TARTU ÜLIKOOL
Arvutiteaduse instituut
Informaatika õppekava

Joonas Halapuu

Eestikeelse veebipõhise Linuxi käsurea õpikeskkonna loomine

Bakalaureusetöö (9 EAP)

Juhendaja: Alo Peets

Tartu 2022

Eestikeelse veebipõhise Linuxi käsurea õpikeskkonna loomine

Lühikokkuvõte:

Tartu Ülikooli kursuste „Operatsioonisüsteemid“ ning „Andmeturve“ esimeste praktikumide käigus õpivad tudengid Linuxi käsurea kasutamist sooritades seal erinevaid tegevusi. Arvestuse jaoks pidid tudengid varasemalt esitama mitmeid ekraanivaateid eduka lahendamise tõestuseks, mille kontrollimine oli õppejõudude jaoks ajakulukas. Täiendavalt pidid tudengid ootama õppejõu tagasisidet, et olla veendunud oma lahenduskäigus. Käesoleva bakalaureusetöö eesmärgiks oli luua veebipõhise Linuxi käsurea õpikeskkond, kus tudengid saaksid interaktiivselt harjutada Linuxi käsurea kasutamist ja süsteem kontrolliks ülesannete edukat lahendamist automaatselt.

Antud töös kirjeldatakse õpikeskkonna nõudeid ja tööpõhimõtteid. Õpikeskkonna arendamiseks kasutati veebiliideses JavaScripti, HTML’i ja CSS’i. Veebiserveri arenduseks kasutati Node.js’i ja Dockeri konteinertehnoloogiat. Lõputöö viimastes peatükkides tuuakse välja rakenduse testimise tulemused ning edasiste arenduste võimalused. Töö tulemusena valmis Linuxi käsurea õpikeskkond, kus on implementeeritud automaatkontrolli süsteem. Keskkonda katsetati edukalt 100 tudengiga aines „Andmeturve“ (LTAT.06.002).

Võtmesõnad: käsurida, Linux, Docker, operatsioonisüsteemid, andmeturve, õppematerjalid

CERCS: P175 Informaatika, süsteemiteooria

Creation of a Web-Based Linux Command-line Learning Environment for Estonians

Abstract:

Students traditionally learn the Linux command-line interface by solving various tasks using it in courses "Operating systems" and "Computer Security" taught at the University of Tartu. Students had to submit screenshots as proof of each successful solution in the past. Each command had to be looked through separately, which was time-consuming for teachers. In addition, students had to wait for the teacher's feedback to see their results. The aim of this bachelor thesis is to create a web-based Linux command line learning environment where students could interactively practice their skills. The system would automatically check the successful completion of tasks.

This paper describes the requirements of the learning environment and the working principles. JavaScript, HTML, and CSS were used to develop the learning environment’s in web interface. The Node.js web server and Docker container technology were used to develop the environment’s server application. The final chapters of this thesis present the application testing results and the possibilities for further developments. As a result of this thesis, the Linux command line learning environment was created with an automatic-check system and successfully tested with 100 students in “Computer Security” (LTAT.06.002) course.

Keywords: command line, Linux, Docker, operating systems, computer security, teaching materials

CERCS: P175 Informatics, systems theory

Sisukord

Sissejuhatus

Tartu Ülikoolis õpetatavate kursuste „Operatsioonisüsteemid“ (LTAT.06.001) ning „Andmeturve“ (LTAT.06.002) esimestes praktikumides õpetatakse ning tuletatakse meelde Linuxi käsurida. Harjutamiseks kasutatakse Ubuntu 20.04 VirtualBox virtuaalmasinat, mille tudengid oma arvutisse seadistavad. Praktikumides sooritatakse käsureal ülesandeid, mille lahenduskäigud peab üles laadima kursuse wiki- või praktikumi lehele. Praktikumide juhendaja vaatab esitatud lahenduskäigud ühekaupa üle ja märgib tulemuse hindetabelisse. Tulenevalt rohkest tudengite arvust nõuab lahenduste kontrollimine palju ajaressurssi. Lisaks ülesannete kontrollimise ajalisele kulule, võivad käsitsi kontrollimisel lihtsamad vead inimliku eksimuse tõttu märkamata jääda. Samuti ei saa tudengid kohest tagasisidet oma tööle, vaid peavad ootama praktikumi juhendaja tagasisidet.

Töö eesmärgiks oli luua eestikeelne automaatkontrolliga Linuxi käsurea õpikeskkond, mille kasutamiseks piisaks veebilehitsejast. Õpikeskkonna peamiseks sihiks oli vähendada ülesannete kontrollimiseks kuluvat aega. Veebikeskkonna eeliseks on mugavus ja lihtsus – üliõpilased ei pea ootama virtuaalmasina käivitumist ega muretsema virtuaalmasina seadistamise ja haldamise pärast. Samuti võimaldab veebikeskkond võrreldes virtuaalmasinaga kohest tagasisidet, kui ülesanne on edukalt lahendatud.

Käesoleva töö esimeses peatükis tutvustatakse lühidalt Linuxi käsurida ning käsurea tundmise vajalikkust. Samuti tuuakse välja alternatiivsed käsurea lahendused koos nende plusside ja miinustega. Lisaks kirjeldatakse lipumängu, automaatkontrolli lahendust ja olemasolevate lahenduste mõju töö käigus valmivale praktilisele osale. Teises peatükis tuuakse välja veebirakenduse funktsionaalsed ja mittefunktsionaalsed nõuded ning kasutuslood. Sellele järgnevas peatüks antakse ülevaade rakenduses kasutatud tehnoloogiatest ja rakenduse poolt kasutatavatest portidest. Neljandas peatükis kirjeldatakse rakenduse tööpõhimõtteid, arhitektuuri ning veebiliidese disaini. Viiendas peatükis kirjeldatakse rakenduse testimise läbiviimist ja testimise tulemusi. Lisaks analüüsitakse viiendas peatükis valminud rakendust.

Mõisted ja terminid

API (ingl *Application Programming Interface*) ehk rakendusprogrammi liides on arvuti rakendusprogrammiga määratud reeglistik, mille alusel rakendusprogramm kasutab teise rakendusprogrammi teenuseid[1].

Brauseriküpsis (ingl *cookie*) ehk lühemalt **küpsis** on väike tekstistring, mille kirjutab veebiserver kliendi brauserisse ja mille esitab brauser igal järgmisel pöördumisel[2].

CTF (ingl *capture the flag*) ehk **lipumäng** on mäng, kus osalejad peavad leidma üles ära peidetud lipud. Iga lipp annab vihje järgmise lipu asukoha kohta ning sedamoodi saab jõuda lõpplahenduseni.

Dockeri Kettatõmmis (ingl *Docker image*) on Dockeri konteineri mall, mille põhjal luuakse konteinereid[3].

Külaliskasutaja on kasutaja, kes pole nimeliselt lõputöö käigus valminud rakendusse sisseloginud.

Nimeline kasutaja on kasutaja, kes on nimeliselt lõputöö käigus valminud rakendusse loginud.

Orkestratsioon on keerukate arvutisüsteemide, vahetarkvara ja teenuste automatiseeritud korraldus, koordineerimine ja haldus[2].

Pordi siirdamine (ingl *port mapping*) on ühe protokolli järgi struktureeritud andmete edastus teise protokolli vormingus, võimaldab ühel võrgul edastada andmeid teise võrgu ühenduste kaudu[2].

Port on lõpp-punkt, mille kaudu signaalid saabuvad võrku või väljuvad võrgust[2]. Läbi pordi saabuvad ja väljuvad andmepaketid ning igal IP aadressil on 65536 porti.

PTY (ingl *pseudo terminal*) on pseudo terminal, mis pole otseses ühenduses kerneliga, vaid on ühenduses vaheprogrammi (näiteks SSH abil), mis vahendab andmeid.

Rakendusserver (ingl *application server*) on programm, mis teeb ära kõik rakendusoperatsioonid ühelt poolt kasutajate ja teiselt poolt organisatsiooni põhi-ärirakenduste või andmebaaside vahel[1].

SSH (ingl *secure shell*) ehk **turvaline kest** on mitmeplatvormiline protokollisari, mis võimaldab turvalist krüpteeritud kaugpöördust[2].

WebSocket on standardprotokoll kiireks kahesuunaliseks andmeedastuseks üheainsa TCP-ühenduse kaudu brauseri ja veebiserveri vahel[2].

Veebiliides (ingl *web interface*) on liides, mis võimaldab veebiserveris olevad veebilehti külastada ning kasutada.

---

[1] Pärineb info- ja sidetehnoloogia sõnaraamatust http://vallaste.ee/

[2] Pärineb andmekaitse ja infoturbe leksikonist AKIT https://akit.cyber.ee/.

[3] Pärineb Dockeri dokumentatsioonist https://docs.docker.com/get-started/overview/#images

1. Valdkonna kirjeldus

Järgnevas peatükis kirjeldatakse põgusalt Linuxil põhinevate operatsioonisüsteemide kasutusvaldkondi ja tutvustatakse Linuxi käsurida ning käsurea tundmise vajalikkust. Täiendavalt tuuakse välja alternatiivsed käsurea lahendused koos nende plusside ja miinustega. Lisaks kirjeldatakse lipumängu ja automaatkontrolli lahendust ning olemasolevate lahenduste mõju töö käigus valminud praktilisele osale.

1.1 Linuxi käsurida

Linux on vabavaraline operatsioonisüsteem, mille lõi 1991. aastal Linus Torvalds. Linuxi, millel põhineb mitmeid operatsioonisüsteeme, peamiseks kasutajaliideseks on käsurida. Mrkonjić [1] on kirjutanud, et Linuxil põhinevat operatsioonisüsteemi Andorid kasutati 2021. jaanuari seisuga enamikes nutitelefonides. Mrkonjić toob välja, et Linuxil põhinevad operatsioonisüsteemid leiavad kasutust veebiserverites, pilveteenustes, super-arvutites ning värkvõrgu seadmetes. Ligi pooled tarkvaraarendajatest kasutavad arendustegevuse käigus Linuxil põhinevaid süsteeme [2].

Luues käsureapõhist kursust väidab Goldstein [3], et tehnoloogia arengu tulemusena kasutatakse käsurea asemel rohkem graafilist kasutajaliidest. Kuigi käsurea populaarsus arvutikasutajate seas on languses, toob Goldsteini välja järgnevad käsurea eelised: käsurea kasutamine on graafilise kasutajaliidese kasutamisest efektiivsem, käsurida annab suurema kontrolli kasutajale administratiivsete ülesannete täitmiseks, lisaks võimaldab käsurida kiirelt installeerida rakendusi ning kasutada erinevaid tööriistu korduvate ülesannete tegemiseks [3:41].

Eelneva põhjal võib väita, et tänapäeval leiavad Linuxil põhinevad operatsioonisüsteemid laialdast rakendust ning nende seadistamiseks on vaja käsurea kasutamisoskust.

1.2 Alternatiivsed käsurea õppimise lahendused

Internetis leidub mitmeid veebirakendusi ja lehekülgi, mis õpetavad tasuta Linuxi käsurida. Osad leheküljed pakuvad teoreetilist tausta, teised aga ainult harjutuskeskkonda, kus operatsioonisüsteemi proovida ning mõni lehekülg pakub mõlemat.

1.2.1 Webminal

Üheks õppimisvariandiks on lehekülg Webminal [4]. Webminal on nelja inimese poolt 2010. aastal loodud käsurea õppimise veebileht, mis jookseb Linode pilveteenuses. Webminalis luuakse igale soovijale Linux CentOS 7 virtuaalmasinas kasutaja. Selleks peab kasutajaks registreerima, kuid see protsess ei võta kaua aega. Pärast registreerumist ja masinasse sisselogimist saab juurdepääsu käsureale ning ülesannetele (vt lisa I).

Webminal leheküljel on väga hästi lahendatud kogu ekraaniala kasutamine ja teksti suuruse muutmine. Samuti on keskkonnas kättesaadavad manuaalileheküljed ning ülesanded koos õppematerjalidega nii teksti kui ka video kujul.

2019. aastal oli veebirakendusel üle 142 000 kasutaja ning vähemalt poole aasta jooksul ei kustutata kasutajaid masinast. Webminali üheks miinuseks on aeglane terminal, mis võib

tuleneda veebirakenduse kehvast skaleeritavusest (ingl *scalability*). Pärast 30 minutit inaktiivsust terminaliaknas katkestatakse sessioon ning tuleb uuesti sisse logida. Lisaks ei ole Webminali leheküljel automaatkontrolli ning ei saa kasutada *sudo* käske eelnevalt administraatorilt luba küsimata. Webminalis luuakse ühte operatsioonisüsteemi kasutajaid juurde, kuid eraldi konteinerites olevad kasutajad oleks paremini isoleeritud ning ei mõjutaks teineteist.

1.2.2 Virtuaalarvutid veebist

Käsurea ja/või graafilise kasutajaliidesega erinevate operatsioonisüsteemide testimiseks saab kasutada lehekülge bellard.org[4]. See lehekülg võimaldab üsna lihtsalt katsetada erinevaid operatsioonisüsteeme kasutaja loomiseta.

