Technická univerzita v Košiciach
Fakulta elektrotechniky a informatiky

Kolaborácia vo virtuálnej realite: komunikačná a riadiaca časť

Diplomová práca

2020 **Bc. Michal Ivan**

Technická univerzita v Košiciach
Fakulta elektrotechniky a informatiky

Kolaborácia vo virtuálnej realite: komunikačná a riadiaca časť

Diplomová práca

Študijný program:	Informatika
Študijný odbor:	9.2.1. Informatika
Školiace pracovisko:	Katedra počítačov a informatiky (KPI)
Školiteľ:	Ing. Štefan Korečko, PhD.
Konzultant:	Ing. Marián Hudák

Košice 2020 **Bc. Michal Ivan**

Názov práce:	Kolaborácia vo virtuálnej realite: komunikačná a riadiaca časť
Pracovisko:	Katedra počítačov a informatiky, Technická univerzita v Košiciach
Autor:	Bc. Michal Ivan
Školiteľ:	Ing. Štefan Korečko, PhD.
Konzultant:	Ing. Marián Hudák
Dátum:	4. 5. 2020
Kľúčové slová:	virtuálna realita, a-frame, networked-aframe, správa klientov

Abstrakt: Hlavným cieľom tejto práce je navrhnúť riešenie pre získavanie informácii o pripojených klientoch v kolaboratívnom virtuálnom prostredí. Pre tieto potreby sme vytvorili klienta s rozšírenými právomocami – administrátora. V analytickej časti sú analyzované existujúce riešenia kolaboratívnych virtuálnych prostredí, ich implementácia a využívanie administrátora v systéme. Tiež je analyzovaná architektúra použitá v našom kolaboratívnom prostredí LIRKIS. V syntaktickej časti sú navrhnuté a implementované komponenty pre administrátora, pre bežných klientov a pre server. Komponenty poskytujú možnosť získavania aktuálnych informácií o pripojených klientoch a posielanie správ v samostatných dátových kanáloch. Následne boli komponenty implementované do prostredia testovacej scény. Testovanie riešenia prebehlo používateľským testovaním použiteľnosti a záťažovým testovaním pri počte pripojených klientov väčšom ako 10. Na základe testovania môžeme konštatovať, že vytvorené riešenie je intuitívne, jednoduché na používanie a dokáže stabilne poskytovať všetky opísané funkcionality aj pri väčšom počte prihlásených klientov.

Thesis title:	Collaboration in virtual reality: communication and control part
Department:	Department of Computers and Informatics, Technical University of Košice
Author:	Bc. Michal Ivan
Supervisor:	Ing. Štefan Korečko, PhD.
Tutor:	Ing. Marián Hudák
Date:	4. 5. 2020
Keywords:	virtual reality, a-frame, networked-aframe, user management

Abstract: The main goal of this diploma thesis is to design a solution for obtaining information about connected clients in a collaborative virtual environment. For these needs we have created a client with extended authority – administrator. In the analytical part there are analyzed existing solutions of collaborative virtual environments, their implementation and administrator's use in the system. Also the architecture used in our collaborative LIRKIS environment is analyzed. In the syntactic part there are designed and implemented components for administrator, for common clients and for server. Components provide the ability to get up-to-date information about connected clients and send messages in separate data channels. Subsequently, the components were implemented in the test scene environment. The solution was tested by user usability testing and stress testing with more than 10 connected clients. Based on testing, we can conclude that the created solution is intuitive, easy to use and can stably provide all the described functionalities even with a larger number of logged in clients.

57692

TECHNICKÁ UNIVERZITA V KOŠICIACH

FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Katedra počítačov a informatiky

ZADANIE DIPLOMOVEJ PRÁCE

Študijný odbor: **Informatika**
Študijný program: **Informatika**

Názov práce:

Kolaborácia vo virtuálnej realite: komunikačná a riadiaca časť

Collaboration in Virtual Reality: Communication and Control Component

Študent: **Bc. Michal Ivan**
Školiteľ: **Ing. Štefan Korečko, PhD.**
Školiace pracovisko: **Katedra počítačov a informatiky**
Konzultant práce: **Ing. Marián Hudák**
Pracovisko konzultanta: **Katedra počítačov a informatiky**

Pokyny na vypracovanie diplomovej práce:

1. Oboznámiť sa s existujúcimi riešeniami webovej virtuálnej reality a analyzovať možnosti a existujúce prístupy ich využitia pre riadenie a komunikáciu v kolaboratívnej virtuálnej realite (KVR).
2. Na základe analýzy navrhnúť a implementovať komunikačnú a riadiacu časť systému KVR a to s použitím webových technológií.
3. Pri návrhu a implementácii spolupracovať s riešiteľmi/kami záverečných prác, pracujúcich na ďalších častiach systému KVR.
4. Experimentálne overiť použiteľnosť implementovaného riešenia a porovnať ho s podobnými existujúcimi riešeniami.
5. Vypracovať dokumentáciu podľa pokynov vedúceho práce.

Jazyk, v ktorom sa práca vypracuje: slovenský
Termín pre odovzdanie práce: 04.05.2020
Dátum zadania diplomovej práce: 31.10.2019

TECHNICKÁ UNIVERZITA V KOŠICIACH FAKULTA ELEKTROTECHNIKY A INFORMATIKY

12.

.....
prof. Ing. Liberios Vokorokos, PhD.
dekan fakulty

Čestné vyhlásenie

Vyhlasujem, že som záverečnú prácu vypracoval samostatne s použitím uvedenej odbornej literatúry.

Košice, 4.5.2020 .....

Vlastnoručný podpis

Poďakovanie

Touto cestou vyslovujem poďakovanie svojmu vedúcemu práce a konzultantovi za ich čas, pomoc, odborné vedenie, cenné rady a pripomienky pri vypracovaní mojej diplomovej práce.

S láskou v srdci by som rád vyjadril poďakovanie aj mojej rodine, predovšetkým mamke a priateľke za ich podporu a povzbudenie počas celého môjho štúdia.

Obsah

Zoznam obrázkov

Zoznam tabuliek

Motivácia

V každom virtuálnom priestore je potrebné mať kontrolu nad klientmi, ktorí sa v ňom nachádzajú. Či už ide o internetové fóra, herné servery alebo sociálne siete, vždy je vytvorený klient s vyššími právomocami ako bežní klienti a dokáže monitorovať aktivitu bežných klientov. Takýto klient sa nazýva administrátor.

Cieľom tejto práce je vytvoriť práve takéto administrátorské rozhranie pre použitie v multiplatformovom virtuálnom priestore, v ktorom sú použité technológie *A-frame* a *networked-aframe*. Administrátor tak bude má kontrolu nad ostatnými klientmi, zhromažďuje záznamy o ich aktivite a je schopný sledovať ich pozíciu v reálnom čase.

1 Úvod

Virtuálne prostredie s viacerými používateľmi (MUVE) je virtuálne prostredie, zdieľané medzi účastníkmi v počítačovej sieti. Podľa definície od *Sørakera* [1] môžeme virtuálne prostredie vnímať ako projekciu na obrazovke počítača alebo prostredníctvom obrazoviek umiestnených na hlave, ktoré blokujú všetky ostatné vizuálne podnety a poskytujú pohľad prvej osoby. *Steuer* [2] hovorí, že vo virtuálnych prostrediach je informácia lepšie opísaná ako sprístupnená pre interakciu. Preto sa nové médiá zameriavajú na vzťah jednotlivca s prostredím, v ktorom sa nachádza a na interakciu jednotlivca s týmto prostredím. Používatelia sa vo virtuálnom prostredí považujú za samostatné bytosti. Inými slovami, používatelia dostanú grafické reprezentácie samých seba označované ako avatar [3]. Avatarom sprostredkujú ostatným svoju identitu, prítomnosť a umiestnenie v priestore. Taktiež každá interakcia používateľa s virtuálnym prostredím alebo s inými používateľmi sa uskutočňuje prostredníctvom avatara.

Kolaboratívne virtuálne prostredie (CVE) je počítačový, distribuovaný, virtuálny priestor, v ktorom sa ľudia môžu stretávať a interagovať s ostatnými používateľmi, s virtuálnymi agentmi alebo s virtuálnymi objektmi [4]. CVE sa môžu líšiť v oblasti zobrazenia prostredia. Prostredie môže byť 3D grafický priestor, 2D prostredie, ale aj textové rozhranie. Prístup k CVE nie je obmedzený na stolné zariadenia, ale zahŕňa aj mobilné alebo iné nositeľné zariadenia ako VR sety.

Záverom je, že v zdieľanom kolaboratívnom virtuálnom prostredí sa všetka interakcia odohráva v reálnom čase, svet je zdieľaný, podporuje viacerých pripojených používateľov, ktorí môžu byť pripojení na rôznych platformách.

2 Formulácia úlohy

Cieľom tejto diplomovej práce je implementovať knižnicu administrátora virtuálnej scény. Virtuálna scéna využíva knižnicu *A-frame* a nadstavbu *networked-aframe*.

Zhrnieme si úlohy pre vypracovanie záverečnej práce v bodoch:

1. Analyzovať existujúce riešenia webovej virtuálnej reality, ich využitie pre riadenie a komunikáciu v kolaboratívnej virtuálnej realite (KVR).
2. Analyzovať architektúru systému KVR.
3. Na základe analýzy navrhnúť a implementovať komunikačnú a riadiacu časť systému KVR, a to s použitím webových technológií: *A-frame* a *networked-aframe*.
4. Pri návrhu a implementácii spolupracovať s riešiteľmi, riešiteľkami záverečných prác, pracujúcich na ďalších častiach systému KVR.
5. Integrovať všetky riešenia do jedného virtuálneho prostredia.
6. Experimentálne overiť použiteľnosť implementovaného riešenia. Overiť funkčnosť riešenia s čo najväčším počtom pripojených klientov v KVR.
7. Vypracovať dokumentáciu podľa pokynov vedúceho práce.

Táto knižnica má obsahovať metódy pre získavanie zoznamu prihlásených klientov a ich pozície na scéne v reálnom čase. Všetky tieto funkcie budú dostupné pre špecifického klienta – administrátora.

3 Analýza existujúcich riešení

V tejto časti analýzy si priblížime existujúce riešenia CVE, kde sa zameriame na manažment používateľov – klientov, odlíšenie administrátora od ostatných klientov, právomoci administrátora a na architektúru týchto riešení. Architektonicky nás zaujíma architektúra klient-server a použitie softvérového rámca *A-Frame* [5] na vytvorenie scény a rámca *networked-aframe* [6] na prepojenie klientov v jednej miestnosti.

3.1 Teachyverse

Systém Teachyverse umožňuje E-learning ponorením študentov do virtuálnej prednáškovej miestnosti [7]. Virtuálne vzdelávacie prostredie (VLE) je prostredie, ktoré je špeciálne navrhnuté na učenie. VLE sa používa na školenia zamerané na špecifické úlohy, pri ktorých nie je možné trénovať na základe reality z rôznych dôvodov, ako sú napríklad nárazy, fyzické nebezpečenstvo alebo fyzické obmedzenia [8]. Umožňuje študentom a lektorom stretnúť sa na prednáške vo VLE. Prednáška sa môže konať na diaľku, takže nie je potrebné byť v rovnakom fyzickom priestore. Študenti a lektori sú zastúpení 3D avatarmi, (pozri obrázok 3.1). Prenos zvuku v reálnom čase umožňuje prirodzenú komunikáciu medzi študentmi a medzi lektormi a študentmi.

3.1.1 Implementácia

Teachyverse je implementovaný v Unity s podporou HTC Vive a Google Cardboard. Súčasný prototyp Teachyverse je založený na použití Google CardBoard a smartfónov. Celá komunikácia je riešená prostredníctvom architektúry klient-server medzi študentmi/divákmi a lektorom. Snímky z prednášky môžu byť načítané zo súboru PDF alebo Zip a sú distribuované

prostredníctvom centralizovaného súborového servera všetkým pripojeným klientom.

Obr. 3.1: Virtuálne prostredie prednáškovej miestnosti s lektorom a žiakmi.

Zdroj: [7]

3.1.2 Manažment klientov

V Teachyverse sú v súčastnosti dve klientské roly: študent a lektor. Lektor dokáže povoľovať alebo zakazovať niektoré funkcie študentom. Tiež má iný pohľad na virtuálne prostredie. Na podporu lektora, ako je to v tradičných učebných prostrediach, je pred lektorom na úrovni pásu viditeľný pohľad na prezentáciu.

Iba lektor má úplnú kontrolu nad prezentáciou pomocou virtuálneho prezentátora, (na obrázku 3.1 pred lektorom), ktorý umožňuje prepínanie snímok, laserové ukazovanie a poskytuje kontrolu nad nahrávkami z prednášok. Študent má viditeľnú iba prezentačnú plochu, kde je prednáška viditeľná.

Lektor a študenti môžu počas živej prednášky klásť otázky ako počas diskusie. Využíva sa na to interný rečový komunikačný systém. Kvôli obmedzeniu nežiaduceho hluku počas prednášky môže lektor stlmiť všetku komunikáciu študentov. Ak to lektor povolí, každý študent sa môže pohybovať teleportovaním sám, na základe pohybových techník skúmaných v práci [9]. V opačnom prípade je náhodná pozícia v prednáškovej sále pridelená študentovi hneď po vstupe do virtuálneho priestoru.

3.2 Circles

Do platformy VLE patrí aj Circles [10]. Platforma Circles má dve hlavné teórie učenia, na ktoré sa zameriava a to konštruktivizmus – študenti si aktívne budujú svoje vedomosti prostredníctvom experimentálnejšieho modelu prostredia [11] a teória sociálneho poznania – pozorovaním druhých získavajú ľudia vedomosti, pravidlá, zručnosti, stratégie, presvedčenia a postoje [12]. Platforma Circles využíva systém miestností, kde skupina klientov môže spoločne skúmať svety vytvorené vo virtuálnom prostredí. Názov Circles je odvodený od možnosti administrátora – inštruktora spojiť viaceré svety pomocou portálov (pozri obrázok 3.2) do jednej veľkej slučky – kruhu.

3.2.1 Technológia a podporované platformy

Platforma Circles bola vytvorená pomocou WebVR, čo je VR založená na webovom prehliadači využívajúca HTML a JavaScript. Konkrétne bola použitá knižnica *A-Frame* [5] pre vývoj na strane klienta. Tiež sa používajú technológie klient-server *networked-aframe* [6] a Mozilla HUBs [13], využívajúce WebSockets a WebRTC pre viacpoužívateľské funkcie. Server Node.js prevádzkujúci tento systém je nasadený v jednej inštancii Amazonu EC2.

Obr. 3.2: Virtuálne prostredie Circles s portálmi do ďalších svetov.

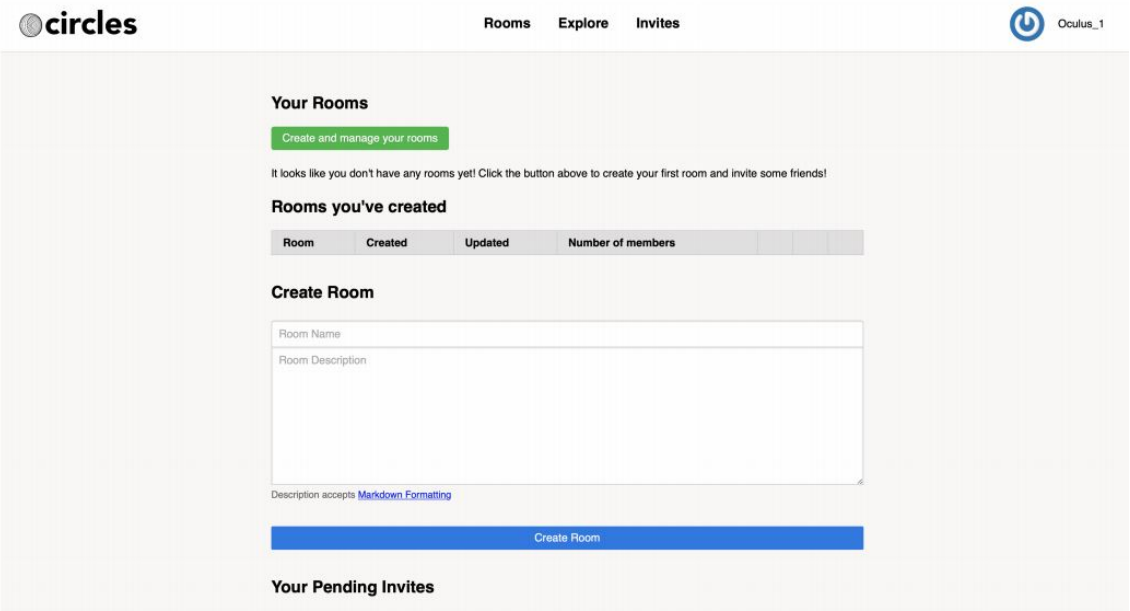
Zdroj: [10]

3.2.2 Klienti v systéme Circles

V platforme Circles sú tri druhy cieľových klientov:

- **Študent:** konečný klient, ktorý skúma a komunikuje s ostatnými klientmi v rámci svetov tak, ako je stanovené inštruktorom.
- **Inštruktor:** riadi vytváranie a prepojenie miestností do kruhov, zároveň sprevádza študentov v jednotlivých svetoch.
- **Vývojár:** navrhuje a vyvíja nové svety s použitím komponentov systému Circles, Javascriptu a systémov na správu sietí a používateľov.

Rolu administrátora v systéme Circles má priradenú inštruktor. V súčastnosti je jeho funkcionalita na chod systému vo vývoji. Inštruktor používa grafické používateľské rozhranie (zobrazené na obrázku 3.3), kde má kontrolu nad miestnosťami a zároveň má k dispozícii zoznam všetkých vytvorených miestností. Tak isto vie vkladať do každej miestnosti portály pre spojenie viacerých miestností do kruhu.



Obr. 3.3: Virtuálne prostredie Circles, rozhranie administrátora.

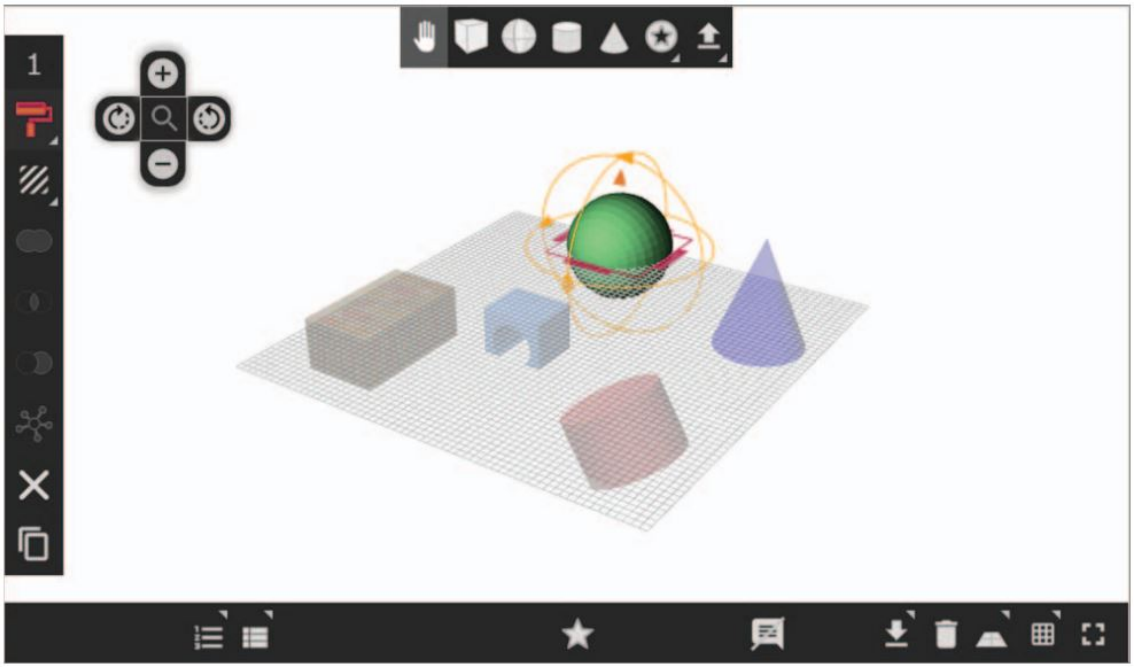
Zdroj: [10]

3.3 Tele-Board Prototyper

Tele-Board Prototyper je 3D modelovací systém pre prototypovanie, založený na webovom prehliadači. Využíva multiplatformové vykresľovacie rozhranie WebGL pre hardvérovo zrýchlenú vizualizáciu 3D modelov. Synchronizácia sa spolieha na výmenu správ založenú na WebSocket komunikácii cez centralizovaný server spolupráce Node.js v reálnom čase [14]. Cieľom Tele-board prototyper-a je umožniť geograficky rozptýleným tímom spolupracovať na 3D modeloch. Na obrázku 3.4 si môžeme pozrieť používateľské rozhranie systému.

Prototyper vychádza zo základu Tele-Board. Tele-Board poskytuje zdieľaný pracovný priestor vo forme povrchu virtuálnej tabule. Údaje obsahu sú automaticky synchronizované centrálnym serverom medzi všetkými pripojenými klientmi [15]. Umožňuje vzdialenú spoluprácu v reálnom čase na základe dvojrozmerného pracovného priestoru, v ktorom môžu klienti vytvárať a manipulovať so zdieľanými artefaktmi, napríklad s poznámkami, kresbami alebo obrázkami. Tele-board podporuje tvorivú tímovú prácu, napríklad pri dizajnovom myslení, aj keď sa členovia tímu nachádzajú na rôznych miestach.

Všetky akcie sa uskutočňujú na jednej interaktívnej tabuli. Akcie sa automaticky prenášajú na všetky ostatné pripojené tabule, ktoré zobrazujú rovnaký obsah. Poznámky je možné vytvoriť priamo na digitálnej tabuli. Tele-board podporuje rôzne platformy ako napríklad inteligentný telefón, digitálne pero alebo počítač. Všetci používatelia môžu vytvárať rýchle poznámky súčasne, aby umožnili paralelnú prácu.



Obr. 3.4: Používateľské rozhranie systému Tele-Board Prototyper.

Zdroj: [14]

3.3.1 Architektúra systému

Súčasti softvérového systému Tele-Board sú webový portál, klient bielej tabule, digitálne poznámky a komponent servera [15]. Nás najviac zaujíma komponent servera. Komponent servera spája všetky časti systému Tele-Board. Všetky udalosti sa prenášajú ako správy XMPP a tým sa synchronizujú klienti pripojení na rovnakú interaktívnu tabuľu. To znamená, že vždy uvidia rovnaký obsah a môžu pracovať na rovnakých položkách. Všetky akcie a poznámky sú stále synchronizované na server a ten ich ukladá do databázy, v ktorej je uložená história. Systém má funkciu aj zastavenia synchronizácie. Klienti potom

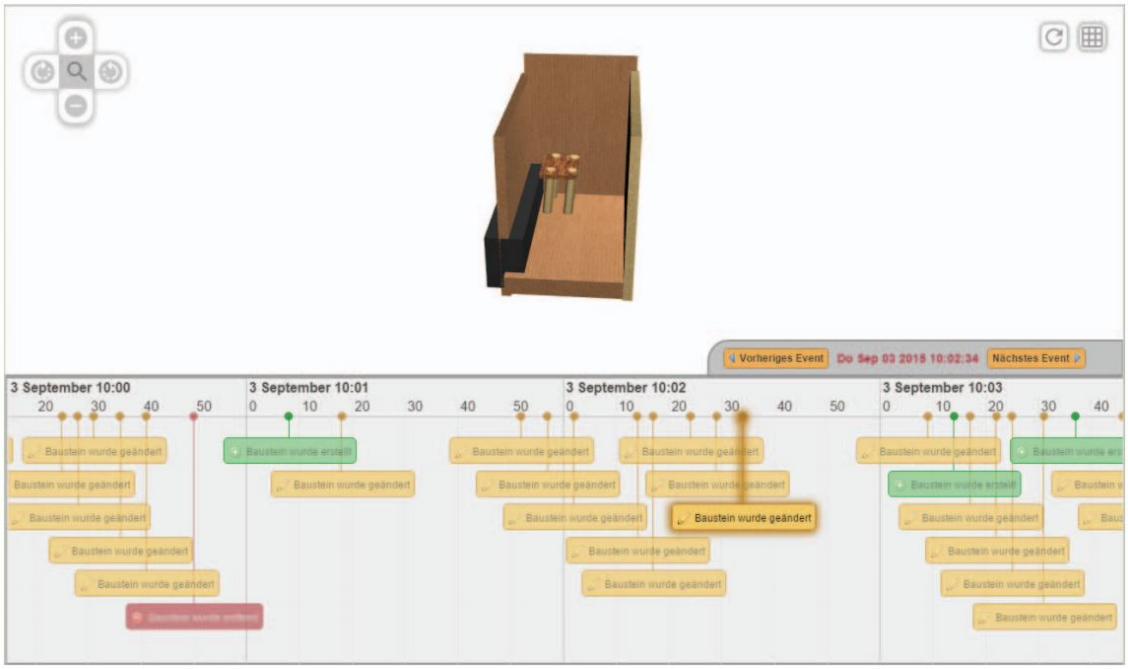
používajú virtuálne tabule bez toho, aby sa rušili.

