TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Karl-Martin Miil IAPB143056

KULUEFEKTIIVSELT MÄNGUSERVERITE RENTIMINE LÄBI VEEBILIIDESE

Bakalaureusetöö

Juhendaja
Martin Rebane
MSc

Tallinn 2021

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: Karl-Martin Miil

Kuupäev: 18. mai 2021

Annotatsioon

Antud lõputöö eesmärgiks on luua veebirakendus, mille kaudu oleks kliendil võimalik soovi korral rentida endale lühiajalisi mänguservereid. Kasutajalt küsitakse raha vastavalt serveri tööajale ja mahule. Läbi selle suudab veebileht pakkuda teenust tunduvalt odavamalt ja keskkonnasõbralikumalt kui tavalised serveripakkujad.

Probleem, mida lõputöö lahendab, on kasutult töötavate mänguserverite peatamine ja seeläbi riistvara kasutuse optimeerimine. Tavaline mänguserveri pakkuja rendib välja serveri kuuks ajaks ning küsib raha terve selle perioodi eest. Tihtipeale seisab aga ostetud teenus kuu aega tühjana, sest mängijad on aktiivsed ainult paaril tunnil ööpäevas. Lahendus on süsteem, mis rendib välja servereid ainult selleks hetkeks, kui inimene seda kasutab.

Lõputöö autori ülesanne on luua rakenduste kogum, mille kaudu saab klient käivitada tarku mänguservereid, mis oskavad end iseseisvalt kinni panna olukorras, kus server töötab tühjalt.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 26 leheküljel, 6 peatükki, 9 joonist, 2 tabelit.

Abstract

Efficiently Renting Out Game Servers via Website

The purpose of this thesis is to create a web application that would offer customers temporary game servers for free or for a fixed hourly cost. This solution will decrease the amount of money needed for renting a game server. It will also decrease the load needed from the Server machines in the data centers, where these game servers would be hosted.

The main problem this thesis aims to solve is to decrease the amount of game servers that are running empty. By doing so there is a possibility to optimize the usage of hardware. Usual server providers rent out game servers for minimum of one month period, and bill the customer for that whole period. Although the player who rented the server does not use this server all the time, but in most cases only for a couple of hours during daytime. This means there is a significant loss in keeping the server running while no one is utilizing it.

The author's task is to create a solution with services. That the client can use to rent a server for the intended use time only. Meaning that the server would know when to shut itself down if it's empty.

The thesis is in Estonian and contains 26 pages of text, 6 chapters, 9 figures, 2 tables.

Nimekiri lühenditest ja terminitest

API	*Application Program Interface* rakendusliides
Boilerplate	kood või koodilõik, mis kordub
CI/CD	*Continious Integration/Continious deployment* Programmi automaatne testimine, serverisse laadimine ja käima panemine
Cache	*Salvestatud vahemälu* teatud perioodis korduvalt kasutatav mälu osa, kiirendamaks programmi tööd
DELETE	Päring rakenduses andmete kustutamiseks
Debug	Protsess koodist vigade leidmiseks
Framework	*Tarkvararaamistik* üldistatud funktsionaalsuse kogumik programmeerimise lihtsustamiseks
GET	Päring rakendusest andmete saamiseks
HTML	*HyperText Markup Language* Standard veebilehtede tegemiseks ja kuvamiseks
IP	*Internet Protocol* interneti protokoll
Integratsiooni testid	*Integration tests* Koodi testid, mis panevad suure osa rakendusest käima, et testida, kas rakendus töötab nii, nagu vaja
JSON	*JavaScript Object Notation* JavaScripti objektidel põhinev lihtsustatud andmevahetusvorming
JWT	*JSON Web Token* Standard kuju andmete saatmiseks üle veebi
LinxGSM	*Linux Game Server Manager* Linuxi mänguserveri Haldaja
Otspunkt	*Endpoint* veebilink mille kaudu on võimalik veebirakendusest pärida andmeid või käivitada äriloogikat
POST	Päring andmete rakendusse saatmiseks
PUT	Päring rakenduses valitud andme väljade uuendamiseks
Presenter klass	*Presenter class* klass, mis näitab välja ainult vajalikku informatsiooni kliendile
S3 bucket	Amazoni Pilverakendus, kus saab hoiustada faile ja veebilehti, et neid maailmale jagada

Teek	*Library* lisafunktsionaalsusega koodi kogu, mida saab enda programmis kasutada
UPDATE	Päring rakenduses andmete uuendamiseks
UUID4	*universally unique identifier v4* juhuslik 128-bitine number, mida kasutatakse andmete märkimiseks
Unit testid	*Unit tests* Äriloogika testid koodis
VPS	*Virtual Private Server* virtuaalne privaatserver

Sisukord

Joonised

Tabelid

1. Sissejuhatus

1.1 Probleemi taustast ja motivatsioonist

Viidates Gametracker.com [1] andmetele, siis 2020. aasta märtsi alguse seisuga on jooksmas 54,915 Gametrackerile nähtavat mänguserverit, milles mängib 171,672 mängijat. Nendest numbritest saab teada, et igas serveris mängib keskmiselt 3.13 mängijat. Probleem on selles, et enamasti ei jaotu inimesed keskmise arvuna üle kõigi serverite, vaid koonduvad mõnda kindlasse kohta - näiteks ühes serveris võib olla 64 inimest, kuid mõnekümnes teises 0 inimest. Selle asemel, et servereid tühjana käimas hoida, saaks need välja lülitada.

Mänguserverid nõuavad arvutiressursse ka siis, kui mitte keegi neid ei kasuta. Võttes näiteks Minecrafti soovitatud ressursside infolehe, saab teada, et hoidmaks avalikult internetis 1-4-kohalist serverit, on vaja vähemalt 2-tuumalise protsessoriga ja 1GB mäluga serverimasinat. Hetkeseisuga saab sellise virtuaalse privaatserveri rentida Ovh.io [2] serveripakkuja lehelt 15 euro eest kuus. Mängija, kes masina rendib, ei kasuta seda konstantselt terve kuu aega, vaid heal juhul ainult mõnikümmend tundi. Kui oleks võimalik tühjad serverid kinni ja nõudluse korral need uuesti käima panna, siis oleks võimalik kliendil säästa raha, makstes ainult tundide eest, millal ta teenust kasutab.

Antud probleemi lahendamiseks pakub välja autor automaatikat kasutava veebirakenduse, mille kaudu saab serverit sooviv inimene endale lühiajalise mänguserveri võtta, konfigureerides seda omal soovil. Käima pandud server on ühendatud tarkvaraga, mis suudab lugeda andmeid serverist ning teab, millal see on tühi või mitte. Juhul kui serveris ei ole kedagi, paneb tarkvara mänguserveri kinni ja seeläbi säästab kliendi raha. Antud töö ei käsitle kliendi tasustamist, mis on seatud tulevikuplaanidesse. Töö raames seab autor endale eesmärgi luua lahendus, mis lubab serveri soovijal tasuta mängu Minecraft serveri võtta nii kauaks, kuni kasutaja seal mängib. Tulevikus saab ka sama serverit läbi veebilehe uuesti käima panna, kui mängul on salvestatav seis, mida klient võiks tahta jätkata.

Samuti on eesmärgiks süsteem võimalikult skaleeritavaks teha, kasutades selleks Amazoni, OVH.ie ja Digital Oceani pilveteenuseid. Plaan on utiliseerida maksimaalselt kasutatavat virtuaalse serveri riistvara, küsides kliendilt, kui palju inimesi hakkab serverit kasutama ja selle põhjal valida talle õige võimsusega server.

2. Analüüs

Antud peatükis kirjeldatakse loodava veebirakenduse funktsionaalsed nõuded, esitatakse sarnased teenused ja selgitatakse töö plaani.

2.1 Funktsionaalsed nõuded

Esmase süsteemi planeerimisel jagunes töö vastavalt kõige tähtsamatele kasutajalugudele:

- kasutaja saab registreeruda ja sisse logida veebirakendusse;
- kasutaja saab tasuta rentida Minecrafti mänguserveri;
- kasutaja saab muuta mänguserveri konfiguratsiooni enne rentimist;
- kasutajal peab olema võimalus server ise kinni panna;
- süsteem peab oskama mänguserveri ise kinni panna, kui see on tühi.

Registreerumist ja sisselogimist on vaja selleks, et inimene näeks enda käivitatud mänguserveri kohta infot ja saaks sellele käsklusi anda.

Pärast kasutaja tegemist ja enne veebilehe kasutamist peab liituja enda e-maili verifitseerima. Antud lisasamm on selleks, et ei tekiks olukorda, kus pahatahtlik inimene teeb kellegi teise e-mailiga konto ja kasutab veebilehte mitteotstarbelisel eesmärgil. Pahatahtlik näide on käivitada üle kümne serveri viie minuti jooksul erinevate kontodega. Kuna tühjana seisvad mänguserverid panevad end ise kinni mõne aja möödudes, siis selline käitumine ei tekitaks väga suurt kahju, kuid oleks siiski koormav andmebaasile ja süsteemile endale.

Esialgseks eesmärgiks on tööle saada Minecrafti mänguserveri haldamine, konfigureerimine ja sulgemine. Minecrafti server otsustus valituks, sest töö autor on eelnevalt tegelenud antud mängu serveritega.

Konfiguratsiooni all on võimalus kliendil valida, kui palju inimesi saab liituda mänguserveriga, selle järgi otsustab süsteem kui võimsat - nii protsessori kui mälu poolest - virtuaalserverit mängu jaoks kasutada. Samuti saab sellel leheküljel muuta mängu enda sätteid, nagu näiteks raskusaste ja mängu tüüp.

2.2 Olemasolevad teenused

Aternos[1] tasuta Minecrafti serverite rentija on üks suurimaid Minecraft serverite pakkujaid, mis hoiab enda servereid üleval reklaamist saadud tulu pealt. Lisaks on neil tütarfirma, mis pakub kuuajalisel tasustamismudelil servereid. Veebilehel saab kasutaja küll serveri käima panna, kuid see ei lähe ise kinni, kui server jääb tühjaks, ja on piiratud ainult mõnetunniseks kasutamiseks[3].

FreeGameHosting[2] on mänguserverite rentija veebileht, mis pakub servereid viieteist-kümnele mängule. Serverite rentimine on tasuta, kuid piiratud perioodiks ja ainult siis, kui vabu servereid on olemas. Piiratud perioodi ajaliselt pole kuskil mainitud, kuid on lisatud, et piirangu vältimiseks tuleks raha eest kuuajaline server rentida. Saab järeldada, et mängud jooksevad ühe suure masina peal ja kui ressurss otsa saab, siis rohkem mänguservereid ei saa[4].

Popflash[3] võistlusserveripakkuja on üks kuulsamaid lühiajaliste serverite pakkujaid. Nad pakuvad servereid ainult ühele mängule: Counter-Strike: Global Offensive, kuid nende teenus on eeskujuks antud tööle. Veebilehelt saab võtta pealtnäha lõputult vabu mänguservereid ükskõik mis hetkel. Süsteem hoiab serverit käimas hetkeni, kuni mängijad seda kasutavad. Popflashi eripära on igakuine tasu kliendile, selleks et tal oleks võimalus server käima panna. See kuutasu on väiksem kui maksta kuu perioodil terve serveri eest. Kuna sellise mänguserveri jaoks on vaja 15-eurost serverit ja teenuse hind ise on 5 eurot, siis hinnavõit on kümme eurot.

Mcprohosting[4] on mänguserveripakkuja, mis pakub igakuise tasu eest servereid mitmetele populaarsetele mängudele kui ka tavalisi virtuaalseid privaatservereid. Veebilehel on reklaamitud ka 7-päevane tasuta periood, mille ajal saab proovida nende serverit. Pärast perioodi tuleb tasuda tavalist igakuist tasu, mis ulatub 7 eurost kuni 20 euroni.