Bellard.org[4] leheküljel puudub konsoolist kopeerimise võimalus ning kahes konsoolis kolmest ei ole võimalik avada manuaalilehekülgi `man` käsuga. Samuti puuduvad leheküljel ülesanded, mida lahendada.

1.2.3 Linux Journey

Tutvustavat teksti koos lühiülesannetega pakub Linux Journey [5], mis kirjeldab Quach'i (lehekülje autori) teekonda Linuxi käsurea õppimisest. Leheküljel on hästi seletatud ja põhjendatud harjutused, mis on algaja ja edasijõudnud kasutaja jaoks huvitavad ning arusaadavad.

Miinusena võib välja tuua, et pärast ülesande lahendamist ei tule teavitust ning veebileht ei jäta meelde, milline ülesanne oli tehtud ning milline mitte. Sellest võib järeldada, et tõenäoliselt on tegu ainult veebiliidese staatilise leheküljega, millel puudub serveri osa. Lisaks võib välja tuua, et Linux Journey erinevate keelte lahendus on poolik. Näiteks kreeka keele puhul on kreeka keeles vaid peamenüü ja enamik nupud, kuid mitmete ülesannete tekstid on ingliskeelsed.

1.3 Muud lahendused

1.3.1 Lipumäng

Lipumängu tüüpi ülesanded rakendavad mängulist õpet. Mänguline aspekt teeb õppimise autori arvates põnevamaks ja huvitavamaks. Järgnevad CTF stiilis ülesanded vajavad SSH ühendust, seega SSH ühenduse loomiseks tuleb Windowsis kasutada mõnd rakendust või paigaldada virtuaalmasin, milles on lihtsam SSH ühendust luua.

OverTheWire'i poolt loodud Bandit [6] on õpetuslik ning algajale mõeldud CTF stiilis mäng, mis annab hea sissejuhatuse sarnastesse mängudesse ja Linuxi süsteemis navigeerimisse. Ülesannete läbimiseks tuleb kasutada väga lihtsaid käske ning käsud, mida vaja võiks

[4] https://bellard.org/jslinux/

minna, on ette antud vihjetena. Igas ülesandes tuleb siseneda uude masinasse ning sealt leitud lippu kasutada järgmisse masinasse sisselogimiseks. Ülesandeid on üle 30 ning iga järgnev ülesanne on veidi keerulisem kui eelmine.

Kursuse „Süsteemihaldus“ esimese praktikumi[5] jooksul tuleb lahendada ülesanne, mis on CTF stiilis ning automaatkontrolliga. Kursusest osa võttes tuleb ühest virtuaalmasinast üles leida mitu lippu selleks, et viimase lipuga saata signaal automaatkontrollile. Kui õige signaal on saadetud, on ka ülesanne tehtud ning tulemus näha.

1.3.2 Automaatkontrollid Tartu Ülikooli arvutiteaduse instituudis

Tartu Ülikooli arvutiteaduse instituudis on tehtud automaatkontrolle mitmetes programmeerimist õpetavates ainetes näiteks „Programmeerimine“, „Programmeerimisharjutused“, „Objektorienteeritud programmeerimine“, „Funktsionaalprogrammeerimine“ jt. Eelloetletud õppeainete automaatkontrollid on ühendatud Moodle keskkonnaga. Lahendused laetakse Moodle keskkonda, kus käivitatakse vastavad testid ning läbitud testide põhjal antakse tagasiside.

Andmebaasidega seotud ainetele on tehtud mitmeid automaatteste. Ühe hiljutise näitena võiks mainida automaatteste, mille koostas M. Kakk [7]. Tema lõputöö käigus valminud automaatkontrolli käivitamiseks tuleb alla tõmmata kokku pakitud andmebaas, see lahti pakkida ning käivitada automaatkontrolli fail. Kuigi sellised automaattestid võimaldavad kiiret andmebaaside kontrollimist, tuleb siiski iga kord teha automatiseeritavaid tegevusi, nagu kokku pakitud andmebaasi lahti pakkimine.

Teistes virtuaalmasinaid kasutavates ainetes nagu „Süsteemihaldus“[6] logib automaatsüsteem iga tudengi virtuaalsesse arvutisse ning käivitab seal automaatteste. Selline automaatkontroll toimiks ka mitmete aine „Operatsioonisüsteemid“ ja aine „Andmeturve“ praktikumide jaoks, kuid nõuaks aine ümbertegemist pilvetehnoloogiale. Hetkel kasutuses oleva VirtualBox lahendusega seda mugavalt teha ei saa.

1.4 Alternatiivse lahenduste analüüs ja sobivus antud töös

Antud töös on plaanis kombineerida erinevate alternatiivsete lahenduste positiivsed küljed ning luua negatiivsetele pooltele paremaid lahendusi, et valmiks õppeainetele sobiv Linuxi käsurea õpikeskkond. Töö väärtuseks on ühtne keskkond, milles on olemas ülesanded, ülesannete lahendamise koht (terminal) ning kohese tagasiside võimalus automaatkontrolli näol. Hetkel puudub eelkirjeldatud tasuta kättesaadav eestikeelne lahendus.

Lõputöö autorile meeldis Webminali ülesannete ning terminali visuaalne kujutus, seepärast otsustas lõputöö autor lähtuda visuaalse poole loomisel Webminali leheküljest. Võrreldes veebist leitavate virtuaalarvutitega on lõputöö käigus valmival veebilehel võimalik kopee-

[5] Süsteemihalduse esimese praktikumi juhend: https://courses.cs.ut.ee/2022/sa/spring/Main/IntroToLinux

[6] Aine „Süsteemihaldus“ koduleht https://courses.cs.ut.ee/2022/sa/spring/Main

rida otse konsooli ning on olemas ülesanded. Antud töö raames mängib olulist rolli automaatkontroll, mis olemasolevatel käsurea lahendustel puudub. Automaatkontrolli lahenduse loomiseks on plaanis võtta eeskuju Tartu Ülikooli olemasolevatest lahendustest. Lisaks on plaanis kasutada konteinertehnoloogiat, erinevalt Webminalist, kus luuakse ühte operatsioonisüsteemi kasutajaid juurde. Konteinertehnoloogia abil luuakse igale kasutajale eraldi konteiner operatsioonisüsteemiga. Sel juhul on kasutajad paremini isoleeritud ning ei mõjuta teineteist.

Käesolevas töös valmiv veebileht võiks jätta pärast ülesande tegemist ülesande meelde sama veebibrauseri piires. Arendatavas keskkonnas ei ole olulised käskude kirjeldused ning õpetused, kuna nende asemel on olemas loengud ja praktikumi materjal, mida ülesannete tegijatel on soovitatav enne läbida. Lahendavate käsurea ülesannete lõbusamaks muutmiseks on plaanis kasutada lipumängu stiilis taktikat (näiteks üks ülesannetest võiks olla kasutaja parooli, mida läheb hilisemates ülesannetes tarvis, leidmine tekstifailist). Töö käigus valmiva rakenduse teeb eriliseks õpikeskkond, milles on ühendatud ülesanded, ülesandeid kontrolliv automaatkontroll ning ülesannete lahendamiseks vajalik keskkond. Sellest lähtudes pannakse paika rakenduse nõuded.

2. Veebirakenduse nõuded

Valmiva rakenduse jaoks on koostatud funktsionaalsed ja mittefunktsionaalsed nõuded. Nõuded said paika pandud esimestel koosolekutel ainete „Andmeturve“ ning „Operatsioonisüsteemid“ vastutava õppejõu ja juhendaja Alo Peetsiga. Mõned nõuded täpsustusid ning muutusid töö käigus. Järgnevalt on toodud veebirakenduse funktsionaalsed nõuded, mittefunktsionaalsed nõuded ning nõuete põhjal valminud kasutuslood.

2.1 Funktsionaalsed nõuded

Järgnevalt on toodud rakenduse funktsionaalsed nõuded:

1. Peab olema võimalik ülesannet (juhiseid) näha ja Linuxi käske sisestada samaaegselt ühes veebikeskkonnas.
2. Iga ülesande juurde käib automaatkontroll, mis loeb ülesande õigeks pärast ülesande lahendamist.
3. Pärast ülesande edukat lahendamist, avaneb järgmine tegemata ülesanne.
4. Pärast ülesande edukat lahendamist markeeritakse tehtud ülesanne ja suletakse ülesande tekst.
5. Ülesannete juurde on võimalik lisada vihjeid.
6. Käsureale saab töölaualt kleepida ning käsurealt saab kopeerida teksti.
7. Rakenduses saab jooksutada käsurea käske.
8. Kasutajate tehtud ülesanded salvestatakse ajatempliga serveris asuvasse faili.
9. Rakenduses peab saama kasutajaid autentida.
10. Rakendust peab saama kasutada nimelise kasutaja loomiseta.
11. Pärast etteantud aega peab olema võimalik konteinereid automaatselt kustutada, eesmärgiga hoida masina ressursikasutus madal.
12. Esileheküljel on juhend sisselogimiseks.
13. Kasutajatel on võimalus anda tagasisidet.

2.2 Mittefunktsionaalsed nõuded

Järgnevalt on toodud rakenduse mittefunktsionaalsed nõuded:

1. Käsurida peab asuma lehe paremal pool ja ülesanded lehe vasakul pool.
2. 100 kasutajat peavad saama paralleelselt rakendusega töötada.
3. Edukalt lahendatud ülesanded märgitakse õigeks vähemalt 1 sekundi jooksul pärast ülesande õigesti sooritamist vähemalt 95% juhtudest.
4. Kasutaja käsurea käsu väljund leitakse selle konteineris läbijooksutamise tulemusel.
5. Rakenduses kasutatav käsurida peab sarnanema Ubuntu 20.04 käsureaga.
6. Kasutajad saavad erinevatest arvutitest ligi pääseda samale konteinerile ainult sama kontoga.
7. Vihje kuvamiseks peab kasutaja viima kursori vihjele.
8. Sisselogitud kasutajal on võimalus näha oma nime ja matrikli numbrit ülesannete lehel isikustatud ekraanivaate tegemise eesmärgil, mis esitatakse praktikumi arvestuseks.
9. Kasutajaliides peab olema kasutatav kõigis > 5% turuosaga veebilehitsejates.

10. Veebikeskkond on kättesaadav Tartu Ülikooli sisevõrgust 97% ajast.
11. Tartu Ülikooli poolt antaval arvutil peab iga konteineri protsessori kasutus jääma alla 10% kogu protsessori kasutusest.
12. Ükski konteiner ei kasuta üle 512 MB mälu.
13. Värvipimedus ei sega rakenduse kasutamist.

2.3 Kasutuslood

Antud alapeatükis on kirjeldatud, kuidas võiks töö praktilise osa raames valmiv käsurea õpikeskkond toimida erinevates situatsioonides.

2.3.1 Kasutuslugu 1 – Kasutaja loomine

Aine „Andmeturve“ kuulaja (edaspidi tudeng) avab praktikumi juhendi, mis viitab Linuxi õpikeskkonnale. Sisenedes lehele esmakordselt pole tudengil kasutajat.

Eelnev seis: Tudengil ei olnud varem kasutajat (kasutaja on aegunud või ei ole loodud) ning tema veebilehitsejas on avatud veebirakenduse esilehekülg (vt peatükk 4.2.1).

Tudeng järgib esilehel antud teavet sisselogimise kohta ning sisestab vastavatesse lahtritesse oma nime ning matriklinumbri. Tudeng vajutab nupule „Logi sisse“.

Järgnev seis: Tudeng saab alustada ülesannetega.

Sündmuste jada põhistseeni puhul:

1. Tudeng on suunatud ülesannete leheküljele (vt peatükk 4.2.2) ning saab alustada ülesannete tegemist.

Sündmuste jada alternatiivse olukorra puhul, mil veebiserveril ei õnnestunud uut konteinerit luua:

1. Tudengit teavitatakse sõnumiga ekraanil ning tudeng saab uuesti proovida.
2. Kui tudeng ei soovi uuesti proovida saab ta veast teada anda.

Sündmuste jada alternatiivse olukorra puhul, mil tudeng on kogemata kirjutanud matriklinumbri väikese tähega või jätnud algustähe ära:

1. Veebilehele ilmub märguanne, mis teavitab valest matriklinumbri vormist.
2. Tudeng saab matriklinumbri parandada ning uuesti nupule „Logi sisse“ vajutada.
3. Tudeng suunatakse ülesannete leheküljele ning ta saab alustada ülesannete tegemist.

2.3.2 Kasutuslugu 2 – Ülesande lahendamine

Tudeng lahendab ülesandeid.

Eelnev seis: Tudeng on sisselogitud ning veebilehitsejas on avatud ülesannete lehekülg.

Järgnev seis: Tudengi edukalt lahendatud ülesanne on märgitud roheliseks.

