TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Kaupo Sinimaa 185685

Lokaalvõrgus töötav meelelahutus veebirakendus

Bakalaureusetöö

Juhendaja: Eduard Petlenkov
Professor

Tallinn 2020

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Kaupo Sinimaa

28.04.2021

Annotatsioon

Käesoleva bakalaureusetöö eesmärk on luua lokaalvõrgus töötav meelelahutusrakendus, mis võimaldab antud võrgus olevatel seadmetel üksteisega suhelda. Selline suhtlus tagaks võimaluse mängida selliselt, kus ühel ekraanil näidatakse mängu staatust ja teistel seadmetel suheldakse mänguga.

Lõputöös seletatakse lahti lahenduse käik ja selle eripärad ning kasutatav tarkvara. Näidatakse loodud demo mängu näitel, kuidas näeb seadmete vaheline suhtlus välja. Tuuakse välja ka tegevused, mida oleks vaja tulevikus teha, et täiustada loodud tööd.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 22 leheküljel, 10 peatükki ja 8 pilti.

Abstract

Local Network Entertainment Web Application

The aim of this thesis is to create a web application, that works on the local network and allows other devices on that network to communicate with eachother in real time. This type of communication allows for creation of games where the status of working game will show on one screen and interaction will be done on secondary devices, preferably on a mobile device. Atleast one device has to be a computer with Linux operating system, others need to be a device, that has access to a web browser.

The properties of this application and used software are explained in detail throughout this thesis aswell as future improvements and extensions to further improve this application. More detailed demonstration and explanation on how this application works is showed with a demo game that is included. Said demo game will demonstrate, how devices can communicate with eachother and what is possible with the framework created for this application.

The thesis is in Estonian and contains 22 pages of text, 10 chapters and 8 pictures.

Lühendite ja mõistete sõnastik

GIT	*Global information tracker*, globaalne informatsiooni jälgija
AJAX	*Asynchronous JavaScript and XML*, päringu tegemise viis *backend*-iga
HTML	*HyperText markup language*, veebilehe mall
SQL	*Structured query language*, andmebaasi pärigukeel
ASGI	*Asynchronous server gateway interface*, päringute edastamise protokoll
IP	*Internet protocol*, sideprotokoll
Host	Server, kuhu seadmed ühenduvad
Backend	Veebiarenduse arvutuslik pool, serveri töötamine
Frontend	Veebiarenduse kujunduslik pool, kasutajaliides

Sisukord

1 Sissejuhatus 8
2 Eesmärk 9
2.1 Probleem 9
2.2 Lahendus 9
3 WebSocket ehk kahesuunaline serveri-kliendi suhtlus 10
4 Riistvara 11
5 Kasutatud ressursid 12
5.1 JavaScript ja HTML 12
5.2 Python 3.8 12
5.3 Teek Django 13
5.4 Teek psycopg2 ja andmebaas PostgreSQL 13
5.5 Teek django-channels 13
6 Realiseeritud lahendus 14
6.1 Programmi raamistik 14
6.2 Peamenüü 15
6.3 Mängud 17
6.3.1 JavaScript ja WebSocket 18
6.3.2 Demo mäng 18
7 Kasutajaliides 19
7.1 Host 19
7.2 Mängija 21
8 Katsetamine 23
9 Järgmised sammud 24
10 Kokkuvõte 25
Kasutatud kirjandus 26
Lisa 1 – Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks 27

Piltide loetelu

Pilt 1. Raamistiku failide struktuur 14
Pilt 2. Peamenüüde failide struktuur 15
Pilt 3. Demo mängu failide struktuur 17
Pilt 4. Hosti peamenüü 19
Pilt 5. Demo mängu ooteruum 20
Pilt 6. Demo mäng töötamas 20
Pilt 7. Mängija ühinemise vaade 21
Pilt 8. Demo mäng mängija vaates 22

1 Sissejuhatus

Käesoleva bakalaureusetöö raames keskendutakse mitme seadme vahelise meelelahutus veebirakenduse loomisele. Kasutatavateks seadmeteks on personaalsed arvutid ning nutiseadmed, millel on olemas veebibrauser. Seadmete vaheline suhtlus toimub juhtmevabalt lokaalvõrgus. Resultaadi kõikide osade valmimise eest vastutab Kaupo Sinimaa.

Arendus sai alguse käesoleva semestri alguses (2021 kevadsemester). Arendamise keelteks on kasutatud Python 3.8, mida kasutab taustal töötav lokaalvõrgu server, ning JavaScript, mida kasutatakse kasutajaliidese loomiseks ja kujundamiseks, taustal töötava serveriga suhtlemiseks ja mängu loogika defineerimiseks. Lokaalvõrgu serveri ülesandeks on loodavate mängude käivitamine, nendega seotud lehekülgede haldamine ning erinevate seadmete suhtluse võimaldamine. Kasutajaliides peab kasutajatele kuvama mängu hetkestaatust ning võimaldama kasutajatel mänguga suhelda.

Lõputöö eesmärgiks on luua vastav arhitektuuriline lahendus, mis võimaldaks ühes seadmes mänge käivitada ning kuvada selle staatust ning teistes seadmetes peab olema võimalus mänguga ühenduda ja suhelda. Põhiliselt oli vaja luua programmikood, mis võimaldab erinevatel seadmetel reaalajas üksteisega suhelda ja andmeid edastada.

