TARTU ÜLIKOOL
Arvutiteaduse instituut
Informaatika õppekava

Taavi Eistre

Veebipõhise Linuxi käsurea õppekeskkonna tootestamine

Bakalaureusetöö (9 EAP)

Juhendaja: Alo Peets

Tartu 2024

Veebipõhise Linuxi käsurea õppekeskkonna tootestamine

Lühikokkuvõte:

Tartu Ülikooli õppeainetes „Operatsioonisüsteemid“ ja „Andmeturve“ on oluliseks käsitlusalaks Linux ja selle käsurida. Seoses vajadusega Linuxit paremini õpetada lõi Joonas Halapuu 2022. aastal oma bakalaureusetöö raames veebipõhise Linuxi käsurea õppekeskkonna prototüübi, mis oli neis õppeainetes kasutusel kaks aastat. Kasutamise käigus ilmnes mitmeid murekohti keskkonna töö stabiilsuses ning vajadusi õppekeskkonna edasiarenduseks. Käesoleva bakalaureusetöö eesmärk oligi olemasoleva keskkonna edasiarendus, täiustamine ning üleviimine pilvetehnoloogiale ehk tootestamine, et tagada parem stabiilsus, skaleeritavus ning ligipääsetavus.

Töös kirjeldatakse eelneva keskkonna puudusi ja potentsiaalseid edasiarenduse võimalusi, uue rakenduse nõudeid ja eesmärke, kasutatud tehnoloogiaid, valminud rakendust ning rakenduse pilvepõhise keskkonna testimise tulemusi. Bakalaureusetöö raames valminud veebipõhine Linuxi õppekeskkond on leitav ja kasutatav nii eesti kui inglise keeles aadressil https://lingid.ee/kasurida.

Võtmesõnad: Linux, käsurida, pilvetehnoloogia, Microsoft Azure, TypeScript, Kubernetes, Docker, MySQL

CERCS: P175 Informaatika, süsteemiteooria

Development of a Web-Based Linux Command-Line Learning Environment

Abstract:

Linux and its command line are one of the main topics of the University of Tartu courses “Operating Systems” and “Computer Security”. In order to improve the ways of teaching Linux, Joonas Halapuu created a web-based Linux command-line learning environment prototype as part of his bachelor’s thesis in 2022. This prototype was used in the courses for a duration of two years. Several concerns about the stability and the need for further development of the learning environment emerged during its use. The aim of this thesis was to further develop, improve and migrate the existing environment to cloud technology to ensure better stability, scalability and accessibility.

The thesis describes the shortcomings and potential enhancements of the previous environment, the requirements and objectives of the new application, the technologies used, the completed application and the results of testing the application in a cloud-based

environment. The web-based Linux learning environment developed as part of the thesis can be found and used in Estonian or English at https://lingid.ee/kasurida.

Keywords: Linux, command line, cloud computing, Microsoft Azure, TypeScript, Kubernetes, Docker, MySQL

CERCS: P175 Informatics, systems theory

Sisukord

Sissejuhatus

Tartu Ülikooli õppeainete „Operatsioonisüsteemid“ ja „Andmeturve“ oluliseks käsitlusalaks on Linuxi operatsioonisüsteem. Esimeste praktikumide raames tutvustatakse tudengitele Linuxi käsurida ning selle kasutusvõimalusi. Seoses õppejõudude sooviga Linuxi käsurida kvaliteetsemalt õpetada lõi Joonas Halapuu 2022. aastal oma bakalaureusetöö „Eestikeelse veebipõhise Linuxi käsurea õpikeskkonna loomine“ raames eestikeelse veebipõhise Linuxi käsurea õppekeskkonna prototüübi [1], mis on olnud kasutusel kaks aastat. Kasutamise käigus on prototüübis välja tulnud mitmeid murekohti seoses stabiilsuse, turvalisuse ning ajakohasusega. Täiendavalt on mitmed õppejõud huvitunud õppekeskkonna kasutamisest ka teistes õppeainetes nii eesti kui ka inglise keeles.

Käesoleva lõputöö eesmärk on Halapuu loodud Linuxi käsurea õppekeskkonna prototüübi tehnoloogiline täiustamine, kasutajaliidese parandamine, ingliskeelse versiooni lisamine, ülesannete modifitseerimisvõimaluse loomine, tulemuste talletamine ning kasutajakontode turvalisuse tagamine. Valminud rakenduse parema kättesaadavuse tagamiseks tuleks see üle viia pilvetehnoloogiale. Rakenduse pilves käitamine suurendaks selle stabiilsust ning võimaldaks seda vastavalt vajadusele skaleerida. Samuti oleks seda võimalik kasutada ka teistes õppeasutustes. Kõike eelnevat arvesse võttes kirjutas käesoleva töö autor enamiku eelnevalt loodud koodist ümber pilvetehnoloogia jaoks sobivamaid tehnoloogiaid kasutades ning oluliselt lisafunktsionaalsust lisades.

Bakalaureusetöö koosneb neljast põhilisest peatükist. Esimeses peatükis antakse ülevaade Linuxist ja selle olulisusest, Halapuu prototüübi probleemidest ja võimalikest edasiarendustest. Täiendavalt analüüsitakse käesoleva bakalaureusetöö juhendaja soove ja defineeritakse õppekeskkonna uued nõuded. Teises peatükis kirjeldatakse ja tutvustatakse rakenduse arendamisel kasutatud tehnoloogiaid ning nende valimise põhimõtteid. Kolmandas peatükis antakse ülevaade valminud rakenduse arhitektuurist ja selle tööprotsessidest. Viimasena kirjeldatakse pilvekeskkonda, mida kasutati veebirakenduse testimiseks, analüüsitakse testimise tulemusi ja kaasnevaid kulusid, kirjeldatakse andmebaasi migratsiooni ning tutvustatakse võimalikke edasiarenduse võimalusi.

1. Mõisted ja terminid

Käsurida (ingl *command line*) on tekstiline liides, kus käske sisestatakse läbi klaviatuuri ning hiirt ei kasutata[1].

Kasutajaliides (ingl *user interface*) on süsteem, mille abiga inimesed suhtlevad arvutisüsteemiga[1].

Server (ingl *server*) on riistvarasüsteem või programm, mis pakub teistele klientarvutitele teenust[1].

Konteiner (ingl *container*) on Dockeri[2] kettapildist saadud protsess, mis on teistest protsessidest isoleeritud ning mida saab jooksutada ükskõik millisel operatsioonisüsteemil[3].

Turvaline kest (ingl *Secure Shell*) ehk **SSH** on protokollisari, mis võimaldab turvalist ja krüpteeritud kaugpöördumist kaugarvutisse[1].

Port (ingl *port*) on liides kahe võrgu vahel suhtluseks[1].

Kettapilt (ingl *image*) on kujutis, mis sisaldab endas isoleeritud failisüsteemi ja vajalikke andmeid, mida on vaja konteineri jooksutamiseks[3].

WebSocket (ingl *WebSocket*) on protokoll, mis võimaldab kiiret kahepoolset sidet serveri ja veebilehitseja vahel[1].

HTTP (ingl *HyperText Transfer Protocol*) on rakenduskihi protokoll, mis määrab ära sõnumite vormingu ja edastusviisi serveri ja veebilehitseja vahel[1].

HTTPS (ingl *HyperText Transfer Protocol Secure*) on rakenduskihi protokolli HTTP turvalisem variant, mis loob transpordikihi protokolli SSL / TLS abil krüpteeritud kanali serveri ja veebilehitseja vahel[1].

Kapsel (ingl *pod*) on Kubernetese[4] kõige väiksem käsitletav üksus, mis sisaldab endas ühte või enam konteinerit, nendega seotud teenuseid ja jagatud andmeid[5].

Orkestreerimine (ingl *orchestration*) on arvutisüsteemide, vahetarkvara ja teenuste automatiseeritud korraldus, koordineerimine ja haldus[1].

---

[1] Seletus võetud Andmekaitse ja infortube leksikonist https://akit.cyber.ee.
[2] Konteineriseerimistarkvara Docker https://www.docker.com.
[3] Seletus võetud Dockeri dokumentatsioonist https://docs.docker.com/get-started.
[4] Orkestreerimistarkvara Kubernetes https://kubernetes.io.
[5] Seletus võetud Kubernetese dokumentatsioonist https://kubernetes.io/docs/concepts/workloads/pods.

Virtuaalmasin (ingl *virtual machine*) on tegeliku või hüpoteetilise arvuti arhitektuuri ja funktsioonide emuleering[6].

SQL (ingl *structured query language*) on deklaratiivne interaktsiooni- ja programmikeel, mida kasutatakse relatsiooniliste andmebaaside kasutamiseks ja haldamiseks[6].

Objekt-relatsioonvastendus (ingl *object-relational mapping*) ehk **ORM** on programmimehhanism, mis võimaldab andmebaasi andmeid töödelda objektipõhistes programmeerimiskeeltes[6].

Rakendusliides (ingl *application programming interface*) ehk **API** on reeglid ja vahendid rakendusprogrammidega suhtlemiseks[6].

CSS (ingl *Cascading Style Sheets*) on formaalkeel märgistuskeelse dokumendi välisilme kirjeldamiseks[6].

Regulaaravaldis (ingl *regular expression*) ehk **regex** on määratud tähistuse ja ehitusega avaldis, mida kasutatakse otsitava teksti malli ja käsitlusviisi spetsifitseerimiseks[6].

Parsimine (ingl *parsing*) on loomuliku või tehiskeele märgijadade analüüsimine mingi formaalgrammatika reeglite järgi[6].

Proksi (ingl *proxy*) ehk **vaheserver** on server, mis vahendab väliseid kliente ja sisemisi teenuseservereid[6].

Andmevoog (ingl *data flow*) on andmete edastuse, kasutamise ja muundamise jada arvutiprogrammi täitmise ajal[6].

Cron (ingl *cron*) on Linuxi utiliit ettemääratud tegumite perioodiliseks täitmiseks ettemääratud aegadel[6].

CPU (ingl *central processing unit*) ehk **keskprotsessor** on funktsionaalüksus, mis koosneb ühest või mitmest protsessorist ja nende sisemäludest[6].

[6] Seletus võetud Andmekaitse ja infortube leksikonist https://akit.cyber.ee.

2. Linuxi õppekeskkond ja loodud rakenduse nõuded

Käesolevas peatükis antakse lühiülevaade Linuxi operatsioonisüsteemist ning aktuaalsusest. Lisaks kirjeldatakse varasema prototüübi vigu, võimalikke edasiarenduse võimalusi ning tuuakse välja bakalaureusetöö vältel edasiarendatava rakenduse eesmärgid ning nõuded.

2.1 Linux ja selle aktuaalsus

The Linux Foundationi liikmete järgi on Linux vabavaraline avatud lähtekoodiga operatsioonisüsteem, mille esimene versioon ilmus 1990ndatel [2]. Linuxit kasutavad väga paljud seadmed, nagu näiteks nutitelefonid, võrguseadmed, autod, nutikad koduseadmed, personaalarvutid kui ka serverid. Linuxi populaarsuse kasvule on kaasa aidanud töökindlus, turvalisus ja avatud lähtekood, mis annab võimaluse seda vastavalt vajadusele modifitseerida. Eelnevast tulenevalt leidub Linuxil mitmeid variatsioone, millest populaarsemad on näiteks Ubuntu[7], Debian[8] ja Fedora[9]. Populaarseim mobiiltelefonide operatsioonisüsteem, Android[10], põhineb samuti Linuxil. Nad on toonud välja, et serveripõhised Linuxi süsteemid tulevad vaikimisi ainult käsurea põhise liidesega.

Graafilise kasutajaliidese puudumine tähendab, et peamine interaktsioon Linuxi baasil serverite ja inimeste vahel toimub läbi käsurea. Järelikult on käsurea oskus Linuxi populaarsuse tõttu serverite seas väga kasulik.

2.2 Halapuu prototüübi vead ja edasiarenduse võimalused

Joonas Halapuu tõi oma bakalaureusetöös välja, et tema prototüüpi saaks edasi arendada andmebaasi lisamisega [1]. See muudaks rakenduse stabiilsemaks, aitaks kasutajate tulemusi talletada sessioonide vahel ja annaks võimaluse lihtsamaks ülesannete modifitseerimiseks. Lisaks saaks Ubuntu konteineritega SSH ühenduse loomiseks kasutajanime ja parooli asemel kasutada turvalisemat salajast ja avalikku võtmepaari. Kontrollimata nime ja matrikli asemel pakuks Tartu Ülikooli poolne autentimine palju paremat verifitseerimise võimalust. Ta kirjutas, et kasutajaliidest võiks ümber viia näiteks Vue.js[11] raamistikule, mis annab võimaluse muuta rakendus üheleherakenduseks.

[7] Ubuntu Linux https://ubuntu.com.
[8] Debian Linux https://www.debian.org.
[9] Fedora Linux https://fedoraproject.org.
[10] Android operatsioonisüsteem https://www.android.com.
[11] JavaScripti raamistik Vue.js https://vuejs.org.

Siinse lõputöö juhendaja Alo Peets tõi välja, et Halapuu lahendus on vahelduva eduga mitu semestrit töötanud Tartu Ülikooli sülearvuti peal ja parem lahendus oleks rakendus tõsta pilve[12]. Internetist kättesaadavat keskkonda oleks tudengeil mugavam kasutada ja muudatused tooks endaga kaasa töökindlust ning stabiilsust. Pilvelahendusest sõltuvalt tekib võimalus vajadusel rakendusele dünaamiliselt rohkem ressurssi lisada. Rakendus peaks kuvama rohkem informatsiooni, kuna Halapuu prototüübis oli vähese informatsiooni tõttu vigade leidmine ja nende likvideerimine keeruline. Murekohaks on veel teiste kasutajate jaoks loodud konteineritele kerge ligipääs, kui asendada veebilehitseja aadressiribal oma sessiooni pordi number kellegi teise omaga. Probleemi likvideerimiseks tuleks lisada mehhanism, mis piiraks ligipääsu teiste kasutajate konteineritele. Õppekeskkonna vastu on huvi üles näidanud mitu Tartu Ülikooli õppejõudu, kes sooviksid seda kasutada erinevates eesti- ja ingliskeelsetes ainetes. Ta mainis, et parema kättesaadavuse tõttu peaks uuel rakendusel olema ka inglise keele valimise võimalus.

Halapuu prototüüpi uurides tuli välja, et selle töö käigus läks mitmel korral Dockeri kettapilti luues midagi valesti ja rakendus lõpetas töötamise. Lisaks kuvas rakendus vahel *MaxListenersExceededWarning* hoiatusteateid, mis pärineb rakenduses kasutatud SSH2[13] raamistikust, mida kasutatakse SSH ühenduste haldamiseks. Hoiatus tekib sellest, et rakendus registreerib ühele WebSocket sündmusele üleliigselt funktsioone. Eelnev tähendas, et keskkond vajas taaskäivitamist umbes iga kahe nädala tagant ning administraator pidi regulaarselt kontrollima, et keskkond pole kokku jooksnud. Lisaks oli varasem prototüüp kättesaadav ainult läbi HTTP protokolli, mis on ebaturvaline, ja Tartu Ülikooli sisevõrgu IP-aadressi. Parandatud versioon peaks kasutama turvalisemat HTTPS ühendust ning olema avalikust internetist nimelahendust kasutades kättesaadav.

