Anssi Korhonen

Vakioitu verkko-oppimisympäristö oppimisanalytiikan mahdollistajana

Insinööri (AMK)

Tieto- ja viestintätekniikka

Kevät 2023

KAMK • University of Applied Sciences

KAMK • University of Applied Sciences

Tiivistelmä

Tekijä: Korhonen Anssi

Työn nimi: Vakioitu verkko-oppimisympäristö oppimisanalytiikan mahdollistajana

Tutkintonimike: Insinööri (AMK), tieto- ja viestintätekniikka

Asiasanat: verkko-oppimisympäristö, oppimisanalytiikka, Visual Studio Code, MOOC, Jupyter, Docker

Tässä työssä tarkasteltiin Kajaanin ammattikorkeakoulussa kehitettyä avoimen lähdekoodin komponentteihin perustuvaa verkko-oppimisympäristöä, jossa myös oppimisanalytiikan hyödyntäminen on mahdollista. Ensisijainen tavoite oli tehdä jo tehty työ näkyväksi eli kuvata ympäristön toimintaperiaatteet, ja tämän jälkeen sekä päivittää ohjelmistojen vanhentuneet versiot tuoreisiin että korjata ympäristössä havaitut mahdolliset virheet. Toissijaisena tavoitteena oli muodostaa ideoita ympäristön jatkokehittämiseksi ja mahdollisesti myös toteuttaa nämä ideat käytännössä.

Lopputuloksena saavutettiin verkko-oppimisympäristö, joka hyödyntää taustalla JupyterHub-teknologiaa, mutta avaa käyttäjän nettiselaimeen Microsoftin Visual Studio Code -ohjelmointieditorin. Ohjelmointieditoriin on valmiiksi asennettuna Helsingin yliopiston TestMyCode-laajennus, mikä mahdollistaa Helsingin yliopiston tarjoaminen avointen MOOC-kurssien suorittamisen ympäristössä.

Opiskelijan kannalta verkko-oppimisympäristön etu on siinä, että hänen ei opintojensa alkuvaiheessa tarvitse käyttää aikaa ohjelmointiympäristöjen asentamiseen, vaan hän pääsee suoraan aloittamaan itse ohjelmoinnin opettelemisen. Lisäksi ympäristön ollessa vakioitu jokaiselle opiskelijalle vältytään mahdollisilta yhteensopivuusongelmilta ja versioristiriidoilta. Verkko-oppimisympäristön käyttö vähentää myös laitevaatimuksia opiskelijan kannalta: riittää, että hänen tietokoneensa kykenee suorittamaan modernia nettiselainta.

Verkko-oppimisympäristöön kuuluu myös oppimisanalytiikan mahdollistaminen. Tämä toteutettiin käyttämällä avoimeen lähdekoodiin perustuvaa Wakapi-ohjelmistoa, joka puolestaan hyödyntää WakaTimen API-rajapintaa ajankäyttötilastojen keräämiseen. Mikäli opiskelija luo käyttäjätunnuksen verkko-oppimisympäristön arkkitehtuuriin sisällytettyyn Wakapi-instanssiin, alkaa Wakapi keräämään dataa hänen ohjelmointieditorin käytöstään. Opiskelija pääsee näkemään datasta muodostettuja visualisointeja Wakapin käyttöliittymän kautta. Opettajaa varten ympäristöön asennettiin Grafana, johon luotiin visualisointien koontinäkymä.

Ihannetapauksessa oman ajankäyttödatan näkeminen synnyttää opiskelijassa oivalluksia, motivoi häntä ja auttaa häntä kehittämään omaa oppimistaan. Opettajan kannalta data voi puolestaan olla arvokasta hänen pyrkiessään kehittämään kurssiaan ja omaa opetustaan. Datasta opettaja näkee esimerkiksi, mitkä tehtävät ovat vaatineet opiskelijoilta eniten aikaa ja mitkä näyttäisivät sujuneen vaivattomasti.

Seuraavassa kehitysvaiheessa voisi pohtia sitä, olisiko ajankäyttödatan lisäksi myös jotain muuta dataa, jota voitaisiin kerätä. Yhtenä vaihtoehtona tässä työssä nostettiin ehdotus datan keräämisestä siitä, kuinka monta kertaa opiskelija on yrittänyt tehdä tehtävää ennen kuin suoritus on mennyt hyväksytysti läpi. Tähän tehtiin myös toteutusehdotus, mutta toteutusta ei kuitenkaan viety käytäntöön asti.

Lisäksi seuraavassa vaiheessa voisi olla hyödyllistä laajentaa opettajaa varten luotua Grafana-koontinäkymää kattavammaksi. Näkymään voisi lisätä visualisointeja, jotka nopeuttavat sen hahmottamista, miten opiskelijoiden ajankäyttö jakautuu kurssin eri tehtävien kesken. Lisäksi opettajan työtä voitaisiin helpottaa luomalla erillinen nettisivu, johon Grafanan näkymät sisällytettäisiin. Tällä tavoin opettajan ei tarvitsisi erikseen kirjautua Grafanaan ja hakea koontinäkymää sitä kautta.

KAMK • University of Applied Sciences

Abstract

Author: Korhonen Anssi

Title of the Publication: Standardized Virtual Learning Environment as an Enabler of Learning Analytics

Degree Title: Bachelor of Engineering, Information and Communication Technology

Keywords: virtual learning environment, learning analytics, Visual Studio Code, MOOC, Jupyter, Docker

The focus of this work was a virtual learning environment developed in Kajaani University of Applied Sciences, which is based in open-source applications and has the capability to utilize learning analytics. The primary objective was to make the work that had already been done visible, that is, to describe how the environment works, and then to update the outdated versions of applications and fix any potential errors detected in the environment. The secondary objective was to generate ideas for further development of the environment and possibly implement these ideas in practice.

As a result, a virtual learning environment was achieved, which utilizes JupyterHub-technology in the background, but opens a Microsoft Visual Studio Code code editor to the user's web browser. The TestMyCode extension from the University of Helsinki is pre-installed in the code editor, which makes it possible to complete MOOC courses of the University of Helsinki in the environment.

From the student's perspective, the advantage of the virtual learning environment is that they don't have to spend time installing development environment but can start learning programming immediately. Additionally, standardizing the environment for all students, potential compatibility issues and version conflicts can be avoided. Using the virtual learning environment also decreases the hardware requirements from the student's perspective: all that is needed is a computer which can run a modern web browser.

Allowing learning analytics is part of the virtual learning environment. This was implemented using open-source Wakapi application, which in turn utilizes the WakaTime API to collect time usage metrics. If a student creates an account to the Wakapi instance, which is included as a part of the architecture of the learning environment, Wakapi starts collecting metrics of the student's usage of the code editor. The student can see the visualizations of the data via Wakapi user interface. For teacher's use Grafana was installed into the environment. In Grafana, a dashboard for viewing metrics was created.

In an ideal scenario, visualizations of time usage metrics generate insights in the student, motivate them, and help them to develop their learning. From the teacher's perspective, data can be valuable in their efforts to develop the course and their own teaching. From the data the teacher can see for example, which exercises have required the most time from students and which ones seemed to have been completed effortlessly.

In the next phase of developing the virtual learning environment it could be useful to consider whether there is any other data besides time usage data that could be collected. One suggestion in this work was to collect data on how many attempts a student has made before successfully completing an exercise. An example implementation proposal was also made for this, but it was not ultimately put into practice.

Additionally, in the next phase it might be useful to expand the Grafana dashboard created for the teacher to be more comprehensive. Visualizations could be added to the dashboard to help quickly grasp how students' time is distributed among the different exercises in the course. In addition, creating a separate web page that incorporates the Grafana dashboard could help streamline the teacher's workload. This way, the teacher wouldn't have to log in to Grafana separately and look for the dashboard that way.

Alkusanat

Jo opintojeni ensimmäisenä vuotena, perehtyessäni Linux-käyttöjärjestelmän saloihin, opin viisauden, jonka otin tuolloin talteen kuvankaappauksen muodossa. Toimikoon kuvankaappaus minulle muistutuksena ajankohdasta, kun tietokoneiden sielunelämä teknisestä näkökulmasta oli vasta aukeamassa minulle. Samalla kertokoon se siitä, miten olen oppinut rakastamaan sitä, kun saa mahdollisuuden kurkistaa pintaa syvemmälle. Se on kaunista, kun näkee, miten asiat todella toimivat!

```
[cloud-user@anssi-kt-centos7-harjoitus2 fortune]$ fortune | cowsay -f bud-frogs
 _____________________________________
/ Life would be so much easier if we  \
| could just look at the source code. |
|                                     |
\ -- Dave Olson                       /
 -------------------------------------
     \
      \
          oO)-.                       .-(Oo
         /__  _\                     /_  __\
         \  \(  |     ()~()         |  )/  /
          \__|\ |    (-___-)        | /|__/
          '  '--'    ==`-'==        '--'  '
```

Sisällys

1 Johdanto 1

2 Verkko-oppimisympäristö 3

2.1 Verkko-oppimisympäristön käsite 3

2.2 Verkko-oppimisympäristön arkkitehtuuri 4

2.3 Opiskelijalta vaadittavat toimenpiteet 11

2.3.1 Kirjautuminen verkko-oppimisympäristöön 12

2.3.2 Kurssien suorittaminen verkko-oppimisympäristössä 12

2.3.3 Oppimisdatan kerääminen 13

3 Oppimisanalytiikka 16

4 Verkko-oppimisympäristön päivittäminen ja korjaukset 20

4.1 Toimenpiteet ennen päivittämistä 20

4.2 Päivittäminen ja korjaukset 21

5 Verkko-oppimisympäristön kehityskohteita 24

5.1 Järjestelmän suorituskyvyn testaaminen 24

5.2 Opiskelijalta vaadittavien esitoimien vähentäminen 26

5.2.1 Tunnistautuminen 26

5.2.2 Kirjautuminen TMC-laajennukseen 28

5.2.3 Tehtävänantojen tarkasteleminen 28

5.2.4 Oppimisanalytiikan mahdollistaminen 29

5.3 Oppimisanalytiikan kehittäminen 30

5.3.1 Lisädatan kerääminen 30

5.3.2 Visualisointien kehittäminen – valmiit ratkaisut 34

5.3.3 Visualisointien kehittäminen – räätälöity ratkaisu 36

5.4 Tekoälyn yhdistäminen VSCodeen 41

6 Yhteenveto 42

Lähteet 45

Liitteet

1 Johdanto

Sekä pilvipalvelut että oppimisanalytiikka ovat päivän sanoja. Näistä ensiksi mainittu lienee sanana tuttu lähes jokaiselle, vaikka ei tietäisi tarkemmin, mitä sanan taakse kätkeytyy. Tekniikan alan ihminen tietää, että kun jokin sijaitsee ”pilvessä”, kyseessä on palvelinresurssi – esimerkiksi skaalattavissa oleva laskentateho tai jokin tietty sovellus – joka toimii internetin yli.

Oppimisanalytiikka ei sen sijaan käsitteenä ole yhtä tunnettu, eikä käsitettä esimerkiksi löydy tätä johdantoa kirjoitettaessa (huhtikuussa 2023) Yleisestä suomalaisesta ontologiasta. Oppimisanalytiikka on tutkimusalue, jossa yhdistyy kaksi hyvin erilaista tieteenhaaraa: kasvatustiede ja data-analytiikka. Datan keräämiseen, käsittelyyn ja myös analysointiin käytetään tietojenkäsittelytieteistä ja data-analytiikasta tuttuja menetelmiä, mutta varsinaisten johtopäätösten tekemiseen tarvitaan pedagogista näkemystä. Myös suunnitteluvaihe – mitä tietoa kerätään ja miksi kerätään – edellyttää kasvatustieteellistä ymmärrystä.

Vaikka oppimisanalytiikka ei ole käsitteenä yleisesti tunnettu, on se herättänyt laajaa kiinnostusta opetuksen ja koulutuksen alalla. Kiinnostuksesta kertovat esimerkiksi useat eri koulutustasoilla järjestetyt hankkeet aiheen ympärillä. Ammattikorkeakoulujen kohdalla tällainen hanke oli Oppimisanalytiikka – avain parempaan oppimiseen ammattikorkeakouluissa, eli APOA. Myös Opetus- ja kulttuuriministeriö on ottanut kantaa aiheeseen laatimalla oppimisanalytiikan viitekehyksen, jossa esitetään hyvät käytännöt oppimisanalytiikan käyttöönottoon ja hyödyntämiseen. Oppimisanalytiikka herättää kiinnostusta, koska siinä nähdään potentiaalia opetuksen ja oppimateriaalin laadun parantamiseen, yksilöllisen oppimisen edistämiseen ja myös mahdollisten ongelmien tunnistamiseen.

Tässä työssä tarkastellaan vakioitua verkko-oppimisympäristöä, johon sisällytetään myös oppimisanalytiikan hyödyntämisen mahdollisuus. Vakioitu verkko-oppimisympäristö tarkoittaa tässä tapauksessa sitä, että se toimii Kajaanin ammattikorkeakoulun ”pilvessä” ja ympäristö on samanlainen kaikille opiskelijoille riippumatta opiskelijan omasta tietokoneesta ja sen tehosta.

Tällä tavoin voidaan varsinkin aloittaville opiskelijoille tarjota niin sanotusti lentävä lähtö ohjelmoinnin opettelemiseen. Opiskelijan ei tarvitse käyttää aikaa eri ympäristöjen asenteluun, eikä vastaan voi myöskään tulla versioristiriitoja tai yhteensopivuusongelmia, kun ympäristö ja sitä suorittava rauta on sama kaikilla. Lisäksi laitteistovaatimukset pienenevät opiskelijan kannalta: opiskelija tarvitsee vain verkkoyhteydellä varustetun koneen, jossa toimii moderni nettiselain.

Oppimisanalytiikan ottaminen mukaan tarjoaa mahdollisuuksia sekä opiskelijoille että opettajille. Opiskelija pääsee näkemään, minkä verran hän on käyttänyt oppimisympäristössä aikaa kunakin päivänä ja miten tuo käytetty aika on jakautunut esimerkiksi eri kurssien, eri tehtävien tai eri ohjelmointikielien välillä. Opettajaa data voi auttaa esimerkiksi tunnistamaan, mitkä asiat näyttäisivät olevan opiskelijoille hankalia ja mitkä helppoja, ja onko jopa jotain asioita, joihin tekeminen näyttäisi tyssäävän kokonaan. Tällainen tieto auttaa arvioimaan, miten kurssia voisi jatkossa kehittää, mutta siitä voi olla hyötyä jo kurssin aikana; opettaja voi esimerkiksi suunnitella datan perusteella, mitä asioita ottaa puheeksi seuraavilla luennolla ja mitä kysyä opiskelijoilta.

Kajaanin ammattikorkeakoulun teknologian lehtori Jaakko Vanhala on kehittänyt edellä mainitun verkko-oppimisympäristön versiota, joka toimii tämän opinnäytetyön pohjana. Lähtötilanteessaan verkko-oppimisympäristö on vielä pelkistetty ja siitä puuttuu joitakin toivottuja ominaisuuksia. Lisäksi ympäristön käyttöönotto vaatii alussa jokaiselta opiskelijalta muutamien asetusten asettamista käsin, mikä ihannetilanteessa voitaisiin automatisoida kokonaan.

Tämän työn tarkoituksena on kuvata verkko-oppimisympäristön toiminta lähtötilanteessa, päivittää ympäristö käyttämään ohjelmistojen uusia versioita, korjata mahdolliset havaitut virheet, etsiä kehittämiskohteita ja laatia toteuttamisehdotuksia näille tunnistetuille kehittämiskohteille. Tavoite on, että syksyllä 2023 aloittavat opiskelijat voivat ottaa päivitetyn ympäristön käyttöön opintojensa alussa.

2 Verkko-oppimisympäristö

Tässä luvussa käsitellään avoimen lähdekoodin komponenteista koostuvaa verkko-oppimisympäristöä. Aluksi määritellään verkko-oppimisympäristön käsite, jonka jälkeen esitellään työssä käytettävän verkko-oppimisympäristön arkkitehtuuri lähtötilanteessa. Luvun lopuksi tarkastellaan niitä toimenpiteitä, joita opiskelijalta vaaditaan, jotta hän pääsee hyödyntämään ympäristön eri ominaisuuksia.

2.1 Verkko-oppimisympäristön käsite

Perusopetuksen opetussuunnitelman perusteissa [1] oppimisympäristön käsite määritellään seuraavasti: ”Oppimisympäristöillä tarkoitetaan tiloja ja paikkoja sekä yhteisöjä ja toimintakäytäntöjä, joissa opiskelu ja oppiminen tapahtuvat” [1, s. 29]. Koulutuksen tietoyhteiskuntakehittäminen 2020 -raportissa [2] mainitaan erikseen myös tekniset ratkaisut: ”Oppimisympäristö-käsitteellä tarkoitetaan erilaisia paikkoja, tiloja, yhteisöjä, teknisiä ratkaisuja, välineitä tai toimintatapoja, jotka tukevat oppimista” [2, s. 24].

Oppimisympäristön voidaan siis ajatella olevan jokin paikka tai tila, ja tämä paikka tai tila voi olla myös teknisesti toteutettu. Kun puhutaan teknisesti toteutetusta oppimisympäristöstä, käsitteiden kirjo laajenee. Aiheeseen perehtymällä huomataan, että käytössä on useita eri toisiaan vastaavia tai vähintäänkin toistensa lähikäsitteitä: verkko-oppimisympäristö, digitaalinen oppimisympäristö, oppimisalusta ja virtuaalinen oppimisympäristö. Lähikäsitteinä puhutaan oppimisjärjestelmistä ja oppimisen hallintajärjestelmistä.

Eräitä verkko-oppimisympäristöjä tai oppimisen hallintajärjestelmiä ovat esimerkiksi Moodle [3], Peda.net [4], ViLLE [5] ja Google Classroom [6].

Tässä työssä käytetään käsitettä verkko-oppimisympäristö ja sillä viitataan erityisesti luvussa 2.2 esiteltävään tekniseen oppimisympäristöratkaisuun, jonka ajatuksena on mahdollistaa tiettyjen ohjelmointikurssien suorittaminen Kajaanin ammattikorkeakoulussa. Tämä verkko-oppimisympäristö tulee ajatella osana laajempaa oppimisen kokonaisuutta Kajaanin ammattikorkeakoulussa. Verkko-oppimisympäristön painopiste on ohjelmoinnin ja modernien ohjelmistotyökalujen käyttämisen oppimisessa. Esimerkiksi oppimiseen liittyvä vuorovaikutus tapahtuu tämän verkko-oppimisympäristön ulkopuolella, tyypillisesti joko luokkahuoneessa tai eri teknisiä välineitä, kuten

Microsoft Teams -sovellusta, sähköpostia tai Discord-viestisovellusta käyttäen. Työkalu vuorovaikutuksen tuomiseksi suoraan työssä esiteltävän kaltaiseen ympäristöön on kuitenkin olemassa [7], mutta sitä ei ole testattu tämän verkko-oppimisympäristön tapauksessa.