Internetis otsides ei avastanud autor ühtegi täpselt sama ideega veebiteenusepakkujat. Enamus neist on siiski kindlalt määratud perioodiks serverite rentimisega tegelevad ettevõtted. Välja arvatud FreeGameHosting, mis pakub küll lühiajaliselt servereid, kuid kindlaks ajahetkeks nagu näiteks kuueks või kaheksaks tunniks. Erandiks on ka Popflash, mis sarnaneb töö ideega, kuid on suunatud ainult ühele mängule ja töötab erilisel kuutasu baasil nagu Spotify.

[1] https://aternos.org/:en/
[2] https://www.freegamehosting.eu/
[3] https://popflash.site/
[4] https://mcprohosting.com/order

2.3 Töö planeerimine ja maht

Projekti idee saadi 2019 novembri alguses ja planeeriti valmis saada esialgne prototüüp 2020 detsembri lõpuks. Töö tegemine jagunes osadeks:

- veebiliidese valmistamine, kus kasutaja saab sisse logida ja serveri võtta;
- (API, *application program interface*) programmeerimine, mis võtab vastu käsklusi veebiliidesest ja suhtleb andmebaasiga ning väliste teenustega;
- läbi (VPS, *virtual private server*) teenusepakkuja API mänguserveri käima panemine;
- serverimasinasse rakenduse tegemine, mis suudab mänguserverit jälgida, konfigureerida ja vajadusel sulgeda.

Tulevikuplaan, kui eelmainitud elemendid on hästi teostatud, oleks valmis saada lisamoodulid, nagu näiteks päris raha eest veebilehel punktide ostmine, serveri jätkamine seisust, kus kasutaja või server selle kinni oli pannud, ja muid teisi lisasid.

2.4 Mänguserverile kuluvad ressursid

Eesmärk on enne serveri käima panemist küsida kliendilt, mitu inimest hakkab mänguserverit kasutama. Nii saab valida vastava võimekusega virtuaalse serveri ja seeläbi kliendile raha säästa. Kuna ei saa eeldada, et klient teaks, kui tugevat serverimasinat läheb tal vaja. Siis oli vaja uurida infot serverite kohta, kui ka testida manuaalselt virtuaalsete serverite piire. Nii saab panna inimeste arvu valiku vastama valitava serveri võimsusega ning teha kliendile otsustamise palju lihtsamaks.

Esimeseks on Minecrafti server ning järgnev info on saadud mänguserveri Wikipedia lehelt[5]. Niisama jooksutamiseks piisab minimaalselt 512MB mälust ja ühest 2Ghz protsessori tuumast. Selles serveris saab mängida heal juhul 3 inimest. Kolme kuni viie inimesega mängimiseks läheb vaja 1GB mälu ja ühetuumalise 2.8Ghz protsessoriga serverit. Viie kuni kaheksa inimesega mängimiseks läheb vaja 3GB mälu ja kahetuumalise 2Ghz protsessoriga serverit. kaheksa kuni kaheteistkümne inimesega mängimiseks on vajalik 5GB mälu ja kahetuumalist 2.8Ghz protsessorit. Alates kaheteistkümnest inimesest on juba vajalik 6GB mälu ja neljatuumalist 2Ghz protsessorit[5]. Antud seaded said ka reaalses elus testitud ja vastasid Wikipedias mainitule.

Teiseks mänguserveriks on CS:GO server. Mitteametlikust Alibaba Cloud artiklist loeb välja, et serveri jooksutamiseks on vaja vähemalt kahe virtuaalse tuumaga ning 2GB

mäluga serverit. See on siis vajalik serveri jooksutamiseks kümnele inimesele. Samuti on mainitud, et on vaja suuremat kettamahtu kui 40GB, sest mängu failid ületavad 40GB [6]. See mitteametlik teooria sai ka reaalses elus läbi proovitud ning vastas tõele. Pannes mänguserver käima ühe tuuma peal, siis mängu ja serveri vaheline ühendus ei ole piisavalt hea, et serveris mängida. Mängu debug aknast on näha, kuidas server ei suuda infot piisavalt kiiresti vastu võtta ja kaotab andmepakette.

Ära peab ka mainima, et virtuaalne server koos selle peal jooksva Ubuntu Linuxiga võtab veel peale mänguserveri lisaresursse ning valides serveri võimsust, kuhu mänguserver käima panna, peab sellega arvestama.

3. Taust ja kasutatavad tehnoloogiad

3.1 Agones teek mänguarendajatele

Agones on teek, mis oskab skaleeritavalt jooksutada mänguservereid Kubernetese baasil. Agonesi tiim alustas aastal 2017 koostöös Ubisofti ja Google Cloudiga, et välja töötada avatud lähtekoodiga lahendus mänguserveritele, mis suudaks iseseisvalt servereid käima panna vastavalt mängijate vajadustele. See on võimalik tänu Kubernetese konteinerite haldussüsteemile. 2019. aasta septembris, pärast kahte aastat koostööd erinevate mängutootjatega, anti välja Agones 1.0.0 [7].

Agones kasutab Kubernetest, et mänguservereid sisaldavaid konteinereid luua, jooksutada ja hallata. Seda siis läbi tavaliste Kubernetese klastritööriistade. Teegiga saab jooksutada servereid nii pilves, lokaalses masinas või mõnes muus serverimasinas. Arendatud funktsionaalsus on eelkõige suunatud mängutootjatele, kes vajavad head skaleeruvust läbi interneti toimivatele mängudele.

Kui kasutada Agonesi teeki antud töös, siis tuleks iga mängu jaoks kirjutada eraldi arendus ja liidestus nii mänguserveriga kui ka veebiliidesega. Sest enamus mänge on erinevate mängumootorite peale ehitatud. Lahendus töötaks, kui arendada süsteem ümber ühe mängu ja ainult sellele servereid pakkuda, mis on küll töö eesmärk. Aga kuna tulevikuplaan on toetada nii palju mänge kui võimalik, siis ei näe autor suurt kasu kasutamaks Agonesi teeki.

3.2 OVH.ie kubernetes cloud

OVH.ie pilveplatvorm jagab enda serverid kaheks: parema protsessoriga serveriteks ning suurema kõvakettamahuga serveriteks. OVH.ie pakub välja ka tööriistad, mis lihtsustavad protsesside automatiseerimist läbi nende enda API. Näitena võib tuua rakenduse automaatset taaskäivitamist koodimuudatuste peale või ressursivajaduse põhine virtuaalserverite käivitamine ehk skaleerimine [8].

Vaadates OVH Kubernetes Cloudi hinnastamise tabelit, on selge, et nad on keskendunud suurtele klientidele, sest enamik servereid, mida saab teenusega käivitada, on võimsuselt

suured ja kallid. Väikeste serverite valik on peaaegu puudulik. Kõige suurem Dockeri virtuaalne server, mida Kubernetes suudab üles panna, on 32-tuumaline ja 180 GB mäluga, ja kõige väiksem, mida ka selle töö juures vaja läheb, on ühetuumaline ja 2 GB mäluga. Eesmärk on võimalikult vähe arvutusresurssi kaotsi lasta, seepärast pole mõttekas kasutada suuri virtuaalseid masinaid. Ja nagu ka eelnevalt uurimisest välja tuli, siis mänguserverid, mida süsteem hakkab jooksutama, tegelt ei nõua nii palju ressursse. Isegi eelmainitud kõige väiksem pakutav server on liiga võimekas[9].

OVH Kubernetes Cloud pakub hinna poolest kõige odavamat väikest serverit Dockeri konteineri jooksutamiseks. Töö kirjutamise hetkel on Sandbox virtuaalse keskkonna ülevalhoidmise tunnihinnaks 0.008 eurot, ehk 5.12 eurot. Probleem on aga selles, et tegu on kõige väiksema virtuaalse masinaga, mida nad pakuvad ja nende enda sõnade järgi selle virtuaalserveri eesmärk ei tohiks olla jooksutada täisrakendust, vaid pigem scripte, mis panevad rakendusi tööle. Samuti on kirjas, et nende serverite jõudlus võib ajaliselt muutuda, ehk pole tagatud sajaprotsendiline töökindlus ja võimsus[9].

Kuna teenus peab suutma vastavalt kasutaja valikule käima panna ka nõutud jõudlusega serveri, siis kahjuks ei saanud OVH cloudi selle süsteemi osaks kasutada, sest kõik järgnevad skaleerivad virtuaalsed masinad on sama suured, kui seda oli esimene. Ja nagu eelnevalt uuritud nõuetele sai teada, siis erinevad mängud ja erinevad kliendid nõuavad erinevat tüüpi ja jõudlusega servereid. Lisaks on vaja kindlat ja töökindlat võimsust, mida antud teenusepakkuja ei suuda enda kõige väiksema võimekusega serveri juures pakkuda.

3.3 Avaliku API-ga ühendatud virtuaalserverite pakkujad

Kolmas uuritav platvorm mänguserverite jooksutamiseks on Avaliku API-ga serveripakkujad, nagu näiteks Digital Ocean, Vultr.com ja, Eesti enda pilveteenust pakkuv Pilw.io. Need ettevõtted rendivad välja väikeseid virtuaalseid privaatservereid (VPS), mida kasutaja saab iga kell läbi nende veebilehe käima panna ja sulgeda, makstes ainult perioodi eest, millal server tegi tööd. See idee on ka otseselt seotud antud töös seatud eesmärgiga ning uurimise tulemusena otsustati kasutada ühte eelmainitud kolmest teenusepakkujast. Projekti jaoks on tähtsal kohal API lihtsus ja ühilduvus Javaga ning serverite hind kuus. Eesmärgiga võimalikult vähe ressursse kulutada nii ajaliselt arendamise mõttes kui ka rahaliselt. Seega siinkohal on hea hetk võrrelda neid kolme antud kriteeriumite põhjal (vt tabel 2).

Tabel 1. *Võrdlus erinevate teenusepakkujate vahel*

Nimi	**Hind kuus**	**Dokumentatsiooni keeled**	**Java liidestuse kvaliteet**
Pilw.io	8 €	Curl	Puudub liidestus
Digital Ocean	4,96 €	Curl, Ruby, GO	Ajakohane Java teek Digital Oceani poolt
Vultr.com	4,96 €	Curl	Kasutajate poolt valmistatud mitte ajakohane teek

Pilw.io on Eesti ettevõte, mis on kolmest kõige noorem ja teenust hakati pakkuma aastal 2018 [10]. Ühe gigabaidise mälu ja ühe tuumaga serveri hind on 0.01050 eurot tunnis ehk 8 eurot kuus, millele on juba juurde arvutatud käibemaks, mida teisel kahel võrreldaval teenusepakkujal polnud [11]. Pilw.io API dokumentatsioon on väga lihtsasti loetav ja kasutajasõbralik. Ette on antud ka näited, kuidas päringuid teha läbi Curli. Kahjuks ei leitud avatud lähtekoodiga teeki, mis ühendaks Pilw.io API ja Java rakenduse. Seega kui valida, Pilw.io-l tuleks näha vaeva, et integreerida antud API Javasse.

