplicația

nepotriva pentru a fi utilizata prin date mobile, consumând în medie 10 GB pe ora.



Figura 1.1: Controllerul Stadia

Totuși, există și mai multe dezavantaje care au făcut ca lansarea acestui serviciu să nu fie la măsura așteptărilor. În primul rand, deși oferă un abonament plătit, nu poate fi considerat "un Netflix al jocurilor", deoarece majoritatea titlurilor disponibile trebuie cumpărate în versiunea pentru Google Stadia înainte de a fi jucate. Această abordare ridică probleme dacă Google decide că proiectul nu este profitabil, caz în care cei ce au cumpărat versiunea pentru Stadia a unui joc nu vor mai putea să o folosească.

Acesta este probabil cauza principală pentru care Stadia nu a reușit încă, la mai bine de 7 luni de la lansare, să stabilească o comunitate solidă de gameri. Din acest motiv, partea multiplayer a jocurilor nu este încă o experiență adekvată, din cauza timpilor foarte mari de așteptare pentru a găsi suficienți jucători spre a începe o sesiune multiplayer.

Sistemul adaugă și un input lag însemnat, acesta fiind în medie cu 60ms mai mare decât cel observat jucând același joc pe dispozitivul local. Desigur, această valoare poate varia foarte mult în funcție de calitatea conexiunii și este în mare parte datorată distanței până la serverul pe care rulează jocul, deci nu poate fi evitată în totalitate.

Pe lângă acestea, mai pot fi amintite și o implementare neoptimă de anti-aliasing, ceea ce face multe jocuri să nu aibă aceeași calitate grafică observată pe alte platforme,



Figura 1.2: Shadow of the Tomb Raider pe platforma Stadia

desincronizări ale fluxului audio față de cel video și momente de lag inexplicabil chiar și pe o conexiune bună la Internet

1.2 GeForce Now

Nvidia este printre primii dezvoltatori ai unei aplicații de Cloud Gaming, versiunea GeForce Now pentru Nvidia Shield (un player media pentru smart TV-uri), inițial cunoscută ca și Nvidia GRID fiind lansată pentru prima dată în 2013. În ianuarie 2017, a fost lansată versiunea beta pentru PC și Mac, devenind accesibilă publicului larg în februarie 2020 în America de Nord și Europa.

Pentru a folosi GeForce Now, conform Nvidia, este nevoie de o conexiune la Internet de cel puțin 25 Mbps pentru gaming la 1080p și 60 de cadre pe secunda. În prezent nu este disponibil streaming-ul la rezoluții mai mari de Full HD.

Pentru a evita supraaglomerarea serverelor, sesiunile de gaming sunt limitate la o oră în cazul versiunii gratuite, sau la 6 ore dacă folosim abonamentul plătit, care costă 5 dolari.

Una dintre cele importante funcționalități este integrarea cu alte mari platforme de gaming deja existente, printre care se numără și Steam. Aceasta vine în avantajul

gamerilor care deja dețin o colecție extinsă de jocuri în format digital, deoarece nu este nevoie să cumpere o nouă versiune a aceluiași joc doar pentru streaming în cadrul GeForce Now.

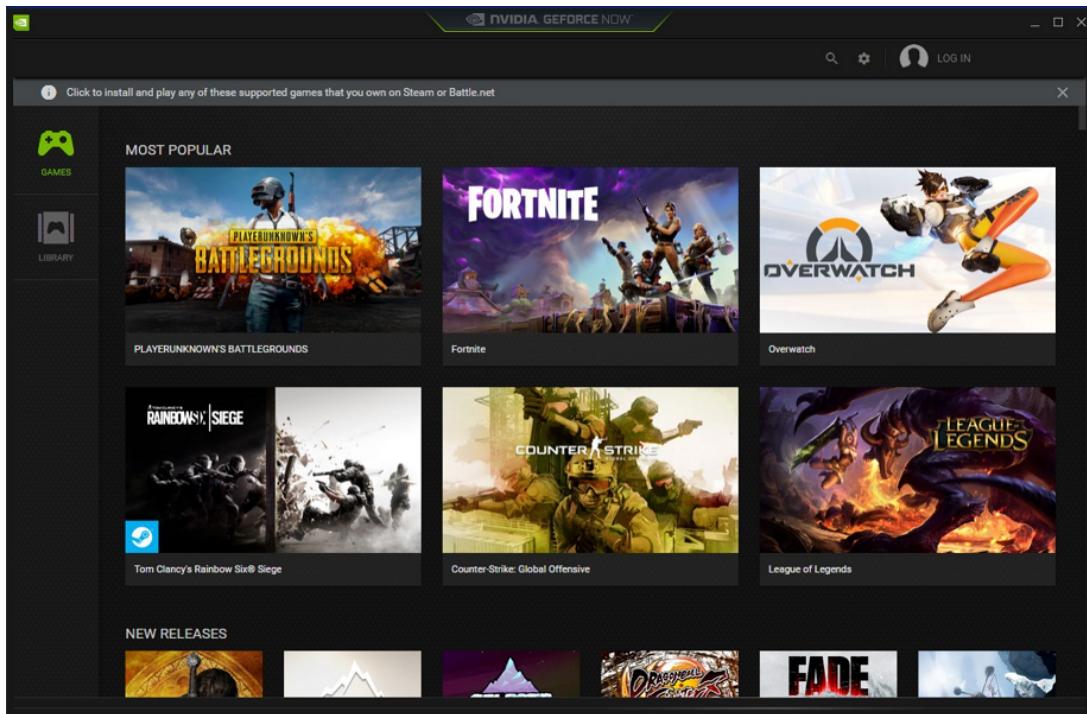


Figura 1.3: Ecranul principal al aplicației GeForce Now

Cu toate acestea, abordarea aleasă de Nvidia a cauzat numeroase controverse privitoare la drepturile de redistribuire a conținutului software, din care a rezultat retragerea anumitor jocuri de pe platformă la cererea dezvoltatorilor.

Pe de-o parte, în viziunea acestora, atunci când cineva cumpără un joc, respectivul primește doar o licență pentru a-l folosi. Această licență este supusă unor termeni și condiții. Multii dezvoltatori consideră că rularea jocului pe platforma GeForce Now echivalează cu redistribuirea acestuia de către Nvidia, chiar dacă cel care folosește efectiv programul deține deja licență pentru uz personal.

Potențialele pierderi de care se tem companiile software sunt acelea ale profitului rezultat din cumpărarea de către același utilizator a mai multe licențe pentru același joc, aşa cum este cazul în prezent (multii gameri dețin atât versiunea pentru PC, cât și una pentru Playstation sau Xbox).

Pe de altă parte, majoritatea consumatorilor consideră că, din moment ce dețin o licență pentru joc, ar trebui să o poată folosi de oriunde, nefiind nevoiți să plătească de fiecare dată când doresc să ruleze jocul pe altă platformă. Mai mult, pasionații de

jocuri video par să credă că Nvidia nu face decât să le închirieze hardware-ul pe care ei își rulează jocurile și că nu au nevoie de permisiunea dezvoltatorului pentru a face aceasta.

Acstea diferențe de opinie demonstrează că influența conceptului de Cloud Gaming asupra industriei jocurilor video are consecințe mult mai mari decât pare la prima vedere, având potențialul de a schimba nu doar modul în care utilizatorii accesează jocurile, ci și felul în care se face distribuția acestora de către dezvoltatori.