Sündmuste jada põhistseeni puhul:

1. Tudengil on lahendatav ülesanne avatud.
2. Tudeng proovib erinevaid käske ülesande lahendamiseks.

3. Tudeng vaatab ülesande kirjelduse juures olevat vihjet, kui see on olemas.
4. Tudeng kasutab käsku `man`, et saada abi käskude käivitamise kohta.
5. Tudeng lahendab ülesande edukalt.
6. Lahendatud ülesanne sulgub, värvub roheliseks ning avaneb järgmine lahendamata ülesanne (kui leidub veel lahendamata ülesandeid).

Sündmuste jada alternatiivse olukorra puhul, mil tudengi tegevus on põhjustanud konteineri mittefunktsioneerimise:

1. Tudeng kustutas kogemata kogu oma virtuaalmasina sisu ning käsud ei tööta enam.
2. Tudeng saab minna esilehele ning logida sisse uue matriklinumbriga, milleks on tudengi matriklinumber ning üks lisanumber.

Sündmuste jada alternatiivse olukorra puhul, mil ülesannete leheküljel katkeb ootamatult ühendus:

1. Tudeng lahendab ülesannet.
2. Tudengil katkeb internetiühendus.
3. Ülesannete leheküljele tuleb teave ühenduse katkemisest.
4. Internetiühendus tuleb tagasi.
5. Ülesannete leheküljele tuleb teave ühenduse loomisest.
6. Tudeng saab jätkata ülesande tegemist.
7. Pärast ülesande edukat lahendamist sulgub ülesande kirjeldus, tehtud ülesanne värvub roheliseks ning avatakse uus ülesanne (kui leidub veel tegemata ülesanne).

Sündmuste jada alternatiivse olukorra puhul, mil veebiserver lakkab töötamast:

1. Tudeng lahendab ülesannet.
2. Ülesannete lehekülg lakkab töötamast, kuna veebiserver ei tööta.
3. Tudeng saab kirjutada tagasisidesse, et veebiserver pole kättesaadav, kuna laetud lehekülje tagasiside link töötab.
4. Pärast veebiserveri parandamist aine õppejõudude poolt saab tudeng keskkonda sisse logida.
5. Õppejõud saadab tudengile teate uuesti töökorras olevast keskkonnast.
6. Tudeng saab jätkata ülesannete lahendamist.
7. Pärast ülesande edukat lahendamist sulgub ülesande kirjeldus, tehtud ülesanne värvub roheliseks ning avatakse uus ülesanne (kui leidub veel tegemata ülesanne).

Sündmuste jada alternatiivse olukorra puhul, mil tudengi arvates esineb automaatkontrollis viga:

1. Tudeng lahendab enda arvates ülesande edukalt, kuid ülesanne ei värvu roheliseks või tudeng lahendab ühe ülesande edukalt, kuid roheliseks värvub rohkem ülesandeid.
1. Tudeng saab kirjeldada olukorda tagasiside küsimustikus esitades kirjelduse, kuidas ta ülesannet lahendas ja lisada oma ekraanivaate.

2.3.3 **Kasutuslugu 3 – Ülesannete lahendamise jätkamine**

Tudeng ei saanud praktikumis ülesandeid lõpuni lahendatud ning soovib kodus jätkata.

Eelnev seis: Tudengil on varem kehtiv kasutaja olemas. Tudeng järgib esilehel antud teavet sisselogimise kohta ning sisestab vastavatesse lahtritesse oma nime ning matriklinumbri. Tudeng vajutab nupule „Logi sisse“.

Järgnev seis: Tudeng saab alustada ülesannetega.

Sündmuste jada põhistseeni puhul:

1. Tudeng kasutab sama arvutit ning veebilehitsejat, mis praktikumis.
2. Tudeng on suunatud ülesannete leheküljele (vt peatükk 4.2.2) ning saab alustada ülesannete tegemist.
3. Tudengil avaneb sama vaade ülesannete leheküljel, mis eelmine kord enne lõpetamist. Ülesanded, mis varem tehtud olid, on endiselt märgitud.
4. Tudeng saab jätkata ülesannetega sealt, kust viimati pooleli jäi.

Sündmuste jada alternatiivse olukorra puhul, mil tudeng sisestas vale matriklinumbri:

1. Tudeng suunatakse uuele ülesannete leheküljele.
2. Uuel ülesannete leheküljel on kõik ülesanded tegemata.
3. Tudeng saab alustada ülesannete tegemist algusest või minna tagasi esilehele ning logida õige matriklinumbriga sisse.

Sündmuste jada alternatiivse olukorra puhul, mil tudengil on viimasest sisselogimisest möödas üle 24 tunni:

1. Tudeng suunatakse uuele ülesannete leheküljele, kuid kõik ülesanded on tegemata.

Sündmuste jada alternatiivse olukorra puhul, mil tudeng kasutab kodus ülesannete jätkamiseks teist arvutit, veebilehitsejat või tühjendas veebilehitseja *local storage*'it:

1. Tudeng suunatakse samale ülesannete leheküljele, kus ta praktikumi lõpetas, kuid ülesanded on märgitud kui tegemata.

Eelnevad kasutuslood kirjeldasid veebiliidese töötamist lõppkasutaja vaates. Järgnevalt antakse ülevaade nõuetele ja kasutuslugudele vastava õpikeskkonna loomiseks vaja minevatest tehnoloogiatest.

3. Rakenduse ülevaade

Järgnevad alapeatükid kirjeldavad rakenduses kasutatud tehnoloogiaid. Lisaks põhjendatakse rakenduse konteinerite poolt kasutatavaid porte.

3.1 Tehnoloogiate valik

Rakendus jaotub mitmeks komponendiks ning tulenevalt töö keerukusest muutusid arendusprotsessi käigus osad tehnoloogiad. Järgnevad alapeatükid aitavad mõista arendusprotsessi käigus tekkinud valikuid. Tehnoloogiate valikus tuuakse välja lisaks kasutatud tehnoloogiatele ka alternatiivsed valikud ning põhjendused, miks otsustati just selle kasuks, mida kasutati.

3.1.1 Veebiliides (*web interface*)

Veebiliidese funktsionaalsuse loomiseks kasutas autor JavaScripti, kuna sellega oli autoril varasem kogemus olemas ning JavaScripti kasutatakse enamikes veebilehtedes [8]. Alternatiivselt JavaScriptile oleks olnud võimalik kasutada ActionScripti. Adobe Flash Player'it pole aastast 2021 toetatud [9], seega polnud ActionScript kuigi mõistlik lahendus.

Veebiliidese lihtsuse tõttu polnud ühtegi teeki ega raamistiku tarvis kasutada, kuigi tulevases edasiarenduses on üheks võimaluseks raamistikku Vue.js[7] kasutamine. Vue.js on antud veebiliidese arendamiseks sobilik, kuna see on kompaktne raamistik [10], millega saab veebirakendust muuta üheleherakenduseks (ingl *Single-page Application*). Üheleherakendus hoiaks ruumi kokku ning kiirendaks laadimist.

Veebistandardi kohaselt kasutati teksti märgistuskeelena HTML'i (ingl *HyperText Markup Language)* ning lehekülje välisilme kirjeldamiseks CSS'i (ingl *Cascading Style Sheets*). Selline valik võimaldab veebilehte kuvada paljudel veebilehitsejatel.

3.1.2 Rakendusserver

Lei jt [11] võrdlesid 2014. aastal PHP (ingl *Hypertext Preprocessor* varasemalt *Personal Home Page*), Node.js[8] ning Pythonit veebiserveritena. Nende uuringus selgus, et Python sobis teistest paremini suure arvutusmahuga veebiserveriteks, samas kui PHP ja Node.js olid paremad kõrge I/O kasutusega serverites. Samas uuringus toodi ka välja, et PHP tuleb Node.js'ist paremini toime väiksema hulga HTTP (ingl *Hypertext Transfer Protocol*) päringutega ning Node.js oli kolmest keelest parima suure hulga HTTP päringute haldamisega. Challapalli jt [12] tegid sarnase uuringu 2021. aastal kus võrdlesid ainult Pythonit ning Node.js ning leidsid samuti, et Node.js veebiserver suudab vastu võtta rohkem HTTP päringuid sekundis kui Pythoni server. Vanemas (2014 a.) uuringus oli kasutatud tolleaegseid programmeerimiskeelte versioone, kuid kaheksa aasta jooksul on keeled palju arenenud. Tuleb nentida, et uuemas (2021 a.) uuringus ei ole testimise käigus kasutatud programmeerimiskeelte versioone välja toodud. Nende uurimuste ja praktiliste katsetuste põhjal võib

[7] Vue.js - The Progressive JavaScript Framework | Vue.js (vuejs.org)
[8] Node.js (nodejs.org)

väita, et Node.js on igati sobilik antud rakenduse arendamiseks, kuna rakendus ei ole arvutusmahu poolest nõudlik, vaid on suure I/O kasutusega (vahendatakse terminali ning konteineri suhtlust).

Node.js'is ning PHP's on olemas teegid SSH ühenduste loomiseks ning Dockeri konteinerite haldamiseks. Lähtuvalt eelnevalt kirjeldatud uuringutulemustest ning autori varasemast JavaScripti kogemusest otsustati Node.js veebiserveri kasuks. Seega kasutatakse rakenduse esi- ning serverpoolel sama programmeerimisekeelt (JavaScripti). Sama programmeerimisekeele kasutamine nii esi- kui ka serveripoolel teeb arendustegevuse lihtsamaks, kuna sama süntaks on laiemalt kasutuses.

3.1.3 **Konteinertehnoloogia**

Konteinertehnoloogia valikul on rakenduse loomise juures suur roll ning konteineriseerimist eelistatakse virtualiseerimisele just kiiruse ja kompaktsuse tõttu. Bentaleb jt [13] väidavad, et laialt levinud konteineriseerimistehnoloogia Docker on lihtne, kasutajasõbralik ning hästi arenenud kogukonnaga. Lisaks Dockerile toovad nad oma uurimuses välja Singularity ning uDockeri, mis on turvalisemad ja pakuvad rohkem võimalusi suuremahuliste teadusarvutuste tegemiseks. uDocker on ühilduv Dockeri keskkonna Docker-Hub'iga, ning seega on Dockeri kasutajal lihtsam üle minna uDockerile.

Teenuste arendusfaasis piisab enamasti mõnest konteinerist, kuid kui toodet või rakendust hakatakse laiemalt pakkuma, võib konteinerite arv kasvada väga kiiresti. Sellepärast on paljude konteinerite haldamiseks spetsiaalsed orkestratsioonid. Moravcik'i ja Kontsek'i [14] sõnul on populaarseteks konteinertehnoloogiate orkestratsioonideks Docker Swarm ning Kubernetes. Nad väidavad, et need orkestratsioonid erinevad keerukuse ja võimekuse poolest. Kuigi Dockeri tehnoloogia on algajale lihtsam ning kiiremini hoomatav, jääb see Kubernetes'ele võimekuse ja valikuvõimaluste poolest alla. Üheks suuremaks erinevuseks kahe orkestratsiooni vahel toovad autorid automaatse skaleeritavuse, mille võimekus Docker Swarm'il puudub.

Konteinertehnoloogia valik ei piira orkestratsiooni valikuvõimalusi, seetõttu võib kasutada Dockerit konteinerite loomiseks ning Kubernetes't loodud konteinerite haldamiseks. Rakenduse tegemiseks osutus sobivaks konteinertehnoloogiaks Docker, kuna töö autoril oli sellega varasemaid kogemusi ning Dockeri kasuks rääkis selle lihtsus ja suur kogukond. Käesolevas rakenduses polnud konteinerite orkestreerimist tarvis, kuna testimise käigus osutus rakendus piisavalt ressursiefektiivseks ning konteinerid olid suhteliselt lühikese elueaga.

3.1.4 **Operatsioonisüsteem**

Konteinerite operatsioonisüsteemina kasutati Ubuntut, seega on paketihalduriks (ingl *package mangaer*) vaikimis APT. Linuxil põhinevad operatsioonisüsteemid, mida konteinerites kasutatakse, on tavaliselt minimeeritud. Minimeerimise käigus on eemaldatud graafiline kasutajaliides, manuaalilehekülјed ja muu kasutajamugavus, et konteiner võtaks võimalikult vähe ruumi [15]. Minimeeritud Ubuntu on kasulik, kuna konteinereid kasutavad

peamiselt ainult programmid ja rakendused. Manuaalilehekülgedele minimeeritud versioonist ligipääsemiseks on tarvis Ubuntu suuremamahulisema versiooni installimine käsuga `unminimize`. Minimal Ubuntu 20.04 võtab ruumi umbes 236MB samas, kui selle suurem versioon võtab ruumi 417MB. Dockeri konteinerid jagavad omavahel suurt osa ruumist (erinevate kasutajate Ubuntu konteinerid on peaaegu identsed), seega pole kahekordne maht probleemiks, kuna see esineb ainult esimese konteineri puhul. Järgnevad sama sisuga konteinerid võtavad umbes samapalju ruumi (25 MB või vähem) mõlema Ubuntu versiooni puhul. Kui alus kettatõmmis on juba tavaversioonis, siis läheb uute konteinerite tegemine kiiresti ning iga kord ei pea käivitama tavaversiooniks uuendamise protsessi.