2.3 Rakenduse parandamise eesmärgid

Analüüsides Halapuu loodud rakenduse puudusi, on edasiarenduse ehk käesoleva töö eesmärgid järgmised:

- Ajakohastada olemasolev prototüüp uuematele tehnoloogiatele ja lahendustele;
- Luua skript, millega õppejõud saaksid rakendust paigaldada ning rakendust lokaalses keskkonnas õppetöö põhiselt kasutada;
- Luua dokumentatsioon, et rakenduse kasutajad oskaksid lahendust käivitada ja hallata;

[12] Lõik pärineb eravestlusest lõputöö juhendaja Alo Peetsiga.
[13] SSH ühenduste JavaScripti raamistik SSH2 https://github.com/mscdex/ssh2.

- Viia rakendus üle pilvetehnoloogiale ning tagada selle skaleeritavus;
- Rakenduse pilve variandile implementeerida e-posti põhine autentimine;
- Rakenduse pilve variandis piirata turvalisuse tagamiseks e-posti domeene;
- Luua võimalus rakendust kasutada inglise keeles;
- Lisada võimalus ülesandeid lisada, muuta või kustutada;
- Lisada või korrastada olemasoleva lahenduse ülesandeid.

Linuxi veebipõhise õppekeskkonna tarkvara arendus ja tootestamine loetakse edukaks, kui kõik eelpool mainitud eesmärgid on täidetud.

2.4 Funktsionaalsed nõuded

Tuginedes Halapuu prototüübile ja uutele eesmärkidele, püstitati edasiarendatavale rakendusele funktsionaalsed nõuded, mis on välja toodud järgnevas loetelus:

1. Rakenduse kasutaja peab saama valida eesti ja inglise keele vahel;
2. Kasutajal peab olema võimalus saada valida heleda ja tumeda kasutajaliidese vahel;
3. Kasutaja peab saama sisestada oma sisselogimisandmeid;
4. Lokaalse rakenduse variandis peab kasutaja saama autentida oma nime ja parooliga;
5. Pilve variandis peab kasutaja saama autentida lubatud ja verifitseeritud e-posti aadressiga;
6. Kasutaja peab pilve versioonis saama e-posti teel verifitseerimiskoodi, mille abil ta saab oma konto aktiveerida;
7. Rakendus peaks säilitama inaktiivsete kasutajate andmeid vastavalt haldaja määratud ajalisele piirangule, mille möödumisel need automaatselt kustutatakse;
8. Kasutaja peab saama käsureale käske sisestada ja neid jooksutada;
9. Ülesannete kirjeldused peavad olema käsurea kõrval nähtavad;
10. Korrektselt lahendatud ülesanne peab muutuma roheliseks;
11. Kasutaja peab saama välja logida ja oma kontot kustutada;
12. Rakendus peab meelde jätma kasutaja tulemuse, et ta saaks vajadusel ülesandeid hiljem jätkata;
13. Igale kasutajale peab tekkima isiklik Ubuntu kapsel, mille abiga saab ta ülesandeid lahendada;
14. Rakendus peab automaatselt ära kustutama kapslid, mis on vastavalt haldaja poolt määratud aeg inaktiivsed olnud;
15. Kasutaja peab vajadusel saama katkise kapsli taaskäivitada;

16. Rakendus peaks pärast taaskäivitust suutma hallata ka eelmisest sessioonist alles jäänud kapsleid;
17. Kulude optimeerimiseks peaks pilve tõstetud rakendus inaktiivse klastri välja lülitama.

Eelpool mainitud funktsionaalsed nõuded on vajalikud, et tagada haldajatele ja kasutajatele töötav õppekeskkond.

2.5 Mittefunktsionaalsed nõuded

Uute eesmärkide ja Halapuu lahenduse järgi püstitati edasiarendatavale rakendusele järgnevad mittefunktsionaalsed nõuded:

1. Pilve tõstetud rakendus peaks olema kättesaadav 99% ajast;
2. Rakendus peaks ühilduma Ubuntu 22.04 või uuemaga;
3. Kasutajad peaksid saama verifitseerimiskoodi oma e-postile hiljemalt 10 sekundi jooksul;
4. Pilve tõstetud rakendus peaks korraga hakkama saama vähemalt 100 kasutajaga;
5. Rakenduse ja Ubuntu kettapildi suurused peaksid olema mõistlikult väikesed;
6. Turvalisuse ja ajakohasuse tagamiseks peaks Ubuntu kettapilt püsima värskendatuna;
7. Pilves oleva rakenduse ja kasutaja vaheline ühendus peaks olema krüpteeritud;
8. Kasutaja peaks ligi pääsema ainult tema jaoks loodud kapslile;
9. Pilve tõstetud rakendus peaks suutma välja lülitatud klastri taaskäivitada maksimaalselt 5 minutiga;
10. Töötava klastri puhul peaks kasutaja kapsel valmima maksimaalselt 1 minutiga.

Eelpool mainitud mittefunktsionaalsed nõuded on vajalikud, et tagada rakenduse töökindlus ja turvalisus.

3. Kasutatud tehnoloogiad ja lahendused

Järgmistes alapeatükkides antakse ülevaade rakenduse arenduseks kasutatud tehnoloogiatest ja lahendustest. Kirjeldatakse lähemalt valitud programmeerimiskeelest, raamistikest, konteinertehnoloogiatest, andmebaasidest ja pilvetehnoloogiatest.

3.1 Kasutajaliides ja server

Halapuu prototüübi arendamiseks oli kasutatud JavaScripti programmeerimiskeelt ja sellele tuginevat Node.js[14] raamistikku [1]. Käesoleva töö raames valminud rakenduse loomiseks on kasutatud TypeScripti[15]. Bogneri ja Merkeli järgi on TypeScript Microsofti poolt tehtud programmeerimiskeel, mis tuli välja 2012. aastal [3]. TypeScript on edasiarendus JavaScriptile, mis võimaldab kasutada ranget tüübisüsteemi ning mis kompileeritakse TypeScripti kompileerija abil tagasi JavaScriptiks, et seda oleks võimalik käivitada. Tüübisüsteem aitab ära määrata, mis tüüpi teatud muutujad, objektid ja funktsiooni argumendid olema peavad. See aitab omakorda ära hoida tüübi- ja programmivigu. Nende uuringu kohaselt on TypeScriptiga kirjutatud kood võrreldes JavaScriptiga parema kvaliteedi ja loetavusega.

Halapuu kirjutas oma lõputöös, et üheks edasiarenduse võimaluseks oleks prototüüp viia üle näiteks Vue.js raamistikule [1]. Siinse lõputöö rakenduse arenduseks on kasutatud Nuxt[16] raamistikku. Nuxti loojad on välja toonud, et Nuxt, täpsemalt Nuxt 3, on avatud lähtekoodiga raamistik, mis põhineb Vue.js ja Node.js raamistikel [4]. Võrreldes Vue.js'iga, mida kasutatakse tavaliselt kasutajaliideste loomiseks, täidab Nuxt nii kasutajaliidese kui ka serveri rolli. Nende väitel on Nuxt kindla projekti struktuuriga ning toetab vaikimisi TypeScripti ja serveripoolset renderdamist, mis toob kaasa lehe kiirema laadimise ja otsingumootorite jaoks optimeerimise.

Nuxt sobis ka siinse rakenduse arendamiseks hästi, kuna sellega on võimalik arendada nii kasutajaliidest kui ka serveri poolt ning kuna see võimaldab kasutada töö autorile varasemalt tuttavat Vue.js raamistikku.

3.2 Konteinerid ja nende orkestreerimine

Halapuu lahendus põhineb Dockeril ja konteineriseerimisel [1]. Pahl jt järgi on konteineriseerimine, mis on alternatiiviks intensiivsemale virtuaalmasinale, rakenduse

[14] Node.js JavaScripti raamistik https://nodejs.org/en.
[15] Programmeerimiskeel TypeScript https://www.typescriptlang.org.
[16] Vue.js'il põhinev Nuxt raamistik https://nuxt.com.

kergekaaluline virtualiseerimine läbi selle kokku pakitud kettapildi [5]. Nende sõnul on üheks populaarsemaks konteinerite tehnoloogiaks Docker, mida kasutatakse nii konteinerite loomiseks ja haldamiseks kui ka kettapiltide loomiseks. Paljude konteinerite käsitsi haldamine on väga tülikas ja aeganõudev. Selle lahendamiseks on välja töötatud tehnoloogiad, mis võimaldavad kasutajatel neid efektiivsemalt hallata ehk orkestreerida. Moravcik ja Kontsek on välja toonud, et hetkel ühed kõige populaarsemad konteinerite orkestreerimise tehnoloogiad on Docker Swarm[17], Kubernetes ja OpenShift[18] [6]. Docker Swarm on Dockeri enda kaubamärgi all arendatud, mistõttu on see väga lihtsalt ühilduv ülejäänud Dockeri keskkonnaga. Kubernetes, mis on nendest kolmest kõige populaarsem ja esialgselt Google'i poolt arendatud, on avaliku lähtekoodiga orkestreerimistarkvara. Võrreldes Docker Swarmiga on Kubernetes Dockerist eraldiseisev. OpenShift, mis on Red Hati poolt arendatud ja põhineb Kubernetesel, on ettevõtetele mõeldud orkestreerimistarkvara. Moravciki ja Kontseki väitel on OpenShifti lihtsam paigaldada ja kasutada, kuid see ei ole nii laialdaselt toetatud kui Kubernetes.

Rakenduse ja selles kasutatud Ubuntu kettapildi loomiseks kasutati konteinertehnoloogiana Dockerit, kuna see on üks enimtuntumaid konteinertehnoloogiaid. Konteinerite orkestreerimiseks osutus sobivaks Kubernetes, kuna see on laialdaselt pilveteenustes toetatud.

3.3 Andmebaas

Kasutajate tulemuste talletamiseks ja rakenduse stabiilsuse parandamiseks on oluline andmebaasi olemasolu. Poljak jt järgi on kolm enim kasutatud andmebaasi MySQL[19], PostgreSQL[20] ja Oracle[21] [7]. MySQL on kõige populaarsem avatud lähtekoodiga andmebaas, mille kiire ja stabiilse toimimise tõttu on see laialdaselt kasutusel näiteks e-kaubanduses. PostgreSQL on samuti avatud lähtekoodiga andmebaas, mis töötati välja California Ülikoolis. PostgreSQL toetab suurel hulgal SQL standardeid, nagu näiteks tabelite vahelised võõrvõtmed. Poljak jt on öelnud, et Oracle andmebaas, mis tuleb kaasa võimalusega luua uusi tabeleid ja andmebaase läbi veebilehitseja, on mõeldud ettevõtetele. Siinse rakenduse arendamiseks sobis oma kiiruse ja stabiilsuse tõttu hästi MySQL.

Andmebaasi ja rakenduse vaheliseks suhtluseks kasutati Prisma[22] raamistikku. Prisma kodulehel on välja toodud, et see on avatud lähtekoodiga objekt-relatsioonvastenduse

[17] Dockeri orkestreerimistarkvara Docker Swarm https://docs.docker.com/engine/swarm.
[18] Orkestreerimistarkvara OpenShift https://www.redhat.com/en/technologies/cloud-computing/openshift.
[19] MySQL andmebaas https://www.mysql.com.
[20] PostgreSQL andmebaas https://www.postgresql.org.
[21] Oracle andmebaas https://www.oracle.com/database.
[22] Objekt-relatsioonvastenduse JavaScript raamistik Prisma https://www.prisma.io.

raamistik, mis on mõeldud kasutuseks Node.js ja TypeScriptiga [8]. Prisma abiga on võimalik andmebaasi tabeleid käsitleda kui TypeScripti tüüpe, mis võimaldab ära hoida rakenduse ja andmebaasi vahelisi tüübivigu ning vigaseid SQL päringuid. Prisma kasutamine aitab ära hoida mitmeid andmebaasiründe id, nagu näiteks SQL süstimine (ingl *SQL injection*). Ka Halapuu katsetas Prismaga oma prototüübi koodihoidla eksperimentaalses harus [1].

3.4 Pilvetehnoloogia

Parema stabiilsuse, skaleeritavuse kui ka kättesaadavuse tagamiseks oli vaja viia rakendus üle pilvetehnoloogiatele. Saraswat ja Tripathi järgi on kolm enim kasutatud pilveteenuste pakkujat Amazon, Microsoft ja Google [9]. Amazon Web Services[23], lühidalt AWS, on üks esimesi ja vanimaid pilveteenuse pakkujaid, mis alustas 2006. aastal. Microsoft Azure[24], mis on kõige populaarsem tarkvara kui teenuse pakkuja (ingl *Software as a Service*), on sobilik nii täieliku kui ka hübriid pilvena. Saraswat ja Tripathi väitel on Google Cloud Platform[25], lühidalt GCP, üks kiireima kasvuga pilveteenusepakkujaid. Rakenduse jooksutamiseks sobis hästi Microsoft Azure keskkond, kuna Microsoft Azure pakub tudengitele iga aasta $100 krediiti tasuta pilveteenuste kasutamiseks ja katsetamiseks [10].

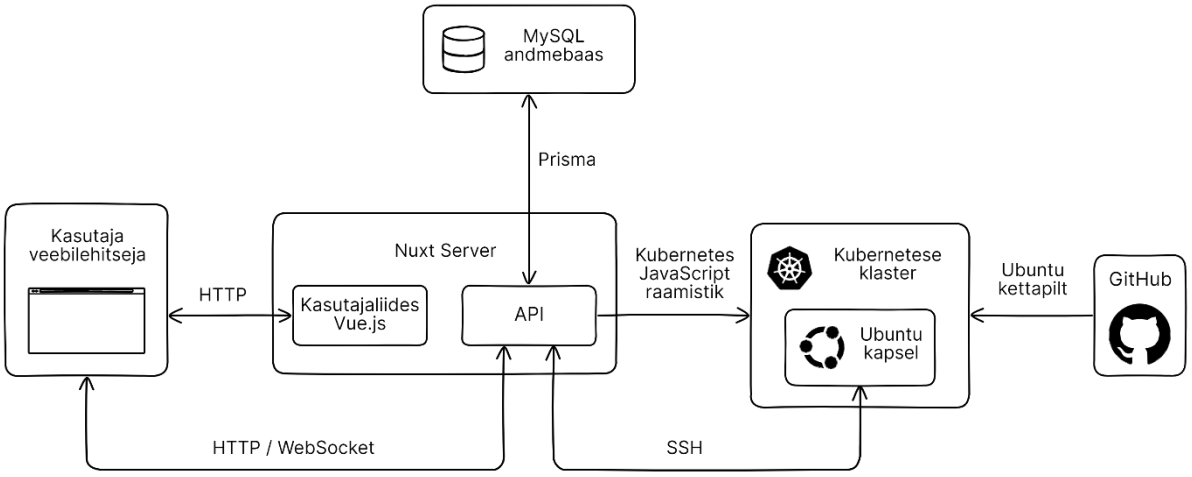
Pilvepõhiseks MySQL andmebaasi sobivaks pakkujaks osutus esialgselt PlanetScale[26]. PlanetScale on populaarne MySQL andmebaasi platvorm, mis põhineb Vitess[27] tehnoloogial [11]. PlanetScale sobis veebirakenduse pilvepõhiseks andmebaasiks hästi, kuna rakenduse arenduse ja testimise ajal pakkusid nad kuni miljard realugemist ning 10 miljonit reakirjutamist kuus tasuta [12][28]. Kuna rakendus ei ole andmebaasi kasutusega kuigi intensiivne ning rakenduse kasutajaskond ei ole väga suur, siis sobis PlanetScale hästi.