Capitolul 2

Detalii de implementare

După cum spuneam și în cele câteva fraze unde am explicitat ce m-a motivat să încep acest proiect, am ales să dezvolt o aplicație de game streaming pentru Android, deoarece am observat lipsa de alternative pentru aceasta platformă în domeniul menționat. AndroidGameStreamer nu își propune să facă parte din categoria aplicațiilor de cloud gaming, ci mai degrabă să devină o soluție pentru pasionatul de jocuri ce detine o librărie digitală extinsă, de care dorește să dispună și atunci când nu se află în fața unei mașini Windows suficient de puternice pentru a rula jocurile. Deși efectuarea input-ului de pe un telefon cu Android nu este cea mai facilă soluție, o astfel de implementare profită de portabilitatea acestor dispozitive, putând fi folosită oricând și oriunde există o conexiune decentă la Internet.

Pentru dezvoltarea acestei idei a fost necesară folosirea a numeroase tehnologii printre care se numără sisteme de operare și API-uri ale acestora, 3 limbaje de programare, sisteme de management a bazelor de date și protocoale de comunicare în rețea.

2.1 Tehnologii folosite

2.1.1 Java

Java este o tehnologie concepută în 1995 de către James Gosling în cadrul companiei americane Sun Microsystems (parte a Oracle Corporation începând cu 2010). Denumita în prima fază Oak, iar mai apoi Green, Java cuprinde atât un limbaj de pro-

gramare de nivel înalt, cât și numeroase platforme destinate dezvoltării aplicațiilor pe aproape orice fel de dispozitiv (PC, smart-carduri, telefoane inteligente, servere etc.). Acesta a fost obiectivul principal încă de la prima versiune, fapt dovedit de motto-ul "write once, run anywhere".

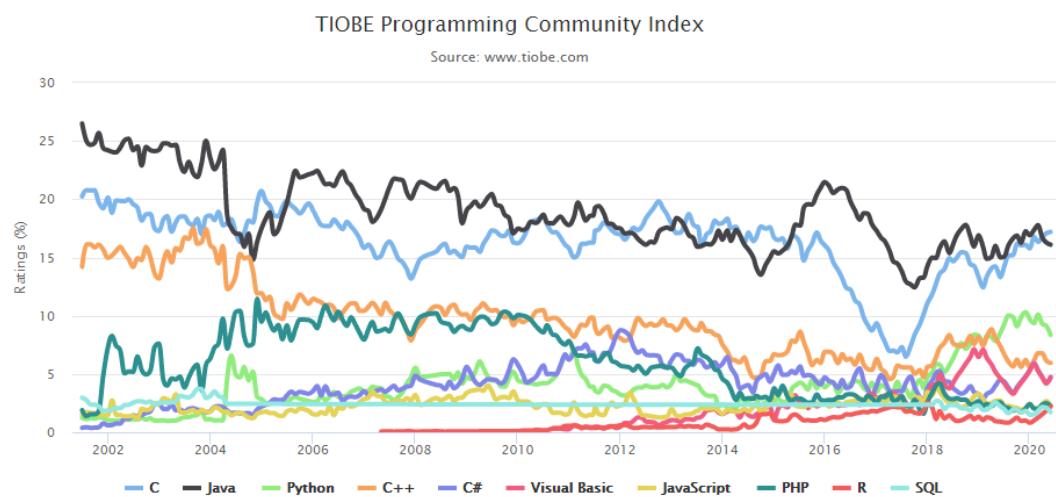
Există 4 astfel de platforme Java:

- Java SE (Standard Edition) - platforma standard
- Java EE (Enterprise Edition) - aplicații pentru sisteme distribuite
- Java FX - aplicații desktop și web
- Java ME (Micro Edition) - aplicații pentru telefoane, imprimante și orice fel de dispozitiv ce are la bază un microcontroller

Câteva dintre caracteristicile limbajului Java sunt următoarele:

- Sintaxa asemănătoare cu C și C++
- Securitate ridicată
- Complet orientat pe obiecte
- Atât compilat, cât și interpretat, ceea ce îl face rapid și portabil, prin eliminarea dezavantajelor precum viteza mică de execuție a limbajelor interpretate și lipsa de portabilitate a celor compile. Rezultatul compilării codului sursă scris în Java este așa-numitul byte-code (cod de octeți), acesta fiind interpretat de mașina virtuală Java care îl traduce în cod mașină specific arhitecturii pe care rulează.
- Suport solid pentru multithreading

Toate aceste avantaje au menținut Java în top 2 că popularitate încă din anul 2005, după cum dovedește și următorul grafic:



Am folosit Java în implementarea modulului pentru Android al aplicației AndroidGameStreamer, fiind limbajul cel mai popular pentru dezvoltarea aplicațiilor pe acest sistem de operare.

2.1.2 Python

Python este un limbaj de programare interpretat, de nivel înalt ce pune accent pe ușurința înțelegерii codului și dispune de o librărie standard extinsă, existând încă și mai multe module externe ce pot fi folosite pentru a implementa cele mai diverse funcționalități.

Am folosit Python 3.6.8 pentru realizarea serverului central și a unei părți însemnante a clientului pentru Windows.

2.1.3 C

Dezvoltat inițial în 1973 la laboratoarele Bell în vederea dezvoltării sistemului de operare Unix, C este un limbaj compilat de nivel jos, al cărui principal avantaj este viteza foarte mare de execuție. Aceasta caracteristică îl face și astăzi, la aproape 50 de ani de la prima apariție, unul dintre cele mai populare limbi de programare.

În cadrul aplicației AndroidGameStreamer a fost folosit pentru dezvoltarea modulului ce interpretează input-ul utilizatorului sub forma de evenimente simulate ale perifericelor.

2.1.4 SQLite

SQLite este o librărie scrisă în C care implementează un sistem de management al bazelor de date relationale. Conform dezvoltatorilor este cea mai utilizată astfel de aplicație din lume, majoritatea limbajelor de programare implementând API-ul său în librăria standard. Deoarece se află în totalitate în domeniul public am ales să folosesc SQLite pentru managementul bazei de date a aplicației AndroidGameStreamer deoarece se află în totalitate în domeniul public, existând în același timp un modul de Python (sqlite3) care implementează API-ul SQLite.

2.1.5 Protocolul TCP

Definit în documentul RFC 793, protocolul TCP stă, alături de protocolul IP, la baza majorității conexiunilor care au loc în Internet. TCP stabilește un canal de conexiune ce garantează transmisia unui flux de date fără erori, fără lipsuri și exact în ordinea în care ele au fost trimise, realizând aceasta prin implementarea unui sistem complex bazat pe pachete de confirmare.

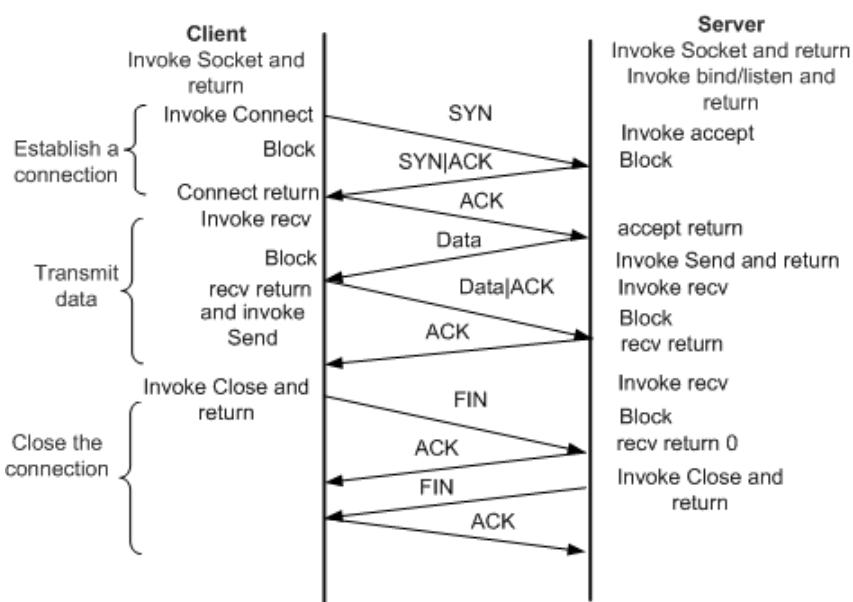


Figura 2.1: O conexiune TCP obișnuită

Am folosit protocolul TCP pentru transmiterea comenziilor între serverul central și cele două tipuri de clienți, fiind nevoie că acestea să ajungă de fiecare dată intacte pentru ca aplicația să funcționeze în modul dorit.