Lisaks Ubuntu versioonile on ka teised Debiani-põhised operatsoonisüsteemid vastavuses rakenduse nõuetega olles sealjuures kompaktsemad. Sellised on näiteks LinuxMint või Mini-Deb. Operatsioonisüsteemi valikul tuleb arvestada, et mida väiksem ja kompaktsem operatsioonisüsteem on, seda vähem on seal kasutajate mugavust parandavaid elemente, rakendusi ning tööriistu. Valides Ubuntust väiksema Linuxi distributsiooni, mille paketihaldur on APT, ning installides sinna vajalikud paketid (tööriistad SSH ühenduseks, manuaalilehekülged ning tekstiredaktorid nagu Vim ja Nano), ei pruugi jääda distributsioon mahult Ubuntust väiksemaks pakkudes kasutajatele piisavat funktsionaalsust õppimiseks.

3.2 Kasutatavate portide valik

IANA (ingl *Internet Assigned Numbers Authority*) poolt on kehtestatud normid portide kasutuse kohta [16]. Esimesed 1024 porti (vahemikus 0–1023) on rangelt määratud tihti kasutust leidvateks otstarveteks. SSH ühendus enamasti üle pordi 22, Telneti protokoll aga üle pordi 23, DNS (ingl *Domain Name Service*) suhtlus käib läbi pordi 53, HTTP (ingl *Hypertext Transfer Protocol*) käib aga üle pordi 80 jne.

Järgmises kategoorias on kasutajate pordid, mis jäävad vahemikku 1024–49151. See vahemik on teistest suurem ning selles vahemikus asuvad paljude firmade äritegevuseks vajaliku suhtluse jaoks kasutust leidvad ühendused. Kasutajate portide vahemiku kuuluvad näiteks Dockeri REST API (pordil 2375 ja turvalisem ühendus SSL'i (ingl *Secure Socets Layer*) põhiselt pordil 2376).

Viimases vahemikus (49152–65535) asuvad dünaamiliselt kasutatavad pordid, ehk see vahemik on jäetud rakendustele kasutamiseks ning selles vahemikus asuvaid porte ei saa IANA kaudu nimetada [17].

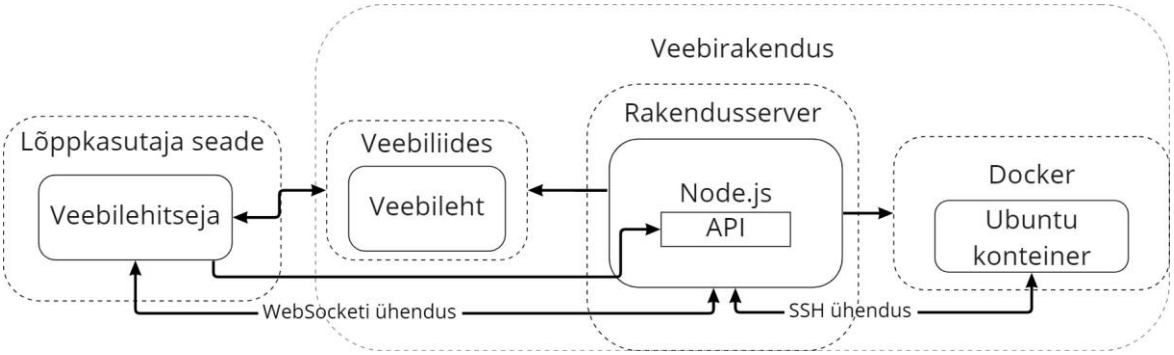
Lähtudes IANA standardist ning rakenduse olemusest on mõistlik rakenduse sees kasutada viimases vahemikus asuvaid porte (49152–65535). Portide haldamist viiakse läbi järjendis. Iga kord, kui järgmine port võetakse kasutusele, lisatakse see järjendisse ning kui port vabastatakse ehk kasutaja ei soovi enam olla arvutiga ühendatud, kustutatakse vastav pordi number järjendist.

4. Rakenduse arhitektuur ja disain

Järgnevates alapeatükkides kirjeldatakse rakendusserveri tööprotsessi ning rakenduse disaini. Rakendusserveris uuritakse lähemalt konteinerite haldust, konteineri ja veebiliidese vahelist suhtlust ning automaattestide toimimist. Lisaks antakse ülevaade rakenduse disainist esilehekülje ja ülesannete lehekülje kujul. Rakenduse kirjeldamisel kasutatakse lähtekoodist pärinevaid koodijuppe, lähtekoodi link on lisas II. Peatüki viimases alapeatükis tutvustatakse installatsioonijuhendit, mis õpetab rakenduse paigaldamist Linux serverisse.

4.1 Rakenduse arhitektuur

Rakendus koosneb kolmest suuremast komponendist. Nendeks on veebiliides, rakendusserver ning konteinerid (vt Joonis 1). Veebiliides vastutab päringute saatmise eest rakendusserverisse. Rakendusserver loob ja leiab konteinereid, loob veebilehe ning vahendab veebilehe ja konteinerite vahelist suhtlust. Nagu peatükis 3.1 kirjeldatud, haldab konteinereid Docker, rakendusserver töötab Node.js veebiserveril ning veebiliides jookseb JavaScriptil. Veebiserveriks testperioodil oli Tartu Ülikooli poolt antud kaasaegne sülearvuti, mis jooksutas lõputöö osana valminud rakendust. Rakendus töötab Tartu Ülikooli sisevõrgus, kuna rakendus pole piisavalt turvaliseks tehtud ning seega turvakaalutlustel ei ole mõistlik lehekülg avalikult internetist kättesaadavaks teha. Sellest tulenevalt peab lõppkasutaja seade olema ühendatud TÜ sisevõrguga. Lisaks sellele peab lõppkasutaja seadmel olema veebi-



lehitseja.

Joonis 1. Veebirakenduse arhitektuur

Rakenduse põhilisteks ülesanneteks on konteineri loomine ja leidmine, konteinerile vastava veebilehe loomine, konteineri ühendamine veebilehega ning ülesannete kontrollimine.

Node.js veebiserveri käivitumisel on serveri IP aadressile minnes võimalik leida veebileht (toodud Joonis 2). Veebilehel saab logida sisse külalisena või nimelise kasutajana. Kui sisselogimise viis on otsustatud ning vajutatakse nupule „Logi sisse“ või „Jätka külalisena“, saadetakse HTTP päring veebilehitsejast rakendusserverisse. Rakendusserveris otsitakse päringule vastav konteiner või luuakse see konteiner, kui seda ei eksisteeri. Dockeriga suhtluseks kasutab Node.js teeki dockerode[9]. See võimaldab Node.js'ist Dockerile käske saata kasutades Dockeri kaugrakendusliidest (ingl *remote API*).

[9] GitHub - apocas/dockerode: Docker + Node = Dockerode (Node.js module for Docker's Remote API)

Enne konteinerite loomist tuleb valmis teha Dockeri kettatõmmis (ingl *image*). Dockeri kettatõmmise põhjal valmistatakse konteinereid ning selle loomiseks kasutatakse *Dockerfile*'is asuvaid juhiseid. Joonis 3 toodud *Dockerfile* kasutab Ubuntu 20.04 konteinerit operatsioonisüsteemina. See fail installeerib operatsioonisüsteemile SSH ühenduse loomiseks vajaliku programmi ning tööriistakomplekti *inotify-tools*, mille osana saab kasutada failijälgimise protsessi *inotifywait*. Operatsioonisüsteemi lisatakse testkasutaja, ülesannete lahendamiseks vajalikud failid, avatakse (ingl *expose*) port 22 SSH ühenduseks, uuendatakse konteineri operatsioonisüsteem ning paigaldatakse manuaalileheküljed. Seejärel on Dockeri kettatõmmis valmis.

Joonis 2. Linuxi käsurea harjutuskeskkonna esilehekülg

```
FROM ubuntu:20.04
RUN apt update && apt install  openssh-server sudo -y
RUN apt install inotify-tools sudo -y
RUN useradd -rm -s /bin/bash -g root -G sudo test
RUN  echo 'test:Test1234' | chpasswd
RUN mkdir /home/test/.ajutine && echo "peidetud faili sisu Prakitkum1 veebruar
2022 \n...\n\nAdmin parool on Test1234" > /home/test/.ajutine/.h2sti_peidetud
RUN touch /home/.veel1Failon2ra_peidetud /home/.sedaEiPeaksKuvamaMeidetud
EXPOSE 22
RUN yes | unminimize
RUN apt update && apt install less
RUN apt install man-db -y
CMD ["/usr/sbin/sshd","-D"]
```

Joonis 3. Faili *Dockerfile* sisu

4.1.1 **Konteineri loomine**

Dockeri kettatõmmise jooksutamisel luuakse uus konteiner, milles on kettatõmmise sisu. Seega on eelnevalt kirjeldatud Dockeri kettatõmmise põhjal loodud konteineris avatud port 22. Konteineris on olemas ülesannete lahendamiseks vajalikud failid ning manuaalileheküljed on kättesaadavad. Pärast konteineri loomist nimetatakse konteineri nimi ümber juhul, kui konteineri loomiseks päringu saatev kasutaja oli nimeline kasutaja (vt Joonis 4). Vastasel juhul (kui konteinerit soovis külaliskasutaja) antakse konteinerile Dockeri poolt suvaline unikaalne nimi. Konteineri loomisel määratakse vastavalt mittefunktsionaalsetele nõuetele konteinerile mälu ning protsessoriaja kasutuse limiidid.

Dockeri sisevõrgus saab konteineriga SSH ühendust luua, kuid väljaspoolt Dockeri konteineri leidmiseks tuleb konteineri port 22 siirdada mõnele teisele pordile vahemikus 49152–65535. Näiteks esimese konteineri port 22 siirdatakse pordile 49152. Pordi siirdamine võimaldab hiljem SSH ühendust luua läbi siirdatud pordi (näites oli selleks 49152). Joonis 4 toodud koodilõigus valib siirdamiseks sobiliku pordi Node.js server (port on salvestatud muutujasse *containerPort*). Väljast nähtavad pordid ei tohi kattuda, seega siirdatakse iga konteineri port erinevale pordile.

Pärast konteineri loomist konteiner käivitatakse ja pannakse valmis SSH ühenduse loomiseks.

```
var docker = new Docker({ port: 22 })
docker.createContainer({
   Image: 'autogen_ubuntu_ssh',
   Tty: true,
   name: userID === 'anonymous' ? "" : userID,
   HostConfig: {
      Memory: 512000000, // 512 Megabytes
      CpuPeriod: 100000,
      CpuQuota: 10000,
      PortBindings: {
         "22/tcp": [{ HostPort: containerPort.toString() }]
      },
   },
}, function (err, container) {
   container.start({}, function (err, data) {
      if (err) reject(err)
      runExec(container, 'service ssh start');
   });
   container.inspect((err, data) => {
      if (err) reject(err)
      resolve(data.Id)
   })
});
```

Joonis 4. Koodilõik failist d*ockerController.js* konteineri loomisest

Loodud konteineri port ja konteineriID hoiustatakse kasutaja veebilehitseja brauseriküpsises, kuna nende järgi on võimalik hiljem kiiremalt sama konteinerit üles leida ning SSH ühendust luua.

4.1.2 Konteineri leidmine

Enne SSH ühenduse loomist on tarvis teada porti, millega ühendus luua. Pordi saab teada küpsisest või konteinerist. Kui küpsis on olemas tuleb veenduda, et konteiner töötab, vastasel juhul pole võimalik ühendust luua. Konteiner leidmist, kirjeldab koodilõik Joonis 5, mis on lihtsustatud paremaks loetavuseks. Küpsise olemasolul saab sealt teada konteineri ID, ning seejärel on võimalik veenduda, et antud ID-ga konteiner töötaks. Kui konteiner töötab, on see valmis SSH ühenduseks. Juhul kui sellist konteinerit pole olemas, tagastatakse staatus 404, ning proovitakse uuesti, kuid kustutatakse küpsis, kuna see oli kehtetu.

Kui küpsist ei ole, kuid leida tuleb nimelise kasutaja konteiner, otsitakse konteinerit, mille nimele vastaks kasutaja matrikli number. Leides sellise konteineri, veendutakse, et see töötab ning seejärel on konteiner valmis kaugühenduseks. Juhul, mil ühelgi konteineril polnud sellist nime, luuakse uus konteiner, mis on SSH ühenduseks valmis. Juhul, kui külaliskasutajal küpsist ei ole luuakse uus konteiner, kuna külaliskasutajat tuvastatakse veebirakenduses vaid küpsise abil.

```
if (containerID) {
   ensureContainerIsRunning(containerID)
      .then(() => {
         resolve({ 'status': 200 })
      }).catch(err => {
         reject({ 'status': 404 })
      });
}
else if (isKnownUser) {
   getContainerIdAndPortByUser(userID).then(idAndPort => {
      resolve({ 'status': 200})
   }).catch(err => {
      makeContainer(userID, containerHost, containerPort)
         .then((containerId) => {
            resolve({ 'status': 201 })
         }).catch((err) => {
            reject({ 'status': 403 })
         })
   });
}
else {
   makeContainer(userID, containerHost, containerPort)
      .then((containerId) => {
         resolve({ 'status': 201 })
      }).catch((err) => {
         reject({ 'status': 403 })
      })
}
```

Joonis 5. Lihtsustatud kujul koodilõik konteineri leidmisest failist d*ockerController.js*

4.1.3 Veebilehte konteineriga ühendamine

Igale eelnevalt kirjeldatud konteinerile luuakse eraldi pordil veebileht. Veebilehel saab kasutaja näha oma nime ja matrikli numbrit. Kui on tegu külaliskasutajaga, näeb kasutaja nime asemel sõna „külaline“. Nimi saadetakse kasutajale Node.js API kaudu. Iga kord, kui uus

kasutaja luuakse, luuakse ka uus GET päring kasutaja konteineri pordi nimelisele leheküljele, ehk kui konteineri port oli 49152, siis GET päring saadetakse aadressile http://<API aadress>/49152.