Käesoleva töö raames valminud veebirakenduse arendamiseks kasutati mitmeid tehnoloogiaid ja lahendusi. Rakendus kirjutati programmeerimiskeeles TypeScript, kasutades peamiselt Nuxt 3 raamistikku. Rakenduse poolt kasutatud Ubuntu kettapildi loomiseks on kasutatud konteinertehnoloogiat Docker ning nende orkestreerimiseks Kubernetest. Andmete talletamiseks on kasutatud MySQL andmebaasi. Lisaks kasutati veebirakenduse pilves käitamiseks Microsoft Azure teenuseid.

[23] Pilveteenuse pakkuja Amazon Web Services https://aws.amazon.com.
[24] Pilveteenuse pakkuja Microsoft Azure https://azure.microsoft.com.
[25] Pilveteenuse pakkuja Google Cloud Platform https://cloud.google.com.
[26] MySQL andmebaasi platvorm PlanetScale https://planetscale.com.
[27] Andmebaasi tehnoloogia Vitess https://vitess.io.
[28] Informatsioon oli kättesaadav 29.03.2024.

4. Valminud rakenduse arhitektuur

Valminud rakendus koosneb neljast peamisest osast, milleks on Kubernetese klaster, andmebaas, kasutajaliides ja serveri rakendusliides (joonis 1). Veebirakenduse keskpunktiks osutub rakendusliides ehk API, mis koordineerib suhtlust kasutaja, andmebaasi ja Kubernetese klastri vahel. Mainitud liides vastutab kasutaja poolt loodud päringute vastamise, kasutajate loomise ja haldamise, Kubernetese kapslite loomise ja kustutamise ning automaatkontrolli tulemuste salvestamise eest.



Joonis 1. Linuxi veebipõhise õppekeskkonna arhitektuur

Rakenduse peamine eesmärk on võimaldada kasutajal veebilehel lahendada käsurea ülesandeid isiklikus ja isoleeritud Linuxi konteineris, mille saavutamiseks loob rakendusliides igale kasutajale Kubernetese klastrisse oma Ubuntu kapsli. Suhtlus Kubernetese klastri ja rakendusliidese vahel on saavutatud Kubernetes JavaScript[29] raamistikuga.

Vastava Ubuntu konteineri kettapilt on kättesaadav rakenduse koodihoidlast[30], milleks on GitHub[31]. GitHub võimaldab koodihoidlas luua töövooge, mis mõne etteantud tegevuse peale rakenduvad. Vastavalt sellele on üles seatud töövoog, mis loob igal pühapäeval kell 00.00 Dockeriga uue rakenduses kasutatud Ubuntu konteineri kettapildi (joonis 2). See parandab rakenduse turvanõudlust, hoides loodavaid Ubuntu konteinerid värskendatuna.

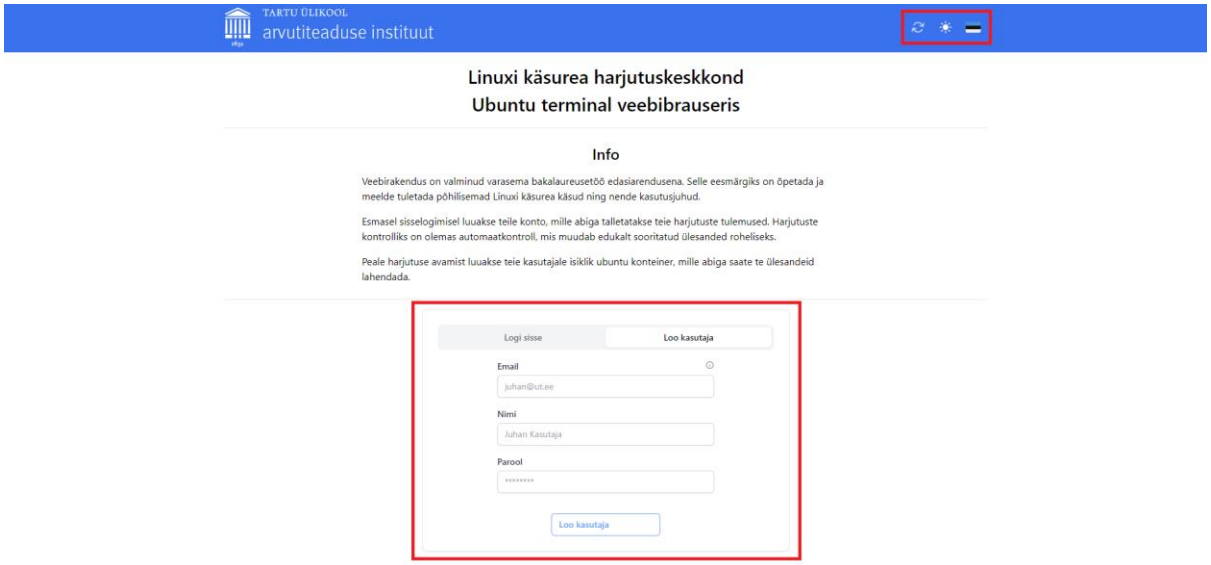
[29] Kubernetese JavaScripti liides https://github.com/kubernetes-client/javascript.
[30] Valminud rakenduse koodihoidla https://github.com/eistre/terminal.
[31] Koodihoidla platvorm GitHub https://github.com.

```yaml
name: Ubuntu image for Linux Terminal Application

on:
  push:
    paths:
      - './Dockerfile.ubuntu'
  schedule:
    - cron: '0 0 * * 0'
  workflow_dispatch:

jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - name: Docker login
        run: docker login --username eistre --password ${{ secrets.GHCR_TOKEN }} ghcr.io
      - name: Build the Ubuntu image
        run: docker build . --file Dockerfile.ubuntu --tag ghcr.io/eistre/terminal-ubuntu:latest
      - name: Push Ubuntu image
        run: docker push ghcr.io/eistre/terminal-ubuntu:latest
```

Joonis 2. Rakenduses kasutatud Ubuntu kettapildi loomise töövoo fail GitHubis

Ülesannete, kasutajate ja nende tulemuste talletamiseks kasutati MySQL andmebaasi. Rakendusliidese ja andmebaasi vaheliseks suhtluseks kasutati Prisma raamistikku. Lisaks sellele on kasutajaliides kasutatav nii eesti kui ka inglise keeles. Keelevaliku muutmiseks tuleks vajutada vastava keele lipuga tähistatud nuppu, mis asub veebilehe üleval paremal nurgas. Heledat ja tumedat varianti saab muuta nupuga, mis asub keelevaliku kõrval. Mõlemad nupud on välja toodud joonisel 3. Kasutajaliidese loomiseks on kasutatud Vue.js raamistikku ning Nuxt moodulit Nuxt UI[32] koos CSS raamistikuga Tailwind[33]. Rakenduse keele vahetamise võimalus eesti ja inglise keele vahel on saavutatud nuxt/i18n[34] mooduliga.

[32] Nuxti kasutajaliidese moodul Nuxt UI https://ui.nuxt.com.
[33] CSS raamistik Tailwind https://tailwindcss.com.
[34] Nuxti keelemoodul nuxt/i18n https://i18n.nuxtjs.org.



Joonis 3. Rakenduse avaleht koos konto loomise valikuga

Järgmistes alapeatükkides kirjeldatakse detailsemalt valminud rakenduse töökäike koos abistavate koodilõikude ja piltidega.

4.1 Kasutaja loomine ja verifitseerimine

Rakenduse käitamine pilvekeskkonnas on ressursikulukas ning võib tunduda ahvatlevana pahatahtlikele isikutele, mille tõttu on avalikust internetist kättesaadavale keskkonnale vajalik lisada ligipääsupiirangud. Enne veebirakenduse kasutamist on vaja luua kasutajakonto. Kontot saab luua kasutajaliidese esilehel (joonis 3). Kontode kasutamine annab võimaluse ülesannete tulemusi omistada ja salvestada. Samuti tagab kasutajakontode kasutamine rakenduse parema turvalisuse, hoides ära üleliigset koormust. Nende kontode haldamiseks on kasutatud Lucia[35] raamistikku. Lucia lihtsustab kasutajakontode loomist, paroolide krüpteerimist ja sessioonide haldamist. Lisaks sellele ühildub Lucia raamistik Prismaga. Turvalisuse tagamiseks valideeritakse kõik kasutaja poolt saabuvad päringud Zod[36] raamistikuga, et vältida poolikuid või vale kujuga päringuid. Peale konto loomist viiakse kasutaja lehele, kus on võimalik valida mitme harjutuse vahel.

[35] Autentimisraamistik Lucia https://lucia-auth.com.
[36] Skeema valideerimisraamistik Zod https://zod.dev.

```
export async function sendMail (
  userId: string, token: string, recipient: string, locale: string, attempt: number = 1
) {
  if (attempt > 5) {
    logger.warn(`Failed to send mail to recipient: ${recipient}`)
    return
  }

  const client = new EmailClient(AZURE_CONNECTION_STRING)

  const messages = localeMessages(locale)

  const message: EmailMessage = {
    senderAddress: AZURE_SENDER,
    content: {
      subject: messages[0],
      plainText:
`${messages[1]}\n${messages[2]}\n${messages[3]}\n${token}\n${messages[4]}\n${messages[5]}`,
      html: `<html lang="${locale}">
              <body>
              <p>${messages[1]}</p>
              <p>${messages[2]}</p>
              <p>${messages[3]}</p>
              <h3>${token}</h3>
              <p>${messages[4]}</p>
              <p>${messages[5]}</p>
              </body>
            </html>`
    },
    recipients: { to: [{ address: recipient }] }
  }

  try {
    codeSent.add(userId)

    const poller = await client.beginSend(message)
    const response = await poller.pollUntilDone()

    Cron(dayjs().add(RESEND_COOLDOWN, 'minutes').toDate(), () => {
      codeSent.delete(userId)
    })

    logger.debug(`Operation ID: ${response.id}`)
  } catch (error) {
    codeSent.delete(userId)
    await sendMail(userId, token, recipient, locale, attempt + 1)
  }
}
```

Joonis 4. E-posti saatmise koodilõik

Turvalisuse tagamiseks ja üleliigsete kulude vältimiseks on pilvele mõeldud rakenduse versioonis võimalik kontot teha ainult lubatud e-posti domeenidega, mis on sisestatud andmebaasi. Lisaks sellele on vaja loodud konto e-posti aadress verifitseerida sinna saadetud kinnituskoodiga. Kuna rakendus on mõeldud jooksutamiseks Microsoft Azure pilves,

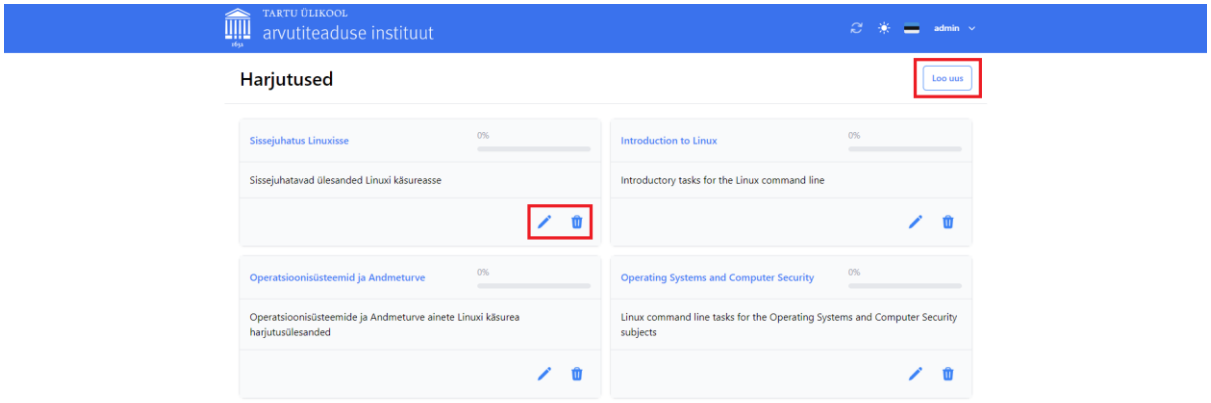
kasutatakse e-posti saatmiseks azure/communication-email[37] raamistikku. Koodilõik e-posti saatmise loogikast on nähtav joonisel 4. Rakendus hoiab *codeSent* muutujas ette määratud aega meeles, kellele on kinnituskood saadetud. Mainitud muutuja piirab kasutajatel uusi kinnituskoode endale pidevalt saatmast, mis aitab pilveteenuste kulusid kokku hoida. Lisaks seab kasutajaliides kinnituskoodide katsetamisel ajalised piirangud, vältimaks koodi ära arvamist. Kuupäevade ja aja haldamiseks on kasutatud raamistikku Day.js[38].

Lisaks tavakasutajatele loob rakendus esmasel käivitamisel ka administratiivse kasutaja, kelle parooli saab ära määrata keskkonna muutujas nimega *ADMIN_PASSWORD*. Administratiivsel kasutajal on õigus ülesandeid luua, muuta ja kustutada, millest on lähemalt kirjutatud järgmises alapeatükis.

4.2 Ülesannete loomine ja modifitseerimine

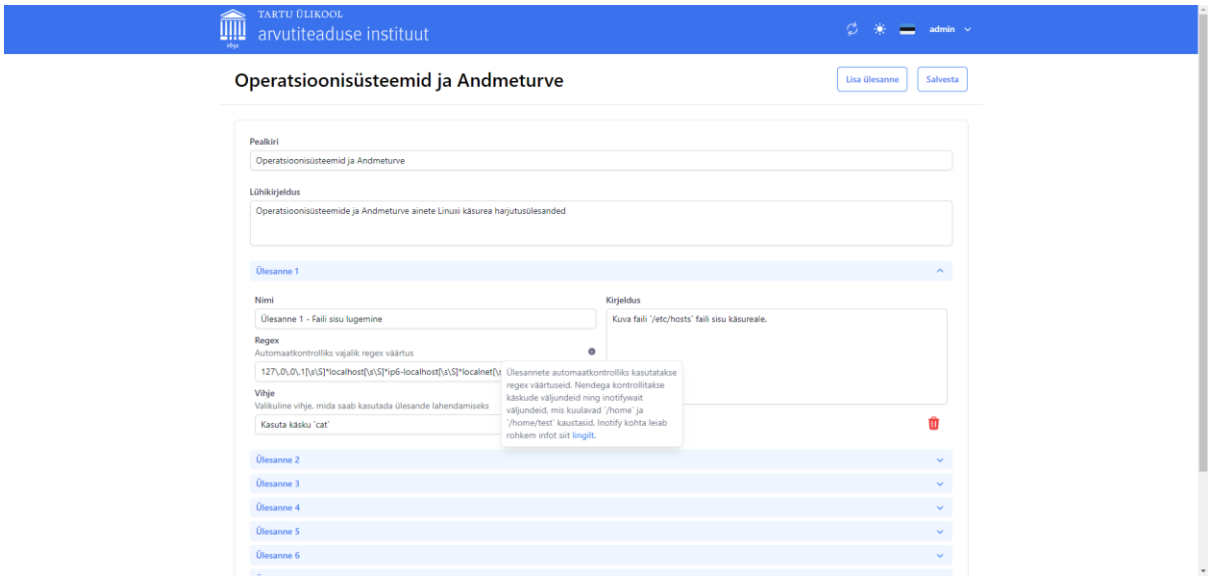
Peale rakendusse autentimist viiakse kasutaja harjutuste lehele, kus on võimalik valida erinevate harjutuskomplektide vahel (joonis 5). Valminud rakenduses on võimalik harjutusi hallata, kuna harjutuste ja nende alla kuuluvate alamülesannete informatsioon on talletatud andmebaasis. Ainus konto, millel on õigus harjutusi muuta, on administratiivne kasutaja. Harjutuste lehel on võimalik muuta või kustutada olemasolevaid harjutusi, millele viitavad vastavalt pliiatsi ja prügikasti ikoonide nupud (joonis 5). Lisaks on võimalik luua uusi harjutusi (joonis 5).