2.1.6 Protocolul UDP

Protocolul UDP este proiectat punând accent pe viteza de transmisie și pe simplitatea implementării. Pentru aceasta sacrifică garanția integrității datelor, renunțând la stabilirea conexiunilor și la pachetele de confirmare.

Am ales să folosesc UDP pentru transmisia fluxului video și a input-ului utilizatorului din motive ce țin în principal de reducerea latentei.

2.2 Arhitectură și implementare

Aplicația este formata din 3 componente principale:

- Serverul central
- Clientul pentru Windows
- Clientul pentru Android

2.2.1 CentralServer

Componenta CentralServer este o aplicație Python care implementează un server TCP concurrent. Acesta folosește porturile 20000 și 20001 pentru a primi cereri de conexiune din partea clientilor. Pentru a putea comunica cu mai mulți clienți în același timp, serverul creează un nou fir de execuție pentru a deservi fiecare conexiune în parte.

CentralServer este un intermediar între instanțele WindowsClient și AndroidClient. Acest lucru este necesar atât din motive de securitate (un utilizator nu se poate conecta decât la mașinile Windows pe care le-a înregistrat în prealabil că fiind ale sale, folosindu-și contul de utilizator), cât și de stabilire a unei conexiuni între cele două tipuri de aplicații client (este foarte probabil ca cel puțin una dintre instanțele modulelor client să se afle în spatele unui router care implementează NAT - Network Address Translation, ceea ce face dificilă realizarea directă a unei conexiuni TCP între clientul de Android și cel ce rulează pe PC).

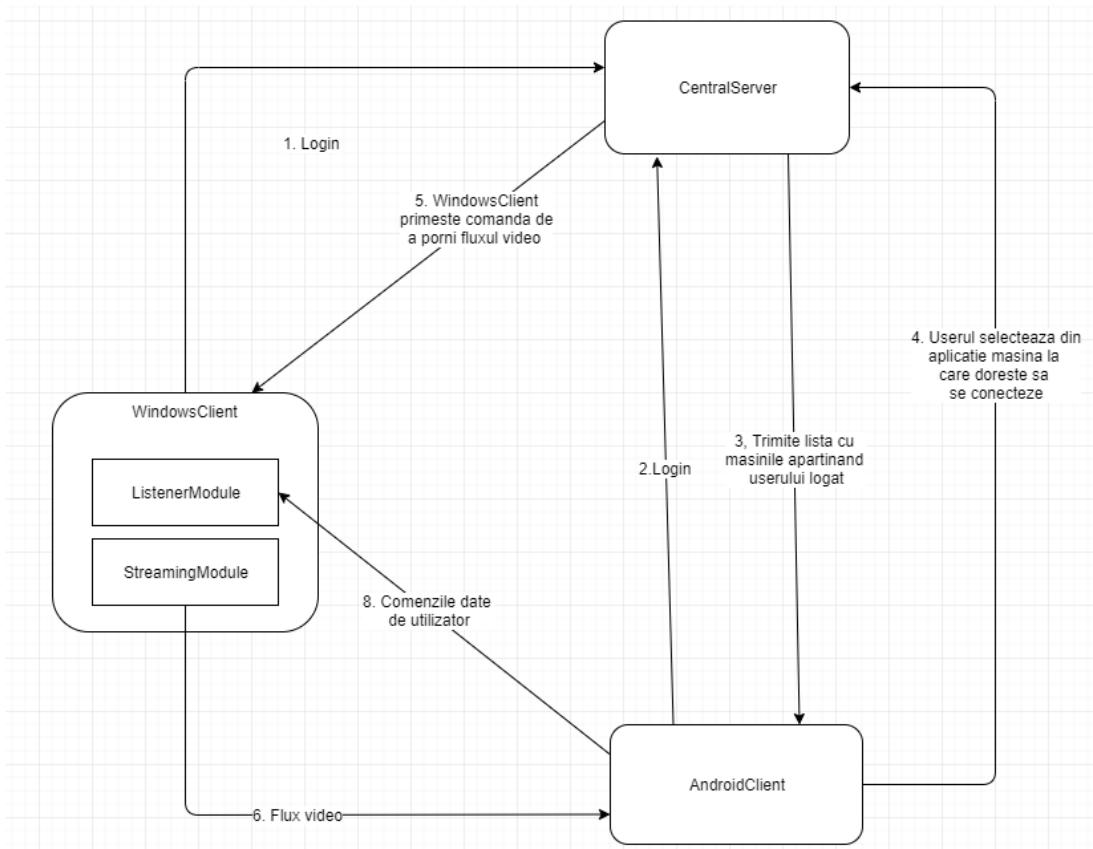


Figura 2.2: Arhitectura aplicației

Portul 20000 este utilizat de clienții care doresc să se autentifice (atât de cei de tip AndroidClient, cât și de instanțele WindowsClient). În cazul în care clientul furnizează date de logare corecte, îi este atribuit un token cu ajutorul căruia vă fi identificat de către server în cele ce urmează.

Clienții apelează portul 20001 pentru a comunica efectiv cu serverul. Imediat ce se realizează conexiunea TCP, clientul trebuie să furnizeze tokenul primit la login. În caz contrar, serverul vă închide conexiunea, iar clientul nu vă putea da nicio comandă.

2.2.2 WindowsClient

La prima rulare a clientului pentru Windows, utilizatorului i se cer datele de logare cu care s-a înregistrat în aplicația mobilă. Dacă autentificarea se realizează cu succes, clientul salvează într-un fișier JSON numele de utilizator și parola (criptată cu funcția hash SHA-256) și adaugă o cheie nouă în regisitrii Windows, astfel încât

aplicația vă pornește de fiecare dată odată cu sistemul, fiind gata să accepte conexiuni fară să fie nevoie să fie lansată în execuție de către utilizator. În acest fel, la următoarele rulări ale aplicației, logarea se va realiza automat. De asemenea, clientul primește și tokenul cu care vă fi identificat de către server în continuare.

În cazul în care utilizatorul dorește să ruleze un joc pe acest PC, la comanda data din aplicația Android, serverul va trimite o notificare către WindowsClient, inclusiv și adresa IP la care se află telefonul de la care s-a dat comanda (și la care va trebui trimis fluxul video).

În continuare, WindowsClient va lansa în execuție cele două submodule ale sale:

- StreamingModule: se ocupă de captura ecranului și de codificarea fluxului video pentru a economisi întârzierea de bandă și a mări performanța în cazul unei conexiuni la Internet de viteza mică. De asemenea, transmite fluxul video către AndroidClient, unde urmează să fie decodificat și afișat pe ecranul dispozitivului. StreamingModule este realizat în Python 3.
- ListenerModule: implementat în C++, are rolul de a primi și de a interpreta apăsările de butoane ale utilizatorului aplicației Android. Acestea sunt formate din coduri de octeți care furnizează informații cu privire la dispozitivul a cărei folosire trebuie simulată (mouse sau tastatură), tipul de eveneu (apăsare scurtă, menținere apăsării etc.) și tastă sau butonul efectiv care se dorește să fie acționat.