Terminali kasutamiseks ühendatakse veebilehe avamise hetkel terminal WebSocketi abil Node.js veebiserveriga, mis omakorda ühendatakse SSH kaudu konteineriga. Sellise suhtluse edastamist veebiserveri vaates kirjeldab Joonis 6. Kui SSH ühendus konteineri ja veebiserveri vahel toimib, hakkab veebiserver informatsiooni veebilehele edastama. Selle tulemusel saab veebilehel olevat terminali akent kasutada käskude sisestamiseks ning väljundi nägemiseks.

```
io.on('connection', function (socket) {
  var conn = new SSHClient();
  conn.on('ready', function () {
    socket.emit('data', '\r\n*** SSH CONNECTION ESTABLISHED ***\r\n');
    startInotify(conn, socket);
    startShellSession(conn, socket);
  }).on('close', function () {
    socket.emit('data', '\r\n*** SSH CONNECTION CLOSED ***\r\n');
    socket.disconnect();
  }).on('error', function (err) {
    socket.emit('data', '\r\n*** SSH CONNECTION ERROR: ' + err.message);
    if (err.message.startsWith('connect ECONNREFUSED')) {
      startWebSocketConnection(host, port, username, password, http);
    }
  }).connect({
    host: host,
    port: port,
    username: username,
    password: password,
  });
});
```

Joonis 6. Koodilõik konteineri ja veebilehe vahelise ühenduse haldamisest

Ülesannete automaatkontrolliks käivitatakse SSH ühenduse algul automaatselt *inotifywait* protsess, mis hakkab jälgima paari kausta. Kui kaustas olevaid faile või kausta ennast muudetakse, saadetakse selle kohta veebiliidesesse teade, mida ei kuvata terminalis.

4.1.4 Ülesannete kontrollimine

Ülesannete, mis on toodud lisas III, kontrollimiseks kasutatakse failide jälgimist ning regulaaravaldise (ingl *regular expression*) sobitamist. Failimuudatusteks kasutatakse eelnevalt mainitud *Inotify* käsku *inotifywait*, mis jälgib ette antud kaustas olevate failide kustutamist, avamist, üle kirjutamist ning loomist. Osad ülesanded ei ole seotud failide muutmisega, seetõttu jälgitakse lisaks käskude väljundeid.

Ülesande kontrollimiseks on vähemalt üks regulaaravaldis, millega käsu väljund peab sobituma. Seega, kui mingisugune tingimus on täidetud, loetakse ülesanne tehtuks. Näiteks faili */etc/hosts* sisu asemel piisab, kui terminal tagastab „127.0.0.1 localhost ip6-localhost localnet allnodes allrouters“ või muud sellesarnast. Üheks võimalikuks käskuks on `echo`

`127.0.0.1localhostip6-localhostlocalnet allnodesallrouters`, et lahendada esimene ülesanne. Selline lahenduskäik eeldaks faili sisu teadmist või JavaScripti faili lugemist.

Veebilehitsejast allatõmmatud Javascripti failist mõistliku informatsiooni välja lugemist raskendab raamistiku kasutamine, mis kompileeriks raamistiku rakenduse JavaScriptiks. Lisaks regulaaravaldise otsimisele terminali väljundist jälgitakse failide muudatusi protsessi *inotifywait* abil. Näiteks kasutatakse ülesandes 6 sellist kontrolli. 6. ülesandes tuleb esmalt kirjutada faili nimi ning seejärel muuta selle faili õigusi. Esimest osa kontrollitakse *inotifywait* abiga (kuulatakse faili muutmist mitte faili loomist). Pärast faili muutmist arvestatakse veebilehitsejas esimene osa tehtuks ning jäädakse ootama ülesande teise poole lahendamist. Teine pool nõuab faili õiguste muutmist ning `ls -la` käsu abil selle välja printimist. Faili õigusi kontrolliva süsteemi puudumise tõttu tuleb otsida käsu väljundite seast rida `-rw-rw----*test*root*andmeturve`, kus tärnide (*) asemel on üks või rohkem suvalist sümbolit, mis ei ole reavahetus. 6. ülesande mõlema osa edukalt täitmise järel värvitakse see roheliseks. Teiste ülesannete kontrollid toimivad samadel põhimõtetel.

Ülesannete koostamisel oli tähtis faili muutmine või ülesande sõnastamine nii, et konsoolist selguks õige võtmesõna või võtmesõnad. Käskude sisestamist polnud mõtet kontrollida, kuna erinevad käsud võivad anda veidi erinevaid õigeid lahendusi. Selle asemel tuli piisavalt täpselt, kuid samas piisavalt üldiselt kontrollida tulemust, mida konteinerist vastu saadeti. Seega tuli kontrollida võtmesõnu üldisemalt. Minnes liiga üldiseks, võib tekkida olukord, kus kogemata lahendatakse mitu ülesannet ühekorraga. Selle vältimiseks pidi ülesanded sõnastama nii, et oleks piisavalt selge mida küsitakse. Automaattesti koostamise keerukuseks oli erinevate õigete lahenduskäikude õigeks lugemine ning valede lahenduskäikude eiramine.

4.2 Veebiliidese disain

Lähtudes rakenduse funktsionaalsetest nõuetest sai arendustöö käigus loodud järgnev veebikeskkond. Veebileht on kättesaadav Tartu Ülikooli sisevõrgust aadressil http://172.17.37.144:80. Järgnevates alapeatükkides on toodud rakenduse visuaalse poole kirjeldus koos illustreerivate piltidega.

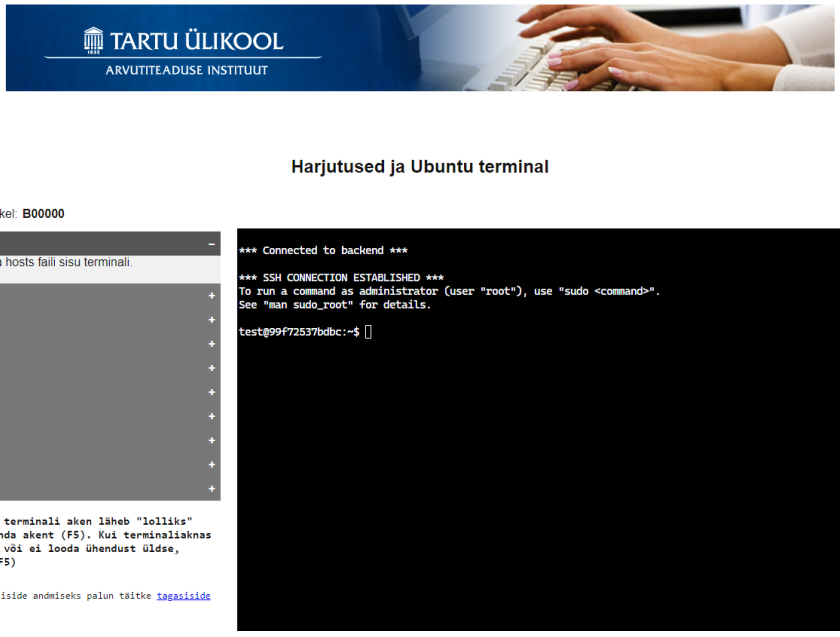
4.2.1 Esilehekülje vaade

Esileheküljel on toodud Tartu Ülikooli arvutiteaduse instituudi logo, lehekülje pealkiri koos tutvustava tekstiga ning registreerimisinfo ja sisselogimise kast. (vt Joonis 2). Registreerimisinfo kohal on toodud tagasiside andmise link.

Sisselogimiseks on kaks võimalust. Kui vajutada „Jätka külalisena“ nuppu, tehakse külaliskasutaja ning nime ja matrikli lahtrid ei pea olema täidetud. Kui vajutada „Logi sisse“ nupule, tuletatakse kasutajale meelde, et esmalt peab täitma nime ja matrikli lahtrid. Kui kasutaja on korrektselt matrikli numbri ja nime lahtrid täitnud (vastavalt registreerimisinfos antud juhendile), saab ka nupule „Logi sisse“ vajutades ülesandeid lahendama hakata.

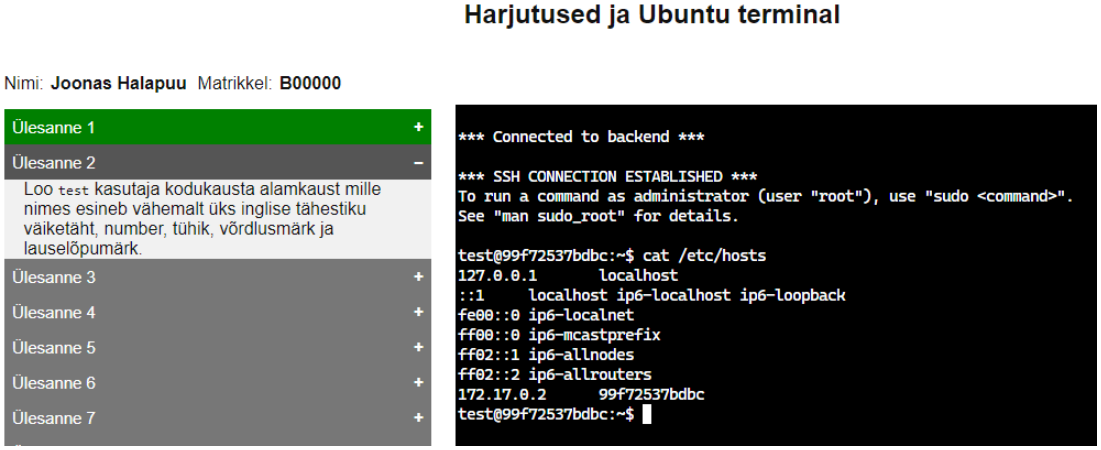
4.2.2 Ülesannete lehekülg

Sisselogimise järel kuvatakse ülesannetelehe vaade (vt Joonis 7). Ülesannete lehel on samuti toodud Tartu Ülikooli arvutiteaduse instituudi logo. Lisaks sellele on lühike lehekülje pealkiri ning selle alla jäävad ülesanded koos terminali aknaga. Terminali aken asub lehekülje paremal pool ning sellest vasakul on ülesanded. Ülesannete kohal asub sisselogitud kasutaja puhul kasutaja sisestatud nimi ning matrikli number. Iga ülesannet saab avada ning sulgeda eraldi. Esimest korda lehele tulles avatakse automaatselt esimene lehekülg. Terminaliaknas on kirjutatud, et nii Socket.io ühendus (ühendus serveriga) kui ka SSH ühendus (serveri ühendus konteineriga) on olemas. Kasutaja saab terminaliaknale klõpsates alustada ülesannete täitmist.



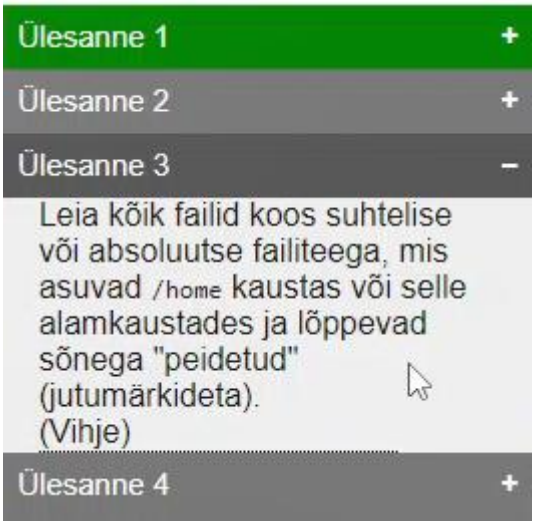
Joonis 7. Ülesannete lehekülg

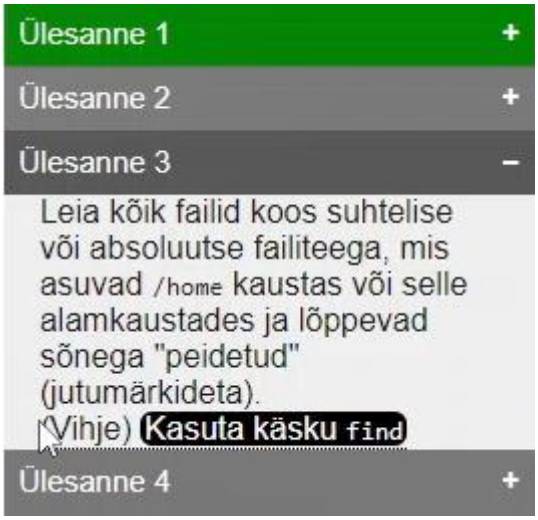
Pärast ülesande lahendamist värvub vastava ülesande riba intuitiivse värviga, millest kasutaja aru saab, et ülesanne on lahendatud. Selleks värviks on roheline. Samuti sulgub lahendatud ülesanne automaatselt ning avaneb järgmine ülesanne (vt Joonis 8).