Digital Ocean on kolmest kõige kuulsam ja kõige vanem sellise teenuse pakkuja, mis asutati aastal 2011 [12]. Ühegigabaidise mälu ja ühe tuumaga serveri hind on 0,006 eurot tunnis ehk 4,13 eurot kuus. Nendele hindadele tuleb juurde arvutada ka hetkel Eesti Vabariigis kehtiv käibemaks 20 protsenti, mis teeb serveri hinnaks ümardatult 0,007 tunnis ehk 4,96 eurot kuus [13]. Digital Oceani API dokumentatsioon on sarnaselt üles ehitatud, nagu on Pilw.io API dokumentatsioon, kuid on detailsem, kirjeldamaks, kuidas iga teenuse funktsionaalsus töötab, ja sisaldab näiteid päringute kohta Curlis lisaks ka programmeerimiskeeltes Ruby ja GO. Digital Oceani pluss on ka see, et teenus on laialdaselt kasutusel ja on olemas avatud lähtekoodiga Digital Oceani poolt valideeritud ja ajakohane teek Java programmeerimiskeeles. Tänu sellele on võimalik antud töö raames säästa arendusaega.

Vultr.com on kolmest vanuselt keskmine serveri pakkuja ja teenust hakati pakkuma aastal 2014 [14]. Ühegigabaidise mälu ja ühe tuumaga serveri hind on täpselt sama, mis on Digital Oceanil. Mis on arusaadav, sest need kaks teenusepakkujat on omavahel suurimad konkurendid [15]. Vultr API dokumentatsioon on sarnane Digital Oceani dokumentatsioonile, kus on väga detailselt kirjas, mida mingi veebiteenuse otspunkt teeb. Samas pole Vultr lisanud näiteid kasutuse kohta teistes programmeerimise keeltes peale Curli. Vultr avaliku koodiga teeki otsides leidus paar projekti Githubis, mida polnud paar aastat uuendatud ja lähemal uurimisel, ei sisaldanud kõiki funktsioone, mida Vultr välja pakub. Samuti ei olnud teegid Vultr-i enda poolt valideeritud.

Uurides neid kolme teenusepakkujat, siis hinna poolest on parimad Digital Ocean ja Vultr, mis pakuvad sama raha eest sama võimsusega serverit. Siinkohal jäi Eesti teenusepakkuja neist maha, müües pea 37 protsenti kallimat teenust. API dokumentatsiooni poolest on kõik kolm väga head, kuid rohkem detaile erinevate võimaluste kohta saab Digital Oceanilt ja Vultrilt. Koodi näidised API dokumentatsioonis oli kõige paremini esile toodud Digital Oceanil, mis pakkus kolmes programmeerimise keeles näiteid. Avatud lähtekoodi otsides, mis aitaks API liidestamist süsteemi, oli kõige paremini hakkama saanud Digital Ocean, millel on ametlik Digital Oceani poolt tunnustatud ja ajakohasena hoitav teek. Seega mänguserverite jooksutamiseks sai valitud Digital Ocean.

3.4 Java Backend API teenus

Süsteemi suurim osa ja teisi teenuseid ühendav rakendus on kirjutatud Java programmeerimiskeeles. Kasutusel on Spring Boot 2.2.0 koos Java versioon 11.0-ga. Spring Boot 2.2.0 ja Java sai valitud sellepärast, et projekti kallal töötavad inimestel on eelnev kogemus antud tööriistadega. Kuna tegu on hobiprojektiga, siis ajavõit arenduse vaates oli igati väärtuslik, kasutadades juba teadaolevaid keeli ja raamistikke.

Kasutusel olev Java 11 sai ka uuendatud Java 13 peale, et oleks võimalik kasutada uut tekstiplokkide lisamoodulit. Lisand lubab sõne kirjutada mitmerealiselt ilma, et kaotaks tekstivormindust [16]. Soov oli selle mooduliga muuta lihtsamaks süsteemis olevate serverite konfiguratsioonifailide muutmine ja salvestamine. Kahjuks antud uus Java lisafunktsioon ei töötanud nii, nagu oli soovitud. Nimelt ei olnud võimalik tekstiplokkide sisse paigutada muutujaid, nagu on see võimalik mõnes teise programmeerimise keeles, nagu näiteks Pythonis või Kotlinis.

Spring on raamistik, mis koondab kokku kõik vajaliku, et valmistada Java veebirakendus. Moodulid, mis Springiga kaasa tulevad, aitavad drastiliselt vähendada arenduseks nõutavat aega. Suurim pluss on, et Springi saab konfigureerida nii, et ta kirjutaks programmeerija eest boilerplate koodi. Siiski Springi raamistik ise ei ole täiuslik ning selleks on välja toodud edasiarendus Spring Boot raamistik, mis võtab rakenduse konfigureerimise töö arendajalt ära ning lubab programmeerijal keskenduda ainult programmi loogikale ja ärile [17].

3.5 Javascripti serverihaldusteenus

Java Backendi ja VPS-i siderakendus on programmeeritud Javascriptis kasutades TypeScripti keelt. TypeScript kompileeritakse javascriptiks ning laetakse üles pilve, täpsemalt

Amazoni S3 teenusesse. Pilvest laetakse kood alla VPS-i, siis kui mänguserver pannakse läbi veebilehe käima. Server jooksutab antud koodi Node versiooniga 12.16.3. Põhiline kasutus on siderakendusel Java backendist päringute tõlkimine tegevusteks. Näiteks serveri sulgemine või siis mõne käskluse mänguserverile andmine.

TypeScript on vabavaraline programmeerimiskeel, mis on tuntud selle poolest, et sellega saab lisada staatilisi tüübikirjeldusi muutujatele. Sellepoolest sarnaneb keel ka näiteks Javale ja C#-le. Javascript on ka Typescripti alamkeel, ehk kõik Javascripti rakendused töötavad ka Typescriptis [18].

3.6 Kasutajaliides

Kasutajaliidesena on kasutusel HTML veebileht, mis on kodeeritud ja kompileeritud, kasutades Vue.js versioon 2 raamistikku. Lisaks sellele on kasutusel Vue.js lisaraamistik Vuetify, mis muudab kasutajaliidese arendamise lihtsamaks. Vuetify-sse on kokku pakitud sadu materiaalse disaini veebielemente, mida on võimalik veebilehe valmistamisel kasutada. Kuna projektiga töötavad inimesed ei ole just kõige paremad kasutajaliideste disainerid, siis valitud raamistikud olid vägagi abiks, sest enamik visuaalseid elemente, nagu näiteks tabelid, modaalid, nupud, on juba raamistikus ette tehtud.

Vue on uuenduslik veebi kasutajaliidese raamistik, mis on disainitud mõttega, et seda oleks võimalikult lihtne liidestada teiste teekidega. Vue ise on ainult veebivaate jaoks mõeldud kerge tööriist. Vue ei sisalda liigseid konfiguratsioonifaile ega nõua kasutajalt liigset mõtteaega, et kuidas struktureerida failide tasemel oma projekti[19].

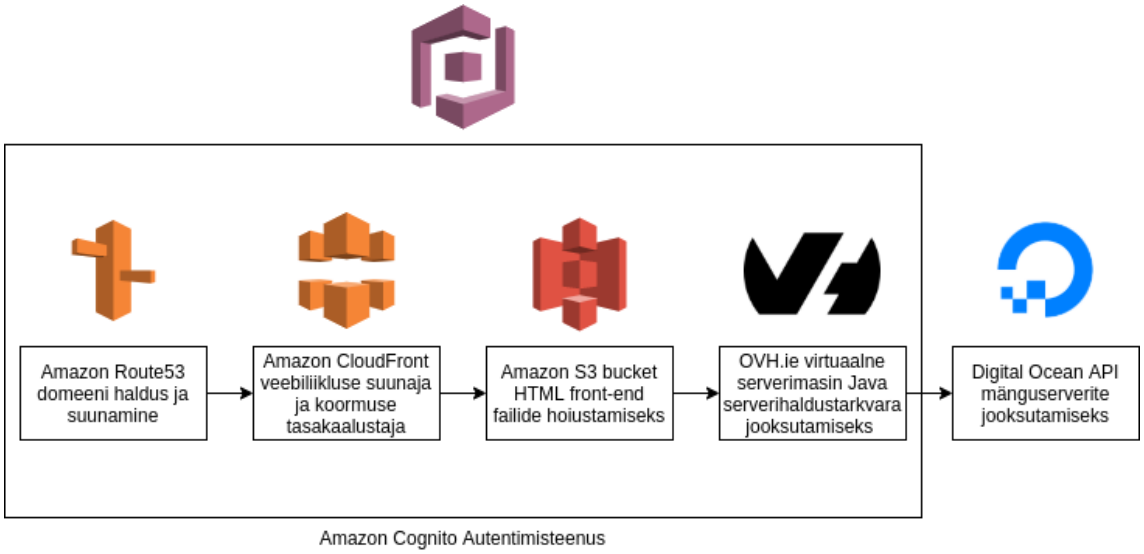
Vuetify on see-eest aga komponentide kogum Vue jaoks. Seda on arendatud aastast 2016 ja sisaldab kõike vajalikku, et valmistada prototüüp. Vuetify komponendid on suunatud valmistamiseks adminipaneele või üheleheküljelisi reklaamlehti. Samuti on kõik Vuetify komponendid mobiilisõbralikud[20].

3.7 Koodihoidla ja automaatne rakenduse uuendamine

Koodihoidlana on kasutusel GitHub, milles hoitakse Java veebiteenuse koodi, manageerimis- ja siderakenduse koodi ning Kasutajaliidese koodi. GitHubil on olemas toode nimega GitHub Actions. Github Actions on automatiseerimiseks mõeldud tööriist, millega on võimalik jooksutada erinevaid, kas siis nende poolt välja töötatud või kasutaja enda poolt kirjutatud scripte. Script võib näiteks koodi muudatuse peale käima panna testid ning kompileerida ja üles laadida koodi, mis on repositooriumis[21].

4. Rakenduse arhitektuur ja kirjeldus

Enne rakenduse kallal töötamist pandi paika eesmärk, et rakendus peab olema skaleeruv kahes aspektis: külastajale peab rakendus tunduma kiire ja rakendusel ei tohi olla piiranguid kliendile. Piirangute all on mõeldud siis juhust, kus süsteem ei saa kliendile mänguserverit pakkuda, sest see ei ole riistvaraliselt võimalik. Selleks on töös kasutusel joonisel 1 nähtavad Amazoni skaleerimist võimaldavad teenused ja Digital Oceani Serveri skaleeriv serveri API. Hetkel ainuke mitteskaleeriv riistvaraline osa töös on OVH.ie serverimasin, kus jookseb Java serverirakendus.

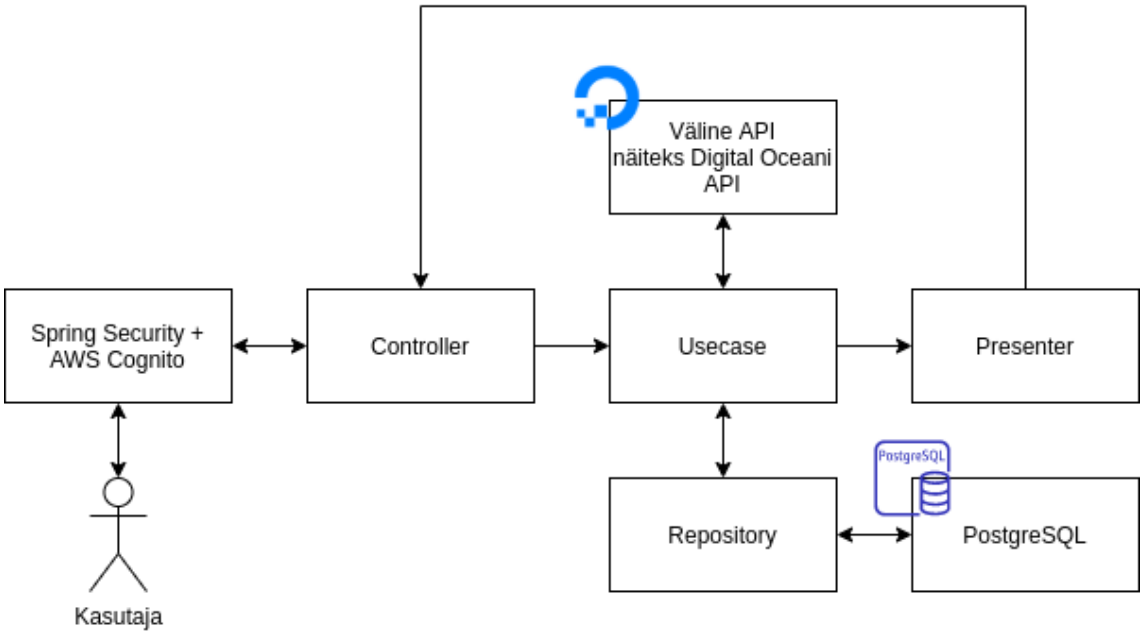


Joonis 1. Üldine rakenduse ülesehitus kasutatavate teenuste põhjal

4.1 Java rakenduse arhitektuur

Java rakendus koosneb mitmest kihist, nagu on näha joonisel 2. Esimene kiht on kasutajale avalik kontrollerite kiht, mis avalikustab erinevaid GET POST PUT UPDATE ja DELETE veebirakenduse otspunkte. Suur osa nendest otspunktidest on kaitstud Spring Security teegi annotatsioonidega. Spring Security on konfigureeritud kasutama Amazon Cognito JWT tokeneid. JWT tokenites on baasinformatsioon kasutaja kohta, mida saab Java rakendus kasutada, et kindlaks teha, kas kasutajal on õigusi antud funktsionaalsusele ligi pääsemiseks või mitte.