[37] Microsoft azure e-posti raamistik https://github.com/Azure/azure-sdk-for-js/tree/main/sdk/communication/communication-email.
[38] Aja halduse raamistik Day.js https://day.js.org.



Joonis 5. Harjutuste avaleht

Peale uue harjutuse loomise või olemasoleva muutmise valiku valimist viiakse kasutaja harjutuse redigeerimise lehele (joonis 6). Redigeerimise lehel on võimalik lisada harjutusele pealkiri, selle sisu lühike kirjeldus ja hallata alamülesandeid. Alamülesanne koosneb neljast komponendist, milleks on pealkiri, kirjeldus, valikuline vihje ja regulaaravaldis. Alamülesanded on harjutuse osad, mis lähevad lahendamise ajal automaatkontrollis arvesse ning mille korrektsust kontrollitakse. Regulaaravaldise lahtri juures on ka informatiivne nupp, mis annab administratiivsele kasutajale vajadusel aru, kuidas ülesannete kontekstis regulaaravaldisi kasutada (joonis 6).



Joonis 6. Harjutuse redigeerimise leht

Peale harjutuse muudatuste salvestamist viiakse kasutaja tagasi harjutuste avalehele. Vajutades harjutuse pealkirja peale viiakse kasutaja harjutuse lehele, peale mida alustab rakendusliides kasutajale Kubernetese kapsli loomist. Mainitud protsessi tutvustatakse lähemalt järgnevas alapeatükis.

4.3 Kubernetese kapsli loomine

Peale harjutuse lehele jõudmist käivitab kasutajaliides Socket.IO[39] raamistiku abil rakendusliidese suunas WebSocket ühenduse, mis algatab Kubernetese Ubuntu kapsli loomise või uuendamise. Suhtlus Kubernetese klastri ja rakendusliidese vahel toimub Kubernetes JavaScript raamistiku abil. Kui kasutajale on varasemalt juba kapsel loodud, siis värskendatakse vastava Ubuntu kapsli aegumise aega. Kapslite aegumist saab kontrollida keskkonna muutujatega *POD_DATE_VALUE*, mis vastab ajaühiku väärtusele, ning *POD_DATE_UNIT*, mis vastab ajaühikule.

Juhul, kui kasutajale hetkel vastavat kapslit ei leidu, algatatakse uue kapsli loomise protsess. Esmalt loetakse mällu Kubernetese spetsifikatsioonid, mis on säilitatud failides „pod.yaml“, „secret.yaml“ ja „service.yaml“. Kuna Kubernetese spetsifikatsioone hoitakse YAML-failides, siis on nende parsimiseks vaja kasutada yaml[40] raamistikku. Peale YAML-failide parsimist alustatakse nimeruumi loomist koos määratud aegumisajaga. Seejärel koostatakse privaatsest ja avalikust võtmest koosnev võtmepaar, kasutades selleks node-forge[41] raamistikku. Privaatse ja avaliku võtmepaari rakendamine muudab SSH ühenduse loomise kasutajanime ja parooliga võrreldes turvalisemaks.

Pärast võtmepaari loomist salvestatakse avalik võti loodavasse Ubuntu kapslisse ning privaatvõti rakenduse aktiivsesse kausta nimega „.ssh“. Lõpuks alustatakse Ubuntu kapsli ja SSH ühendust võimaldava teenuse loomist. Protsessi loogika on esitatud joonisel 7.

[39] WebSocket ühenduste raamistik Socket.IO https://socket.io.
[40] YAML-failide raamistik yaml https://eemeli.org/yaml.
[41] Krüptograafiline raamistik node-forge https://github.com/digitalbazaar/forge.

```
private async createPod (namespace: string) {
   // Kubernetese spetsifikatsioonid loetakse mällu
   const secretYaml = await useStorage('k8s').getItem<string>('secret.yaml')
   const podYaml = await useStorage('k8s').getItem<string>('pod.yaml')
   const serviceYaml = await useStorage('k8s').getItem<string>('service.yaml')

   if (!secretYaml || !podYaml || !serviceYaml) {
     throw new Error('Namespace spec not found')
   }

   // Spetsifikatsioonid muudetakse JavaScript objektideks
   const secret = parse(secretYaml)
   const pod = parse(podYaml)
   const service = parse(serviceYaml)

   // Luuakse nimeruum
   await this.api.createNamespace({
     metadata: {
       name: namespace,
       annotations: {
         expireTime: getExpireDateTime(POD_DATE_VALUE, POD_DATE_UNIT)
       }
     }
   })

   // Luuakse SSH võtmed
   const { publicKey, privateKey } = forge.pki.rsa.generateKeyPair({ bits: 2048 })

   secret.data.id_rsa = Buffer.from(forge.ssh.publicKeyToOpenSSH(publicKey)).toString('base64')
   await useStorage('ssh').setItem<string>(namespace, forge.ssh.privateKeyToOpenSSH(privateKey))

   await this.api.createNamespacedSecret(namespace, secret)

   // Luuakse kapsel
   await this.api.createNamespacedPod(namespace, pod)

   // Luuakse teenus SSH ühenduse jaoks
   await this.api.createNamespacedService(namespace, service)
}
```

Joonis 7. Kubernetese kapsli loomise koodilõik

Peale Ubuntu kapsli loomist oodatakse 3 sekundit, et Ubuntu jõuaks kapsli sees käivituda. Kui Ubuntu kapsel on käivitunud, alustatakse SSH ühenduse loomise protsessiga.

4.4 Kubernetese kapsliga ühenduse loomine

Harjutuse lehele jõudes luuakse kasutaja- ja rakendusliidese vahel turvatud WebSocket ühendus, mis on kaitstud JWT[42] (ingl *JSON Web Token*) tõendit kasutades. Peale kasutaja sisse logimist luuakse talle tõend, mida kasutatakse WebSocket ühenduse alustamisel autentimiseks. Kui WebSocket ühendus on loodud, alustatakse Kubernetese kapsli loomise või värskendamisega (joonis 8).

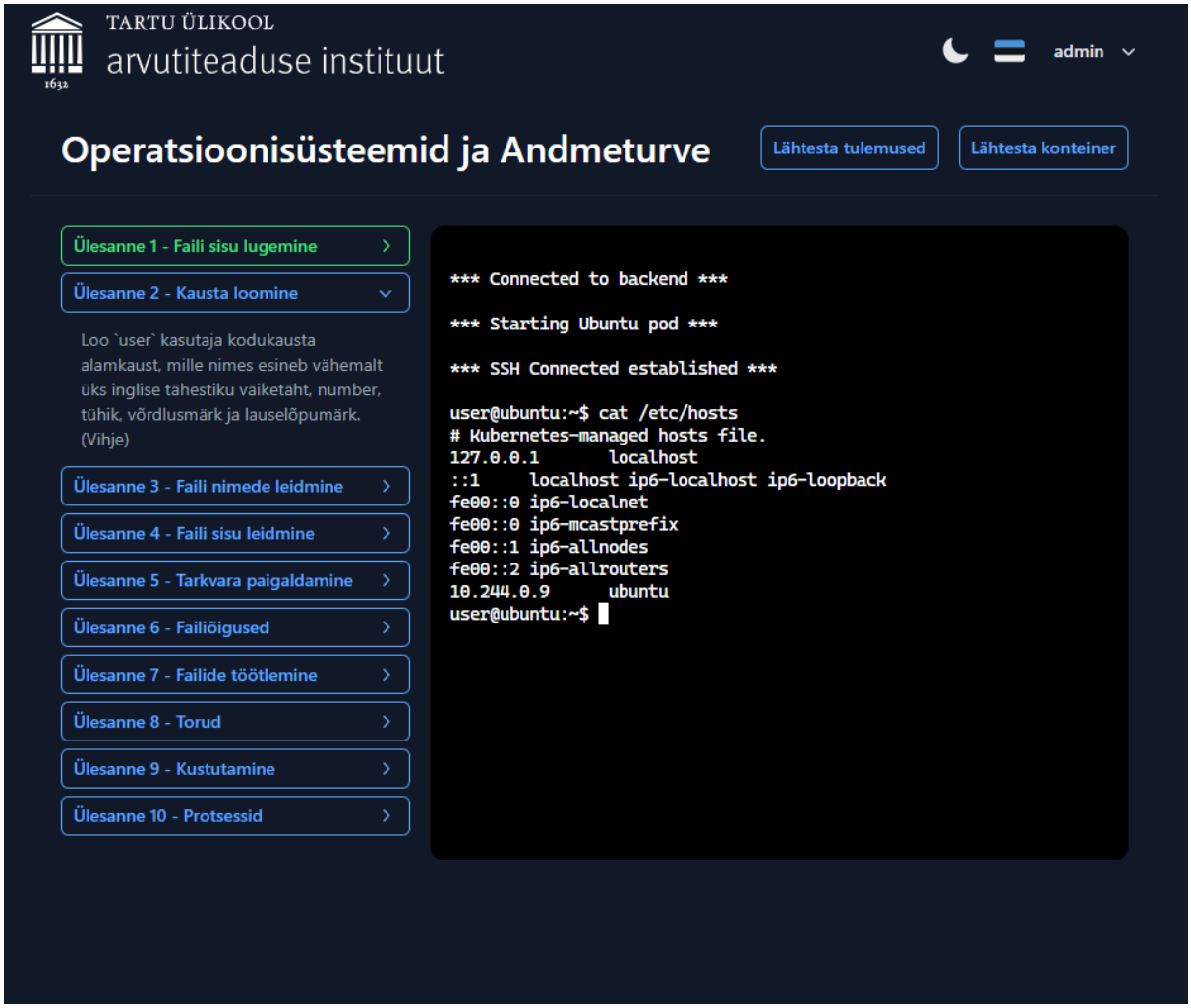
```
async function connectToPod (socket: Socket, connection: { ip: string, port: number }) {
  const ssh = new Client()

  await setProxy(socket, ssh, connection)

  const privateKey = await useStorage('ssh').getItem<string>(`ubuntu-${socket.data.clientId}`)

  if (!privateKey) {
    podLogger.error(`Private key not found for client ${socket.data.clientId}`)
    socket.disconnect()
    await kubernetes.deleteNamespace(`ubuntu-${socket.data.clientId}`)
    return
  }

  ssh.connect({
    host: connection.ip,
    port: connection.port,
    username: 'user',
    privateKey
  })
}

export default function handleSocket (server: Server) {
  if (isCloud) {
    server.on('connection', async () => {
      await startCluster(server)
    })

    emitter.on('clusterStatus', () => {
      server.emit('clusterStatus', { status: azure.getClusterStatus() })
    })
  }

  const proxy = server.of('/terminal')

  // WebSocketi autentimiseks
  // https://socket.io/docs/v4/middlewares/
  proxy.use(verifyToken)

  proxy.on('connection', async (socket) => {
    try {
      if (isCloud && azure.getClusterStatus() !== 'Running') {
        socket.send({ data: '\r\n*** Waiting for cluster to start *** \r\n' })
        await azure.waitForClusterStatus('Running')
      }
```

[42] JWT tõendite loomise raamistik https://jwt.io.

```
    socket.send({ data: '\r\n*** Starting Ubuntu pod ***\r\n' })
    const connection = await kubernetes.createOrUpdatePod(socket.data.clientId)

    // Luuakse ssh ühendus
    if (!socket.disconnected) {
      await connectToPod(socket, connection)
    }
  } catch (error) {
    podLogger.error(`Pod error for client ${socket.data.clientId}`)
    podLogger.error(error)
    socket.disconnect()
  }
 })
}
```

Joonis 8. Kasutajaliidese ja Ubuntu kapsli vahelise ühenduse loomise koodilõik

Pärast Kubernetese kapsli valmimist algab SSH ühenduse loomise protsess. Ühenduse loomiseks tuleb kasutada eelnevalt loodud kasutaja privaatvõtit, mis on salvestatud kausta „.ssh“. Algselt kontrollitakse, kas privaatvõti on olemas. Kui privaatvõtit mingil põhjusel ei leita, siis kustutatakse loodud kapsel ja protsess peatatakse. SSH ühenduse loomiseks on lisaks privaatvõtmele vaja teada ka Ubuntu kapsli IP-aadressi ja vastavat porti. Kapsli IP-aadress ja port on saadud vastava kapsli loomisfunktsiooni tagastusväärtustena. SSH ühenduse loomise protsess on nähtaval joonisel 8.



Joonis 9. Tumedas variandis harjutuse leht

Kui kõik nõutud komponendid on olemas, saavutatakse rakendusliidese ja Ubuntu kapsli vahel SSH ühendus. Loodud ühenduse tagajärjel hakkab rakendusliides käituma nagu proksi, vahendades kasutajaliidese ja Ubuntu vahel andmeid. Peale seda muutub võimalikuks kasutajaliidese käsureal käskude sisestamine, väljundi nägemine ning automaatkontrolli rakendamine (joonis 9). Kasutajaliidese käsurea visualiseerimiseks on kasutatud Xterm.js[43] raamistikku.