Lõpptulemuseks valmis soovitud lahendus, kus lokaalvõrgus töötav server on võimeline käivitama rakendust ning mis läbi veebibrauseri kuvab kasutajatele nende seadmetele kasutajaliidese. Programmikood leiab aset GIT keskkonnas Github [1].

2 Eesmärk

2.1 Probleem

Hetkel ei eksisteeri sellist tasuta tarkvara, kuhu saaks iseseisvalt luua mänge, millega saaks mitmed seadmed korraga suhelda läbi veebibrauseri. Analoogseteks lahendusteks, mis tänapäeval eksisteerivad, on koolides kasutatav **Kahoot** [2], mille põhirõhk on küsimuste kuvamine ekraanil ning nendele vastamine telefonis, ning **Jackbox** [3], mis kasutab suhtlemiseks sama loogikat, kuid tavapäraste küsimuste asemel on siin mängulisem lähenemine. Kõige kasutajasõbralikum neist oleks **Jackbox**, mille kasutajaliides on professionaalselt loodud. Samuti kasutab **Jackbox** antud suhtlusviisi, et luua uusi ja huvitavaid mänge, mida inimesed saavad koos mängida. Siin on probleemiks aga **Jackbox**-i suhteliselt suur hind.

2.2 Lahendus

Luua iseseisvalt tarkvaralahendus, mis analoogselt eelmainitud rakendustele võimaldaks erinevatel seadmetel omavahel suhelda, kuhu oleks suhteliselt lihtne lisada enda loodud mänge ning mis oleks tasuta tarkvara. Selleks loome lokaalvõrgu serveri, kuhu saab lisada loodud mänge ning mis kuvaks seadmetele arusaadava kasutajaliidese. Lõpptulemuse programmikood jäetakse avatuks, mis võimaldab inimestel resultaati kasutada enda personaalsetes seadmetes. Samuti võimaldaks see tulevikus täiendada funktsionaalsust ja uute mängude lisamist.

3 WebSocket ehk kahesuunaline serveri-kliendi suhtlus

Kõige olulisem osa antud projektis on seadmete vahelise suhtluse loomine. Sellise suhtluse loomiseks on kaks varianti: **AJAX** päringud ja **WebSocket** ühendused. **AJAX** päringud võimaldavad JavaScript keeles teha päringuid serverisse, millele server vastab [4]. Seda saab vaadata ka kui küsimus vastus suhtlust, kus veebileht küsib serverilt küsimuse ja server vastab sellele. Kuna bakalaureusetöös on aga vaja luua ühendus, mis võimaldaks ka vastupidise suhtluse ning mis uuendaks seadmeid reaalajas, siis selline lahendus meile ei sobi.

WebSocket on suhtluskanal, mis loob seadmete vahel kahesuunalise reaalajas toimiva suhtluse seadmete vahel [5]. See tähendab, et seadmed saavad üksteisele saata informatsiooni selliselt, et teine seade saab informatsiooni koheselt kätte. Sellist ühendusviisi kasutatakse palju sotsiaalmeedias näiteks kiirsõnumite saatmiseks. Kuna käesolev bakalaureusetöö vajab sellist suhtlusviisi, siis on suhtluse arhitektuuriline lahendus loodud just **WebSocket** tehnoloogiat kasutades.

4 Riistvara

Käesoleva bakalaureusetöö programmi tööle panemiseks on vajalik personaalse arvuti olemasolu. Antud arvutis peab olema installitud Linux operatsioonisüsteem, kus on paigutatud lokaalvõrgu server. Arvuti peaks olema piisavalt võimekas, et samaaegselt hoida serverit töös ja kuvada ekraanile rakenduste hetkeseisu ja sõnumeid. Kuna programm kasutab tagasihoidlikumat ja lihtsamat disaini, siis pole siinkohal vajalik võimsa graafikakaardi olemasolu. Teistel kasutajatel on vajalik kas samamoodi personaalse arvuti või eelistatavalt nutitelefoni olemasolu, millel on veebibrauseri sirvimise võimalus. Kõik kasutatavad seadmed peavad olema ühendatud samasse lokaalvõrku, näiteks võivad seadmed olla ühendatud samasse Wi-Fi võrku.

Kuna bakalaureusetöö eesmärgiks oli luua lokaalvõrgus töötav meelelahutus veebirakendus, millega suhtlus oleks kasutajatel võimalikult lihtne, on ainult ühes seadmes vajalik programmi ülesseadmine. Teised kasutajad saavad suhelda veebibrauserit kasutades.

5 Kasutatud ressursid

Töös loodud programmi serveriks kasutati Django veebiraamistiku, mis lihtsustab seadmetele veebilehe mallide kuvamist. Kahesuunalise suhtluse saavutamiseks mitme seadme vahel kasutasin teeki nimega „django-channels“, mis võimaldab Django raamistiku siseselt luua veebisuhtluse loogikat. Andmebaasiga suhtluseks on vajalik teek nimega „psycopg2“, mis võimaldab Python-i ja PostgreSQL andmebaasi vahelise suhtluse.