Otspunktidest Controlleris liigub tegevus vastavasse äriloogika osasse, projektis kutsutakse



Joonis 2. Java rakenduse loogiline ülesehitus

neid kasutusjuhtudeks (inglise keeles *use case*). Kõik rakenduse äriloogika on jagatud *use case* klasside vahele. See tagab koodi puhtuse, taaskasutuse ning kiirema kompileerimisaja. Samuti on võimalik kasutusjuhu kood lihtsamini antud Java rakendusest välja tõsta ja taaskasutada mõnes teises Java rakenduses.

Kasutusjuhu koodi klassid tegutsevad kas Java repositooriumi klassidega või mõne teise välise teenuse API-ga ning muudavad saadud infot vastavalt vajadusele. Repositooriumi klassid suhtlevad PostgreSQL-iga kus hoitakse andmeid töötavate serverite kohta, samuti hoitakse seal süsteemi tegevuste logi.

Projekti alustades oli andmebaasiks MariaDB, mis on MySQL-i edasiarendus. Otsus minna üle PostgreSQL-i tulenes sellest, et MariaDB ei toetanud tegevusi (UUID4, *Universally unique identifier version 4*) juhusliku 128-bitise numbriga, mida kasutatakse andmete märkimiseks. Näiteks kui serveri id on UUID4 formaadis, siis MariaDB andmebaasis oli selle id-ga väga raske ümber käia, sest seda salvestati baitide kujul. See tähendas, et MariaDB-s ei ole lubatud otsene UUID4 mahasalvestamine andmebaasi, vaid pidi UUID4 konverteerima baitideks. PostgreSQL-il seevastu on olemas UUID4 andmetüüp ning UUID4 saab väga lihtsalt salvestada.

Pärast kasutusjuhtumi koodi läbimist saab kontroller tagasi sõnumi vastusega, mida kasutusjuhtum saadud käskluste peale teinud oli. Tagastatud kuju on pakitud *presenter* klassi sisse. See on klass, mis sisaldab täpselt neid andmeid, mida kliendil vaja on. See väldib olukorra, kus kliendile saadetakse terve andmebaasi objekt, mis võib sisaldada andmeid,

mida klient näha ei tohiks. Mõningal juhul, kus võis tagastada terve andmebaasi objekti, polnud vaja *presenterit* valmistada.

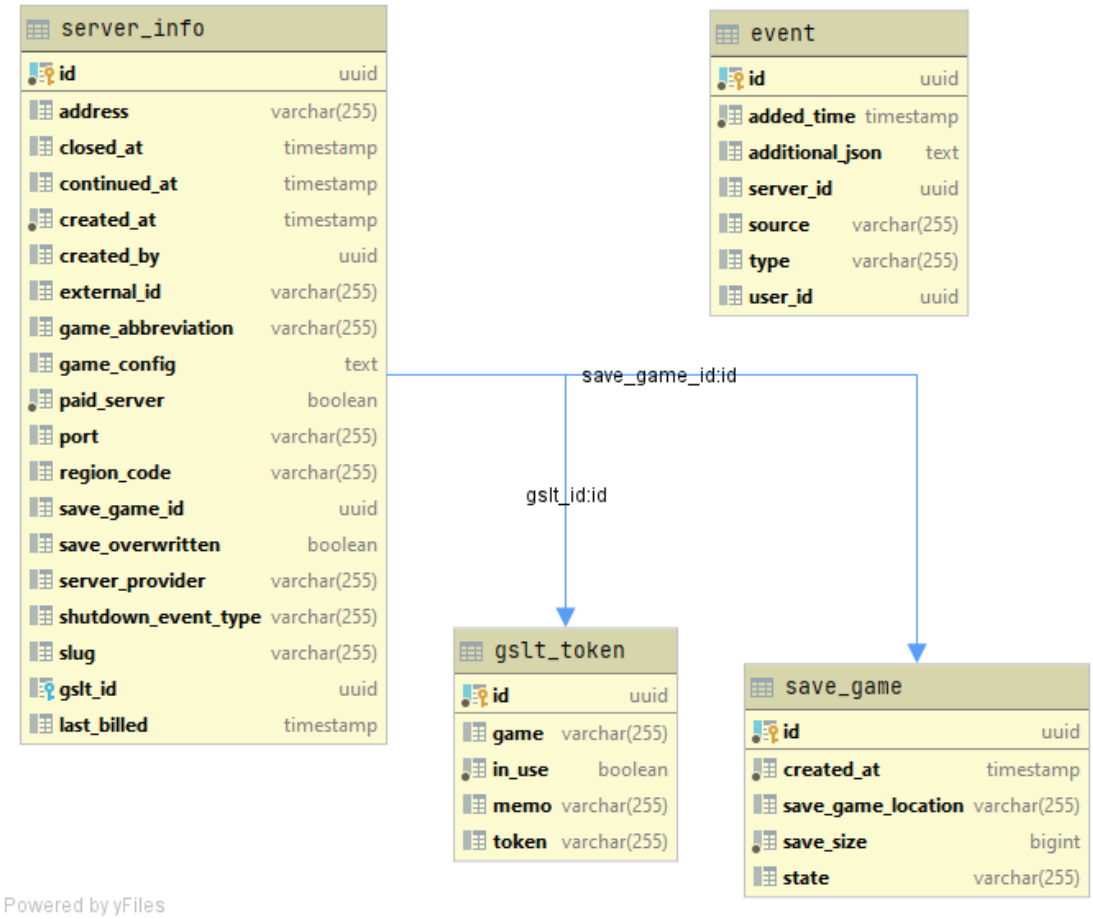
Näitena on toodud mänguserveri alustamise funktsionaalsuse klassidiagramm (vt Lisa 2). Kus on näha kuidas liiguvad käsklused läbi kontrolleri "StartServer" kasutusjuhu loogikasse, mis omakorda suhtleb "GetProvider" kasutusjuhuga, mis tagastab Digital Oceani liidestuskoodi teegi, mis oskab suhelda otseselt Digital Oceani serveritega. Samuti on näha kuidas kasutatakse *presenter* klasse, et kliendile vastav informatsioon tagastada. Seda siis selleks et võimalikult vähe ja ainult tarvilikku infot näidata.

Andmebaas ise on väga lihtne, sest enamus kasutaja autentimisloogikat asub Amazon AWS Cognito teenuses, mis tähendab, et kasutaja kohta rakendus infot ei talleta. Andmebaas hoiab ainult infot aktiivsete ja sulgenud serverite kohta. Andmebaas koosneb neljast tabelist (vt joonis 3). Esimene tabel "server info" hoiab infot käivitatud serveri kohta. Selles tabelis on olemas kõik tähtis, et süsteem oskaks kliendile serveri kohta infot näidata. Teine tabel "save game", hoiab infot mängude kohta, millel on mõned failid, mis salvestatakse peale serveri sulgemist. Kolmas tabel "gslt token" on kasutusel Steami mängude jaoks, mis nõuavad tokenit, et need serverid internetis nähtaval oleks. Neljas tabel "event" on teistest tabelitest eraldatud ja agregeerib üle kõikide rakenduste süsteemi käekäiku ja staatust. See tabel on selleks, et adminil oleks lihtsam aru saada mis serveritega reaalajas toimub ja probleemide korral lihtsamini aru saada kus midagi katki läinud on.

4.2 Front-end rakenduse arhitektuur

Veebiliidese rakendus ei ole täielikult ehitatud nullist projektis osalevate inimeste poolt. Kasutatud on Vue ja Vuetify peale ehitatud baasprojekti, mis kiirendas arendusprotsessi tunduvalt. Kasutusel olev koodipõhi sisaldas kõike vajalikku, et suunduda kohe veebilehe sisu loomisele. Veebilehepõhja kõige olulisemad komponendid olid navigatsiooniriba, sisselogimise ja väljalogimise funktsionaalsus, lehe alumine ääris ja võimalus kasutada JWT tokeneid autentimiseks.

Kasutajaliides kasutab Amazon Cognitot ja selle Javascripti teeki, et kasutajaid sisse logida ja autentida. Autenditud kasutaja saab ligipääsu funktsioonidele, mida tavakasutaja ei näe, nagu näiteks serveri käima panemise nupp ja serveri haldamiseks mõeldud leht.



Joonis 3. Java rakenduse PostgreSQL-i andmebaasidiagramm

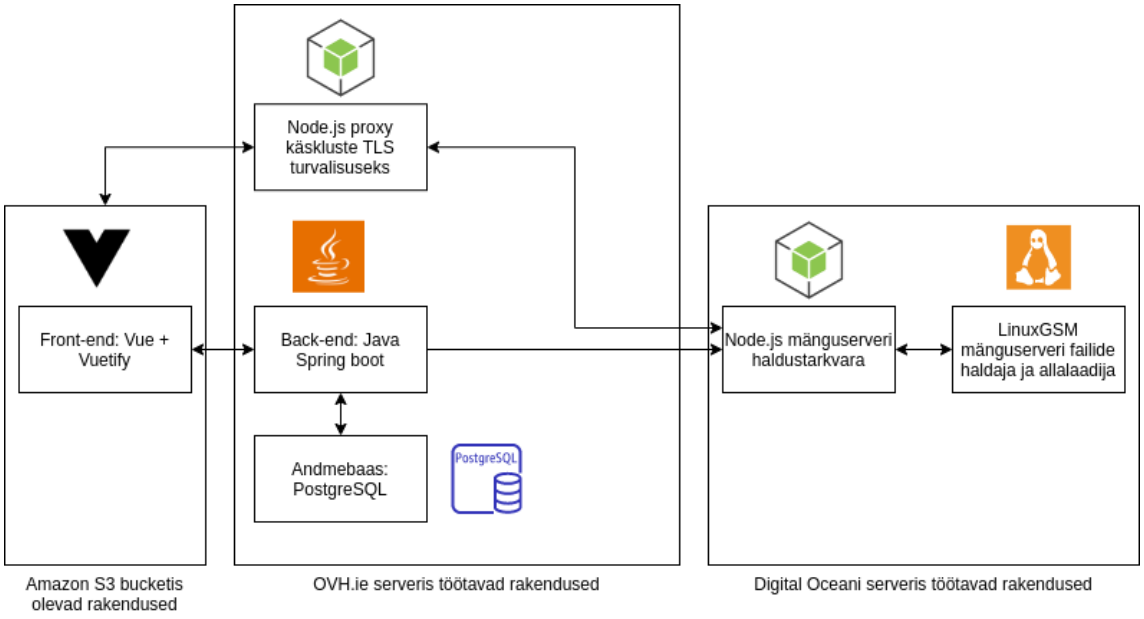
4.3 Mänguserverite ja serveri suhtlus

Kuna eesmärk on valmis teha võimalikult paljusid kliente teenindav serveri rentimise veebileht, siis on vajalik, et rentijal oleks võimalik valida serveri jõudlust. Lisaks on vaja, et tal oleks võimalik lihtsalt konfigureerida enda mänguserver enda soovide järgi. Keeruliseks teeb probleemi see, et igat mänguserverit peab konfigureerima erinevat moodi, sest mängud ei kasuta kõik ühte ja sama mängumootorit. Näiteks Source mängumootori server ei kasuta sama tüüpi konfiguratsiooni faili nagu Minecrafti mängumootori server. Ja kuna süsteem sai planeeritud nii, et tulevikus oleks võimalik käivitada ja hallata võimalikult palju erinevaid mänguservereid väikse vaevaga, siis tuleks leida või ise valmistada siduv programm.