4.5 Ülesannete automaatkontroll

Pärast Ubuntu kapsliga ühenduse loomist saab alustada alamülesannete lahendamisega. Ülesannete kontrollimiseks kasutatakse regulaaravaldisi. Ühenduse loomise protsessis laetakse lahendamata alamülesannete regulaaravaldised mällu. Seejärel jälgib rakendusliides Ubuntu kapsli kahe kausta muudatusi ja käskude väljundeid, märkides sobiva lahendi järel ülesanne

[43] Veebilehitsejas käsurea visualiseerimisraamistik Xterm.js https://xtermjs.org.

täidetuks. Peale korrektse lahenduse leidmist salvestatakse see andmebaasi ning saadetakse WebSocketi kaudu teade kasutajaliidesesse, et märkida vastav ülesanne lahendatuks (joonis 9). Lõpuks kustutatakse vastav ülesanne mälust, vältimaks selle korduvat kontrollimist. Automaatkontrolli rakendamiseks kasutatud *evaluate* funktsiooni koodiosa asub joonisel 10.

```
async function evaluate (socket: Socket, data: string, tasks: { id: number, regex: string }[]) {
  const completed = tasks
    .filter(task => new RegExp(task.regex).test(data))
    .map(task => ({ user_id: socket.data.clientId, task_id: task.id }))

  if (completed.length > 0) {
    completed.forEach((complete) => {
      const index = tasks.findIndex(task => task.id === complete.task_id)
      tasks.splice(index, 1)
    })

    socket.emit('complete', { data: completed.map(task => task.task_id) })

    await db.completedTask.createMany({
      data: completed,
      skipDuplicates: true
    })
  }
}
```

Joonis 10. Ülesannete kontrolliks kasutatud koodilõik

Rakendusliidese ja Ubuntu kapsli SSH ühendus toimib andmevoona, kus andmed liiguvad nende vahel väikeste osadena, mille tagajärjel võib üks väljund jõuda kohale mitmes osas. Väljundi tükeldamine tähendab, et regulaaravaldised ei pruugi alati suurema kontrolli korral korrektselt toimida. Probleemi lahendamiseks kasutatakse vahepuhvrit, mis hoiab endas saabunud andmeosakesi. Kui puhver jõuab kindla pikkuseni või kui andmevoog lõpeb tekstiga *user@ubuntu:*, rakendatakse puhvris olevale väljundile automaatkontroll. Vahepuhver aitab tagada kindlamat automaatkontrolli tööd. Peale kontrolli puhver tühjendatakse ning alustatakse uuesti andmete kogumist.

Lisaks kapsli väljundile jälgitakse kahe Ubuntu kausta muudatusi, milleks kasutatakse Linuxi liidest inotify[44]. Ühenduse loomise käigus käivitatakse käsk *inotifywait /home /home/user -m*, mis avab rakendusliidese ja Ubuntu vahel veel ühe andmevoo, mille eesmärk on edastada eelnevalt mainitud kaustade muudatusi. Sarnaselt väljundi kontrollile rakendatakse ka sellele andmevoole automaatse kontrolli funktsioon *evaluate*, mis kontrollib vastava kausta muutusi ülesande lahendina.

[44] Linuxi inotify liidese manuaali leht https://man7.org/linux/man-pages/man7/inotify.7.html.

4.6 Kubernetese kapsli kustutamine

Süsteemi ressursside optimeerimise eesmärgil viib rakendusliides perioodiliselt läbi inaktiivsete Kubernetese kapslite kustutamise. Inaktiivsuse määrab ära kapslite vastava nimeruumi külge kinnitatud aegumisaeg, mis on ära määratud keskkonna muutujatega *POD_DATE_VALUE*, mis vastab ajaühiku väärtusele, ning *POD_DATE_UNIT*, mis tähistab ajaühikut.

```
async deleteNamespace (namespace: string) {
  await this.api.deleteNamespace(namespace)
  await useStorage('ssh').removeItem(namespace)
}
```

Joonis 11. Kubernetese nimeruumi kustutamise koodilõik

Võrreldes kapsli loomise protsessiga on kustutamine oluliselt lihtsam, kuna Kubernetes võimaldab ressursse nimeruumide kaudu grupeerida. Rakenduse käivitamisel registreeritakse kustutamise töö, mis aktiveerub perioodiliselt iga etteantud ajaühiku järel. Vastava aja väärtust saab määrata keskkonna muutujaga *POD_CRON_TIMER*, mis on esitatud cron-tüüpi ajamärgistusena. Perioodilise töö planeerimiseks kasutatakse Croner[45] raamistikku. Kustutamise töö käivitamisel küsitakse kõigepealt Kubernetese klastri käest kõigi nimeruumide loend, millele järgneb nende filtreerimine. Filtreerimise käigus valitakse välja ainult aegumisaja ületanud nimeruumid. Peale seda kasutatakse *deleteNamespace* funktsiooni, et kõik välja filtreeritud nimeruumid koos nende privaatvõtmetega eemaldada (joonis 11).

4.7 Kubernetese klastri käivitamine ja peatamine

Azure Kubernetese klastri pidev käitamine võib kaasa tuua märkimisväärseid kulusid. Pilveteenuste poolt põhjustatud kulude optimeerimise eesmärgil suudab rakendusliides vastavalt koormusele Azure Kubernetese klastri sisse ja välja lülitada. Klastri kontrollimiseks rakendatakse pilvele mõeldud rakenduse variandis azure/arm-containerservice[46] raamistikku.

Peale autentimist nõudva WebSocket ühenduse, mis luuakse harjutuse lehel, tekitatakse veebirakenduse külastamisel ka autentimist mittevajav ühendus, mille eesmärk on edastada Azure klastri olekut. Kui ühenduse loomisel on klaster aktiivne, jätkub tavapärane toimimine. Kui ühenduse loomisel on klaster välja lülitatud, alustab rakendusliides klastri sisse lülitamist. Kasutajaliides saab üle WebSocket ühenduse teate, et klaster pole hetkel saadaval ning seda

[45] Perioodiliste tegevuste registreerimise raamistik Croner https://croner.56k.guru.
[46] Azure Kubernetese klastri haldamise raamistik https://github.com/Azure/azure-sdk-for-js/tree/main/sdk/containerservice/arm-containerservice.

taaskäivitatakse. Kasutajale kuvatakse sellest teavitus paremas ülanurgas asuva laadimisikooni kaudu (joonis 6). Klastri käivitamise perioodil ei ole kasutajal võimalik uusi kapsleid luua.

Klastri käivitamiseks kasutatakse funktsiooni *startCluster* (joonis 12). Funktsioon ootab, kuni klaster on täielikult käivitatud, enne kui annab kasutajaliidesele WebSocketi kaudu teada, et klaster on nüüd aktiivne. Olulist rolli mängivad muutujad *clusterStatus*, mis talletab mälus klastri hetkeolekut, ning *stopLock*, mis takistab hiljuti käivitatud klastrit vähemalt 30 minutit uuesti peatamast. Klastri pideva käivitamise ja peatamise vältimine on vajalik, et ennetada süsteemivigu [13].

```
async startCluster () {
  this.clusterStatus = 'Starting'
  emitter.emit('clusterStatus')
  clusterLogger.debug('Starting cluster')

  await this.client.managedClusters.beginStartAndWait(AZURE_RESOURCEGROUP, AZURE_CLUSTER)

  this.stopLock = true
  Cron(dayjs().add(30, 'minutes').toDate(), () => {
    this.stopLock = false
  })

  this.clusterStatus = 'Running'
  emitter.emit('clusterStatus')
  clusterLogger.info('Cluster started')
}

async stopCluster () {
  this.clusterStatus = 'Stopping'
  emitter.emit('clusterStatus')
  clusterLogger.debug('Stopping cluster')

  await this.client.managedClusters.beginStopAndWait(AZURE_RESOURCEGROUP, AZURE_CLUSTER)

  this.clusterStatus = 'Stopped'
  emitter.emit('clusterStatus')
  clusterLogger.info('Cluster stopped')
}
```

Joonis 12. Klastri käivitamise ja peatamise loogika koodilõigud

Pilvepõhises rakenduse versioonis kontrollib eelmises alapeatükis mainitud perioodilise kustutamise töö Ubuntu nimeruumide arvu. Kui kapsleid ei tuvastata, algatab rakendusliides klastri peatamise protsessi, kasutades selleks *stopCluster* funktsiooni (joonis 12). Sarnaselt *startCluster* funktsiooniga edastab ka *stopCluster* funktsioon üle WebSocket ühenduse kasutajaliidesele teate, et klastrit peatatakse.

4.8 Rakenduse lokaalne variant

Veebirakenduse parema kättesaadavuse tagamiseks on võimalik seda käivitada ka lokaalsest arvutist. See võimaldab õppejõududel ja õpetajatel rakendust õppetöö raames õppeasutuse sisevõrgus jooksutada, tagades seeläbi parema privaatsuse ja kontrolli rakenduse üle. Rakenduse lokaalsel variandil puudub e-postiga konto loomine ja selle verifitseerimine, mille asemel tuleb kontod teha kasutajanime ja parooliga. Lisaks puudub lokaalsel variandil ka Kubernetese klastri sisse ja välja lülitamine, kuna vastav loogika on sobilik ainult Microsoft Azure platvormile. Pilve või lokaalse variandi valiku määrab keskkonnamuutuja *NUXT_PUBLIC_RUNTIME*, mis käivitab rakenduse pilve variandis ainult juhul, kui selle väärtus on „CLOUD“.

```
bash <(curl -fsSL https://raw.githubusercontent.com/eistre/terminal/master/local/setup.sh)
```

Joonis 13. Lokaalse variandi automaatse paigalduse skripti käsk

Rakenduse lokaalseks käivitamiseks on vajalik, et vastavas arvutis oleks installitud Node.js, töötav Kubernetese klaster ning MySQL andmebaas. Kuna nende komponentide seadistamine võib osutuda keeruliseks, on rakenduse koodihoidlas olemas skript (joonis 13), mille eesmärk on lihtsustada paigaldamisprotsessi Ubuntu keskkonnas. Skripti käivitamisel küsitakse kõigepealt kasutajalt andmebaasi ja rakenduse administraatori parooli. Seejärel antakse võimalus valida, kas süsteem peaks operatsioonisüsteemi käivitamisel ise rakendust automaatselt käivitama või mitte. Peale seda algab paigaldusprotsess, mis hõlmab Node.js ja MicroK8s[47], mis on üks Kubernetese variantidest, paigaldamist Ubuntule. Pärast MicroK8s paigaldamist luuakse klastrisse MySQL andmebaas. Kui eelnevad sammud on edukalt lõpule viidud, paigaldatakse ka veebirakendus. Skripti lõpetamisel on süsteem valmis rakendust käivitama ning kõik vajalikud komponendid on korrektselt seadistatud. Peale käivitamist on veebirakendus kättesaadav aadressidelt *http://<KOHALIK_IP>* või *http://<KOHALIK_IP>:3000*. Juhendid pilve- ja lokaalse variandi käsitsi paigaldamiseks on saadaval vastavalt lisades 1 ja 2.

4.9 Baasharjutused

Kuigi käesoleva töö raames valminud rakendus sisaldab võimalust uute harjutuste loomiseks ja olemasolevate modifitseerimiseks, tuleb veebirakendus vaikimisi kaasa kahe baasharjutuste

[47] Ubuntu loojate poolt tehtud Kubernetese variant MicroK8s https://microk8s.io.

komplektiga. Need komplektid on kättesaadavad pärast rakenduse paigaldamist ning on saadaval nii eesti kui ka inglise keeles.

Esimene komplekt on suunatud algajatele ning keskendub lihtsamatele Linuxi käsurea toimingutele, nagu kaustade vahel navigeerimine, failide sisu lugemine ning nende kustutamine. Teine komplekt, mis on esimesest keerukam, oli algselt koostatud Halapuu poolt ja oli ette nähtud kasutamiseks Tartu Ülikooli õppeainetes „Operatsioonisüsteemid“ ja „Andmeturve“.

Kuna Halapuu loodud komplekt sisaldas alamülesandeid, mis ei pruukinud alati toimida ega sobituda uue rakenduses kasutatud automaatkontrolliga, oli vajalik teha mõned muudatused. Muudatuste tulemusena muutusid alamülesanded teistest sõltumatuks ning töökindlamaks. Lisaks, vastavalt bakalaureusetöö juhendaja Alo Peetsi soovile, lisati alamülesannete hulka ka ülesanne, mis käsitleb torusid (ingl *pipes*)[48] (joonis 14). Baasharjutuste komplektid on leitavad lisas 3.

```
{
   title: 'Ülesanne 8 - Torud',
   content: 'Kuva ainult `df -h` käsu väljundi `Filesystem` veerg.',
   hint: 'Kasuta torusid (pipes) ja `cut` käsku',
   regex: /Filesystem\s+overlay\s+tmpfs[\s\S]+?\/dev\//.source,
   exercise_id: ee
}
```

Joonis 14. Torusid käsitlev alamülesanne

Käesoleva töö raames valminud Linuxi õppekeskkond koosneb neljast peamisest komponendist, milleks on kasutaja- ja rakendusliides, andmebaas ning Kubernetese klaster. Rakenduse põhiliseks punktiks osutub rakendusliides ehk API, mis vastutab kõikide komponentide vahelise andmevahetuse eest. Valminud õppekeskkonnas on võimalik luua kontosid ning lahendada isoleeritud Ubuntu konteinerites vastavaid harjutusi. Lisaks võimaldab rakendus administratiivse kasutajaga harjutusi modifitseerida. Parema kättesaadavuse tagamiseks leidub õppekeskkonnal ka paigaldamise skript, mis võimaldab keskkonda jooksutada lokaalselt.

[48] Pärineb eravestlusest lõputöö juhendaja Alo Peetsiga.

5. Testimine ja tulemused

Järgnevas peatükis käsitletakse põhjalikumalt valminud rakenduse testimiskeskkonda ja seotud kulusid. Lisaks sellele antakse ülevaade rakenduse testimisprotsessist, saavutatud tulemustest, ootamatust andmebaasi migratsioonist ning edasiarenduse võimalustest.

5.1 Testimise keskkond

Valminud rakendus seati üles Microsoft Azure pilvekeskkonda, kasutades sealset Azure Container Apps[49] teenust. Microsofti järgi on Azure Container Apps Kubernetese tehnoloogial põhinev platvorm, mille abil on võimalik jooksutada automaatselt hallatud konteineripõhiseid rakendusi [14]. Lisaks võimaldab Microsoft igakuiselt kasutada 180 000 sekundit tasuta virtuaalset CPU-d ning 2 miljonit tasuta päringut rakenduse serveri vastu.

Veebirakendus on üles seadistatud üheainsa koopiana Kesk-Rootsi piirkonnas (ingl *Sweden Central*), millele on reserveeritud 0.5 CPU tuuma ning 1 GB mälu. Rakenduse konteineri kettapilt on sarnaselt Ubuntu konteineri kettapildile kättesaadav rakenduse koodihoidlast. Testkeskkonnas on inaktiivsete Ubuntu kapslite ja kasutajate aegumisajaks määratud vastavalt 2 tundi ja 2 kuud.

Kubernetese klastrina on kasutatud Azure Kubernetes Service[50] teenust. Microsoft on välja toonud, et Azure Kubernetes Service pakub klastrile automaatset haldamist ja skaleeritavust [15]. Vastava teenuse odavaim plaan hõlmab endas tasuta klastri haldamist, kus kulud tekivad üksnes ainult klastris kasutatud virtuaalmasinate, kettamahu ning võrguliikluse pealt.

Klaster ise on üles seatud Ida-Norra piirkonda, kuna Azure piiras tudengi kontoga Kubernetes Service ning Container Apps keskkondi luua samas piirkonnas. Lisaks oli klaster esialgselt üles seadistatud kuni kaheni skaleeriva virtuaalmasinate kogumiga, mille virtuaalmasinad on B2s tüüpi suuruse ning 128 GB kettaruumiga. B2s tüüpi virtuaalmasin tähendab, et sellele on määratud 2 CPU tuuma ja 4 GB mälu [16], mille pidev jooksutamine läheks Azure hinnakalkulaatori[51] järgi maksma keskmiselt $32 kuus.