2.2.3 AndroidClient

Lansarea aplicației de Android ne aduce la o activitate de login. Utilizatorul introduce un username și o parolă care sunt trimise serverului pentru a fi comparate cu înregistrările aflate în baza de date. Parola este criptată cu funcția hash SHA-256, în acest mod asigurându-ne că nu poate fi interceptată în drumul său spre destinație.

În cazul în care nu avem un cont de utilizator, tot din activitatea inițială putem crea unul, ceea ce face că serverul să efectueze o instrucțiune de tip INSERT în propriul tabel cu conturi de utilizator.

Dacă logarea s-a realizat cu succes, AndroidClient descarcă din server o lista cu mașinile la care are acces utilizatorul curent. După ce acesta alege o mașină, se trimite

această informație la server, care se ocupă de comunicarea cu WindowsClient. După ce alege și aplicația pe care dorește să o lanseze în execuție, utilizatorului i se prezintă o altă activitate în cadrul căreia apar atât fluxul video de la mașina ce rulează Windows, cât și o serie de controale prin intermediul cărora poate interacționa cu aceasta și, implicit, cu jocul video care rulează pe aceasta.

2.2.4 StreamingModule: Compresia și transmiterea fluxului video

În acest sens, este utilizat protocolul UDP, fiind cel mai potrivit deoarece este mai rapid. Aceasta se datorează faptului că nu este nevoie de realizarea unei conexiuni înainte de a putea trimite pachetele cu informația ce trebuie transmisă. De asemenea, headerul UDP are un overhead mai mic (8 octeți vs. minim 20 de octeți în cazul headerului TCP), ceea ce ajută la economisirea lățimii de banda.

De asemenea, latența este mult redusă prin folosirea UDP, dat fiind faptul că nu este nevoie să așteptăm primirea pachetelor de tip ACK (pachete de confirmare), cum s-ar întâmplă dacă am alege să utilizăm protocolul TCP.

Deoarece UDP nu garantează primirea pachetelor de către destinatar, merită explicat pe scurt ce se întâmplă atunci când unele pachete se pierd în rețea. Spre deosebire de, să spunem, transmiterea unui document HTML (atunci când dorim să vizualizăm o pagină web - caz în care este nevoie ca informația să ajungă completă, nemodificată și exact în ordinea în care a fost trimisă), în aplicațiile de streaming video, câteva pachete lipsă vor cauza artefacte (glitch-uri grafice) sau, cel mult, lipsă unor cadre din fluxul video primit de destinatar.

Dacă am folosi TCP, deși suntem siguri că aceste artefacte și cadre lipsă vor fi evitate, apar alte probleme mult mai serioase. De exemplu, dacă se pierd câteva pachete de confirmare, protocolul TCP trebuie să retransmitem informația, care face parte dintr-un frame video care a fost deja afișat (și prin urmare este inutilă). Deoarece cadrele noi nu se pot trimite decât după ce acelea vechi au fost retransmise (dacă este cazul), fluxul video poate rămâne în urmă, iar utilizatorul nu vă poate vedea ce se întâmplă în momentul de față pe mașina care rulează jocul.

Pentru perspectivă, până și o latență de 120 ms este considerată prea mare pentru a avea o experiență de gaming adecvată. Prin urmare, utilizarea protocolului TCP

cauzează mai multe probleme decât rezolvă, și de aceea am ales să folosesc UDP pentru transmiterea fluxului video.

În vederea efectuării capturii de ecran am folosit librăria Python **d3dshot**, care reprezintă un wrapper pentru un subset al API-ului Windows (mai exact Desktop Duplication API). d3dshot este capabil să captureze imagini ca obiecte de tip PIL.Image (PIL - Python Imaging Library).

Pentru a face aplicația utilizabilă într-un scenariu real, trebuie ca aceasta să aibă nevoie de o lățime de banda cât mai mică pentru a transmite imaginea. Să considerăm în continuare următorul scenariu: dorim să transmitem un flux video la rezoluție Full HD (1920x1080 pixeli) cu un framerate de 60 fps (cadre pe secunda). Imaginele captureate de la ecran sunt inițial în format RGB (pentru fiecare pixel se stochează intensitatea culorilor roșu, verde și albastru, fiecare sub forma unui număr între 0 și 255, adică pe 8 biți).

Prin urmare, pentru stocarea informației unui singur cadru avem nevoie de:

$$1920 * 1080 * 8 * 3 = 4976640 \text{biți} = 4.98 \text{Mbiți}$$

În fiecare secundă trebuie trimise pe rețea 60 de cadre, deci lățimea de banda consumată vă fi:

$$4.98 * 60 = 298.8 \text{Mbps},$$

viteză care în majoritatea cazurilor nu se atinge nici prin Wi-Fi, ca să nu mai vorbim de conexiunea prin date mobile.

De aceea, pentru ca aplicația să poată fi utilizată în practică, avem nevoie să comprimăm informația înainte de a o trimite în rețea. Pentru aceasta putem folosi unul dintre numeroasele codec-uri video existente. În cazul aplicației AndroidGameStreamer am folosit codecul MJPEG (Motion JPEG), acesta fiind relativ simplu de implementat, deoarece fiecare cadru este comprimat cu ajutorul algoritmului de compresie pentru imagini JPEG (de unde și denumirea), implementat în cadrul PIL (Python Imaging Library). Tot ce trebuie făcut apoi pentru a obține fișierul video comprimat este să concatenăm imaginile în format JPEG rezultate.

Conform testelor pe care le-am făcut, efectuând o singură captură a ecranului (la rezoluție 1080p) și comprimând-o cu algoritmul JPEG, rezultă fișiere ce ocupă între 100 și 210 KB (0.8-1.7 Mbiti), în funcție de factorul de calitate specificat la compresie și de complexitatea imaginii. Comparativ cu dimensiunea unei imagini necomprimate (4.98 Mbiti), se poate observa o economie de spațiu (sau a lățimii de bandă necesare în cazul în care dorim să trimitem informația prin rețea) de 65-85%. Desigur, se pot obține factori de compresie chiar mai mari, caz în care degradarea calității imaginii devine perceptibilă.

După compresie, tot ce rămâne de făcut în cazul aplicației AndroidGameStreamer este să trimitem cadrul prin rețea. Chiar și o imagine comprimată este prea mare pentru un singur pachet UDP, deci aceasta este împărțită în mai multe pachete diferite, fiecăruia asociindu-se un număr de identificare, ce va fi folosit pentru a determina la recepționare dacă unele părți ale cadrului s-au pierdut pe rețea. Modulul de Android al aplicației va recepționa aceste pachete și va reconstrui imaginea ce urmează a fi afișată pe ecran. În cazul în care unele părți ale cadrului au ajuns în altă ordine sau s-au pierdut (se poate întâmpla deoarece protocolul UDP nu verifică integritatea datelor trimise), cadrul respectiv nu va fi afișat, deoarece algoritmul JPEG nu suportă pierderi de date și, prin urmare, cadrul respectiv nu ar putea fi reconstituit.