Õnneks selline erinevaid mänge konfigureerida lubav programm on juba olemas. (LinuxGSM, *Linux game server manager*) on Linuxile arendatud vabavaraline käsurea rakendus, mille kaudu saab kiiresti ja lihtsalt käivitada, hallata ja konfigureerida üle 80 erineva mänguserveri tüübi. Igale mängule on spetsiifiliselt kirjutatud juhised, kuidas ja kuhu peaks

käima konfiguratsioon ning kuidas on kõige parem server käima panna ja sulgeda. [22].

Tänu LinuxGSM-le sai väga palju aega säästetud sellega, et ei pidanud kirjutama enda konfiguratsioonirakendust. Lisaks annab LinuxGSM võimaluse pakkuda rohkem valikuid ka tulevastele kliendile, kellel on erisoov mõne mängu osas, sest mänguvalik LinuxGSM-il on suur ja laieneb iga välja tuleva mänguga.



Joonis 4. valminud rakendused ja nende omavaheline seotus

Kuna otsest suhtlust LinxGSM-i ja Java rakenduse vahel ei ole võimlik teha, siis selleks oli vajalik lisada üks vaherakendus, mis suudab võtta vastu sõnumid Java rakendusest ja need edasi anda LinuxGSM-le. See rakendus asub koos LinuxGSM-iga mänguserverit jooksutavas masinas. Lisaks annab see lisaprogramm käsklusi ka operatsioonisüsteemile. Näiteks juhul kui server aktiivsuse puudumisel ise kinni ei lähe, siis pärast mõnda aega antakse läbi vaherakenduse sulgemiskäsk serveri masinale. Joonisel 4 on näha erinevate kasutusel olevate rakenduste suhtlusteekonnad.

Lisaks on olemas veel väike proxy rakendus kirjutatud Node.js-is, mis suunab veebiliidesest käsklusi mänguserveri konsooli. Seda selleks, et veebilehitseja ja Digital Oceani serveris jooksva serverihaldustarkvara vaheline ühendus oleks kaitstud TLS sertifikaadi krüpteeringuga.

4.4 Serveripakkujate valik

Kompileeritud HTML ja Javascriptiga valminud kasutajaliides on üleval Amazoni S3 pilves. Lisaks on kasutuses Amazoni CloudFront, mis oskab suunata veebiliikluse just

sellele S3 failihoidlale, kus hoiustatakse front-end faile. CloudFronti kasutatakse selleks, et hajutada võimalikku suuremat liiklust, mis väldib veebilehe mahaminemise koormuse tõttu. Halb on see, et CloudFront muudab failid kättesaadavaks Amazoni lingi kaudu, mis ei ole kasutajasõbralik. Selleks, et need lingid oleks ilusamad, on kasutuses Amazoni Route53 teenus, mis on domeeni suunamiseks CloudFronti teenusele. Kaks domeeninime, mis sai projekti käigus registreeritud, on Gamehostz.com ja Gamehost.ee.

Java back-end teenus jookseb OVH.ie serverirentijalt renditud virtuaalses privaatserveris. Samuti jookseb selles serveris ka Springi poolt kasutatav PostgreSQL server, kus hoitakse serverite kohta aktiivset infot. Selleks et Java rakendus maailmale nähtav oleks, on kasutusel Nginx, mis suunab sissetuleva liikluse Java rakendusse. OVH VPS sai just eelkõige valitud sellepärast, see oli kõige odavam. Kui kõiki neid teenuseid, mis jooksevad hetkel OVH VPS-is, jooksutada Amazonis, siis oleksid kulud olnud kolmekordsed. Arenduse ja testimise faasis sobib ka kõige odavam variant.

Mänguservereid on vaja kuskil jooksutada, kas siis oma riistvara peal või rentida suurem server mõnelt teenusepakkujalt, nagu on OVH.ie. Suurte serveritega on probleem jällegi see, et serverit ei ole võimalik sajaprotsendiliselt igal kellaajal kasutada. Näiteks öösiti on vähem kliente kui päeval ja päevasel ajal võib tekkida puudus serveri võimsusest, kui kliente on liiga palju. See tekitab väga suure skaleerimise probleemi, mille tegelikult on lahendanud eelmainitud olemasolevate lahenduste all Agones, kui kasutada Kubernetest ja mõnd pilveteenusepakkujat. Kuna Agones on aga mõeldud pigem mängutootjatele, mitte serveri teenuse pakkujatele, siis oli vaja otsida muu lahendus.

Selleks lahenduseks said lühiajalised serveripakkujad, nagu Digital Ocean, Vultr või Eesti enda sarnase teenuse pakkuja Pilw.io. Serverid, mida nad pakuvad, on virtuaalsed privaatsed serverid (VPS), nagu pakub ka OVH.ie. Ainult et nende teenust eristab hinnastamine. Nad küsivad raha ainult aja eest, millal server tegelikult töötas. Ehk kui server on kuu aja jooksul üleval olnud ainult 24 tundi, siis kuu lõpus tehakse arve 24 tunni eest. Eelnevast uurimisest selgus, et nendest kõigist parima API-ga oli Digital Ocean, seega otsustati selle teenusepakkujaga esimene integratsioon teha. Tulevikuplaan on liita Vultr kui ka Pilw.io, et pakkuda erinevates regioonides parema ühendusega servereid.

4.5 Skaleeritavus

Alustades kasutajale nähtavast veebiliidesest, siis veebiliides on kompileeritud failidena üles laetud Amazoni S3 bucketisse, sealt on see nähtav Amazoni CloudFront teenusele. CloudFront tagab selle, et suure koormuse korral ei läheks veebileht maha. CloudFront testib järjepidevalt internetiühendust ja jõudlust kliendi ja Amazoni serverite vahel. Tänu

sellele oskab teenus suunata päringud serverisse, kus on võimalik tagada parim kogemus kasutajale[23].

Mänguserverid jooksevad selle töö kirjutamise hetkel Digital Oceani virtualiseeritud serverites, mida nad kutsuvad *dropletiteks*. Digital Oceani API on otseselt integreeritud Java rakendusega. Kui kasutaja võtab endale süsteemist mänguserveri, saadetakse sõnum Digital Oceanisse, et käivitada vastava võimsusega server. Serveris jookseb Linux koos meie enda mänguserveri manageerimis- ja sidetarkvaraga ning LinuxGSM-iga. Siinkohal peab ära mainima, et Digital Oceani kui teenusepakkuja saab iga hetk välja vahetada mõne teise sarnase ettevõtte vastu. Digital Ocean seab küll piiranguid, et korraga võib käimas olla kümme *dropleti*, kuid e-maili teel suheldes saime selle piirangu enda kontodelt maha võetud. Faktiliselt ei ole teada, kui suurt koormust Digital Oceani serverid kannatavad, seega ei saa siinkohal väita, et skaleeritavus oleks lõpmatu. Kui tulevikus peaks Digital Oceaniga piir ette tulema, siis on võimalus integreerida juurde teisi sarnaseid teenusepakkujaid.

Rakenduse hetkene kitsaskoht skaleerimise osas on Java serveri ja PostgreSQL andmebaasi hostimine. Nimelt VPS-iga ei ole võimalik neid kahte rakendust skaleerida. Plaanis on suuremate koormuste korral Java rakendus kolida Kubernetese peale, mis suudaks automaatselt uusi Java servereid käima panna vastavalt vajadusele. Samuti oleks sel juhul vajalik PostgreSQL andmebaasi liigutamine mõne pilveteenusepakkuja juurde, nagu näiteks Amazoni pakutavad lahendused.

4.6 CI/CD arhitektuur

Java rakendusele on tehtud Github Actions script sellisena, et iga koodimuudatuse peale läheb tööle Java koodi kompileerimine veebirakenduseks. Seejärel selle rakenduse vastu pannakse käima unit-testid ja integratsioonitestid. Kui testid lähevad läbi, siis laetakse kompileeritud kood Dockeri repositooriumisse. Seepeale läheb käima järgmine osa tööst OVH.ie VPS-i serveris. Tõmmatakse alla kompileeritud kood Dockeri repositooriumist ja käivitatakse see Dockeri konteineris. Nii on tagatud, et uus kood jõuab peale igat ülevaadatud muudatust testkeskkonda.

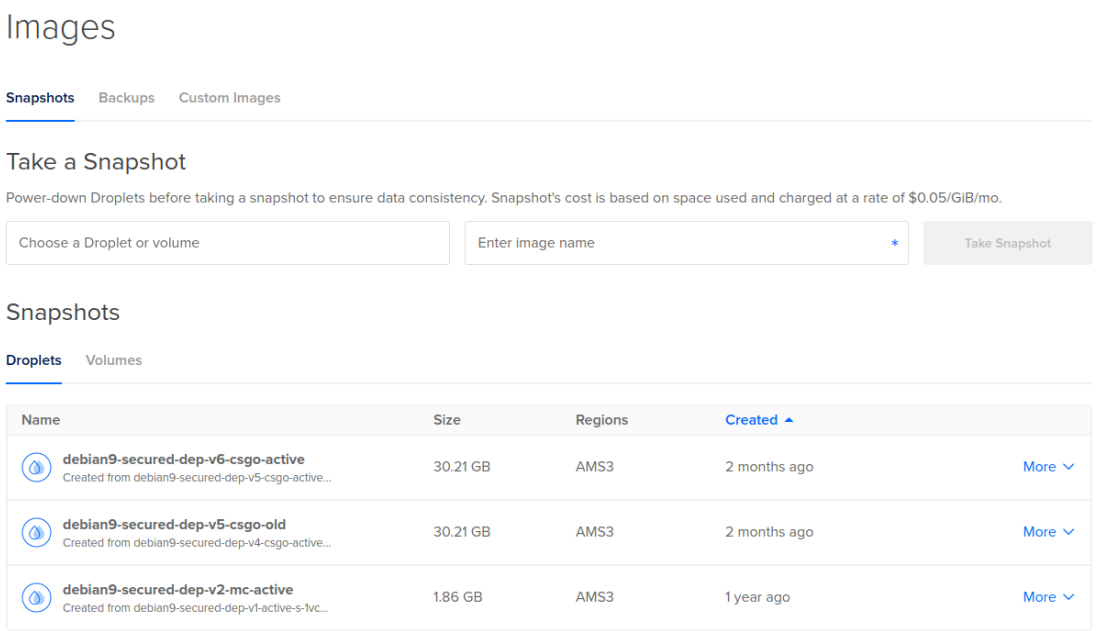
Vue kasutajaliidesele on kirjutatud Github Actions script, mis koodimuudatuse peale kompileerib Vue koodi tavaliseks HTMl-iks ja Javascriptiks. Seepeale laetakse kompileeritud failid Amazoni S3 pilve. Antud asukoht pilves on tehtud avalikuks, et terve maailm saaks sellele ligi. Samuti pärast failide üleslaadimist peab Amazoni CloudFront saatma sõnumi kõikidele kunagistele veebilehe külastajatele, et uus versioon saidist on üleval ja on vaja uuendada veebilehitseja cache.