Úpravy digitálneho 3D modelu na jednom mieste sa musia preniesť do všetkých pripojených systémov ukazujúcich na tento špecifický model. Tento synchronizačný mechanizmus je implementovaný pomocou rovnakého prístupu v reálnom čase ako v systéme Tele-board [14]. Akékoľvek zmeny 3D objektu alebo pracovnej roviny sú okamžite synchronizované medzi všetkými zúčastnenými klientmi. To znamená, že klienti môžu sledovať pohyby objektu a nevidia iba jeho konečnú polohu. Realizácia tejto funkcionality vyžaduje množstvo synchronizačných správ v smere prichádzajúcom aj odchádzajúcom. Použitie protokolu WebSocket [16], ktorý je založený na obojsmernom prenose cez príslušné API webového prehliadača sa ukázalo ako veľmi efektívne a vhodné. V architektúre riešenia sa nachádzajú dva servery. Synchronizačný server, ktorý sa zaoberá synchronizáciou údajov 3D modelov a konvenčný, ktorý poskytuje HTTP stránku a zdroje CSS a JavaScript.

Synchronizačný server ukladá dáta komunikačných správ do serverovej databázy spolu s časovou pečiatkou a príslušným identifikátorom prototypu. Tento prístup bol prevzatý z aplikácie Tele-board [17]. Táto koncepcia umožňuje zostaviť kompletný stav 3D modelu vrátane všetkých objektov, ktoré obsahuje v danom časovom okamihu. Tieto údaje sú tiež použité na „historické“ zobrazenie konkrétneho 3D prototypu. „Historické“ zobrazenie poskytuje zoznam zmien jednotlivých častí prototypu, počas celého času vývoja. Po spustení aplikácie Prototyper vo webovom prehliadači sa automaticky načíta posledný zaznamenaný stav príslušného 3D modelu. Sledovanie vývoja prototypu (pozri obrázok 3.5) je možné vďaka „historickým“ dátam.

3.3.2 Správa klientov

Rozdelenie klientských rolí je prebraté zo systému Tele-board [15]. Webový portál je vstupným bodom systému Tele-Board. Slúži ako administračné rozhranie, ktoré umožňuje klientom spravovať projekty a súvisiace panely. Klienti sú rozdelení na bežných klientov a administrátorov. Administrátor dokáže vytvárať nové projekty, priradiť ostatných klientov na vybraný projekt a vidí aj ich aktivitu na vybranom projekte. Bežný klient nevie vytvárať nový projekt. Ak je priradený na projekt, tak môže meniť všetky veci v danom projekte, no nemôže pozvať ostatných používateľov do projektu.



Obr. 3.5: Virtuálne používateľské rozhranie systému Tele-Board prototyper so zobrazenou históriou modelu.

Zdroj: [14]

3.4 TogetherVR

Vo väčšine súčasných systémov, ktoré ponúkajú podporu pre viacerých klientov vo virtuálnom spoločnom priestore sú klienti reprezentovaní pomocou avatara. Na vyriešenie tohto problému sa vyvíja modulárny webový rámec TogetherVR [18], ktorý rozširuje súčasné možnosti video konferencií s novými funkciami VR. Klient nevidí len grafickú reprezentáciu iného klienta, ale priamo obraz iného klienta a jeho okolie. Predstavme si, že máme dve konkrétne a podobné užívateľské miesta, každé s prenosným počítačom, technológiu Oculus Rift HMD, webovú kameru, pozadie zelenej obrazovky a ovládač. Obaja používatelia môžu zdieľať veľkú zelenú obrazovku alebo môžu mať prostredie rozdelené na dva rôzne fyzické priestory. Je dôležité si uvedomiť, že fyzické prostredie používateľa je zarovnané s virtuálnym prostredím, takže ak sa používateľ pozrie do webovej kamery, tak sa pozerá priamo na inú osobu (pozri obrázok 3.6).

Obr. 3.6: Obrazovka klienta pri používaní aplikácie TogetherVR.

Zdroj: [19]

3.4.1 Architektúra TogetherVR

V systéme TogetherVR sú použité technológie ako SimpleWebRTC, AngularJS, Node.js, WebVR, A-Frame a dash.js. Vstupný bod do aplikácie ponúka server TogetherVR. Tento server ponúka modulárnu webovú aplikáciu založenú na AngularJS [18]. Aplikácia je založená na štyroch moduloch. Nás najviac zaujíma modul zobrazenia obsahu vo VR a modul prepojenia klientov. Modul zobrazenia obsahu vo VR – miestnosť využíva rámec A-Frame, ktorý spája WebVR spolu s three.js v jednoduchom skriptovacom jazyku. To umožňuje prezerať obsah VR platforiem ako je napríklad Oculus Rift. Modul na nastavenie spojení WebRTC –

podpora priamej komunikácie medzi používateľmi prostredníctvom zvuku a videa. TogetherVR zatiaľ nerieši správu pripojených klientov. V čase písania tejto práce bola aplikácia TogetherVR len v demo verzii. TogetherVR sa zameriava na vzdialenú prítomnosť klientov vo virtuálnom prostredí [19].

V našom virtuálnom prostredí poskytujeme klientom možnosť pripojiť sa aj prostredníctvom Microsoft HoloLens. Ten vytvára rozšírenú realitu. Všetci klienti sú reprezentovaní virtuálnymi avatarmi. Preto myšlienka zameniť avatara za skutočného klienta je zaujímavá do budúcna.

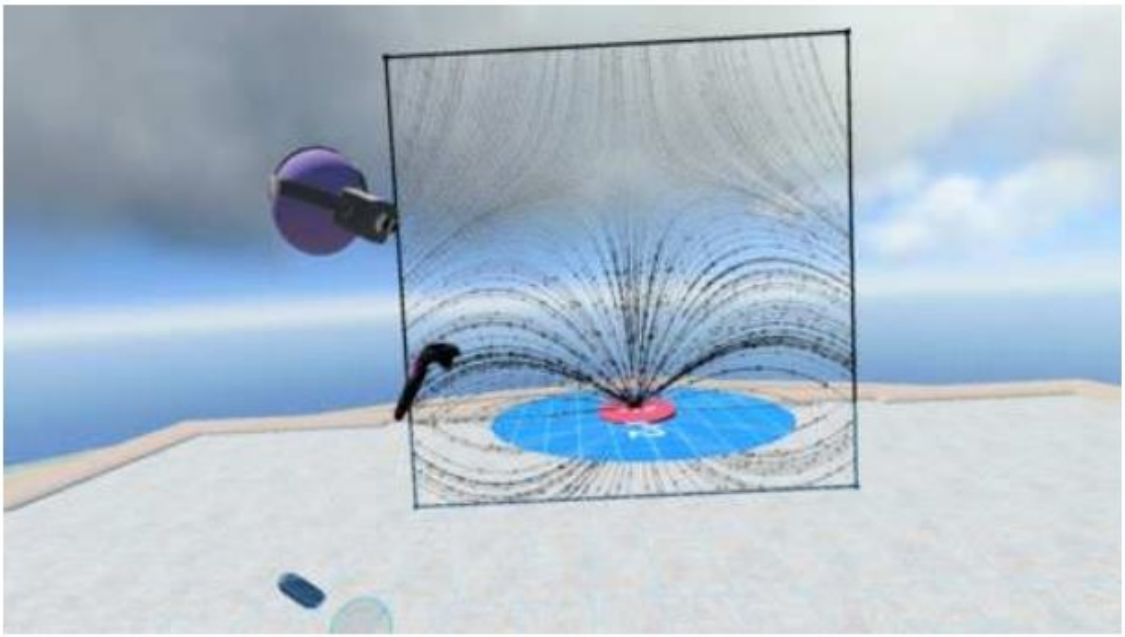
3.5 ElectroVR

ElectroVR umožňuje dvom účastníkom navzájom zaberať jeden fyzický a zároveň virtuálny priestor, v ktorom sa môžu učiť prostredníctvom priestorovej simulácie elektrostatiky [20]. ElectroVR demo nepodporuje veľké množstvo klientov. Je orientované skôr na hlboký priestorový prehľad, ako na algebraické a kvantitatívne analytické zručnosti. Účastníci vstupujú do systému spoločne a majú možnosť preskúmať lištu nástrojov, alebo si môžu prezrieť nahraté sekvencie. Klienti vo virtuálnom prostredí sú reprezentovaní pomocou avatarov a ovládajú rôzne nástroje, pričom vysvetľujú fyzikálne princípy, ktoré sú zobrazované v sekvenciách (pozri obrázok 3.7). Panel nástrojov obsahuje tri druhy položiek: distribúciu náboja, vizualizačné objekty a ručné nástroje. Distribúcia náboja vytvára elektrické pole, ktoré preniká do 3D priestoru. Vizualizačné objekty umožňujú účastníkom modifikovať pole vložením objektu do vytvoreného poľa. Ručné nástroje určujú funkcie ručných ovládačov ako napríklad rotácia objektu alebo výber objektu.

3.5.1 Architektúra systému ElectroVR

ElectroVR má architektúru klient-server. Každá inštancia klienta tiež spúšťa svoju vlastnú kópiu simulácie. Polohy objektov a klientov sú pravidelne synchronizované so serverom. Následne je simulácia vykreslená klientovi na základe získaných údajov zo servera. Počas pobytu vo virtuálnom prostredí systému ElectroVR je zaznamenávaný zvuk pomocou mikrofónu klientov, pohyb ich avatarov a históriu vývoja virtuálneho prostredia [20].

Systém ElectroVR podporuje iba platformu HTC Vive. Taktiež systém ElectroVR nerieši správu pripojených klientov. Je určený pre málo pripojených klientov, ktorí musia byť spolu v jednom fyzickom priestore, ale vidia sa iba prostredníctvom avatarov vo virtuálnom prostredí.



Obr. 3.7: Zobrazenie virtuálneho prostredia ElectroVR.

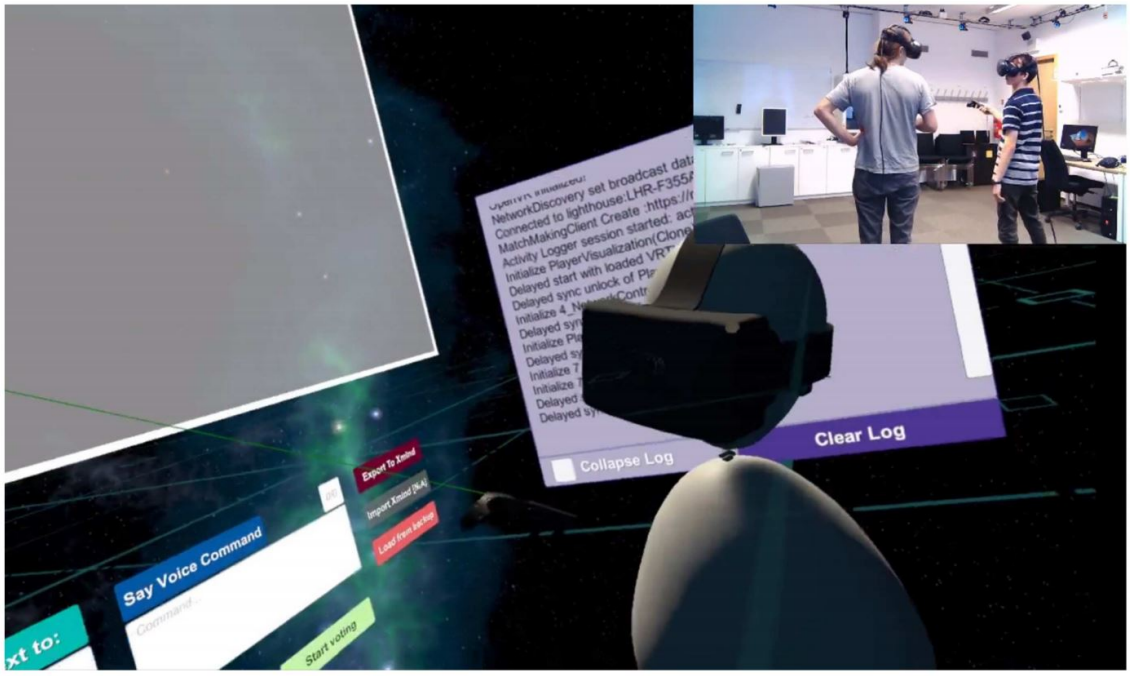
Zdroj: [20]

3.6 Interaktívna a multimodálna virtuálna myšlienková mapa

Interaktívna a multimodálna virtuálna myšlienková mapa (IMVMM) uľahčuje rôzne typy inteligentných nápadov, brainstormingov alebo akékoľvek iné, väčšinou tvorivé činnosti. Účastníci sa môžu nachádzať v rôznych fyzických prostrediach a môžu mať spoločný cieľ na určitú tému v obmedzenom čase [21]. Používatelia môžu zoskupovať poznámky patriace do konkrétnej kategórie a intuitívne s nimi interagovať pomocou kombinácie rôznych spôsobov ineterakcie. Po dokončení úlohy majú klienti prístup k obsahu, ktorý za daný čas vytvorili a hodnotia ho.

Hlavnou časťou systému IMVMM je plátno s mapou mysle. Plátno slúži ako virtuálna stena, ktorá používateľom poskytuje priestor na umiestňovanie nápadov a zdieľanie ich s ostatnými. Všetky uzly sú umiestnené na tejto 2D stene, aby sa proces udržal čo najbližšie k skutočnému svetu a zároveň poskytovali podobný vizuálny štýl ako bežné softvérové nástroje mapovania mysle.

Interakcia s HMD je možná vďaka doplnku Virtual Reality Toolkit. Na spustenie aplikácie sa vyžaduje SteamVR, pretože poskytuje aplikačné rozhrania pre zariadenia VR. Aplikácia bola navrhnutá tak, aby bola funkčná s HMD, ale je možné ju použiť aj so stolnými počítačmi alebo notebookmi. V tomto prípade sa ako vstupné zariadenie vyžaduje klávesnica a myš. Ak je prítomný aj mikrofón, služba rozpoznávania reči sa môže stále používať ako spôsob interakcie. Pokiaľ ide o HMD, systém je implementovaný tak, aby pracoval s HTC Vive (Pro) a jedným ovládačom. Používateľ musí pracovať iba s dvoma tlačidlami ovládača (touchpad a trigger), aby mal úplnú kontrolu nad funkciami systému. Keď používateľ stlačí dotykovú plochu, laserové ukazovátko sa vysiela z ovládača. Spúšťač slúži ako tlačidlo „akcie“; keď používateľ ukazuje na nejaký prvok, stlačením spúšťača sa spustí príslušná akcia.



Obr. 3.8: Virtuálne prostredie systému IMVMM.

Zdroj: [21]

3.6.1 Architektúra systému

Softvér IMVMM bol vyvinutý v Unity v jazyku C#. Jadrom komunikačnej časti systému IMVMM je platforma CVE opísaná v článku [22], využívajúca technológiu Unity Networking. Systém pracuje na modeli hostiteľ – klient, v ktorom je jedným používateľom server a klient súčasne, zatiaľ čo ostatní používatelia sú len klienti. Každý používateľ je reprezentovaný ako avatar (ako je znázornené na obrázku 3.8) s HMD a VR ovládačom v rukách. Pozície avatara a ovládača sú synchronizované v sieti. Online spolupráca zahŕňa aj systém uzlovania, ktorý bráni používateľom modifikovať uzol, zatiaľ čo iný používateľ s ním momentálne pracuje. Zdieľať je potrebné aj laserové ukazovátko pripojené k ovládačom, ktoré používateľom umožňuje získať okamžitú spätnú väzbu o mieste, na ktoré ukazujú a zároveň vidia kam ukazuje iný používateľ. Keď klient ukazuje na uzol, systém lokálne skontroluje, či je uzol pre tohto klienta uzamknutý alebo nie. Ak je uzol už uzamknutý, nedá sa vybrať. Ak uzol ešte nemá manažéra, server urobí žiadajúceho používateľa manažérom tohto uzla a odošle informáciu ostatným klientom, že uzol je uzamknutý. Ak zrušíte výber uzla, odošle sa správa o odomknutí na server, ktorý potom tieto informácie rozšíri na klientov.

3.6.2 Manažment klientov

Keďže je softvér IMVMM určený pre viacerých používateľov, systém obsahuje viacero rolí. Administrátorom celého brainstormingového cvičenia je moderátor. Všetci ostatní pripojení klienti sú bežnými používateľmi systému [22]. Zoberme si napríklad hlasovanie o výbere najvhodnejšej z vytvorených myšlienok. Moderátor vedie hlasovanie brainstormingu, kde priradí myšlienkam vybraným do hlasovania farby. Potom pridelí rovnaký počet bodov všetkým klientom, ktoré rozdelia medzi vybrané myšlienky. Pri každom takomto kole hlasovania ostatní klienti uvidia iba počet pridelených bodov vybranému nápadu. Keď sa moderátor pokúsi ukončiť hlasovacie kolo, systém skontroluje, či všetci účastníci rozdelili všetky svoje body a až potom sa ukončí kolo.

3.7 Záver

V tejto časti si zhrnieme jednotlivé systémy. Súčasne si vyberieme vhodné vlastnosti, ktoré môžeme použiť v našom riešení. Zhrnutie jednotlivých systémov a ich vlastností si zosumarizujeme v bodoch:

- **Teachyverse** – je implementovaný v Unity, má podporu zariadení HTC Vive a inteligentných telefónov. Administrátorom v tomto systéme je lektor, ktorý má možnosť ovládať prezentáciu pomocou špecifického rozhrania, ktoré vidí len on. Keďže systém poskytuje prenos zvuku v reálnom čase, môže stlmiť všetku komunikáciu študentov.

- **Circles** – bol vytvorený pomocou A-Frame a networked-aframe, tak isto ako je implementované naše prostredie a využíva rovnakú technológiu pripojenia klientov: klient-server. Rolu administrátora má priradenú inštruktor, ktorý sprevádza študentov virtuálnymi svetmi a prepája jednotlivé svety medzi sebou.

- **Tele-Board Prototyper** – je založený na architektúre klient-server. Používa WebGL na vykresľovanie modelov a je zložený z dvoch serverov. Prvý server poskytuje webové rozhranie s plochou pre vykreslenie prototypu. Druhý server sa stará o vykreslenie modelu, synchronizáciu práce medzi ostatných klientov a zberom dát do histórie o vývoje prototypu. Administrátor má od ostatných klientov možnosť vytvárať projekty a tiež môže priradiť klientov na vybrané projekty. Podpora iba počítača bez možnosti plného ponorenia používateľa.

- **TogetherVR** – je určený pre dvoch pripojených klientov. Využíva technológiu A-Frame a server SimpleWebRTC. V systéme sa vo virtuálnom prostredí nahrádza avatar so samotným obrazom klienta, pomocou obrazu web kamery, ktorá sníma klienta. Neposkytuje žiadny manažment klientov. Systém podporuje iba Oculus Rift.

- **ElectroVR** – je určený pre malý počet klientov(2–5). Architektúra systému je klient-server. Každý klient si spúšťa vlastnú simuláciu a až po zdieľaní je vykreslená ostatným klientom. Systém podporuje iba HTC Vive.

- **IMVMM** – využíva SteamVR a je implementovaná v Unity. Systém je navrhnutý pre viacero používateľov. Využíva architektúru klient-server. Rolu administrátora má moderátor, ktorý riadi hlasovacie kolá, rozdeľuje body a spočítava hlasy pre vybranú myšlienku. Podporuje len platformu HTC Vive.

Ako vidíme zo zhrnutia, nie každý systém má podporu viacerých platforiem, ako napríklad ElectroVR, TogetherVR alebo IMVMM. Našim cieľom je vytvoriť multi-platformové riešenie, bez nutnosti použitia externých knižníc ako využíva IMVMM. Práve preto je využitie technológie A-Frame výhodné. A-Frame poskytuje možnosť pripojenia viacerých platforiem od bežných displejov počítačov cez mobilné telefóny, ktoré môžeme použiť aj ako HMD, až po rôzne VR sety.

Naše riešenie má byť použiteľné aj s väčším počtom pripojených klientov naraz, tak ako Circles, ktorý využíva tie isté technológie. V skoro všetkých opísaných systémoch určených pre viacej ako 3 používateľov je architektúra klient-server výhodná a tiež ju využívajú nami používané technológie A-Frame a networked-aframe. Technológiou networked-aframe zdieľame scénu a entity vytvorené v A-Frame. Networked-aframe tiež poskytuje rôzne operácie pre manažovanie klientov ako posielanie správ, obmedzenie práv, získavanie údajov o pohybe klientov a tiež možnosť odpojenia iného klienta. Možnosť vytvorenia klienta s vyššími právomocami je výhodná pre systémy, kde sú viacerí klienti ako aj v systémoch Teachyverse – stlmenie zvukov žiakov; Circles, Tele-Board Prototyper – pozývanie klientov do CVE.

Funkcia ukladania histórie aktivity klientov ako je v systéme Tele-Board Prototyper je vhodná aj do nášho riešenia, ale v trochu inej podobe. Stačí len získavať časy prihlásenia a odhlásenia klientov napríklad, aby sme zistili v akých časoch je CVE najviac vyťažené.

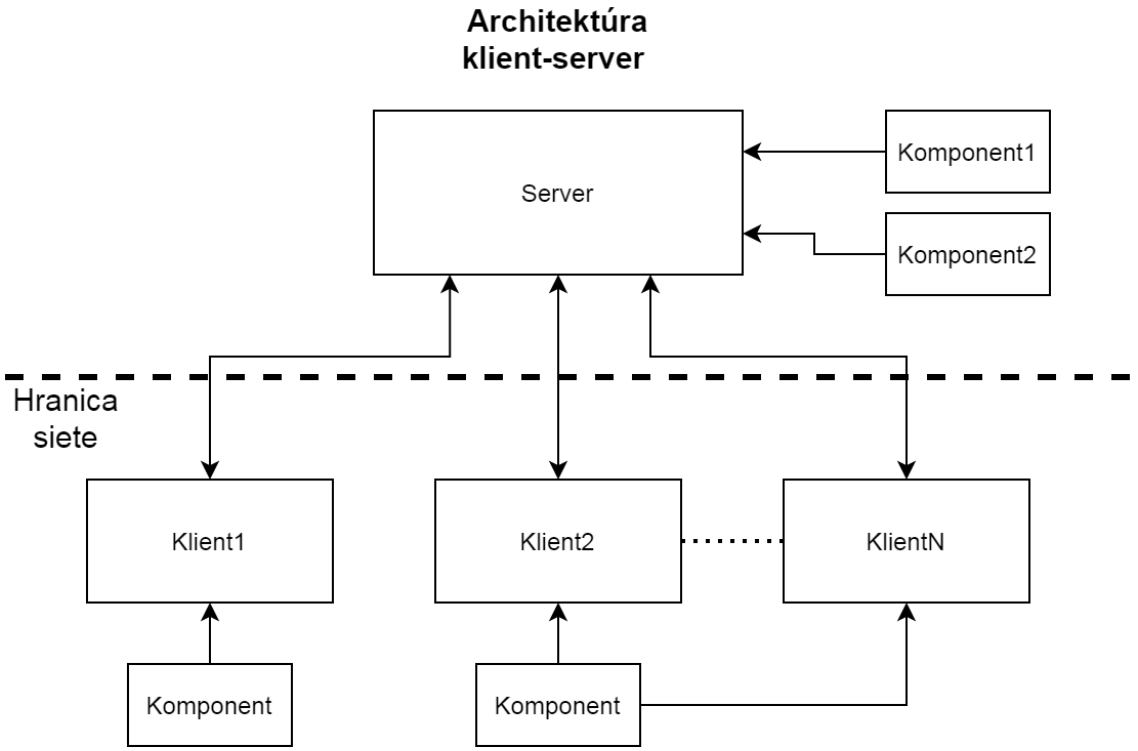
4 Analýza architektúry systému LIRKIS G-CVE

V tejto kapitole sa venujeme analýze architektúry systému LIRKIS G-CVE. Ako sme už skôr spomínali LIRKIS G-CVE využíva technológiu networked-aframe, ktorá má architektúru klient-server. Práve architektúru klient-server aj všetky použité nadstavby si opíšeme v tejto kapitole.

4.1 Architektúra klient-server

Na obrázku 4.1 môžeme vidieť všeobecnú architektúru klient-server. Server je základom architektúry. Musí mať dostatočný výkon a kapacitu na zvládanie prijímania, spracovávania žiadostí a posielania odpovedí, viacerým súčasne pripojeným klientom. Tiež obsahuje komponenty, ktoré poskytujú základnú alebo rozširujúcu funkcionalitu pre klientov. Klient v tejto architektúre odosiela požiadavky na server, čaká na odpovede a zobrazuje prijaté údaje používateľovi. Komponenty klientov sa môžu pre klientov s inými právomocami líšiť alebo jedna skupina klientov môže použiť rovnaký komponent pre každého. Preto je potrebné odlíšiť klientov s inými právomocami od ostatných.