5.1 JavaScript ja HTML

Frontend programmeerimiskeelteks kujunesid JavaScript ja HTML. JavaScript defineerib ära töötava mängu loogika ja sündmused, mis edastatakse läbi *backend* poolel loodud **WebSocket** ühenduse teistele seadmetele. HTML ja JavaScript-i koostöös luuakse seadmete ekraanidele programmi visuaalne pool, kus HTML vastutab elementide positsiooni eest ja JavaScript vastutab elementide funktsionaalsuse ja animatsioonide väljanägemise eest.

5.2 Python 3.8

Backend programmeerimiskeeleks sai valitud Python 3.8. Interpreteeriva keelena on Python piisavalt efektiivne ja funktsioonide rikas, et tagada käesoleva programmi töötamise stabiilsus ja kiirus. Python tagab tänu oma lihtsale ja loetavale koodi struktuurile ja vabavara teekide meeletule kogusele ka kiirema arendustegevuse, kus koodi kirjutamine võtab tunduvalt vähem aega. Bakalaureusetöös Python loob lokaalvõrgu serveri, tegeleb andmebaasist andmete lugemise ja salvestamisega ja haldab **WebSocket** ühendust, mille abil seadmed reaalajas omavahel suhtlevad.

5.3 Teek Django

Veebiarenduse raamistikuks kujunes Python-i baasil loodud veebiraamistik Django [6]. Django abil saame hõlpsasti luua lokaalvõrgus töötava serveri, kus on võimalik lihtsasti ära defineerida aadressid, kuhu seadmed saavad ühendada, ja vastavalt aadressile ka veebilehe malli, mida brauserid seadmetele kuvavad. Tänu Django suurele populaarsusele veebiarenduses on Django jaoks loodud mitmeid vabavara teeke, millega on võimalik täiendada loodud serveri funktsionaalsust.

Django tagab lihtsama ja arusaadavama suhtluse andmebaasiga, kasutades selleks Python-i klasse. Selline lähenemine muudab andmebaasi andmete kirjutamise ja lugemise lihtsamaks ja kiiremaks, elimineerides vajaduse arendajal kirjutada keerukat SQL koodi. Django toetab paljusid andmebaase, nagu näiteks MySQL, PostgreSQL ja paljud muud.

5.4 Teek psycopg2 ja andmebaas PostgreSQL

Kasutatavaks andmebaasiks osutus PostgreSQL andmebaas, mille suhtlusega tegeleb Python-i teek Django. Selleks, et Python keeles oleks võimalik selline suhtlemine, on vajalik käesolev teek. Teek sisaldab endas andmebaasi draiverit, mis defineerib suhtluse andmebaasi ja Python keele vahel, lubades Python-iga andmebaasi kirjutamise ja lugemise mitmelõimuliselt (ingl. Keeles *multi-threaded*) [7].

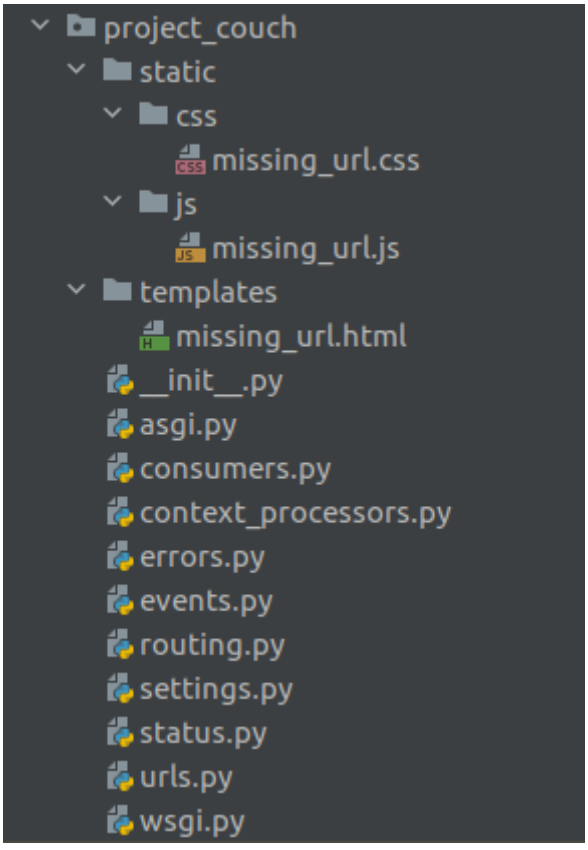
5.5 Teek django-channels

Rakendustes on vajalik luua kahesuunaline suhtlus seadmete vahel, et rakendused kuvaksid kõikides seadmetes ajakohast informatsiooni. Antud teek loob Django raamistiku siseselt **WebSocket** ühenduste võimekuse. Teek on üles ehitatud **ASGI** standardile, mis tagab seadmete vahel pikaajalise ühenduse ja kahesuunalise suhtluse [8].

6 Realiseeritud lahendus

Programmi lõpptulemus koosneb põhiliselt kolmest elemendist. Nendeks elementideks on programmi raamistik, peamenüü ja mängud. Programmi kõige esimeseks vaateks on peamenüü, kus kuvatakse projekti nimi ning kus kuvatakse kõik mängud, mida parasjagu saab valida. Mängijate jaoks on olemas eraldi lehekülg, mille kaudu saab ühineda mängudega. Selleks on vaja sisestada vastava mängu toa kood ning mängija enda ekraaninimi. Kui mängijad on sisestanud vastavad andmed, laetakse mängijatele mängule vastav ooteekraan. Mängud käivitatakse arvutis, kust mänge laetakse.

