

faceSPACE

Aplikácia pre videokonferencie

Projektová dokumentácia

Riešitelia

Bc. Kristína Hrebeňárová

Bc. Miloslav Smičík

Bc. Peter Strýček

Verzia 1.0

Obsah

1	Úvod	1
2	Návrh softvérového riešenia	2
3	Implementácia softvérového riešenia	6
4	Používateľská príručka	11
5	Vyhodnotenie	17

1 Úvod

V úvodnej časti dokumentu sa venujeme definícii fundamentálnych cieľov projektu a opisu štruktúry dokumentu.

Ciele projektu

Primárnym cieľom projektu je vytvorenie funkčnej webovej aplikácie pre vytváranie videokonferencií, ktorá umožní užívateľom interagovať v reálnom čase cez video či zvuk. Pri vývoji aplikácie sa snažíme dosiahnuť vysokú mieru modulárnosti a znovupoužiteľnosti kódu pomocou využitia návrhových vzorov, vhodného použitia moderných technológií a dodržiavaním princípov čistého kódu.

Skrátený zoznam nami definovaných funkcionálnych a nefunkcionálnych požiadaviek na softvér je nasledovný.

1. Prívetivé a intuitívne používateľské rozhranie.
2. Jednoduchá správa a parametrizácia videokonferencií.
3. Zdieľanie multimediálneho obsahu v reálnom čase.
4. Vytváranie záznamu priebehu videokonferencií.
5. Snaha o minimalizáciu latencie prenosu.
6. Modulárny charakter softvérového riešenia, ktoré zabezpečí jednoduché rozšírenie aplikácie podľa budúcich potrieb.

Štruktúra a formát dokumentu

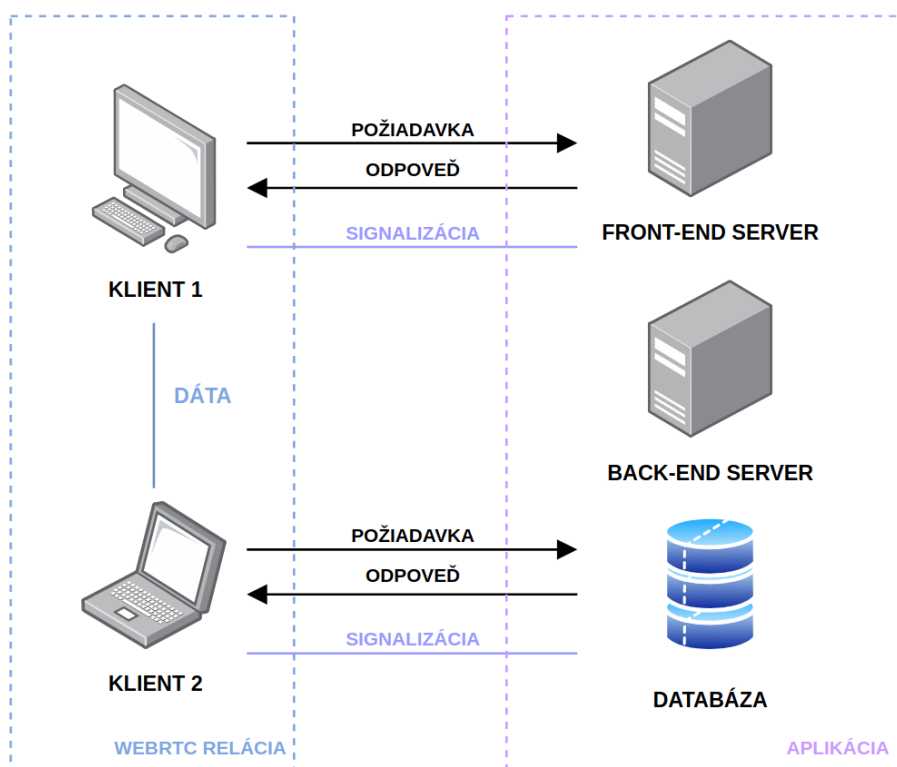
Návrhu architektúry a dizajnu softvérového riešenia sa venujeme v dedikovanej kapitole [2](#). Technické detaily implementácie softvéru sú opísané v kapitole číslo [3](#). Predposledná kapitola (číslo [4](#)) slúži ako používateľská príručka. Záver dokumentu obsahujúci vyhodnotenie miery splnenia definovaných cieľov a požiadavok je v kapitole číslo [5](#).

2 Návrh softvérového riešenia

Pri návrhu nášho softvérového riešenia sme kládli vysoký dôraz na využitie vyhovujúcich návrhových vzorov. Základom nášho návrhu je architektúra **Model-View-Controller**, ktorá vhodne separuje softvérové súčasti do troch hlavných typov:

1. **Model**. Zahŕňa triedy a entity reprezentujúce aplikačné dáta.
2. **View**. Zodpovedá za prezentáciu dát klientom.
3. **Controller**. Spracováva prichádzajúce požiadavky a rozhoduje, ktoré akcie je potrebné vykonať. Dáta získava z modelu a spracované ich posiela naspäť používateľovi.

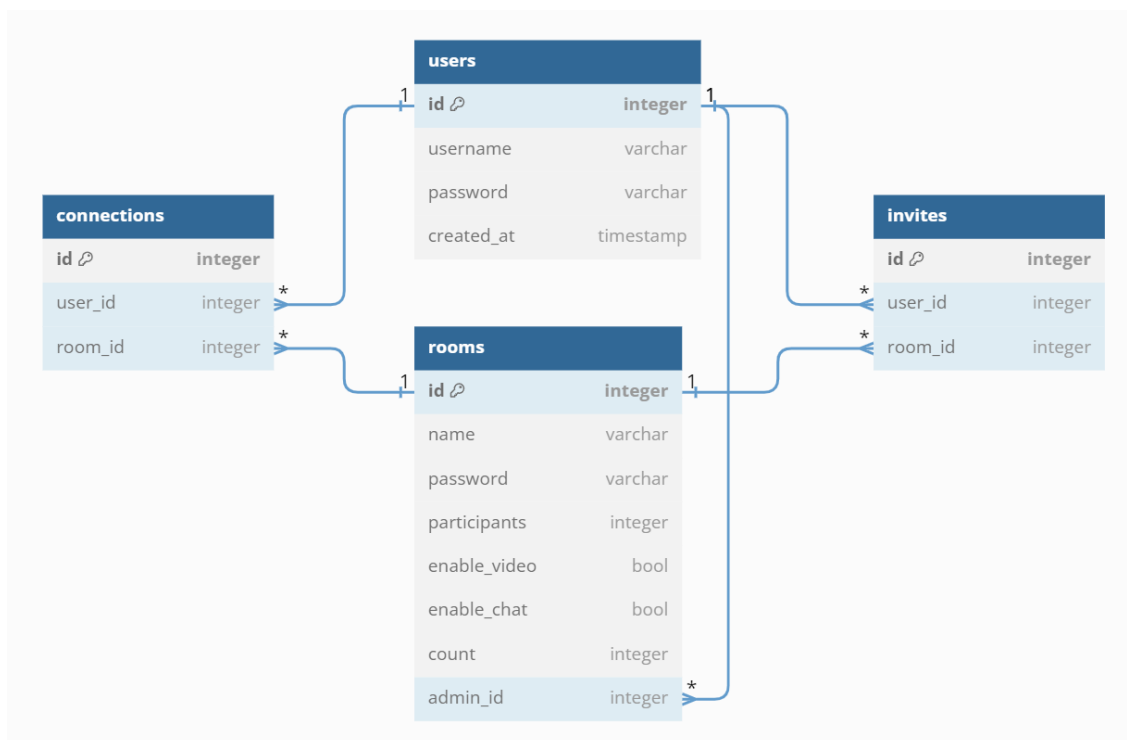
Abstraktný a zjednodušený návrh architektúry systému je zobrazený na obrázku číslo 2.1. V ňom predstavuje databáza *Model*, front-end server *View*, a back-end server *Controller*. Diagram architektúry takisto zahŕňa aj integráciu **WebRTC**, ktorá nám bude slúžiť na komunikáciu, textovú aj audiovizuálnu, medzi používateľmi v reálnom čase.