Server môže byť lokálny alebo globálny. Pre lokálny server predstavuje hranica siete len lokálnu sieť – server je prístupný len zariadeniam na spoločnej LAN sieti. Globálny server je spustený na inštancii, ktorá je prístupná na voľne dostupnej doméne. Vtedy hranica siete predstavuje rozdiel medzi voľne dostupnou doménou servera a klientskou internetovou adresou.



Obr. 4.1: Architektúra klient-server.

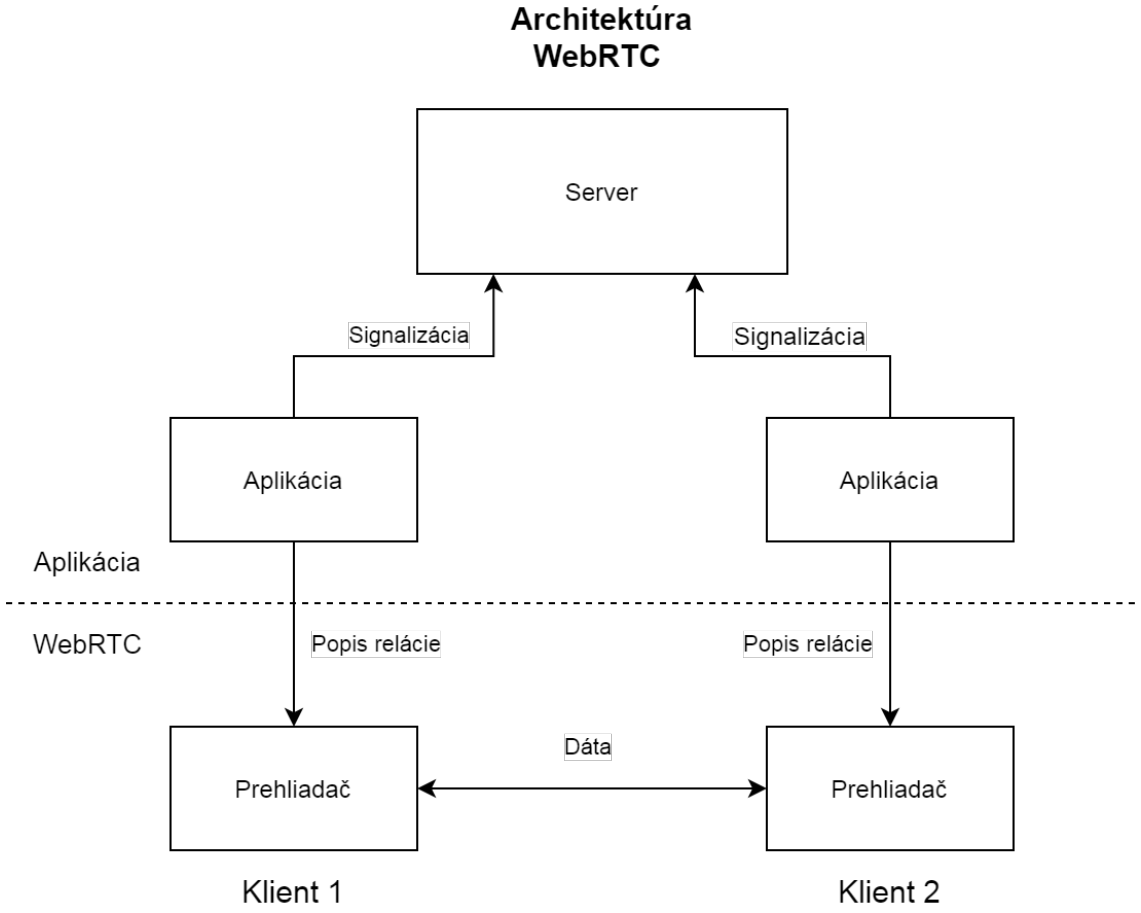
Zdroj: [23]

4.2 Architektúra WebRTC

Networked-aframe vo verzii 0.6.0, použitej aj v našom prostredí LIRKIS G-CVE, využíva WebRTC [24] s nadstavbou EasyRTC [25]. WebRTC je technológia, ktorá umožňuje dvom klientom komunikovať peer-to-peer, to znamená vymieňať si dáta, ktoré neprechádzajú serverom ako je to na obrázku 4.2. Zahŕňa schopnosť zachytávať, prenášať, prijímať a prehrávať zvukové, obrazové údaje, ako aj textové a surové bajtové polia. Práve systém Circles spomenutý v analýze využíva technológiu WebRTC.

Dôležité je si uvedomiť: peer-to-peer neznamená, že nie sú zapojené žiadne servery. Znamená to, že bežné dáta nimi neprechádzajú. Server je stále potrebný, aby si dvaja klienti vymieňali základné informácie, o tom kde sú v sieti a aké kodeky na kódovanie a dekódovanie dátového toku podporujú, aby si mohli vytvoriť toto spojenie typu peer-to-peer. Aplikácia odošle tieto signalizačné informácie serveru. Následne zapíše do prehliadača popis relácie, ktorú môže

nadviazať s iným klientom. Potom prebehne nadviazanie priameho spojenia a vytvorenie dátového kanálu. Následne si môžu klienti posielať údaje medzi sebou bez potreby servera, teda peer-to-peer. Hranica aplikácie a WebRTC nám oddeľuje, kompetencie aplikácie a WebRTC. Po odoslaní dát serveru a sprostredkovaním popisu relácie, WebRTC sám nadviaže spojenie s iným klientom a vytvorí dátový kanál.



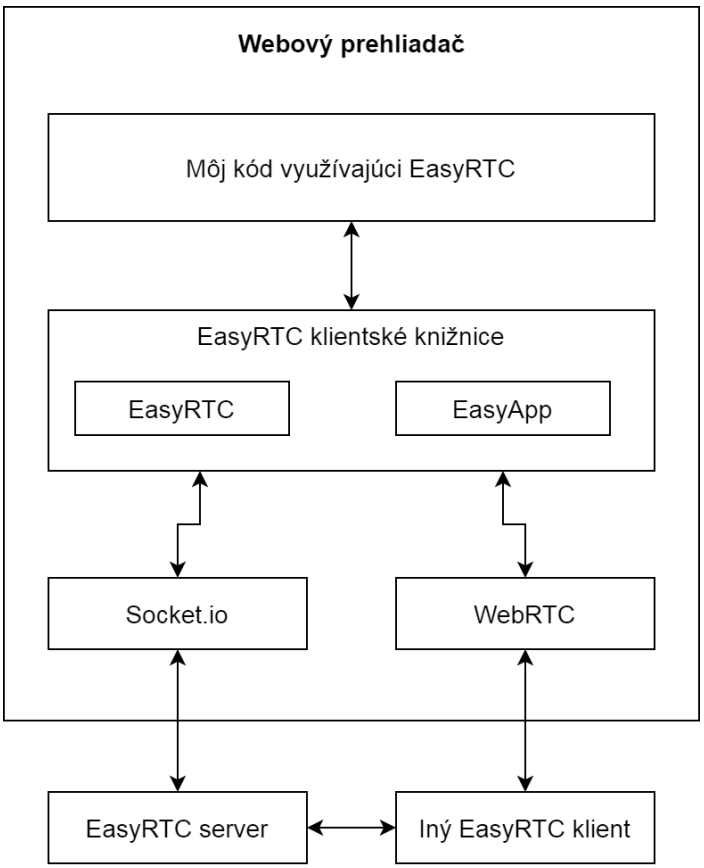
Obr. 4.2: Architektúra WebRTC.

Zdroj: https://www.html5rocks.com/en/tutorials/webrtc/basics/

4.3 Architektúra EasyRTC

EasyRTC je rozhranie WebRTC API. EasyRTC využíva klientske knižnice, ktorými riadi signalizáciu a vo veľkej miere izoluje aplikácie od prebiehajúcich zmien v rozhraní WebRTC. EasyRTC nám uľahčuje prácu s WebRTC funkciami.

Na obrázku 4.3 vidíme webový prehliadač, kde sa vykonávajú funkcie z EasyRTC knižnice klienta. Komunikácia prebieha cez socket s EasyRTC serverom a pomocou WebRTC s iným klientom používajúcim EasyRTC.



Obr. 4.3: Architektúra EasyRTC

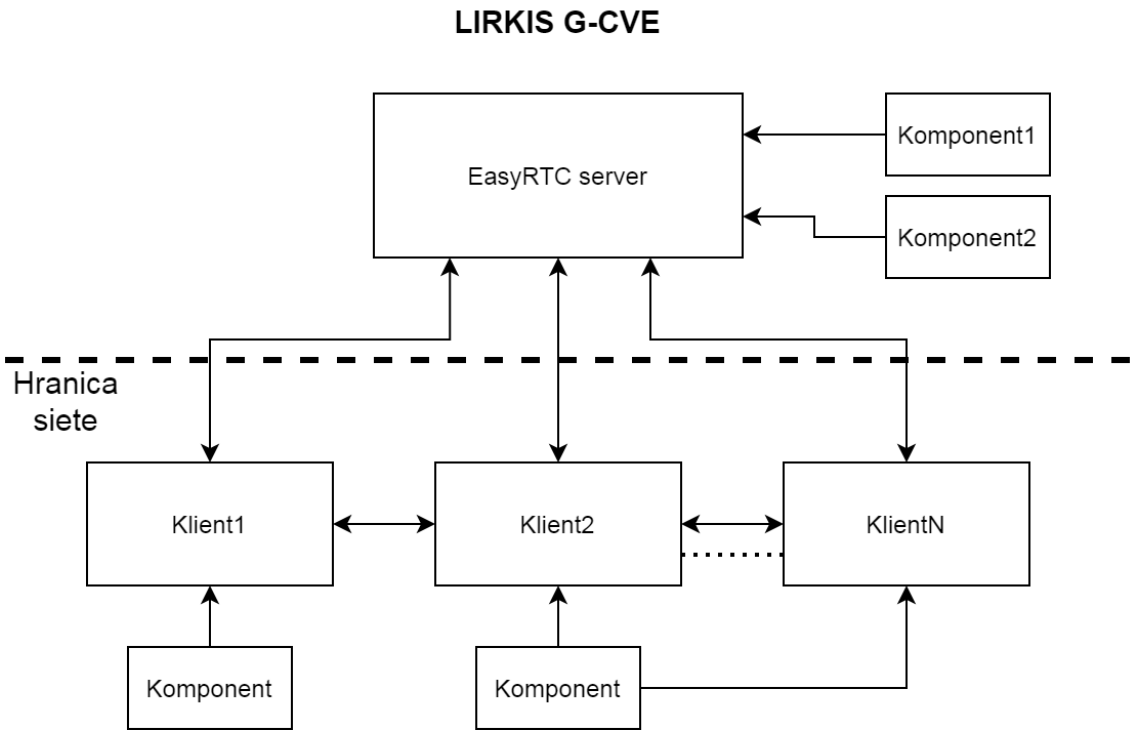
Zdroj: [25]

4.4 Architektúra systému LIRKIS G-CVE

Po opísaní všetkých použitých technológii je celková architektúra systému LIRKIS G-CVE zobrazená na obrázku 4.4.

Základom tejto architektúry je model klient-server. Server môže byť buď lokálny alebo globálny. S tým sa mení aj hranica siete popísaná pri obrázku 4.1. Využitím EasyRTC a zároveň WebRTC môžu klienti komunikovať priamo medzi sebou. Týmto sa kladú nižšie nároky na server.

Taktiež môžeme stavať na možnosti jednoduchého pridávania komponentov serveru alebo klientom, ktorými rozširujeme funkcionalitu celého systému. Môžeme pridávať komponenty buď jednotlivo každému klientovi vlastný, alebo rovnaký komponent môžu používať viacerý klienti. Každý klient si však vytvorí vlastný objekt daného komponentu.



Obr. 4.4: Architektúra klient - server v LIRKIS G-CVE.

5 Návrh riešenia

V systéme LIRKIS G-CVE je použitý softvérový rámec *A-frame* [5], ktorým sú vytvorené entity. Pre zdieľanie týchto entít v CVE využívame technológiu *networked-aframe* [6].

Virtuálne prostredie LIRKIS má mať podporu pre čo najviac pripojených klientov a zároveň má poskytovať čo najširší výber platforiem, ktoré môže používateľ použiť pre pripojenie do CVE. Z analýzy už vieme, že systém, ktorý podporuje viacerých klientov, by mal mať aj klientsku rolu administrátora. Administrátor má rozšírené práva v CVE a má možnosť manažovať klientov s bežnými právami.

Pre tieto potreby si v tejto kapitole navrhneme administrátorské rozhranie, ktoré je možné vložiť do ľubovoľnej existujúcej scény. Administrátorom je klient, ktorý získava informácie o ostatných pripojených klientoch. Tiež má k dispozícii zoznam pripojených klientov, zoznam klientov s ich aktuálnou polohou v CVE a zároveň si dokáže získať a prezrieť históriu pripojených klientov.

Najprv si priblížime LIRKIS G-CVE ako celok a moduly ďalších riešiteľov ([26] [27]), ktorý tento školský rok pracujú v tomto CVE, ale riešia iné problémy.

5.1 Kolaboratívne prostredie v laboratóriu LIRKIS

V rámci tejto práce riešime problematiku riadiacej a komunikačnej časti. Zároveň s touto prácou sa vyvíjajú moduly, ktoré umožňujú klientovi využívať v zdieľanom kolaboratívnom prostredí rozšírenú realitu, a to pomocou systému Microsfot HoloLens [26] a mobilného telefónu [27] s operačným systémom Android. Oba moduly poskytujú možnosť pohybu v priestore bez nutnosti použitia ovládača. Klient sa tak môže pohybovať v reálnom prostredí a zároveň sa jeho pohyb prenáša do virtuálneho prostredia ostatným pripojeným klientom.

5.2 Modul komunikačnej časti a jeho začlenenie do systému LIRKIS G-CVE

V analýze v sekcii 4.4 sme si už rozobrali architektúru systému LIRKIS G-CVE a aj architektúru klient-server. Základom pri návrhu riešenia bolo použitie komponentov a ich znovu použiteľnosť inými klientmi. Na obrázku 4.4 z analýzy architektúry systému vidíme, že klient1 využíva vlastný komponent, zatiaľ čo klient2 až klientN využívajú rovnaký komponent. Práve preto využijeme túto vlastnosť a navrhneme si komponenty, ktoré využíva len administrátor a komponenty, ktoré využívajú ostatní klienti v CVE.

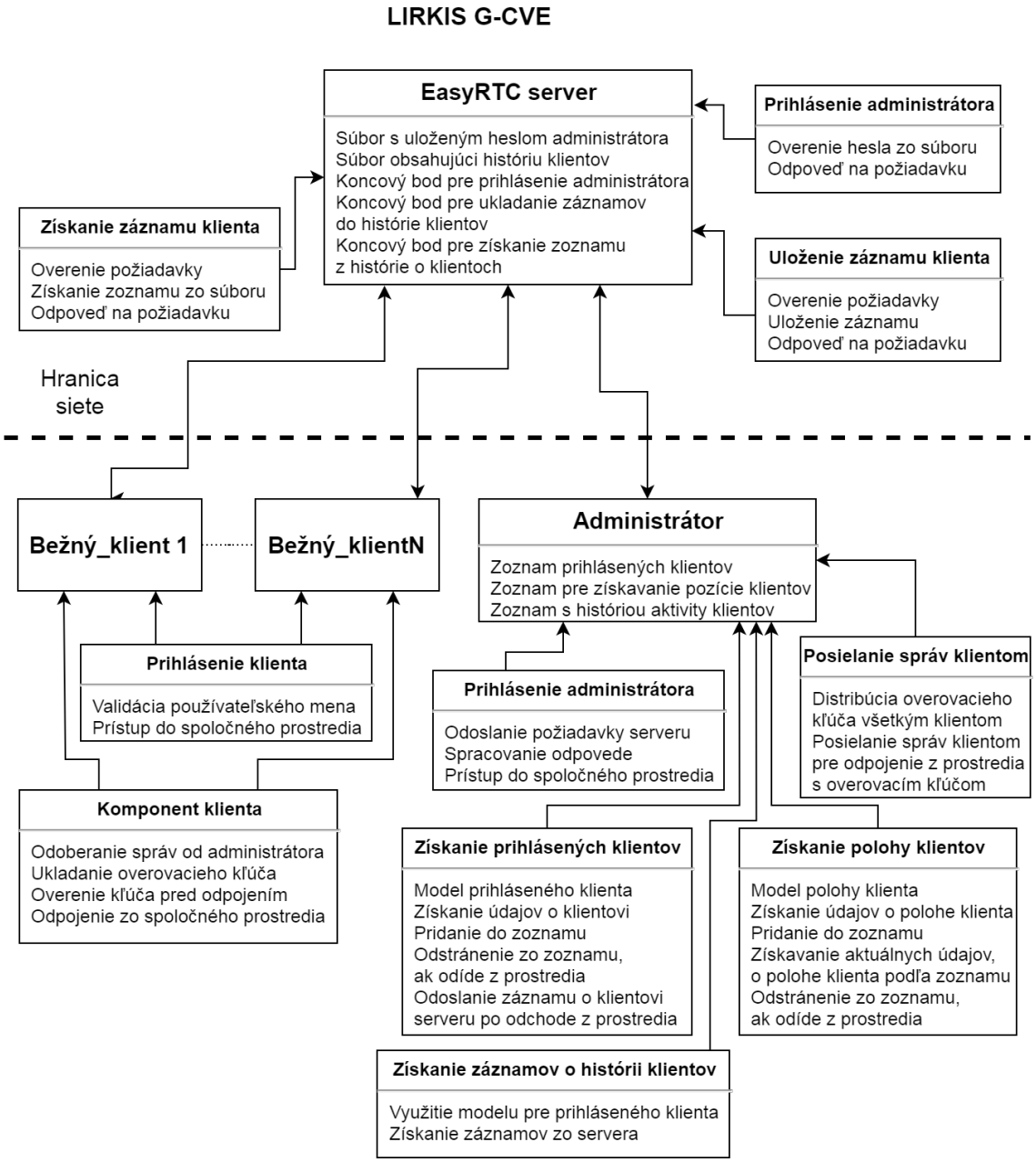
Pre predstavu začlenenia sme si navrhli konceptuálny diagram modulu komunikačnej a riadiacej časti na základe architektúry systému LIRKIS G-CVE z obrázku 4.4. V konceptuálnom diagrame nášho riešenia, (obrázok 5.1), môžeme vidieť systém ako celok s komponentmi servera, administrátora a bežného klienta.

Server obsahuje všetky koncové body, ktoré jednotlivé komponenty využívajú. Obsahuje tiež súbor s heslom pre prihlásenie administrátora a súbor pre ukladanie a získavanie histórie klientov. Na server sa pripájajú tak klienti ako aj administrátor, no komponenty využíva len administrátor. Slúži aj ako signalizačný server pre nadviazanie peer-to-peer spojenia medzi všetkými klientmi a zároveň pomocou EasyRTC a networked-aframe vytvára spoločný virtuálny priestor, do ktorého sa všetci klienti pripoja.

Bežní klienti majú len komponenty na úrovni webového prehliadača. Validácia používateľského mena prebieha len v prehliadači klienta a až pri vstupe do spoločného prostredia kontaktuje server. Prijímanie správ je možné až po pripojení klienta do spoločného prostredia.

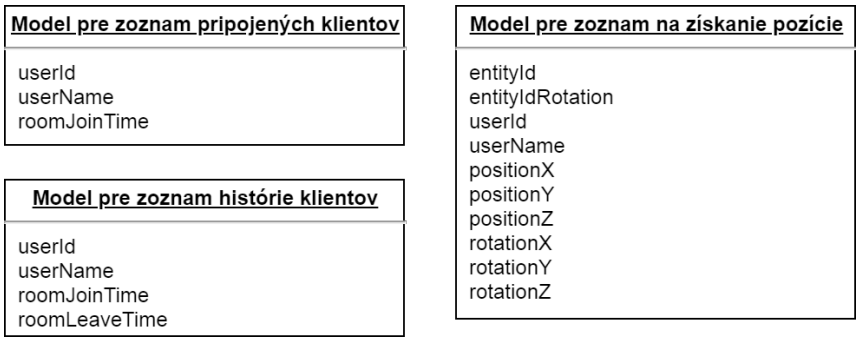
Administrátor obsahuje tri zoznamy, ktoré jednotlivé komponenty naplnia dátami. Najprv je tam komponent pre prihlásenie, ktorý využíva koncový bod servera a komponent pre prihlásenie administrátora. Po úspešnom prihlásení sa rozposlú správy všetkým klientom s overovacím kľúčom administrátora.

Následne sa môžu začať plniť zoznamy. V komponente pre získanie záznamov o histórii požiadavkou na server získame históriu klientov v prostredí. Zoznam naplníme pomocou rozšíreného modelu pre prihlásených klientov.



Obr. 5.1: Konceptuálny diagram modulu komunikačnej a riadiacej časti v systéme LIRKIS G-CVE.

V komponente pre získanie pripojených klientov vkladáme do zoznamu objekty pripojených klientov, použitím modelu pre prihlásených klientov. Pre získanie polohy klientov potrebujeme naplniť zoznam pre pozície na základe prihlásených klientov. Zoznam naplníme objektmi z modelu pre získanie polohy klienta. Jednotlivé modely nájdeme na obrázku 5.2.



Obr. 5.2: Modely objektov vkladaných do jednotlivých zoznamov.

Komponenty pre získavanie polohy a prihlásených klientov sa starajú aj o odstránenie jednotlivých klientov zo zoznamov po ich odchode z CVE. Komponent posielania správ využijeme aj pri poslaní súkromnej správy klientovi, ktorého chceme odpojiť z CVE.

Navrhnuté používateľské rozhranie je doplnkom pre lepšiu demonštráciu fungovania riešenia. Riešenie má byť použiteľné aj bez používateľského rozhrania. Navrhnuté riešenie má byť implementované tak, aby bolo ľahko znovu použiteľné pre ľubovoľnú scénu.

Keď už máme predstavu o systéme ako celku, môžeme si detailnejšie navrhnúť a popísať jednotlivé komponenty servera, administrátora a bežného klienta. Najprv si však upresníme rolu administrátora v systéme LIRKIS G-CVE.

5.3 Administrátor v systéme LIRKIS G-CVE

V systémoch Teachyverse, Circles, Tele-Board Prototyper alebo IMVMM je administrátor klient s rozšírenými funkcionalitami. Aj v našom riešení je administrátorom klient. V našom riešení ho od ostatných klientov rozlišujeme osobitnou stránkou pre prihlásenie. Pre zabezpečenie integrity je pre vstup do prostredia vytvorený spôsob autentifikácie – zadaním hesla, či daný klient je skutočne administrátor alebo sa len zaňho vydáva. Stránka je verejne dostupná, ale len administrátor vie presné znenie internetovej adresy (URL) a heslo.

5.4 Návrh komponentov servera

V súvislosti s komponentmi, ktoré využíva administrátor pre prihlásenie, získavanie a ukladanie histórie aktivity klientov, sme navrhli komponenty servera. Server obsahuje tri koncové body pre jednotlivé požiadavky zasielané klientom administrátora. Každý koncový bod tvorí funkcionalitu jedného komponentu. Tiež obsahuje súbor so zoznamom o histórii klientov a súbor, kde je uložené heslo pre administrátora. Koncové body pre získavanie a ukladanie histórie klientov zahŕňajú mechanizmus zabezpečujúci odoslanie odpovede len prihlásenému administrátorovi. Pri všetkých troch koncových bodoch sa očakáva odpoveď servera na strane klienta, ktorá sa skladá zo zoznamu klientov alebo len vracia status požiadavky. Preto sme si navrhli aj formu odpovede jednotlivých koncových bodov.

Pre bežného klienta nie je potrebné implementovať žiadne komponenty ani koncové body na strane servera. Všetky potrebné komponenty pre komunikáciu so serverom a vytvorenie dátového kanálu, sú už implementované v technológiách WebRTC (4.2) a EasyRTC (4.3).

5.4.1 Komponent pre prihlásenie s overením hesla

Prvý komponent obsahuje koncový bod, ktorý zabezpečuje prihlásenie administrátora. V požiadavke od klienta obdrží používateľom zadané heslo, ktoré je porovnané s heslom uloženým v súbore. Následne odošle klientovi výsledok komparácie, či je zadané heslo zhodné s tým, ktoré je v súbore. V prípade zadania správneho hesla odošle číselný status požiadavky 200 – OK, úspešné prihlásenie. V prípade zadania zlého prístupového hesla odošle číselný status odpovede 401 – neoprávnený prístup. V prípade inej chyby alebo neúspešného prístupu k súboru odošle len číselný status odpovede 500 – chyba servera.

5.4.2 Komponent pre uloženie záznamu klienta

Druhý komponent a zároveň koncový bod slúži na ukladanie záznamov, ktoré odošle administrátor v tele požiadavky. Telo požiadavky obsahuje záznam, opísaný v komponente administrátora pre ukladanie a získavanie histórie

o aktivite klientov (5.5.5). Po obdržaní požiadavky server skontroluje, či požiadavka prišla od prihláseného administrátora. Následne otvorí súbor, kde sa ukladajú záznamy o aktivitách klientov a pripojí záznam z požiadavky na koniec tohto zoznamu. Súbor potom uloží a zatvorí. Po uzavretí súboru odošle odpoveď, buď o úspešnom alebo neúspešnom vložení nového záznamu do zoznamu. V prípade úspešného vloženia nového záznamu odošle číselný status 200 – OK. V prípade neautorizovaného prístupu odošle len číselný status odpovede 401 – neoprávnený prístup. V prípade inej chyby alebo neúspešného prístupu k súboru odošle len číselný status odpovede 500 – chyba servera.