Klastris olevate kapslitega ühenduse loomiseks tuli virtuaalmasinate kogum seadistada avalike IP-aadressidega ning avada nendel masinatel pordid vahemikus 30 000 kuni 32 767. Seadistused on vajalikud selleks, et tagada SSH ühenduse kättesaadavus vastavale

[49] Azure Container Apps teenus https://azure.microsoft.com/en-us/products/container-apps.
[50] Azure Kubernetes Service teenus https://azure.microsoft.com/en-us/products/kubernetes-service.
[51] Azure hinnakalkulaator https://azure.microsoft.com/en-gb/pricing/calculator.

virtuaalmasinale, mis parasjagu kapslit jooksutab. Pordivahemik on määratud vahemikus 30 000 kuni 32 767, kuna kapsli valmimisel lisatakse vastavasse nimeruumi Kubernetese *NodePort* teenus, mis võimaldab kapslile määratud pordi kaudu välist juurdepääsu [17]. Virtuaalmasinate kogumil portide lubamist on võimalik sooritada Azure Command-Line Interface[52] kaudu [18]. Selle saavutamiseks kasutatav käsurea käsklus on leitav joonisel 15.

```
az aks nodepool update --resource-group <GRUPI NIMI> --cluster-name <KLASTRI NIMI> --name
<MASINATE KOGUMI NIMI> --allowed-host-ports 30000-32767/tcp
```

Joonis 15. Masinate kogumi pordi avamise käsk

E-postiga loodud kasutajate verifitseerimiseks ning kapslite SSH võtmete talletamiseks taaskäivituste vahel on kasutatud vastavalt Azure Communication Services[53] ning Azure Files[54] teenuseid. Rakenduse MySQL andmebaas oli üles seatud PlanetScale platvormile.

Name	Type	Location
AzureManagedDomain (terminal-email/AzureManagedDomain)	Email Communication Services Domain	Global
CPU Usage Percentage - terminal-cluster	Metric alert rule	Global
Memory Working Set Percentage - terminal-cluster	Metric alert rule	Global
RecommendedAlertRules-AG-1	Action group	Global
terminal	Container App	Sweden Central
terminal-cluster	Kubernetes service	Norway East
terminal-communication	Communication Service	Global
terminal-database	Azure Database for MySQL flexible server	Sweden Central
terminal-email	Email Communication Service	Global
terminal-environment	Container Apps Environment	Sweden Central
terminalstorage	Storage account	Sweden Central
workspaceterminal9f80	Log Analytics workspace	Sweden Central

Joonis 16. Rakenduse jaoks loodud grupi sisu

Peale keskkonna edukat üles seadmist on veebirakendus kasutatav ja kättesaadav. Joonisel 16 on välja toodud nimekiri loodud keskkonnast, koos seal kasutatud teenuste ja tehnoloogiatega. Veebirakendus ise on kättesaadav Microsoft Azure poolt loodud aadressil https://terminal.agreeableground-a83f8169.swedencentral.azurecontainerapps.io või lühemalt https://lingid.ee/kasurida.

[52] Azure Command-Line Interface https://learn.microsoft.com/en-us/cli/azure.
[53] E-postide saatmiseks kasutatud Azure Communication Services https://azure.microsoft.com/en-us/products/communication-services.
[54] SSH võtmete talletamiseks kasutatud Azure Files teenus https://learn.microsoft.com/en-us/azure/storage/files/storage-files-introduction.

5.2 Testimise tulemused

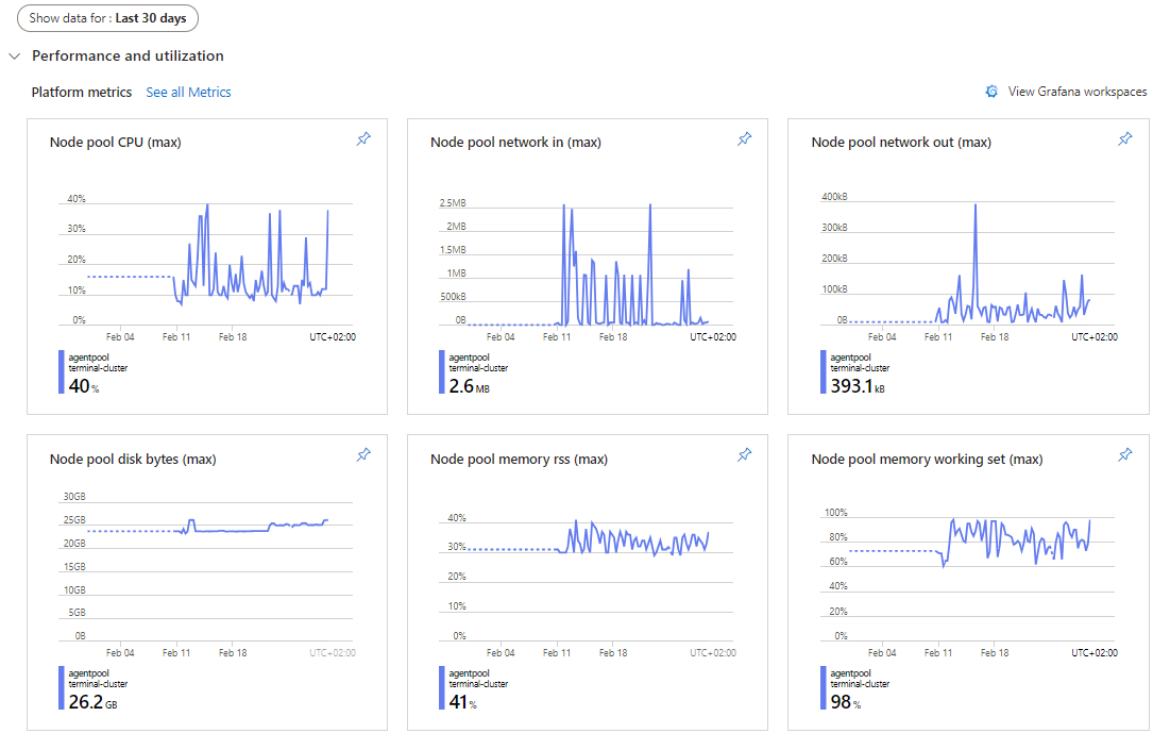
Seadistatud rakendus seati üles 17. jaanuaril 2024. a., peale mida on see olnud pidevas töös. Avalikku rakendust testiti aktiivselt kahel korral. Esimene testimise sessioon leidis aset 18. jaanuaril 2024, mil Tartu Ülikoolis korraldati informaatika õpetamise konverents[55]. Konverentsil tutvustati valminud rakendust juhendaja Alo Peetsi poolt läbi viidud töötoas umbes kümnele informaatika õpetajale. E-postiga seotud võimalike tõrgete vältimiseks konfigureeriti täiendavalt salajane e-posti domeen *@test.sec*, mille kasutamine ei nõudnud rakenduse poolt verifitseerimist. Töötoast saadud tagasiside oli positiivne. Mitmed informaatika õpetajad väljendasid huvi veebirakendust kasutada oma tundides. Töötoa käigus avastati rakenduse automaatkontrollis viga, kus andmevoo kontrollimatu tükeldamise tulemusel jäid mõned ülesanded kontrollimata. Probleemi lahendamiseks on kasutatud vahepuhvrit, mida kirjeldatakse lähemalt alapeatükis 4.5.

Teine testimisperiood toimus ajavahemikus 12. veebruar kuni 29. veebruar 2024, kus rakendust testiti Tartu Ülikooli õppeaine „Andmeturve“ raames. Testimine korraldati kursuse esimese praktikumi[56] vältel ning veebikeskkonda kasutas sellel perioodil 382 inimest. Testimisperioodi käigus suutsid vastava praktikumi harjutuse kõik 10 alamülesannet edukalt lahendada 354 testijat. Juhendaja Alo Peetsi sõnul kulges testimisprotsess edukalt ja ta jäi valminud keskkonnaga rahule[57]. Testijad olid uue keskkonna disainiga rahul ning enamik sai keskkonnale probleemideta ligi. Ta märkis, et mõnel üksikul tudengil tekkis probleeme oma e-posti verifitseerimisega, kuid see probleem lahendati eelmainitud salajase e-posti domeeni kasutamisega.

Teise testimissessiooni käigus ilmnes kaks peamist probleemi. Esimene probleem tekkis kasutajakontode loomise protsessis. Paljud tudengid olid segaduses, kuna nende kooli poolt antud e-posti aadress on kujul *eesnimi.perenimi@ut.ee*, kuid paljud neist proovisid kontosid luua oma õppeinfosüsteemis kasutatava kasutajanimega kujul *kasutajanimi@ut.ee*. Eksimuse tulemusel ei jõudnud verifitseerimiskoodid õigetesse postkastidesse ning kontod jäid verifitseerimata. Teine probleem tekkis kapslite loomise protsessil. Eelnevas alapeatükis on mainitud, et algse keskkonna Kubernetese klaster oli seadistatud kuni kaheni skaleeruva B2s tüüpi virtuaalmasinate kogumiga. Kuna need virtuaalmasinad on üsna väikesed, siis

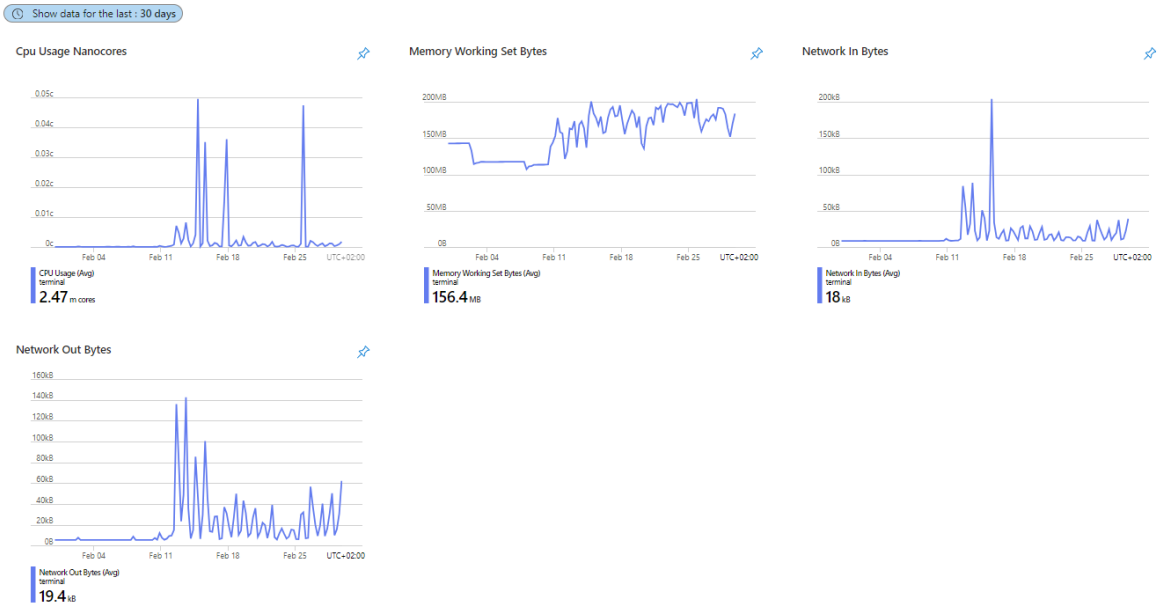
[55] Informaatika õpetamise konverents 2024 https://didaktika.cs.ut.ee/sundmused/konverents-2024.
[56] Rakenduse testimiseks kasutatud Andmeturve kursuse esimese praktikumi juhend https://courses.cs.ut.ee/2024/turve/spring/Main/Praktikum1.
[57] Pärineb eravestlusest lõputöö juhendaja Alo Peetsiga.

testimisperioodi kõrgeima koormuse ajal ei suutnud nad Kubernetest ja kõiki Ubuntu kapsleid mahutada. Seetõttu piiras Azure stabiilsuse tagamiseks nende loomist ning ühel või kahel tudengil jäi kapsel loomata. Probleemi lahendamiseks suurendati klastri virtuaalmasinate skaleeritavust kahelt kuuele, mis tagab vastava koormuse korral kolm korda parema skaleeritavuse.



Joonis 17. Azure Kubernetese klastri näidud veebruaris 2024

Suurima koormuse juures oli klastris korraga umbes 30 kapslit. Joonisel 17 on esitatud Kubernetese klastri keskmised näidud veebruaris. Joonisel on punktiirjoonega näha, et klaster oli 11. veebruarini rakenduse poolt kulude optimeerimise eesmärgil välja lülitatud. CPU kasutuse osas tõusis see suurema koormusega päevadel umbes 40%-ni. Keskmiselt jäi see vahemikku 10% kuni 25%. Võrguliikluse analüüs näitab, et kuna rakendusliides ja klaster olid omavahel ühendatud peamiselt ainult SSH ühenduste kaudu, olid võrguliikluse määrad madalad. Mälu kasutuse poolest ulatus see suurema koormuse perioodil peaaegu 100%-ni. Mälu kasutusest võib järeldada, et virtuaalmasinatel tekkisid mäluga seotud probleemid, mille tõttu jäi mõni kapsel loomata.

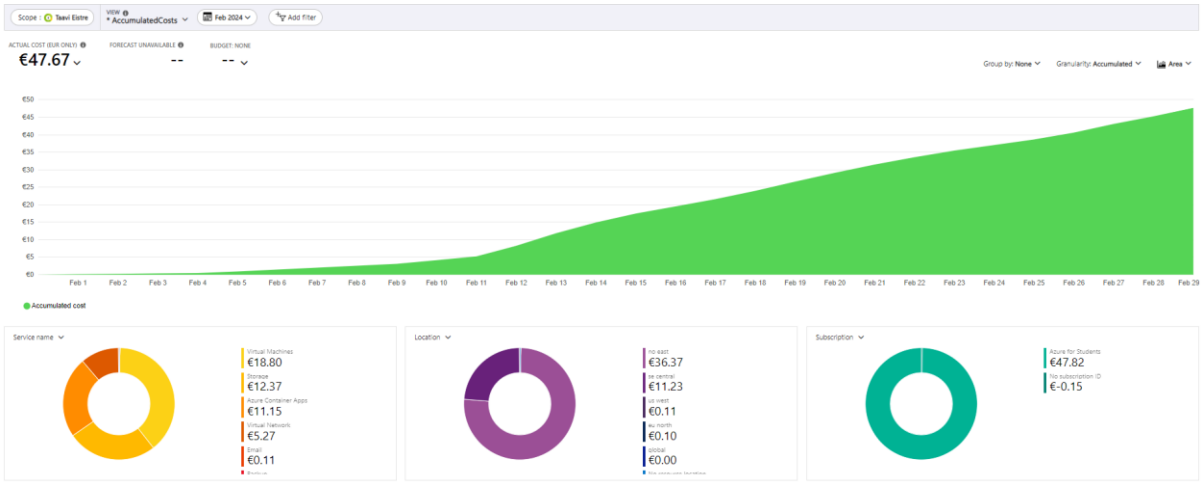


Joonis 18. Rakenduse keskmised näidud veebruaris 2024

Võrreldes Kubernetese klastriga töötas rakendus terve veebruarikuu vältel. Joonisel 18 on esitatud rakenduse Container Apps keskkonna keskmised näidud veebruaris. Sarnaselt Kubernetese klastri näitajatega on kuni 11. veebruarini rakenduse mõõdud madalad. CPU kasutuse analüüsist selgub, et koormuse perioodil tõusis see umbes 0.05 tuuma tasemele, ehk rakendusele määratud 0.5 tuuma osutus piisavaks. Sama kehtib ka mälukasutuse kohta. Jooniselt võib märgata, et keskmine mälukasutus püsis umbes 150 MB juures, tõustes suurema koormuse korral umbes 200 MB-ni. Kuna rakendusele on eraldatud 1 GB mälu, võib järeldada, et see on rohkem kui piisav. Üllatavaks osutus rakenduse võrguliiklus, mis oli keskmiselt madalam võrreldes klastri omaga, kuna lisaks SSH ühendustele peab rakendus hakkama saama ka kasutajate poolse andmevahetusega.