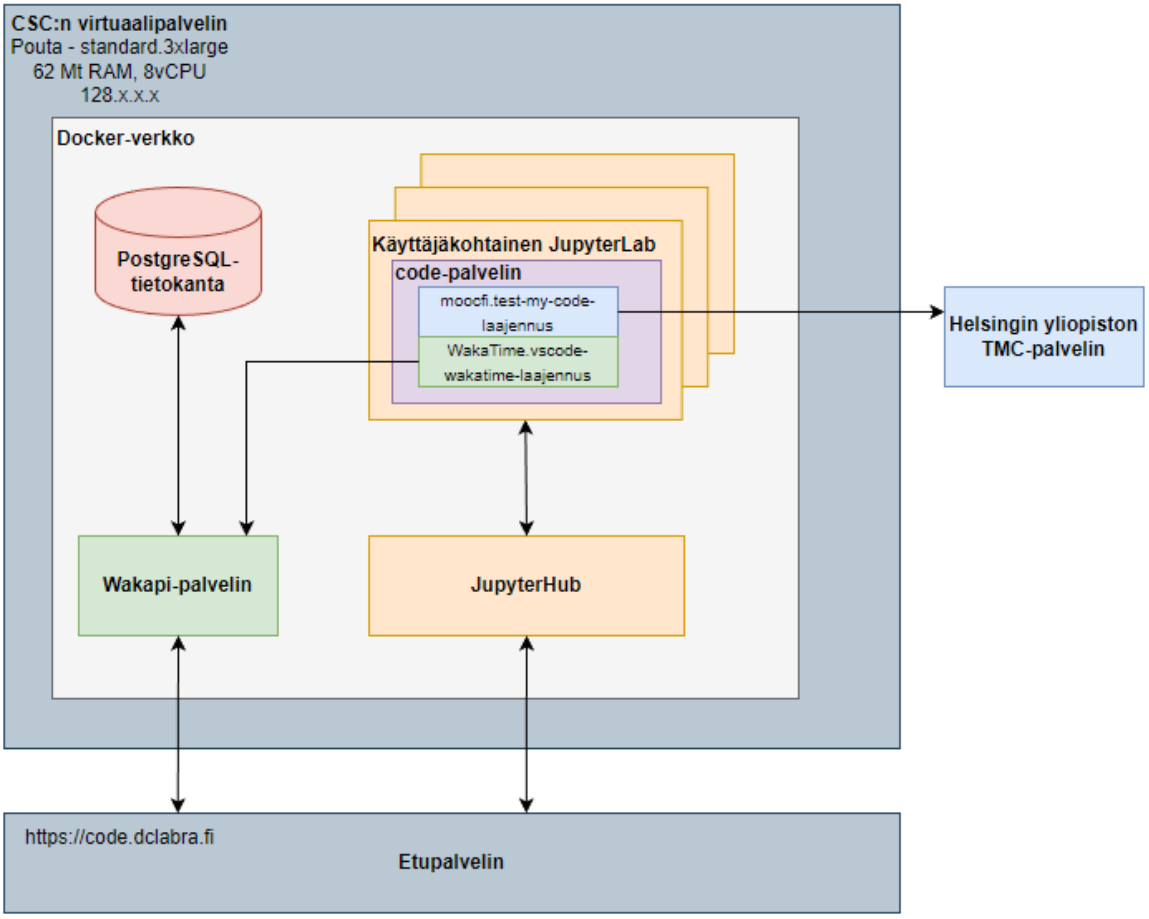
Samalla tämä verkko-oppimisympäristö kattaa ajatuksen oppimisanalytiikan mahdollistamisesta, mitä käsitellään luvussa *3 Oppimisanalytiikka*.

2.2 Verkko-oppimisympäristön arkkitehtuuri

Verkko-oppimisympäristön arkkitehtuuri on esitelty kuvassa 1. Kuten kuvasta käy ilmi, itse ympäristöä ajetaan CSC:n virtuaalipalvelimella ja liikennöinti ulkoverkkoon tapahtuu ammattikorkeakoulun etupalvelimen kautta. Yhteen liimaavana komponenttina ympäristössä toimii Jupyter.

Jupyter on kokoelma työkaluja, jotka mahdollistavat web-pohjaisen hajautetun ohjelmistokehityksen laajalle käyttäjäjoukolle [8]. Näistä keskeinen komponentti on JupyterLab, joka toimii yhden käyttäjän käyttöliittymänä. Sen avulla käynnistetään ja hallinnoidaan eri ohjelmointieditorien muistioita eli niin sanottuja notebookeja. Lisäksi tiedostojen hallinta ja komentotulkin käyttö on JupyterLabin avulla mahdollista. [9.] Toinen keskeinen komponentti on JupyterHub, jonka avulla voidaan huolehtia käyttäjien tunnistamisesta ja palvella laajaa käyttäjäjoukkoa [10], kuten vaikkapa kaikkia ohjelmoinnista kiinnostuneita korkeakouluopiskelijoita.

Erityisenä uutena ideana tässä verkko-oppimisympäristössä toimii Code-palvelin [11], joka on asennettu Jupyterin yhteyteen. Code-palvelin mahdollistaa Microsoftin Visual Studio Code -ohjelmointieditorin käyttämisen nettiselaimella. Code-palvelin on maalattu arkkitehtuurikuvassa violetilla värillä ja esitelty yhdessä muiden verkko-oppimisympäristön komponenttien kanssa kuvan alapuolella.



Kuva 1. Verkko-oppimisympäristön arkkitehtuuri

Etupalvelin. Liikenne verkko-oppimisympäristöön tapahtuu etupalvelimen kautta. Etupalvelin sijaitsee ammattikorkeakoulun DC-labra-palvelimella.

CSC-virtuaalipalvelin. Varsinainen verkko-oppimisympäristö sijaitsee CSC:n virtuaalipalvelimella. Ympäristö on rakennettu samaan Docker-verkkoon, jossa kaikkia ympäristön osia suoritetaan erillisissä Docker-konteissa. Virtuaalipalvelin ei suoraan vastaa pyyntöihin ulkopuolelta, vaan se keskustelee etupalvelimen kanssa ja vaatii sertifikaattien olevan ajan tasalla. Kuvassa 2 esitellään CSC:n Pouta-palvelun [12] virtuaalipalvelininstanssien hallinnointinäkymä. Esimerkkikuvan näkymässä on nähtävillä kaksi instanssia: testipalvelin uusien päivitysten ja ominaisuuksien testaamiseksi sekä alkuperäinen, sittemmin päivittämätön palvelin.

Instance ID = | Filter | Launch Instance | Delete Instances | More Actions

Displaying 2 items

Instance Name	Image Name	IP Address	Flavor	Key Pair	Status	Availability Zone	Task	Power State	Time since created	Actions
anssi-test	-	192.168.1.17 Floating IPs: 128.[redacted]	standard.large	[redacted]	Active	nova	None	Running	1 week, 1 day	Create Snapshot
code-server-cpu-kamk	-	192.168.1.16 Floating IPs: 128.[redacted]	standard.3xlarge	[redacted]	Active	nova	None	Running	1 year, 2 months	Create Snapshot

Kuva 2. CSC:n Pouta-palvelun virtuaalipalvelinten hallinnointinäkymä

Docker-verkko. Kaikki verkko-oppimisympäristöön liittyvät toiminnallisuudet ovat kontitettu Docker-teknologialla ja kaikki kontit toimivat saman Docker-verkon sisällä. Tässä tapauksessa kaikki kontit sijaitsevat myös samalla virtuaalipalvelimella, mutta näin ei toki tarvitse olla. Esimerkiksi Kubernetes-teknlogia [13] mahdollistaa laajan hajautetun arkkitehtuurin luomisen. Tässä tapauksessa verkko-oppimisympäristöä ei kuitenkaan ole tarkoitus kytkeä osaksi muita kokonaisuuksia, eikä Kuberneteksen asentaminen pelkästään yhdelle virtuaalipalvelimelle toisi merkittävää lisäarvoa. Kuvassa 3 esitellään, miltä virtuaalipalvelimen komentotulkissa näyttää, kun käynnissä olevat kontit listataan `docker ps` -komennolla. Nyt kuvassa näkyvä konttien määrä on maltillinen, mutta otettaessa ympäristö aktiiviseen käyttöön tulee kontteja yksi lisää aina jokaista käyttäjää kohden. Kuvasta on leikattu näkymää porttien ohjauksista sekä poistettu konttien id-arvoa koskeva sarake, jotta kuvaa on saatu kapeammaksi.

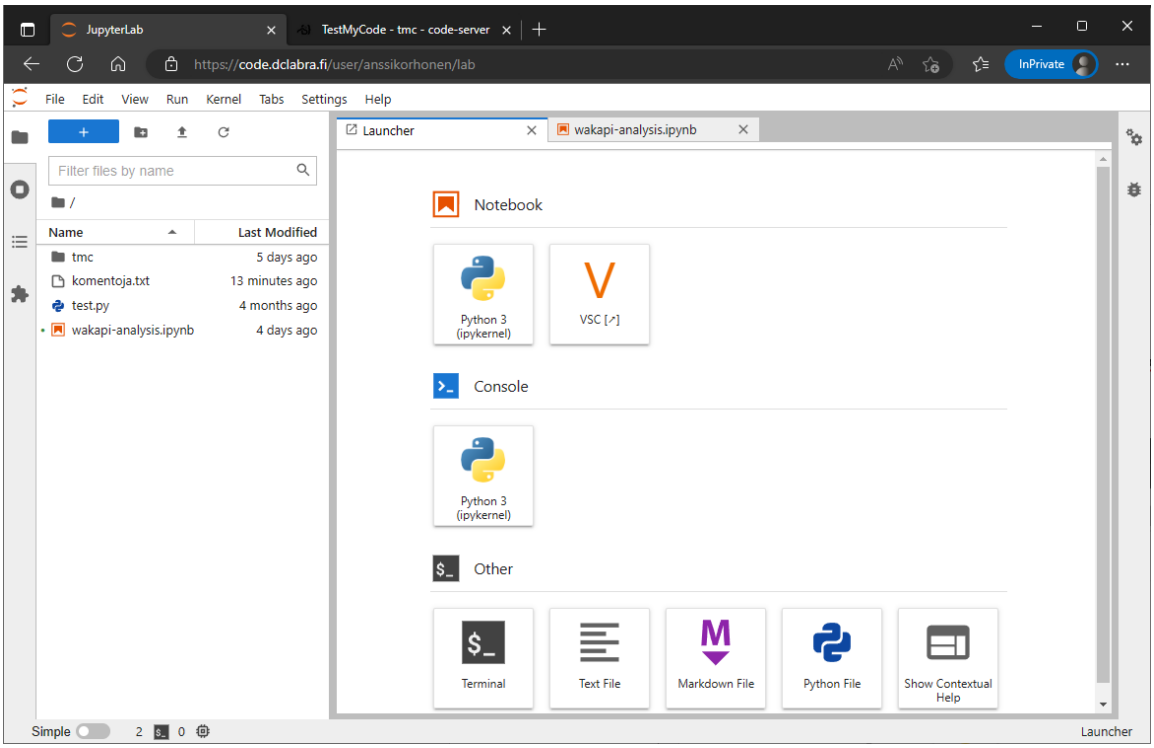
```
IMAGE                          COMMAND                  CREATED         STATUS        PORTS        NAMES
kamk/code-server-cpu:latest    "tini -g -- start-no…"   3 months ago    Up 5 weeks    8888/tcp     jupyter-vs
kamk/code-server-cpu:latest    "tini -g -- start-no…"   4 months ago    Up 4 days     8888/tcp     jupyter-anssikorhonen
kamk/code-server-cpu:latest    "tini -g -- start-no…"   4 months ago    Up 5 weeks    8888/tcp     jupyter-jaakkovan
code-server-kamk_jupyterhub    "jupyterhub"             4 months ago    Up 7 weeks    0.0.0.0:8(   jupyterhub
ghcr.io/muety/wakapi:latest    "/bin/sh -c /app/ent…"   4 months ago    Up 7 weeks    3000/tcp,    wakapi-server
postgres:12.3                  "docker-entrypoint.s…"   4 months ago    Up 4 months   5432/tcp     wakapi-db
kamk/code-server-cpu           "tini -g -- start-no…"   10 months ago   Up 7 weeks    8888/tcp     code-server-kamk_vscode-cpu_1
```

Kuva 3. Docker-kontit virtuaalipalvelimella

JupyterHub. JupyterHub [10] on keskitetty palvelin, joka huolehtii Jupyter-instanssien perustamisesta yksittäisille käyttäjille. Autentikoinnissa käytetään OAuth-menetelmää [14] ja käyttäjä tunnistetaan Kajaanin ammattikorkeakoulun GitLab-käyttäjätunnusten avulla.

Käyttäjäkohtainen JupyterLab. JupyterLab on ohjelmistokehitysympäristö, johon on mahdollista asentaa useita eri ohjelmointikieliä ja avata näistä omia muistiovälilehtiä eli niin sanottuja notebookeja [9]. Tässä tapauksessa erityisyytenä myös Code-palvelin on asennettu JupyterLabin yhteyteen ja Code-palvelin käynnistetään vastaavaan tapaan kuin tavalliset muistiovälilehdetkin.

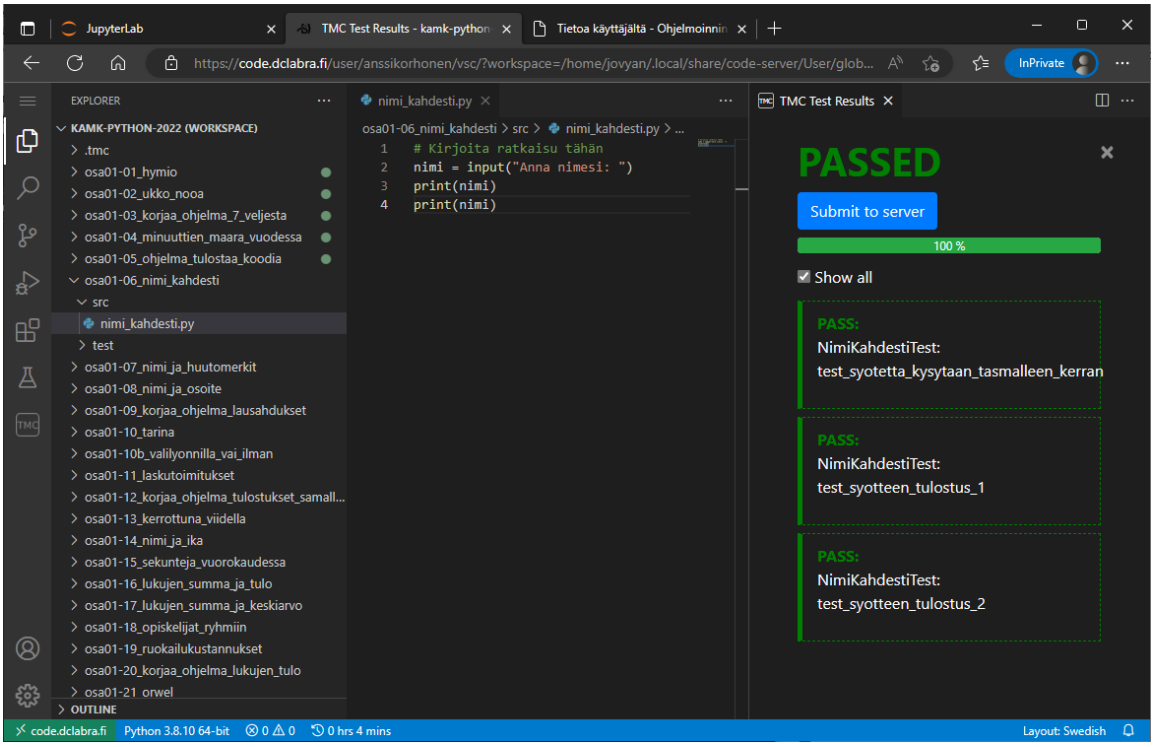
Kuvassa 4 on esimerkki käyttäjäkohtaisesta instanssista. Käyttäjän tiedostot näkyvät kuvassa vasemmassa palkissa ja eri toimintojen käynnistysnäkymä oikeassa palkissa. Käynnistettäessä eri toimintoja, kuten vaikkapa Python-muistio, aukeavat ne tyypillisesti JupyterLabin omiin sisäisiin välilehtiin. Tässä Code-palvelin tekee kuitenkin poikkeuksen sen auetessa nettiselaimen uuteen välilehteen. Sivuhuomiona mainittakoon, että kuvassa Code-palvelimen käynnistimen kuvake näkyy virheellisesti pelkkänä V-kirjaimena. Tämä korjataan päivityksen yhteydessä luvussa *4.2 Päivittäminen ja korjaukset*.