5.4.3 Komponent pre získanie záznamov o klientoch

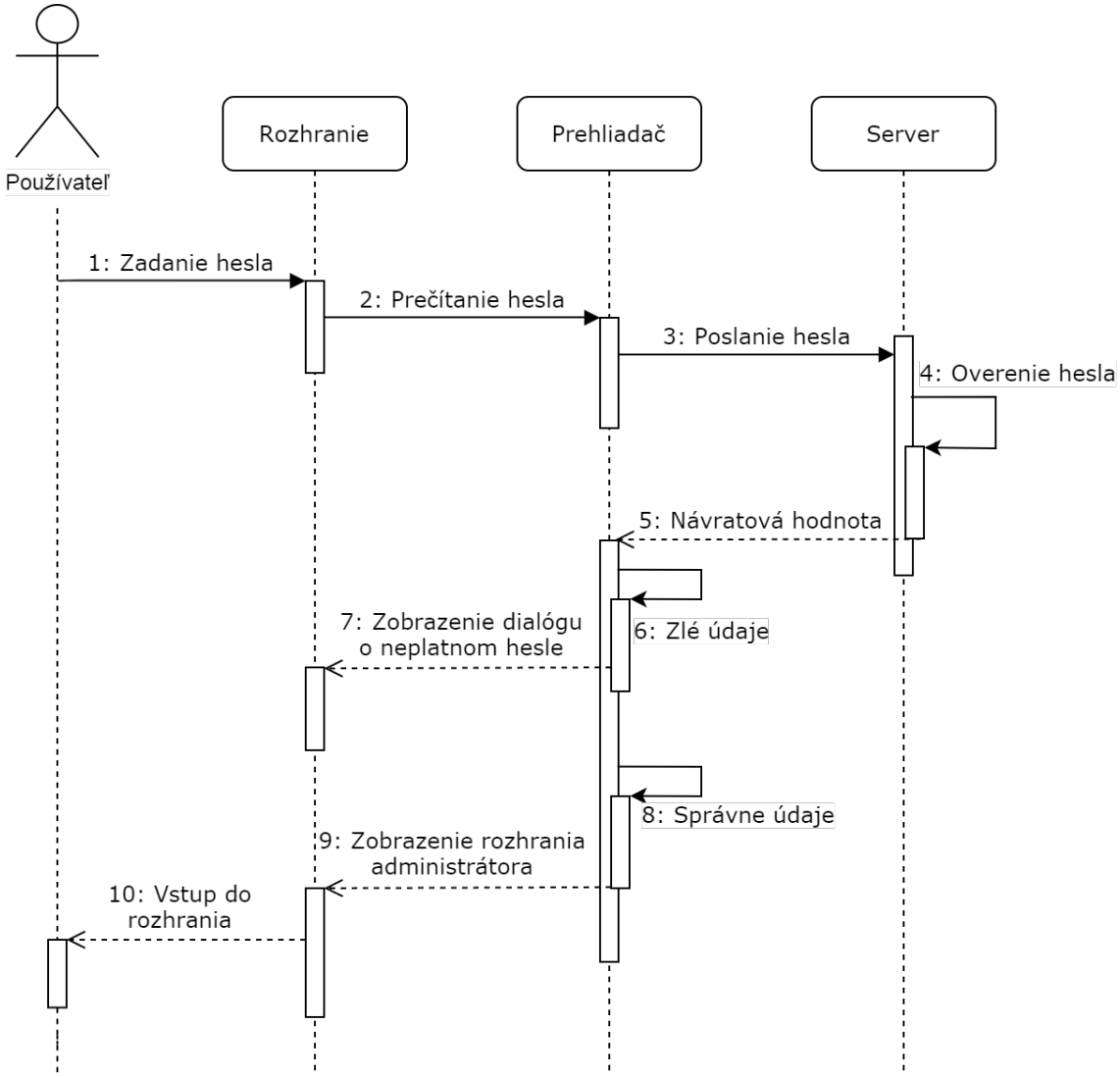
Tretí komponent zahŕňa tretí koncový bod, ktorý sa stará o načítanie zoznamu o histórii klientov a jeho následne odoslanie klientovi. Po zachytení požiadavky z nej server zistí, či jej odosielateľ je prihlásený administrátor. Následne si zo súboru vyberie záznamy o aktivite. Vzhľadom na možnosť, že daný súbor môže obsahovať príliš veľa záznamov, vyberáme z neho len 50 najnovších záznamov. V prípade, že je zoznam kratší pošlú sa všetky záznamy. Po úspešnom prečítaní záznamov sa odošle odpoveď klientovi, ktorá obsahuje daný zoznam. V prípade neúspešného prístupu sa odošle len číselný status odpovede 500 – chyba servera. V prípade neautorizovaného prístupu sa odošle len číselný status odpovede 401 – neoprávnený prístup.

5.5 Návrh komponentov administrátora

Keďže náš systém LIRKIS G-CVE využíva A-Frame, navrhneme si a použijeme komponenty samostatne pre každú jednu funkcionalitu. Komponenty sú súčasťou klientskej časti. Využijeme teda funkcionality, ktoré nám poskytuje EasyRTC a networked-aframe. Komponenty sú opakovane použiteľné moduly alebo dátové kontajnery, ktoré môžu byť pripojené k entitám na zabezpečenie vzhľadu, správania sa alebo funkcionality. Entity sú objekty, ku ktorým je možné pripojiť komponenty. Komponenty pre administrátora prideľujeme jeho HTML entite.

5.5.1 Komponent pre prihlásenie administrátora

Tento komponent rieši skôr opísanú problematiku odlíšenia administrátora od ostatných klientov. Predtým ako administrátor vstúpi do CVE, overujeme, či ide o používateľa oprávneného byť administrátorom. To znamená, že klient sa nachádza vo virtuálnom prostredí, ale nie je súčasťou zdieľanej scény. Používateľovi sa zobrazí modálny dialóg, kde zadáva heslo pre vstup do administrátorského rozhrania. Heslo pre prístup je uložené na strane servera. Na obrázku 5.3 je zobrazený sekvenčný diagram prihlásenia.



Obr. 5.3: Sekvenčný diagram prihlásenia administrátora do systému LIRKIS G-CVE.

Používateľ zadá do zobrazeného dialógu svoje heslo. Prehliadač prečíta heslo a odošle ho na server, kde server overí, či zadané heslo je správne. Server následne vracia odpoveď prehliadaču, ktorý ju spracuje a vyhodnotí správnosť návratovej hodnoty. Následne sa zobrazí buď dialóg o neplatnom hesle, alebo sa zobrazí používateľské rozhranie administrátora a pripojí klienta do zdieľanej scény.

Nakoniec si zhrnieme, čo všetko obsahuje komponent pre prihlásenie administrátora:

- Zobrazenie dialógu alebo inej možnosti pre zadania hesla.
- Overenie identity administrátora, pomocou hesla, ktoré zadá pre vstup do systému.
- Overenie hesla na strane servera.
- Pripojenie do CVE sa realizuje až po overení správnosti hesla. Dovtedy je klient v lokálnom virtuálnom prostredí.

5.5.2 Komponent pre získavanie zoznamu pripojených klientov

Komponent poskytuje administrátorovi lepší prehľad o pripojených klientoch v CVE. Navrhli sme preto komponent, ktorý obsahuje zoznam pripojených klientov. Samotný networked-aframe nám poskytuje metódu, pre získavanie aktuálne pripojených klientov. Náš zoznam je ešte rozšírený o používateľské meno. Používateľské meno klient zadáva pri vstupe do CVE. Tiež je zobrazené nad hlavou každého avatara klienta, takže vieme identifikovať daného klienta zo zoznamu. Zoznam obsahuje aj čas pripojenia klienta do CVE, aby mal administrátor prehľad, ako dlho sa klient nachádza v prostredí. Do tohto zoznamu nezahŕňame administrátora.

Atribúty objektu klienta uloženého v zozname pripojených klientov sú:

- Identifikačné číslo klienta, získané pomocou networked-aframe.
- Používateľské meno, ktoré zadal používateľ pri vstupe do CVE.
- Čas pripojenia používateľa do CVE.

5.5.3 Komponent pre posielanie súkromnej správy inému klientovi

V systéme Teachyverse môže lektor stíšiť konverzáciu všetkých študentov, ak je to nutné. Práve preto má administrátor v našom systéme funkciu pre obmedzenie nechceného správania klientov v CVE.

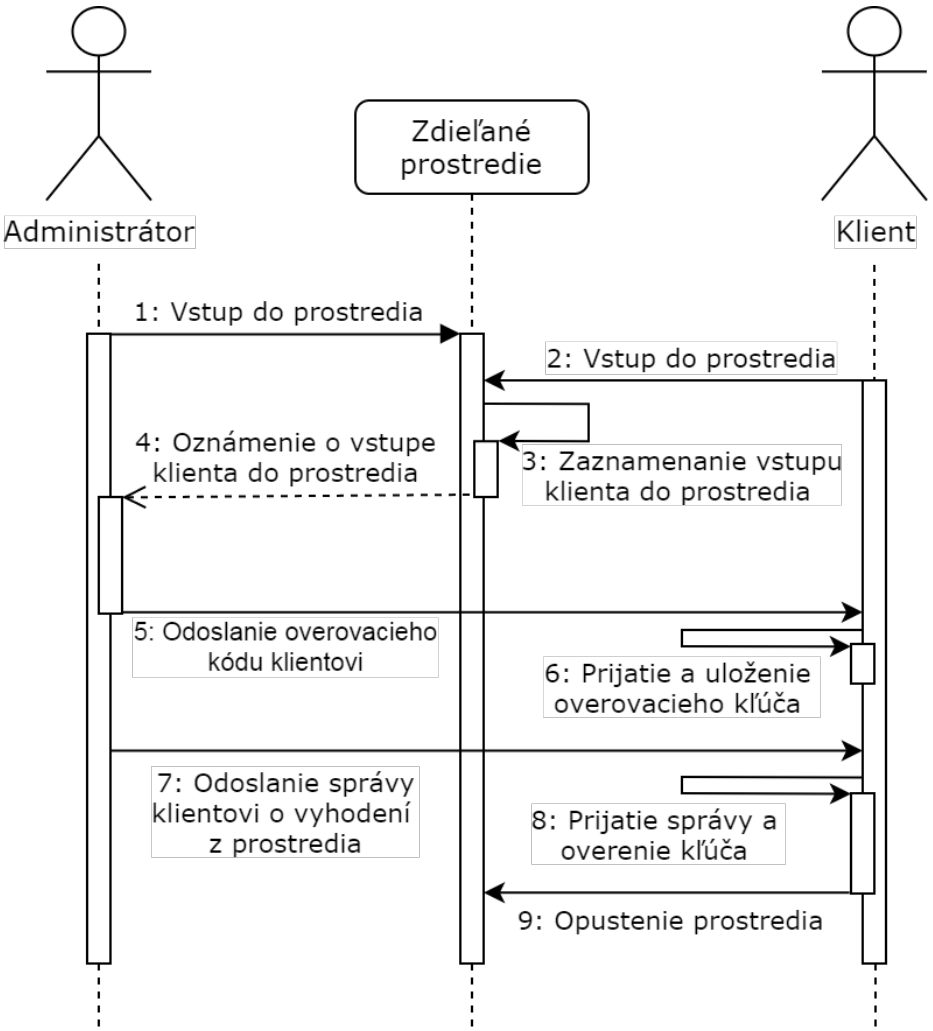
Navrhli sme komponent, ktorým môže administrátor odpojiť klienta z CVE. Komponent využíva existujúci zoznam pripojených klientov. Administrátor po pripojení klienta s ním vytvorí peer-to-peer spojenie, ako sme si vysvetlili v analýze použitých technológii. Využitím technológie EasyRTC sa s každým klientom vytvorí peer-to-peer spojenie a dátový kanál. Pre vyhodenie klienta zo scény, je potrebné vedieť jeho identifikačné číslo, ktoré je v zozname pripojených klientov. Po výbere klienta, ktorého chceme vyhodiť z CVE mu pošleme správu. S využitím networked-aframe pridávame kanálu správ vlastný názov, pre odlíšenie od ostatných synchronizačných správ, ktoré každý klient pravidelne prijíma. Pre zabezpečenie dôveryhodnosti, že správa pre odpojenie klienta zo scény bola odoslaná administrátorom, sme vytvorili samostatný kanál správ. Týmto kanálom správ administrátor po pripojení klienta do scény odosiela v tele správy kód. Tento overovací kód odosiela všetkým pripojeným klientom v scéne. Overovací kód obsahuje prvých 10 číslic networked-aframe identifikačného čísla administrátora, aby si klient overil odosielateľa správy pre odpojenie zo scény.

Na sekvenčnom diagrame 5.4 vidíme, že po vstupe klienta do prostredia mu administrátor priamo odosiela správu s overovacím kódom. Klient následne príjme a uloží overovací kľúč. Tiež je znázornené aj poslanie správy o vyhodení klienta z prostredia. Tak ako správa obsahujúca overovací kľúč tak aj správa o vyhodení z prostredia sa posiela priamo. Po prijatí správy, klient overí odosielateľa správy podľa overovacieho kľúča a ak správa bola odoslaná administrátorom opustí zdieľané prostredie. Rozdiel medzi správou s overovacím kľúčom a správou o vyhodení z prostredia je, že správa o vyhodení sa posiela priamo vybranému klientovi, no overovací kľúč sa posiela všetkým pripojeným klientom.

Podľa vyššie uvedeného rozboru si zhrnieme, čo všetko obsahuje komponent pre posielanie súkromnej správy inému klientovi:

- Overovací kód z networked-aframe identifikačného čísla administrátora.

- Samostatný kanál správ pre posielanie overovacieho kódu všetkým pripojeným klientom.
- Samostatný kanál pre posielanie správ priamo klientovi, ktorého chceme zo scény odpojiť.



Obr. 5.4: Sekvenčný diagram pre posielanie správ medzi klientom a administrátorom.

5.5.4 Komponent pre získavanie aktuálnej polohy klientov

Pre získavanie aktuálnej polohy potrebujeme využiť informácie o zdieľaných entitách ostatných klientov. Informácie o zdieľaných entitách má networked-aframe. Ten ich poskytuje všetkým klientom v samostatnom dátovom kanáli. Každý klient si svoju scénu generuje sám a informácie o zmene polohy poskytuje networked-aframe každému klientovi, aby si vedel znovu vygenerovať scénu s údajmi o aktuálnej polohe každého klienta v priestore. V našom riešení je vytvorený samostatný zoznam klientov, ktorý obsahuje identifikátor klienta, používateľské meno klienta a identifikačné číslo zdieľanej entity, ktorej polohu chceme sledovať. Následne sa údaje o polohe získavajú v slučke pre každý jeden objekt v zozname. Identifikačné číslo entity je časť avatara klienta, ktorej polohu chceme v CVE sledovať. Zaujímať nás bude pozícia vo všetkých troch osiach (X,Y,Z) a tiež rotácia.

Na základe návrhu, objekt uložený v zozname s pozíciami klientov má atribúty:

- Identifikačné číslo klienta, získané pomocou networked-aframe.
- Používateľské meno, ktoré zadal používateľ pri vstupe do CVE.
- Identifikačné číslo zdieľanej entity zaregistrované v networked-aframe.
- Poloha entity v osi X,Y,Z – získavaná zo zdieľanej entity.
- Rotácia entity v osi X,Y,Z – získavaná zo zdieľanej entity.

5.5.5 Komponent pre ukladanie a získavanie histórie o aktivite klientov

Pre vytvorenie tohto komponentu sme sa inšpirovali riešení systému Tele-Borad Prototyper, v ktorom sa každá zmena navrhovaného prototypu ukladala, aby sme si mohli spätne pozrieť každú fázu vývoja prototypu. V našom riešení ukladáme záznamy o tom, kedy a ako dlho boli klienti v CVE. Pre získavanie potrebných údajov sme využili zoznam pripojených klientov. Pri odpojení klienta z CVE je potrebné do objektu v zozname pripojených klientov pridať záznam o čase odpojenia klienta z CVE. Tento objekt sa následne odosiela serveru, ktorý ho uloží. Záznamy o aktivitách administrátora sa neposielajú.

Ak máme uložené údaje o aktivite na serveri, môžeme ich poskytnúť administrátorovi v klientskom rozhraní. Preto je vytvorený zoznam klientov, ktorý obsahuje údaje získané zo servera. Naplnenie tohto zoznamu sa realizuje hneď po autentifikácii administrátora a zoznam je administrátorovi dostupný po celý čas pobytu v CVE.

V komponente pre ukladanie a získavanie histórie o aktivite klientov je potrebné:

- Získať zoznam o aktivite klientov zo servera.
- Nájsť práve odhláseného klienta v zozname pripojených klientov a pridať mu atribút – čas odpojenia.
- Vytvoriť žiadosť na server, ktorá obsahuje práve odpojeného klienta.
- Vymazať klienta zo zoznamu pripojených klientov a zo zoznamu na získavanie polohy.

5.6 Návrh komponentov bežného klienta

V tejto sekcii sa zaoberáme návrhom komponentov pre bežného klienta. Pri administrátorovi počítame so skúseným používateľom, ktorý tiež disponuje dostatočným výkonom. Naše riešenie je dostupné pre čo najviac platforiem, predovšetkým pre bežných klientov. Entita klienta tak obsahuje čo najmenej komponentov a je nenáročná na výkon zariadenia. Zdieľanie entít medzi ostatných pripojených klientov je vyriešené použitím technológie networked-aframe. Bežnému klientovi sme navrhli komponent pre vstup do CVE a komponent pre prijatie správy od administrátora.

5.6.1 Komponent pre prihlásenie klienta

V tomto komponente sme navrhli ako by mal do CVE vstúpiť bežný klient. Po načítaní prostredia sa klient nachádza iba v lokálnom virtuálnom prostredí. Po načítaní celého virtuálneho prostredia sa používateľovi zobrazí modálny dialóg, v ktorom zadá používateľské meno.

Zadané používateľské meno má tieto obmedzenia:

1. Nesmie byť prázdne.
2. Nesmie obsahovať medzeru.
3. Nesmie mať viac ako 10 znakov.
4. Nesmie byť zhodné ani obsahovať sekvenciu znakov, ktorá sa zhoduje s používateľským menom administrátora.

Ak zadané používateľské meno neporušuje žiadnu z týchto podmienok, tak je klient pripojený do zdieľanej scény. Zadané používateľské meno sa počas celej doby pobytu zobrazuje nad hlavou avatara konkrétneho klienta. Tiež je viditeľné aj ostatným používateľom, nie len administrátorovi.

5.6.2 Komponent pre prijímanie správ od iného klienta

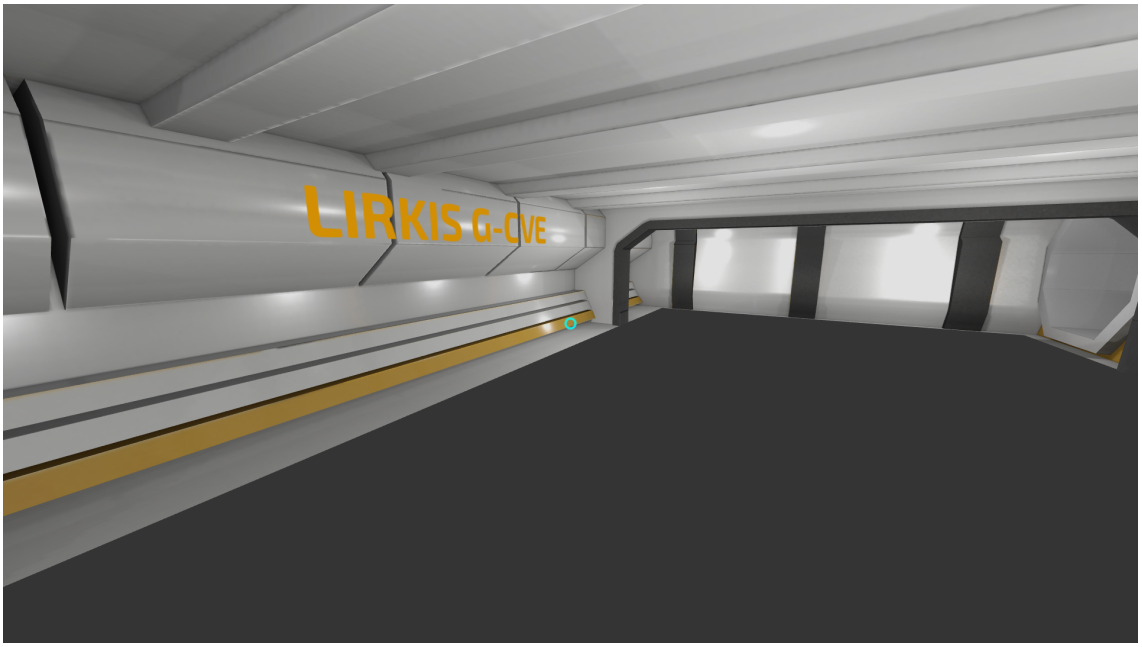
Tento komponent klienta zabezpečuje prijatie správy od administrátora. Táto funkcia je implementovaná v technológii networked-aframe. Po vytvorení dátového kanálu nám umožňuje špecifikovať názov správ, ktoré odchytávame. Po obdržaní správy sa klient odpojí zo zdieľaného virtuálneho prostredia. Klient zostane v lokálnom virtuálnom prostredí a do konzoly prehliadača obdrží správu o tom, že bol vyhodený zo scény. V tomto komponente je najdôležitejšia vec synchronizácia s komponentom administrátora, aby klient pre odpojenie zo scény odoberal správny kanál správ od administrátora. Pre zabezpečenie dôveryhodnosti, že správa pre odpojenie prišla od administrátora, je potrebné odoberať aj kanál správ, v ktorých administrátor posiela overovací kód. Pomocou neho overujeme odosielateľa správy o odpojení zo scény.

Funkcionalita komponentu klienta pre prijímanie správ zahŕňa:

- Odoberať kanál správ pre získavanie overovacieho kódu.
- Odoberať kanál správ pre získavanie správ pre odpojenie zo scény.
- Overiť odosielateľa správy pre odpojenie zo scény pomocou overovacieho kódu.

5.7 Návrh riešenia pre testovaciu scénu

Celé naše riešenie je integrované v testovacej scéne, kde sú použité všetky doteraz navrhnuté komponenty a zároveň sú v nej integrované aj moduly ostatných riešiteľov spomínaných v sekcii 5.1. Pre tieto potreby bola zvolená scéna intersim, zobrazená na obrázku 5.5. Autorom scény je Ing. Marián Hudák.



Obr. 5.5: Scéna intersim systému LIRKIS.

Scéna poskytuje zdieľané virtuálne prostredie, kde môžeme integrovať jednotlivé moduly a komponenty. Na začiatku má scéna len jedného klienta, ktorý si môže zadať používateľské meno a výber počiatočnej pozície v scéne. V scéne sa nachádza aj cBot – robot s automatickým pohybom po scéne. Pre vstup do scény ako cBot, je nutné zadať heslo ako je to u administrátora. Scéna nám tiež poskytuje šablónu pre vytvorenie avatarov klientov. Pre integráciu všetkých troch modulov bolo potrebné odlišovať zariadenie a prehliadač používateľa, s ktorým sa chce pripojiť do scény. Napríklad pre zariadenie HoloLens je potrebné použiť iného avatara ako u ostatných používateľov. V rámci tejto scény bolo potrebné navrhnúť aj používateľské rozhranie celého riešenia. Začali sme návrhom úvodnej stránky, prešli sme na návrh používateľského rozhrania administrátora a taktiež sme navrhli modálny dialóg pre prihlásenie administrátora, bežného klienta a cBot-a.

5.7.1 Návrh úvodnej stránky

V tejto sekcii predstavíme návrh úvodnej stránky projektu. Tá obsahuje názov projektu a tlačidlá pre výber typu klienta, s ktorým chceme vstúpiť do CVE. Tlačidlá presmerujú používateľa na stránku administrátora, bežného klienta alebo cBot-a. V rámci úvodnej stránky sa vyrieši aj problematika výberu avatara podľa platformy, s ktorou sa chce používateľ prihlásiť do CVE.

5.7.2 Návrh modálneho dialógu pre prihlásenie

Modálny dialóg pre prihlásenie je použitý pri prihlásení administrátora, cBot-a a bežného klienta. Dialóg sa používateľom zobrazí po načítaní scény. Scéna je však len lokálna. Pre pripojenie do globálnej scény sa musia splniť podmienky prihlásenia určené pre vybraného klienta. Pre administrátora a cBot-a je potrebné použiť komponent pre prihlásenie administrátora navrhnutý v sekcii 5.5.1. Používateľské meno administrátora a cBot-a je dané a uložené na serveri. V prípade bežného klienta overujeme len jeho validitu s požiadavkami, ktoré sme navrhli v sekcii 5.6.1. Nie je potrebná žiadna priama komunikácia zo serverom, celá validácia prebieha len na strane klienta.