Joonis 8. Lahendatud ülesande vaade

Teatud ülesannete kirjelduste juurde käivad vihjed. Need vihjed on abiks ülesannete lahendajatele, kes on hätta jäänud. Kuna vihjete eesmärk ei ole ülesannete lahendamist liiga lihtsaks teha, on vihjed ülesande avamise hetkel peidus ning avanevad alles siis, kui kasutaja on kursori viinud vihje peale (vt Joonis 9 ja Joonis 10).





4.3 Installatsiooni juhend

Järgnevalt on toodud rakenduse installatsiooni juhend:

1. Paigaldada ja seadistada Ubuntu 20.04 arvuti operatsioonisüsteemiks.
2. Uuendada tarkvara – software updater'iga ja käsuga `sudo apt Update`.
3. Laadida alla järgnevad tarkvarad näiteks käsuga `sudo apt install` <tarkvara nimi>:
    a. git;
    b. nodejs;
    c. npm.
4. Tõmmata koopia koodihoidlast (ingl *source code repository*) käsuga `git clone git@gitlab.com:JoonasHalapuu/ubuntuterminal.git` sobivasse kausta.
5. Katsetusharu versiooni valimiseks käivitada käsk `git checkout ExperimeintWithVue` lähtekoodi kaustas.
6. Laadida alla rakenduse käivitamiseks vajalikud teegid käivitades käsu `npm install` allatõmatud hoidla kaustas.
7. Dockeri installeerimisjuhendit järgides installida Docker. Dockeri installeerimisjuhend: Install Docker Engine on Ubuntu | Docker Documentation.
8. Seadistada *sudoless* Docker selleks, et Node.js saaks käivitada konteinereid ülemkasutaja (ingl *superuser*) õigusteta. Juhend *sudoless* Dockeri seadistamiseks: Post-installation steps for Linux | Docker Documentation.
9. Luua .env fail allatõmmatud koodiga samasse kasuta, kuhu lisada lokaalse arvuti IP aadress kujul `HOST = "[ip aadress]"` ning sama IP aadress ka failidesse `startingPage.js` ja `terminalDisplay.js` algusesse konstandi `HOST` väärtuseks.

10. Olenevalt Inotify vaikeväärtustest saab rakendust korraga kasutada kuni 5 kasutajat. Inotify instantside suurendamiseks käivitada järgnevad käsud:
    a. `sysctl -n -w fs.inotify.max_user_instances=540`;
    b. `sysctl -n -w fs.inotify.max_user_watches=15600384`.
11. Node.js'il port 80 kuulamise võimaldamiseks Ubuntus käivitada järgnevad käsud:
    a. `sudo apt-get install libcap2-bin` ;
    b. `sudo setcap cap_net_bind_service=+ep` `` `readlink -f \`which node\` ``.
12. Tartu Ülikooli võrgus VPN'iga lehekülje ligipääsetavaks tegemiseks tuleb Docker panna mõnele teisele IP aadressile. Selleks käivitada järgmised käsud:
    a. Lisa faili `/lib/systemd/system/docker.service` rea `ExecStart=/usr/bin/dockerd` lõppu `--bip "172.80.0.0/16"`.
    b. Seejärel käivita järgnevad käsud:
        i. `systemctl daemon-reload`;
        ii. `systemctl start docker`.
13. Käivitada rakendus käsuga `node api.js` olles alla tõmmatud koodi peakaustas.

Seejärel saab kasutada rakendust minnes arvuti IP aadressile, kus rakendus käivitati. Täiendavalt tuleb seadistada veebiserver lubama rakenduse toimimiseks vajalikke porte, milleks on pordid 80/tcp, 8080/tcp ning 5000/tcp. Pilvekeskkonnas ülesseadmiseks tuleb teha läbi samad käsud, kuid soovitatav on käivitada katsetusharus olev versioon, vastasel juhul peaks pilvekeskkonnas avama pordid vahemikus 49152–65535. Eelneva juhendi põhjal seadistatud rakenduse testimist kirjeldatakse järgnevas peatükis.

5. Rakenduse testimine ja analüüs

Järgnevas peatükis tehakse ülevaade töö käigus valminud rakenduse testimise läbiviimisest. Lisaks kirjeldatakse testimisel saadud tulemusi. Seejärel võetakse saadud tulemused kokku, tehakse nende põhjal järeldusi ning uuritakse edasise arengu võimalusi.

5.1 Testimise läbiviimine

Testimise ajal töötas rakendus Tartu Ülikooli sülearvutil HP elitebook 840 G8, millel on 16 GB mälu ning 512 GB SSD salvestusruumi. Suurima koormuse juures (umbes 50 kasutajat) polnud mälu kasutus üle 4GB, seega oleks tõenäoliselt piisanud 8 GB mälust ning 32 GB salvestusruumist.

Rakendust testiti aktiivselt vahemikus 7. veebruar kuni 23. veebruar 2022. Pärast seda jäi server käima ning testimine jätkus väiksemal koormusel. Aktiivse testimise ajavahemikul kasutas rakendust 109 inimest Tartu Ülikooli kursuse „Andmeturve“ (LTAT.06.002) esimese praktikumi[10] raames. Testimisel parema ülevaate saamiseks koguti tudengite ülesannete sooritamiseks kuluvat aega, ehk pärast iga ülesande lahendamist salvestati ülesande eduka lahendamise ajahetk. Rakenduse testimiseks järgisid tudengid sama sündmuste jada, mis oli kirjeldatud peatüki 2.3 põhistseenides.

5.2 Testimise tulemused

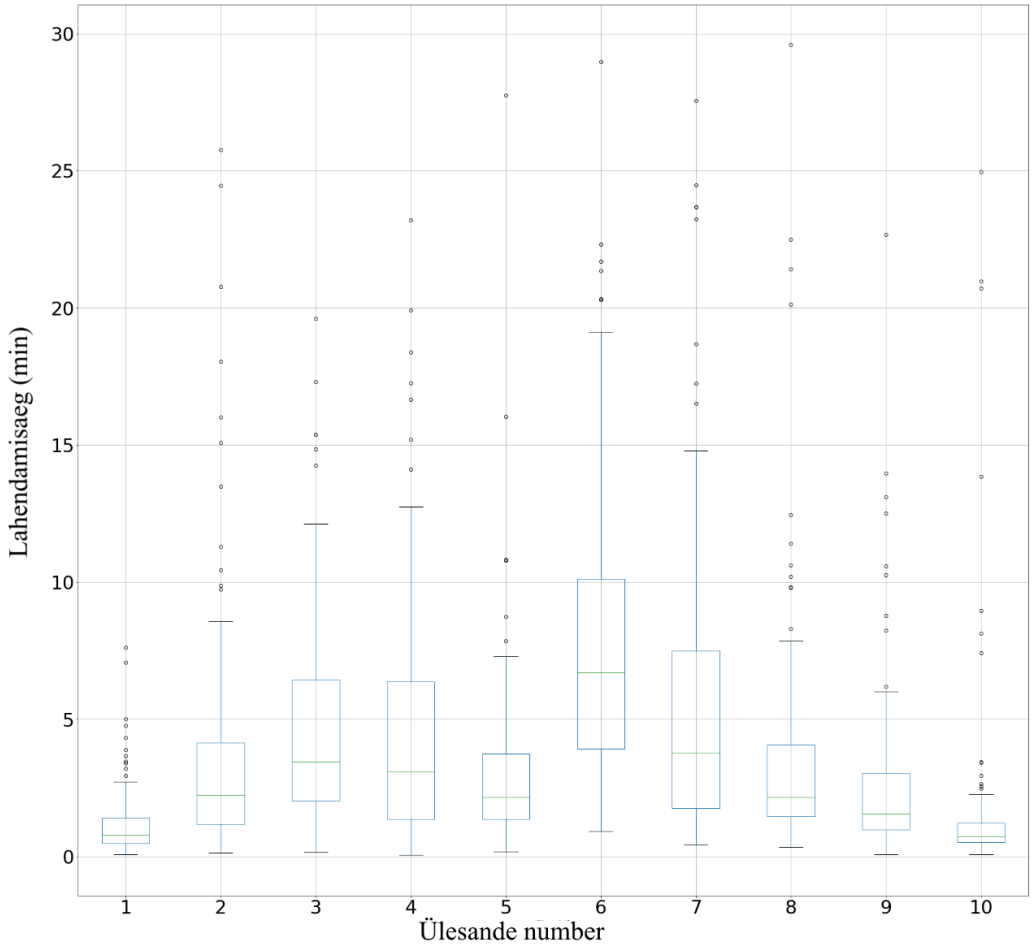
Esimesel nädalal esines veebiserveri töös katkestus, kuna aku oli tühjaks saanud arvutil, millel töötas veebiserver (inimlik viga). Järgnevatel kuudel sellist katkestust ette ei tulnud, kuid umbes korra kuus esines siiski katkestusi rakenduse töös erinevatel põhjustel (näiteks USB võrgukaart jooksis kokku või Linux ise hangus). Rakenduse töö taastamiseks piisas sellistel juhtudel demoserveri taaskäivitamisest. Aktiivsel testimisperioodil oli testarvutis 40 töötavat konteinerit ning sellise koormuse juures oli kogu arvuti mälukasutus alla 4 GB.

Ülesannete osas tekitasid tudengite seas segadust ebapädevad automaatkontrollid ja ülesannete liiga üldine kirjeldus. Samuti tuli testimise käigus välja, et veebilehele ei saa ligi tudengid, kelle arvutis töötab rakendus Docker. Dockeri alamvõrguks (ingl *subnet*) on vaikimisi 172.17.0.0 [18] ning kuna Tartu Ülikooli sisevõrk on samas alamvõrgus, ei ole võimalik Ülikooli sisevõrgus olevale veebiserveritele ligi pääseda. Alamvõrkude kattuvus on ka muudes õppeainetes esinev levinud probleem, mille lahenduseks on tudengi arvutis Dockeri alamvõrgu vaikeväärtuse muutmine.

Google vormiga koguti tagasisidet erinevate vigade ja üldise kasutusmugavuse kohta. Google vormi tuli vastuseid 13. Testimise ajahetkel oli osa kasutusmugavusest puudulik (vt lisa IV), seetõttu kirjeldati tagasisides ülesannete lahendamisel tekkinud ebamugavusi. Kuigi veebiliideses leiti testimisel paar viga, toimis veebirakendus üldiselt edukalt ning nii tudengid kui õppejõud kiitsid valminud Linuxi õpikeskkonda.

[10] Aine „Andmeturve“ esimese praktikumi juhend: https://courses.cs.ut.ee/2022/turve/spring/Main/Praktikum1. Töös kirjeldatud keskkonda tuleb kasutada ülesandes 3.

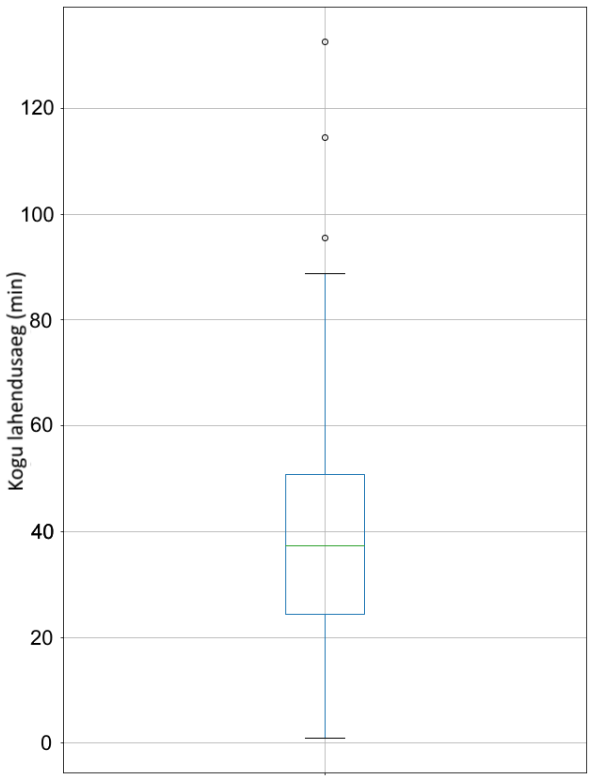
Ülesannete edukal lahendamisel kogutud ajahetki uurides (vt Joonis 11) selgub, et ajaliselt kõige mahukamateks ülesanneteks osutusid kaheosalised ülesanded ehk sellised ülesanded, mille tegemiseks tuli teha mitu tegevust. Nendeks ülesanneteks olid 6. ja 7. ülesanne[11]. Ülesande nr 6 jõudsid edukalt lõpuni teha 95 tudengit. Sellele ülesandele lisas keerukust üsna habras automaatkontroll. Nimelt ei tuvastanud automaatkontroll õiget lahendust, kui pool ülesandest oli tehtud ühe SSH ühendusega ning ülejäänud osa teise SSH ühendusega (näiteks erinevates arvutites, veebilehitseja akendes või lehekülge uuendades).