Kuva 4. Käyttäjäkohtainen JupyterLab-näkymä

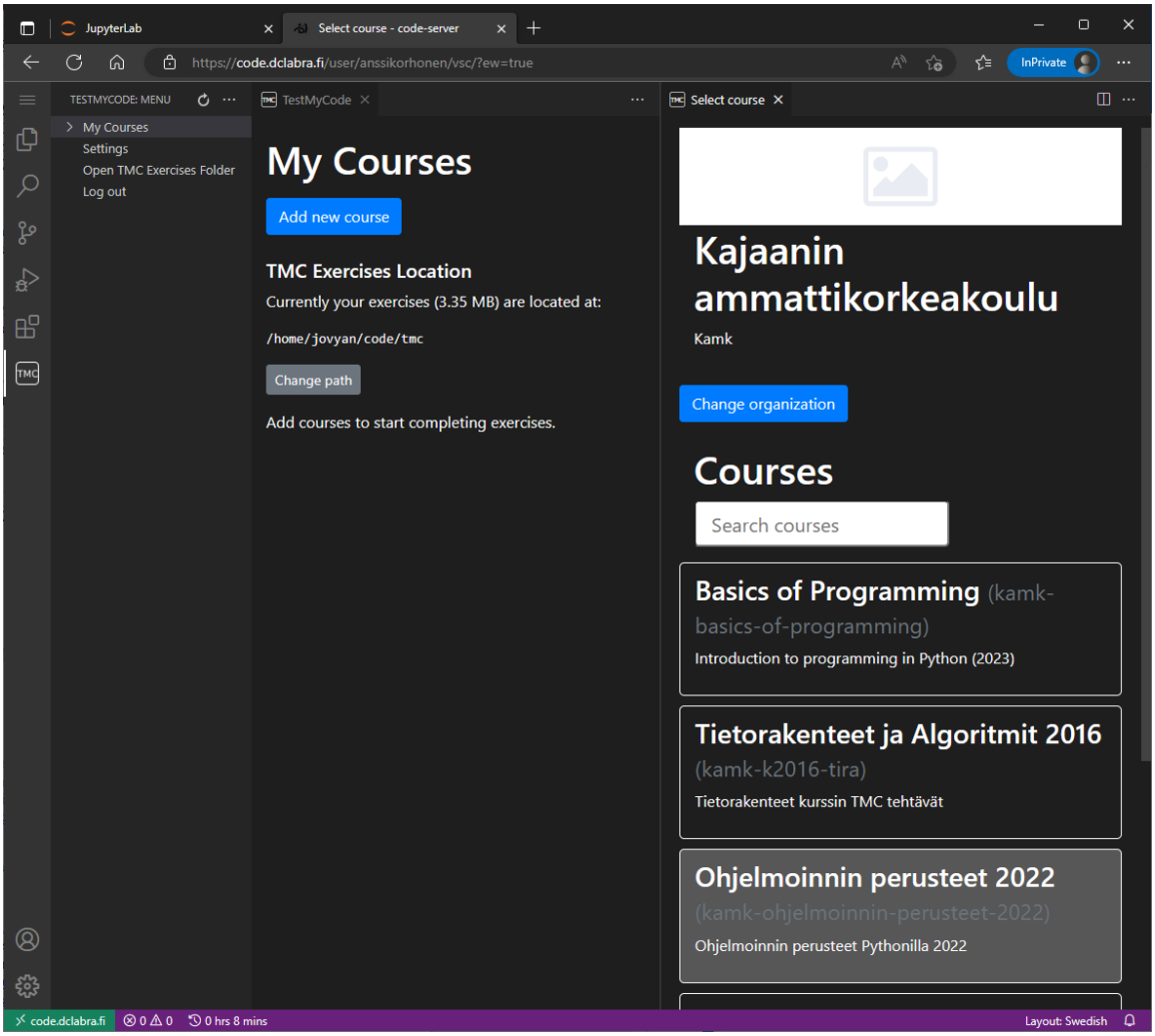
Code-palvelin. Microsoft Visual Studio Code (jatkossa VSCode) on vapaan lähdekoodin ohjelmointieditori [15], josta on kehitetty myös web-selaimella toimiva versio [16]. Tässä yhteydessä hyödynnetään nimenomaan web-selaimella toimivaa versiota VSCodesta, joka esiasennetaan jokaisen Jupyter-käyttäjän henkilökohtaiseen instanssiin. Juuri VSCoden hyödyntäminen tekee Jupyter-ohjelmistokehitysympäristöstä tässä tapauksessa myös verkko-oppimisympäristön. Eri VSCoden laajennukset mahdollistavat eräiden ohjelmointikurssien suorittamisen ja myös oppimisanalytiikan ottamisen mukaan (oppimisanalytiikasta, ks. luku 3). VSCodeen voidaan asentaa käytännössä lähes minkä tahansa ohjelmointikielten tulkkeja ja kääntäjiä. Tällä tavoin on mahdollista rakentaa hyvin kattava kehitysympäristö, joka voidaan räätälöidä käyttäjätaholle mieleisiin paketteihin. Kaikki tarpeellinen on esiasennettuna valmiina. Tässä verkko-oppimisympäristössä pää-

huomio on Python-ohjelmointikielessä, mutta toimivuutta on testattu myös GCC-käännösohjelmakokoelmalla [17], jossa on tuki muun muassa C- ja C++-ohjelmointikielille. Kuvassa 5 näkyy esimerkkitilanne, kuinka verkko-oppimisympäristön VSCodea hyödyntäen voidaan suorittaa ohjelmointikurssia. Kuvassa käyttäjä on juuri kirjoittanut oman ratkaisunsa ja ajanut sen test-my-code-laajennuksen testiohjelman läpi.



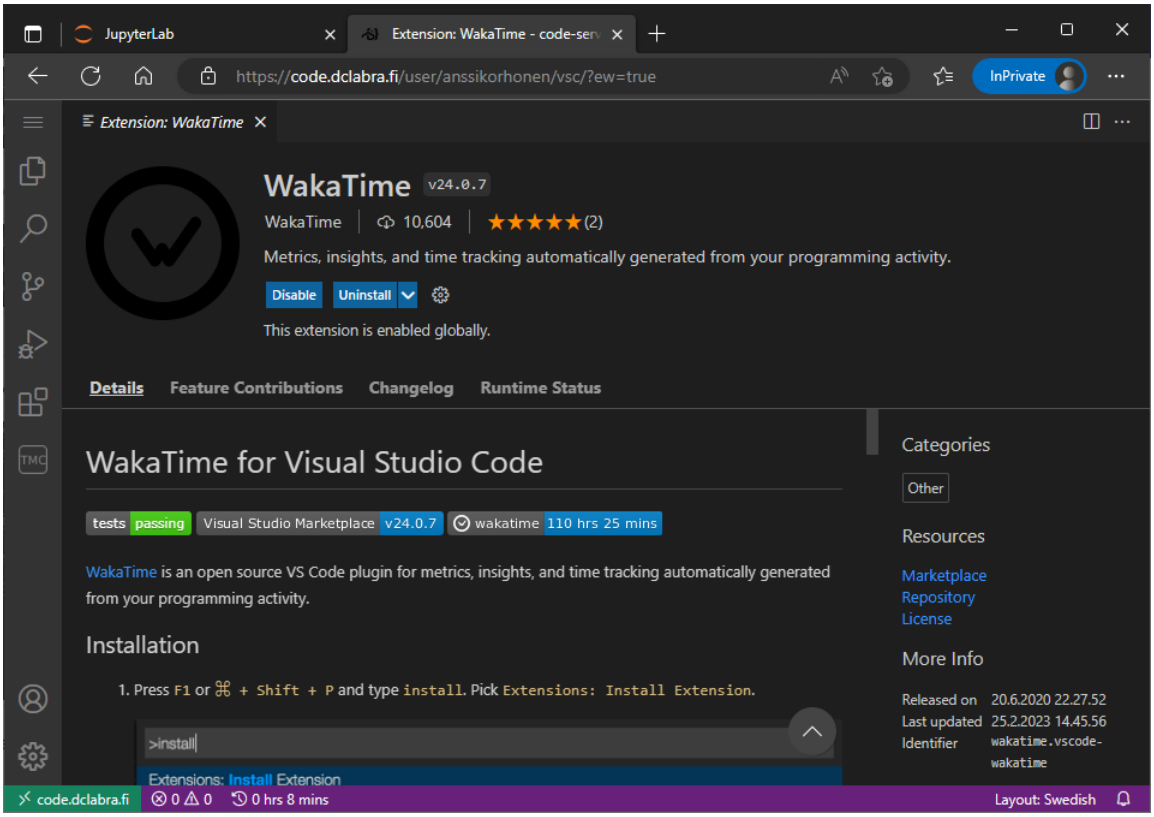
Kuva 5. Tehtävien suorittaminen VSCodella

Moocfi.test-my-code-laajennus. Test-my-code on Helsingin yliopiston tietojenkäsittelytieteen laitoksella kehitetty ja Agile Education Research -tutkimusryhmän ylläpitämä avoimen lähdekoodin laajennus [18], joka mahdollistaa Helsingin yliopiston useiden eri ohjelmointikurssien suorittamisen. Laajennuksen avulla tehtävät voi ladata omalle alustalle ja tämän jälkeen tehtäviä voi suorittaa omassa VSCode-ympäristössä. Laajennuksen sisältämä testiohjelma antaa palautteen, onko suoritus mennyt oikein vai väärin, ja tietyissä tapauksissa antaa opiskelijalle myös vihjeitä oikean ratkaisun löytämiseksi. Kun käyttäjä on tyytyväinen vastaukseensa, voi hän laajennuksen avulla lähettää sen Helsingin yliopiston palvelimelle. Tällöin eteneminen kurssilla on nähtävissä myös ”virallisesti”. Kuvassa 6 on esimerkkitilanne, kuinka käyttäjä on valitsemassa uutta suoritettavaa kurssia TMC-laajennuksen avulla.



Kuva 6. Kurssin valinta VSCoden TMC-laajennuksessa

WakaTime.vscode-wakatime-laajennus. WakaTime tarjoaa avoimeen lähdekoodiin perustuvia laajennuksia ohjelmistoihin, kuten ohjelmointieditoreihin [19]. Laajennukset keräävät ajankäyttötilastoja, joita voi hyödyntää oman toimintansa analysointiin. Laajennukset ovat ilmaisia, mutta WakaTime ei itsessään tarjoa mahdollisuutta tallentaa kertyviä tilastoja paikallisesti tai omille palvelimille. Luomalla tunnuksen WakaTimen nettisivuilla [19] voi käyttäjä generoida itselleen API-avaimen, jolloin data tallennetaan WakaTimen tietokantoihin. Käyttäjä pääsee käsiksi kahden viimeisimmän viikon analytiikkatietoihinsa, mutta historiatietoihin pääseminen on maksullista. Kuvassa 7 on esimerkkinäkymä VSCodeen asennetusta WakaTime-laajennuksesta.



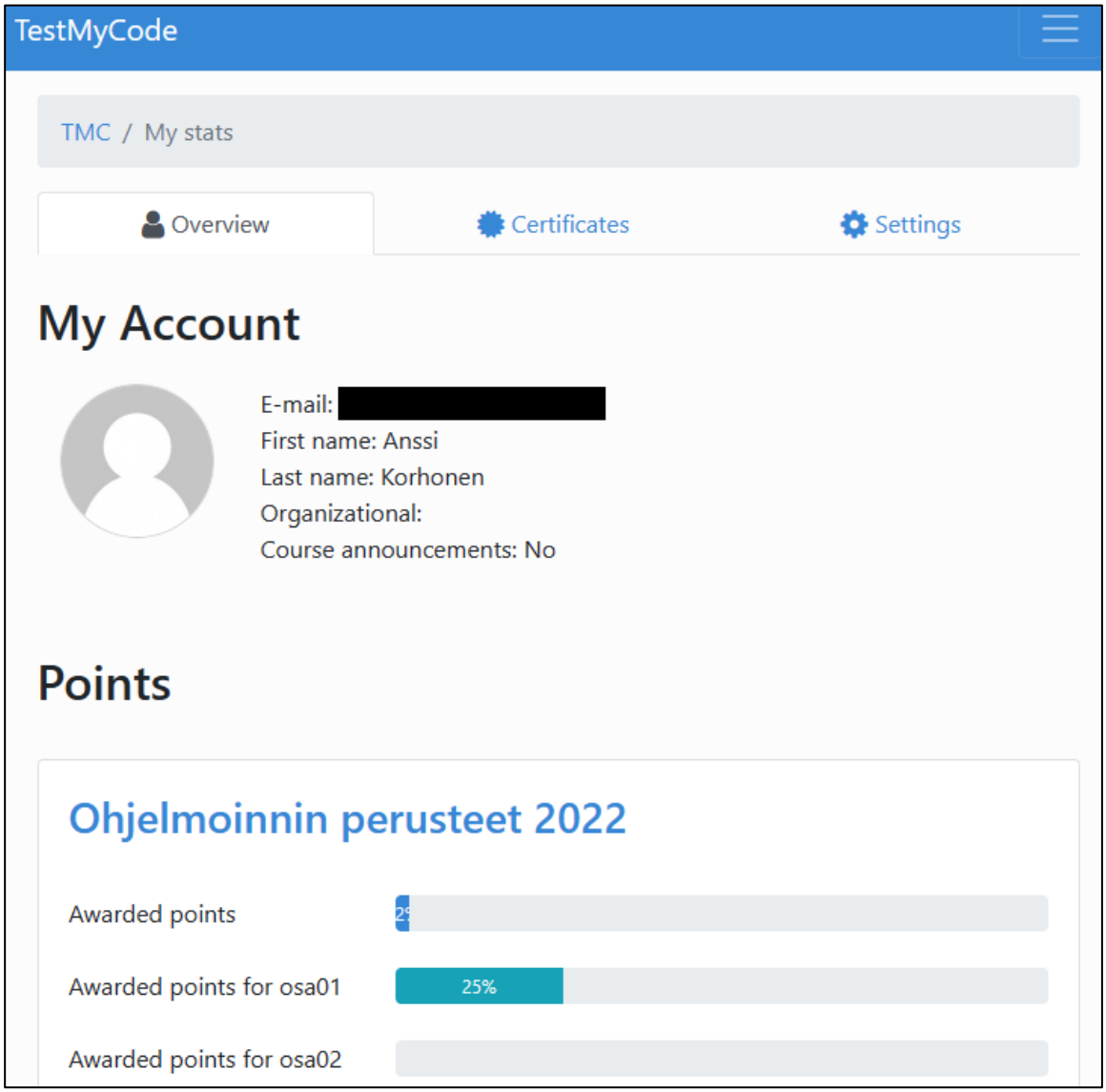
Kuva 7. VSCoden WakaTime-laajennus

Wakapi-palvelin. Wakapi on avoimen lähdekoodin ratkaisu WakaTimen luomien laajennusten hyödyntämiseksi niin, että kertyneet ajankäyttötiedot voi tallentaa paikallisesti tai omalle Wakapi-palvelininstanssille. Wakapi-palvelimeen kuuluu myös web-palvelinsivu, jossa käyttäjä voi katsella visualisointeja omasta ajankäytöstään. Web-palvelinsivu on siis osa itse ylläpidettävää palvelininstanssia ja jotta käyttäjästä voidaan kerätä dataa, tulee käyttäjän luoda käyttäjätunnus tähän instanssiin.

PostgreSQL-tietokanta. PostgreSQL on avoimeen lähdekoodiin perustuva objekti-relaatio-tietokanta [20]. Wakapi-palvelin käyttää tätä WakaTime-datan tallentamiseen. Oppimisanalytiikkaa käsittelevässä luvussa *3 Oppimisanalytiikka* on esimerkkikuva (kuva 10, s. 18) Wakapi-palvelimen tietokantaan tallentamasta tietueesta.

Helsingin yliopiston TMC-palvelin. TMC- eli TestMyCode-palvelin on Helsingin yliopiston palvelin, jonne VSCoden TMC-laajennus lähettää tiedon ohjelmointikurssin suoritetuista tehtävistä. TMC-palvelimen arkkitehtuurin tietäminen ei ole tässä rakennetun verkko-oppimisympäristön kannalta oleellista, mutta opiskelijan kannalta keskeistä on se, että nimenomaan Helsingin yliopiston palvelimelle kirjautuneet tehtäväsuoritukset otetaan huomioon kurssin arvioinnissa. Suorituksia

voi tämän jälkeen tarkastella myös kirjautumalla MOOC.fi-tunnuksilla (ks. luku *2.3.2 Kurssien suorittaminen verkko-oppimisympäristössä*) TestMyCoden sivuille [21]. Kuvassa 8 on esimerkki kurssien suoritusnäkymästä TMC:n sivujen kautta.



Kuva 8. Kurssin suoritusten tarkastelu TestMyCode-sivuston kautta

2.3 Opiskelijalta vaadittavat toimenpiteet

Voidakseen käyttää verkko-oppimisympäristöä opiskelija tarvitsee verkkoyhteydellä varustetun tietokoneen, joka kykenee suorittamaan modernia nettiselainta. Tämän lisäksi opiskelijalta vaaditaan muutamia esivalmisteluja, jotta 1) hän voi kirjautua verkko-oppimisympäristöön, 2) hän voi suorittaa ympäristössä kursseja ja 3) jotta oppimisdataa voidaan kerätä ja esittää.

2.3.1 Kirjautuminen verkko-oppimisympäristöön

Voidakseen kirjautua ympäristöön opiskelijan tulee tunnistautua. Tunnistautumisessa käytetään OAuth2-tunnistautumisprotokollaa [22]. Tämä OAuth2-protokollan avulla muodostettu tunnistautumisidentiteetti luodaan GitLabin avulla [23]. GitLab-instanssina toimii Kajaniin Ammattikorkeakoulun omilla palvelimilla [24] suoritettava GitLab Ultimate [25]. Verkko-oppimisympäristö, jonka alustana on JupyterHub, puolestaan tarjoaa mahdollisuuden hyödyntää GitLab-käyttäjätunnuksilla kirjautumista. Tämä tietenkin edellyttää sitä, että opiskelijan on ensin rekisteröidyttävä ammattikorkeakoulun GitLabiin ennen kuin hänen on mahdollista käyttää verkko-oppimisympäristöä.

Muitakin tunnistautumistapoja on olemassa OAuth2-protokollan lisäksi. Osaa tunnistamistavoista Jupyter tukee suoraan, osa voisi vaatia huomattavan paljon määrittelyä, eikä toimivuus olisi silti taattu. Erilaisia tunnistautumisvaihtoehtoja on tarkasteltu luvussa *5.2 Opiskelijalta vaadittavien esitoimien vähentäminen*.

Verkko-oppimisympäristöön kirjautumisen lisäksi opiskelijan on annettava vielä erikseen salasana käynnistäessään Code-palvelimen. Salasana on etukäteen asetettu ja kaikille käyttäjille sama. Alkuperäinen idea erillisen salasanan asettamiseksi oli rajoittaa Code-palvelimen käyttäjäjoukkoa ympäristön testausvaiheessa. Käyttäjillä oli pääsy itse Jupyter-ympäristöön, mutta Code-palvelimen käyttöä rajattiin resurssien riittävyyden varmistamiseksi. Ympäristön päivittämisen yhteydessä tämä salasanavaatimus otetaan pois, kuten todetaan luvussa *4.2 Päivittäminen ja korjaukset*.

2.3.2 Kurssien suorittaminen verkko-oppimisympäristössä

Verkko-oppimisympäristön VSCode-ohjelmointieditoriin on esiasennettuna TMC-laajennus (ks. luku *2.2 Verkko-oppimisympäristön arkkitehtuuri*). Opiskelija tarvitsee kuitenkin Helsingin yliopiston MOOC.fi-tunnukset TMC:n käyttöön. MOOC tulee sanoista Massive Open Online Course ja tarkoittaa nimensä mukaisesti massiivista avointa verkkokurssia, eli kurssia, johon on pääsy kaikilla [26]. Tunnukset luodaan MOOC.fi-palvelussa ja rekisteröitymisen yhteydessä käytetään Kajaanin ammattikorkeakoulun sähköpostiosoitetta.