Oba typy dialógov obsahujú popis, čo má požívateľ zadať, aby sa dostal do CVE. Dialógy zobrazujú len jeden vstup pre používateľa. V prípade administrátora a cBot-a je to vstup na zadanie hesla. V prípade bežného klienta na zadanie používateľského mena. Dialógy zobrazujú aj hlásenia pri zle zadanom hesle alebo pri zlom používateľskom mene klienta. V oboch typoch dialógov je aj odkaz pre návrat na úvodnú stránku a tlačidlo pre potvrdenie zadaného vstupu.

Navrhnuté dialógy nie sú viazané priamo na scénu. Sú ľahko použiteľné pre ľubovoľnú scénu. Dialógy pre prihlásenie sú grafickým rozhraním komponentu pre prihlásenie administrátora 5.5.1 a komponentu pre prihlásenie klienta 5.6.1. Ak sa však modálne dialógy nepoužijú, tak musí byť možnosť prihlásenia sa cez konzolu prehliadača pre oba typy dialógov.

5.7.3 Návrh používateľského rozhrania administrátora

Používateľské rozhranie administrátora má vhodnou formou reprezentovať údaje v zoznamoch, ktoré sú opísané v komponentoch pre získavanie zoznamu pripojených klientov 5.5.2, pre získavanie aktuálnej pozície klientov 5.5.4 a pre ukladanie a získavanie histórie klientov 5.5.5. Preto sme navrhli tabuľky. Tie zobrazujú údaje z jednotlivých zoznamov.

Tabuľky sú zobrazené na celú šírku obrazovky jednotlivo pod sebou. Prvá je tabuľka zo zoznamom pripojených klientov. Druhou v poradí je tabuľka pre zobrazovanie aktuálnej pozície každého jedného klienta. Treťou je tabuľka zobrazujúca históriu klientov v scéne. Administrátor je súčasťou CVE, takže posledným prvkom grafického rozhrania administrátora je samotná scéna, v ktorej sa nachádza a môže sa v nej pohybovať.

Tabuľky sú implementované tak, že nie sú závislé od scény intersim a sú ľahko použiteľné pre inú scénu. Zároveň však naše riešenie nie je závislé od tabuliek. To znamená, že keď zakážeme používanie tabuliek alebo nebudú zahrnuté v scéne, tak administrátor má možnosť získať všetky zoznamy z konzoly prehliadača.

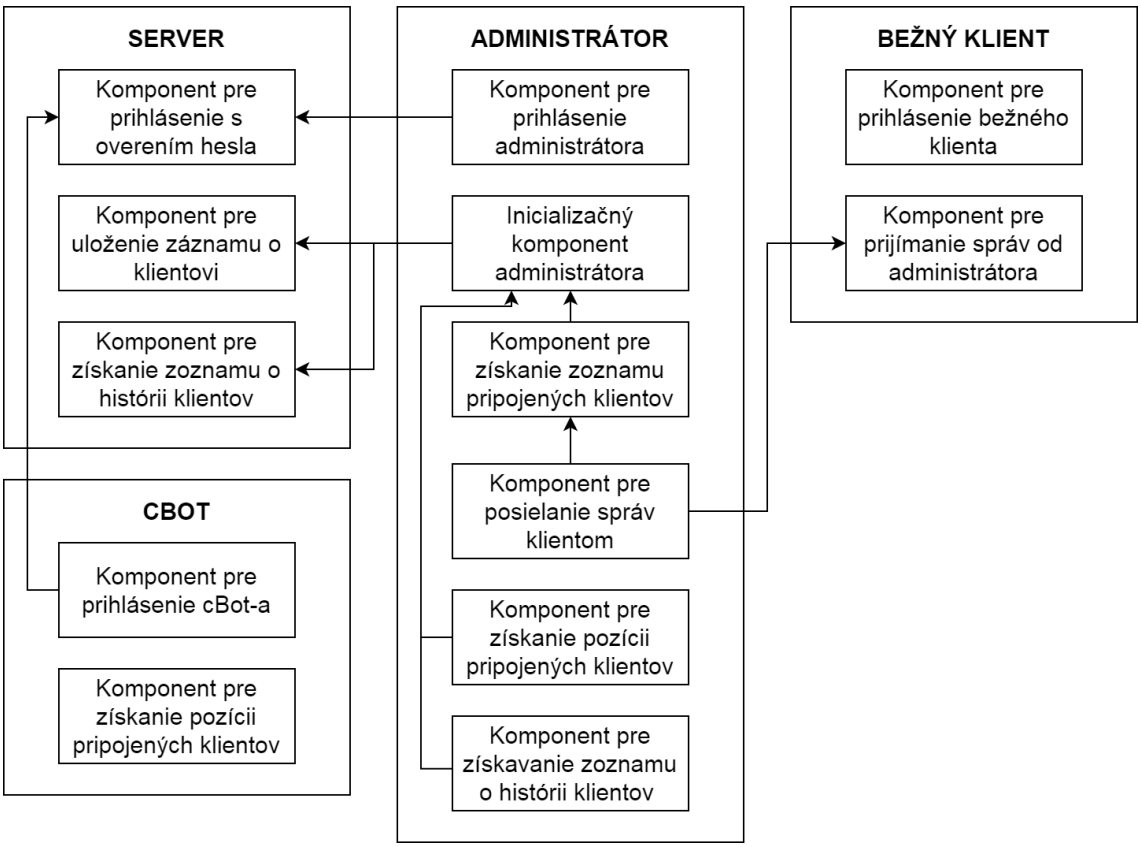
6 Implementácia riešenia

Po navrhnutí riešenia si v tejto kapitole opíšeme implementáciu jednotlivých komponentov do testovacej scény. Tiež si priblížime aj implementáciu používateľského rozhrania pre naše riešenie. Začneme implementáciou komponentov servera. Pokračujeme implementáciou komponentov administrátora a bežného klienta. Pri každom komponente si opíšeme aj implementáciu jeho používateľského rozhrania a integráciu do testovacej scény od úvodnej obrazovky cez modálne dialógy pre prihlásenie až po samotné grafické používateľské rozhranie administrátora.

Na začiatok si priblížime celkovú architektúru systému podľa kontextového diagramu zobrazeného na obrázku 6.1. Podľa konceptuálneho návrhu sme implementovali jednotlivé komponenty serveru, bežnému klientovi, administrátorovi a pre potreby testovacej scény aj cBot-ovi. Komponenty servera sú implementované podľa návrhu a na kontextovom diagrame máme vyznačené, ktoré komponenty klientov ich využívajú. Komponenty servera sú implementované pre potreby administrátora. Funkcionalitu pre prihlásenie s overením hesla na serveri má implementovanú aj cBot, aby nebol voľne prístupný bežným používateľom. Bežný používateľ využíva pre prihlásenie interný komponent pre prihlásenie, ktorý si nevyžaduje komunikáciu so serverom.

Na diagrame vidíme v komponentoch administrátora interné závislosti. Jednou z požadovaných vlastností komponentov administrátora podľa návrhu bolo použitie komponentov nezávisle na sebe. Táto podmienka zostala zachovaná, avšak pre lepšie použitie pre testovaciu scénu bol vytvorený inicializačný komponent, ktorý si priblížime pri opise komponentov administrátora. Špeciálne použitie si vyžaduje komponent pre posielanie správ klientom. Ten je závislý od komponentu pre získanie zoznamu pripojených

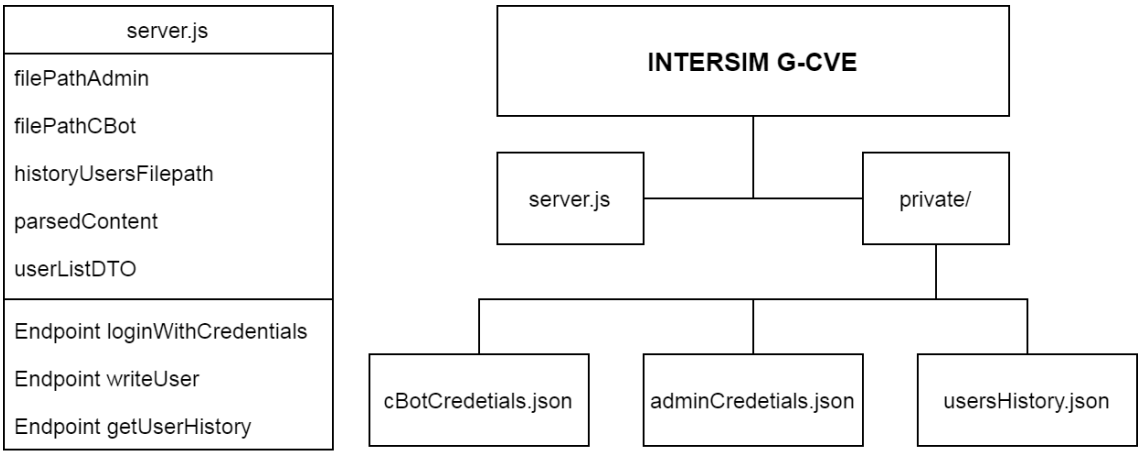
klientov. Taktiež komponent komunikuje priamo s komponentom bežného klienta pre prijatie správ od administrátora. Bližšie si ich opíšeme v sekcii implementácie komponentov administrátora.



Obr. 6.1: Kontextový diagram využitia implementovaných komponentov.

6.1 Implementácia komponentov servera

Z návrhu riešenia už máme rozdelenie jednotlivých komponentov pre administrátora aj bežných klientov. Administrátor využíva priamo komponenty servera (pozri obrázok 6.1), preto je potrebné najprv implementovať komponenty servera, aby sme mohli požiadavky z komponentu administrátora adresovať už vytvorenému koncovému bodu servera. Implementácia komponentov servera sa nachádza v súbore server.js v hlavnom adresári celého systému (pozri obrázok 6.2). Ostatné potrebné priečinky si vytvoríme v súkromnom (*private*) adresári.



Obr. 6.2: Štruktúra skriptu server.js a schéma štruktúry priečinkov so súbormi potrebných pre implementáciu komponentov servera.

6.1.1 Komponent pre prihlásenie s overením hesla

Ako prvý si implementujeme komponent pre prihlásenie s overením hesla. Tento komponent zahŕňa vytvorenie koncového bodu na strane servera, vytvorenie súkromného adresára, v ktorom sa nachádza heslo pre vstup klienta s potrebou obmedzenia vstupu v našom prípade administrátora a vytvorenie logiky pre overovanie správnosti zadaného hesla. Na začiatku si vytvoríme v súkromnom (*private*, pozri 6.2) adresári servera súbor, do ktorého uložíme heslo pre vstup administrátora do prostredia. Súbor je typu JSON. Názov atribútu pre získanie hodnoty hesla je *password*. Hodnota hesla je ľubovoľná, avšak názov atribútu musí zostať rovnaký. Súbor s uloženým heslo administrátora má názov *adminCredentials.json*.

Po vytvorení potrebného súboru, sme ďalej vytvorili pomocou rámca express, koncový bod *loginWithCredentials*, pre požiadavky typu POST zo strany klienta. Funkcia spätného volania pre POST požiadavku obsahuje logiku overenia hesla zadaného používateľom voči heslu uloženému v súbore. Heslo zadané používateľom získavame z tela požiadavky. Po zistení, či sa zadané heslo zhoduje s uloženým, odošleme klientovi odpoveď. Kladná odpoveď obsahuje status 200 – OK, autentifikácia administrátora prebehla v poriadku a pripojí sa do CVE. Záporná odpoveď obsahuje status 401, ktorý znamená neoprávnený prístup. Spracovanie odpovede zo servera rieši komponent pre prihlásenie administrátora.

Pre testovaciu scénu sme implementovali komponent pre prihlásenie cBot-a. Pre prihlasovanie cBot-a sme vytvorili osobitný súbor s názvom *cBotCredentials.json*. Súbor je umiestnený v súkromnom adresári servera 6.2. Pre prihlásenie cBot-a nie je potrebný samostatný koncový bod. Podľa tela požiadavky zisťujeme, či prišla od cBot-a alebo administrátora. Podľa toho sa rozhodujeme, ktorú z konštánt pre cestu k súboru si vyberieme. Implementácia pre overenia hesla je rovnaká ako v prípade administrátora.

6.1.2 Komponent pre uloženie záznamu klienta

Podľa návrhu má tento komponent ukladať záznam o aktivite klienta, ktorý práve opustil scénu. Pre tento komponent sme vytvorili súbor, do ktorého sa ukladajú záznamy o klientoch. Tak ako aj v prípade súborov pre uloženie hesla, aj súbor s históriou o aktivite klientov vytvoríme v súkromnom adresári (6.2), aby k nemu nemali klienti priamy prístup. Meno súboru je *usersHistory.json*. Typ JSON je pre jednoduchšie ukladanie objektov klientov, ktoré nám pošle klient – administrátor. Na začiatku obsahuje iba prázdny zoznam.

Pomocou rámca express sme vytvorili ďalší koncový bod *writeUser*, pre požiadavky typu POST. Funkcia spätného volania pre POST požiadavku obsahuje spracovanie tela požiadavky. V tele požiadavky očakávame objekt zo zoznamu pre históriu klientov podľa modelu z obrázka 5.2. Vo funkcii sme definovali cestu k súboru podľa konštanty *historyUsersFilepath* a využívame pomocnú premennú *parsedContent* pre prevod z reťazca na objekt. Pred otváraním súboru si najprv overíme hlavičku požiadavky. Hlavička nám potvrdí, či prišla od prihláseného administrátora. Ak áno, do zoznamu získaného zo súboru pripojíme objekt do zoznamu a uložíme súbor. Ak počas otvárania súboru alebo počas zápisu nastane chyba, tak sa odošle odpoveď so statusom 500 – interná chyba servera. Ak požiadavka neprišla od prihláseného administrátora, tak sa v odpovedi odošle status 401 – neoprávnený. V prípade úspešného zapísania údajov a zároveň uloženia súboru sa v odpovedi odošle status 200 – OK.

6.1.3 Komponent pre získanie záznamov o klientoch

Z návrhu vyplýva, že komponent má zabezpečovať prečítanie záznamov zo súboru a vrátenie 50 najnovších záznamov administrátorovi. Preto sme vytvorili koncový bod *getUserHistory*, tentoraz typu GET. Cestu k súboru, kde sa záznamy nachádzajú je definovaná konštantou *historyUsersFilepath*. Súbor už existuje, pretože bol vytvorený v komponente pre ukladanie záznamu klienta. Tiež je použitá rovnaká metóda overenia, či požiadavka prišla od prihláseného administrátora a to pomocou hlavičky požiadavky. Dôležitý je pre tento komponent výber záznamov zo súboru, ktorého implementácia je uvedená v zdrojovom kóde nižšie.

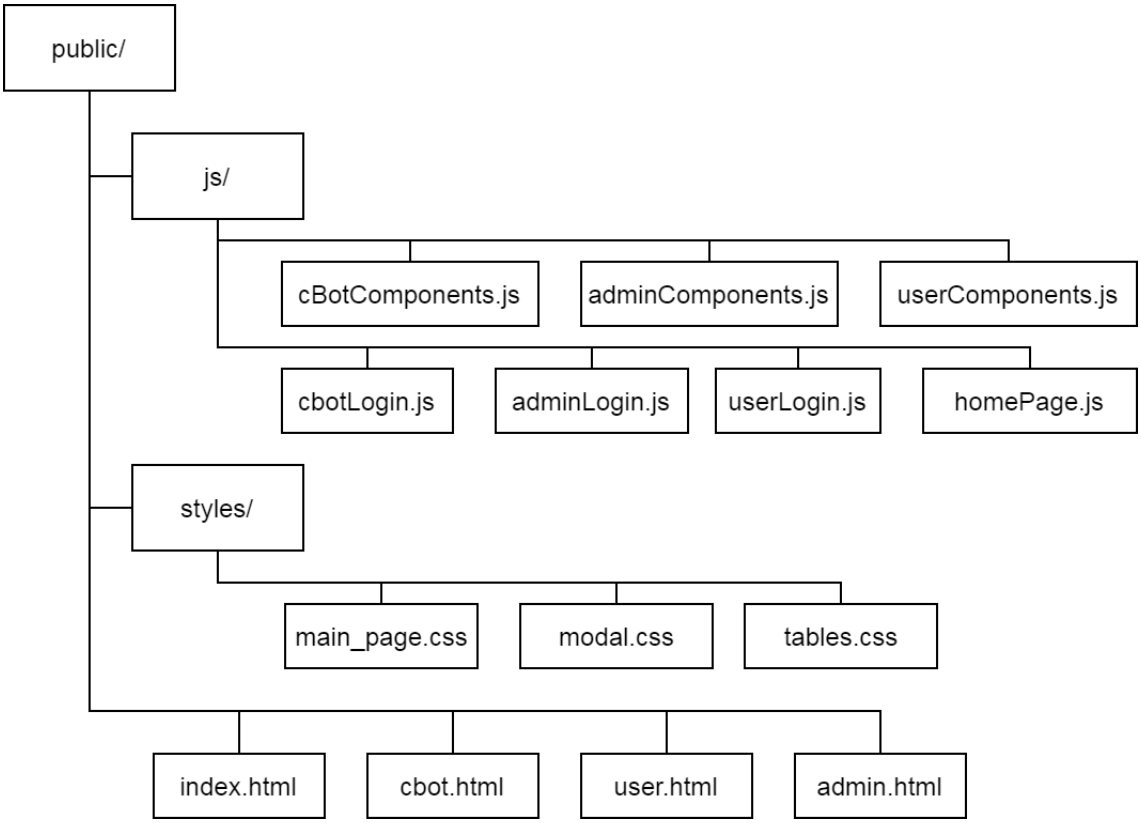
```
parsedUsersFromFile = JSON.parse(data);
const startIndex = parsedUsersFromFile.length - 50;
if (startIndex >= 0) {
    userListDTO = parsedUsersFromFile.slice(startIndex, startIndex +
    ↪  50);
} else {
    userListDTO = parsedUsersFromFile;
    }    res.status(200).send(userListDTO);
```

Po overení požiadavky a otvorení súboru sa dáta prečítajú a spracujú do objektu do premennej *parsedUsersFromFile*. Ak máme zoznam objektov vieme si zistiť počiatočný index pre výber zo zoznamu. Ak je v zozname menej ako 50 objektov, odošle sa celý zoznam. Pretože nové záznamy sa pridávajú na koniec, tak podľa začiatočného indexu si vyberieme posledných 50 záznamov, ktoré sú zároveň najnovšími. V tele odpovedi posielame administrátorovi dáta, ktoré ho zaujímajú a nemusí ich filtrovať.

6.2 Implementácia komponentov administrátora

V návrhu riešenia sme spomínali osobitný komponent pre každú jednu funkcionalitu administrátora (pozri obrázok 5.1). Ako sa však počas vývoja ukázalo, jednoduchšie bolo zlúčiť niektoré funkcionality do jedného A-frame komponentu. Kontextový diagram 6.1 nám znázorňuje aj závislosti medzi komponentmi administrátora. V tejto sekcii si opíšeme implementáciu

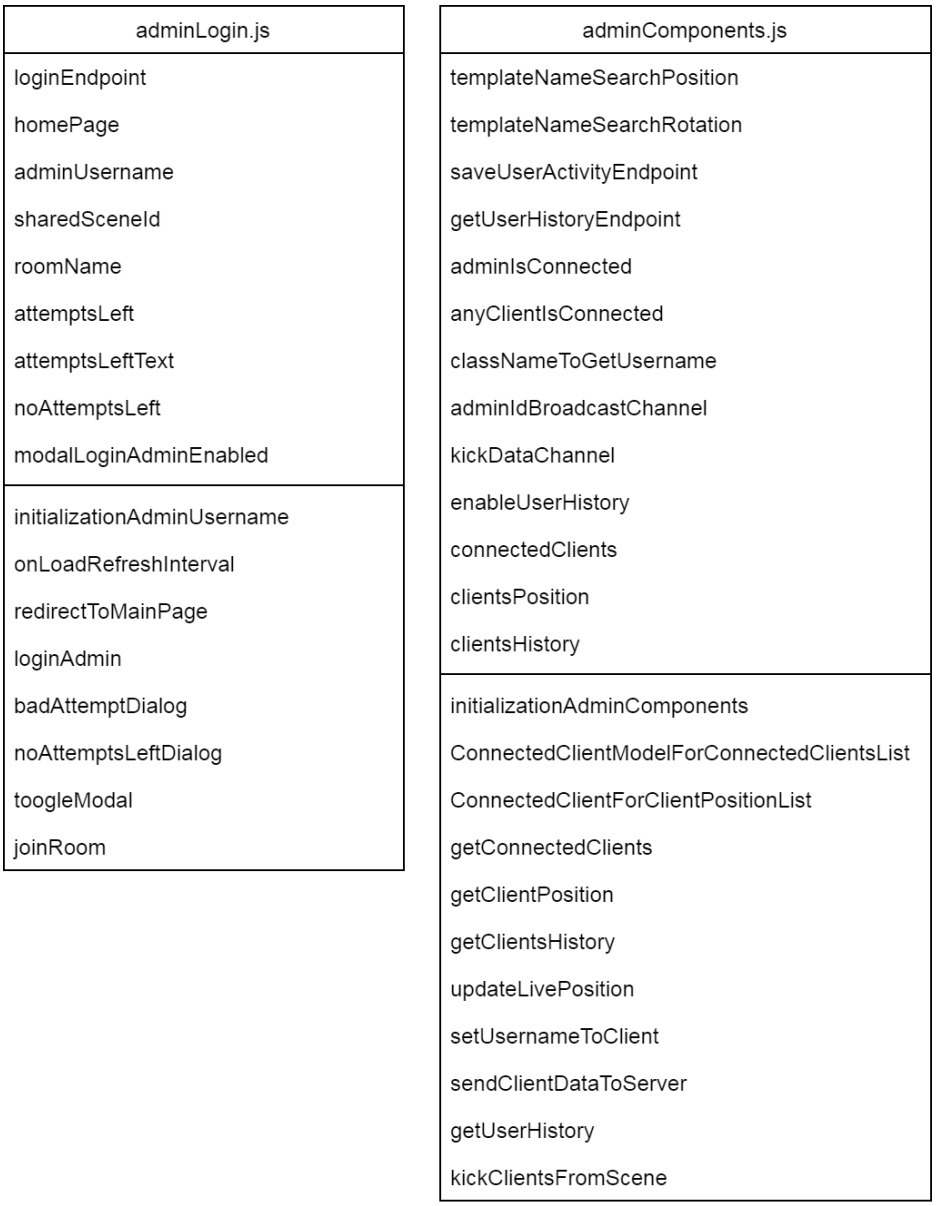
jednotlivých komponentov pre administrátora s využitím funkcií implementovaných v technológii networked-aframe. Tiež sme počas implementácie vytvorili pomocné funkcie a XML HTTP požiadavky, ktorých využitie si opíšeme v rámci jednotlivých komponentov. Všetky skripty administrátora sú implementované v súbore public. Na obrázku 6.3, vidíme štruktúru súboru public v testovacej scéne.



Obr. 6.3: Štruktúra priečinku public.

Priečinok je umiestnený v hlavnom priečinku projektu. V najvyššej úrovni priečinka public, sú umiestnené HTML dokumenty. V HTML dokumentoch sú implementované entity, scéna, používateľské rozhranie a aj výber skriptov, ktoré sa majú na danej stránke použiť. Pre potreby dizajnu tabuliek, modálneho dialógu a celého používateľského rozhrania je vytvorený priečinok *styles*, v ktorom sú umiestnené všetky súbory na úpravu štýlov. V priečinku *js* sú umiestnené skripty, ktoré jednotliví klienti využívajú.

Pre komponent administrátora nás zaujímajú súbory *admin.html*, *adminLogin.js* a *adminComponents.js*. Na obrázku 6.4 môžeme vidieť štruktúru skriptu administrátora pre prihlásenie a skriptu, v ktorom sa nachádzajú komponenty administrátora. V hornej časti sú uvedené premenné a konštanty. V spodnej časti štruktúry skriptu sú vymenované funkcie, ktoré využívame. Využitie funkcií a premenných si bližšie opíšeme v implementácii jednotlivých komponentov. Tak isto aj ich úlohu v konkrétnom komponente.



Obr. 6.4: Štruktúra skriptu adminLogin.js a adminComponents.js.