6.1 Programmi raamistik

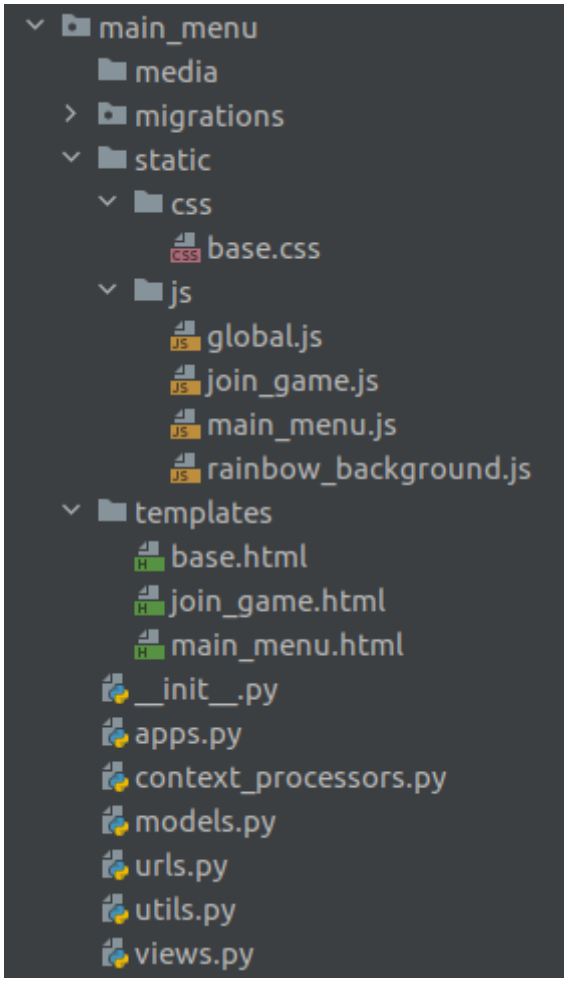


Pilt 1. Raamistiku failide struktuur

Pildil (vt. Pilt 1. Raamistiku failide struktuur) on välja toodud kõik failid, mis moodustavad programmi raamistiku. Need defineerivad kõik vajalikud seaded ja ühenduste definitsioonid, et server töötaks lokaalvõrgus korrektselt. Programmile pääseb ligi läbi veebibrauseri. Selleks on vaja seadmes sisestada hosti IP aadress. IP aadressiks on lokaalvõrgus ruuteri poolt määratud aadress antud seadmele, näiteks **192.168.1.160**.

Programmi seaded asuvad **project_couch/settings.py** kus defineeritakse põhiliselt andmebaasi andmed ja olemasolevate mängude nimekiri dünaamiliselt. **WebSocket** suhtluse loogika asub failis **project_couch/consumers.py**, mis tegeleb seadmete ühendamisega ja erinevate sündmuste edastamisega seadmest seadmesse. Kui seade peaks minema aadressile, mida ei eksisteeri, siis kuvatakse sellele seadmele veateade ja suunatakse tagasi peamenüüsse.

6.2 Peamenüü



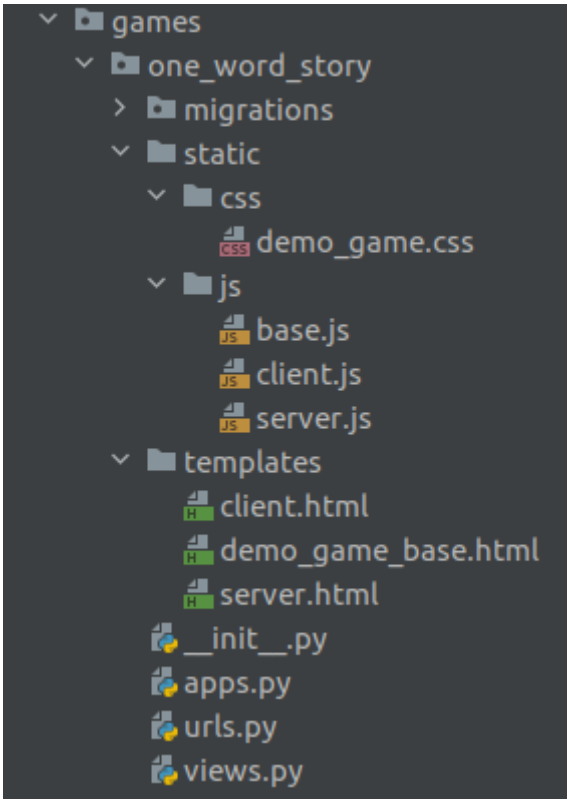
Pilt 2. Peamenüüde failide struktuur

Pildil (vt. Pilt 2. Peamenüüde failide struktuur) on ära toodud kõik failid, mis moodustuvad hosti peamenüü ja mängijate ühinemise ekraani. Kaustas **static/** on nii hosti kui mängija vajalikud JavaScript failid, mis võimaldavad suhelda *backend* poolega. Samuti on siin defineeritud kujundusega seotud atribuudid. Failis

static/js/rainbow_background.js toimub tausta värvi muutmine sujuvalt ühelt värvilt teisele. Tsüklis vahetatakse värve vastavalt vikerkaare värvidele.