Serveri manageerimis- ja sidetarkvarale on samuti kirjutatud GitHub Actionsis script, mis iga koodimuutuse peale kompileerib koodi ning laeb antud koodi Amazoni S3 pilve üles. See S3 pilve asukoht pole kõigile avalik, vaid ainult Digital Oceani serverile, millel on võti nende failide allatõmbamiseks. Pärast failide üleslaadimist saab mänguserveri VPS need failid alla tõmmata ja käima panna.

4.7 Rakenduse optimeerimine

Mõne suurema Source mängumootoriga mängu puhul läheb server kaua käima, sest virtuaalne server peab tõmbama mänguserverifailid alla läbi Steami allalaadimispeeglite. Nagu näiteks CS:GO serveri suurus on 24GB, siis seda serverit tõmbab alla 100mbit/s kiirusega 34 minutit. Mõõdetud allalaadimise kiirus jääb kuskile 100mbit/s juurde, kuid kõigub vastavalt Steami enda koormusele. Siit tekib ka probleem, et kui kliendidle tahta anda server võimalikult kiirelt, siis pool tundi serveri ootamiseks ei ole aktsepteeritav.

Probleemi leevendamiseks prooviti vahetada viruaalse privaatserveri füüsilist asukohta. Pannes server käima Prantsusmaa Digital Oceani asukoha asemel Amsterdami regioonis, mis on lähemal Steami Rootsi serverile, saadi allalaadimise kiiruseks järjepideval erinevatel aegadel testimisel 120mbit/s, ja serveri allalaadimise ajaks saadi 28 minutit. Mis ei ole siiski piisavalt hea, et kasutajale kiiret teenuse kogemust pakkuda.



Joonis 5. Mänguserverite kettapildid Digital Oceanis

Digital Oceanil on võimalik teha ka käimasolevast serverist kõvakettapilt. See tähendab, et mänguserveri faile ei pea iga kord läbi Steami serverite alla tõmbama. VPS-i saab

käima panna kettapildist, mis kopeeritakse Digital Oceani infrastruktuuri siseselt, näidatud joonisel 5. Antud meetodiga läheb CS:GO server käima, nupuvajutusest mängimiseni, alla kümne minuti. Aeg kulub kettapildi kopeerimiseks ja käivitamiseks. Digital Ocean võtab iga salvestatud kettapildi eest 0.041 eurot gigabaidi eest kuus. Mis teeb CS:GO serveri kettapildi talletamise hinnaks, mille lõppsuuruseks koos Linuxi endaga on 30GB, 30 * 0.41 = 12.30 eurot kuus. See hind on võrdlemisi väike, kui vaadata, mis ajakulu kliendile ära hoitakse.

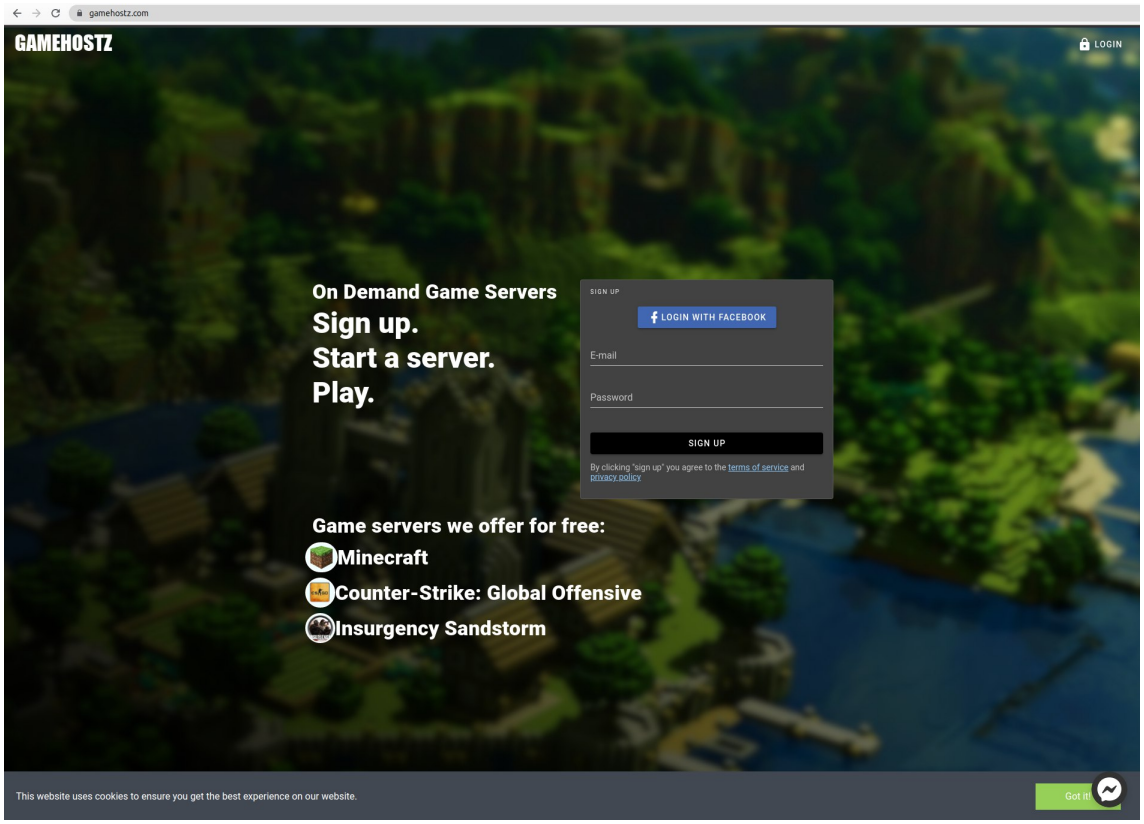
Kui tahta serveri käivitamise aega CS:GO serveril veel väiksemaks saada, võiks kasutada puhvrit käimasolevate serveritega, mida kasutaja saaks koheselt võtta. Niipea kui server ära võetakse, pannakse uus server valmis järgmisele kliendile. Selline metoodika töötaks mängudes, millel ei ole vaja salvestada kasutaja muudatusi serverisse. Probleem tekib näiteks Minecrafti serveriga, kus igal mängijal on omaenda maailm, ja mida ei saaks puhverdatud serverina klientidele anda. Probleem tekiks ka teiste mänguserveritega, sest kasutaja saab valida, kui palju inimesi hakkab serveris mängima. Kasutades puhverdatud serverit, ei saaks inimene seda valida ja maksaks liiga suure serveri eest liiga palju või halvemal juhul jääks serveris ruumist puudu.

Tulevikuplaan on eelmainitud probleemid ette võtta ja leida lahendused. Aga hetkel on ajutise lahendusena kasutusel Digital Oceani kõvakettatõmmised, et serveri ülesminemise aega väiksemaks saada. Väiksemaltel mängudel, nagu näiteks Minecraft, seda vaja pole ja server läheb alla 2 minuti käima.

5. Rakenduse kasutamine

5.1 Esileht ja registreerumine

Hetkel asub veebileht aadressil gamehostz.com[1]. Esimene leht, mida antud lingile minnes nähakse, on väike tutvustusrakenduse kohta ja registreerumise vorm, mida on näha Joonisel 6. Et mänguserverit rentida, on vaja teha endale kasutaja, sisestades registreerimisvormile enda e-mail ja parool. Sellepeale saadetakse kasutaja e-mailile kinnituskiri. Kirjas on link, mis suunab tagasi veebilehele ja antud e-mail seotakse kasutajaga. Pärast lingile vajutamist saab kohe jätkata rakenduse kasutamist ilma, et peaks lisaks eraldi sisse logima.



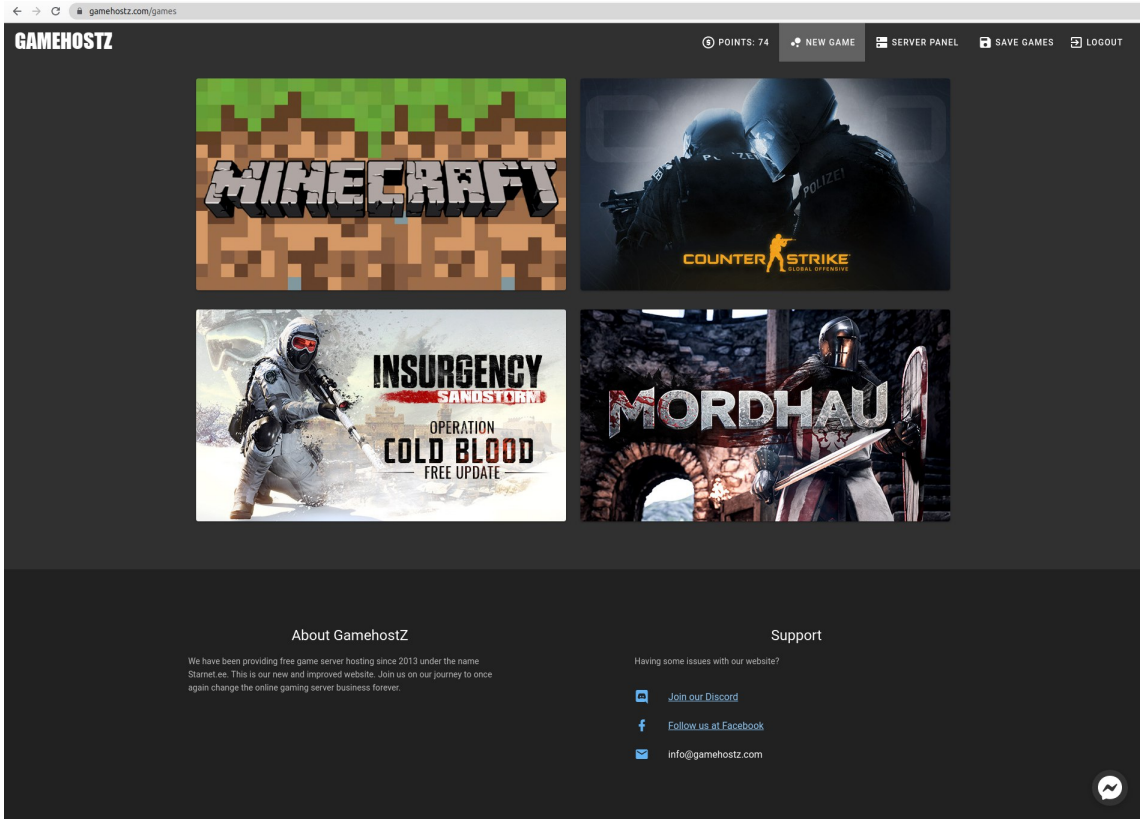
Joonis 6. Valminud rakenduse, Gamehostz.com esileht

Kui aga kliendil on juba kasutaja olemas, siis saab ta enda kasutajaga sisse logida sisselogimise lehel. Nupp sellele lehele on Joonis 6 paremal üleval nurgas. Sisselogimise lehel asub ka nupp, kus saab vahetada kasutaja parooli juhul, kui see on ununenud.