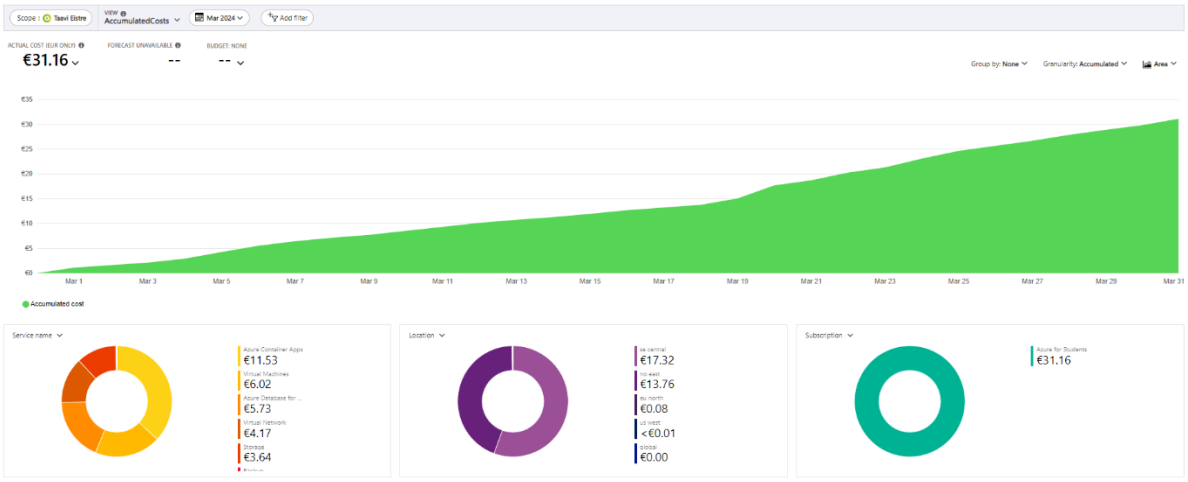
5.3 Testimise kulud

Kuna rakenduse testimiseks ja käitamiseks on kasutatud Microsoft Azure pilveteenuseid, mis on tasulised, siis nendega kaasnesid kulud. Veebruar 2024 oli periood, mil rakendus oli pidevalt aktiivne ning mil toimus ka kõrgeima koormusega testimine. Sel kuul olid pilveteenuste kulud kokku €47.67 (joonis 19), mis teeb umbes €0.125 testija kohta.



Joonis 19. Azure pilveteenuste kulud veebruaris 2024

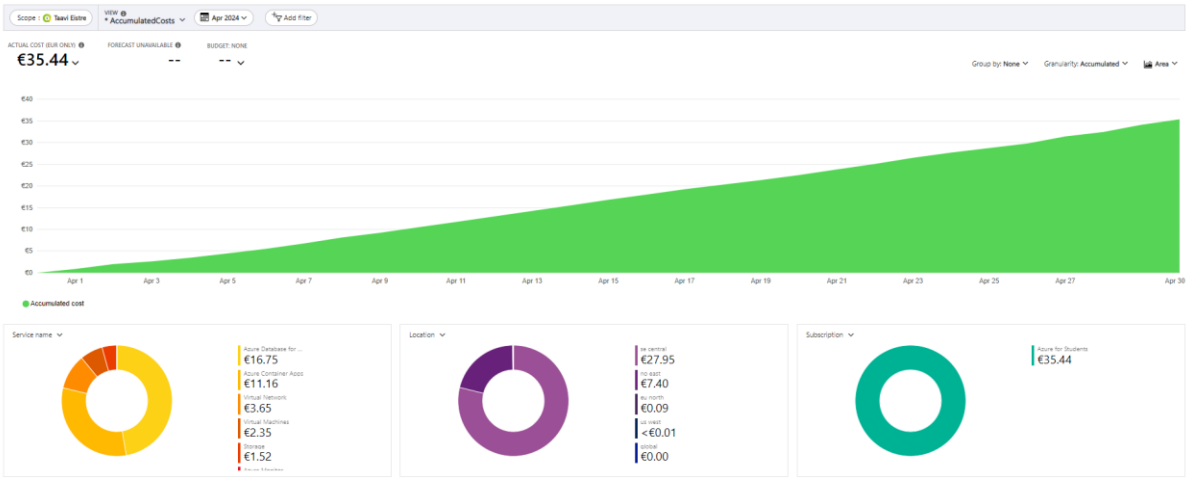
Veebruarikuu kulutused koos detailsema informatsiooniga on esitatud joonisel 19. Joonise põhjal on selgesti nähtav, et 11. veebruarini oli rakendus vähese koormuse tõttu passiivne. Peale seda algas testimine ning ka sellega kaasnev koormus. Teenuste ülevaatest on välja toodud, kui suured olid kulud erinevate komponentide ja teenuste kasutamisel. Kõige suuremad kulud tulid Kubernetese klastri virtuaalmasinate eest, mis läksid kokku maksma €18.8. Lisaks kaasnesid kulud ka kettamahtudele summas €12.37 ning virtuaalsele võrgule €5.27. Rakenduse kasutamine Container Apps keskkonnas tõi kaasa kulud summas €11.15. Lisaks nendele tuli arvestada veel verifitseerimiskoodide saatmisest tuleneva e-posti kuluga, mis maksis kokku €0.11.



Joonis 20. Azure pilveteenuste kulud märtsis 2024

Märtsikuu, mida võib pidada veebruariga võrreldes passiivsemaks selle vähese koormuse tõttu, kulud on välja toodud joonisel 20. Pilveteenuste peale kulus kokku €31.16, millest tuleb võrdluse eesmärgil maha arvestada €5.73 andmebaasi kulu, mis tekkis kuu keskel toimunud

migratsiooni tõttu ning mida käsitletakse põhjalikumalt järgmises alapeatükis. Seega tulid sel kuul kuludeks kokku €25.43, millest kõige suurema osa moodustas Container Apps keskkonnas asuv veebirakendus. Võrreldes eelneva kuuga tuleb märkida, et suurim kulude vähenemine tulenes virtuaalmasinate ja kettamahtude kasutamisest, mille tulemusel säästeti kokku €21.51.



Joonis 21. Azure pilveteenuste kulud aprillis 2024

Aprillikuus, mis oli koormuse poolest sarnane märtsikuuga, kulus pilveteenuste peale kokku €35.44. Antud kuu kulutused on detailsemalt esitatud joonisel 21. Kõige suuremaks erinevuseks eelneva kuuga tuli andmebaasi kuludes, kuna võrreldes märtsiga töötas uus MySQL andmebaas terve aprillikuu. Võrreldes veebruariga, mil virtuaalmasinate ja kettamahtude peale kulus kokku €31.17, läks aprillis samade ressursside peale €3.87.

5.4 Andmebaasi migratsioon

PlanetScale tegevjuht Lambert avaldas 6. märtsil 2024. aastal teadaande, milles teavitas ettevõtte kavatsusest lõpetada tasuta hobi teenuse pakkumine alates 8. aprillist 2024 [19]. Seejärel hinnastati Kesk-Euroopa piirkonnas, kus testimisperioodil kasutatud andmebaas asus, odavaim andmebaasi teenus $47 peale [20]. Hinnastamisest tulenevalt tekkis kulude kokkuhoiu huvides vajadus läbi viia andmebaasi migratsioon.

Kuna kogu ülejäänud rakenduse infrastruktuur paiknes Microsoft Azure pilvekeskkonnas, tundus andmebaasi migreerimine sinna asjakohane valik. Azure hinnakalkulaatori järgi lisaks B1ms tüüpi MySQL andmebaas, mis on olemasolevatest valikutest üks kergekaalulisemaid, kuludele juurde ligikaudu €17 kuus.

```
# Tõmbab PlanetScale andmebaasist alla skeema ja kõik andmed
pscale database dump terminal main

# Salvestab andmebaasi skeema ja andmete sisu ühte faili
cat *-schema.sql > all_schema.sql
cat *.00001.sql > all_data.sql

# Sisestab andmed uue andmebaasi sisse
mysql -h <ANDMEBAASI_IP> -u <KASUTAJANIMI> -p <ANDMEBAAS> < all_schema.sql
mysql -h <ANDMEBAASI_IP> -u <KASUTAJANIMI> -p <ANDMEBAAS> < all_data.sql
```

Joonis 22. Andmebaasi migratsiooniks vajalikud käsud

Pärast uue andmebaasi loomist tuli üle kanda nii skeema kui ka testimisperioodi jooksul tekkinud andmed. Migratsiooni teostamiseks kasutati PlanetScale[58] ja MySQL[59] käsurea liidest. Algselt oli vaja PlanetScale andmebaasist kätte saada skeema kui ka andmed. Seejärel tuli need uude andmebaasi üles laadida. Protsessi läbiviimiseks nõutud käsud on leitavad jooniselt 22.

5.5 Edasiarenduse võimalused

Uues rakenduses on tähtsal kohal harjutuste modulaarsus ning modifitseerimisvõimalus. Andmebaasi ja uue kasutajaliidese olemasolu võimaldab kiiresti harjutusi modifitseerida ning juurde lisada. Lõputöö käigus valminud veebirakenduses on hetkel olemas kaks harjutuse komplekti, mis on kättesaadavad nii eesti kui ka inglise keeles. Modulaarsusest ja modifitseerimisvõimalusest tulenevalt tekib võimalus lisada keskkonda uusi harjutusi, mis on kättesaadavad mitmes keeles ning pakuvad väljakutseid erinevatele haridustasemetele.

Valminud rakendus peab korrektse toimimise tagamiseks järjepidevalt töötama, kuna vanade ja inaktiivsete Kubernetese kapslite ning kasutajate eemaldamine nõuab perioodilist käivitamist. Pilvepõhisel variandil kaasneb järjepideva tööga suurem kulu. Üks võimalus nende kulude optimeerimiseks oleks rakendus ümber kirjutada selliselt, et kasutada ära funktsioon teenusena (ingl *function as a service*) võimalusi. Funktsioon teenusena kasutamine kaotab ära vajaduse rakendust pidevalt jooksutada ning võimaldab perioodilisi töid käivitada väljaspool rakendust iseseisvate funktsioonidena. Rakendus ise peaks siis töötama sama kaua kui Kubernetese klaster, käivitudes alles siis kui keegi veebilehte külastab.

Käesoleva bakalaureusetöö tulemusel loodud õppekeskkond on peamiselt mõeldud harjutuskomplektide lahendamiseks. Kui kasutaja ei suuda mõnda alamülesannet lahendada,

[58] Andmebaasi migratsiooniks kasutatud PlanetScale käsurea liides https://planetscale.com/features/cli.
[59] Andmebaasi migratsiooniks kasutatud MySQL käsurea liides https://dev.mysql.com/doc/refman/8.0/en/mysql.html.

peab ta leidma edasisi juhiseid internetist. Üheks võimalikuks edasiarenduseks oleks luua võimalus lisada igale harjutusele kasulikku õppematerjali, mis hõlbustaks lahendamisprotsessi, kuna interneti asemel oleks vastavad juhised kättesaadavad samast veebirakendusest.

Kokkuvõte

Tartu Ülikooli õppeainetes „Operatsioonisüsteemid“ ja „Andmeturve“ on oluliseks käsitlusalaks Linuxi käsurida. Seoses õppejõudude sooviga Linuxi käsurida kvaliteetsemalt õpetada lõi Joonas Halapuu 2022. aastal oma bakalaureusetöö raames veebipõhise Linuxi käsurea õppekeskkonna prototüübi, mis oli mainitud õppeainetes kasutusel kaks aastat. Vastava perioodi vältel ilmnes mitmeid murekohti ning võimalusi edasiarenduseks.

Käesoleva bakalaureusetöö raames arendas töö autor edasi Joonas Halapuu poolt loodud prototüüpi, kirjutades enamiku eelnevalt loodud koodist ümber pilvetehnoloogia jaoks sobivamaid tehnoloogiaid kasutades ning oluliselt lisafunktsionaalsust lisades. Esmalt pandi paika edasiarendatava rakenduse nõuded ning eesmärgid. Valminud rakenduse arendamiseks kasutati TypeScripti ja Node.js raamistikku. Kasutajaliidese ja serveri arenduseks rakendati Vue.js’i ning sellel põhinevat Nuxt 3 raamistikku. Andmebaasiks ning konteinerite orkestreerimiseks kasutati vastavalt MySQL andmebaasi ning Kubernetest. Valminud rakenduse keskkond koosneb kasutaja- ja rakendusliidesest, MySQL andmebaasist ning Kubernetese klastrist. Veebirakenduse keskpunktiks osutub rakendusliides ehk API, mis koordineerib suhtlust kasutaja, andmebaasi ja Kubernetese klastri vahel.

Veebilehe külastajatel on võimalus luua oma e-posti aadressiga kasutajakontosid, mida tuleb e-postile saadetud verifitseerimiskoodiga kinnitada. Pärast konto verifitseerimist antakse kasutajale valik olemasolevate harjutuste vahel. Peale harjutuse valimist alustab rakendusliides Kubernetese kapsli loomise või uuendamisega. Seejärel luuakse kasutajaliidese ning vastava kapsli vahel WebSocket ja SSH ühendus, peale mida saab kasutaja hakata alamülesandeid lahendama. Rakendusliides jälgib kasutaja ja kapsli vahelist andmevahetust, salvestades õigesti lahendatud alamülesanded andmebaasi ning muutes need kasutajaliideses roheliseks. Pärast alamülesannete lahendamist eemaldab rakendusliides määratud aja jooskul inaktiivsed kapslid klastrist ressursside optimeerimise eesmärgil. Microsoft Azure pilvekeskkonnas toimiv rakendus lülitab inaktiivsel perioodil kulude optimeerimise eesmärgil ka klastri välja.