2.2.5 ListenerModule: transmisia și interpretarea input-ului efectuat de utilizator

Când vine vorba de transmisia input-ului dat de utilizatorul modulului pentru Android, pierderea unor pachete în rețea nu este o problemă (atâtă timp cât nu este excesivă), pentru că modulul AndroidClient este proiectat în aşa fel încât să furnizeze update-uri frecvente către ListenerModule (aproximativ la fiecare 5-10 ms) cu privire la ce butoane sunt apăsate de către utilizator. Prin urmare, pierderea unui pachet nu va cauza decât o întârziere minimă a rezultatului acțiunii pe care utilizatorul dorește să o întreprindă.

În cazul simulării unor apăsări de butoane, o latență mare creează tot atâtea probleme ca și întârzierea fluxului video, dat fiind că utilizatorul nu poate vedea imediat rezultatele acțiunilor sale. Este evident că o asemenea situație ar cauza o experiență de

gaming precară.

Deci și în acest caz protocolul UDP este cel mai potrivit (din motivele amintite mai sus), deoarece urmărим iarăși să avem latență minimă posibilă.

Utilizatorul are trei modalități de a controla jocul de pe telefonul cu Android:

- un joystick virtual în partea din stânga jos a ecranului, folosit pentru mișcarea caracterului (echivalentul tastelor W, A, S, D în joc)
- jumătatea dreaptă a ecranului, programată să recunoască gesturile utilizatorului; este folosită la schimbarea perspectivei (echivalentul mișcărilor de mouse)
- diferite butoane virtuale prin care se pot realiza alte acțiuni necesare în joc, de exemplu tras cu arma, săritură, aruncarea unei grenade etc.

```
static final byte MOUSE_TAP = 0x11;
static final byte MOUSE_HOLD = 0x12;
static final byte MOUSE_RELEASE = 0x13;
static final byte MOUSE_MOVE = 0x14;
static final byte KEYBOARD_TAP = 0x21;
static final byte KEYBOARD_HOLD = 0x22;
static final byte KEYBOARD_RELEASE = 0x23;
```

Figura 2.3: Tipurile de evenimente suportate de aplicație

Având aceste informații primite de la AndroidClient, ListenerModule poate simula orice eveniment ar putea fi generat folosind o tastatură sau un mouse. Numărul de pixeli cu care trebuie mișcat mouse-ul pe axe X și Y este calculat înmulțind viteza gestului făcut de utilizator pe ecranul telefonului cu o valoare a sensibilității ce poate fi modificată în funcție de preferințe.

În ceea ce privește simularea apăsării tastelor, aceasta are la baza funcția SendInput din cadrul API-ului Windows. Funcția respectivă ia ca argument structuri specifice care oferă o flexibilitate mare în definirea evenimentului care se dorește a fi simut. Evenimentele artificiale sunt inserate direct în fluxul de input al tastaturii sau, după caz, al mouse-ului, deci au efect global, nefiind nevoie să selectăm prin parametri

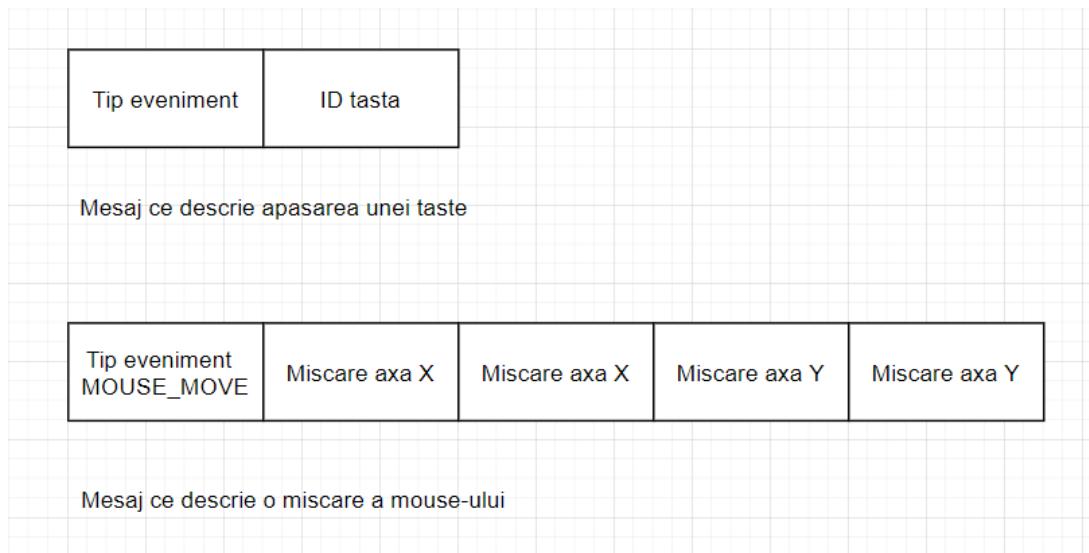


Figura 2.4: Mesajele cu privire la evenimente (fiecare dreptunghi reprezintă un octet)

aplicația care va primi evenimentele (acestea sunt trimise către fereastra care are focus în momentul respectiv), ceea ce ar face implementarea mai dificilă.

Capitolul 3

Scenarii de utilizare

3.1 Aplicația pentru Windows

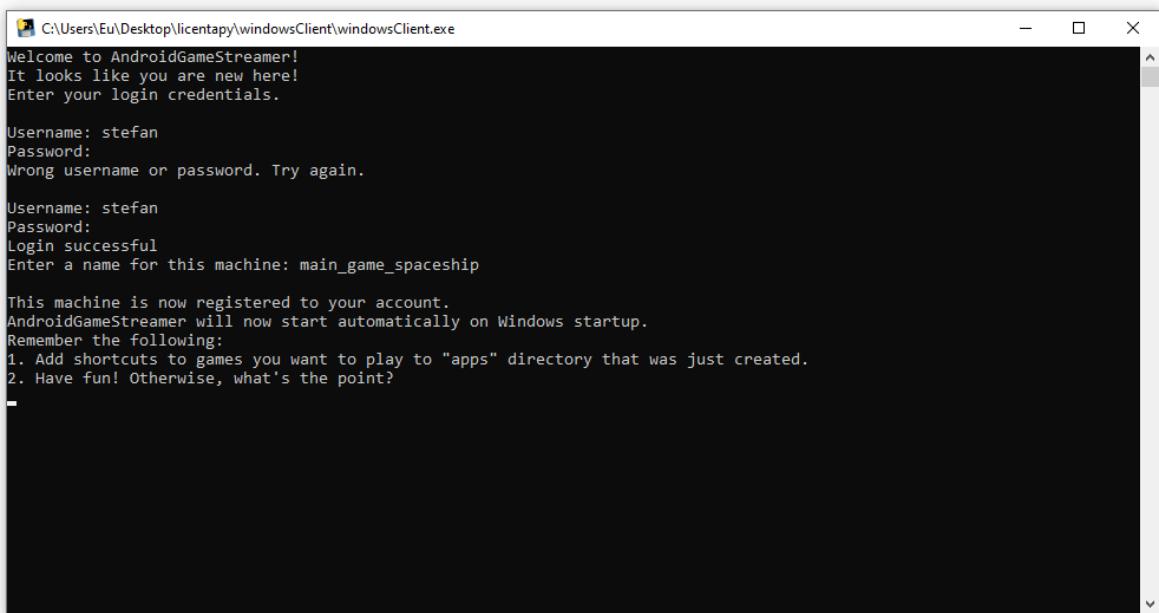


Figura 3.1: Logarea de pe o mașină după instalarea programului

Aplicația pentru Windows rulează în consolă. Această abordare a fost aleasă pentru simplitatea implementării și nu afectează experiența utilizatorului, dat fiind că acesta are nevoie să se logheze o singură dată după instalarea aplicației. Dacă acesta s-a autentificat cu succes, mașina este înregistrată de server ca aparținând utilizatorului curent. Pe viitor, aplicația va porni odată cu sistemul de operare și se va autentifica automat cu credențialele furnizate inițial, fiind gata să ruleze jocuri fără a fi nevoie de

niciun fel de input din partea utilizatorului.