Andmebaasiga suhtlus toimub läbi Django teegi, kasutades selleks Python-i klasse mis on defineeritud failis **main_menu/models.py**. Iga klass kujutab endas tabelit andmebaasis, kus klassi muutujad kujutavad endas tabeli välja. Klassi **Games** kasutatakse uute mängude loomiseks ja mängude staatuse määramiseks. Mängu staatused on defineeritud failis **project_couch/status.py**. Klass **Players** salvestab mänguga ühinenud mängija vajalikud andmed. Kaustas **main_menu/migrations** asuvad kõik vajalikud failid selleks, et Django oskaks luua andmebaasi vajalikud tabelid programmi töötamiseks. Kõik mängud leitakse üles dünaamiliselt selle järgi, mis kasutad asuvad kaustas **games**.

6.3 Mängud



Pilt 3. Demo mängu failide struktuur

Pildil (vt. Pilt 3. Demo mängu failide struktuur) on välja toodud mängude kaust, kus on näidatud demo mängu failid. Kõik saadaolevad mängud asuvad kaustas **project_couch/games**. Siin asuvad kaustad hoiavad endas kõiki vajalikke faile, mis defineerivad mängu kulgemise ja loogika. Aadressid, kuhu mängijaid edukal ühinemisel edasi suunatakse, asuvad failis **urls.py**, Kus defineeritakse ära mängu ühinemise ja loomise aadress ning samuti mängu lõpuresultaadi salvestamise aadress. Seal failis defineeritakse ära, mis funktsiooni mingi aadress käivitab. Kõik funktsioonid asuvad failis **views.py**. Funktsiooni **join_room** ülesandeks on selgeks teha, kas ühenduv seade on host või mängija ja vastavalt sellele anda seadmele tagasi korrektne veebilehe mall. Kui ühenduv seade on host, siis lisaks õige malli andmisele luuakse andmebaasi ka märge algatatud mängu kohta. Andmebaasis hoitakse algatatud mängu nime, mängu toa koodi, mängu staatust ja ühendatud mängijate nimekirja. Mängu toa kood genereeritakse automaatselt mängu käivitamisel. Kood koosneb viiest suvalisest tähest, mille mängijad peavad ühinedes sisestama enda seadmetes vastavasse kasti. Kui algatatud mäng on läbi, siis mängu jooksul tekkinud lõppresultaat salvestatakse andmebaasi **save_result** funktsiooni kasutades. Lõppresultaadi kuju eest vastutab *frontend*. Fail **apps.py** annab

mängule tema programmilise nime. Mängu detailsemad ja loetavamad kirjeldused paiknevad failis **__init__.py**, kus defineeritakse ära mängu nimi ja lühikirjeldus. Kõik veebilehe mallid, mida seadmetele näidatakse, asuvad kaustas **/templates**. Mängu disaini ja loogika eest vastutavad **/static** kaustas olevad failid, mis antakse seadmetele vastavalt sellele, kas seade on host või mängija.

6.3.1 JavaScript ja WebSocket

Mängu põhiline loogika ja kulgemise on loodud JavaScript failides. Selleks, et seade saaks ühineda toaga, on vaja ühendada *backend* poolel loodud **WebSocket** ühendusega. Selleks kasutatakse paika pandud aadressi malli, mis koosneb domeenist, mängu programmilisest nimest, mängu toa koodist ja mängija enda nimest. Kui ühendus on edukalt loodud, siis on seade ka edukalt ühendunud mänguga. Edasine programm töötab suuresti sündmuste baasil, kus kui mingi teatud sündmus juhtub ja see on ära defineeritud, siis käivitatakse mingi funktsioon. Põhilised sündmused, mis on ära defineeritud, on **player_joined**, **player_left**, **start_game** ja **close_game**. Sündmusi on võimalik ka juurde lisada, kui mäng seda peaks nõudma. Kõik sündmused edastatakse kõikidele mängus olevatele seadmetele läbi eelnevalt loodud **WebSocket** ühenduse.