Kun opiskelija on luonut MOOC.fi-tunnukset, käyttää hän niitä kirjautumiseen TMC-laajennukseen VSCodessa. Kirjautumisen jälkeen opiskelija lataa suoritettavan kurssin tehtävät laajennuksen kautta. Kun tehtävät on ladattu, niitä on mahdollista suorittaa.

Tehtävien suorittamisen aikana opiskelijalla on lisäksi tarpeen olla erikseen auki nettiselaimen välilehti, jossa tehtävien varsinaiset kysymykset ovat [kuten esimerkiksi 27]. VSCoden TMC-laajennus lataa ympäristöön tehtävien suoritus- ja palautusrungon, mutta ei itse kysymyksiä.

Ladatut kurssit jäävät talteen opiskelijan henkilökohtaiseen instanssiin ja pysyvät tallessa myös nettiselaimen sulkemisen jälkeen. Kuitenkin, mikäli opiskelijalla on riittävän pitkä aikaväli edellisestä kirjautumisesta (säädettävissä), JupyterHub sulkee kontin, jossa opiskelijan instanssia suoritetaan. Vaikka käyttäjän tiedostot pysyvät tallessa, suljettaessa kontti palautuu lähtötilaansa. Tämä tarkoittaa sitä, että opiskelijan on kirjauduttava uudelleen TMC-laajennukseen ja valittava suorituksessa olevat kurssit uudelleen. Itse tehtävät ja suoritukset jäävät talteen, mutta TMC-laajennus ei tunnista niitä ennen uudelleenkirjautumista ja kurssien uudelleenvalitsemista.

2.3.3 Oppimisdatan kerääminen

Jotta kurssin suorittamisesta voidaan kerätä oppimisen analysoinnin kannalta merkityksellistä dataa, edellytetään opiskelijalta muutamia alkutoimenpiteitä. Tämä vaihe ei kuitenkaan ole kurssin suorittamisen kannalta välttämätön ja opiskelijalla on mahdollisuus olla osallistumatta datan tuottamiseen.

Opiskelijan kannalta datan kerääminen voi olla motivoivaa, koska hän pääsee itse näkemään visualisointeja datasta ja sitä myöten seuraamaan, kuinka paljon hän on käyttänyt aikaa esimerkiksi eri tehtävien ratkaisemiseen. Esimerkiksi Centria-ammattikorkeakoulun pilotissa opiskelijat kokivat oppimisanalytiikkaa hyödyntävän itsearvioinnin jopa erittäin hyödylliseksi [28, s. 28]. Myös Satakunnan ammattikorkeakoulun pilotissa opiskelijat kokivat mahdollisuuden seurata omaa etenemistään kurssilla motivoivaksi [29, s. 17–18]. Lapin ammattikorkeakoulussa tehtiin havainto oppimisanalytiikan mahdollisuudesta edistää tiimien työskentelyä [30].

Lisäksi erityisesti data-analytiikan opiskelijoiden kannalta yhtenä motivaattorina voidaan pitää sitä, että he pääsevät näkemään, mitä datan kerääminen analytiikkatarkoituksiin voi konkreettisella tasolla tarkoittaa. Kun opiskelijoiden tietämys kasvaa, voidaan myöhemmillä kursseilla tuoda

ilmi, miltä kerätty data näyttää tietokantatasolla, miten tietokantaan päästään käsiksi ja miten opiskelija voi itse visualisoida ja analysoida tietokannassa olevaa dataa eri työkaluja käyttäen.

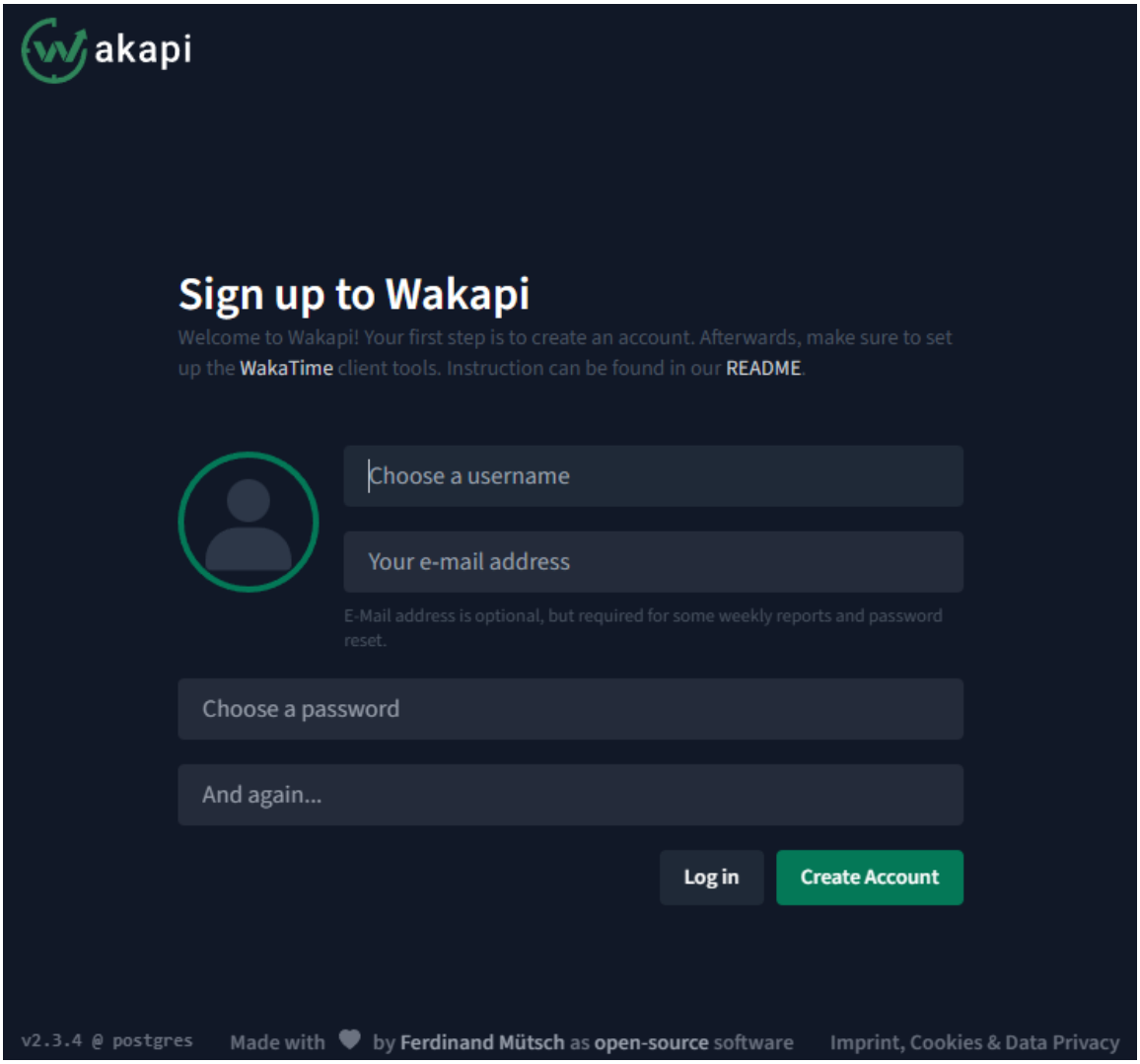
Jotta tätä dataa voidaan kerätä, tarvitaan opiskelijalta paitsi suostumus myös muutamia konkreettisia toimenpiteitä. Tehtävien suorittamiseen käytettyä aikaa seurataan avoimeen lähdekoodiin perustuvaa WakaTime-laajennusta [31] ja WakaTimen API-rajapintaa [32] hyödyntämällä. WakaTime-laajennus on esiasennettuna verkko-oppimisympäristössä, mutta laajennuksen käyttö vaatii API-avaimen asettamisen. Tämä opiskelijan täytyy tehdä itse.

WakaTime tarjoaa itsessään mahdollisuuden luoda käyttäjätunnukset ja henkilökohtaisen API-avaimen. Tämän reitin käyttäminen tarkoittaisi kuitenkin sitä, että data tallentuisi WakaTimen palvelimille, sitä ei pääsisi tarkastelemaan tietokantatasolla ja lisäksi käyttäjän olisi mahdollista tarkastella datan visualisointeja ilmaiseksi ainoastaan kahden viimeisen viikon ajalta.

Tämän vuoksi käytetään lisäksi toista avoimeen lähdekoodiin perustuvaa ohjelmistoa, Wakapia [33]. Wakapi hyödyntää WakaTimen API-rajapintaa ja WakaTimen eri ohjelmistoihin tekemiä laajennuksia [34] (tässä tapauksessa VSCoden laajennusta), mutta generoi oman API-avaimen ja mahdollistaa datan tallentamisen omaan tietokantaan. Tällä tavoin voidaan varmistaa, että kontrolli dataan säilyy itsellä ja kaikkiin historiatietoihin on mahdollista päästä käsiksi.

Opiskelijan kannalta tämä tarkoittaa sitä, että hän luo ammattikorkeakoulun ylläpitämälle Wakapi-palvelimelle tunnukset ja ottaa tunnusten luomisen yhteydessä saamansa API-avaimen talteen. Tämä API-avain tallennetaan VSCoden kautta verkko-oppimisympäristöstä löytyvään WakaTimen määrittelytiedostoon.

Tässä tapauksessa Wakapi-palvelimeen yhdistetään saman etupalvelimen kautta kuin varsinaiseen verkko-oppimisympäristöönkin. Wakapin web-käyttöliittymä on kuitenkin määritelty palvelemaan eri portissa ristiriitojen välttämiseksi. Kuvassa 9 esitellään näkymä Wakapin web-käyttöliittymän tunnustenluontisivusta.



Kuva 9. Wakapin web-käyttöliittymän tunnustenluontisivu

Wakapille on olemassa ainakin yksi varteenotettava vaihtoehto nimeltä Hakatime, joka Wakapin tapaan hyödyntää WakaTimen API-rajapintaa ja mahdollistaa datan hallinnoimisen itse [35]. Hakatimea vaihtoehtona Wakapille pohditaan tarkemmin sekä luvussa *5.2.4 Oppimisanalytiikan mahdollistaminen* että luvussa *5.3.2 Visualisointien kehittäminen – valmiit ratkaisut*.

3 Oppimisanalytiikka

Oppimisanalytiikka on oppijoita ja heidän oppimisympäristöään koskevan datan mittaamista, keräämistä, analysointia ja raportointia sillä tarkoituksella, että sekä oppimista että oppimisympäristöjä voidaan ymmärtää ja optimoida [36]. Tällä tavoin George Siemens määritteli oppimisanalytiikan käsitteen kutsuessaan tutkijoita lähettämään tutkimusartikkeleita 2011 pidettyyn ensimmäiseen oppimisanalytiikan kansainväliseen konferenssiin.

Tämän jälkeen tutkimusta on kertynyt tasaisesti vuosien aikana, oppimisanalytiikka on saanut oman akateemisen aikakauslehtensä [37] ja aihe on alkanut herättää kiinnostusta myös Suomessa. Suomessa on järjestetty eri koulutustasoille useita eri hankkeita, jotka suoraan koskevat oppimisanalytiikkaa [ks. esim. 38, 39, 40, 41, 42 ja 43]. Näistä APOA-hankkeen [39] aikana laadittiin Oppimisanalytiikan käsikirja [44], jonka tarkoitus on auttaa syventämään tietämystä aiheesta. Tämän lisäksi on hankkeita ja tahoja, jotka koskevat laajemmin digitaalisen oppimisen edistämistä, mutta sivuavat osin myös oppimisanalytiikkaa [ks. esim. 45, 46, 47, 48 ja 49]. Näistä Suomen eOppimiskeskus [45] ylläpitää Oppimisanalytiikkaverkosto-blogia [50]. Turun yliopistoon on perustettu Oppimisanalytiikan tutkimusinstituutti [51]. Lisäksi Opetus- ja kulttuuriministeriö laatinut oppimisanalytiikan viitekehyksen, jossa esitetään hyvät käytännöt oppimisanalytiikan käyttöönotossa ja hyödyntämisessä [52].

Koska tämän työn painopiste on tekninen, huomio kiinnitetään itse oppimisanalytiikan mahdollistamiseen. Tämän työn puitteissa huolehditaan datan keräämisestä, sen tallentamisesta ja myös joidenkin visualisointien tuottamisesta. Kuitenkin varsinainen analysoiminen rajataan työn ulkopuolelle. Työssä ei pyritä vastaamaan siihen, mitä kerätylle datalle tulisi lopulta tehdä ja millaisia johtopäätöksiä siitä kannattaisi esittää.

Oppimisanalytiikkaa voidaan tarkastella erilaisten kysymysten kautta. Eräässä viittausmallissa esitellään oppimisanalytiikan neljä ulottuvuutta [53, s. 321], jotka on esitetty yksinkertaisina kysymyksinä:

- **Mitä?** Millaista dataa järjestelmä kerää, hallinnoi ja käyttää analytiikkaa varten?
- **Kuka?** Keitä analysoidaan?
- **Miksi?** Miksi kerättyä dataa analysoidaan?
- **Miten?** Miten kerätyn datan analysointi toteutetaan?

Näiden kysymysten kautta hahmottuu, mihin oppimisanalytiikan ulottuvuuksiin on tässä työssä mielekästä ottaa kantaa. Tässä työssä keskeistä on datan mittaaminen, kerääminen ja osin myös raportointi. Tämän myötä työn puitteissa voidaan vastata kolmeen ensimmäiseen kysymykseen. Koska varsinainen analysointi rajataan työn ulkopuolelle, jää samalla analysoinnin toteutusta koskeva kysymys vastaamatta.

Taulukossa 1 esitellään, millaista dataa järjestelmä kerää. Taulukkoon on listattu tietueeseen tallennettavat keskeiset kentät.

Taulukko 1. Tietokantaan tallennettava data

Kenttä	**Lisätieto**
Opiskelijan Wakapi-käyttäjätunnus	Opiskelijan itsensä nimeämä, opiskelijan yksilöivä tieto
Kohteen nimi	Kohde, jonka parissa käyttäjä työskentelee; yleensä tiedoston nimi
Kohteen tyyppi	Esimerkiksi tiedosto
Toiminnan tyyppi	Esimerkiksi koodaaminen
Päätelty projektin nimi	Tyypillisesti nykyisen työkansion nimi
Ohjelmointikieli	Esimerkiksi Python
Kirjoittaako käyttäjä parhaillaan	boolean
Käytettävä editori	Tässä tapauksessa aina vscode
Käyttöjärjestelmä	Tässä tapauksessa aina linux
Tietokoneen isäntänimi	Tässä tapauksessa JupyterLab-instanssin Docker-kontin id
Käyttäjäagentti	wakatime-laajennuksen versio

Lisäksi tallennetaan aina tietueen id ja hash eli tarkiste. Tietueeseen tallennetaan myös kolme muuta kenttää: branch, origin, origin_id, koska ne ovat osa Wakapin tietokannan taulun rakennetta. Nämä kentät jäävät kuitenkin aina tyhjiksi silloin, kun ympäristössä suoritetaan kurssitehtäviä.

Kuva 10 havainnollistaa kerättävää dataa esimerkkitietueen kautta. Siinä on nähtävillä konkreettinen esimerkki siitä, miltä data näyttää tietokannassa.

```
-[ RECORD 108 ]--+-------------------------------------------------------------------------------------------------
id               | 315
user_id          | anssik
entity           | /home/jovyan/code/tmc/kamk-python-2022/osa01-01_hymio/src/hymio.py
type             | file
category         | coding
project          | osa01-01_hymio
branch           |
language         | Python
is_write         | f
editor           | vscode
operating_system | linux
machine          | 533313d13a2f
user_agent       | wakatime/v1.70.1 (linux-5.4.0-113-generic-x86_64) go1.19.7 vscode/1.63.0 vscode-wakatime/24.0.7
time             | 2023-03-31 10:02:46.07
hash             | 567ac72fcf15949c
origin           |
origin_id        |
created_at       | 2023-03-31 10:02:46.159
```

Kuva 10. Esimerkkitietue kerätystä datasta

Taho, jota analysoidaan, ovat ne verkko-oppimisympäristön käyttäjät, jotka ovat luoneet myös Wakapi-tunnuksen ammattikorkeakoulun palvelimelle.

Syy, miksi analysoidaan, on tarkasteltavissa sekä opiskelijan että opettajan näkökulmasta. Opiskelijan on mahdollista itse tarkastella kerätystä datasta muodostettuja visualisointeja ja siten arvioida omaa oppimistaan ja oppimisen vaatimaa aikaa eri kohdissa. Opettaja puolestaan voi arvioida, mitkä kurssin kohdat ovat vaatineet opiskelijoilta eniten aikaa ja mitkä näyttävät sujuneen nopeasti. Lisäksi opettaja näkee, milloin ja missä tahdissa tehtäviä on suoritettu – esimerkiksi tasaisesti kurssin aikana tai juuri ennen aikarajoja. Näitä tietoja opettaja voi käyttää opetuksen ja kurssin suunnittelussa.

Kuvassa 11 nähdään esimerkki kertyneen datan visualisoinnista Wakapin web-käyttöliittymän kautta. Kuvassa mainitut ”total heartbeats”, eli ”sydämen sykkeiden kokonaismäärä”, viittaa WakaTimen tapaan tallentaa dataa [54]. Käytännössä yksi ”heartbeat” tarkoittaa yhtä tietuetta tietokannassa kuvan 10 mukaisesti. Kuvassa näkyvät projektit saavat yleensä nimensä nykyisen työkansion nimen perusteella.