Această funcționalitate simplifică mult interacțiunea cu utilizatorul. Pentru a se putea juca, tot ce are acesta de făcut este să se asigure că mașina ce urmează să ruleze jocul este pornită și conectată la Internet.

Pentru a putea porni de la distanță un joc, în directorul "apps" al modulului de Windows trebuie să existe un shortcut către acesta. Lista cu shortcut-urile disponibile în acest director, mai exact lista jocurilor ce pot fi rulate va fi trimisă către AndroidClient astfel încât utilizatorul să poată alege jocul pe care dorește să îl ruleze.

3.2 Aplicația pentru Android

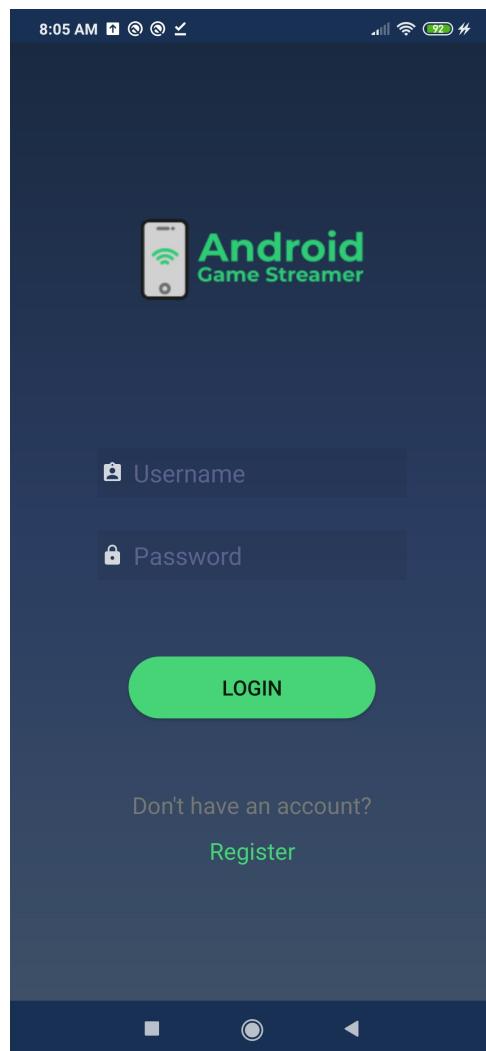


Figura 3.2: Ecranul de login

La pornirea aplicației pe telefon apare ecranul de login.

Daca nu avem cont, apăsam pe butonul "Register" și suntem aduși pe ecranul de înregistrare, unde ne putem crea unul.

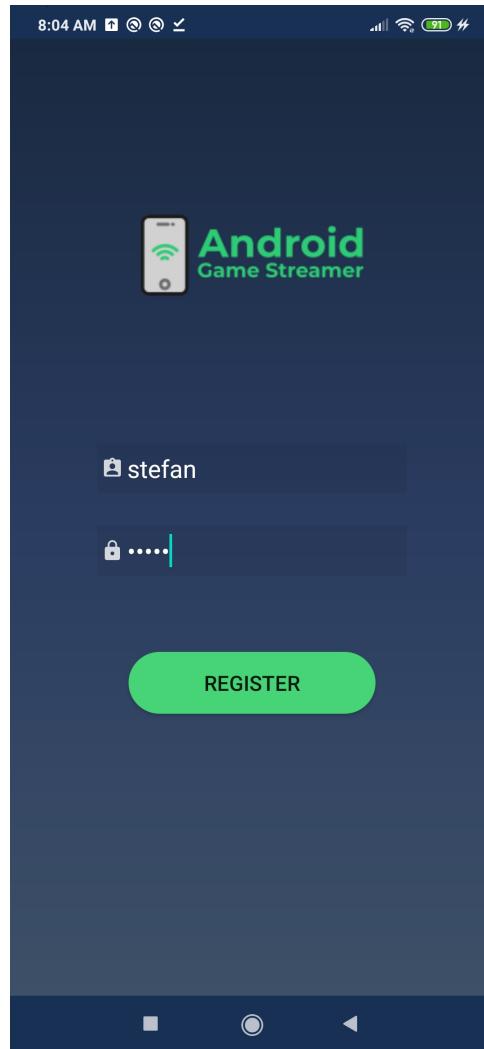


Figura 3.3: Ecranul de înregistrare

Dacă logarea s-a realizat cu succes, suntem aduși pe un nou ecran în care apar mașinile înregistrate sub acest cont de utilizator (aceleia pe care s-a logat cu acest cont). Ele sunt marcate cu verde dacă sunt online, sau cu roșu în caz contrar.

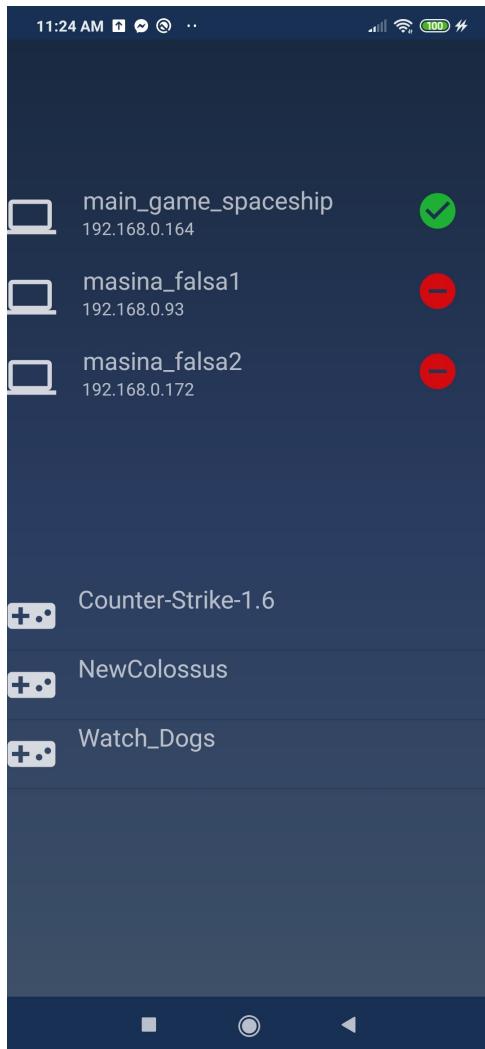


Figura 3.4: Mașinile și jocurile disponibile pentru utilizatorul curent

Apăsând pe oricare dintre ele se afișează în josul paginii lista cu aplicațiile ale căror shortcut-uri se află în directorul "apps" pe mașină respectivă. Putem alege oricare dintre acestea, caz în care se pornește fluxul video și jocul selectat. Suntem aduși pe o alta pagină, pe care putem vedea imaginea de pe ecranul mașinii și unde dispunem de controalele necesare pentru joc.