Joonis 11. Ülesannete ajaline kulu

Tudengid lahendasid ülesandeid keskmiselt (mediaanväärtus) 40 minutit (toodud Joonis 12). Selgus, et 50% lahendajatest lahendas ülesandeid kuni 38 minutit, samas kui 75% tudengitest lahendas ülesandeid kuni 51 minutit. Arvestades praktikumi kestvuseks 90 minutit, lahendas keskmine tudeng valminud keskkonnas ülesandeid umbes pool praktikumi ajast.

[11] Ülesannete kirjeldused on toodud lisas III.



Joonis 12. Ülesannete lahendusaeg tudengite kohta kokku

Tagasiside küsimisel õppejõududelt, kes praktikume parandavad selgus, et nende arvates oli rakenduse kasutamine kiire ja konkreetne. Parandajate jaoks lihtsustas rakendus tööd ning kiirendas parandamiseks kuluvat aega oluliselt.

5.3 Rakenduse analüüs

Valminud rakendus läbis aktiivse testimise edukalt ning oli kasutuskõlbulik. Mitmed testimise käigus tekkinud vead parandati enne aktiivse testimisperioodi lõppu. Peatükis 3.1 püstitatud 13 funktsionaalset nõuet said täidetud ning rakenduse testimisel edukalt testitud. Mittefunktsionaalseid nõudeid testiti samuti rakenduse testimise käigus. Kuigi testimisperioodil 100 kasutajat paralleelselt rakendust ühelgi ajahetkel ei kasutanud, oli veebiserveril jõudluse varu ka eeldatavasti 100 aktiivse kasutajaga töötamiseks.

Rakenduse arendusprotsessi käigus oli algselt põhirõhk rakendusserveri arendamisel. Seepärast on rakendusserver veebiliidesest viimistletum. Rakenduse aktiivselt testitud versioonis luuakse uuele pordile iga konteiner ning veebileht, mis konteineriga ühendatakse. Kuigi konteinerite puhul on üks port konteineri kohta üsna minimaalne, pole selline portide kasutus veebilehtede jaoks vajalik. Rakenduse veebiliidesele peaks päringute tegemine käima üle ühe pordi, seega tuleks iga veebilehe jaoks uue pordi kasutamise asemel kasutada ainult ühte porti. Ühel pordil veebilehtede hoidmine on saavutatav dünaamiliselt alamlehtede genereerimise või üheleherakenduse (ingl S*ingle-page Application*) kasutamisega. Selline uuendus võimaldab poole vähem porte kasutada. Sellest tulenevalt alustati pärast aktiivset testimist rakenduse veebiliidese üheleherakenduseks muutmisega (link üheleherakendusega lähtekoodile on lisas II). Üheleherakendusi saab luua raamistike abiga. Nagu peatükis 3.1.1

selgus, on üheleherakenduse loomiseks ning töös käsitletud veebiliidese arendamiseks sobilik JavaScripti raamistik Vue. Lisaks üheleherakenduseks muutmisele aitab Vue peita JavaScriptis esinevate ülesannete automaatkontrollide sisu, kuna raamistik kompileerib Vue rakenduse JavaScriptiks.

5.4 Edasine arendus

Järgnevalt on kirjeldatud lõputöö skoobist suure mahu tõttu välja jäänud edasiarenduse võimalusi.

Ülesannete lisamise ja modifitseerimise lihtsustamiseks saaks luua selle jaoks veebiliidese. Käesolevas töös saab ülesandeid ja automaatkontrolle muuta ainult JavaScripti kirjutades, kuna need on püsikodeeritud (ingl *hardcoded*). Liides võiks võimaldada kiiremat ja mugavamat ülesannete modifitseerimist koos automaatkontrollidega. See aitaks kaasa uute ülesannete tekkele ning automaatkontrolli täiustamisele.

Veebiliides ülesannete muutmiseks vajaks andmebaasi olemasolu, kus muudetud ülesanded salvestatakse. Andmebaasi käesoleva töö käigus valminud rakenduses ei ole. Tehtud ülesanded salvestatakse lõppkasutaja veebilehitseja salvestusruumi *localStorage*. Andmebaasi kasutades saaks seal hoida nii ülesannete andmeid kui ka tudengite tehtud ülesannete andmeid. See võimaldaks mitmest erinevast arvutist sama kasutajaga sisselogides näha täpselt sama ülesannete seisu. Töö käigus valminud rakenduses kuvatakse ainult need ülesanded tehtud seisus, mis selles veebilehitsejas valmis tehti. Lisaks annaks andmebaasi kasutamine rakendusele stabiilsust juurde. Lõputöö käigus valminud versioonis ei ole kaheosalise kontrolliga ülesannet võimalik teha poole kaupa. See tähendab, et tehes esimese osa ning seejärel lehte uuendades ei jäta süsteem meelde, et esimene pool sai tehtud. Selle tulemusena teise osa lahendamisel ei näita õpekeskkond ülesannet lahendatuna.

Konteineritega SSH ühenduse loomist saab muuta turvalisemaks salajaste ja avalike võtmete kasutamisega. Käesolev rakendus kasutab SSH üheduseks kasutaja nime, parooli ning porti, seega saab igaüks SSH ühendust luua, kellel on kasutaja nimi, parool ja pordi number teada.

Sisselogimisel ei kontrollita matrikli numbri olemasolu õppeinfosüsteemis, kuid sellise kontrolli võimaldaks TÜ SSO (ingl *Single-Sign-On*) või LDAP (ingl *Lightweight Directory Access Protocol*) sisselogimine.

Lõputöö käigus valminud rakendus vähendab oluliselt praktikumide parandamise aega, kuid on võimalik tööde parandamist veelgi kiirendada luues Moodle'ga sünkroniseerimise. See tähendaks, et kui tudeng on ülesande lahendanud, saadetakse Moodle'sse automaatselt teade, et antud üliõpilane on edukalt selle ülesande lahendanud. Seega ei peaks õppejõud seda osa praktikumist parandama, kuna see on täielikult automatiseeritud. Moodle'ga sünkroniseerimiseks seab käesoleva töö puhul piirangu õppejõudude eelistus kasutada courses.cs.ut.ee keskkonda ja hoiustada hindeid eraldi hindetabelis.

Kokkuvõte

Linuxi käsurea õppimiseks on olemas mitmeid inglise keelseid veebirakendusi ja -lehti. Mõned rakendused õpetavad teoreetilisi teadmisi samas teised võimaldavad lisaks ülesannetele kasutada käsurida. Ülesannete automaatkontrolli lahendus sellistel rakendustel enamasti puudub, samuti ei saa selliste lehekülgede kättesaadavuses ning stabiilsuses kindel olla. Lõputöö eesmärk oli luua automaatkontrolliga eestikeelne Linuxi käsurea harjutuskeskkond veebis, mille kasutamiseks piisaks veebilehitsejast ja internetiühendusest.

Veebirakenduse loomiseks pandi paika nõuded, mille järgi rakendust arendati. Rakenduse arendusel kasutati JavaScripti, HTML'i ja CSS'i veebiliideses ning Node.js'i rakendusserveris. Konteinertehnoloogiaks valiti Docker. Valminud rakenduse põhilisteks osadeks on konteinerite loomine, veebilehtede ühendamine konteineritega ning ülesannete kontrollimine. Veebiliides koosneb esilehest ning ülesannete lehest. Süsteemi kasutajad saavad esilehel luua nimelise või külaliskasutaja, seejärel luuakse kasutajale konteiner ning kasutaja suunatakse ülesannete lehele. Ülesannete leht ühendatakse läbi WebSocketi rakendusserveriga ning rakendusserver ühendatakse läbi SSH konteineriga. Ülesannete lehel on võimalik vaadata ülesandeid ning sisestada käsureale käske ülesannete lahendamiseks. Ülesannete automaatseks kontrollimiseks kasutatakse nii käskude väljundit kui ka failide muudatuste jälgimist. Eduka ülesande lahenduse korral värvitakse ülesanne roheliseks ning avatakse järgmine ülesanne.

Käesoleva bakalaureuse töö raames loodi Linuxi õppimise veebikeskkond, mis võimaldab õppuritel saada kohest tagasisidet ning kiirendab õppejõudude tööd. Valminud veebirakendusega on täidetud töö alguses koos aine õppejõuga seatud funktsionaalsed ja mitte-funktsionaalsed nõuded. Veebikeskkonda testiti veebruaris 2022 Tartu Ülikooli kursuse „Andmeturve" esimeses praktikumis 100 tudengi peal ning loodud keskkond vastab aine õppejõudude soovidele ja ootustele. Lõputöö raames valmid rakendus on valmis kasutamiseks ainete järgmistel toimumistel ning kasutav Tartu Ülikooli sisevõrgust aadressil http://172.17.37.144. Juhul kui veebileht ei ole kättesaadav eelnevalt väljatoodud aadressilt pöörduda juhendaja Alo Peetsi (alo.peets@ut.ee) poole sooviga kasutada veebilehte Linuxi õppimiseks.

Viidatud kirjandus

[1] Mrkonjić E. Linux Statistics: Market Share, Usage, and Fun Facts. [Veebimaterjal].; 2021 [vaadatud 2022 aprill 10. Kättesaadav aadressil: https://writersblocklive.com/blog/linux-statistics/.

[2] Vailshery LS. Software development operating system distribution globally 2018 to 2021. [Veebimaterjal].; 2022 [vaadatud 2022 Aprill 10. Kättesaadav aadressil: https://www.statista.com/statistics/869211/worldwide-software-development-operating-system/.

[3] Goldstein I. What! No GUI? -- Teaching a Text Based Command Line Oriented Introduction to Computer Science Course. Information Systems Education Journal. 2019 veebruar; 17(1): 40-41.

[4] Lakshmipathi , Freston , Elija , Blender3d. [Webminal]. [Veebimaterjal].; 2021 [vaadatud 2022 märts 30. Kättesaadav aadressil: https://www.webminal.org/.

[5] Quach C. https://linuxjourney.com/. [Veebimaterjal].; 2017 [vaadatud 2022 märts 30. Kättesaadav aadressil: https://linuxjourney.com/.

[6] Acker SV, morla. Bandit. [Veebimaterjal].; 2022 [vaadatud 2022 märts 30. Kättesaadav aadressil: https://overthewire.org/wargames/bandit/.

[7] Kakk M. Automaattestid andmebaaside ainele. 2020. TÜ arvutiteaduse instituudi bakalaureusetöö.

[8] W3Techs. Web Technology Surveys. [Veebimaterjal].; 2022 [vaadatud 2022 mai 8. Kättesaadav aadressil: https://w3techs.com/technologies/overview/client_side_language.

[9] Adobe. Adobe Flash Player EOL General Information Page. [Veebimaterjal].; 2021 [vaadatud 2022 aprill 10. Kättesaadav aadressil: https://www.adobe.com/products/flashplayer/end-of-life.html.

[10] Breedis R. Comparison of JavaScript User Interface Frameworks. 2021. TÜ arvutiteaduse instituudi bakalaureusetöö.

[11] Lei K, Ma Y, Tan Z. Performance Comparison and Evaluation of Web Development Technologies in PHP, Python, and Node.js. In IEEE 17th International Conference on Computational Science and Engineering; 2014; Chengdu.

[12] Challapalli SSN, Kaushik P, Suman S, Shivahare BD, Bibhu V, Gupta AD. Web Development and performance comparison of Web Development Technologies in Node.js and Python. In International Conference on Technological Advancements and Innovations; 2021; Tashkent.

[13] Bentaleb O, Belloum A, SA, El-Maouhab A. Containerization technologies: taxonomies, applications and challenges. The Journal of Supercomputing. 2022; 78(5): 1150-1154.

[14] Moravcik M, Kontsek M. Overview of Docker container orchestration tools. In Institute of Electrical and Electronics Engineers; 2020; Košice. p. 476-479.

[15] Canonical. Minimal Ubuntu, on public clouds and Docker Hub. [Veebimaterjal].; 2018 [vaadatud 2022 märts 29. Kättesaadav aadressil: https://ubuntu.com/blog/minimal-ubuntu-released.

[16] Eddy W, Iyengar J, Kohler E, Kojo M, Lear E, Nishida Y, et al. Service Name and Transport Protocol Port Number Registry. [Veebimaterjal].; 2022 [vaadatud 2022 aprill 11. Kättesaadav aadressil: https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml.

[17] Cotton M, Eggert L, Touch J, Westerlund M, Cheshire S. Procedures for the Management of the Service Name and Transport Protocol Port Number Registry. [Veebimaterjal].; 2011 [vaadatud 2022 aprill 11. Kättesaadav aadressil: https://www.rfc-editor.org/rfc/rfc6335.html#section-8.1.2.