Valminud rakenduses on võimalik harjutusi hallata, kuna harjutuste ja nende alla kuuluvate alamülesannete informatsioon on talletatud andmebaasis. Harjutusi saab muuta ainult administratiivsete õigustega kasutaja. Rakendus võimaldab muuta harjutuse pealkirja, lühikirjeldust ning alamülesandeid. Alamülesanne koosneb neljast komponendist, milleks on pealkiri, kirjeldus, valikuline vihje ja regulaaravaldis. Alamülesanded on harjutuse osad, mis lähevad lahendamise ajal automaatkontrollis arvesse ning mille korrektsust kontrollitakse.

Rakendus paigaldati Microsoft Azure pilvekeskkonda. Testimine viidi läbi kahes sessioonis. Esimene sessioon leidis aset 2024. aasta jaanuaris, kus töötoa raames demonstreeriti rakendust ligikaudu kümnele õpetajale. Teine testimissessioon toimus 2024. aasta veebruaris Tartu Ülikooli õppeaine „Andmeturve“ esimese praktikumi raames, kus keskkonda kasutas 382 testijat. Rakenduse keskkond suutis testimisperioodidel esinenud koormusele vastu pidada ning testimine lõppes edukalt. Veebruaris, mil toimus kõige suurema koormusega testimine, olid rakenduse kulud pilvekeskkonnas kokku €47.67, mis teeb umbes €0.125 testija kohta. Aprillis, mil koormus oli madal ning uus Azure MySQL andmebaas töötas terve kuu, olid pilveteenuste kulud kokku €35.44. Kõik käesoleva bakalaureusetöö juhendaja poolt seatud nõuded ja eesmärgid said lõputöö käigus edukalt täidetud ning valminud lahendust on testitud nõuete kontekstis suure kasutajate hulga juures. Bakalaureusetöö raames valminud veebipõhine Linuxi õppekeskkond on leitav ja kasutatav nii eesti kui inglise keeles aadressil https://lingid.ee/kasurida.

Viidatud kirjandus

[1] Halapuu, J. Eestikeelse veebipõhise Linuxi käsurea õpikeskkonna loomine, TÜ arvutiteaduse instituudi bakalaureusetöö. 2022. https://comserv.cs.ut.ee/ati_thesis/datasheet.php?id=74682 (13.05.2024)

[2] The Linux Foundation. What is Linux? https://www.linux.com/what-is-linux (13.05.2024)

[3] Bogner J, Merkel M. To Type or Not to Type? A Systematic Comparison of the Software Quality of JavaScript and TypeScript Applications on GitHub. *2022 IEEE/ACM 19th International Conference on Mining Software Repositories (MSR), Mining Software Repositories (MSR), 2022 IEEE/ACM 19th International Conference on, MSR*., 2022, 659-664. https://doi-org.ezproxy.utlib.ut.ee/10.1145/3524842.3528454

[4] Nuxt. Introduction. https://nuxt.com/docs/getting-started/introduction (13.05.2024)

[5] Pahl C, Brogi A, Soldani J, Jamshidi P. Cloud Container Technologies: A State-of-the-Art Review. *IEEE Transactions on Cloud Computing, Cloud Computing, IEEE Transactions on, IEEE Trans Cloud Comput*., 2019, 7(3), 678. https://doi-org.ezproxy.utlib.ut.ee/10.1109/TCC.2017.2702586

[6] Moravcik M, Kontsek M. Overview of Docker container orchestration tools. *2020 18th International Conference on Emerging eLearning Technologies and Applications (ICETA), Emerging eLearning Technologies and Applications (ICETA), 2020 18th International Conference o*., 2020, 475-480. https://doi-org.ezproxy.utlib.ut.ee/10.1109/ICETA51985.2020.9379236

[7] Poljak R, Poscic P, Jaksic D. Comparative analysis of the selected relational database management systems. *2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2017 40th International Convention on*., 2017, 1496. https://doi-org.ezproxy.utlib.ut.ee/10.23919/MIPRO.2017.7973658

[8] Prisma Data. What is Prisma ORM? https://www.prisma.io/docs/orm/overview/introduction/what-is-prisma (13.05.2024)

[9] Saraswat M, Tripathi RC. Cloud Computing: Comparison and Analysis of Cloud Service Providers-AWs, Microsoft and Google. *2020 9th International Conference System Modeling and Advancement in Research Trends (SMART), System Modeling and Advancement in Research Trends (SMART), 2020 9th International Conference*., 2020, 281-283. https://doi-org.ezproxy.utlib.ut.ee/10.1109/SMART50582.2020.9337100

[10] Microsoft. Azure for Students – Free Account Credit. https://azure.microsoft.com/en-us/free/students (13.05.2024)

[11] PlanetScale. What is PlanetScale. https://planetscale.com/docs/concepts/what-is-planetscale (13.05.2024)

[12] PlanetScale. PlanetScale Hobby plan. https://planetscale.com/docs/concepts/hobby-plan (29.03.2024)

[13] Microsoft. Stop and start an Azure Kubernetes Service (AKS) cluster. https://learn.microsoft.com/en-gb/azure/aks/start-stop-cluster?tabs=azure-cli#start-an-aks-cluster (13.05.2024)

[14] Microsoft. Azure Container Apps. https://azure.microsoft.com/en-us/products/container-apps (13.05.2024)

[15] Microsoft. Azure Kubernetes Service (AKS). https://azure.microsoft.com/en-us/products/kubernetes-service (13.05.2024)

[16] Microsoft. B-series burstable virtual machine sizes. https://learn.microsoft.com/en-us/azure/virtual-machines/sizes-b-series-burstable (13.05.2024)

[17] The Kubernetes Authors. Service. https://kubernetes.io/docs/concepts/services-networking/service/#type-nodeport (13.05.2024)

[18] Microsoft. Use instance-level public IPs in Azure Kubernetes Service (AKS) https://learn.microsoft.com/en-us/azure/aks/use-node-public-ips#allow-host-port-connections-and-add-node-pools-to-application-security-groups (13.05.2024)

[19] Lambert S. PlanetScale forever. 2024. https://planetscale.com/blog/planetscale-forever (13.05.2024)

[20] PlanetScale. PlanetScale pricing. https://planetscale.com/pricing (13.05.2024)

Lisad

I. Rakenduse pilve variandi paigaldamise juhend

Järgnevalt on välja toodud rakenduse pilve variandi paigaldamise juhend:

1. Loo uus Azure Kubernetes Service klaster koos järgnevate seadistustega:
    a. Autentimine – *Local accounts with Kubernetes RBAC*;
    b. *Node pool – Enable public IP per node*;
    c. Võrgu konfiguratsioon – *Kubenet*.
2. Liigu klastri lehele ning vali nupp „*Connect*“, peale mida järgi „*Azure CLI*“ ühendamise juhendit.
3. Kasuta käsku *cat $HOME/.kube/config | base64*, ning jäta saadud kodeeringus klastri ühenduse konfiguratsioon meelde.
4. Luba klastri virtuaalmasinatel avalikud pordid vahemikus 30 000 – 32 767 järgneva käsuga:
    a. az aks nodepool update --resource-group <GRUPI NIMI> --cluster-name <KLASTRI NIMI> --name <NODE POOL> --allowed-host-ports 30000-32767/tcp
5. Loo uus Azure MySQL andmebaas koos järgnevate seadistustega:
    a. MySQL versioon – 8.0;
    b. Autentimine – *MySQL authentication only*;
    c. Ühendumise meetod – *Public Access*;
    d. Tulemüür – *Allow public access from any Azure service within Azure to this server*;
    e. Lisaks vali „*Add current client IP address*“.
6. Liigu andmebaasi lehele ning vali alamleht nimega „*Databases*“, peale mida loo uus rakenduse jaoks uus andmebaas.
7. Andmebaasi ja rakenduse ühilduvuse tagamiseks lae rakenduse koodihoidlast alla fail „prisma/schema.sql“, paigalda oma arvutisse MySQL käsurea liides ning jooksuta järgnevat käsklust:
    a. *mysql -h <ANDMEBAASI_IP> -u <KASUTAJANIMI> -p <ANDMEBAAS> < schema.sql*
8. Loo Communication Services ning Email Communication Services teenused.

9. Liigu Email Communication alamlehele „*Provision domains*“ ning vali „*Azure domain*“.
10. Liigu Communication Services alamlehele „*Domains*“ ning vali „*Connect domain*“, peale mida luuakse e-posti teenusega ühendus.
11. Loo uus Storage account ning liigu alamlehele „*File shares*“, kus tuleb luua uus „*File share*“ koos „*Hot*“ tasemega.
12. Loo uus Container Apps teenus järgnevate seadistustega:
    a. Kettapildi allikas – *Docker Hub or other registries*;
    b. *Registry login server – ghcr.io*;
    c. *Image and tag – ghcr.io/eistre/terminal*;
    d. Täida keskkonnamuutujate väljad vastavalt koodihoidlas oleva „.env.example“ faili järgi;
    e. *Ingress – Enabled*;
    f. *Ingress traffic – Accepting traffic from anywhere*;
    g. *Client certificate mode – Ignore*;
    h. *Target port* – 3000;
    i. *Session affinity – Enabled.*
13. Võta lahti loodud rakenduse Container Apps Environment keskkond, vali alamleht „*Azure Files*“ ning lisa 11. sammus loodud „*File share*“. Määra „*Access mode*“ väärtuseks „*Read/Write*“.
14. Liigu rakenduse Container App lehele, vali alamleht „*Identity*“ ning muuda „*System assigned*“ staatus „*On*“ olekusse.
15. Peale seda vali „*Azure role assignments*“ ning vajuta „*Add role assignment*“. Lisa järgnev roll:
    a. Skoop – *Resource group*;
    b. Roll – *Contributor*.
16. Liigu rakenduse Container App lehele ning vali alamleht „*Revisions and replicas*“ ning vajuta „*Create new revision*“.
17. Liigu „*Volumes*“ lehele ning lisa eelnevalt lisatud „*Azure File*“.
18. Mine tagasi „*Container*“ lehele ning vajuta rakenduse konteineri peale.
19. Vali „*Volume mounts*“, lisa eelnevalt lisatud „*Azure File*“ ning määra „*Mount path*“ väärtuseks */usr/src/app/.ssh*.

II. Rakenduse lokaalse variandi paigaldamise juhend

Järgnevalt on välja toodud rakenduse lokaalse variandi paigaldamise juhend:

1. Paigaldage arvutisse Kubernetes (soovitatavalt MicroK8s).
2. Paigaldage arvutisse Node.js (alates versioonist 21).
3. Laadige koodihoidlast alla lähtekood käsuga:
    a. *git clone https://github.com/eistre/terminal.git*
4. Liikuge käsureaga alla laetud projekti kausta.
5. Veenduge, et teie arvutis on aktiivne MySQL andmebaas. Kui töötav andmebaas puudub, võite kasutada projektis näitena välja toodud Docker Compose faili. Selleks veenduge, et Docker oleks paigaldatud ning käivitage järgnev käsk:
    a. *docker compose up -d*
    b. Vastava näite puhul kasutage „.env“ failis *DATABASE_URL* keskkonna muutuja väärtust: *mysql://root:root@localhost:3306/terminal*
6. Käivitage projekti kaustas järgnevad käsud:
    a. *npm install* – paigaldab rakenduse jooksutamiseks vajalikud teegid;
    b. *npx prisma generate* – genereerib Prisma kliendi;
    c. *npm run build* – ehitab rakenduse valmis.
7. Looge fail „.env“ ning täitke see vastavalt „.env.example“ faili järgi.
8. Rakendage rakenduse andmebaasi skeema loodud andmebaasile järgneva käsuga:
    a. *npx prisma db push*
9. Käivitage rakendus käsuga:
    a. *npm run preview*

III. Baasharjutuste komplektid

Järgnevalt on välja toodud rakenduses kasutatud baasharjutuste komplektid eesti ja inglise keeles.

```
export const createSimpleExercises = async () => {
  const { id: ee } = await db.exercise.create({
    data: {
      title: 'Sissejuhatus Linuxisse',
      description: 'Sissejuhatavad ülesanded Linuxi käsureasse'
    }
  })

  const { id: en } = await db.exercise.create({
    data: {
      title: 'Introduction to Linux',
      description: 'Introductory tasks for the Linux command line'
    }
  })

  // Luuakse esimese komplekti eestikeelsed alamülesanded
  await db.task.createMany({
    data: [
      {
        title: 'Ülesanne 1 - Failisüsteem',
        content: 'Kuva oma praeguse kausta sisu.',
        hint: 'Kausta sisu kuvamiseks saab kasutada käsku `ls`',
        regex: /ls[\s\S]+?simple_file\s/.source,
        exercise_id: ee
      },
      // Ülejäänud 8 alamülesannet
      {
        title: 'Ülesanne 10 - Protsessid',
        content: 'Kuva kõik käimasolevad protsessid.',
        hint: 'Kõikide käimasolevate protsesside kuvamiseks saab kasutada käsku `ps`',
        regex: /PID\s+?TTY\s+?TIME\s+?CMD/.source,
        exercise_id: ee
      }
    ]
  })

  // Luuakse esimese komplekti ingliskeelsed alamülesanded
  await db.task.createMany({
    data: [
      {
        title: 'Task 1 - File system',
        content: 'Display the contents of your current directory.',
        hint: 'To display the contents of a directory, you can use the `ls` command',
        regex: /ls[\s\S]+?simple_file\s/.source,
        exercise_id: en
      },
      // Ülejäänud 8 alamülesannet
      {
        title: 'Task 10 - Processes',
        content: 'Display all running processes.',
        hint: 'To display all running processes, you can use the `ps` command',
        regex: /PID\s+?TTY\s+?TIME\s+?CMD/.source,
        exercise_id: en
```

```
      }
    ]
  })
}

export const createNormalExercises = async () => {
  const { id: ee } = await db.exercise.create({
    data: {
      title: 'Operatsioonisüsteemid ja Andmeturve',
      description: 'Operatsioonisüsteemide ja Andmeturve ainete Linuxi käsurea harjutusülesanded'
    }
  })

  const { id: en } = await db.exercise.create({
    data: {
      title: 'Operating Systems and Computer Security',
      description: 'Linux command line tasks for the Operating Systems and Computer Security subjects'
    }
  })

  // Luuakse teise komplekti eestikeelsed alamülesanded
  await db.task.createMany({
    data: [
      {
        title: 'Ülesanne 1 - Faili sisu lugemine',
        content: 'Kuva faili `/etc/hosts` faili sisu käsureale.',
        hint: 'Kasuta käsku `cat`',
        regex: /127\.0\.0\.1[\s\S]*localhost[\s\S]*ip6-
localhost[\s\S]*localnet[\s\S]*allnodes[\s\S]*allrouters/.source,
        exercise_id: ee
      },
      // Ülejäänud 8 alamülesannet
      {
        title: 'Ülesanne 10 - Protsessid',
        content: 'Kuva kõik käimasolevad protsessid.',
        hint: 'Uuri käsku `ps`',
        regex: /\d+.*\d+:\d+.*inotifywait/.source,
        exercise_id: ee
      }
    ]
  })

  // Luuakse teise komplekti ingliskeelsed alamülesanded
  await db.task.createMany({
    data: [
      // 10 ingliskeelset alamülesannet
    ]
  })
}
```

IV. Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, Taavi Eistre,

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose **„Veebipõhise Linuxi käsurea õppekeskkonna tootestamine“**, 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.

Taavi Eistre

13.05.2024