Obr. 2.1: Návrh architektúry systému faceSPACE.

Návrh modelu databázy

Zoznam jednotlivých entít, ktoré v systéme identifikujeme a udržiavame, spolu s ich vzájomnými vzťahmi je zobrazený na diagrame na obrázku číslo 2.2.



Obr. 2.2: Databázový model entít.

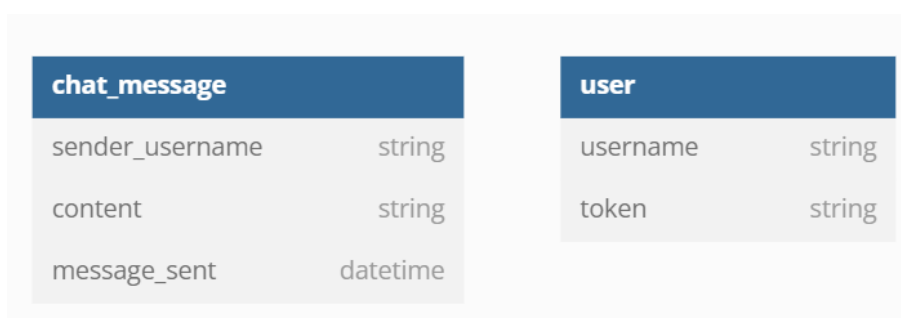
Hlavným aktérom v našom systéme je používateľ – **User**. V databáze o ňom budeme udržiavať informácie ako je jeho prihlasovacie meno (**username**), heslo (**password** v plain-texte, pretože bezpečnosť systému v prvej iterácii nie je našou prioritou) a dátum registrácie (**created_at**).

Ďalšou kľúčovou súčasťou sú konferenčné miestnosti – **Room**. O tých si v systéme udržiavame ich názov (**name**), prípadné heslo (**password**), maximálny akceptujúci počet účastníkov konferencie (**participants**), počet aktuálne pripojených používateľov (**count**), či príznaky **enable_video**, resp. **enable_chat**, ktoré obmedzujú prenos na iba zvukový, resp. zamedzia textovej komunikácii pripojených používateľov v podobe chatu. Tiež si udržiavame informáciu o tom, kto miestnosť založil (**admin_id**).

Administrátor miestnosti bude môcť pri jej vytváraní pozvať ostatných používateľov aplikácie. Dáta o tom, kto koho a kam pozval budeme udržiavať v tabuľke pozvánok – **invites**.

Jednotlivé WebRTC spojenia používateľov a konferenčných miestností budeme udržiavať v tabuľke **connections**.

Okrem entít opísaných vyššie v systéme ešte evidujeme ďalšie entity, ktoré však nemusia byť perzistované v databáze. Ide napríklad o prihláseného používateľa a chatovú správu (obrázok číslo 2.3).



Obr. 2.3: Databázový model entít.

Modelovanie funkcionalít systému faceSPACE

Všetky funkcionality, ktoré vychádzajú z vyššie definovaných funkcionálnych a nefunkcionálnych požiadavok (sekcia číslo 1), sme namodelovali pomocou kohézneho diagramu prípadov použitia. Ten je zobrazený na obrázku číslo 2.4.