Kuva 11. Datan visualisointinäkymä Wakapin web-käyttöliittymässä

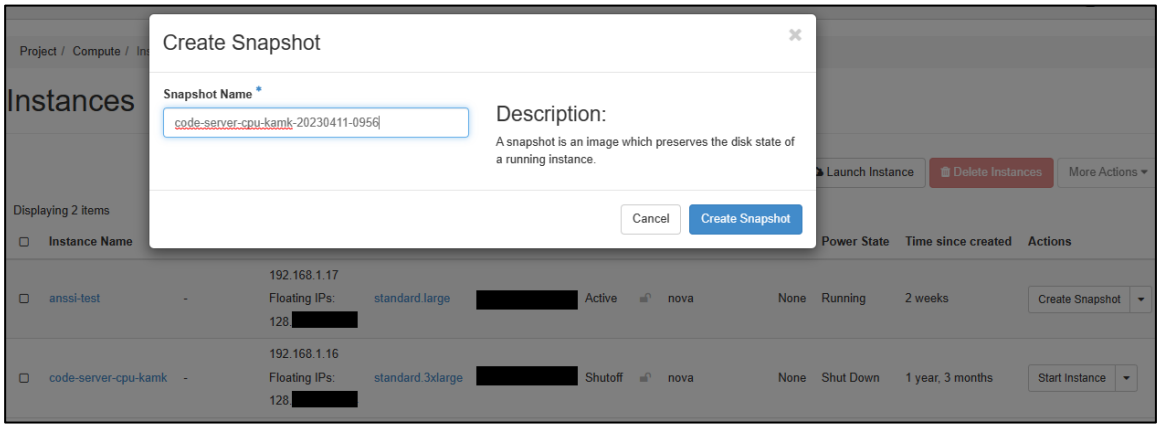
Kuten kuvasta nähdään, Wakapin tarjoamat visualisoinnit painottuvat piirakkakuvioihin. Visualisointivaihtoehtoja voidaan laajentaa erilaisilla lisäratkaisuilla. Näitä lisäratkaisuja käydään tarkemmin läpi luvuissa *5.3.2 Visualisointien kehittäminen – valmiit ratkaisut* ja *5.3.3 Visualisointien kehittäminen – räätälöity ratkaisu*.

4 Verkko-oppimisympäristön päivittäminen ja korjaukset

Ennen ympäristön päivittämistä on hyvä varmistaa, että itse päivitys ei johda peruuttamattomiin ongelmiin. Tämä varmistusprosessi käydään läpi ensin omassa alaluvussaan. Toisessa alaluvussa käsitellään varsinainen päivitys ja sen yhteydessä tehtävät korjaukset.

4.1 Toimenpiteet ennen päivittämistä

Ennen ympäristön päivittämistä otetaan nykyisestä ympäristöstä tilannekuva (*snapshot*), jotta mahdollisen ongelmatilanteen tapahtuessa voidaan palata ajankohtaan ennen päivittämistä. Tilannekuvan ottaminen onnistuu CSC:n Pouta-palvelun web-käyttöliittymän instanssien hallinnan kautta. Ennen tilannekuvan ottamista suositellaan instanssi sammutettavaksi, jotta varmistetaan tilannekuvan eheys [55]. Kuvassa 12 esitellään näkymä Pouta-palvelun web-käyttöliittymästä tilannekuvan luontitilanteessa.



Kuva 12. Tilannekuvan luominen virtuaalikoneen instanssille Pouta-palvelussa

Kun instanssista otetaan tilannekuva, luodaan automaattisesti tilannekuvat myös instanssiin liitetyistä tiedostojärjestelmistä (*volumes*). Tämän ympäristön tapauksessa tiedostojärjestelmiä on käytössä kaksi: 1) tiedostojärjestelmä käyttöjärjestelmän juurihakemistoa (*root*) varten ja 2) tiedostojärjestelmä käyttäjädataa varten. Kuvassa 13 nähdään esimerkki luoduista tiedostojärjestelmien tilannekuvista.

Volume Snapshots

2023 | Delete Volume Snapshots

Displaying 4 items

Name	Description	Size	Status	Volume Name	Actions
snapshot for code-server-cpu-kamk-20230411-0956	-	100GiB	Available	code-server-root-disk	Create Volume
snapshot for code-server-cpu-kamk-20230411-0956	Docker data volume for KAMK code-server	200GiB	Available	code-server-data-kamk	Create Volume

Kuva 13. Tiedostojärjestelmien tilannekuvien näkymä Pouta-palvelussa

Kun tilannekuva on luotu, voidaan päivittäminen aloittaa. Ihannetilanteessa tilannekuvaan ei tarvitse koskaan palata, vaan se jää pelkäksi varatoimenpiteeksi.

4.2 Päivittäminen ja korjaukset

Päivityksessä päivitetään arkkitehtuurikuvan (kuva 1, s. 5) mukaisen Docker-ympäristön kontit uusimpiin versioihinsa. Tietyssä tapauksessa noudatetaan kuitenkin jotakin tiettyä versiota, mutta ladataan siitä uusin vedos, mikäli sellainen on. Näin toimitaan esimerkiksi käyttäjäkohtaisten JupyterLab-konttien kanssa, joiden pohjana käytetään `jupyter/minimal-notebook`-konttia [56], mutta käyttöjärjestelmäksi valitaan Ubuntu 20.04, eikä esimerkiksi Ubuntu 22.04, joka on tuorein saatavilla oleva Ubuntu-versio.

JupyterHub. JupyterHubin osalta päivitys on vaivaton. Docker Hubista ladataan tuorein JupyterHubin levykuva ja vanha JupyterHub-kontti poistetaan. Muutoin itse JupyterHubin määrityksiä ei ole tarpeen muokata. Myöhemmin, kun myös muut kontit on päivitetty, ajetaan kontit ylös komennolla `docker-compose up --build`, jolloin muokatut määritteet ja päivitetyt levykuvat tulevat käyttöön sen sijaan, että Docker käyttäisi vanhoja, välimuistissa olevia aiemmin luotuja levykuvia. JupyterHubin päivitys korjaa myös käyttäjän ohjautumisen kirjautumisen jälkeen: nyt käyttäjä ohjautuu suoraan VSCode-ympäristöön, eikä hänen tarvitse klikata itseään sinne JupyterLab-ympäristön kautta. Päivitys nostaa JupyterHubin versiosta 2.0.2 versioon 3.1.1.

Käyttäjäkohtainen JupyterLab. Myös JupyterLabiin ladataan tuorein levykuva Docker Hubista, mutta käyttöjärjestelmäpohjana käytetään Ubuntu 20.04:ää eikä Ubuntun tuoreinta saatavilla olevaa LTS-versiota (*Long Time Support*, pitkäaikainen tuki) eli 22.04:ää. Tähän päädytään, koska itse virtuaalipalvelin, johon verkko-oppimisympäristö on asennettu, käyttää Ubuntu 20.04:ää. On

mahdollista, että Ubuntu 22.04:n käyttäminen kontissa tästä huolimatta ei silti aiheuttaisi ristiriitoja. Ratkaisuun päädytään kuitenkin varmuuden vuoksi. Ratkaisu kuitenkin pakottaa lataamaan ohjelmistopakettien uusimmat versiot Ubuntun tietovarastoista (*repositories*). Eli toisin sanoen kontin määritystiedostoon (Dockerfile), ennen muiden pakettien asentamista, täytyy kirjoittaa rivi `RUN apt update && apt upgrade -y`. Mikäli `upgrade` jätetään tekemättä, kontin levykuvan luominen kaatuu versioristiriitoihin. Lisäksi kontin määritystiedostoon lisätään VSCoden ikonin kopiointi konttiin. Samalla JupyterLabin määritystiedostoon (`jupyter_notebook_config.py`) VSCoden ikonin määrittely oikein. Näin VSCoden ikoni näkyy toivotulla tavalla JupyterLabin käynnistimessä. Tämän merkitys kuitenkin vähenee, koska itse JupyterHubin päivityksen jälkeen käyttäjä ohjataan suoraan VSCode-ympäristöön, eikä hän lähtökohtaisesti näe JupyterLab-ympäristöä lainkaan. Päivitys nostaa JupyterLabin versiosta 3.2.8 versioon 3.4.8.

Code-palvelin. Code-palvelin on tässä tapauksessa osa JupyterLab-konttia, eli Code-palvelimen päivitys tehdään muokkaamalla JupyterLab-kontin määritystiedostoa (`Dockerfile`). Tuorein versio ladataan Code-palvelimen GitHub-tietovarastosta [57]. Samalla erillinen salasanavaatimus Code-palvelimen käytölle otetaan pois. Kirjauduttuaan palvelimelle käyttäjälle avautuu VSCode-näkymä välittömästi. Päivitys nostaa Code-palvelimen versiosta 4.0.1 versioon 4.11.0.

Wakapi-palvelin. Kuten JupyterHubin, myös Wakapin päivittäminen on suhteellisen vaivatonta. Tosin hieman odottamatta ladattaessa Wakapin tuorein versio näkyy Wakapin web-käyttöliittymässä versionumeron sijaan teksti `dev`. Ei ole täysin selvää, mistä tämä johtuu, mutta Wakapin GitHub-tietovaraston tehtäviä (*issues*) tutkimalla selviää, että ainakin muutaman kerran aiemmin kehitystyössä versionumeron kohdalle on unohtunut väärä teksti [58 ja 59]. Lopulta selkeyden vuoksi päädytään käyttämään Wakapin toiseksi uusinta versiota, jossa versionumero näkyy oikein. Eli päivitys nostaa Wakapin versiosta 2.3.4 versioon 2.6.2.

PostgreSQL-tietokanta. Wakapin dataa varten olevan PostgreSQL-tietokannan yhteydessä joudutaan tekemään muutamia korjaustoimenpiteitä. Kontille itsessään ei sinänsä ladata uutta levykuvaa Docker Hubista. Näin siksi, että uutta levykuvaa ei Wakapin suosittelemalle PostgreSQL-versiolle ole ilmestynyt. Kuitenkin docker-composen määritystiedostosta (tässä tapauksessa `docker-compose-wakapi.yml`) on jäänyt aiemmalla asennuskerralla puuttumaan yksi rivi. Vaikka tietokannalle luodaan tiedostojärjestelmä (*volume*), itse tietokantakonttia ei tulla koskaan liittäneeksi (*mount*) tähän tiedostojärjestelmään. Toisin sanoen kaikki tähän saakka kertynyt data on pelkästään kontissa, mikä johtaa siihen, että poistamalla kontin myös data menetetään. Ennen vanhan kontin poistamista data kopioidaan luotuun tiedostojärjestelmään root-käyttäjänä komennolla `docker cp <vanhan kontin nimi>:/var/lib/postgresql/data`

`<tiedostojärjestelmän polku>/_data`. Tämän jälkeen docker-composen määritystiedostoon lisätään myös rivi tietokannan liittämisestä tähän tiedostojärjestelmään. Vanha kontti voidaan poistaa ja uusi kontti hakee datan liitetystä tiedostojärjestelmästä.

5 Verkko-oppimisympäristön kehityskohteita

Ympäristön kehityskohteita tarkastellaan neljästä eri näkökulmasta, kutakin omassa alaluvussaan. Ensiksi tarkastellaan itse järjestelmää ja sen suorituskykyä. Pyrkimyksenä on arvioida järjestelmän laitteistovaatimuksia ja siten sopivaa palvelinkokoonpanoa. Toiseksi tarkastellaan esitoimenpiteitä, joita opiskelijalta edellytetään ennen verkko-oppimisympäristön hyödyntämistä. Alaluvussa arvioidaan mahdollisia tapoja vähentää näitä esitoimenpidevaatimuksia. Kolmanneksi tarkastellaan oppimisanalytiikkaa ja sen kehittämistä. Alaluvussa annetaan toteutusehdotus lisädatan keräämiseksi ja tehdään käytännön toteutus visualisointien kehittämiseksi. Neljänneksi pohditaan lyhyesti tekoälyn merkitystä ja sitä, miten sitä voisi hyödyntää verkko-oppimisympäristön yhteydessä.

5.1 Järjestelmän suorituskyvyn testaaminen

Tämän työn puitteissa ei testata virtuaalipalvelimen suorituskykyä. Valittu virtuaalipalvelimen kokoonpano perustuu siis valistuneeseen arvaukseen, eikä käytännön testeihin tai kokemukseen. Suorituskyvyn testaaminen tässä vaiheessa vaatisi joko useiden yhtäaikaisten käyttäjien asettaman kuorman simuloimista jollakin tapaa tai käytännön testaajien pyytämistä mukaan.

Tällaisen testiprosessin järjestäminen tämän opinnäytetyön puitteissa katsotaan liian vaativaksi ja aikaa vieväksi operaatioksi. Ensimmäinen käytännön testi tulee siis olemaan syksyllä 2023, kun aloittavat opiskelijat ottavat ympäristön käyttöön. Tavoite on, että järjestelmä kykenee palvelemaan noin sataa yhtäaikaista käyttäjää.

JupyterHubin dokumentaation mukaan mahdollisissa suorituskykyhaasteissa pullonkaulauksi koituu useimmiten järjestelmän keskusmuistin riittämättömyys. Mahdollinen rajallinen laskentateho ei ole yleensä yhtä suuri ongelma, koska rajallinen laskentateho johtaa tyypillisesti vain pidempään laskenta-aikaan – eli käyttäjän näkökulmasta pidempään odotteluun. Sen sijaan muistin loppuminen voi johtaa koko järjestelmän kaatumiseen ja tietojen menettämiseen. [60].

JupyterHubin dokumentaatiossa [60] annetaan esimerkkejä mahdollisesta tyypillisestä ’kevytkäyttäjästä’ ja ’raskaskäyttäjästä’ (resurssien käytön näkökulmasta). Esimerkissä kevytkäyttäjä käyttää keskusmuistia 0,5 gigatavua ja raskaskäyttäjä 2,0 gigatavua.

Koska verkko-oppimisympäristö on ainakin vielä tässä vaiheessa kohdistettu nimenomaan opintojaan aloittaville opiskelijoille, lienee todennäköistä, että yksittäisen opiskelijan järjestelmälle asettama muistikuorma on lähempänä kevytkäyttäjää kuin raskaskäyttäjää. Verkko-oppimisympäristössä on siis tarkoitus opetella ohjelmoinnin perusasioita, eikä ensisijaisena tarkoituksena ole tehdä raskasta kehitystyötä.

Mikäli edelleen jatketaan JupyterHubin dokumentaation [60] esimerkin mukaiseksi, voidaan ajatella, että itse järjestelmälle varataan 8 gigatavua keskusmuistia käyttöön. Tästä eteenpäin vaadittava keskusmuistin määrä tulee käyttäjien asettamasta kuormasta.

Taulukossa 2 hahmotellaan mahdollisia muistinkulutusskenaarioita eri yhtäaikaiskäyttäjämäärittäin. Taulukosta nähdään, että raskaskäyttäjämäärän lisääntyminen nostaa nopeasti vaatimusta muistikapasiteetille, vaikka yhtäaikaisten käyttäjien kokonaismäärä olisi alle satakin. Lisäksi taulukosta voidaan todeta, että keskusmuistin vaatimusväli vaihtelee esitetyissä skenaarioissa välillä 58–73.

Taulukko 2. Järjestelmän muistinkäytön arviointia käyttäjämäärittäin

Muistivaatimus (Gt) ↓		**Yhtäaikaiset käyttäjämäärät**					
Järjestelmä	8	1	1	1	1	1	1
Kevytkäyttäjä	0,5	100	95	90	80	70	60
Raskaskäyttäjä	2	0	5	10	10	10	10
Muistivaatimus yhteensä →		58	65.5	73	68	63	58

Koska palvelintarjoajien kapasiteettitarjonta vaihtelee yleensä portaittain, ei esimerkiksi keskusmuistin määrää voida tyypillisesti valita liukusääteisesti. Toisinaan portaiden väli voi olla suurikin. Kun katsotaan CSC:n Pouta-palvelun standardivalikoiman kokoonpanovaihtoehtoja, muuttuu muistitarjonta portaittain aina noin kaksinkertaiseksi edelliseen portaaseen verrattuna [61]. Eli esimerkiksi muistimäärästä 31 gigatavua hypätään määrään 62 gigatavua.

Maksimimäärä Pouta-palvelun standardivalikoimassa on juuri 62 gigatavua. Yllä hahmotellun taulukon perusteella näyttää, että tietyissä tilanteissa tämä määrä saattaa ylittyä. On kuitenkin mahdollista, että kevytkäyttäjälle asetettu 0,5 gigatavun muistinkulutus on sekin yläkanttiin. Suurin osa yksittäisen käyttäjän ajasta tuskin on erityisen muisti-intensiivistä toimintaa. Iso osa ajasta on todennäköisesti tehtävien lukemista ja koodin kirjoittamista, mikä ei lähtökohtaisesti nosta muistivaatimuksia korkealle. Lyhyen testaamisen perusteella käyttäjäkontti, joka on pelkästään päällä, mutta jossa ei suoriteta koodia, käyttää 62 gigatavun muistikapasiteetista Linuxin `top`-komennon mukaan 0,2 prosenttia kokonaiskapasiteetista, mikä on noin 124 megatavua.

Valistuneen arvauksen perusteella CSC:n Pouta-palvelun `standard.3xlarge`-paketti [61], jossa on 62 gigatavua keskusmustia, todennäköisesti riittää. Asiasta varmistuminen vaatii kuitenkin käytännön testaamista.