6.3.2 Demo mäng

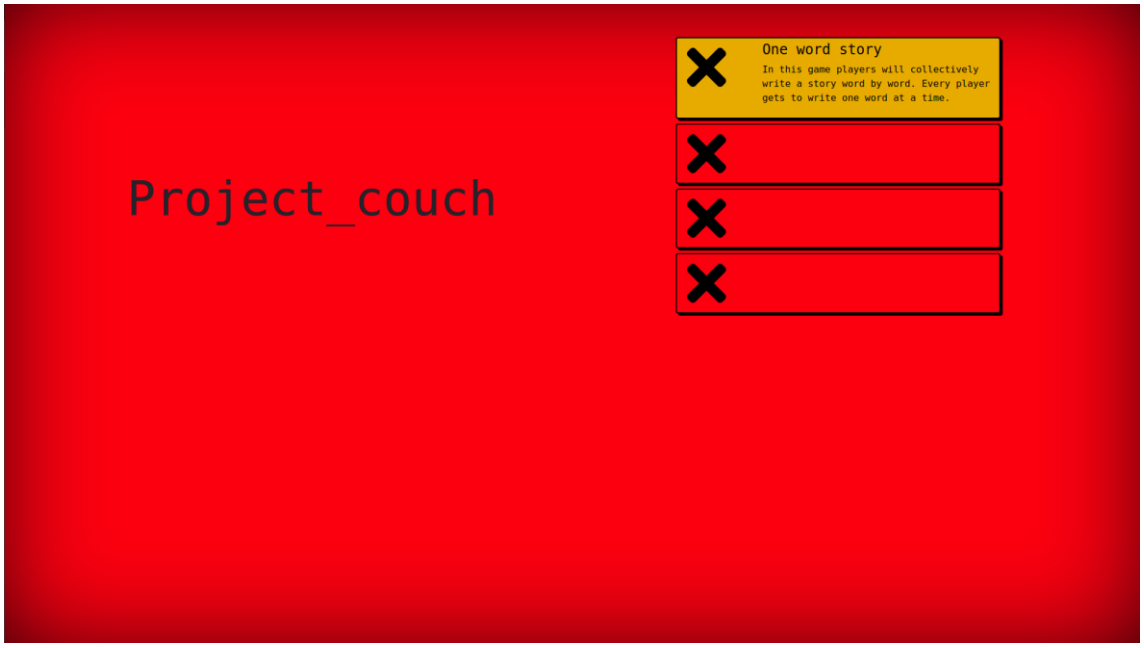
Loodud mängude raamistiku jaoks sai loodud ka mäng, mis demonstreerib mängu töötamist ja selle loogikat. Selleks mänguks on ühe sõna muinasjutt, mille eesmärgiks on seltskonnaga luua lugu selliselt, kus kõik mängijad tohivad korraga öelda ainult ühe sõna. Sõnu öeldakse mängijate seas ükshaaval.

Loodud demo mäng on suuteline ära demonstreerima, kuidas mängus olevad seadmed omavahel suhtlevad ja kuidas mängud antud programmis töötavad. Luues uue mängu toa kuvatakse hosti ekraanile mängu toa kood. Lisaks saab valida, kui kaua peaks mäng kestma. Kui mäng käivitatakse, siis hakkavad mängijad ükshaaval sõnu sisestama. Kui parasjagu on mängija kord, siis sisestab mängija tekstilahtrisse sõna ja edastab sõna hostile, kus see kuvatakse ekraanile kõigile nähtavaks. Selliselt ükshaaval sõnu sisestades loovad mängija oma niinimetatud muinasjutu. Kui aeg saab otsa, siis mängijad enam sõnu sisestada ei saa ja kuvatakse kõigile lõppresultaat, mis salvestatakse ka andmebaasi.

7 Kasutajaliides

Programmi kasutajaliidese eesmärk on olla informatiivne, arusaadav ja samal ajal ka atraktiivne. Käesolevas bakalaureusetöös oli oluline, et ekraanile ilmuvad elemendid oleksid arusaadavad ja samaaegselt ka visuaalselt ilusad, et tagada mängijate soov kasutada loodud programmi. Seetõttu läks palju rõhku sellele, et elementide visuaalne pool ja animatsioonid oleksid sobivad ja sujuvad.

7.1 Host



Pilt 4. Hosti peamenüü

Hosti ekraanile ilmuv põhiekraan (vt. Pilt 4. Hosti peamenüü) kuvab paremal pool ekraani kõiki mänge ning vasakul pool mängu enda nime. Mängu listi kastid reageerivad hiire liikumisele selliselt, et kui hiir on kasti peal, muutub kast väiksemaks. See muudab programmi reageerivamaks, andes kohest tagasisidet kasutaja tegevusele. Erinevate mängude visuaalse poole tagab mängu looja ise. Menüüs kasutatavad efektid on leitavad kaustas **project_couch/main_menu/static**.

Pilt 5. Demo mängu ooteruum

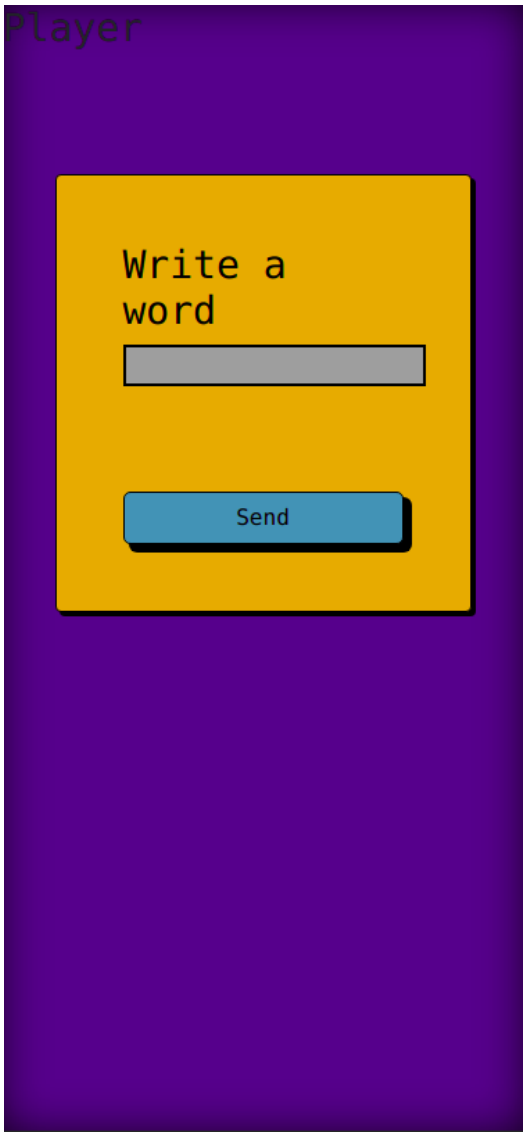
Programmiga koos olev demo mäng kasutab menüüs kasutatavaid efekte ja lahendusi. Mängu alguses kuvab host mängu toa koodi ja ühinenud mängijate nimekirja (vt. Pilt 5. Demo mängu ooteruum). Kui mäng käivitub, kuvatakse mängijatele suurelt mängu taimer, mis reaalajas väheneb. Valgel taustal on ka suurelt näha käesoleva mängu loodud tekst, kuhu lisandub sõnu igal edastamisel (vt. Pilt 6. Demo mäng töötamas).