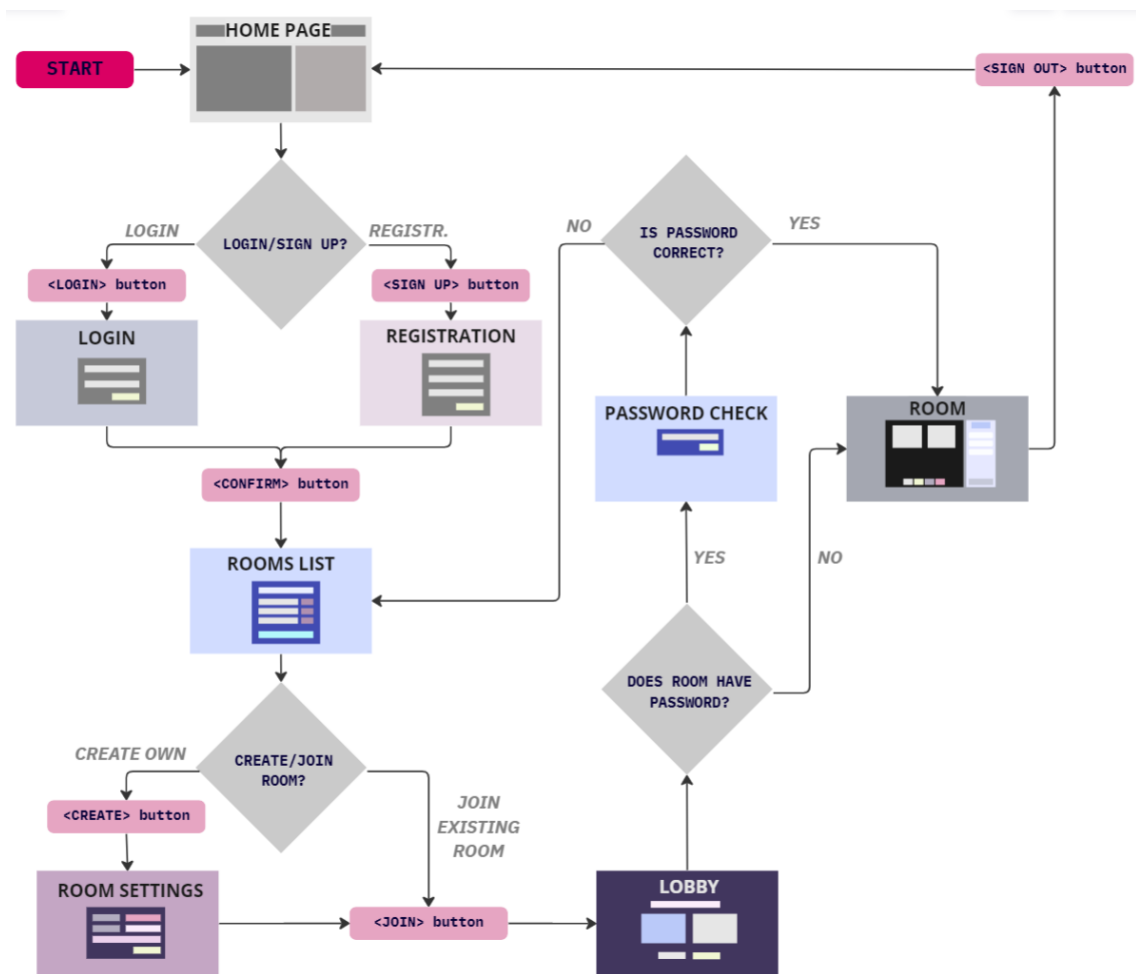


Obr. 2.4: Diagram prípadov použitia.

Popis jednotlivých funkcionalít je nasledovný:

1. **Registrácia používateľa.** Túto funkciu môže využiť len neprihlásený a ešte nezaregistrovaný používateľ. Pri registrácii sa použije dedikovaný registračný formulár a entita používateľa sa uloží do databázy.
2. **Prihlásenie sa do systému.** Funkcionalita je určená už registrovaným používateľom, ktorí vyplnia prihlasovací formulár, ktorého validita sa skontroluje na back-ende. V rámci úspešného prihlásenia nastaví back-end klientovi token prihláseného používateľa.
3. **Odhlásenie sa zo systému.** Prihlásenému používateľovi zmaže token a tým ho zo systému odhlási.
4. **Vytvorenie konferencie.** Ľubovoľný používateľ bude môcť vytvoriť vlastnú videokonferenčnú miestnosť pomocou dedikovaného formulára, v ktorom používateľ bude môcť miestnosť parametrizovať (prípád **Nastavenie parametrov**) a pozvať ostatných používateľov – prípad **Pridávanie účastníkov**.
5. **Pripojenie sa do konferencie.** Prihlásený používateľ sa bude môcť napojiť do miestnosti (pokiaľ tak počet jej aktívnych účastníkov dovoľuje). V rámci korektného prístupu do miestnosti bude musieť zadať heslo (ak je nastavené) – prípad **Zadanie hesla pre prístup**. Počas hovoru bude môcť používateľ zazdieľať obsah svojej obrazovky (prípád **Zdieľanie obsahu obrazovky**) alebo nahrávať záznam stretnutia – prípad **Nahrávanie záznamu**, ktorý sa po ukončení nahrávania automaticky stiahne.

Návrh funkcionalít sme namodelovali aj pomocou *user flow* diagramu. Diagram je vizualizovaný na obrázku číslo 2.5



Obr. 2.5: User flow diagram.

3 Implementácia softvérového riešenia

V nasledujúcej sekcii popíšeme detaily implementácie jednotlivých súčasti softvéru. Zahrnieme popis implementácie jednotlivých funkcionalít na back-ende a front-ende. Postupne opíšeme inovatívne knižnice, ktoré nám umožnili vytvoriť real-time komunikáciu a videohovory.

Technologický stack

Pre implementáciu nášho riešenia sme vybrali nasledovný zoznam technológií.

- **Front-end:** TypeScript, Angular, WebRTC
- **Back-end:** C#, .NET, ASP.NET Core, Entity Framework
- **Databáza:** EF Core In-Memory Database

Real-time komunikácia server-klient - SignalR

SignalR je ASP.NET knižnica zabezpečujúca komunikáciu v reálnom čase medzi serverom a klientom. V našej aplikácii sa využíva na správu používateľov v miestnosti a zabezpečuje aj funkcionality chatu. Samotná implementácia je rozdelená na dve časti, ktoré spolu komunikujú - back-end a front-end.

Back-end SignalR

Implementácia na back-ende takzvaného **SignalR hubu** (trieda implementujúca úlohy SignalR) sa nachádza v súbore **SignalR/ConferenceHub.cs**. V tejto triede sa nachádzajú základné funkcionality SignalR knižnice, ako napríklad pripájanie a odpájanie klienta. Klient je pripojený na Hub iba ak sa nachádza v miestnosti. A teda začiatok a koniec pripojenia sa viažu na odpojenie a pripojenie do konkrétnej miestnosti. Celý Hub implementuje nasledujúce metódy:

1. **OnConnectedAsync()** - metóda má za úlohu spracovať a zaznamenať pripojenie klienta do určitej miestnosti. Identifikátor miestnosti, do ktorej sa klient chce pripojiť sa nachádza v *query parametri* prichádzajúcej požiadavky. Používateľ je autentifikovaný pomocou **JWT HTTP-only cookie** a následne je v kontexte požiadavky uložené jeho používateľské meno, ktoré je v metóde následne extrahované. Po úvodnom získaní potrebných dát z požiadavky sa zavolá metóda **JoinRoom()** z perzistentnej časti aplikácie. Táto metóda skontroluje, či je používateľ oprávnený pripojiť sa do miestnosti a ak áno tak vytvorí v databáze pripojenie (**Connection**) a inkrementuje sa počet aktívnych klientov v miestnosti. Následne sa klient oficiálne pridá do Hub skupiny, ktorá prináleží danej miestnosti. Následne sa všetkým pripojeným používateľom v miestnosti aktualizuje pomocou SignalR správy zoznam pripojených používateľov a samotný novopripojený používateľ je zaznamenaný do miestnosti.
2. **OnDisconnectedAsync()** - metóda má na starosti korektné odpojenie používateľa z miestnosti. Zavolá sa vždy keď je SignalR spojenie prerušené alebo ukončené. Metóda

si získa potrebné údaje podobne ako `OnConnectedAsync()` a následne zavolá metódu `LeaveRoom()` z perzistentnej časti aplikácie, ktorá odstráni prislúchajúcu `Connection` z databázy a dekrementuje počet klientov v danej miestnosti.

3. `SendMessage()` je metóda zabezpečujúca funkcionality chatu v miestnosti. Používateľ jednoducho pošle obsah správy a metóda zistí, v ktorej miestnosti sa nachádza, vytvorí objekt správy s obsahom, autorom a časom. Tento objekt je následne v reálnom čase poslaný každému používateľovi v rovnakej miestnosti.

Front-end SignalR

Samozrejme s vyššie opísaným backend riešením musí existovať klient, ktorý je schopný s knižnicou SignalR komunikovať. Na tento účel sme použili `npm` balík `@microsoft/signalr`, ktorý implementuje funkcionality určené na komunikáciu so SignalR Hubom na backend-e. V súbore `conference-hub.service.ts` sa nachádza samotná implementácia komunikujúca s backend-ovým naprítovníkom. Samotná implementácia začína metódou `createHubConnection()`, ktorá slúži na navedenie komunikácie s backend-om. Nastavuje sa v nej pripojenie a hooky na dve správy, ktoré backend implementácia môže potencionálne poslať klientovi:

1. `NewMessage` - Správa obsahuje novú správu, ktorú poslal jeden z používateľov do chatu v miestnosti.
2. `AllConnected` - Správa, ktorá sa pošle vždy po pripojení nového používateľa do miestnosti a aktualizuje aktuálny zoznam klientov v miestnosti.