[1] https://gamehostz.com

5.2 Mängu valimine

Pärast kasutaja sisselogimist suunatakse ta mänguvaliku juurde, mis on näha Joonisel 7. Hetkel on valikus neli mängu ja reaalselt töötab ainult kaks, CS:GO ja Minecraft. Mängu saab valida pildile peale klikkides. Lisaks pildi peal hiirega hõljudes näeb teksti, mis ütleb "alusta server". Kuvatud nupu pildid tulevad serveris asuvast konfiguratsioonifailist, kuhu on sisestatud pildi URL.

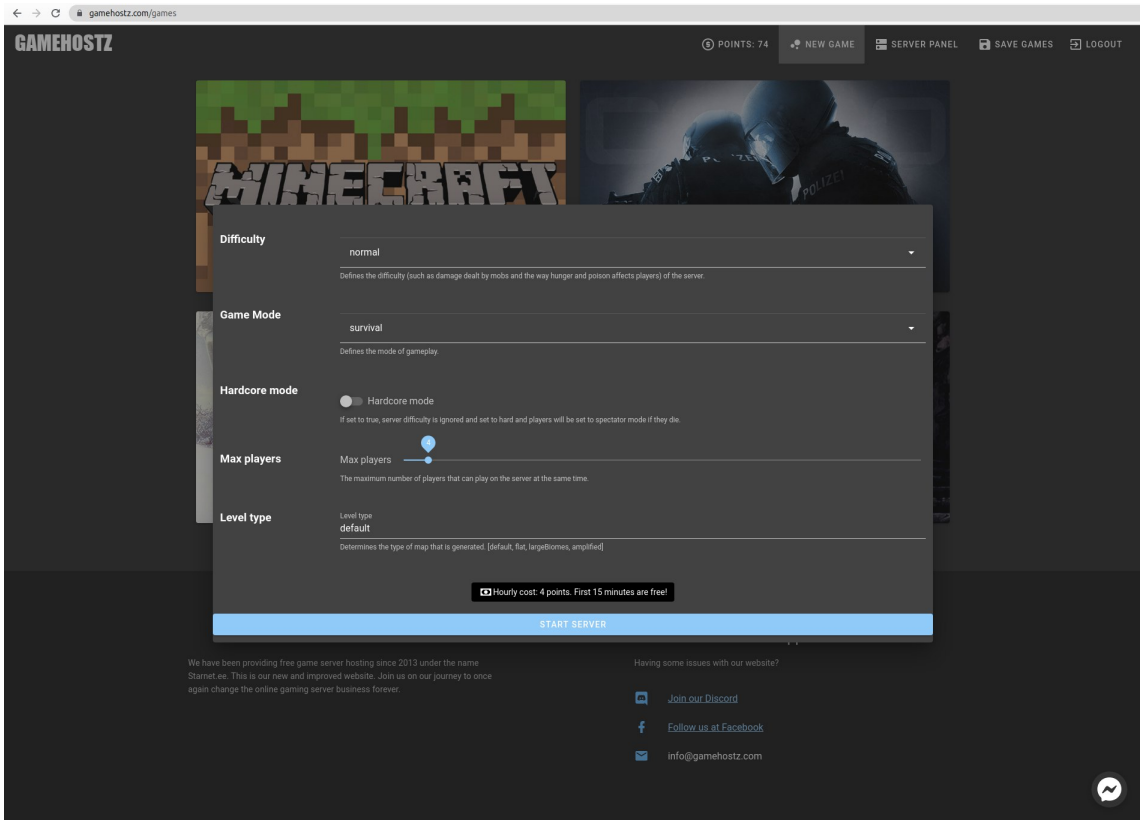


Joonis 7. Mänguserveri valimise leht

Mänge saab lisada sellele lehele läbi süsteemi konfiguratsiooni faili JSON formaadis, mis sisaldab kohustuslikke väljasid, nagu näiteks mängu nimi, mängu failide asukohad, maksimummängijate arv ja miinimummängijate arv.

5.3 Mänguserveri konfigureerimine

Pärast mängu pildile vajutamist avaneb serveri konfigureerimise aken Joonisel 8, millel saab muuta erinevaid serveri parameetreid, nagu näiteks mängu raskusaste, mängu tüüp, mängijate arv serveris ja palju muud. Kõik need seaded on manuaalselt ühendatud mängu serveri seadefailidega, kasutades süsteemi enda konfiguratsiooni JSON faili, mis on nähtaval antud töö Lisa 1 all. Konfiguratsioonifail sisaldab ka vaikeväärtusi kõikidele seadetele, mida näidatakse modaali avamisel.



Joonis 8. Mänguserveri konfigureerimise aken

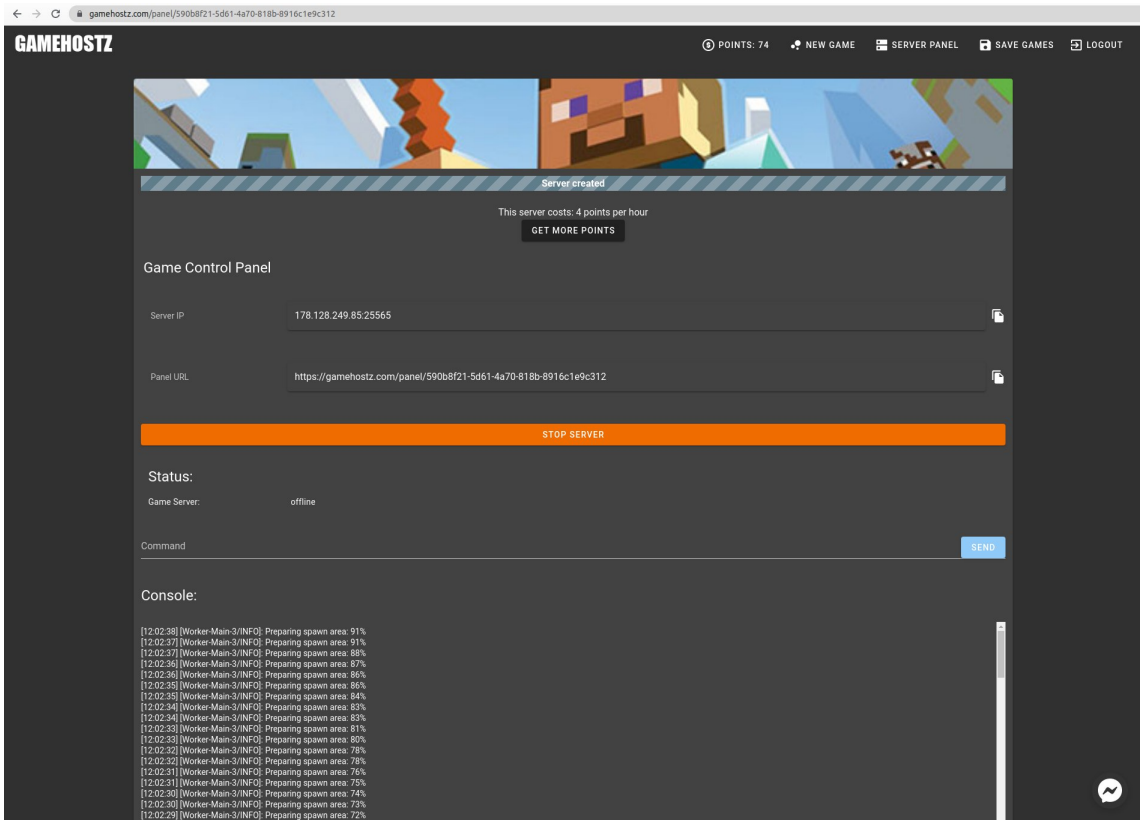
Serveri käimapanemise hetkel kontrollitakse kasutaja poolt sisestatud seaded serveri enda konfiguratsiooni vastu, et vältida mänguserveri kokkujooksmist ja pahatahtlikke tegevusi kasutajate poolt.

5.4 Mänguserveri paneel

Pärast konfiguratsiooni kinnitamist ja serveri käivitamist algab mitu protsessi. Esimene süsteemi ülesanne on valideerida kasutaja sisendid, seejärel saata Digital Oceanisse läbi nende API sõnum, et panna käima server. Seejärel jääb mängupaneel ootama Digital Oceani virtuaalse serveri IP-d, mida kasutajale kuvada. Joonisel 9 on näha paneeli, millega saab kontrollida mänguserverit. Paneelil on Serveri IP, Paneeli URL, nupp, millega server kinni panna, ning konsool, millel on otseühendus mänguserveriga. Paneeli üleval osas on näha ka laadimisriba, mis annab kasutajale teada, kui kaua läheb aega serveri ülesseadmiseks.

Üks asi, mida Joonisel 9 ei ole näha, on serveri staatus. Staatus ilmub siis, kui süsteem saab kätte info serveri kohta, et server on täielikult käivitunud. Sellelt saab näha serveris olevate inimeste nimesid ja muid serverile omaseid seadeid.

Pärast mänguserveri käimaminemist saab klient serveris mängida niikaua, kui ta tahab.



Joonis 9. Mänguserveri haldamise paneel

Ning peale serverist lahkumist läheb käima viieminutiline taimer, mis kontrollib, kas server on tühi või mitte. Kui serverisse viie minuti jooksul kedagi sisse ei tule, sulgeb taimer serveri nii süsteemis kui ka Digital Oceanis.

5.5 Kuluefektiivsus, kasutades töös valmistatud rakendust

Minecrafti foorumist leitud küsitluse põhjal saab väita, et üks tavaline Minecrafti mängija mängib keskmiselt 4 tundi päevas[24]. Selle põhjal saab arvutada serveri kulukuse valemiga: 4(tundi)*30(päeva)*0,0058(eurot) = 0.696 eurot kuus. Muidugi tekib küsimus, et server on selleks, et mängida sõpradega, mitte üksi. Arvestades, et inimesed mängivad tavaliselt ainult päevasel ajal, siis peaks tundide arvuks võtma 12 tundi päevas, saades serveri kulukuseks 12(tundi)*30(päeva)*0,0058(eurot) = 2.088 eurot kuus. Võrreldes seda ühe tavalise serveripakkuja baashinnaga, nagu näiteks **Mcprohosting**[2] pakutava 1GB mäluga Minecrafti mänguserveriga, mis maksab 6.61 eurot kuus, on näha, et töös arendatud süsteem on kliendile 6.61 - 2.088 = 4.522 eurot soodsam kui tavalise serveri rentimine. Seda muidugi eeldusel, et kasutustunnid jäävad alla 12 tunni ööpäevas.

[2]https://mcprohosting.com/order

Tabel 2. *Võrdlus valminud rakenduse ja tavalise teenusepakkuja vahel*

Nimi	**Hind kuus**	**Hind tunnis**	**Serveri käivitumisaeg**
Valminud rakendus	3.19 € kuus	0.009 € tunnis	181 sekundit
Mcprohosting	6.61 € kuus	0.009 € tunnis	65 sekundit
Apex Minecraft Hosting	4,93 € kuus	0.007 € tunnis	103 sekundit

Kuna veebilehe ülevalhoidmine koos Amazoni teenustega ja Java serveriga on ka lisakulu, siis tuleks ka see serveri hinna sisse arvutada. Hetkeseisuga saadab Amazon arveid ainult veebidomeeni ülevalhoidmise eest, mis on 2 eurot kuus - teisi teenuseid ei kasutata piisavalt ja need on Amazoni poolt tasuta. Enamus koormust on Java serveril, mis jookseb keskmise suurusega OVH.ie VPS-i peal ning mille hinnaks on 20 eurot kuus, mis teeks hetkeseisuga kogu süsteemi ülevalhoidmise hinnaks 22 eurot kuus. See hind tuleks ära jagada kõikide klientide peale, kes antud teenust kasutavad. Võttes oletuslikuks olukorraks, kus on 20 klienti, kes rendivad serverit 12-ks tunniks ööpäevas, tuleks igale kliendile serveri kulu 22/20 + 2.088 = 3.188, mis on 6.61 -3.188 = 3.422 eurot soodsam kui tavalise serveri rentimine.