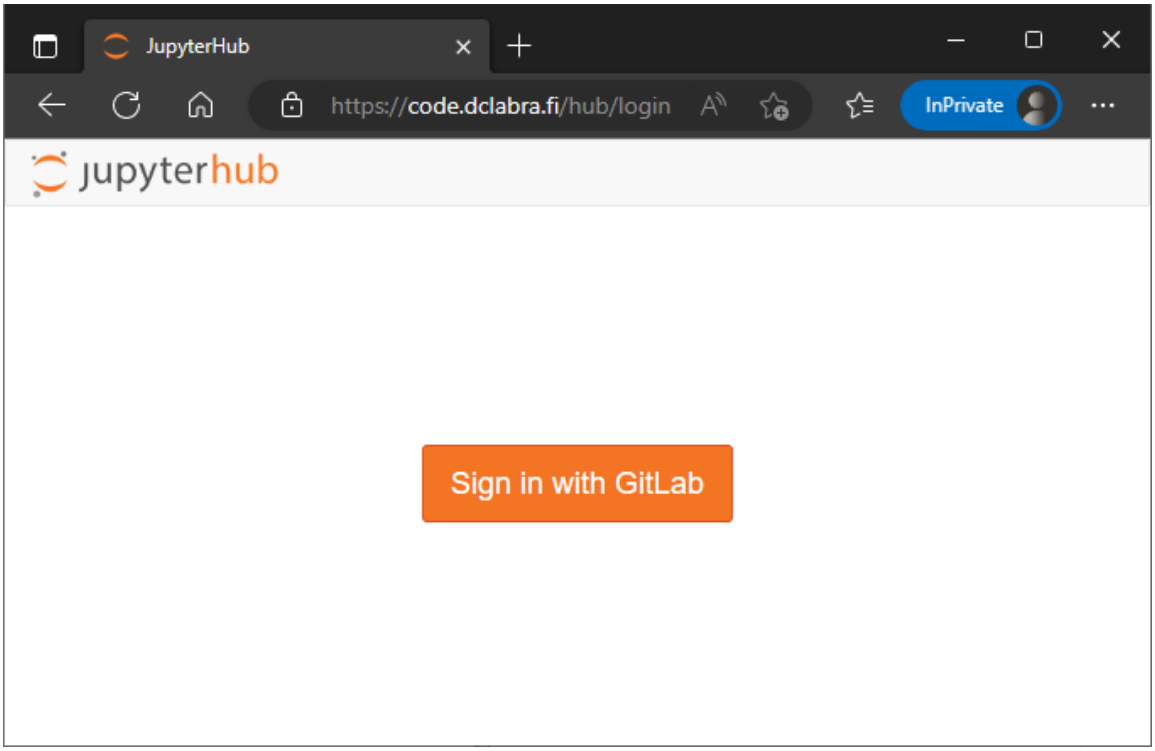
5.2 Opiskelijalta vaadittavien esitoimien vähentäminen

Päivitetty ympäristö vaatii opiskelijalta yhä seuraavat toimenpiteet ennen kuin hän voi suorittaa verkko-oppimisympäristössä kursseja: hänen tulee 1) tunnistautua GitLab-tunnuksilla, 2) kirjautua TMC-laajennukseen MOOC.fi-tunnuksilla ja 3) ladata suorittamansa kurssin tehtäväpohjat omaan verkko-oppimisympäristöönsä. Mikäli opiskelija on ollut kirjautumatta verkko-oppimisympäristöön riittävän pitkän ajan, tulee hänen kirjautua uudelleen myös TMC-laajennukseen ja poimia uudelleen jo aiemmin valitsemansa kurssit (vaikkakaan kursseja ei tarvitse ladata uudelleen). Kurssia suorittaessaan käyttäjällä tulee olla erikseen avoinna nettiselaimen välilehti, jossa kurssin tehtävien tehtävänannot ovat. Tämä siksi, että verkko-oppimisympäristöön ladataan pelkästään tehtävien suoritusrunko, mutta ei itse kysymyksiä.

Oppimisanalytiikan hyödyntämisen mahdollistamiseksi opiskelijan tulee lisäksi yhä luoda Wakapi-tunnus ja asettaa Wakapin generoima API-avain WakaTimen määritystiedostoon. Sen sijaan VSCoden käynnistäminen JupyterLabin kautta ja erillisen salasanan antaminen VSCodea käynnistettäessä jäävät päivitetyssä versiossa pois opiskelijan toimenpidevaatimuksista.

5.2.1 Tunnistautuminen

Tunnistautuminen on vaihe, joka on välttämätön ja jota ei ole missään nimessä mielekästä poistaa. Voidaan kuitenkin kysyä, onko valittu tunnistautumistapa paras mahdollinen. Nyt hyödynnetään Kajaanin ammattikorkeakoulun GitLab-instanssin tunnistautumista. Tätä voidaan pitää mielekkäänä tunnistautumistapana tekniikan opiskelijoiden kohdalla, koska tekniikan opiskelijat tulevat joka tapauksessa luomaan tunnukset myös Kajaanin ammattikorkeakoulun GitLabiin. Toisaalta samalla tullaan rajanneeksi verkko-oppimisympäristön mahdollista käyttäjäjoukkoa. Mikäli verkko-oppimisympäristössä on tarkoitus tarjota ohjelmointikursseja myös muille opiskelijoille kuin niille, joilla on Kajaanin ammattikorkeakoulun GitLab-tunnukset, tulee miettiä muita tunnistautumisvaihtoehtoja. Kuvassa 14 on nähtävillä kirjautumisnäkymä, kun käyttäjä on syöttänyt selaimen osoitekenttään verkko-oppimisympäristön osoitteen kirjautumattomana.



Kuva 14. Verkko-oppimisympäristön kirjautumisnäkymä

Yksi mielenkiintoinen tunnistautumisvaihtoehto voisi olla Haka-käyttäjätunnistusjärjestelmän hyödyntäminen. Haka on CSC:n kehittämä ja monien Suomen korkeakoulujen (kuten Kajaanin ammattikorkeakoulun) ja tutkimuslaitosten käyttämä tunnistusjärjestelmä [62]. Haka-tunnukset ovat siis tunnukset, jotka jokainen Kajaanin ammattikorkeakoulun – ja myös monen muun suomalaisen korkeakoulun – opiskelija saa. Käyttämällä Haka-tunnuksia putoaisi sitä myöten pois vaatimus erillisten GitLab-tunnusten omaamisesta. Toisaalta voidaan kysyä, kasvaisiko tällöin potentiaalinen käyttäjäjoukko jopa jo liian suureksi.

Lisäksi on hyvä huomioida, että Jupyter ei tue Haka-kirjautumista suoraan. Tämän työn puitteissa ei selvitetä, miten Jupyterin Haka-kirjautuminen toteutettaisiin, mutta on mahdollista, että se olisi hyvin vaativa operaatio.

Sen sijaan Jupyterilta löytyy suora tuki esimerkiksi LTI-tunnistautumiselle [63] LTI-kirjautuminen voi tulla kysymykseen, mikäli halutaan hyödyntää Moodle-tunnuksia. Myöskään tätä kirjautumisvaihtoa ei kuitenkaan testata tämän työn puitteissa.

5.2.2 Kirjautuminen TMC-laajennukseen

Ihannetapauksessa opiskelija ei tarvitsisi erillisiä tunnuksia TMC-laajennukseen kirjautumiseen. Tässä tapauksessa nojataan kuitenkin vahvasti Helsingin yliopiston avoimiin ratkaisuihin. Jotta opiskelija ei tarvitsisi MOOC.fi-tunnuksia, tulisi Kajaanin ammattikorkeakoulun luoda omat palvelimet TMC:n käytölle ja laatia laajennuksista omat versionsa. Tämä olisi todennäköisesti mittava urakka ja vaatisi lisäksi jatkuvaa ylläpitoa.

Sen sijaan mahdollisesti helpommin poistettavissa oleva lisävaiva voisi olla tilanne, jossa opiskelijan tulee kirjautua TMC-laajennukseen uudelleen ja poimia suorittamansa kurssit uudelleen, mikäli hän ei ole käyttänyt verkko-oppimisympäristöä riittävän pitkään aikaan. Tämä tarkoittaisi sitä, että käyttäjien JupyterLab-kontti tulisi määritellä laajemmin säilyttämään saavutettu tilansa. Tällä hetkellä säilytetään vain opiskelijan luoma data, mutta datakansion ulkopuolelle jäävät muutokset palaavat alkuperäiseen tilaansa. Se, että toimitaan näin, takaa sen, että kenenkään ei ole mahdollista saada omaa instanssiaan pysyvästi niin sanotusti sekaisin. Tämän myötä opiskelijan on turvallisempaa kokeilla asioita ilman pelkoa siitä, että mikään menisi ”rikki”. Mikäli kontti määritellään säilyttämään oma tilansa laajemmin, tarkoittaa se sitä, että asioiden mennessä sekaisin voidaan joutua turvautumaan järjestelmän ylläpitäjän apuun.

5.2.3 Tehtävänantojen tarkasteleminen

Yhtenä, ehkä hivenen harmillisena puolena verkko-oppimisympäristössä on se, että suoritettavien kurssien varsinaisia tehtävänantoja ei voi tarkastella suoraan ympäristössä. Ympäristössä on ainoastaan tehtävien suoritusrunko. Opiskelijalla tulee olla nettiselaimessa auki erillinen välilehti, jossa tehtävänannot näkyvät. Ideaalitilanteessa riittäisi, että opiskelijan tarvitsee avata ainoastaan verkko-oppimisympäristö.

Lisäksi se aika, minkä opiskelija käyttää tehtävänantojen ja niiden yhteydessä olevan teorian lukemiseen, ei kerry Wakapin keräämiin ajankäyttötilastoihin. Dataa ei kerry, koska Wakapi ei tunnista verkko-oppimisympäristön ulkopuolella tapahtuvaa toimintaa.

Ohjelmointikurssien tehtäväsivujen lähdekoodi on kuitenkin myös avoimesti saatavilla GitHubissa [kuten esimerkiksi 63]. Tehtävänannot ja niihin liittyvä teoria löytyy lähdekoodista markdown-muodossa, jota VSCode osaa esittää. Nämä markdown-tiedostot ja niihin liittyvät kuvat tulisi ladata opiskelijan konttiin ja mielellään tietysti ne tulisi asettaa myös helposti edettävään muotoon.

Tähän täytyisi kehittää erillinen ratkaisu, se täytyisi tehdä jokaiselle kurssille erikseen ja sitä täytyisi myös ylläpitää mahdolliset kurssien tehtäviin liittyvät päivitykset huomioiden.

5.2.4 Oppimisanalytiikan mahdollistaminen

Oppimisanalytiikan mahdollistaminen edellyttää Wakapi-tunnusten luomista ja Wakapin generoiman API-avaimen lisäämistä WakaTimen määritystiedostoon. Wakapissa ei tällä hetkellä ole yksinkertaista keinoa automatisoida käyttäjätunnusten ja API-avaimen luomista, vaan tämä tapahtuu web-käyttöliittymän kautta. Sinnikäs ohjelmistokehittäjä löytää varmasti ratkaisut tällaisessakin tilanteessa. Ratkaisu saattaisi olla toteutettavissa esimerkiksi hyödyntämällä Pythonin Selenium-kirjastoa, joka mahdollistaa selainten automatisoinnin [65]. Tunnusten automaattinen luominen voitaisiin mahdollisesti toteuttaa tällaisen Seleniumiin pohjautuvan ratkaisun avulla. Tällaisen ratkaisun kehittäminen kuitenkin lisäisi järjestelmän kompleksisuutta ja lisäisi todennäköisesti myös järjestelmän kehitykseen vaadittavaa aikaa päivitystilanteissa.

Toinen, helpompikin ratkaisu saattaa olla olemassa. Wakapille on nimittäin olemassa WakaTimen rajapintaa hyödyntävä toinen vaihtoehto: Hakatime [35]. Tämän verkko-oppimisympäristön tapauksessa päädyttiin valitsemaan juuri Wakapi, koska sen kehitystyö *vaikuttaa* aktiivisemmalta ja se *vaikuttaa* suositummalta muiden käyttäjien keskuudessa. Tämän myötä syntyy vaikutelma varmatoimisemmasta vaihtoehdosta. Tähän vaikutelmaan on päädytty vertaamalla Wakapin [33] ja Hakatimen [35] GitHub-tietovarastoihin (*repository*) tehtyjen tallennettujen muutosten määrää (*commit*), tietovarastojen saamien tähtien (*stars*) määrää, tietovarastosta muiden käyttäjien muodostamien haarojen (*forks*) määrää ja viimeisimmän julkaisun (*release*) päivämäärää sekä yleisesti tietovaraston tiedostojen viimeisimpiä muokkausajankohtia.

Noilla yllä mainituilla meriiteillä Wakapi päihittää Hakatimen selvästi elettäessä kevättä 2023. Hakatimessa on kuitenkin sen tietovaraston mukaan [35] muutamia komentorivivaihtoehtoja, joita Wakapissa ei ole. Hakatime tarjoaa mahdollisuuden luoda uusia käyttäjiä ja uusia tunnistautumisavaimia suoraan komentorivin kautta. Tämän työn yhteydessä tätä vaihtoehtoa ei testata, vaan päädytään Wakapin käyttöön edellä mainituilla perusteilla. Kuitenkin mahdollisuus luoda käyttäjiä komentoriviltä vaikuttaa siinä määrin houkuttelevalta, että ehkäpä sitä on suositeltavaa testata seuraavan päivitysversion yhteydessä.

Samalla on kuitenkin hyvä nostaa esiin vielä yksi näkökulma liittyen käyttäjätunnusten automaattiseen luomiseen verrattuna siihen, että opiskelija luo käyttäjätunnuksen itse: Kun opiskelija luo

käyttäjätunnuksen itse, kontrolli säilyy selkeämmin hänellä itsellään. Tällöin hän on varmuudella tietoinen omasta käyttäjätunnuksestaan ja tietää myös, miten hän pääsee katsomaan kertyneen datan raportointinäkymiä Wakapin web-käyttöliittymässä. Mikäli käyttäjätunnus luodaan automaattisesti taustalla, opiskelijan kontrolli saattaa hämärtyä. Tällöin opiskelijalle täytyy erikseen kertoa, mikä hänen käyttäjätunnuksensa on ja miten sitä pääsee hyödyntämään. Käyttäjätunnuksen automaattinen luominen ei siis poista tarvetta erilliselle Wakapi/Hakatime -käyttäjätunnukselle sinänsä – opiskelijalta asia saattaa vain mennä niin sanotusti ohi, mikäli tunnus luodaan automaattisesti.

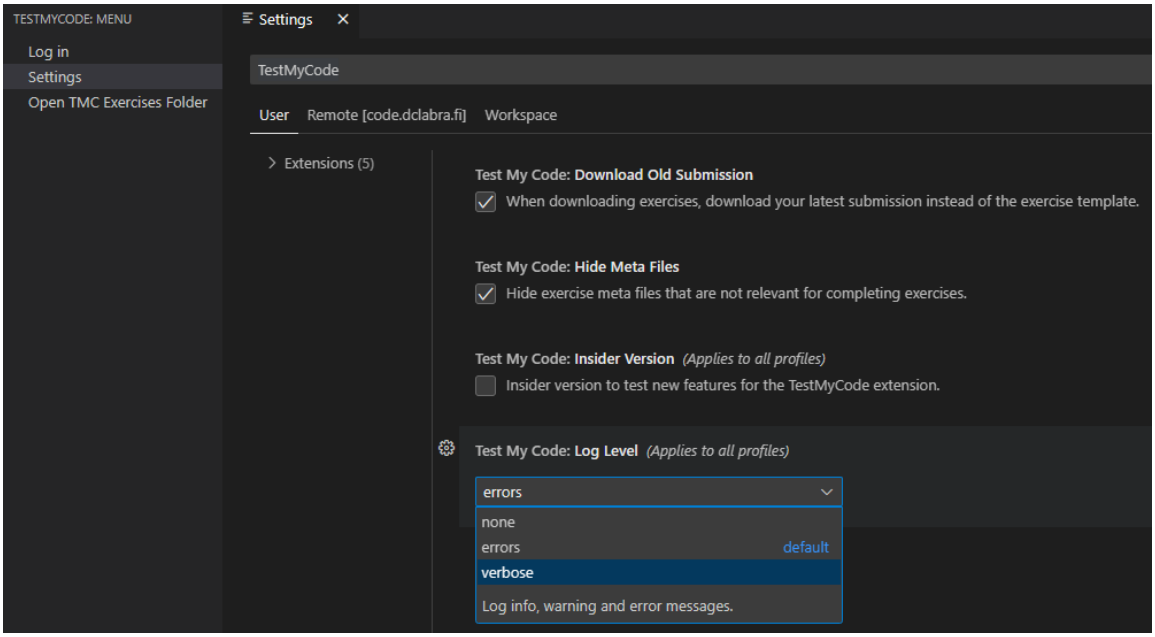
5.3 Oppimisanalytiikan kehittäminen

5.3.1 Lisädatan kerääminen

Dataan, jota kerätään WakaTimen API-rajapintaa hyödyntämällä, liittyy oppimisanalytiikan näkökulmasta ainakin yksi harmillinen puute: tietoa kertyy suorittamiseen käytetystä ajasta, mutta ei esimerkiksi siitä, kuinka monta kertaa jonkin tietyn tehtävän suorittamista on yritetty ennen kuin se on mennyt läpi testiohjelmasta.

Suoritusyritysten määrä on kuitenkin kerättävissä. Yksi tapa lähestyä asiaa voisi olla muokkaamalla joko TMC-laajennuksen lähdekoodia, itse suoritettavan kurssin tehtävien lähdekoodia tai mahdollisesti molempia. Tämä edellyttäisi kuitenkin jälleen, paitsi ensin perehtymistä Helsingin yliopiston ilmaiseksi tarjoamien ratkaisujen lähdekoodiin, myös merkittävää ylläpidollista lisävaivaa, kun valmiiden ratkaisujen rinnalla olisi ylläpidettäviä omia versioita näistä ratkaisuista.

Toinen tapa lähestyä asiaa on tarkastella sitä kertyvien lokitiedostojen kautta. Mikäli TMC-laajennuksen lokien keräämistason nostaa korkeimmalle eli `verbose`-tasolle, myös suoritusyritykset kirjautuvat lokiin. Kuvassa 15 nähdään, miltä tämä `verbose`-tason valinta näyttää TMC-laajennuksen asetusvalikossa.