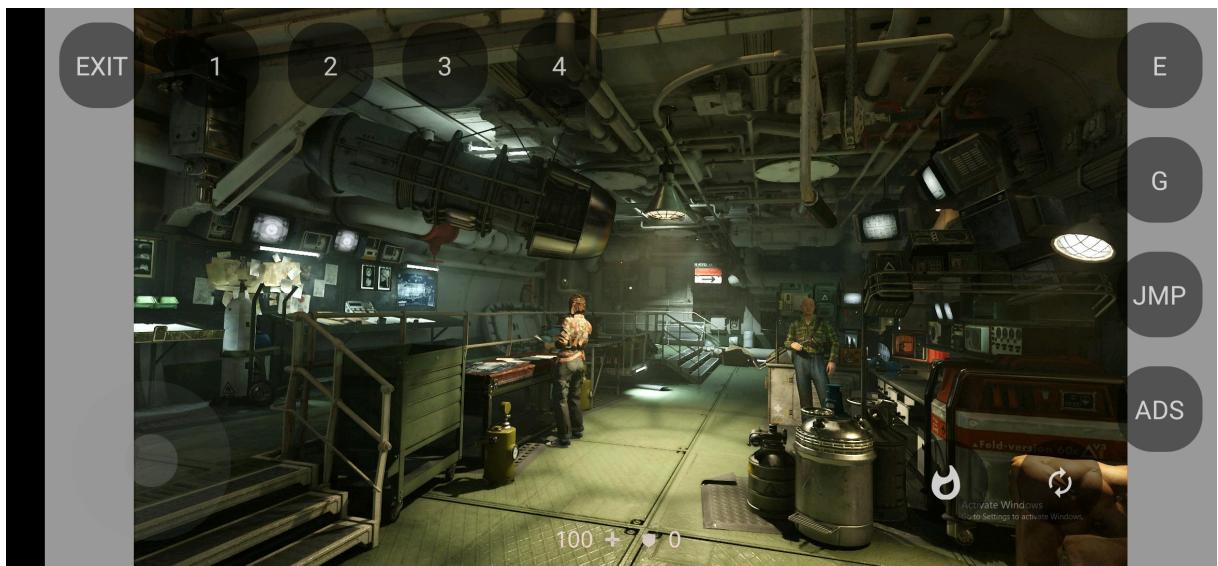


Figura 3.5: Jocul Wolfenstein II: The New Colossus în aplicatie

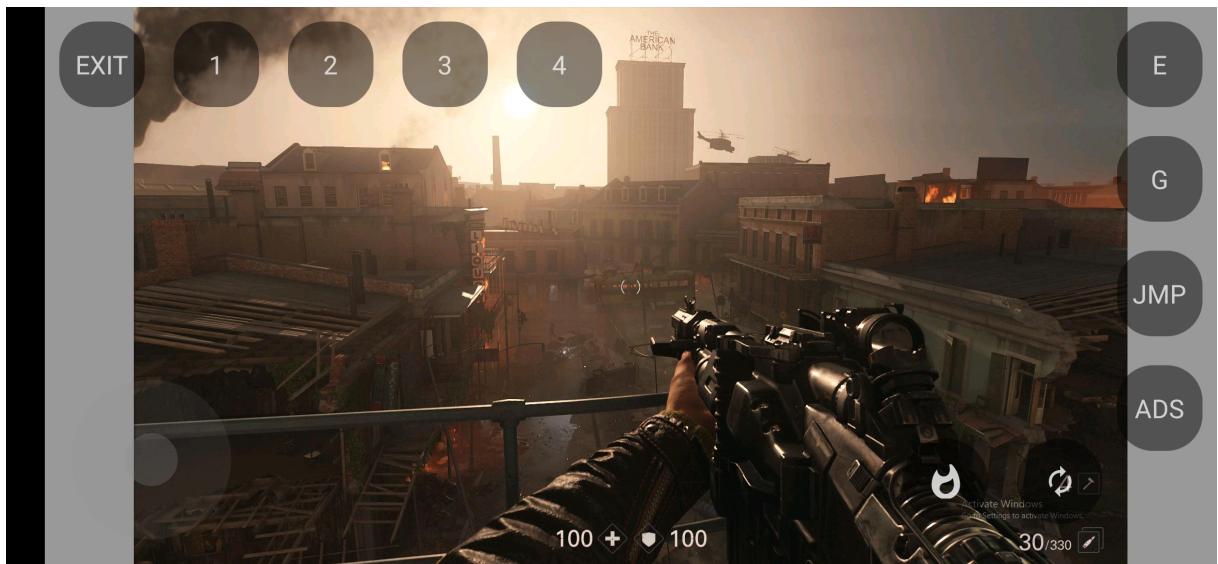


Figura 3.6: Jocul Wolfenstein II: The New Colossus în aplicatie

Capitolul 4

Direcții de dezvoltare

4.1 Suport pentru controller

Într-un joc de tip First Person Shooter, există 3 acțiuni care în foarte multe situații trebuie realizate simultan: mișcarea caracterului, mișcarea perspectivei (pentru a putea ținti) și trasul cu arma. Dacă folosind mouse-ul și tastatura acestea sunt ușor de realizat, nu același lucru se poate spune despre ecranul unui telefon cu Android. Având la dispoziție butoanele virtuale de pe acesta din urmă, putem realiza oricare două acțiuni simultan, dar niciodată pe toate trei. Am observat că în acest caz performanța jucătorului scade dramatic.

Soluția la acest inconvenient este să implementăm în aplicația de Android suport pentru conectarea controllerelor externe. Gamepad-urile de tipul celor folosite de consola de jocuri Xbox a Microsoft se pot conecta atât prin cablu, cât și wireless, fiind compatibile cu mai multe tipuri de dispozitive. Începând cu versiunea API 9, platforma Android oferă suport pentru astfel de controlere, făcând implementarea avan-tajoasă atât din punctul de vedere al simplității, cât și a impactului pe care l-ar avea spre îmbunătățirea experienței de joc. Într-un joc de tip First Person Shooter, există 3 acțiuni care în foarte multe situații trebuie realizate simultan: mișcarea caracterului, mișcarea perspectivei (pentru a putea ținti) și trasul cu arma. Dacă folosind mouse-ul și tastatura acestea sunt ușor de realizat, nu același lucru se poate spune despre ecranul unui telefon cu Android. Având la dispoziție butoanele virtuale de pe acesta din urmă, putem realiza oricare două acțiuni simultan, dar niciodată pe toate trei. Am observat

că în acest caz performanta jucătorului scade dramatic.

Soluția la acest inconvenient este să implementăm în aplicația de Android suport pentru conectarea controllerelor externe. Gamepad-urile de tipul celor folosite de consola de jocuri Xbox a Microsoft se pot conecta atât prin cablu, cât și wireless, fiind compatibile cu mai multe tipuri de dispozitive. Începând cu versiunea API 9, platforma Android oferă suport pentru astfel de controlere, făcând implementarea avantajoasă atât din punctul de vedere al simplității, cât și a impactului pe care l-ar avea spre îmbunătățirea experienței de joc.

4.2 Fluxul video

Codecul MJPEG are numeroase avantaje (simplitatea implementării, calitate superioară a imaginii, mai ales în scenele cu multă mișcare). Totuși, are un dezavantaj major care îl face să nu fie codecul optim în aplicațiile de game streaming.

În primul rând, conform observațiilor mele, aplicația (folosind codecul MJPEG) are nevoie de o lățime de bandă de cel puțin 50 Mbps pentru transmiterea fluxului video. Deși în general fluxul video se transmite cu succes atunci când folosim aplicația fiind conectați la Wi-Fi, nu același lucru poate fi spus despre conexiunile de date mobile. De asemenea, chiar dacă România beneficiază de conexiuni la Internet de până la 1 Gbps deosebit de accesibile, alte țări se confrunta cu preturi mult mai mari pentru același tip de servicii. Prin urmare, mulți consumatori aleg un abonament ce oferă viteză mică (de multe ori mai puțin de 50 Mbps necesari aplicației AndroidGameStreamer), dat fiind că este mai puțin costisitor.