Ďalšia metóda spravuje ukončenie pripojenia a proste pripojenie ukončí a prípadne vypíše chybu do konzoly. Posledná metóda slúži na odoslanie správy do chatu, kde sa vyvolá `SendMessage` správa spolu s obsahom správy.

Integrácia WebRTC - peer.js

Na funkcionality videohovorov sme použili obľúbenú knižnicu využívajúcu WebRTC, ktorá sa volá `peer.js`. Samotná knižnica je schopná veľmi jednoducho vytvoriť **peer2peer spojenia** a prenášať napríklad obraz a zvuk vo forme *streamu*. Keďže knižnica funguje cez `peer2peer`, back-end v tomto procese priamo nefiguruje. Knižnica sa používa veľmi jednoducho.

Celý systém prepojení je robený len na základe identifikátorov vo forme stringu a netreba riešiť technické detaily ako sieťové spojenia a parametre. Knižnica funguje pomocou napojenia sa na sprostredkovací server, ktorý je defaultne *PeerServer Cloud* od vývojárov knižnice. Server slúži na sprostredkovanie spojenia medzi dvoma peer-mi, ktorí následne komunikujú priamo medzi sebou - p2p (prípadne cez TURN server ak p2p nie je možný, napríklad v prípade, že obaja účastníci sú za NAT).

Samotná implementácia sa začne okamžite po pripojení do miestnosti. V súbore `room.component.ts` sa vytvorí nový objekt `Peer`, ktorému sa nastaví hook na hovor - pri prijatí konkrétneho hovoru sa volajúcemu pošle stream a samotnému hovoru nastavia hooky na stream a zatvorenie spojenia. Stream slúži na prijatie prichádzajúceho streamu, ktorý je následne zobrazený v miestnosti ako video iného používateľa. Hook na zatvorenie spojenia tento stream vymaže.

Druhá časť implementácie slúži na pripojenie k ostatným používateľom - prakticky vyvolá hooky opísané v predošlom odseku u ostatných klientov. Prakticky funguje tak, že každému zo zoznamu používateľov v miestnosti zavolá spolu so streamom vlastného videa. Vytvorí taktiež hook na odpoveď streamu rovnako ako v inicializácii, ktorý prichádzajúci stream zobrazí v miestnosti.

Zdieľanie obrazovky

Aplikácia taktiež umožňuje jednotlivým účastníkom miestnosti zdieľať svoju obrazovku. Na tieto účely je potrebné vytvoriť nové peer2peer spojenia na zdieľanie nového separátneho streamu. Streamy sú zobrazené podobným spôsobom ako web kamery, t.j. html tag `<video>` obsahuje property `srcObject` s daným streamom typu `MediaStream`.

V súbore `room.component.ts` sa podobne ako pri hovore vytvorí objekt `Peer`, ktorý očakáva správu v podobe `screen_` + meno používateľa. V prípade, že tomuto používateľovi dôjde stream v tomto peer2peer spojení, ide o zdieľanie obrazovky iného účastníka. Taktiež môže prísť správa o ukončení streamu toho účastníka, ktorý zdieľal obrazovku.

Obrazovku je možné zdieľať iba ak nikto iný už nezdieľa. Ak nie je taká situácia a účastník sa rozhodne zdieľať obrazovku, pomocou funkcie `getDisplayMedia` rozhrania `MediaDevices` sa vyžiada povolenie na snímanie a následne výber čo presne chce účastník zdieľať – napríklad celú obrazovku, kartu v prehliadači a okno. Potom po výbere sa snímanie reprezentované ako stream rozpošle všetkým účastníkom. Podobne pri ukončení streamu sa táto informácia rozpošle každému.

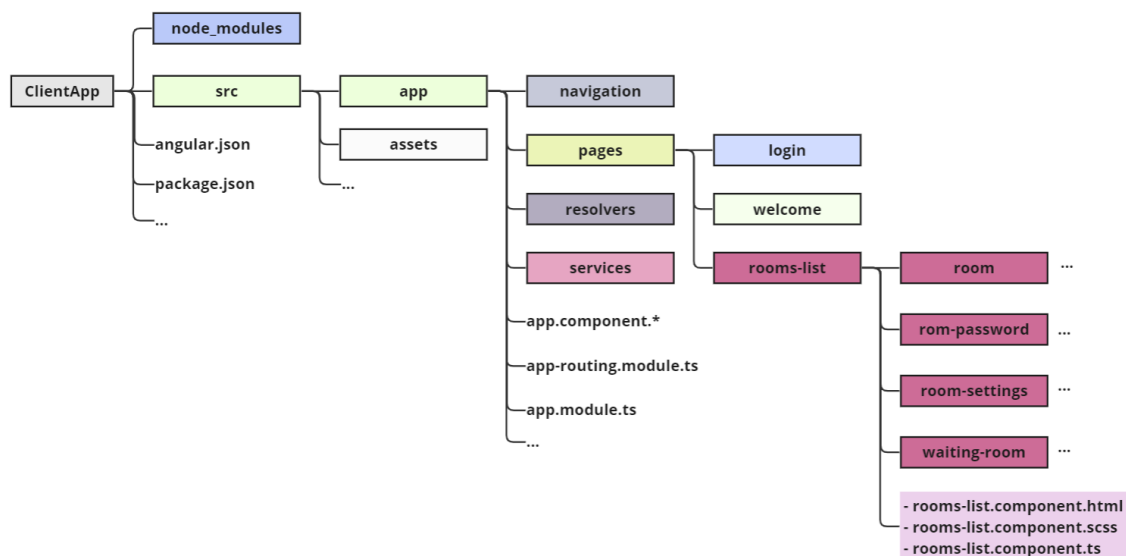
Nahrávanie

Nahrávanie je umožnené rozhraním `MediaRecorder` vo formáte WebM, pretože je **podporovaný** takmer v každom bežnom prehliadači. Jednotlivé dáta prichádzajúce z tohto rozhrania sa ukladajú do buffera. Po skončení nahrávania sa nahrávka účastníkovi uloží ako súbor opäť vo formáte WebM.

Používateľské rozhranie

Používateľské rozhranie zastrešuje front-end server. Ten sme implementovali v modernom frameworku **Angular** (v16.1). Ide o open-source framework vyvíjaný spoločnosťou Google a zároveň jeden z najpoužívanejších moderných frameworkov pre vývoj **Single Page Applications** (SPA) a dynamických webových stránok.