Kuva 15. TMC-laajennuksen lokien keräystason muuttaminen

Ainakin tässä työssä testatun kurssin suoritusyritykset kirjautuvat lokiin aina samalla syntaksilla ja oletettavasti näin on kaikkien kurssien kohdalla niin kauan kuin TMC-laajennuksen lähdekoodia ei muuteta. Suoritusyritykset saa eriteltyä lokitiedostoista näkyviin esimerkiksi yksinkertaisella komentoriviskriptillä, josta nähdään esimerkki kuvassa 16.

```
(base) jovyan@533313d13a2f:~$ for i in $(find -iname *.log | grep TestMyCode); do (cat $i | grep "Tests finished for"); done
[2023-03-31 09:30:30:771] [INFO] Tests finished for osa01-01_hymio
[2023-03-31 09:32:38:744] [INFO] Tests finished for osa01-01_hymio
[2023-03-31 09:33:08:587] [INFO] Tests finished for osa01-01_hymio
[2023-03-31 09:33:37:080] [INFO] Tests finished for osa01-01_hymio
[2023-03-31 09:49:13:703] [INFO] Tests finished for osa01-02_ukko_nooa
[2023-03-31 09:49:22:014] [INFO] Tests finished for osa01-02_ukko_nooa
[2023-03-31 10:03:38:373] [INFO] Tests finished for osa01-02_ukko_nooa
```

Kuva 16. Tehtävien suoritusyrityskertojen tarkasteleminen lokitiedostoista

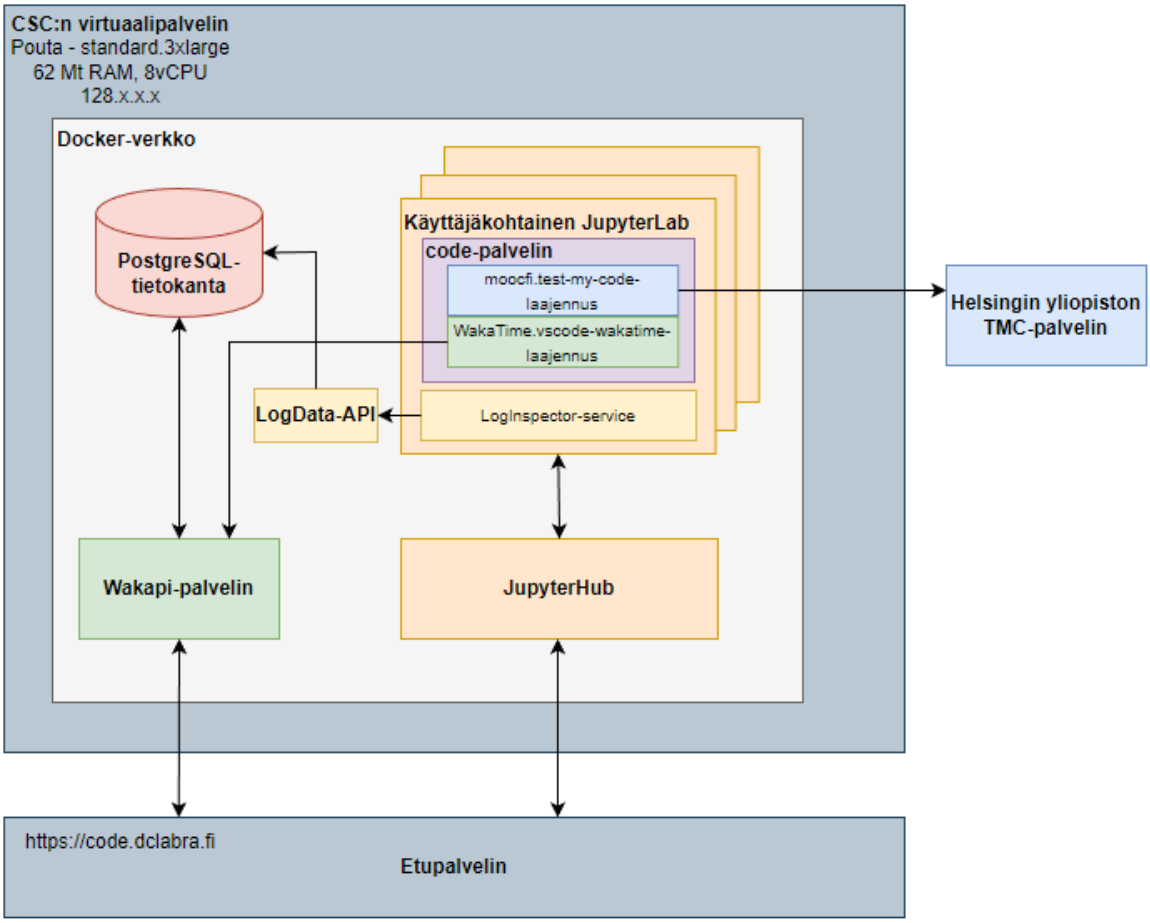
Kuten kuvasta voidaan havaita, lokiin tallentuu suoritusyrityksen tarkka kellonaika ja se, mitä tehtävää on yritetty. Itse suoritettavan kurssin nimi ei tallennu lokiin, vaan se tulee erikseen päätellä tehtävien nimen perusteella. Lisäksi tällaisesta yksittäisesti lokitiedoston rivistä ei käy ilmi, onko suoritusyritys mennyt läpi testiohjelmasta. Tämä tieto kuitenkin on myös nähtävillä lokitiedostoista, mutta ei samalla rivillä. Tieto tulee päätellä siitä, ovatko kaikki tehtävälle tehdyt testit menneet läpi vai onko niistä yksikin epäonnistunut. Testit ovat tehtäväkohtaisia. Mikäli siis tieto tehtävän läpimenosta halutaan sisällyttää mukaan, on se mahdollista tehdä, mutta se lisää käytettävän skriptin monimutkaisuutta.

Kun tiedetään, että suoritusyrityskerrat ovat haettavissa lokitiedostoista, voidaan rakentaa ratkaisu yrityskertojen tallentamiseksi tietokantaan. Tällöin ei yksi kertaluontoisesti ajettava skripti

riitä, vaan tarvitaan menetelmä, joka käy lokitiedostot taustalla läpi tietyin väliajoin ja lähettää tiedot eteenpäin, jotta ne voidaan tallentaa tietokantaan. Tämä tarkoittaa muutoksia ympäristön arkkitehtuuriin.

Yksi tapa lähestyä asiaa on lisätä arkkitehtuuriin kontti, jossa on REST API -rajapintaa noudattava palvelin. Palvelin kuuntelee käyttäjien JupyterLab-konteista lähetettyjä http-pyyntöjä, jotka sisältävät lokitietoa. Tämän tiedon palvelin parsii haluttuun muotoon, lisää siihen tiedon, mistä kontistä http-pyyntö tuli (eli kuka käyttäjä sen lähetti) ja tallentaa tiedot tietokantaan. Tietokantaan tallentamista varten arkkitehtuuriin tarvitaan joko toinen tietokantakontti Wakapille luodun tietokantakontin lisäksi tai lokeista kerätty tieto kerätään samaan tietokantaan, mutta erilliseen tauluun.

Käyttäjäkonttia tulee muokata niin, että siihen luodaan Linux-palvelu (*service*), joka käynnistyy aina kontin käynnistämisen yhteydessä. Tämä palvelu suorittaa skriptiä, joka määrätyin välein lukee lokitiedot ja lähettää ne http-pyyntönä REST API -rajapintaa noudattavalle lokitietojen kuuntelijapalvelimelle. Kuvassa 17 esitellään vaihtoehtoinen arkkitehtuuri, johon on lisätty suoritusyrityskertojen kerääminen lokitiedoista.



Kuva 17. Vaihtoehtoinen ympäristön arkkitehtuuri, joka kerää tietoa myös lokitiedostoista

Yllä esitellyssä muodossaan verkko-oppimisympäristö ei kuitenkaan mahdollista suoritusyrityskertojen visualisointeja. Data kerätään WakaTimen rajapinnasta erillisenä, joten Wakapi ei tunnista sitä, eikä sitä voi tarkastella Wakapin visualisointinäkymien kautta. Jotta suoritusyritysdataa voidaan tarkastella, täytyy sitä varten luoda erillinen visualisointiratkaisu.

Lisäksi on syytä tuoda esiin, että yllä esitellyn menetelmän mukaan toteutettuna lokeja tarkasteleva skripti on lähtökohtaisesti aina päällä. Skriptille ei kuitenkaan päädy suoritusyrityskertatietoja lähetettäväksi, ellei TMC-laajennuksen lokitietojen keräämistasoa koroteta `verbose`-tasolle, kuten kuvassa 15 esitetään. Opiskelijan kannalta asetelma ei välttämättä olisi täysin selkeä. Mikäli opiskelija haluaisi korottaa lokitietojen keräystasoa silkasta mielenkiinnosta, ei hänellä olisi samalla mahdollisuutta olla osallistumatta datan keräämiseen. Tämä ei tietenkään ole toivottava asetelma.

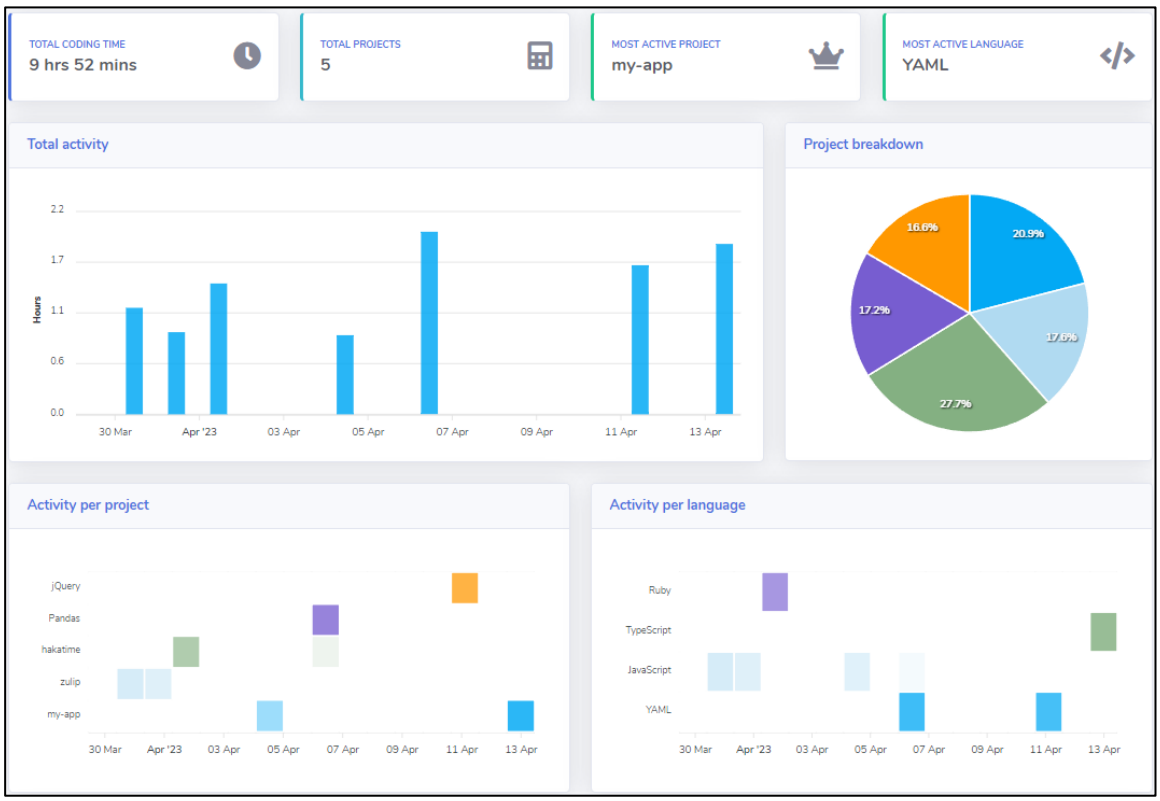
5.3.2 Visualisointien kehittäminen – valmiit ratkaisut

Oppimisdatan visualisoinnissa nojataan lähtökohtaisesti Wakapin web-käyttöliittymän tarjoamiin visualisointeihin. Tämä tarkoittaa sitä, että on tyydyttävä niihin visualiratkaisuihin, joita web-käyttöliittymä tarjoaa – omien näkymien lisääminen Wakapin käyttöliittymään ei ole mahdollista (muokkaamatta Wakapin lähdekoodia). Lisäksi tämä tarkoittaa sitä, että ainoastaan WakaTimen rajapinnan kautta kerätty data on tarkasteltavissa. Esimerkiksi suoritusyrityskerroista kertova data, jota kerättäisiin edellisellä sivulla esitellyn vaihtoehtoisen arkkitehtuurikuvan (kuva 17) mukaisesti, ei olisi tarkasteltavissa Wakapin käyttöliittymän kautta.

Mikäli kaikki kerättävä data, kerättiinpä se sitten mitä väylää pitkin tahansa, halutaan visualisoida samassa näkymässä, täytyy rakentaa oma visualisointiratkaisu. Mikäli puolestaan halutaan ainoastaan vaihtoehto WakaTime-rajapinnan kautta kerätyn datan visualisoinneille, on tähän olemassa ainakin kaksi erilaista valmista ratkaisua. Yksi lähestymistapa olisi korvata Wakapi sen vaihtoehdolla, Hakatimellä [35]. Toinen lähestymistapa olisi käyttää WakaTime Exporteria [66], jonka avulla WakaTime-tilastot voidaan ohjata Prometheus-monitorointityökaluun [67] ja sitä kautta Grafanaan [68].

Hakatime. Hakatimeä Wakapin vaihtoehtona tarkastellaan myös luvussa *5.2.4 Oppimisanalytiikan mahdollistaminen*. Siinä yhteydessä Hakatimeä tarkastellaan käyttäjätunnusten luomisen näkökulmasta Wakapiin verrattuna. Hakatime kuitenkin eroaa Wakapista myös visualisointinäkymillään.

Hakatimen demon, johon löytyy linkki ja tarvittavat salasanat Hakatimen GitHub-tietovaraston kautta [35], avulla voidaan tarkastella Hakatimen tarjoamia visualisointiratkaisuja. Tästä on nähtävillä osittainen esimerkki kuvassa 18. Erilaiset visualisointiratkaisut ovat totta kai myös makuasioita, mutta todettakoon, että Hakatimen visualisointinäkymässä on esitysmuotoja, joita Wakapi ei tarjoa.



Kuva 18. Osittainen kuvankaappaus Hakatimen visualisointinäkymän demosta

WakaTime Exporter. Toinen valmis ratkaisu WakaTime-tilastojen visualisointiin on WakaTime Exporter [66]. WakaTime Exporterin etu on se, että se voidaan ottaa käyttöön Wakapin tai Hakatimen rinnalle, eikä se siis siten ole korvaava toimenpide. Kysymyksiä herättävä puoli WakaTime Exporterissa on puolestaan se, että GitHub-tietovarastonsa mukaan [66] se on yhä alpha-versiossaan ja siten altis rikkoville muutoksille. Samaan aikaan viimeisin versiojulkaisu on lokakuun alusta vuodelta 2020, eli tämän opinnäytetyön kirjoittamisen hetkellä kahden ja puolen vuoden takaa.

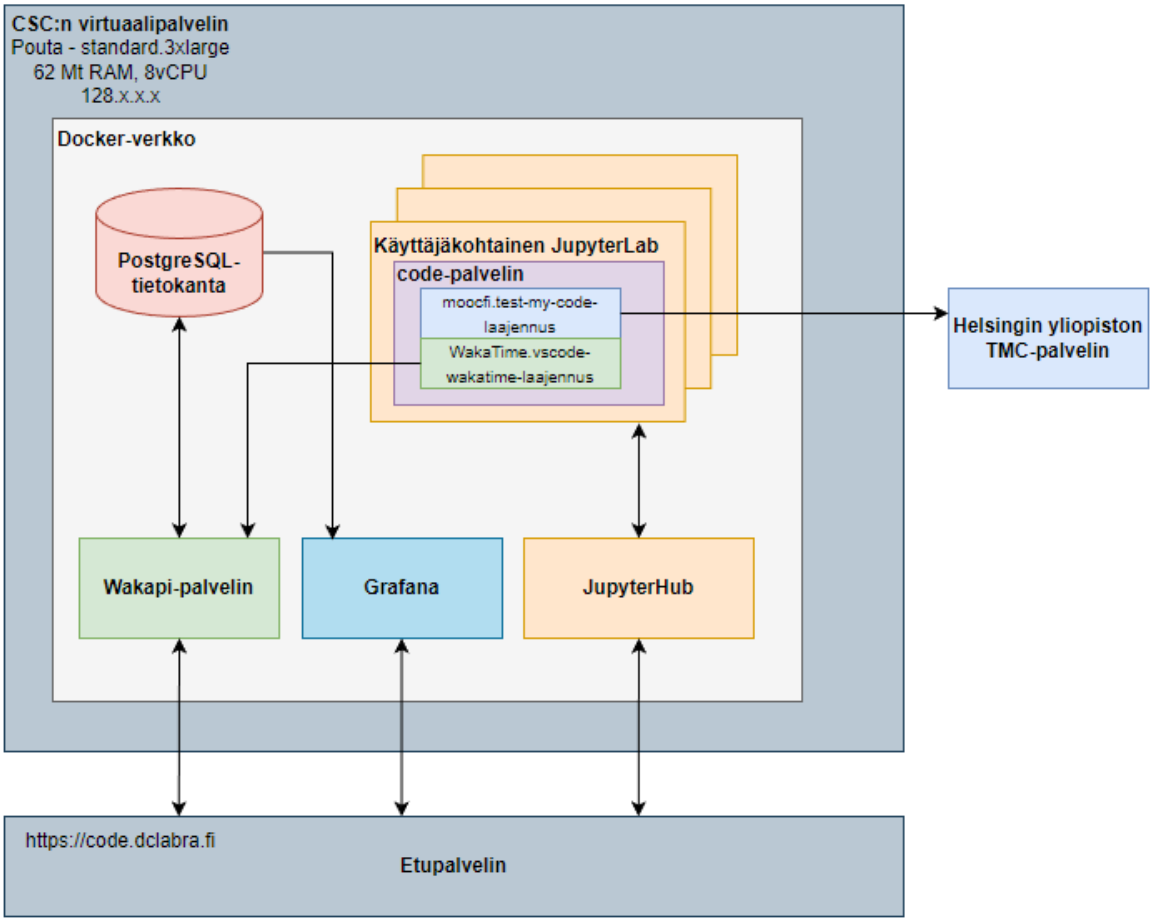
Mikäli WakaTime Exporter halutaan ottaa käyttöön, edellyttää myös se muutoksia ympäristön arkkitehtuuriin. Itse WakaTime Exporter tulee asentaa käyttäjäkohtaisen JupyterLab-kontin sisään. Tämän lisäksi tarvitaan kontit sekä Grafanalle että Prometheukselle. Tässä tapauksessa data ohjataan Prometheukseen, josta Grafana puolestaan hakee sen visualisointinäkymiä varten.

On kuitenkin huomattava, että niin Wakapi kuin Hakatimekin myös tallentavat datan, mutta Prometheuksen sijaan PostgreSQL-tietokantaan. Toisin sanoen käytettäessä WakaTime Exporteria data tulee tallennetuksi kahteen kertaan. Lisäksi WakaTime Exporter tarvitsee tiedoksi WakaTimen API-avaimen, eli tässä tapauksessa Wakapin kautta generoidun avaimen. Tämä tarkoittaa yhtä lisätoimea opiskelijan näkökulmasta. Mikäli halutaan luoda erillinen visualisointinäkymä Grafanan kautta, ehkäpä onkin mielekkäämpää luoda sellainen itse. Tällöin voidaan lisäksi poimia

data muista lähteistä kuin tietokannasta, johon tallennetaan nimenomaan WakaTimen rajapinnan kautta tuleva data.

5.3.3 Visualisointien kehittäminen – räätälöity ratkaisu

Koska tarjolla olevat valmiit ratkaisut eivät ole tähän ympäristöön sopivia, luodaan räätälöity ratkaisu. Ratkaisu luodaan Grafanan avulla. Visualisointi-ideoita poimitaan edellisessä kappaleessa mainituista Hakatimesta ja WakaTime Exporterista. Kuvassa 19 nähdään, kuinka muutos vaikuttaa ympäristön arkkitehtuuriin. Grafana luodaan omaan konttiinsa. Sen sijaan datalähteenä käytetään ympäristön arkkitehtuuriin entuudestaan kuuluvaa PostgreSQL-tietokantaa. Grafanaan luodaan koontinäkymä (*dashboard*), johon lisätään halutun kaltaisia visualisointeja sisältäviä paneeleja. Tätä koontinäkymää päästään katselemaan Grafanan web-käyttöliittymän kautta.