Acest neajuns poate fi rezolvat prin folosirea unui alt codec video, de exemplu H.264. Aceasta atinge o rată de compresie a fluxului video mult mai mare, având nevoie de o lățime de bandă de doar 10 Mbps pentru un flux video Full HD (1080p) la 60 de cadre pe secundă.

Totuși, implementarea acestuia în cadrul unei aplicații de game streaming este relativ dificilă, din cel puțin două motive:

- Nu există suport nativ în librăria standard Android pentru codecul H.264 decât în cazul în care pentru transmisia datelor se folosește protocolul HTTP sau RTSP

(Real Time Streaming Protocol). Primul poate fi folosit pentru vizionarea fișierelor video înregistrate în prealabil, dar este nepotrivit pentru streaming-ul în timp real. Prin protocolul RTSP nu se poate transmite fluxul video în mod peer-to-peer, fiind nevoie de un server intermediu la care să se conecteze ambele tipuri de clienți (cei care generează fluxurile video și cei care le consumă).

Dezvoltând aplicația, am testat protocolul RTSP, obținând rezultate deloc mulțumitoare. Chiar și într-un mediu de testare în care toate componentele aplicației se aflau în aceeași rețea locală, latența fluxului video ajungea până la 2 secunde, caz în care este absolut imposibil ca utilizatorul să se poată juca.

Dacă doream să transmit imaginea folosind protocolul UDP, aş fi fost nevoit să implementez personal o parte importantă a procesului de decodificare a fluxului video comprimat (există librarii low-level de Android care pot decodifica H.264, dar este nevoie de separarea unităților componente ale fluxului comprimat numite NAL - Network Abstraction Units. Acestea pot ocupa o parte dintr-un segment UDP, un segment întreg sau mai multe segmente.) Este evident deci că implementarea ar deveni foarte complicată.

- Multe dintre librăriile de C++ ce implementează codecul H.264 dispun de API-uri a căror documentație este neclară sau, mai rău, total inexistentă, ceea ce ar face dificilă și implementarea codificării imaginii pe mașina Windows.

Există desigur și opțiunea de a folosi programe separate pentru codificare (de exemplu FFmpeg sau VLC), dar acestea măresc fără a fi necesar dimensiunea clientului pentru Windows al aplicației de game streaming, dat fiind că dispun de numeroase funcționalități care nu sunt de folos în acest scenariu.

De asemenea, pentru cazul în care conexiunea la Internet este mai puțin rapidă, aplicația ar beneficia și de setarea adaptivă a lătimii de bandă folosite. În acest sens, dat fiind că telefonul are un ecran mic, putem sacrifica temporar rezoluția fluxului video pentru a obține latență mai mică și un framerate mai bun, dat fiind că acestea sunt mult mai importante pentru o experiență bună de game streaming decât calitatea imaginii, mai ales în cazul de față (imagină vă fi proiectată pe telefon, care dispune de un ecran relativ mic, deci o scădere a rezoluției are șanse mai mari să treacă neobservată decât dacă am fi folosit pentru proiectare un laptop).

4.3 Fluxul audio

În prezent, aplicația AndroidGameStreamer nu suportă transmisia fluxului audio. Totuși, acest lucru este necesar pentru o experiență de gaming adekvată. O modalitate de implementare ar fi capturarea sunetului de la mașina ce rulează jocul, codificarea cu codecul MP3 pentru economisirea lățimii de bandă necesare, și trimitera prin UDP către aplicația de Android unde urmează a fi decodificat și trimis către difuzor, având grija să fie sincronizat cu fluxul video.

4.4 Setări în aplicația Android

Deși controalele standard disponibile în aplicația de Android sunt suficiente pentru a juca aproape orice joc de tip First Person Shooter, de multe ori tastă ce trebuie acționată pentru a realiza o anumită acțiune diferă de la un joc la altul. De aceea, este necesară implementarea unei pagini de setări. De aici utilizatorul vă poate schimba tastă a carei apăsare este simulață prin atingerea butoanelor virtuale din aplicație. De asemenea, trebuie făcută posibilă adăugarea de noi butoane în cazul în care jocul este mai complex, sau, dimpotrivă, eliminarea butoanelor de care nu este nevoie pentru a evita supraaglomerarea ecranului telefonului.

Alte setări utile ar fi modificarea sensibilității inputului pentru jumătatea dreaptă a ecranului, aceea care se ocupă de simularea mișcării mouse-ului, sau simularea apăsării tastei SHIFT automat atunci când joystick-ul virtual este împins mai mult în față. În majoritatea jocurilor tastă SHIFT face caracterul să alerge, prin urmare este folosită des. De aceea, unii utilizatori ar prefera nu un buton dedicat care să simuleze apăsarea acestei taste, ci simularea ei tot din joystick-ul de mișcare, în funcție de poziția acestuia.

Concluzii

AndroidGameStreamer nu este nici pe departe o aplicație matură care ar putea fi lansata pe piață la momentul actual, dat fiind că are numeroase neajunsuri. Cele mai supărătoare dintre acestea sunt framerate-ul scăzut (până la 20fps) și latența destul de ridicată.

Aplicația reprezintă totuși rodul cercetării pe care am întreprins-o și pe parcursul căreia am învățat o mulțime de lucruri. Pot spune chiar că a fost nevoie să mă informez despre concepte care la momentul în care mi-a venit ideea acestui proiect nu păreau a avea vreo legătură cu el.

Cred că realizarea acestei lucrări de licență a fost un prilej nemaipomenit să îmbin pasiunea mea pentru jocurile video cu cercetarea în mai multe ramuri ale informaticii, reușind să-mi mențină interesul viu pe tot parcursul timpului în care am lucrat la implementare.

Bibliografie

- Cristian Frasinaru, *Curs practic de Java*
- <https://www.androidauthority.com/how-to-store-data-locally-in-android-app-717190/>
- <http://stupidpythonideas.blogspot.com/2013/05/sockets-are-byte-streams-not-message.html>
- <https://docs.microsoft.com/en-us/windows/win32/api/winuser/nf-winuser-sendinput>
- <http://www.philipstorr.id.au/pcbook/book3/scancode.htm>
- <https://developer.android.com/>
- <https://github.com/zerokol/JoystickView>
- <https://www.electronicsforu.com/technology-trends/news/the-benefits-of-cloud-gaming>
- <https://www.forbes.com/sites/erikkain/2019/11/26/google-stadia-review-the-good-the-bad-and-the-ugly/#513d014e6cc1>
- <https://www.androidauthority.com/google-stadia-review-1055351/>
- <https://www.pcgamer.com/heres-how-stadias-input-lag-compares-to-native-pc-gaming/>
- <https://www.theverge.com/2020/3/2/21161469/nvidia-geforce-now-cloud-gaming-service-developers-controversy-licensing>
- <https://www.cnet.com/reviews/nvidia-geforce-now-review/>
- <https://www.sqlite.org/index.html>

Surse pentru imagini:

- <https://www.techradar.com/reviews/google-stadia-controller>
- <https://www.tomsguide.com/news/google-stadia-is-finally-free-how-to-try-it-now>
- <https://www.pcworld.com/article/3257484/nvidia-geforce-now-pc-system-reqs-price-features-performance.html>
- <https://support.huawei.com/enterprise/en/doc/EDOC1100094294/2c6f7bd5/tcp>
- <https://app.diagrams.net/> (pentru diagrame)