Implementácia je rozdelená do modulov a komponentov. Štruktúra projektu odzrkadľuje aj samotnú štruktúru webovej stránky. Každá podstránka vychádza z vlastného dedikovaného Angular komponentu, v ktorom je implementovaná jej funkcionálna (component.ts) a aj jej vzhľad (component.html resp. component.scss). Štruktúra front-endu je zachytená pomocou diagramu súborovej štruktúry na obrázku číslo 3.1.



Obr. 3.1: Štruktúra zdrojových súborov projektu.

API na back-end je implementovaná v jednotlivých servisoch (napr. **RoomService** v zdrojovom súbore `room.service.ts`). Tu pomocou `HttpClienta` vysielame rôzne druhy HTTP požiadavok (GET, PUT, DELETE, a POST) na náš back-end server, ktorý ich obsluhuje.

Pre efektívne načítavanie dát ešte pred finálnym zobrazením podstránky, používame **resolver**. Resolver poskytuje spôsob ako načítať potrebné dáta ešte pred samotnou inicializáciou webovej stránky a to ešte počas procesu presmerovania. Takýto resolver máme implementovaný v zdrojovom súbore `room.resolver.ts`. Vďaka nemu načítame detaily o konkrétnej konferenčnej miestnosti, ktoré sa následne v používateľskom rozhraní zobrazia zdanlivo bez čakania.

Dátové objekty (entity) sa na front-end prenášajú v podobe *Data Transfer Objects* (DTOs), ktoré sú definované ako *interface*.

Externé knižnice

V priečinku `node_modules` sa nachádzajú externé knižnice. Zoznam použitých externých knižníc je v súbore `package.json`. Okrem knižníc podporujúcich Angular, používame aj nasledovné:

- **fontawesome-free** - knižnica pre podporu ikon v používateľskom rozhraní,
- **ngx-bootstrap** - knižnica, ktorá integruje **Bootstrap** do Angular projektu,
- **ngx-webcam** - minimalistická knižnica pre management webkamier,
- **primeng** - obširna knižnica UI komponentov,
- **primeicons** - knižnica ikon.

Testovanie

Aplikáciu sme testovali priebežne. Testovanie prebiehalo v náročných podmienkach a to kvôli charakteru samotnej aplikácie, ktorá si na overenie funkčnosti vyžaduje súbežné pripojenie viacerých používateľov. Napriek tomu sa nám podarilo otestovať funkcionality konferencií v jednej konferenčnej miestnosti s viacerými (3) používateľmi. Aplikáciu sme mali spustenú

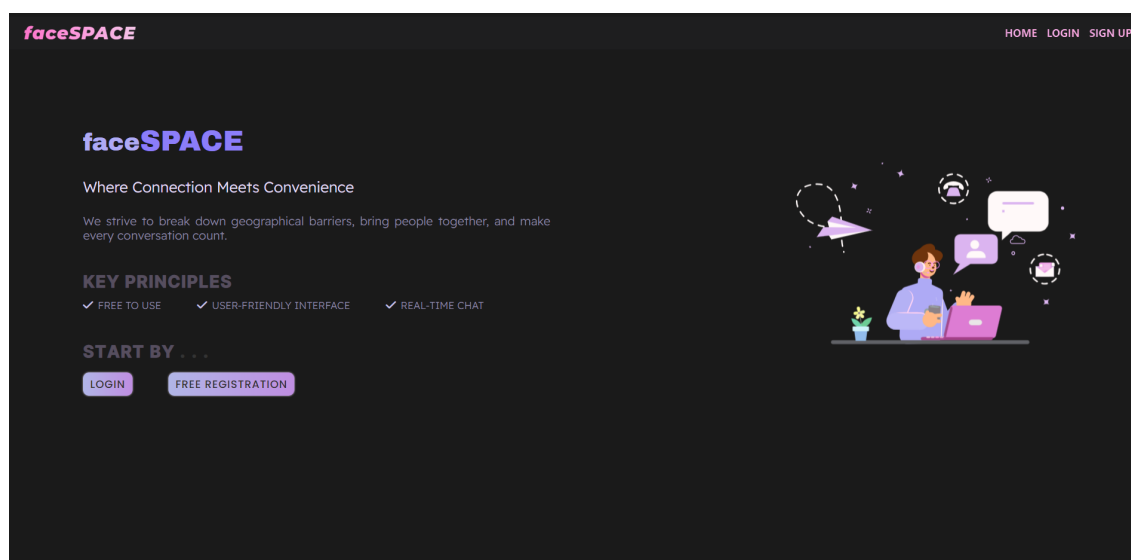
iba lokálne. Kvôli zdieľaným cookies sme museli pri testovaní použiť 2 rôzne webové prehliadače (Google Chrome (v normálnom a inkognito móde) a Edge). Limitujúcim faktorom pri testovaní bolo aj jedno zariadenie integrovanej kamery, ktoré sa nedalo prepoužiť naprieč viacerými prehliadačmi. To sme obišli implementáciou virtuálnej kamery cez OBS.

Dokázali sme overiť funkčnosť pripojenia používateľa (admina) a ostatných používateľov do miestnosti. Takisto sme overili funkčnosť zdieľania obrazovky, posielania správ v chate, či vytvárania záznamu z webkamery používateľa.

4 Používateľská príručka

Postup pre spustenie aplikácie je v súbore `READ.ME`. Popisuje komponenty potrebné pre beh aplikácie a spôsob inštalácie externých knižníc, ktoré sa využívajú na back-ende a front-ende. Táto časť dokumentu popisuje komponenty používateľského rozhrania a odporúčané použitie webovej aplikácie **faceSPACE**.

Hneď po spustení je používateľ presmerovaný na domovskú stránku – *Home page*. Stránka obsahuje úvodný text, kľúčové aspekty aplikácie a animáciu. Screenshot obrazovky je na obrázku číslo 4.1.



Obr. 4.1: Domovská stránka webovej aplikácie – *home page*.

Neprihlásený používateľ má v menu v navigácii možnosť novej registrácie a prihlásenia sa. Registračný formulár je zachytený na obrázku číslo 4.2 a variant formuláru pre prihlásenie používateľa je na obrázku číslo 4.3. V prípade, že používateľ svoje heslo zabudne, môže ho resetovať – obrázok číslo 4.4.

The screenshot shows the 'faceSPACE' website with a dark theme. In the top left corner is the 'faceSPACE' logo in pink. In the top right corner are the links 'HOME', 'LOGIN', and 'SIGN UP' in white. The main content area features a 'REGISTRATION' form centered on the page. The form has a title 'REGISTRATION' in white. It contains three input fields: 'Username', 'Password', and 'E-mail', each with a light gray placeholder. Below the 'E-mail' field are two buttons: a light blue 'LOGIN' button and a darker blue 'SIGN UP' button.

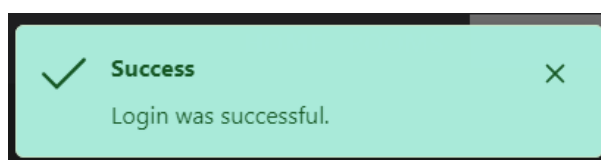
Obr. 4.2: Registračný formulár.