Kuluefektiivsus väljenduks ka serveriparkides, kus antud süsteem oma mänguservereid jooksutab. Kuna server ei tööta 24 tundi ööpäevas, siis võib väita ka, et voolutarbimine serveripargis on väiksem ja seeläbi paiskub läbi elektritootmise vähem CO2 gaase atmosfääri.

6. Kokkuvõte

Lõputöö eesmärk oli luua rakendus, mis suudab vastavalt kliendi soovile käima panna Minecrafti mänguserveri ning hoida seda serverit üleval niikaua, kuni klient seda kasutab. Seeläbi kokku hoida serveri ülevalhoidmiseks kuluvat raha ja ka kaudselt serveripargis kasutatavat elektrit. Erinevalt teistest sarnastest teenusepakkujatest suudab arendatud süsteem olla odavam, kuna server, mida klient kasutab, ei tööta ööpäevaringselt. Siinkohal peab välja tooma olukorra, kus klient ei lase serveril sulgeda, mille peale teenuse hind sarnaneb tavalistele teenusepakkujatele.

Töö raames vaadeldi ja kasutati sarnaseid olemasolevaid mänguservereid pakkuvaid teenuseid. Leitud serveripakkujaid pakuvad servereid ainult kuu ajaliseks perioodiks ja enneaegse serveri sulgemise eest raha tagasi ei kanta. Leidus ka paar erandit, mis töötavad fikseeritud perioodiks servereid välja rentides. Need rakendused sarnanevad töös valmistatule, kuid ei ole täpselt sama ideega ega säästa nii suures koguses kliendile raha.

Suur rõhk pandi ka arendusmugavuse ja -efektiivsuse peale. Uuriti erinevate tarkvaraliste võimaluste kohta ning otsustati kasutada Amazoni AWS ja Digital Oceani ettevõtete teenuseid, mis lihtsustasid arendamist. Nagu näiteks kasutaja autentimiseks kasutati Amazon Cognitot, mis hoidis kokku tohutul hulgal aega kasutaja sisselogimise ja registreerimise funktsionaalsuse pealt. Eelmainitud teenuste ümber ehitati Java ja JavaScripti programmeerimiskeeltes serverihaldusrakenduste kogumik, mis täidavad eesmärgis kirjeldatut. Samuti valmistati, kasutades Github Actionsit, automatiseeritud koodi kasutajateni viiv protsess. See tähendab, et pärast iga koodimuudatust jõuavad muudatused koheselt või väikse viitega klientideni.

Tulemusena valmis tasuta lühiajalisi konfigureeritavaid Minecrafti mänguservereid pakkuv programmide kogumik, mis asub hetkel avalikult lehel Gamehostz.com. Veebileht ehitati Vue raamistiku abil ning sellel lehel saab käima panna konfigureeritavaid mänguservereid mängule Minecraft. Lisaks saab nendele serveritele kasutaja läbi kasutajaliidese käsklusi saata nagu näiteks serveri sulgemine või mõned muud mängu spetsiifilised käsklused. Käsklused liiguvad töö raames programmeeritud Java API-sse, mis suhtleb andmebaasiga ja teiste väliste teenustega.

Teiste väliste teenuste all, millega Java API suhtleb, osutus kõige tähtsamaks Digital Oceani servereid pakkuv veebiliides, mille kaudu pannakse käima virtuaalseid privaatservereid, kus töötavad mänguserverid ja serverite manageerimise vahekiht. Antud serveripakkuja valiti uurimise tagajärjel. Vaadeldi erinevaid serveri pakkujaid ja toodi välja plussid ning miinused finantsilise, võimsuse kui ka arendaja mugavuse võtmes. Lõpuks jäi peale töö kirjutaja arvates parima hinna ja kvaliteedi suhtega teenusepakkuja.

Digital Oceani privaatserverites töötab spetsiaalselt kodeeritud vahekiht, mis haldab mänguservereid ja võtab vastu käsklusi, mis saadetakse kasutajaliidesest, kui ka Java rakendusest endast. Java rakendusse arendati funktsionaalsus, mis suudab mänguservereid aktiivsuse puudumisel kinni panna. Seeläbi saab valminud rakendus säästa raha kliendile ja veebilehe omanikule. Lisaks hoitakse nii kokku ka elektri pealt Digital Oceani serveriparkides, mis omakorda vähendab CO2 väljalaset atmosfääri. Valminud töö on hobiprojekt ja valmis kaheaastase arenduse tulemusena. Avaliku versiooni koodist leiab BitBucketi repositooriumist [25].

Kasutatud kirjandus

[1] Gametracker.com. *Gametracker Stats*. URL: hhttps://www.gametracker.com (vaadatud 20.09.2020).

[2] OVH. *The performance of VPS, the availability of the cloud*. URL: https://www.ovhcloud.com/en-ie/vps/vps-cloud/ (vaadatud 20.09.2020).

[3] Aternos. *Minecraft servers.Free. Forever.* URL: https://aternos.org/:en/ (vaadatud 20.09.2020).

[4] Freegamehosting. *Because gaming is free*. URL: https://www.freegamehosting.eu/#mc (vaadatud 20.09.2020).

[5] Minecraft Wiki. *Server/Requirements/Dedicated*. URL: https://minecraft.gamepedia.com/Server/Requirements/Dedicated (vaadatud 03.10.2020).

[6] Alexandru Andrei. *Create Your Own Dedicated Server for Playing Counter Strike: Global Offensive (CS:GO)*. URL: https://medium.com/@Alibaba_Cloud/create-your-own-dedicated-server-for-playing-counter-strike-global-offensive-cs-go-5914ff8d436a (vaadatud 03.10.2020).

[7] Agones. *Announcing Agones 1.0.0*. September 2019. URL: https://agones.dev/site/blog/2019/09/25/announcing-agones-1.0.0/ (vaadatud 20.09.2020).

[8] Ovhcloud. *Innovate faster with cloud automation tools*. URL: https://www.ovhcloud.com/en-ie/public-cloud/orchestration/ (vaadatud 03.10.2020).

[9] Ovhcloud. *High-performance cloud solutions for the best possible prices*. URL: https://www.ovhcloud.com/en-ie/public-cloud/prices/#568 (vaadatud 03.10.2020).

[10] Pilw.io. *Pilw.io Meist*. URL: https://pilw.io/firmast/meist/ (vaadatud 27.01.2021).

[11] Pilw.io. *Pricing*. URL: https://pilw.io/en/price/ (vaadatud 27.01.2021).

[12] DigitalOcean Wikipedia. *DigitalOcean Wikipedia leht*. URL: https://en.wikipedia.org/wiki/DigitalOcean (vaadatud 27.01.2021).

[13] DigitalOcean. *Simple, predictable pricing*. URL: https://www.digitalocean.com/pricing/ (vaadatud 27.01.2021).

[14] Vultr. *Welcome To Vultr*. URL: https://www.vultr.com/company/about-us/ (vaadatud 27.01.2021).

[15] Vultr. *Powerful Compute Instances*. URL: https://www.vultr.com/products/cloud-compute/#pricing (vaadatud 27.01.2021).

[16] Michael Rasmussen. *Using Text Blocks in Java 13*. URL: https://www.jrebel.com/blog/using-text-blocks-in-java-13 (vaadatud 21.09.2020).

[17] Baeldung. *A Comparison Between Spring and Spring Boot*. URL: https://www.baeldung.com/spring-vs-spring-boot (vaadatud 21.09.2020).

[18] Wikipedia. *TypeScript on vabavaraline programmeerimiskeel, mida arendab Microsoft*. URL: https://et.wikipedia.org/wiki/TypeScript (vaadatud 22.09.2020).

[19] Vuejs. *What is Vue.js?* URL: https://vuejs.org/v2/guide/ (vaadatud 22.09.2020).

[20] Vuetifyjs. *What's the difference?* URL: https://vuetifyjs.com/en/introduction/why-vuetify/ (vaadatud 22.09.2020).

[21] GitHub. *Automate your workflow from idea to production*. URL: https://github.com/features/actions (vaadatud 22.09.2020).

[22] Phil P. *How to Deploy a Minecraft Server in Linux*. URL: https://support.us.ovhcloud.com/hc/en-us/articles/360002499679-How-to-Deploy-a-Minecraft-Server-in-Linux (vaadatud 21.09.2020).

[23] CloudFront. *Network optimizations for optimal performance*. URL: https://aws.amazon.com/cloudfront/features/?nc=sn&loc=2 (vaadatud 23.09.2020).

[24] PolarBjorn. *How long do you play minecraft every day?* URL: https://www.minecraftforum.net/forums/minecraft-java-edition/survival-mode/283886-how-long-do-you-play-minecraft-every-day (vaadatud 19.02.2021).

[25] Karl-Martin Miil. *Copy of Gamehostz.com repository*. URL: https://bitbucket.org/karlmartinm/gamehostz_thesis/src/master/ (vaadatud 30.04.2021).

Lisad

Lisa 1 - Mängu konfiguratsioonifail serveris


```
{
  "name": "Minecraft",
  "abbreviation": "mc",
  "link": "https://www.minecraft.net",
  "thumbnail": "http://steam.cryotank.net/wp-content/gallery/minecraft/
  "primaryPort": 25565,
  "secondsToStart": 180,
  "ports": [
    25565
  ],
  "folder": "minecraft",
  "lgsmName": "mcserver",
  "supportsSave": true,
  "config": [
    {
      "name": "main-settings",
      "file": "server.properties",
      "directory": "/",
      "format": "mc_server_properties_format",
      "ram": [
        {
          "maxPlayers": 4,
          "ramAmount": "s-1vcpu-1gb"
        },
        {
          "maxPlayers": 8,
          "ramAmount": "s-1vcpu-2gb"
        },
        {
```

```
      "maxPlayers": 16,
      "ramAmount": "s-1vcpu-3gb"
    },
    {
      "maxPlayers": 32,
      "ramAmount": "s-2vcpu-4gb"
    },
    {
      "maxPlayers": 64,
      "ramAmount": "s-4vcpu-8gb"
    }
  ],
  "settings": [
    {
      "name": "difficulty",
      "display": "Difficulty",
      "description": "Defines the difficulty (such as damage dealt
      "link": "https://minecraft.gamepedia.com/Server.properties#Ja
      "type": "enum",
      "allowedValues": [
        "peaceful",
        "easy",
        "normal",
        "hard"
      ],
      "defaultValue": "normal"
    },
    {
      "name": "gamemode",
      "display": "Game Mode",
      "description": "Defines the mode of gameplay.",
      "link": "https://minecraft.gamepedia.com/Server.properties#Ja
      "type": "enum",
      "allowedValues": [
        "survival",
        "creative",
        "adventure",
        "spectator"
      ],
```

```
        "defaultValue": "survival"
      },
      {
        "name": "hardcore",
        "display": "Hardcore mode",
        "description": "If set to true, server difficulty is ignored
        "link": "https://minecraft.gamepedia.com/Server.properties#Ja
        "type": "boolean",
        "defaultValue": false
      },
      {
        "name": "max-players",
        "display": "Max players",
        "description": "The maximum number of players that can play o
        "link": "https://minecraft.gamepedia.com/Server.properties#Ja
        "type": "range",
        "minValue": 1,
        "maxValue": 64,
        "defaultValue": 4
      },
      {
        "name": "level-type",
        "display": "Level type",
        "description": "Determines the type of map that is generated.
        "link": "https://minecraft.gamepedia.com/Server.properties#Ja
        "type": "string",
        "defaultValue": "default"
      }
    ]
  }
 ]
}
```

Lisa 2 - Serveri käivitamise kasutusjuhu klassidiagramm Java rakenduses