Pilt 6. Demo mäng töötamas

7.2 Mängija

Pilt 7. Mängija ühinemise vaade

Mängijate põhimenüü ekraan koosneb kahest tekstiväljast (vt. Pilt 7.Mängija ühinemise vaade). Ühte on vaja sisestada käesoleva mängu kood ja teise on vaja sisestada mängija enda nimi. Mängija nimi võib olla ükskõik mis ja antud nimi kuvatakse selliselt, kuidas mängija on selle kirjutanud. Mängu toa koodi võib mängija kirjutada nii suurte kui väikeste tähtedega. Kui ühinemisel on tekkinud probleeme, siis kuvatakse veateade punaselt tekstiväljade all. Nendeks võib olla näiteks mängu toa puudumine või keelatud ühendumine toaga, mille mäng juba käib. Mängude disainiga seotud failid asuvad mängu enda kaustas.



Pilt 8. Demo mäng mängija vaates

Kaasasolev demo mäng kasutab samu efekte mis hosti menüü. Mängu jooksul kuvatakse ekraanile põhiliselt tekst, kus antakse teade, mis parajasti toimub. Kui on mängija kord kirjutada sõna, siis kuvatakse mängija ekraanil tekstiväli, kuhu saab kasutaja kirjutada kas ühe sõne või kirjavahemärgi (vt. Pilt 8.Demo mäng mängija vaates).

8 Katsetamine

Arendustegevuse üheks kõige olulisemaks punktiks on tehtud töö katsetamine. See aitab arendajal leida vigu, mida ta ise pole võimeline nägema, suuresti tänu faktile, et arendaja peab haldama programmi töö juures suurt enamust. Kuna käesolev töö on loodud ühe isiku poolt, on oluline, et programmi katsetaksid inimesed, kes ei ole kaasatud arendustegevuses.

Antud töö katsetamine toimus samas lokaalvõrgus, kus mängijad katsetasid demo mängu. Mängu protsessi näidati suurel ekraanil ja mängijad ühinesid mänguga läbi nutitelefonide. Mängu testimine võttis kokku ligikaudu tund, mille jooksul katsetati erinevaid aspekte mängu juures. Põhilisteks probleemideks kujunesid mängijate nimevalikud, kus teatud nimesid ei kuvatud suurel ekraanil. Samuti kui ühinenud mängija värskendas brauseri lehte, tekkis suurele ekraanile uus nimi, aga vana nimi ei kustunud. Lisaks leiti, et praegu määratud mängu kestuse maksimaalne aeg on liiga lühike. Praegu maksimaalseks kestuseks saab valida 5 minutit. Ülejäänud mäng töötas vastavalt esialgu määratud eesmärkidele.

9 Järgmised sammud

Realiseeritud lahendus täidab eesmärke, mis bakalaureusetöö käigus kehtestati, kuid loodud programm oleks edaspidisel arendusel võimeline palju rohkemaks.

Alustuseks oleks vaja luua rohkem mänge, mida kasutajad saaksid omavahel mängida. Kui praeguse demo mängu eesmärgiks oli rohkem demonstreerida seadmete vahelist suhtlust, siis edaspidised mängud saaksid olla palju detailsemad ja põnevamad. Uute mängude idee piirideks oleks arendaja enda fantaasia.

Praeguseks suureks ebamugavuseks on vajadus antud programm üles seade personaalses arvutis. Arvestades ka asjaolu, et praegune programm töötab ainult Linux operatsioonisüsteemi peal, võib see tavalise kasutaja jaoks olla väga arusaamatu ja raske, mis omakorda demotiveerib inimesi antud programmi kasutamast. Üheks võimalikuks lahenduseks on programmi installimise lihtsamaks tegemine selliselt, kus kasutaja käivitab installimise programmi, mis seab kogu mängu tema eest üles. Sellisel juhul tuleks kindlasti lisada ka tugi Windows operatsioonisüsteemi jaoks, mida kasutavad enamus inimesed tänapäeval. Teiseks variandiks oleks kas rentida või ise ehitada server, kuhu saaks programmi üles seada selliselt, et ligipääs mängule on igalpool võimalik ilma midagi installimata. Viimane variant oleks eelistatum, kuna tavainimeste jaoks muudab see mängu palju kättesaadavamaks.