The screenshot shows the 'faceSPACE' website with a dark theme. In the top left corner is the 'faceSPACE' logo in pink. In the top right corner are the links 'HOME', 'LOGIN', and 'SIGN UP' in white. The main content area features a 'LOGIN' form centered on the page. The form has a title 'LOGIN' in white. It contains two input fields: 'Username' and 'Password', each with a light gray placeholder. Below the 'Password' field is a link 'FORGOTTEN PASSWORD?' in white. At the bottom of the form are two buttons: a light blue 'SIGN UP' button and a pink 'LOGIN' button.

Obr. 4.3: Formulár pre prihlásenie používateľa.

Obr. 4.4: Formulár pre resetovanie zabudnutého hesla používateľa.

Používateľské rozhranie je obohatené o notifikácie signalizujúce úspešné či neúspešné vykonanie akcie iniciovanej používateľom. Príklad takej notifikácie je na obrázku číslo 4.5.



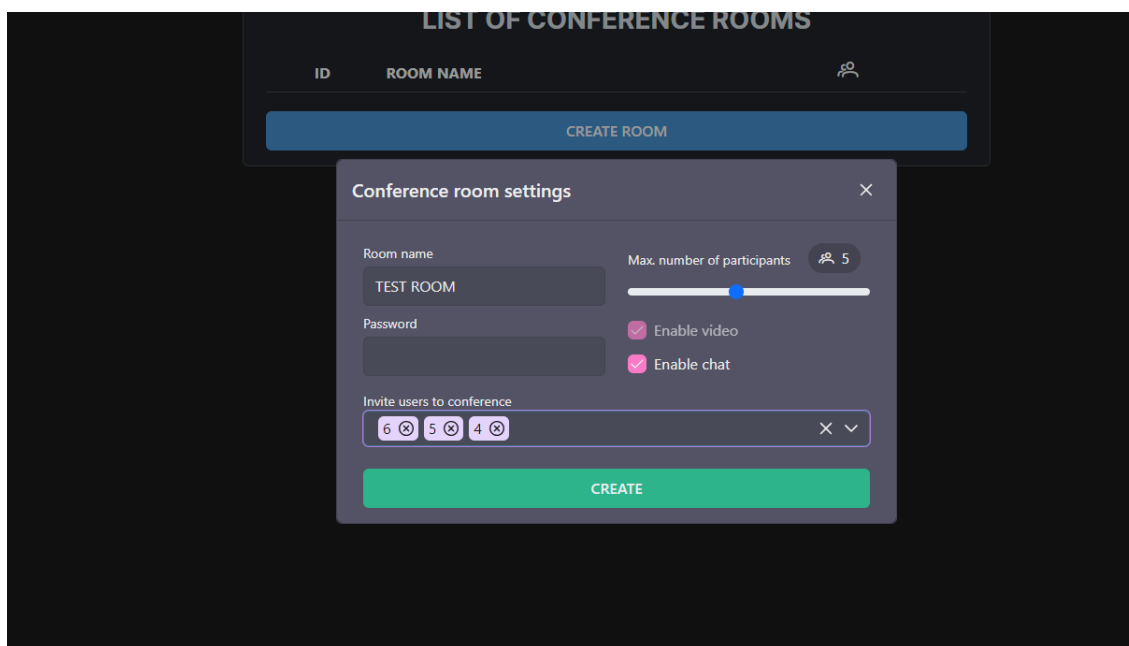
Obr. 4.5: Sprievodná notifikácia signalizujúca úspešné prihlásenie používateľa.

Po prihlásení je používateľ presmerovaný na stránku venovanú zoznamu konferenčných miestností. Obrazovka je na obrázku číslo 4.6.

Obr. 4.6: Zoznam existujúcich miestností.

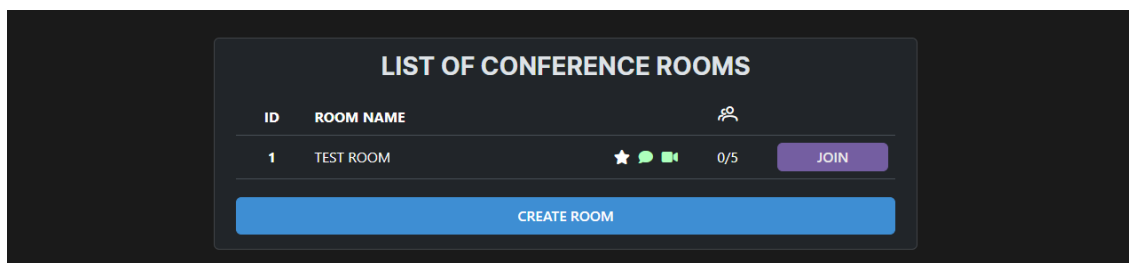
Každý používateľ má možnosť vytvoriť si vlastnú miestnosť. Formulár v dialógovom okne je zobrazený na obrázku číslo 4.7. V ňom vie používateľ špecifikovať unikátny názov a nepovinné heslo miestnosti, maximálny počet účastníkov a takisto má možnosť vypnutia chatovej

funkcionality. Vie tiež pozvať jednotlivých používateľov z globálneho zoznamu používateľov.

The image shows a 'Conference room settings' modal window. At the top, it has a title bar with a close button. Below the title bar, there are three input fields: 'Room name' (containing 'TEST ROOM'), 'Max. number of participants' (a slider set to 5), and 'Password'. To the right of the 'Max. number of participants' slider, there are two checkboxes: 'Enable video' and 'Enable chat', both of which are checked. Below these fields, there is a section titled 'Invite users to conference' with a list of user avatars (6, 5, 4) and a dropdown menu. At the bottom of the modal is a large green 'CREATE' button.

Obr. 4.7: Formulár na vytvorenie novej miestnosti.