6.2.1 Komponent pre prihlásenie administrátora

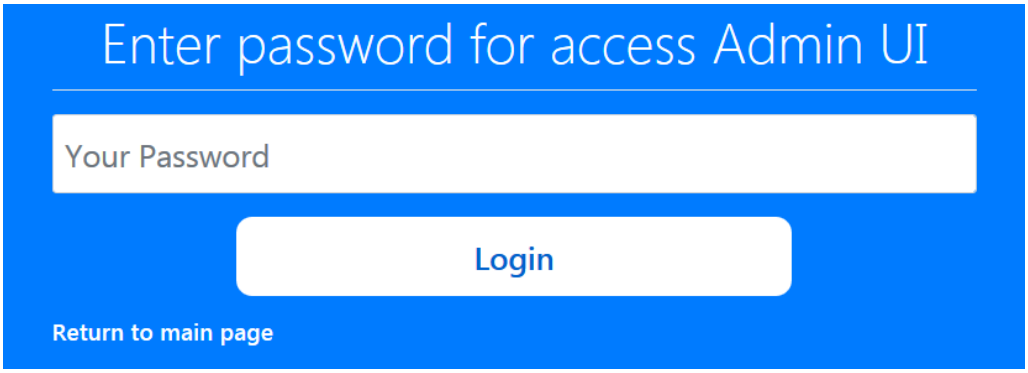
Tento komponent zabezpečuje prihlásenie do zdieľanej scény až po zadaní správneho hesla, ktoré sa overuje na serveri. Administrátora odlíšime aj použitím inej URL, ktorá obsahuje aj inú HTML stránku. Tá využíva iné skripty a komponenty. Pre zabezpečenie, prístupu k rozšíreným funkcionalitám, je potrebné použitie autentifikácie pomocou hesla, ktoré poznajú len vybraní používatelia. Názov skriptu je *adminLogin.js* a presné umiestnenie v projekte si môžeme pozrieť na obrázku 6.3. Obrázok 6.4 zobrazuje jeho štruktúru.

V komponentoch servera sme vytvorili koncový bod, ktorý zadané heslo overí. Adresa koncového bodu je uložená v konštante *loginEndpoint*. Našou úlohou v tomto komponente je poslať zadané heslo koncovému bodu servera a odchytávať odpoveď. Implementácia funkcie overenia hesla a odchytávanie odpovede servera je riešená vo funkcii *loginAdmin*, ktorej vstupným argumentom je práve heslo zadané používateľom. V prípade správnej odpovede pripojíme administrátora do CVE. Pripojenie rieši funkcia *joinRoom*, ktorej vstupným argumentom je názov spoločnej miestnosti, uložený v konštante *roomName*. Ak zadané heslo nie je správne, tak má administrátor možnosť zadať heslo znova. Zároveň sa mu zobrazí hodnota konštanty *attemptsLeftText*, ktorá obsahuje text o zle zadanom hesle a zostávajúci počet pokusov. Hodnota konštanty *attemptsLeft* určuje maximálny počet pokusov opakujúcich sa po sebe pre zadanie hesla.

V ukážke zdrojového kódu nižšie vidíme odchytenie odpovede servera, ktorá nám oznamuje zadanie správneho hesla používateľom. Volaním funkcie *joinRoom* pripojíme administrátora do CVE. Skrytie modálneho dialógu sa realizuje len v prípade kladnej hodnoty premennej *modalLoginAdminEnabled*. Pre použitie prihlásenia bez používateľského rozhrania sú pre informovanie používateľa o priebehu prihlásenia použité logy do konzoly prehliadača.

```
if (responseObj === "OK") {
    if (modalLoginAdminEnabled === true) {
        toggleModal();
    }
    console.info("Password verified successfuly");
    joinRoom(sharedSceneId);}
```

Po zavolaní skriptu HTML stránkou sa hneď zavolá funkcia *initializationAdminUsername*, ktorá inicializuje A-frame komponent, do ktorého sa vloží používateľské meno administrátora z konštanty *adminUsername*. Pre použitie používateľského rozhrania pre prihlásenie sa po načítaní celej stánky a virtuálneho priestoru zavolá funkcia *onLoadRefreshInterval*, ktorá najprv skontroluje hodnotu premennej *modalLoginAdminEnabled*. Ak je hodnota *true*, tak vyhľadá jednotlivé HTML elementy a priradí ich premenným pre používateľské rozhranie. Po priradení všetkých elementov sa pomocou funkcie *toogleModal* zobrazí modálny dialóg pre prihlásenie. Modálny dialóg je zobrazený na obrázku 6.5.



Obr. 6.5: Dizajn modálneho dialógu pre prihlásenie administrátora.

6.2.2 Komponent pre získanie zoznamu pripojených klientov

Komponent pre získanie zoznamu pripojených klientov ako aj všetky komponenty administrátora, okrem komponentu pre prihlásenie, zobrazené v kontextovom diagrame 6.1 sú umiestnené v skripte *adminComponents.js*. Štruktúra skriptu je na obrázku 6.4. Umiestnenie zobrazené v štruktúre priečinku je zobrazené na obrázku 6.3. Po zavolaní skriptu sa zavolá funkcia *initializationAdminComponents*, ktorá vytvorí aj A-frame komponent pre získanie zoznamu pripojených klientov s názvom *admin-connected-clients*. Podľa konceptuálneho návrhu riešenia zobrazeného na obrázku 5.1, má komponent pre získanie zoznamu pripojených klientov riešiť aj odosielanie záznamu o histórii klienta na server. Táto funkcionalita sa počas implementácie presunula do nového – inicializačného komponentu, ktorý si opíšeme neskôr.

Podľa návrhu má komponent získavať zoznam aktuálne prihlásených klientov. Podľa modelu objektu uloženého v zozname prihlásených klientov z obrázku 5.2 je potrebné implementovať spôsob získavania identifikačného čísla klienta, čas pripojenia a zároveň získanie jeho používateľského mena. Objekty klientov s týmito atribútmi si uložíme do zoznamu *connectedClients*.

Komponent zachytáva udalosť pripojenia klienta do CVE (*clientConnected*). Po zachytení tejto udalosti si z nej zistíme identifikačné číslo práve pripojeného klienta. Pomocou technológie networked-aframe zistíme zoznam všetkých pripojených klientov. Ak máme zoznam pripojených klientov vrátený z networked-aframe a identifikačné číslo práve pripojeného klienta, zistíme si čas pripojenia klientov do scény zo zoznamu, ktorý vrátil networked-aframe.

Po zistení času pripojenia klienta si vytvoríme pomocou funkcie *ConnectedClientModelForConnectedClientsList* objekt, ktorý vložíme do zoznamu *connectedClients*. Funkcia *ConnectedClientModelForConnectedClientsList* má dva vstupné parametre – identifikačné číslo klienta a čas pripojenia klienta do CVE. Ak je povolené zobrazenie používateľského rozhrania a tabuľky pre zobrazenie pripojených klientov, tak si po uložení nových údajov aktualizujeme tabuľku s novým zoznamom. V prípade, že toto používateľské rozhranie je zakázané, tak je používateľ informovaný o nových dátach v konzole prehliadača informačnými logmi.

Používateľské meno sa do zoznamu pridáva až po vytvorení zdieľanej entity avatara v CVE. Po zachytení udalosti vytvorenia entity avatara sa zavolá funkcia *setUsernameToClient*. Vstupným parametrom funkcie je identifikačné číslo používateľa, ktorému sa zdieľaná entita avatara vytvorila. Premenná *classNameToGetUsername* označuje entitu, ktorá obsahuje používateľské meno klienta. Vyberieme si všetky entity podľa tejto premennej a následne si nájdeme entitu patriacu klientovi, ktorý nám prišiel ako vstupný parameter funkcie. Po nájdení entity priradíme používateľské meno klienta objektu zo zoznamu pripojených klientov, ktorý reprezentuje daného používateľa. Pre nájdenie správneho objektu v zozname pripojených používateľov, použijeme znovu jeho identifikačné číslo.

6.2.3 Komponent pre posielanie správ klientom

Pre posielanie správ klientom je potrebné mať zoznam pripojených klientov. Preto je v kontextovom diagrame 6.1 znázornená závislosť od komponentu pre získanie zoznamu pripojených klientov. Pre fungovanie posielania správ klientovi, je potrebné implementovať klientovi komponent pre prijímanie správ od administrátora. Tento komponent si priblížime pri komponentoch bežného klienta. Implementácia tohto komponentu sa nachádza v súbore *adminComponents.js* v A-frame komponente *admin-connected-clients*.

V implementácii je potrebné odchytiť udalosť pripojenia klienta do CVE, pre odoslanie overovacieho kľúča všetkým pripojeným klientom a odoslanie správy priamo klientovi, ktorého chceme z CVE odpojiť. Pre odchytávanie udalosti pripojenia klienta využijeme už existujúcu implementáciu z komponentu pre získanie zoznamu pripojených klientov. Po zachytení udalosti pripojenia klienta pošleme pomocou networked-aframe správu (*broadcast*) všetkým pripojeným klientom s tajným kľúčom administrátora. V ukážke zo zdrojového kódu nižšie vidíme poslanie tejto správy cez kanál uložený v konštante *adminIdBroadcastChannel*.

```
NAF.connection.broadcastData(adminIdBroadcastChannel,
↪  NAF.clientId.slice(0, 10));
```

Telo správy tvorí overovací kľúč, ktorý sa skladá z prvých 10 znakov identifikačného čísla administrátora. Odchytávanie správy a ukladanie overovacieho kľúča na strane bežného klienta si opíšeme v sekcii komponent pre prijímanie správ od administrátora.

Pre odpojenie klienta z CVE využijeme funkciu *kickClientsFromScene*, ktorej vstupným argumentom je zoznam identifikačných čísel klientov, ktorých chceme odpojiť z CVE. Identifikačné čísla klientov získame zo zoznamu pripojených klientov. Samotná funkcia *kickClientsFromScene* prechádza zoznamom, ktorý dostala ako vstupný parameter a každému klientovi odošle súkromnú správu. Posielanie správy je zobrazené na ukážke zdrojového kódu.

```
NAF.connection.sendData(user.userId, kickDataChannel, "kick");
```

Pre odoslanie súkromnej správy klientovi sa nám zmenil aj dátový kanál. Pre opustenie CVE používame dátový kanál, ktorého názov je uložený

v konštante *kickDataChannel*. Telo správy je jednoduché slovo. Celá táto komunikácia bola implementovaná tak, aby dodržala postupnosť zo sekvenčného diagramu na obrázku 5.4 z návrhu komponentu pre posielanie súkromnej správy inému klientovi.

6.2.4 Komponent pre získanie pozícii pripojených klientov

Komponent získavania pozícii klientov bol najnáročnejší na implementáciu. Podmienkou z návrhu bolo získavať pozíciu hneď po pohybe iného klienta, nie po nejakom striktne danom intervale. Implementácia tohto komponentu sa nachádza v súbore *adminComponents.js*, presnejšie v implementácii A-frame komponentu s názvom *admin-position-of-clients*. Ako sme už spomínali, A-frame komponenty sa inicializujú po zavolaní skriptu *adminComponents.js* z HTML stránky *admin.html*, kde sa ako jediná priamo volá funkcia *initializationAdminComponents*, ktorá vytvorí všetky A-frame komponenty.

A-frame komponent má metódy životného cyklu z ktorých sme doteraz využívali len metódu *init*, ktorá sa volá pri inicializácii samotného komponentu. Pre získavanie aktuálnych pozícií klientov v CVE, využívame aj metódu životného cyklu *tick*, ktorá sa volá v každej vykresľovacej slučke scény. Pre získavanie pozícií klientov využívame zoznam *clientsPosition*, v ktorom sa nachádzajú objekty podľa modelu pre zoznam na získanie pozícii z obrázku 5.2. Vytváranie objektov do zoznamu rieši funkcia *ConnectedClientForClientPositionList*.

Začneme inicializáciou komponentu v metóde *init*, kde si zaregistrujeme poslucháča na udalosť vytvorenia zdieľanej entity (*entityCreated*) avatara bežného klienta. Po vytvorení avatara bežného klienta si najprv vyberieme entity, ktoré nás zaujímajú podľa konštanty pre získanie pozície avatara klienta *templateNameSearchPosition* a pre rotáciu hlavy avatara klienta. Je to konštanta s názvom *templateNameSearchRotation*. Hodnoty do tejto šablóny zadávame podľa návrhu avatara klienta, tak aby entita označená ako *networked* v HTML súbore zdieľala potrebný atribút pre polohu alebo rotáciu.

Po nájdení správnej entity avatara bežného klienta potrebujeme z entity získať identifikačné číslo klienta, ktorému patrí a identifikačné číslo entity. Identifikačné číslo klienta slúži ako primárny kľúč pre ukladanie a priraďovanie identifikačných čísel entít. Získame ho z vytvorenej entity, presnejšie atribútu

networked, kde hľadáme atribút *owner*. Identifikačné číslo entity pre získanie pozície klienta v priestore ukladáme do atribútu objektu pod názvom *entityId* a pre získanie rotácie hlavy má atribút objektu názov *entityIdRotation*. Identifikačné číslo entity získame tiež z atribútu networked, ale v ňom hľadáme atribút *networkId*. Po priradení atribútu objektu sa objekt pridá do zoznamu. V implementácii je ošetrený aj prípad, ak už existuje v zozname klientov pre pozíciu záznam o klientovi s jeho identifikačným číslom, ale nemá priradenú hodnotu atribútu *entityIdRotation* alebo *entityId*. Vtedy sa už existujúcemu objektu zo zoznamu na získanie pozícii klientov, pridá chýbajúci atribút a príslušná hodnota identifikačného čísla entity. Po pridaní nového objektu do zoznamu sa zavolá funkcia *setUsernameToClient*, ktorá priradí klientovi používateľské meno. Zároveň sa nastaví hodnota premennej *anyClientIsConnected* na *true*, teda kladnú. Implementácia funkcie *setUsernameToClient* je opísaná v sekcii 6.2.2.

Získavanie aktuálnych pozícii klientov je implementované vo funkcii *updateLivePosition*, ktorá sa volá v metóde *tick*. Volanie metódy *tick* sme si vysvetlili na začiatku tohto komponentu. Funkcia *updateLivePosition* sa zavolá vždy, no získavanie pozície klientov prebehne až keď je pripojený aspoň jeden klient – hodnota premennej *anyClientIsConnected* je nastavená na *true*. Vo vnútri podmienky si pomocou networked-aframe a identifikačného čísla entity získame objekt entity s aktuálnou pozíciou a rotáciou v CVE. V ukážke zo zdrojového kódu nižšie vidíme získanie objektu entity pre pozíciu a jej uloženie do lokálnej premennej.

```
let clientEntityPosition =
↪  NAF.entities.getEntity(client.entityId).object3D.position;
client.positionX = clientEntityPosition.x.toFixed(4);
client.positionY = clientEntityPosition.y.toFixed(4);
client.positionZ = clientEntityPosition.z.toFixed(4);
```

Priraďovanie hodnôt prebieha jednotlivo pre každú jednu os a to vybratím atribútu zo získaného objektu. Získaná hodnota ukladaná do zoznamu pozícií sa pre lepšiu čitateľnosť zaokrúhľuje na štyri desatinné miesta. Rovnaký postup je aj pre získavanie rotácie hlavy avatara klienta.

6.2.5 Komponent pre získavanie zoznamu o histórii klientov

Z návrhu vyplýva, že komponent má poskytnúť administrátorovi prehľad o posledných 50 pripojených klientoch v CVE. Implementácia logiky komponentu pre získavanie zoznamu o histórii klientov sa nachádza vo funkcii *getUserHistory*. Komponent pre získavanie zoznamu o histórii klientov má aj samostatný A-frame komponent s názvom *admin-history-of-clients*, v ktorom sa len nastaví hodnota premennej *enableUserHistory* na *true*, ak sa komponent používa v entite administrátora. Vytvorí sa vo funkcii *initializationAdminComponents* v skripte *adminComponents.js*, tak ako aj predchádzajúce komponenty, ktoré pracujú so zoznamami klientov. Volanie funkcie *getUserHistory* rieši inicializačný komponent (pozri obrázok 6.1).

V tele funkcie *getUserHistory* vytvárame XMLHTTP požiadavku typu GET, ktorú posielame na vytvorený koncový bod servera uložený v konštante *getUserHistoryEndpoint*. Do hlavičky požiadavky pridávame vlastný atribút, ktorým dávame serveru potvrdenie, že požiadavka prišla od prihláseného administrátora. Telo požiadavky zostáva prázdne. Pri spracovaní odpovedi servera najprv zistíme jej status. Ak je status 200 – OK, tak do zoznamu o histórii klientov (*clientsHistory*) vložíme telo požiadavky, v ktorom nám prišiel zoznam klientov.

6.2.6 Inicializačný komponent administrátora

Inicializačný komponent vznikol pri implementácii riešenia a nemáme ho pri návrhu riešenia. V kontextovom diagrame 6.1 vidíme, že od inicializačného komponentu sú závislé komponenty: získanie zoznamu pripojených klientov, pre získanie pozícii pripojených klientov a komponent pre získavanie zoznamu o histórii klientov. Všetky komponenty sa dajú použiť nezávisle od seba. Jediná ich závislosť je od inicializačného komponentu. Inicializačný komponent má vlastný A-frame komponent s názvom *admin-initialization*. Komponent sa vytvára vo funkcii *initializationAdminComponents*.

Inicializačný komponent obsahuje dvoch poslucháčov udalostí. Prvým je poslucháč pre udalosť pripojenia klienta administrátora do CVE - *connected*, v ktorom je riešená:

- Inicializácia používateľského rozhrania administrátora.
- Volanie funkcie pre získanie zoznamu o histórii klientov zo servera.

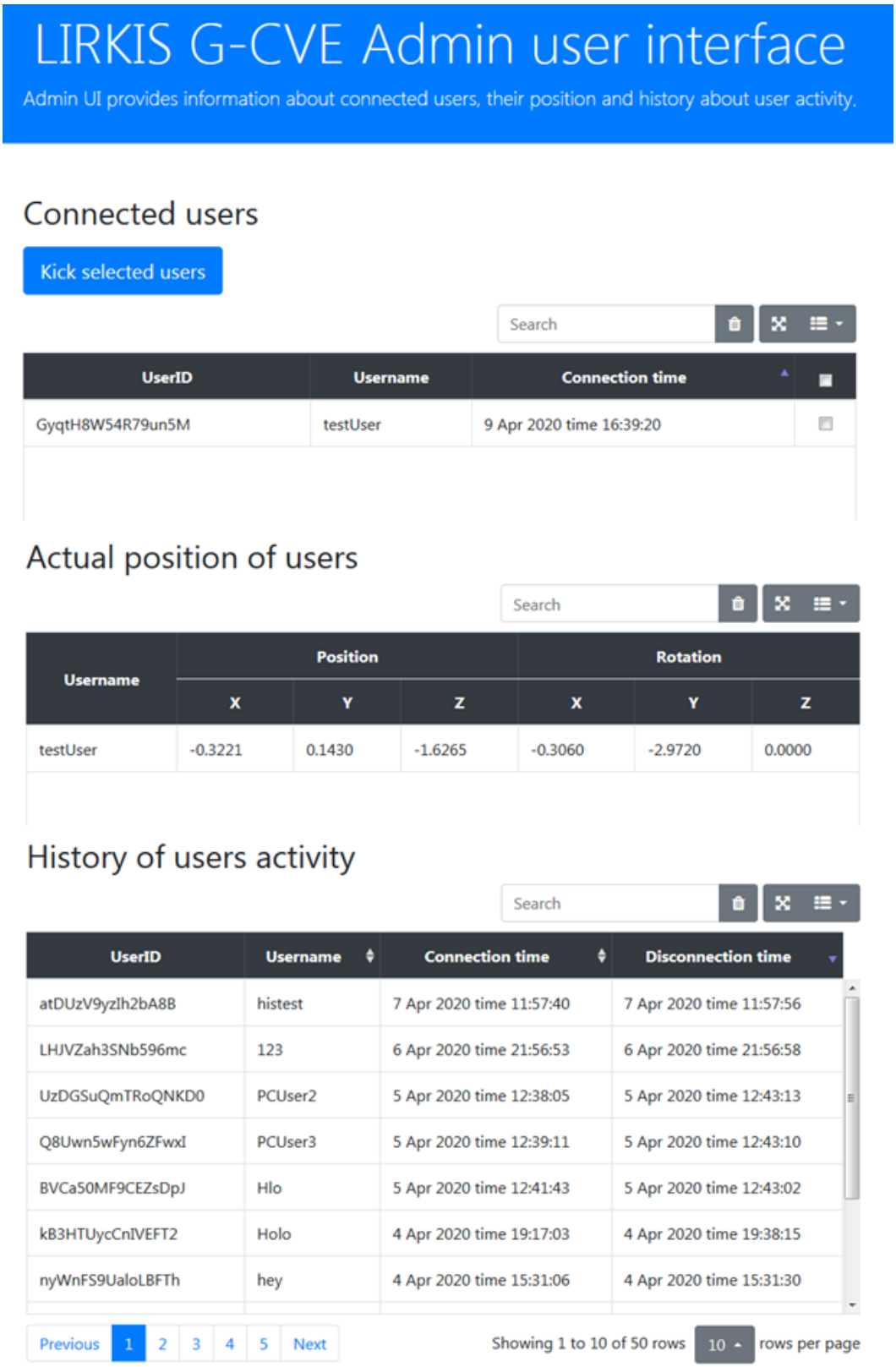
Druhým je poslucháč pre udalosť odpojenia iného klienta z CVE – *clientDisconnected*, v ktorom riešime:

- Odstránenie záznamu klienta po jeho odpojení zo zoznamu pripojených klientov *connectedClients* a zo zoznamu aktuálnych pozícií klientov *clientsPosition*.
- Vytvorenie a odoslanie požiadavky serveru zo záznamom o práve odpojenom klientovi do zoznamu o histórii klientov.

Začneme implementáciou inicializácie používateľského rozhrania administrátora. Používateľské rozhranie môže byť úplne zakázané. Pri tomto prípade používateľ používa konzolu prehliadača pre využívanie aktívnych komponentov. Tabuľky jednotlivých komponentov sú nezávislé na sebe a vieme používať každý jeden komponent aj bez tabuľky. Pre používateľské rozhranie testovacej scény je každá tabuľka povolená.

Dizajn používateľského rozhrania je na obrázku 6.6. Tabuľky sú typu BootstrapTables a poskytujú responzívny dizajn, filtrovanie, hľadanie záznamu alebo skrývanie stĺpcov. Tabuľka *Connected users* vykresľuje zoznam *connectedClients*. Tabuľka *Actual position of users* zobrazuje obsah zoznamu *clientsPosition*.

Volanie funkcie *getUserHistory*, ktorej implementáciu sme si opísali v časti 6.2.5, je podmienené použitím komponentu pre získavanie zoznamu o histórii klientov, teda hodnotu *true* premennej *enableUserHistory*. Na obrázku 6.6 môžeme vidieť tabuľku *History of users activity*, ktorá vykresľuje zoznam *clientsHistory*. Dĺžka zoznamu získaneho zo servera je 50 a tabuľka zobrazuje najviac 10 záznamov na jeden strane.



Obr. 6.6: Dizajn používateľského rozhrania administrátora.

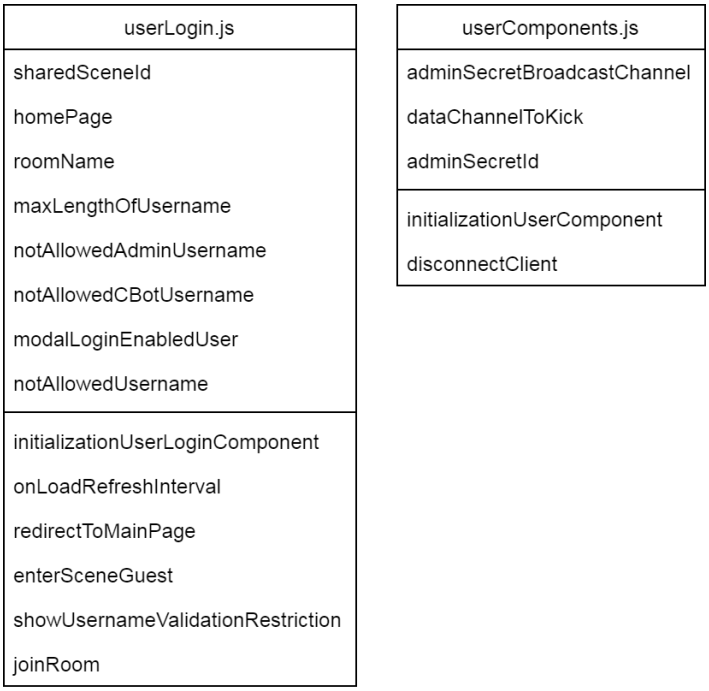
Ostatné funkcionality inicializačného komponentu administrátora boli opísané v návrhu riešenia aj v konceptuálnom návrhu riešenia 5.1 pri komponentoch pre získanie pozícii pripojených klientov a pre získanie zoznamu pripojených klientov. Pre jednoduchosť riešenia a menšiu duplicitu kódu, sme spojili odstránenie odpojeného klienta zo zoznamov do funkcie vykonanej po zachytení udalosti odpojenia klienta z CVE. V nej si zistíme identifikačné číslo práve odpojeného klienta z detailu udalosti. Následne vyhľadáme klienta v oboch zoznamoch *connectedClients* a *clientsPosition*. Po nájdení objektu klienta v zozname pre získavanie aktuálnej pozície (*clientsPosition*), ho odstránime zo zoznamu. Pri zozname pripojených klientov (*connectedClients*) si objekt klienta vyberieme a priradíme mu atribút *roomLeaveTime* získaním aktuálneho času. Ak je povolené zaznamenávanie histórie o klientoch (*enableUserHistory*), tak zavoláme funkciu *sendClientDataToServer*. Ak nie, tak len odstránime objekt zo zoznamu.