Vajalik oleks uute mängude lisamise lihtsustamine. Selleks tuleks vähendada arendustöö mahtu, mida uue mängu jaoks tuleb teha. Üheks lahenduseks on võimalikult palju koodi ühtsustada selliselt, et arendaja ei peaks iga uue mängu puhul kirjutama ühte ja sama koodi mitu korda. Kuigi praeguses töös on palju koodi selliselt ühtsustatud, on seda tehtud rohkem *backend* poolel Python failides. Vajalik oleks ka mitmete JavaScript koodide ja funktsioonide ühtsustamine, mis hõlpsustaks uute mängude loomise kiirust ja mis muudaks mängude loomise arusaadavamaks.

Pikemas perspektiivis oleks vajalik luua programmile ka vastav dokumentatsioon, mis kirjeldaks põhjalikult ära programmi töötamist, kuidas programm tööle panna ning milliseid olemasolevaid funktsioone tuleks kasutada, et lisada uut funktsionaalsust.

10 Kokkuvõte

Kui kunagi oli tavaline inimestel kokku saada ning ühe ekraani taga mitmekesi mängida sama mängu, siis tänapäeval on olukord muutunud, kuna mitmed suured mängud ei toeta enam taolist mängimisviisi. Kuigi on olemas alternatiivid, nagu näiteks Kahoot või Jackbox, siis ei leidu nende seas suurt populaarsust suuresti tänu üksluisusele või suurele hinnale. Käesoleva bakalaureusetöö eesmärk oli luua vastav mängude raamistik, mis võimaldaks inimestel kogeda taolisi seltskonnamänge mugavalt ja tasuta.

Selleks sai loodud lokaalvõrgus töötav server, mille arhitektuur sai loodud Python 3.8-ga seda suuresti tänu Python-iga arendamise lihtsusele ja võimekusele ning vabavara teekide suurele arvule. Loodavad mängud on kirjutatud JavaScript programmeerimiskeeles, mis kujundavad mängu loogikat, suhtluse teiste seadmetega ja sujuva animatsiooni, et mäng oleks mängijate jaoks kutsuv. Seadmed suhtlevad omavahel kasutades **WebSocket** suhtluskanalit, mis võimaldab seadmetel kahesuunalise suhtluse reaalajas. Selline suhtlusviis on populaarne sotsiaalmeedia kiirsõnumites.

Programmi töötamise demonstreerimiseks sai loodud demo mäng, kus mängijad saavad ükshaalav sõnu kirjutades luua oma loo. Selleks antakse mängijatele piiratud aeg, mille mängijad saavad enne mängu algust ise valida. Mängijate kirjutatud sõnad kuvatakse suurelt hosti ekraanil.

Valmis saadud tulem kavatsetakse jätta avatud lähtekoodiga kättesaadavaks. See võimaldab ka teistel inimestel tulevikus programmi funktsionaalsust täiendada ja mänge juurde lisada. Samuti võimaldab see teistel inimestel programmi kasutamist tasuta. Et tagada programmile võimalikult lihtne ligipääsetavus, oleks ka vajalik programmi eraldi serverisse ülesseadmine, et mängudele oleks võimalik ligi pääseda ükskõik mis seadmest ja ükskõik kus kohast, kuniks mängijate seadmetes on veebibrauser ja interneti ühenduvus.

Kasutatud kirjandus

[1] „GitHub - project_couch,“ [Võrgumaterjal]. Saadaval: https://github.com/kauposinimaa/project_couch/ (05.05.2021)

[2] „Kahoot,“ [Võrgumaterjal]. Saadaval: https://kahoot.it/ (29.04.2021)

[3] „Jackbox Games,“ [Võrgumaterjal]. Saadaval: https://www.jackboxgames.com/what-is-jackbox/ (29.04.2021)

[4] „AJAX - Send a Request To a Server,“ [Võrgumaterjal]. Saadaval: https://www.w3schools.com/xml/ajax_xmlhttprequest_send.asp (03.05.2021)

[5] „The WebSocket API (WebSockets),“ [Võrgumaterjal]. Saadaval: https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API (03.05.2021)

[6] „Django,“ [Võrgumaterjal]. Saadaval: https://docs.djangoproject.com/en/3.2/ (05.05.2021)

[7] „PyPi - psycopg2,“ [Võrgumaterjal]. Saadaval: https://pypi.org/project/psycopg2/ (29.04.2021)

[8] „Django Channels,“ [Võrgumaterjal]. Saadaval: https://channels.readthedocs.io/en/stable/ (30.04.2021)

Lisa 1 – Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks[1]

Mina, Kaupo Sinimaa

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Lokaalvõrgus töötav meelelahutus veebirakendus“, mille juhendaja on Eduard Petlenkov.
   1.1. reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
   1.2. üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.
2. Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.
3. Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

28.04.2021

1 Lihtlitsents ei kehti juurdepääsupiirangu kehtivuse ajal vastavalt üliõpilase taotlusele lõputööle juurdepääsupiirangu kehtestamiseks, mis on allkirjastatud teaduskonna dekaani poolt, välja arvatud ülikooli õigus lõputööd reprodutseerida üksnes säilitamise eesmärgil. Kui lõputöö on loonud kaks või enam isikut oma ühise loomingulise tegevusega ning lõputöö kaas- või ühisautor(id) ei ole andnud lõputööd kaitsvale üliõpilasele kindlaksmääratud tähtajaks nõusolekut lõputöö reprodutseerimiseks ja avalikustamiseks vastavalt lihtlitsentsi punktidele 1.1. ja 1.2, siis lihtlitsents nimetatud tähtaja jooksul ei kehti.