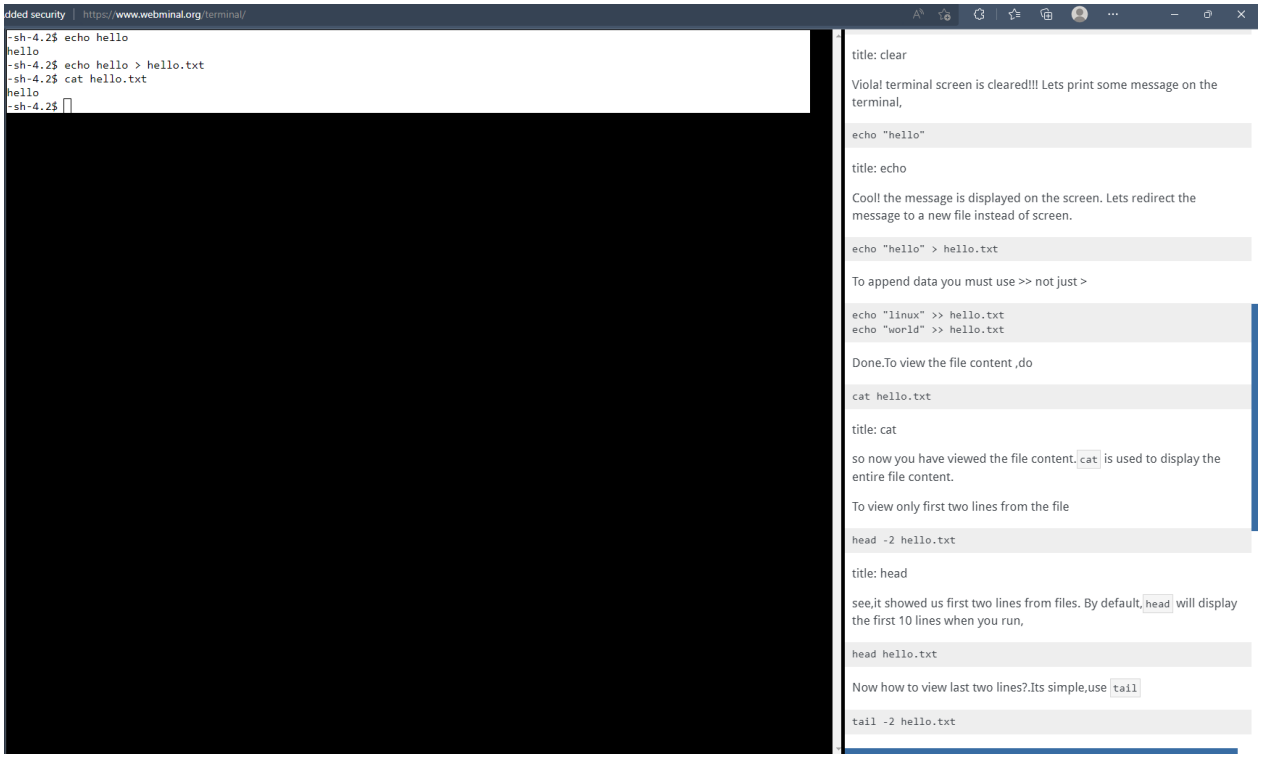
[18] Patel A. Medium. [Veebimaterjal].; 2021 [vaadatud 2022 mai 8. Kättesaadav aadressil: https://medium.com/codebrace/understanding-docker-networks-and-resolving-conflict-with-docker-subnet-ip-range-bfaad092a7ea.

[19] The info valley. What Is Linux: An Overview of the Linux Operating System. [Veebimaterjal].; 2020 [vaadatud 2022 aprill 10. Kättesaadav aadressil: https://medium.com/@theinfovalley097/what-is-linux-an-overview-of-the-linux-operating-system-77bc7421c7e5.

Lisad

I. Webminal'i käsurea lehekülg

(https://www.webminal.org/terminal/)



II. Lähtekood

Lõputöö käigus valminud rakenduse lähtekood on kättesaadav aadressil https://gitlab.com/JoonasHalapuu/ubuntuterminal

Edasiarendustega ning raamistiku Vue.js kasutav vähem testitud versioon on leitav aadressilt https://gitlab.com/JoonasHalapuu/ubuntuterminal/-/tree/ExperimeintWithVue.

III. Õpikeskkonna ülesanded vihjetega

Järgnevalt on toodud rakenduses kasutatud ülesannete tekstid ning vihjed, mis ülesannete juurde käisid. Ülesanded töötati välja koos lõputöö juhendajaga.

Tabel 1. Töös kasutatud ülesanded vihjetega

Ülesande number	Ülesande tekst	Ülesande vihje
1	Liigu kausta `/etc` ning kuva `hosts` faili sisu terminali.	kasuta käsku `cat`
2	Loo `test` kasutaja kodukausta alamkaust mille nimes esineb vähemalt üks inglise tähestiku väiketäht, number, tühik, võrdlusmärk ja lauselõpumärk.	-
3	Leia kõik failid koos suhtelise või absoluutse failiteega, mis asuvad `/home` kaustas või selle alamkaustades ja lõppevad sõnega "peidetud" (jutumärkideta).	Kasuta käsku `find`
4	Leia fail `/home` kaustast või alamkaustast mis sisaldab teksti `parool` (Jäta parool meelde, sul läheb seda hiljem vaja!).	-
5	Paigaldage tarkvara nimega `nano`. Leidke, kus asuvad rakenduse `nano` binaarid	`sudo apt ...` , `whereis`, sudo parooli leidsite eelmises punktis
6	Loo kasutaja `test` kodukausta fail nimega andmeturve ja faili sisuks pane oma nimi (täpitähtede asemel kasuta näiteks numbreid või teisi tähti). Muuda faili õigusi nii, et faili omanik (`test`) ja faili grupp (`root`) saaks faili kirjutada ja lugeda aga mitte käivitada ja teistele kasutajatele oleks keelatud failile ligipääs. Enesekontrolliks ja automaatkontrolli käivitamiseks käivitage käsk `ls -la /home/test/andmeturve`	-
7	Laadige alla lehekülg `https://courses.cs.ut.ee/MTAT.TK.012/2015_fall/uploads/Main/anton_hansen_tammsaare_tode_ja_oigus_i.txt`. Leidke **ainult** allatõmmatud failis esinevate **sõnade** arv.	vaja läheb käske `wget` ja `wc` . Uurige `wc` lisaparameetreid.
8	Loo oma eesnimega kasutaja kelle kodukaustaks on `/home/[eesnimi]`.	-
9	Kustutage kaust `/home/test/.ajutine` ja selle sisu.	-
10	Kuvage kõik käimasolevad protsessid.	uurige käsku `ps`

IV. Testimise käigus tekkinud vead ja nende parandused

Exit käsk

Viga: exit käsk tekitas veateate, kui sellega lõpetada SSH ühendus. Veateadet ei püütud kinni ning veebiserver jooksis kokku.

Põhjus: WebSocket oli endiselt ühendatud ning saatis signaale edasi, kuigi SSH ühendus oli lõppenud, seega polnud konteinerit, millele neid käske edastada.

Lahendus: Peale SSH ühenduse sulgemist (nt exit käsu tõttu) sulgetakse ka WebSocketi ühendus. Huvitavaks tegi vea väljatulemine Ubuntu serveril, kuigi WSL'is (ingl *Windows Subsystem for Linux*) sellist viga ei esinenud.

***Inotifywait* protsess**

Viga: *Inotifywait* protsess ei läinud pärast ühenduse katkestamist kinni või ei käivitunud üldse. *Inotifywait* protsess on vajalik failimuudatuste jälgimiseks. Kui *Inotifywait* ei tööta, ei tööta ka osad automaatkontrollid.

Põhjus: *Inotify* jälgimis protsessidele on limiidid. Kui limiit saab täis, ei looda uusi protsesse.

Lahendus: rekursiivse kausta jälgimise asemel (ehk vaadatakse kõiki kaustas olevaid kaustu, nende alamkaustu jne) jälgida ainult paari kindlat kausta ning ülesannete automaatkontroll selle põhjal ümber teha. Lisaks tuleb pärast WebSocketi ühenduse sulgemist sulgeda ka SSH ühendus koos ühenduse jaoks loodud *Inotifywait* protsessidega.

unminimize käsk

Viga: mitmel arvutil unminimize käsu jooksutamine kasutab suurt osa veebiserveri jõudlusest paar minutit.

Põhjus: manuaalilehekülgede vaatamiseks tuli konteiner esmalt tavaolekusse viia, kuna esialgses versioonis oli konteiner minimaalsel kujul. Unminimize käsk kasutas liigselt protsessorit ning mälu.

Lahendus: esialgu keelati lihtsalt unminimize käsk ära, kuid töö hilisemas käigus selgus, et konteinereid ei ole vaja hoida minimaalsel kujul ning kui kõik konteinerid on algusest peale tavaversioonis, kasutavad nad peaaegu sama palju ruumi kui minimaalsel versioonil konteinerid. Seega oleksid manuaalileheküljed koheselt saadavad ning unminimize käsku jooksutamine ei teeks midagi.

Regulaaravaldis Kuna automaatkontrollid põhinevad regulaaravaldistega ülesannete õige lahenduse ära arvamises siis on tulnud juhuseid, kus regulaaravaldis on ära arvanud mitu erinevat ülesannet korraga või pole seda üldse teinud.

Terminaliaken

Viga: terminaliakna read hakkavad üksteist üle kirjutama selle asemel, et järgmiselt realt alustada.

Põhjus: terminaliakna ja SSH ühendused polnud seadistatud samade mõõtmetega. Sellest tulenevalt võis SSH ühendus alustada järgmisele reale kirjutamist samas kui terminaliaken oli poole rea peal.

Lahendus: SSH ühenduse ja terminaliakna ridade ja veergude arvu samaks muutmine.

Vale kasutaja nimi ja matriklinumbri kuvamine

Viga: Uue kasutaja tegemisel näidati eelneva kasutaja nime ja matrikli numbrit.

Põhjus: Veebiserver hoidis GET päringute asukohad alles ka pärast restarti. Iga kord uut kasutajat luues luuakse ka uus lehekülg või kasutatakse vana (sel juhul tuleb kasutaja mitmendat korda). Kui server kinni pannakse peaksid veebilehed kustuma, kuid need jäävad alles. Pärast veebiserveri tagasi tööle panemist alustatakse algusest portide kontrolliga ning võetakse need mis pole juba kasutuses (konteinerite jaoks). Veebilehtede jaoks lihtsalt eeldatakse, et veebileht eksisteeris varem, siis see on õige veebileht.

Lahendus: algne lahendus oli uue konteineri loomisega teha uus lehekülg ning mitte vana lehekülge kasutada isegi kui see eksisteeris. Hilisemas töös katsetatakse Vue.js raamistikuga ühelehe rakenduse loomist.

Ülesannete sõnastus

Viga: ülesannete lahendamiseks kasutati käske ning lahendusi, mis polnud soovitud, kuid polnud täiesti valed lahendused.

Põhjus: ülesannete sõnastus oli ebaselge ning sellest tulenevalt kasutati näiteks `find` käsu asemel käsku `ls`.

Lahendus: vihjete lisamine ning selgem ülesannete sõnastus.

Zombi konteinerid

Viga: rakendusserveri taaskäivitamise järel jäid konteinerid tööle, kuid neid ei kustutatud pärast etteantud aega.

Põhjus: taimer, mis ettemääratud aja pärast peaks konteineri kustutama, peatatakse veebiserveri peatamisega, kuid pärast veebiserveri käivitumist luuakse uus taimer, mis ei tea, millal varem olemas olnud konteinerid tuleks kustutada.

Lahendus: rakendusserveri käivitamisel tuleb varem eksisteerinud konteinerid lisada taimeri kustutusnimekirja.

V. Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, Joonas Halapuu,

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose **Eestikeelse veebipõhise Linuxi käsurea õpikeskkonna loomine**, mille juhendaja on **Alo Peets**, reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada digitaalarhiivi DSpace kuni autoriõiguse kehtivuse lõppemiseni.

2. Annan Tartu Ülikoolile loa teha punktis 1 nimetatud teos üldsusele kättesaadavaks Tartu Ülikooli veebikeskkonna, sealhulgas digitaalarhiivi DSpace kaudu Creative Commonsi litsentsiga CC BY NC ND 3.0, mis lubab autorile viidates teost reprodutseerida, levitada ja üldsusele suunata ning keelab luua tuletatud teost ja kasutada teost ärieesmärgil, kuni autoriõiguse kehtivuse lõppemiseni.

3. Olen teadlik, et punktides 1 ja 2 nimetatud õigused jäävad alles ka autorile.

4. Kinnitan, et lihtlitsentsi andmisega ei riku ma teiste isikute intellektuaalomandi ega isikuandmete kaitse õigusaktidest tulenevaid õigusi.

Joonas Halapuu

10.05.2022