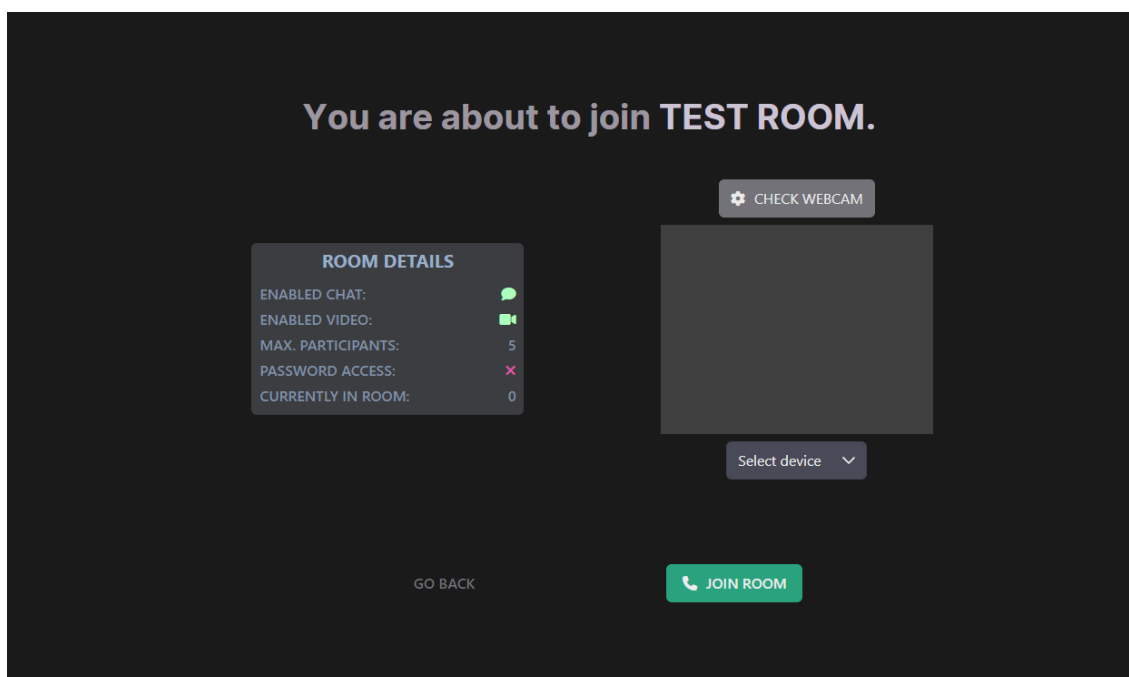
Po úspešnom vytvorení miestnosti sa miestnosť zobrazí v zozname (obrázok číslo 4.8). Vlastnosti miestnosti ako je prístup na heslo, obmedzenie videohovoru, či obmedzenie chatovej funkcionality je znázornené vedľa názvu miestnosti. Miestnosť, do ktorej bol používateľ pozvaný, je označená ikonou hviezdíčky.

The image shows a table titled 'LIST OF CONFERENCE ROOMS'. The table has three columns: 'ID', 'ROOM NAME', and a column with icons for room settings. The first row shows '1' in the ID column, 'TEST ROOM' in the ROOM NAME column, and a star icon, a speech bubble icon, a video camera icon, and '0/5' in the settings column. Below the table is a blue 'CREATE ROOM' button.

ID	ROOM NAME	
1	TEST ROOM	★ 🗨️ 📹 0/5

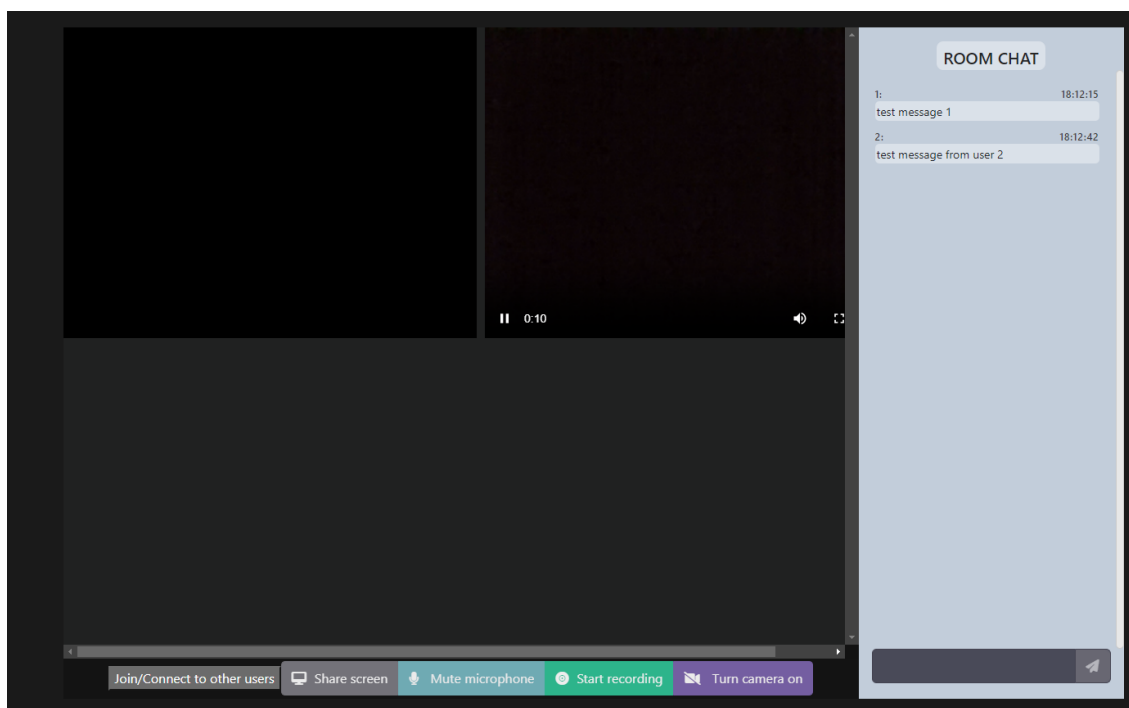
Obr. 4.8: Zoznam miestností obsahujúci novovytvorenú miestnosť.

Stlačením tlačidla **JOIN** sa používateľ dostane do *lobby*. Ide o čakaciu miestnosť, v ktorej si môže overiť správne fungovanie svojej webkamery. Obrazovka je znázornená na obrázku číslo 4.9. V ľavej časti stránky je prehľad detailov miestnosti. Zachytáva aj počet pripojených používateľov a skutočnosť, či je prístup do miestnosti obmedzený na heslo.