Funkcia *sendClientDataToServer* sa stará o vytvorenie a odoslanie požiadavky serveru so záznamom o práve odpojenom klientovi. Vstupným parametrom je práve objekt klienta aj s atribútom času opustenia CVE. Funkcia vytvorí XML HTTP požiadavku typu POST. Pridá tiež hlavičku na overenie, či požiadavka prišla od prihláseného administrátora. Objekt klienta sa odošle v tele požiadavky. Ak bola požiadavka spracovaná serverom správne, tak sa zavolá funkcia *getUserHistory*, pre aktualizovanie zoznamu o histórii klientov.

6.3 Implementácia komponentov bežného klienta

Už z návrhu riešenia a návrhu konceptuálneho modelu 5.1 vidíme, že komponenty bežného klienta budu použité všetkými bežnými klientmi v našej testovacej scéne. Keďže je naše riešenie navrhnuté pre použitie v prostredí, kde sa nachádza čo najviac súčasne pripojených klientov, ale počet požiadaviek na server je obmedzený, preto je aj verifikácia používateľského mena riešená len na úrovni klienta. Tak isto klient neposiela žiadnu požiadavku priamo na server, len sa zaregistruje prostredníctvom networked-aframe. Networked-aframe následne sprostredkuje dátové kanály s ostatnými klientmi a informuje ich o pripojení nového klienta. Prihlásenie klienta je implementované v skripte *userLogin.js*. Vytváranie komponentu bežného klienta je realizované

v skripte *userComponents.js*. Bežný klient využíva súbor *user.html*. Ich umiestnenie je znázornené na obrázku 6.3. Štruktúra jednotlivých skriptov, definované konštanty, premenné a funkcie sú zobrazené na obrázku 6.7.



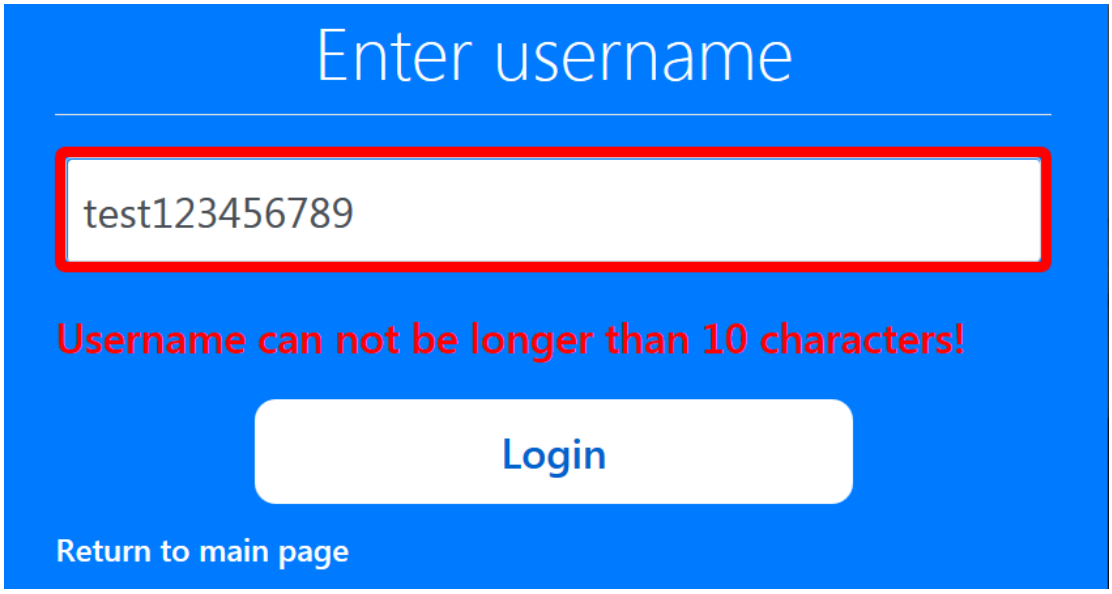
Obr. 6.7: Štruktúra skriptu userLogin.js a userComponents.js.

6.3.1 Komponent pre prihlásenie bežného klienta

Bežný klient nemusí pre vstup do CVE zadávať heslo, tak ako to bolo v prípade administrátora a cBot-a. Komponent pre prihlásenie bežného klienta overuje jeho používateľské meno. Neposiela žiadne požiadavky na server. Celá validácia sa odohráva na strane klienta. Tak ako u administrátora, je k dispozícií dialóg pre prihlásenie a komponent pre prihlásenie sme umiestnili do samostatného skriptu *userLogin.js*. Pri inicializácii prihlásenia sa zavolá funkcia *initializationUserLoginComponent*. Používanie modálneho dialógu na prihlásenie je uložené v konštante *modalLoginEnabledUser*. Inicializácia používateľského rozhrania sa vykoná po načítaní stránky vo funkcii *onLoadRefreshInterval*.

Po zadaní používateľského mena sa zavolá funkcia *enterSceneGuest*, ktorej vstupným parametrom je zadané používateľského mena. V tele funkcie sa overí používateľské meno voči obmedzeniam, ktoré sme uviedli v návrhu riešenia

v časti 5.6.1. Všetky tieto obmedzenia máme uložené v konštantách, aby sa dali ľahko meniť. Napríklad obmedzenie ohľadom dĺžky je uložené v konštante *maxLengthOfUsername*. Pre každé jedno obmedzenie je vytvorená aj chybová hláška, ktorá sa zobrazí v modálnom dialógu. Takýto prípad je zobrazený na obrázku 6.8. V prípade zakázaného používania dialógu, sa správa o porušení obmedzenia používateľského mena zobrazí v konzolových logoch prehliadača.



Obr. 6.8: Dizajn modálneho dialógu pre prihlásenie bežného klienta so zobrazením upozornenia o príliš dlhom používateľskom mene.

Aj bežný klient je pred zadaním validného používateľského mena v lokálnej scéne. O pripojenie do CVE sa stará funkcia *joinRoom*, ktorej vstupným argumentom je konštanta *roomName*, ktorá obsahuje názov zdieľanej miestnosti. Funkcia *joinRoom* je volaná po overení používateľského mena z funkcie *enterSceneGuest*.

6.3.2 Komponent pre prijímanie správ od administrátora

Pre prijatie správy, musí klient odoberať rovnaký dátový kanál, prostredníctvom ktorého administrátor posiela údaje. Vytvorenie komponentu, ktorý odoberá správy je v skripte *userComponents.js*. Názov A-frame komponentu je *subscribe-to-receive-kick-message*. A-frame komponent sa vytára pri inicializácii skriptu zavolaním funkcie *initializationUserComponent*. Pri inicializácii A-frame

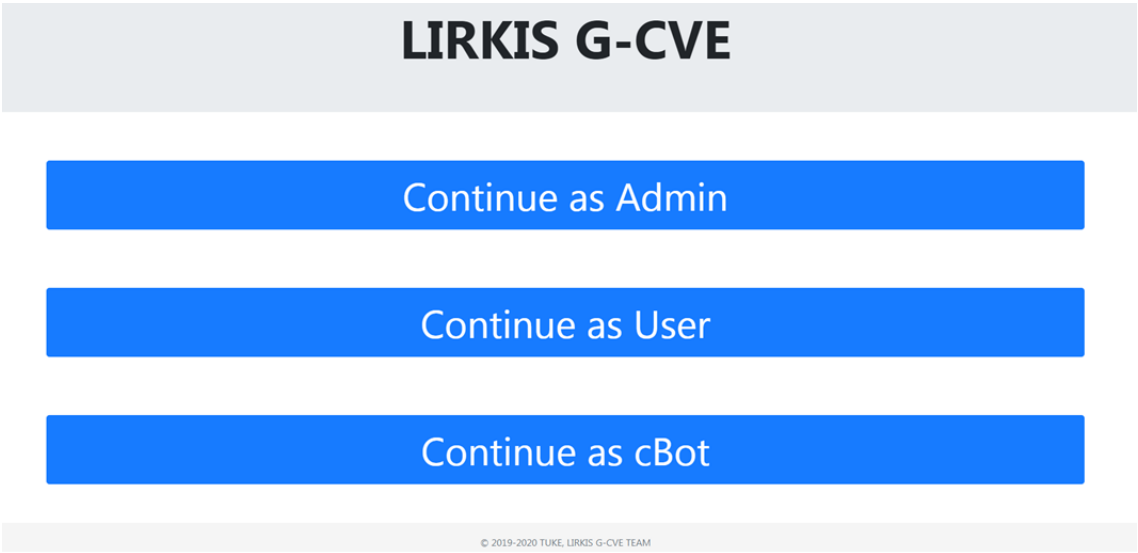
komponentu si vytvoríme poslucháča udalosti pripojenia svojho klienta do CVE. Názov tejto udalosti je *connected*.

Po pripojení klienta do CVE sa pomocou networked-aframe funkcie *subscribeToDataChannel* pripojíme na dátový kanál. Potrebujeme sa pripojiť na dátový kanál, v ktorom nám administrátor pošle overovací kľúč a na dátový kanál, ktorým administrátor odpojí klienta z CVE. Názvy kanálov sú uložené v konštantách *adminSecretBroadcastChannel* pre získanie overovacieho kľúča a *dataChannelToKick* pre prijatie správy na opustenie CVE.

Po prijatí správy v dátovom kanáli pre získanie overovacieho kľúča, si overovací kľúč uloží do premennej *adminSecretId*. Ak klient príjme správu v dátovom kanáli pre opustenie CVE, najprv si overí odosielateľa správy pomocou overovacieho kľúča a následne zavolá funkciu *disconnectClient*, ktorá odpojí klienta z CVE. Klient sa tak dostane do lokálnej scény a oznámenie o opustení CVE je zobrazené v logoch konzoly prehliadača.

6.4 Implementácia úvodnej stránky

V testovacej scéne intersim sme vytvorili troch nových klientov, ktorých prvotne odlišujeme volaním vlastnej HTML stránky. Navigácia medzi rôznymi klientmi prepisovaním URL nie je vôbec intuitívna. V časti 5.1 sme spomínali ďalších riešiteľov problematiky systému LIRKIS G-CVE. Tí sa zameriavali len na implementáciu bežného klienta. V súvislosti s použitím rôznych zariadení, mobilného telefónu a technológie HoloLens, bolo potrebné zmeniť implementáciu entít avatarov. Každý z riešiteľov teda volá inú URL, na ktorej sa nachádza ich implementácia modulu. Pre potrebu integrácie všetkých riešení do spoločného systému, jednoduchú navigáciu a automatické presmerovanie na správnu HTML stránku sme vytvorili úvodnú stránku. Úvodná stránka je implementovaná v HTML súbore s názvom *index.html* a využíva skript *homePage.js* pre riešenie logiky navigácie používateľa na správnu URL. Umiestnenie v štruktúre priečinku *public* je na obrázku 6.3.



Obr. 6.9: Dizajn úvodnej stránky.

V tejto práci bol vytvorený dizajn používateľského rozhrania a navigácia pre vstup CVE ako administrátor, bežný klient alebo cBot. Dizajn úvodnej obrazovky je zobrazený na obrázku 6.9.

6.5 Implementácia komponentov cBot-a

Pre potreby testovacej scény intersim, je potrebné implementovať aj komponenty cBot-a. Vstup do CVE ako cBot je chránený heslom. Využívame rovnaký koncový bod pre prihlásenie aký používa administrátor, (pozri kontextový diagram 6.1). Klient cBot vie získavať pozície všetkých prihlásených klientov v scéne. Využili sme na to upravený komponent administrátora pre získavanie zoznamu aktuálnych pozícií klientov. HTML stránka cBot-a je *cBot.html*, v ktorej voláme skripty *cBotLogin.js* pre prihlásenie a *cBotComponents.js* pre implementáciu komponentu pre získavanie aktuálnych pozícií. Štruktúra skriptov je zobrazená na obrázku 6.10. Umiestnenie v štruktúre priečinkov si môžeme pozrieť na obrázku 6.3.

Obr. 6.10: Štruktúra skriptu cBotLogin.js a cBotComponents.js.

6.5.1 Komponent pre prihlásenie cBota

Komponent prihlásenia cBota má rovnakú štruktúru ako komponent pre prihlásenie administrátora okrem konštanty používateľského mena *cBotUsername*, premennej pre umožnenie prihlásenia pomocou modálneho dialógu *modalLoginEnabledCbot*, inicializačnej funkcie *cBotInitializationLogin* a názvu inicializačnej funkcie pre prihlásenie cBota *loginCBot*. Implementácia, komunikácia so serverom a postup pre prihlásenie je rovnaký ako v prípade prihlásenia administrátora (pozri 6.2.1).

6.5.2 Komponent pre získanie pozícii pripojených klientov cBot

Pre implementáciu využijeme riešenie z komponentu administrátora pre získanie pozícií, (pozri 6.2.4). Z kontextového diagramu vidíme, že komponent pre získanie pozícií nemá závislosť na inom komponente. CBot nepoužíva žiadne používateľské rozhranie, stačí mu len získavať polohy všetkých klientov v CVE. Oproti komponentu použitom v rozhraní administrátora, komponent cBot-a zachytáva aj udalosť odpojenia klienta z CVE, pri ktorej ho odstráni zo zoznamu.

7 Testovanie riešenia

Testovanie softvéru je proces slúžiaci na kontrolu kvality softvérového produktu. Cieľom testovania je overenie požadovanej kvality softvéru z hľadiska funkčnosti, spoľahlivosti, výkonnosti, použiteľnosti a vyhľadanie chýb softvéru. Tento proces je systematický.

Testovanie tejto práce sa realizovalo v dvoch častiach. V prvej časti prebehlo testovanie používateľského rozhrania administrátora na vzorke viacerých používateľov pomocou metódy testovania použiteľnosti. Druhá časť pozostávala zo záťažového testu používateľského rozhrania.

7.1 Testovanie použiteľnosti používateľského rozhrania administrátora

Pre testovanie riešenia využijeme priame testovanie a empirickú metódu testovania použiteľnosti systému (*SUS*). Metóda testovania použiteľnosti poskytuje rýchlu a spoľahlivú klasifikáciu jednoduchosti používania aplikácie. Testovanie sa vykonáva na vzorke používateľov, ktorí na základe vytvorených scenárov použitia, vyplnia dotazník s desiatimi otázkami. Každá otázka má päť možnosti odpovedí. Možnosti sú zoradené od rozhodne súhlasím až po výrazne nesúhlasím. Takéto testovanie je typu *black box* – používateľ na základe scenárov použitia vie, čo má softvér robiť, no nepozná implementáciu.

7.1.1 Návrh priebehu testovania

Účastníci testovania pri testovaní použiteľnosti riešenia postupujú podľa zadaných úloh. Zadané úlohy sú opísané v krokoch, pre každý scenár použitia. Scenár je súhrn všetkých krokov používateľa pre vykonanie zadanej úlohy. Používatelia sú informovaní o spisovaní myšlienok pri testovaní aplikácie. Tieto myšlienky sú užitočné pre zlepšenie UX – používateľského zážitku. Po vykonaní zadaných úloh každý účastník testovania vyplní dotazník použiteľnosti systému (*SUS*). Na konci dotazníka sú polia pre myšlienky na zlepšenie riešenia a tiež informácie o rýchlosti internetového pripojenia klienta a veľkosti RAM pamäte počítača, na ktorom sa testovanie realizovalo.

7.1.2 Scenáre použitia

Dokument scenárov použitia obsahuje informácie pre priebeh testovania, URL adresu prostredia *intersim-gcve* a informáciu o používaní internetového prehliadača Mozila Firefox, v ktorom je potrebné vykonať testovanie používateľského rozhrania. Toto obmedzenie je spôsobené využívaním technológie A-frame verzie 0.9.0. Úvodnou stránkou pre začiatok každého testovacieho scenára je hlavná stránka *index.html*.

Scenár číslo jedna je zameraný na prácu s tabuľkou so zoznamom pripojených klientov a odpojenie klienta z CVE. V prvom scenári si tiež používateľ pozrie záznam o klietovi, ktorého odpojil z CVE. Scenárom číslo dva si overíme prácu s tabuľkou s aktuálnymi pozíciami klientov. V scenári číslo tri si používateľ vyskúša vyhľadanie záznamu v tabuľke s históriou klientov.

Scenár číslo 1: Prihlásenie sa do administrátorského rozhrania a vyhodenie bežného klienta zo zdieľaného prostredia.

S1.1 Vybrať možnosť prihlásenia sa ako administrátor.

S1.2 Prihlásiť sa s pomocou hesla *welcome*.

S1.3 V novom okne prehliadača otvoriť hlavnú stránku projektu.

S1.4 Vybrať možnosť prihlásenia sa ako bežný klient.

S1.5 Prihlásiť sa s používateľským menom *kick-* a 3 písmena svojho mena.

S1.6 V okne s prihláseným administrátorom, odpojiť bežného klienta zo zdieľaného prostredia.

S1.7 Skontrolovať v tabuľke pre zobrazenie históriii klientov, záznam vyhodeného klienta.

Scenár číslo 2: Zobraziť si iba pozíciu v osi X a Z. Posunúť sa s bežným klientom v scéne, v osi Z na hodnotu 8.

S2.1 V novom okne prehliadača otvoriť hlavnú stránku projektu.

S2.2 Vybrať možnosť prihlásenia sa ako bežný klient.

S2.3 Prihlásiť sa s používateľským menom *pos-* a 3 písmena svojho mena.

S2.4 V tabuľke s aktuálnou pozíciou klientov, vybrať zobrazenie len X a Z osi pre pozíciu klientov.

S2.5 Posunúť sa s bežným klientom v scéne na pozíciu v osi Z na hodnotu 8.

S2.6 Zatvoriť okno bežného klienta.

S2.7 Skontrolovať v tabuľke pre zobrazenie histórie klientov záznam o práve odpojenom klientovi.

Scenár číslo 3: Vyhľadať záznam v tabuľke so zoznamom o histórii klientov.

S3.1 Vyhľadať v tabuľke pre zobrazenie histórie klientov záznam o klientovi s používateľským menom: *testSearch*.

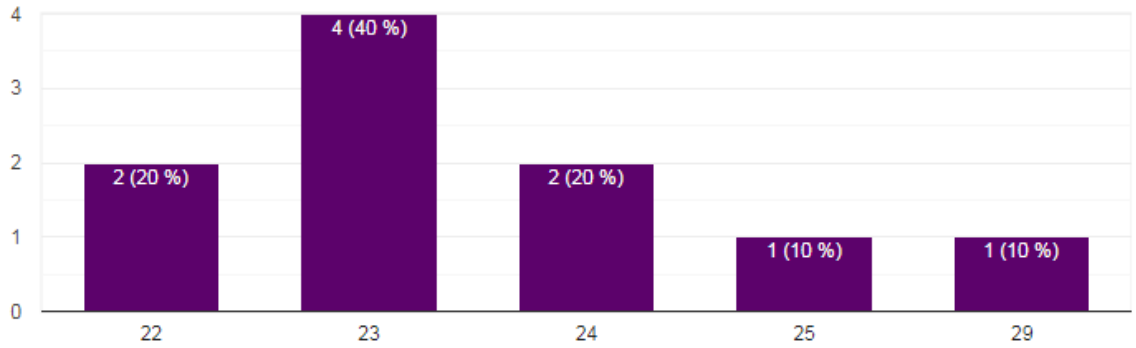
S3.2 Zatvoriť okno administrátora.

7.1.3 Opis priebehu testovania

Testovanie prebiehalo u každého účastníka testovania samostatne na vlastnom počítači. Každý účastník dostal dokument s úvodom, ktorý obsahoval usmernenie o použití iba podporovaného prehliadača a scenáre použitia. Po ukončení všetkých scenárov, každý účastník testovania vyplnil dotazník *SUS*, v ktorom boli aj otázky o priebehu testovania a o úspešnom vykonaní jednotlivých scenárov.

7.1.4 Účastníci testovania

Testovania sa zapojilo 10 používateľov. Všetci účastníci testovania sú študentmi vysokej školy a zároveň starší ako 20 a mladší ako 30 rokov. Na obrázku 7.1 vidíme vek jednotlivých účastníkov testovania a ich vekový priemer je 23.8 roku.



Obr. 7.1: Vek účastníkov testovania.

V dotazníku boli aj otázky o samotných účastníkoch testovania. Podľa zozbieraných odpovedí 6 účastníkov už malo skúsenosť s VR a 4 boli úplní začiatočníci. Len 2 účastníci testovania mali rýchlosť internetového pripojenia menej ako 10 Mbps a žiaden účastník testovania nemal na počítači, na ktorom vykonával testovanie kapacitu RAM menšiu ako 8 Gb.

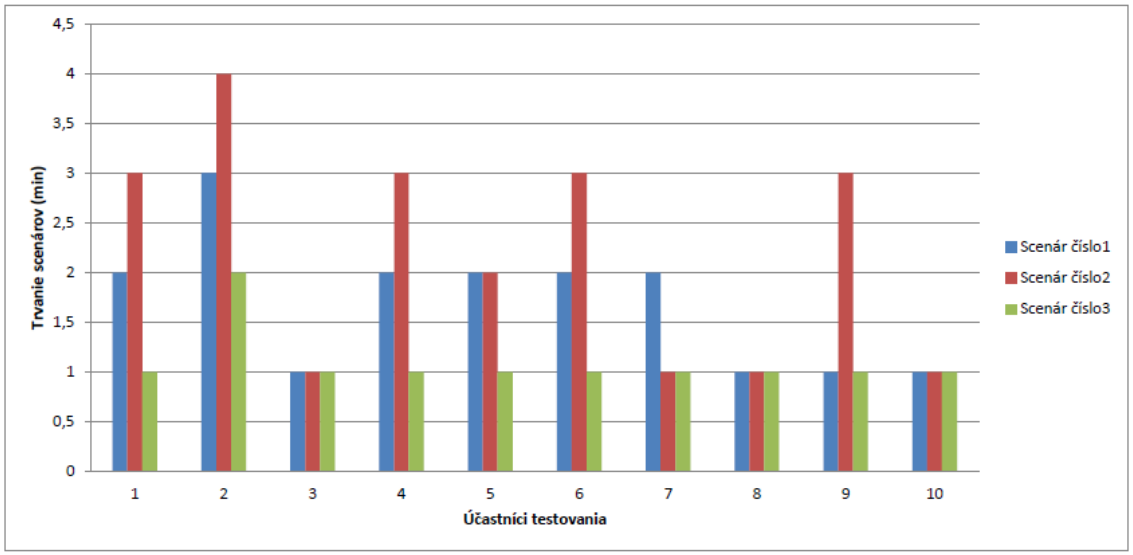
7.1.5 Vyhodnotenie dotazníka testovania použiteľnosti

Každý účastník testovania odpovedal v dotazníku na otázky hodnotami 1–5. Podľa *SUS* hodnota 1 znamená „jednoznačne nesúhlasím“ a hodnota 5 znamená „jednoznačne súhlasím“. Vyhodnotenie otázok dotazníka *SUS*, je na obrázku 7.2. Priemerná hodnota odpovedí *SUS* dotazníka bola 81,5 bodu, čo je v rámci škály hodnotenia *SUS* nadpriemerné skóre.

Otázky/Používateľ	1	2	3	4	5	6	7	8	9	10
1. Myslím si, že by som rád používal túto aplikáciu často.	3	4	4	4	2	3	3	5	4	2
2. Táto aplikácia je pre mňa zbytočne zložitá.	2	2	1	1	1	2	1	1	2	2
3. Aplikácia sa mi zdala ľahko použiteľná.	3	5	5	4	5	5	4	5	4	3
4. Myslím si, že by som potreboval technickú podporu aby som vedel používať túto aplikáciu.	3	2	1	1	1	3	4	1	1	1
5. Jednotlivé funkcie v aplikácii sa mi zdali byť dobre integrované.	4	5	5	5	5	4	5	4	4	4
6. Zdalo sa mi, že v aplikácii je príliš veľa nezrovnalostí.	4	1	1	2	2	3	1	2	2	2
7. Myslím si, že väčšina ľudí by sa aplikáciu naučila veľmi rýchlo používať.	3	5	5	5	4	4	4	5	5	5
8. Aplikácia mi príde byť ťažkopádna na použitie.	1	2	1	1	1	2	1	2	1	2
9. Pri používaní aplikácie sa cítim veľmi sebavedomo.	3	5	4	4	4	5	4	5	5	4
10. Než som mohol začať pracovať s aplikáciou, musel som sa naučiť veľa vecí.	2	1	1	1	1	3	4	1	2	1
81,5	60	90	95	90	85	70	73	93	85	75

Obr. 7.2: Vyhodnotenie otázok dotazníka *SUS*.

Na základe odpovedí o rýchlosti vykonania jednotlivých scenárov použitia, máme k dispozícii graf zobrazený na obrázku 7.3. Ten zobrazuje rýchlosť vykonania scenárov konkrétnym účastníkom testovania v minútach. Priemerný čas testovania bol 5 minút. Vykonanie scenára číslo 1 trvalo v priemere 1 minútu a 42 sekúnd. Najdlhší priemerný čas vykonania bol zaznamenaný pre scenár číslo 2 a to 2 minúty a 12 sekúnd. Účastníci testovania zvládli najrýchlejšie vykonať scenár číslo 3, ktorého priemerný čas vykonania je 1 minúta a 6 sekúnd.



Obr. 7.3: Graf rýchlosti vykonania jednotlivých scenárov použitia.