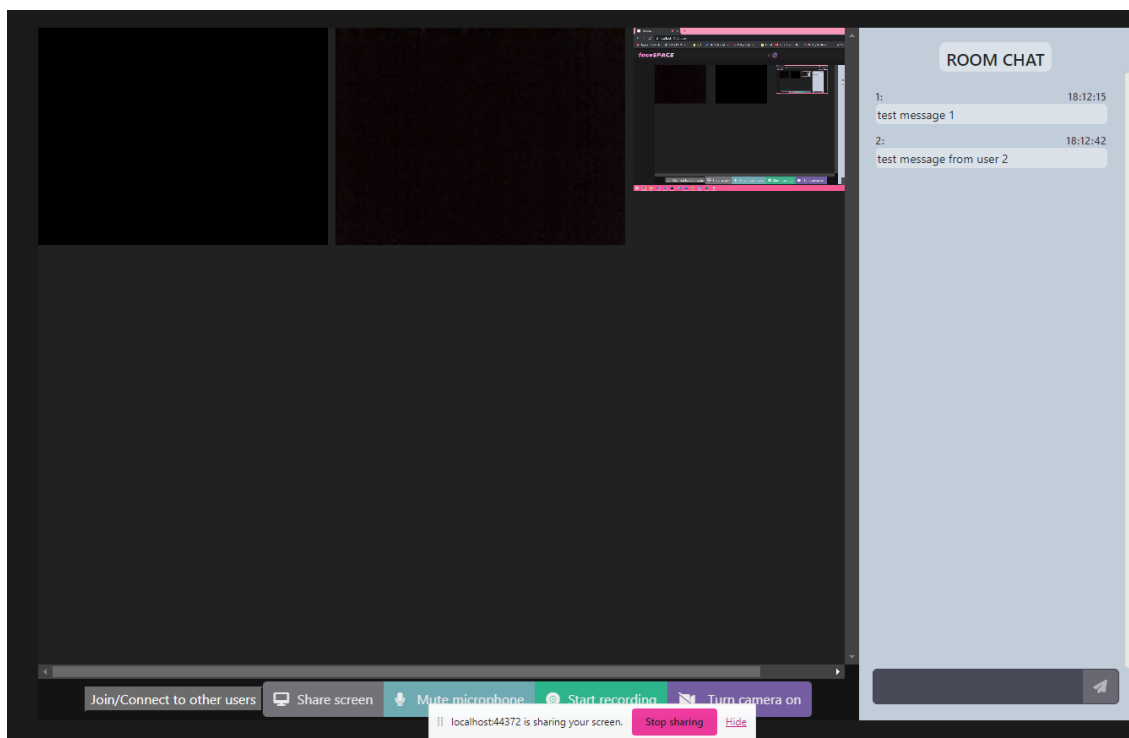
Obr. 4.9: Zoznam miestností obsahujúci novovytvorenú miestnosť.

Po stlačení tlačidla **JOIN ROOM** je používateľ presmerovaný do konkrétnej miestnosti, v ktorej prebieha hovor. Obrázok číslo 4.10 už zachytáva dvojicu pripojených používateľov spolu s chatom. V spodnej časti obrazovky je séria tlačidiel, ktorými sa vie používateľ do hovoru napojiť (toto je potrebné pre korektné vytvorenie WebRTC spojenia medzi používateľmi), zazdieľať svoju obrazovku, vypnúť a zapnúť svoj mikrofón či webkameru, alebo nahráť záznam svojej kamery.



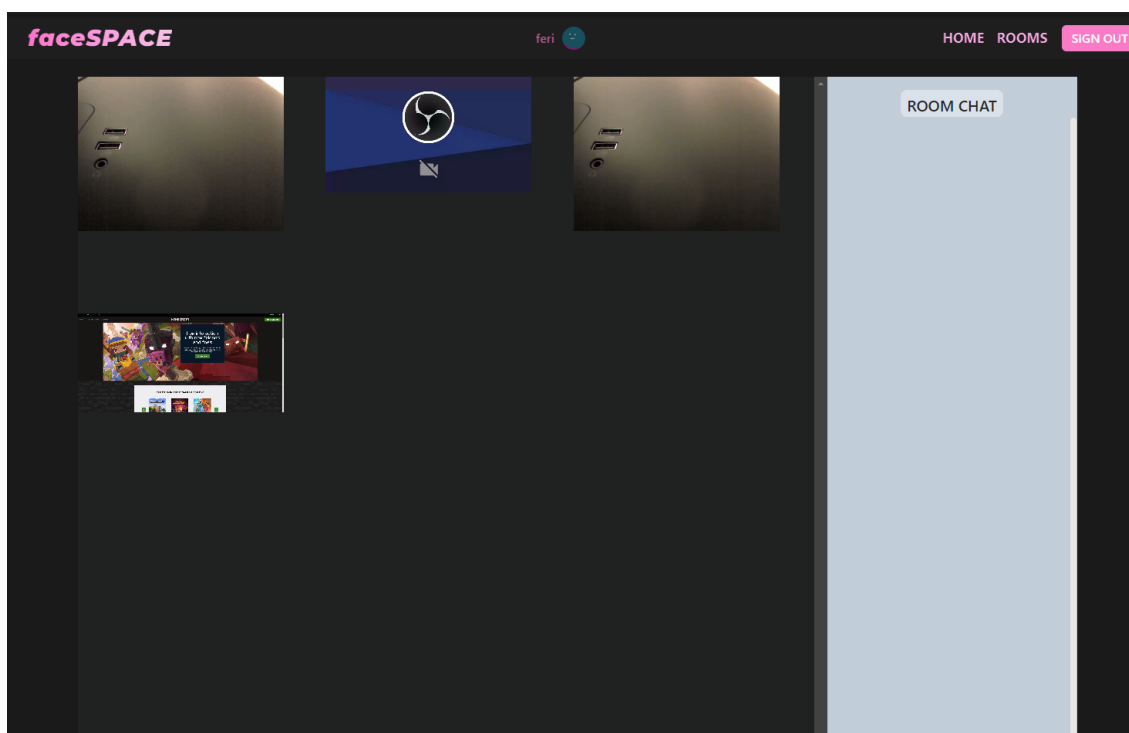
Obr. 4.10: Konferenčná miestnosť.

Príklad zdieľania obrazovky jedného pripojeného účastníka je na obrázku číslo 4.11.



Obr. 4.11: Príklad zdieľania obrazovky účastníka.

Pri pripojení účastníka alebo zapnutí zdieľania obrazovky sa počet videí zobrazovaných v hlavnom komponente podstránky zvyšuje. Pre prehľadnosť sa organizujú do *gridu* a postupne sa zmenšuje ich veľkosť. Príklad 4 aktívnych videí v komponente je na obrázku číslo 4.12.



Obr. 4.12: Príklad 4 aktívnych videí v gride miestnosti.

5 Vyhodnotenie

Cieľom nášho projektu bolo vytvoriť modernú aplikáciu pre videokonferencie, ktorá by spĺňala súčasné požiadavky na komunikáciu v online prostredí. Výsledkom je funkčná aplikácia, ktorá zahŕňa **všetky** funkcionality, ktoré boli v úvode riešenia projektu vymedzené mnohými funkčnými a nefunkčnými požiadavkami.

Pri jej návrhu a implementácii sme dbali na princípy čistého kódu a využitie návrhových vzorov. Pri implementácii sme použili vhodné moderné technológie.

Softvérové riešenie dovoľuje používateľom vzájomne komunikovať cez hlasový hovor, videohovor a chat. Aplikáciu sme patrične otestovali.

Aktuálna verzia riešenia obsahuje aj isté nedostatky. Ide napríklad o *bezpečnostný faktor*, ktorému sme v tejto prvotnej implementácii takmer žiadnu pozornosť nevenovali. Nedostatky, ktoré primárne spočívali v obmedzených podmienkach na testovanie, zahŕňajú aj korektné ukončenie WebRTC spojenia klienta a pripojenie klienta do viacerých miestností naraz. Priestor na zlepšenie vnímame aj vo forme vytvárania záznamu z konferenčného stretnutia, ktoré by zahŕňalo ukladanie audiovizuálnych streamov všetkých aktívne pripojených používateľov spolu s chatovou komunikáciou.

Napriek spomínaným nedostatkom je aplikácia funkčná a spĺňa všetky požiadavky.