7.2 Celkové vyhodnotenie testovania použiteľnosti riešenia

Podľa výsledkov dotazníka *SUS* môžeme hodnotiť naše riešenie za intuitívne, ľahké na používanie a prehľadné. V dotazníku boli aj otázky ohľadom dizajnu hlavnej stránky, modálnych dialógov a používateľského rozhrania administrátora. Všetci účastníci testovania odpovedali, že dizajn celého riešenia bol prehľadný a páčil sa im. Riešenie preto môžeme hodnotiť ako *user friendly*, teda používateľsky prívetivé.

Pri riešení zadaných úloh účastníci testovania nepotrebovali žiadnu pomoc. Celé testovanie prebehlo rýchlo a bez problémov, čo dokazuje aj graf 7.3 o rýchlosti vykonania jednotlivých scenárov použitia.

Jediná vec, ktorá bola požadovaná viacerými účastníkmi bola podpora prehliadača Google Chrome. Podporu tohto prehliadača by mal zabezpečiť prechod na novšiu verziu technológie A-frame. Predpokladáme, že tento prechod bude realizovaný v blízkej budúcnosti.

7.3 Záťažové testovanie riešenia

Pod záťažovým testovaním riešenia, myslíme použiteľnosť a rýchlosť odozvy používateľského rozhrania administrátora, pri počte súčasne pripojených klientov väčším ako 10. Zamerali sme sa na rýchlosť vytvorenia entity a priradenia používateľského mena klienta do zoznamu o pripojených klientov, po pripojení do CVE. Ďalej sme si overili rýchlosť vykonania akcie komponentu pre posielanie správ klientovi v CVE, pri väčšom počte pripojených klientov. Posledným bodom bola doba odozvy tabuľky s aktuálnymi pozíciami klientov pri počte klientov väčším ako 15. Počas celého záťažového testovania bol administrátor prihlásený na samostatnom počítači. Bežní klienti sa prihlasovali z iných počítačov a aj z iných sietí. Rýchlosť pripojenia administrátora bola na úrovni: 94.74 Mbps sťahovanie a 9.25 Mbps nahrávanie.

7.3.1 Rýchlosť vytvorenia entity bežného klienta a priradenie používateľského mena objektu v zozname

Toto testovanie je zamerané na rýchlosť vytvorenia zdieľanej entity bežného klienta a priradenia používateľského mena správnemu objektu v zozname pripojených klientov administrátora. Testovanie prebehlo na prihlásenom klientovi administrátora. Meral sa čas od udalosti pripojenia klienta *clientConnected*, medzičas bol vytvorenie entity *entityCreated* a posledný zaznamenaný čas bol priradenie používateľského mena objektu v zozname. Ukladali sme výsledný čas od pripojenia klienta, po priradenie používateľského mena do zoznamu pripojených klientov.

Pred samotným testovaním sme overili priemernú dobu pri prázdnom CVE, v ktorom sa nachádzal len administrátor a do CVE sa pripájal jeden bežný klient. Pri 30 pokusoch nám vyšla priemerná doba od prihlásenia bežného klienta po priradenie používateľského mena 0,338 sekundy. Pričom od prihlásenia po vytvorenie entity bola priemerná doba 0,219 sekundy. Priradenie používateľského mena po vytvorení entity trvalo priemerne 0,019 sekundy.

Záťažové testovanie prebiehalo postupným pripájaním bežných klientov do CVE. Naraz sa nám podarilo pripojiť 15 klientov. Priemerný čas od prihlásenia klienta po priradenie používateľského mena bol 0,574 sekundy. Priemerný čas od vytvorenia entity po priradenie používateľského mena bol 0,023 sekundy. Počas tohto testovania sme narazili na limity servera, ktorý je poskytovaný zdarma na stránke *glitch.com*. Po načítaní poslednej entity bežného klienta do CVE, celý server prestal reagovať a po 30 sekundách odpojil všetkých pripojených klientov.

7.3.2 Testovanie rýchlosti komponentu pre posielanie správ klientovi

Komponent pošle súkromnú správu klientovi, ktorého chceme odpojiť z prostredia. Už v predchádzajúcom teste sme objavili limity servera. Preto testovanie prebiehalo v CVE s 15 pripojenými klientmi. Cieľom bolo odpojiť klientov po jednom pomocou tabuľky so zoznamom pripojených klientov a tlačidla *kick selected users*. Podmienky testovania sú uvedené na začiatku sekcie.

Ako prvé sme namerali hodnoty v prostredí s jedným pripojeným klientom, ktorého sme odpojili z CVE. Zaznamenávali sme čas od stlačenia tlačidla po vymazanie klienta zo zoznamu pripojených klientov. Výsledky boli konzistentné a priemerná doba odpojenia klienta bola 0,624 sekundy.

Pri CVE s 15 pripojenými klientmi boli výsledky horšie. Čas odpojenia prvého klienta zo scény bol 9,578 sekundy. Čas odpojenia druhého 8,833 sekundy a čas odpojenia tretieho klienta bol 7,862 sekundy. V scéne zostalo 12 pripojených klientov a čas odpojenia ďalšieho klienta bol 0,688 sekundy, teda tesne nad úrovňou priemerného času z prázdneho CVE. Priemerný čas bol 2,16 sekundy. Tento čas výrazne ovplyvnili prvé tri merania, ktorých vysoké hodnoty sú spôsobené nedostatočným výkonom servera.

7.3.3 Doba odozvy tabuľky s aktuálnymi pozíciami klientov

Pozície klientov sa získavajú každú snímku obrazovky (FPS). Táto funkcia sa tak môže vykonať aj 60 krát za sekundu. Preto je potrebné overiť jej použiteľnosť v CVE s väčším počtom klientov. Tabuľka zobrazí nové dáta až po zaznamenaní novej pozície alebo rotácie ľubovoľného klienta. Tento test je zameraný na rýchlosť odozvy používateľského rozhrania aj scény administrátora pri väčšom počte pripojených klientov. Snahou je zistiť limity takto navrhnutého systému, či dokáže vykresľovať tabuľku aktuálnymi pozíciami bez výraznejšieho oneskorenia a zároveň zvládne vykresľovať CVE administrátorovi. Nevieme zmerať presný čas, kedy presne pripojený klient stlačil klávesu pre pohyb a kedy to zaznamená klient administrátora na rôznych počítačoch.

Vieme však zmerať dobu odozvy pohybu klienta na jednom počítači. V jednom okne prehliadača je prihlásený administrátor a v druhom okne prehliadača je prihlásený bežný klient. Na strane bežného klienta je zaznamenávaný čas stlačenia klávesy pre pohyb. Administrátor zaznamenáva čas, kedy bola zmenená hodnoty na osi Z. Pri 30 vykonaných meraniach nám vyšla priemerná hodnota zaznamenania pohybu na strane administrátora od stlačenia klávesy 0,340 sekundy.

Keďže sme už objavili limity servera, do CVE sme pripájali klientov po jednom v 30 sekundových intervaloch. Pri 10 pripojených a pohybujúcich sa klientoch fungovalo všetko v poriadku. Ďalší klient sa pripojil a spôsobil oneskorenie aktualizácii údajov v tabuľke a pri snahe pripojiť 12. klienta, server prestal reagovať a odpojil všetkých klientov.

Tento test odhalil aj vizuálnu chybu v používateľskom rozhraní administrátora, presnejšie v tabuľke s aktuálnymi pozíciami klientov. V tabuľke pre zobrazenie aktuálnej polohy klientov bolo viditeľných iba prvých 6 pripojených klientov. Nájdená chyba v riešení bola odstránená.

7.4 Vyhodnotenie záťažové testovania riešenia

Záťažové testovanie celého riešenia dopadlo výborne. Systém ako celok fungoval správne a všetky komponenty fungovali výborne v rámci možností servera. Server je poskytovaný stránkou glitch.com a verzia zdarma ponúka 512MB RAM pamäti a neznámy procesor. Keďže administrátor neustále komunikuje so serverom prostredníctvom technológie networked-aframe, je potrebný oveľa väčší výkon servera. Rozhranie administrátora fungovalo správne, pokiaľ sme nenarazili na spomínané limity a to 10 pripojených a pohybujúcich sa klientov a jeden administrátor.

Z testovania vytvorenia entity bežného klienta a priradenie používateľského mena objektu v zozname vyplýva, že funkcia pre priradenie používateľského mena je použiteľná v riešení a nijako neobmedzuje výkonnosť riešenia. Funkcia pre priradenie používateľského mena čaká na vytvorenie zdieľanej entity bežného klienta, preto je záznam v tabuľke o prihlásenom klientovi bez používateľského mena. Po vytvorení entity bežného klienta je priradenie používateľského mena klientovi do 0,03 sekundy. V tabuľke 7.1, vidíme priemerné časy jednotlivých akcií pri testovaní. V ľavom stĺpci je popis akcie. V strednom stĺpci sú hodnoty pre jedného pripojeného klienta. V stĺpci napravo sú hodnoty zaznamenané z prostredia, kde bolo pripojených súčasne 15 klientov.

Popis testovanej akcie	Jeden klient	15 klientov
Vytvorenie entity	0,219 sekundy	0,374 sekundy
Najpomalšie vytvorenie	0,28 sekundy	0,402 sekundy
Najrýchlejšie vytvorenie	0,189 sekundy	0,304 sekundy
Priradenia používateľského mena	0,019 sekundy	0,023 sekundy
Najpomalšie priradenie	0,026 sekundy	0,030 sekundy
Najrýchlejšie priradenie	0,015 sekundy	0,017 sekundy
Priemerný celkový čas	**0,338 sekundy**	**0,574 sekundy**
Najpomalší celkový čas	0,616 sekundy	0,789 sekundy
Najrýchlejší celkový čas	0,266 sekundy	0,270 sekundy

Tabuľka 7.1: Vyhodnotenie testovania rýchlosti od pripojenia klienta po priradenie používateľského mena klientovi v zozname pripojených klientov.

Komponent pre posielanie správ klientovi funguje v CVE spoľahlivo, bez väčšieho oneskorenia pre scénu s 12 pripojenými klientmi. Ak je počet pripojených klientov väčší, komponent funguje ďalej, ale s oneskorením pokiaľ server neodpojí všetkých klientov pre nedostatočný výkon. V tabuľke 7.2 môžeme vidieť výsledky aj s nameranými minimami a maximami pre odpojenie klienta z prostredia.

Odpojenie klienta z CVE	Jeden klient	15 klientov
Najpomalší celkový čas	0,821 sekundy	9,578 sekundy
Najrýchlejší celkový čas	0,598 sekundy	0,612 sekundy
Priemerný celkový čas	0,624 sekundy	2,161 sekundy

Tabuľka 7.2: Vyhodnotenie testovania rýchlosti komponentu pre posielanie správ klientovi.

Testovanie tabuľky pre zobrazenie aktuálnych pozícií klientov odhalilo limity servera pre maximálne 10 pripojených klientov, ktorí sa pohybujú a vizuálnu chybu, ktorá bola odstránená. Pred dosiahnutím limitov servera, tabuľka bez oneskorenia zobrazovala údaje s aktuálnymi polohami pripojených klientov.

8 Vyhodnotenie riešenia

V analýze riešenia sme si prestavili existujúce riešenia, na základe ktorých sme zhodnotili, že má význam mať v CVE klienta s rozšírenými právomocami. Preto sme podľa vzoru jednotlivých riešení vytvorili klienta – administrátora, ktorý zaznamenáva údaje o pripojených klientoch. Systém Teachyverse náš inšpiroval k vytvoreniu možnosti obmedzenia nechceného správania klientov v CVE. Preto sme navrhli a implementovali komponent pre posielanie správ, pomocou ktorého vieme odpojiť bežného klienta z CVE.

V riešeniach spomínaných v analýze, majú administrátori možnosť priradiť klientov na projekt alebo vytvárať prostredia, do ktorých sa klienti pripoja. Naše riešenie administrátora zaznamenáva a získava aktuálne údaje o pripojených klientoch do zoznamu pripojených klientov a ich aktuálnu polohu v CVE. Tiež ukladáme záznamy o klientoch s časmi ich pripojenia a odpojenia na server do CVE, takže máme prehľad aj o využívaní CVE.

Riešenie je ľahko použiteľné v ľubovoľnej A-frame scéne, ktorá má vytvoreného aspoň jedného klienta. Postup integrácie riešenia do ľubovoľnej scény je uvedený v systémovej príručke, ktorá je prílohou tejto záverečnej práce. V systémovej príručke je vypracovaná aj dokumentácia k riešeniu. Každý komponent riešenia sa dá použiť nezávisle na inom komponente. Jediný komponent, ktorý má závislosť je komponent pre posielanie správ. A to od komponentu bežného klienta pre odoberanie správ administrátora.

Pri testovaním práce sme overili použiteľnosť implementovaného používateľského rozhrania. Používateľské rozhranie ako aj celé riešenie zvládlo aj záťažové skúšky pri pripojení viacerých klientov v CVE, nie je závislé na použití používateľského rozhrania a je navrhnuté a implementované tak, že vieme jednotlivé komponenty používateľského rozhrania povoliť alebo zakázať a to aj na strane bežného klienta.

Zhrnutie všetkého, čo sa nám podarilo implementovať v rámci modulu komunikačnej a riadiacej časti systému LIRKIS G-CVE:

- Odlíšenie bežného klienta a klienta s rozšírenými právomocami – v našom riešení sa tento klient nazýva administrátor.
- Dizajn a implementácia úvodnej stránky riešenia.
- Zabezpečenie vstupu do CVE pre administrátora pomocou hesla overovaného na strane servera.
- Overenie používateľského mena bežného klienta voči obmedzeniam, ktoré boli špecifikované, ešte pred vstupom do CVE.
- Vytvorenie komponentu pre získavanie zoznamu pripojených klientov s časom pripojenia do CVE a používateľským menom.
- Vytvorenie komponentu pre získavanie aktuálnej pozície a rotácie hlavy pripojených klientov v CVE.
- Vytvorenie komponentu pre ukladanie a zobrazenie histórie o aktivite klientov v CVE.
- Dizajn a implementácia dialógov pre prihlásenie administrátora a pre prihlásenie bežného klienta.
- Dizajn a implementácia používateľského rozhrania administrátora.
- Používateľské testovanie použiteľnosti a záťažové testovanie riešenia.

9 Záver

Hlavným cieľom tejto práce bolo vytvoriť modul do systému LIRKIS G-CVE, ktorý umožní získavať aktuálne údaje o pripojených klientoch v CVE ako ich pozíciu, rotáciu hlavy a zoznam práve pripojených klientov.

V analytickej časti tejto práce sme sa zamerali na existujúce systémy, ktoré poskytujú CVE. Všetky riešenia, ktoré poskytovali rozšírené právomoci, mali pre ich používanie vytvoreného samostatného klienta. Daný klient mal rozšírene právomoci v rámci projektu ako prideľovanie ľudí na projekt, vytváranie nových projektov a úprava existujúcich. Výstupom analýzy bolo vytvorenie samostatného klienta pre využívanie osobitných funkcionalít. Tohto klienta sme si nazvali administrátor. V analýze použitých technológii v systéme LIRKIS G-CVE sme si vysvetlili ich fungovanie a správnosť použitia pre systém, ktorý je určený pre viacero naraz pripojených klientov.

Navrhli a implementovali sme funkcionality administrátora pre získavanie zoznamu aktálne pripojených klientov a ukladanie záznamu o dĺžke pobytu bežného klienta v CVE. Podľa vzoru systému Teachyverse, sme administrátorovi a bežnému klientovi vytvorili komponent pre komunikáciu, ktorým vie administrátor odpojiť klienta z CVE. Výnimočnosť tejto práce je, že administrátor získava aktuálne informácie o polohe ostatných klientov v scéne. Pre zobrazenie všetkých získaných dát sme navrhli a implementovali používateľské rozhranie administrátora so zabezpečením prístupu k týmto údajom – autentifikáciou. Tá je riešená formou prihlásenia sa pomocou modálneho dialógu s overením hesla na strane servera. Bežný klient musí pre prihlásenie zadať svoje používateľské meno, ktoré má svoje obmedzenia a validácia prebieha len na strane klienta.

Procesom testovania riešenia sme overili použiteľnosť používateľského rozhrania a schopnosť fungovania aj pri väčšom počte klientov. Administrátor má prehľad o pripojených klientoch, môže ich odpojiť z prostredia. Tiež má k dispozícii ich aktuálnu polohu v prostredí a môže si v histórii klientov pozrieť posledných 50 pripojených klientov s časom ich pripojenia a odpojenia z CVE.

Výstupom tejto práce je modul, ktorý obsahuje komponenty pre získavanie údajov o pripojených klientoch, ukladanie záznamov o klientoch a možnosť odpojiť klientov z prostredia.

Naše riešenie, či už ako celok alebo aj samostatné komponenty, vieme integrovať pre ľubovoľnú scénu. Môžeme napríklad pridaním hlasovej konverzácie vytvoriť prostredie pre komunikáciu, kde by administrátor vedel klienta s nevhodným správaním odpojiť z prostredia. Tiež vieme zo získaných údajov o polohe klientov integrovať klienta s autonómnym pohybov, ktorý si vie dynamický vypočítať cestu k cieľu.

Literatúra

[1] Johnny Hartz Søraker. „Virtual entities, environments, worlds and reality: Suggested definitions and taxonomy“. In: *Trust and virtual worlds: Contemporary perspectives* 63 (2011), s. 44–72.

[2] Jonathan Steuer. „Defining virtual reality: Dimensions determining telepresence“. In: *Journal of communication* 42.4 (1992), s. 73–93. DOI: 10.1111/j.1460-2466.1992.tb00812.x. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1460-2466.1992.tb00812.x. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1460-2466.1992.tb00812.x.

[3] Steve Benford et al. „Collaborative virtual environments“. In: *Communications of the ACM* 44.7 (2001), s. 79–85. URL: https://dl.acm.org/doi/fullHtml/10.1145/379300.379322.

[4] Elizabeth F Churchill, David N Snowdon a Alan J Munro. *Collaborative virtual environments: digital places and spaces for interaction*. Springer Science & Business Media, 2012.

[5] MIT. *A-Frame VR env*. 2015. URL: aframe.io (cit. 30. 01. 2020).

[6] Hayden Lee. *Networked-Aframe*. 2017. URL: https://github.com/networked-aframe/networked-aframe (cit. 30. 01. 2020).

[7] Karola Marky et al. „Teachyverse: Collaborative E-Learning in Virtual Reality Lecture Halls“. In: *Proceedings of Mensch Und Computer 2019*. MuC'19. Hamburg, Germany: Association for Computing Machinery, 2019, s. 831–834. ISBN: 9781450371988. DOI: 10.1145/3340764.3344917. URL: https://doi.org/10.1145/3340764.3344917.

[8] Laura Freina a Michela Ott. „A literature review on immersive virtual reality in education: state of the art and perspectives“. In: *The International Scientific Conference eLearning and Software for Education*. Zv. 1. 133. 2015, s. 10–1007.

[9] Markus Funk et al. „Assessing the Accuracy of Point & Teleport Locomotion with Orientation Indication for Virtual Reality Using Curved Trajectories“. In: *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. CHI '19. Glasgow, Scotland Uk: Association for Computing Machinery, 2019. ISBN: 9781450359702. DOI: 10.1145/3290605.3300377. URL: https://doi.org/10.1145/3290605.3300377.

[10] A. Scavarelli, A. Arya a R. J. Teather. „Circles: exploring multi-platform accessible, socially scalable VR in the classroom“. In: *2019 IEEE Games, Entertainment, Media Conference (GEM)*. IEEE. Jún 2019, s. 1–4. DOI: 10.1109/GEM.2019.8897532.

[11] John Dewey. „Experience and education“. In: *The Educational Forum*. Zv. 50. 3. Taylor & Francis. 1986, s. 241–252.

[12] Dale H Schunk. *Learning theories an educational perspective sixth edition*. Pearson, 2012.

[13] Mozilla. *Mozilla Hubs*. 2018. URL: https://github.com/mozilla/hubs (cit. 30. 01. 2020).

[14] M. Wenzel, A. Klinger a C. Meinel. „Tele-Board Prototyper - Distributed 3D Modeling in a Web-Based Real-Time Collaboration System“. In: *2016 International Conference on Collaboration Technologies and Systems (CTS)*. IEEE. Okt. 2016, s. 446–453. DOI: 10.1109/CTS.2016.0084.

[15] Raja Gumienny et al. „Supporting Creative Collaboration in Globally Distributed Companies“. In: *Proceedings of the 2013 Conference on Computer Supported Cooperative Work*. CSCW '13. San Antonio, Texas, USA: Association for Computing Machinery, 2013, s. 995–1007. ISBN: 9781450313315. DOI: 10.1145/2441776.2441890. URL: https://doi.org/10.1145/2441776.2441890.

[16] Ian Fette a Alexey Melnikov. *The websocket protocol*. 2011. URL: http://www.hjp.at/doc/rfc/rfc6455.html.

[17] Lutz Gericke a Christoph Meinel. „Evaluating an instant messaging protocol for digital whiteboard applications“. In: *Proceedings on the International Conference on Internet Computing (ICOMP)*. The Steering Committee of The World Congress in Computer Science, Computer ... 2011, s. 1.

[18] S. Gunkel et al. „WebVR meets WebRTC: Towards 360-degree social VR experiences“. In: *2017 IEEE Virtual Reality (VR)*. IEEE, mar. 2017, s. 457–458. DOI: 10.1109/VR.2017.7892377.

[19] M. J. Prins et al. „TogetherVR: A Framework for Photorealistic Shared Media Experiences in 360-Degree VR“. In: *SMPTE Motion Imaging Journal* 127.7 (aug. 2018), s. 39–44. ISSN: 2160-2492. DOI: 10.5594/JMI.2018.2840618.

[20] Scott W Greenwald et al. „ElectroVR: An Electrostatic Playground for Collaborative, Simulation-Based Exploratory Learning in Immersive Virtual Reality“. In: *A Wide Lens: Combining Embodied, Enactive, Extended, and Embedded Learning in Collaborative Settings*, (jún 2019), s. 977–1000. URL: https://repository.isls.org//handle/1/1761.

[21] David Kut'ák et al. „An Interactive and Multimodal Virtual Mind Map for Future Workplace“. In: *Frontiers in ICT* 6 (2019), s. 14. URL: https://pdfs.semanticscholar.org/7a66/e70d465a8a752ccb409a7126b224298b5eed.pdf.

[22] M. Doležal, J. Chmelík a F. Liarokapis. „An immersive virtual environment for collaborative geovisualization“. In: *2017 9th International Conference on Virtual Worlds and Games for Serious Applications (VS-Games)*. Sept. 2017, s. 272–275. DOI: 10.1109/VS-GAMES.2017.8056613.

[23] "M.J. Callaghan et al. „"Client–server architecture for collaborative remote experimentation"“. In: *"Journal of Network and Computer Applications"* 30.4 (2007), s. 1295–1308. ISSN: 1084-8045. DOI: "https://doi.org/10.1016/j.jnca.2006.09.006". URL: http://www.sciencedirect.com/science/article/pii/S1084804506000749.

[24] Google. *WebRTC*. 2011. URL: https://webrtc.org/ (cit. 01. 03. 2020).

[25] Priologic Software Inc. *EasyRTC*. 2016. URL: https://easyrtc.com/ (cit. 01. 03. 2020).

[26] Bc. Kristína Matiková. „Kolaborácia vo virtuálnej realite: vstupno-výstupná časť pre MS HoloLens“. (V tlači). Dipl. pr. Technická univerzita v Košiciach, máj 2020.

[27] Bc. Tomáš Balluch. „Kolaborácia vo virtuálnej realite: vstupno-výstupná časť pre inteligentné telefóny“. (V tlači). Dipl. pr. Technická univerzita v Košiciach, máj 2020.

Zoznam skratiek

CVE Collaborative Virtual Environment – Kolaboratívne virtuálne prostredie.

FPS Frame per second – Snímok za sekundu.

G-CVE Global colaborative virtual environtments – Globálne kolaboratívne virtuálne prostredia.

HMD Head-mounted display – Displej pripevnený na hlave.

HTML HyperText Markup Language - Hypertextový značkovací jazyk.

HTTP Hypertext transfer protocol – Hypertextový prenosový protokol.

IMVMM An Interactive and Multimodal Virtual Mind Map – Interaktívna a multimodálna virtuálna myšlienková mapa.

KVR Kolaboratívna virtuálna realita.

LAN Local Area Network – Lokálna počítačová sieť.

LIRKIS Laboratórium Inteligentných Rozhraní Komunikačných a Informačných Systémov.

MUVE Multi-user virtual environment – Viacpoužívateľské virtuálne prostredie.

URL Uniform Resource Locator – Jednotný vyhľadávač prostriedku.

VLE Virtual Learning Environments – Virtuálne vzdelávacie prostredie.

VR Virtual reality – Virtuálna realita.

XML eXtensible Markup Language – Rozšíriteľný značkovací jazyk.

XMPP Extensible Messaging and Presence Protocol – Protokol rozšíriteľnej správy a prítomnosti.

Zoznam príloh

Príloha A CD médium – záverečná práca v elektronickej podobe

Príloha B Používateľská príručka

Príloha C Systémová príručka