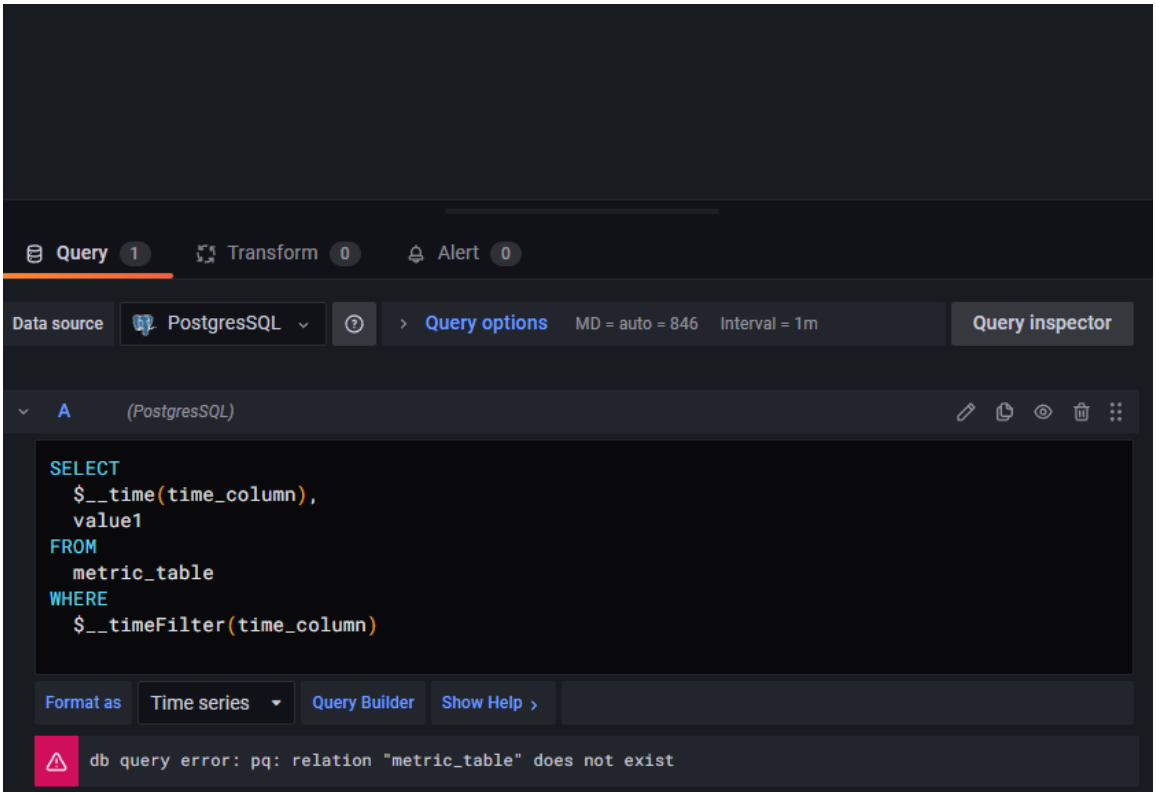


Kuva 19. Arkkitehtuuri, jossa on mukana Grafana visualisointien lisäämiseksi

PostgreSQL-tietokanta määritellään Grafanaan datalähteeksi Grafana-kontin luontivaiheessa, jolloin Grafana osaa yhdistää tietokantaan automaattisesti ympäristön pystyttämisen jälkeen. Datalähteen määritykset kirjoitetaan YAML-tiedostoon. Tämän määritystiedoston tulee löytyä Grafana-kontin `/etc/grafana/provisioning/datasources` -kansiosta. Tiedosto voidaan 1) joko kopioida Grafana-kontin mainittuun kansioon tai 2) säilyttää jossakin virtuaalipalvelimen kansiossa, joka liitetään (*mount*) osaksi kontin tiedostojärjestelmää (*volume*). Esimerkki datalähteen määrityksistä YAML-tiedostoon on tarkasteltavissa Grafanan dokumentaatiosta [69].

Datalähteen lisäksi tarvitaan itse koontinäkymä. Tämä on helpointa tehdä luomalla ensin Grafanalle kehityskontti ja rakentaa näkymä Grafanan web-käyttöliittymän kautta. Valmis näkymä viedään JSON-tiedostoksi, joka otetaan yhdessä näkymään liittyvän YAML-määrittelytiedoston kanssa osaksi arkkitehtuurin Docker-määrittelytiedostoja. Tällöin näkymä ilmestyy automaattisesti Grafanaan tarkasteltavasti arkkitehtuurin pystyttämisen yhteydessä – aivan kuten datalähdekin.

Kuvassa 20 esitellään tilanne, jossa Grafanaan luodaan uutta visualisoinnin sisältävää paneelia. Datalähteeksi on asetettu PostgreSQL, mutta varsinaista kyselyä ei ole vielä kirjoitettu – eli paneeliin ei vielä ilmesty visualisointia.



Kuva 20. Paneelin luominen Grafanassa

Data haetaan Wakapin PostgreSQL-tietokantaan tallentamista tiedoista. Pohjana on `heartbeats`-taulu, joka esitellään luvussa *3 Oppimisanalytiikka* ja jonka tietueesta on nähtävillä esimerkki mainitussa luvussa olevassa kuvassa 10. Keskeiseksi kysymykseksi muodostuu, miten ohjelmointiin käytetty aika lasketaan taulun tietueiden perusteella. Yksittäinen tietuehan sisältää vain yhden aikaleiman – ei ajallista välimatkaa.

Ajan kertyminen lasketaan taulun vierekkäisten tietueiden ajallisten erotusten perusteella. Mikäli aikaleimat ovat määritellyn etäisyyden päässä toisistaan, niiden muodostama ajallinen välimatka kumuloituu verkko-oppimisympäristössä käytetyksi ajaksi. Mikäli taas tietueiden välissä on määriteltyä etäisyyttä pidempi väli, oletetaan, että välissä opiskelija on tehnyt jotakin muuta.

Koska WakaTime tallentaa uusia tietueita suhteellisen tiheästi käyttäjän ollessa aktiivinen, määritellään tietueiden väliseksi maksimietäisyydeksi kaksi minuuttia. Mikäli etäisyys ylittyy, ei tietueiden aikaväliä lasketa yhteen. Kaksi minuuttia on sama väli kuin mitä on käytetty Wakapin GitHub-tietovarastossa olevassa esimerkkiskriptissä [70].

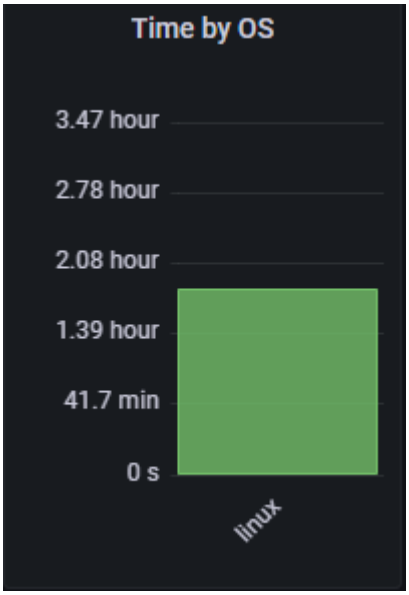
Alla, ohjelmakoodissa 1, esitellään SQL-kysely, jota laadittaessa pohjana on käytetty mainittua esimerkkiskriptiä. Kyselyn avulla tarkastellaan opiskelijan ajankäytön jakautumista käyttöjärjestelmäkohtaisesti. Tämän verkko-oppimisympäristön tapauksessa se tarkoittaa samalla myös kokonaisaikaa, koska käyttöjärjestelmä on aina Linux.

```
SELECT operating_system, SUM(GREATEST(1, diff)) as "metric"
  FROM (
    SELECT time as "time", project, language, editor,
      operating_system, machine, branch,
      EXTRACT(EPOCH FROM (LEAST((time - LAG(time) over w),
        '00:02:00'))) AS "diff"
    FROM heartbeats
    WHERE user_id = '${user}'
    WINDOW w AS (ORDER BY time)
    ) AS "s2"
WHERE
  diff IS NOT NULL
  AND $__timeFilter(time)
GROUP BY operating_system
ORDER BY metric DESC;
```

Ohjelmakoodi 1. Kokonaisajankäytön hakeminen käyttöjärjestelmittäin ryhmiteltynä

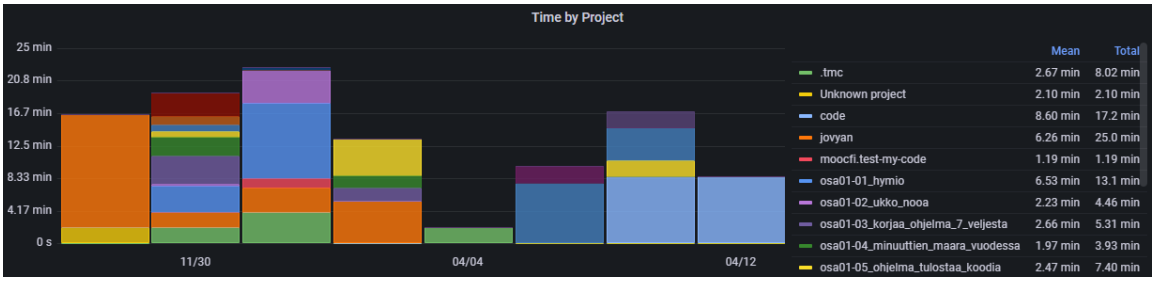
Ohjelmakoodissa näkyvät $-merkit viittavat Grafanassa käytettäviin muuttujiin ja metodeihin. `$__timeFilter` on Grafanan sisäinen metodi aikavälin dynaamista määrittelemistä varten. Ohjelmakoodissa esiintyvä `${user}`-muuttuja puolestaan on näkymäkohtainen, itse luotu muuttuja, joka mahdollistaa käyttäjän valitsemisen pudotusvalikon kautta.

Valitsemalla Grafanan paneelin asetuksista toivotut määritykset saadaan aikaiseksi yksinkertainen visualisointi ajan jakautumisesta käyttöjärjestelmittäin. Tässä tapauksessa kaikki aika kohdistuu Linux-käyttöjärjestelmälle, kuten kuvan 21 esimerkistä nähdään.

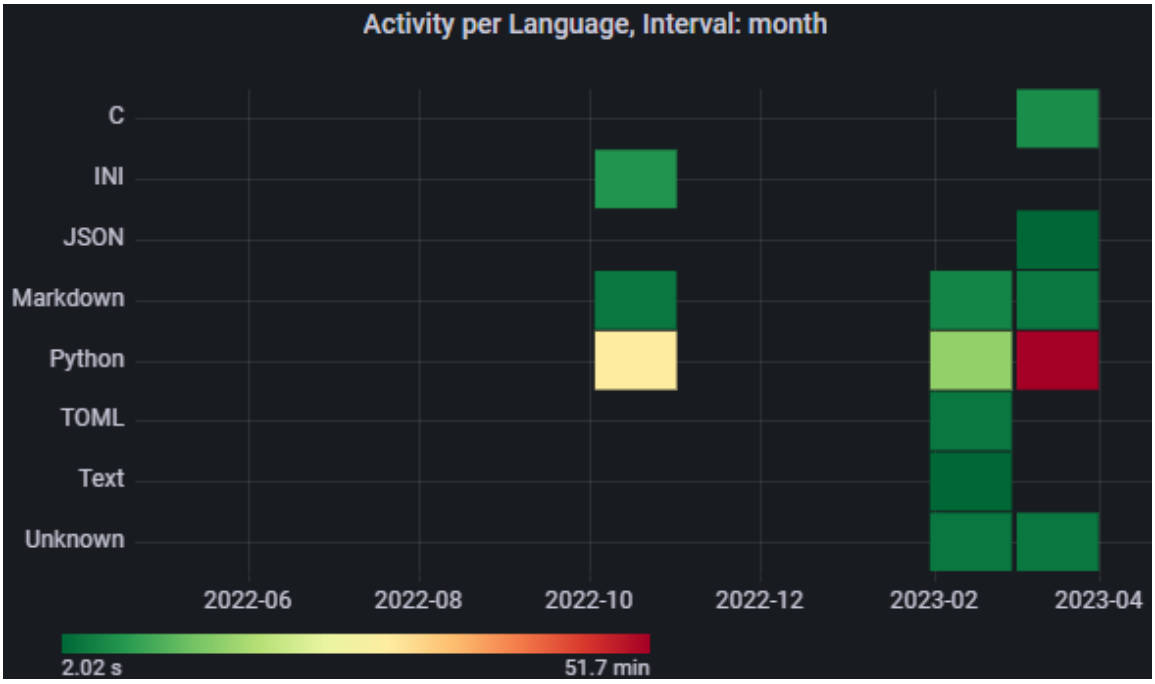


Kuva 21. Grafana-paneeli – ajankäytön jakautuminen käyttöjärjestelmittäin

Kun ideasta saadaan kiinni, voidaan luoda monimutkaisempia kyselyitä ja siten monimutkaisempia visualisointeja. Kuvassa 22 esitellään paneeli, johon on visualisoitu ajankäytön jakautuminen projekteittain. Kuvassa 23 puolestaan nähdään ajankäytön jakautuminen ohjelmointikielittäin.

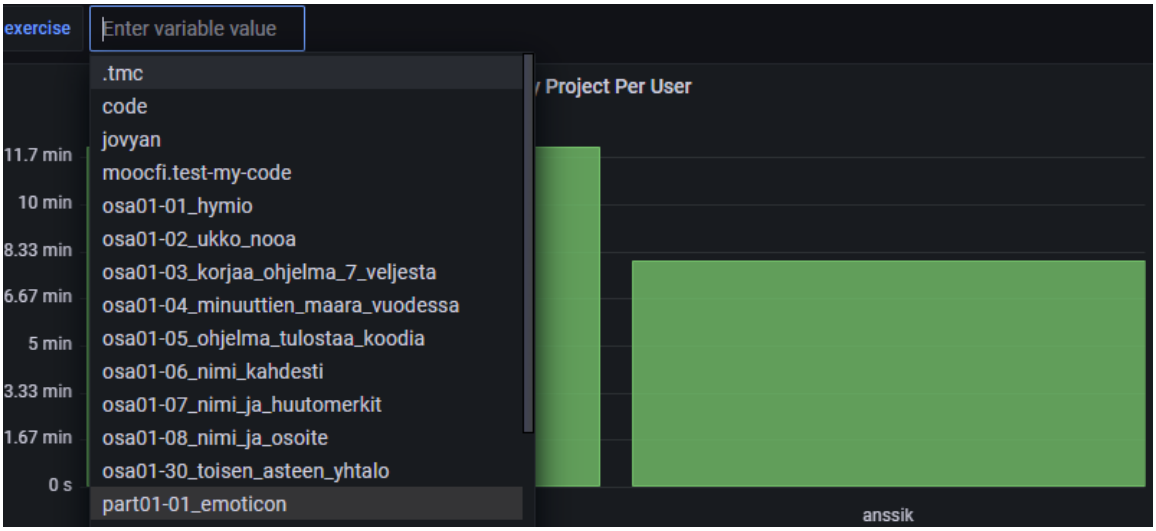


Kuva 22. Grafana-paneeli – ajankäytön jakautuminen projekteittain



Kuva 23. Grafana-paneeli – ajankäytön jakautuminen ohjelmointikielittäin

Erilaisia paneeleita voidaan luoda tarpeen mukaan ja yhdistää nämä koontinäkymään. Esitellyt paneelit ovat koosteita yhden käyttäjän ajankäytöstä. Ne ovat kuvankaappauksia laajemmasta koontinäytöstä. Tämä koontinäyttö on kokonaisuudessaan liitteessä 1. On mahdollista luoda myös näkymiä, joissa opiskelijoiden ajankäyttöä verrataan tehtävittäin. Tällainen näkymä voi olla opettajan kannalta arvokas: siitä välittyy yhdellä vilkaisulla, minkä verran aikaa tehtävät ovat opiskelijoilta vaatineet. Tämän työn puitteissa vertailunäkymien kehitys jää vielä vajaaksi, eikä kokonaista koontinäkymää luoda. Yksittäisestä paneelista nähdään kuitenkin esimerkki kuvassa 24.



Kuva 24. Grafana-paneeli – oppilaiden ajankäytön jakautuminen tehtävittäin

5.4 Tekoälyn yhdistäminen VSCodeen

Tekoälyn kehitys on tänä päivänä nopeaa ja esimerkiksi paljon keskustelua herättänyt ChatGPT [71] julkaistiin muutama päivä sen jälkeen, kun tämän työn suunnitelma oli lyöty lukkoon ja tiedonhaku aloitettu. Tekoälyn kehittyessä herää väistämättä kysymys myös siitä, miten tekoäly tulisi huomioida omassa työssään vai tulisiko. ChatGPT näyttää esimerkiksi osaavan ratkaista tässä työssä testatun ohjelmointikurssin [27] tehtäviä.

Tämän työn puitteissa ei ole tarkoitus pohtia tekoälyn vaikutuksia laajemmin eikä myöskään etsiä mahdollista sopivaa suhtautumista tekoälyyn. Sivutaan kuitenkin ajatusta siitä, miten tekoälyn voisi mahdollisesti tuoda osaksi tämän työn verkko-oppimisympäristöä.

Yksinkertaisimmillaan se voi tarkoittaa laajennuksen lisäämistä VSCodeen, minkä avulla esimerkiksi juuri ChatGPT:n kaltainen apuri on tarjolla suoraan VSCoden käyttöliittymän kautta. Ehkäpä opiskelijalle olisi mahdollista luoda ikään kuin digitaalinen apuri tai kollega. Ajatusta voidaan laajentaa luomalla niin sanottu puhuva pää, joka on tekoälyn generoima video ihmisestä. Kenties osaksi VSCode-ohjelmointiympäristöä voitaisiin integroida digitaalinen assistentti, joka lausuu ääneen ChatGPT:n antamat ehdotukset.

Todettakoon, että tekniikka ei kuitenkaan ole vielä sillä tasolla, että tällaista tekoälyassistentin videopuhetta voitaisiin generoida reaaliaikaisesti tai että tällaiset ratkaisut olisivat ainakaan suoraan helposti integroitavissa VSCodeen. Itse videopuhetta generoivia ratkaisuja sinänsä, mikäli unohdetaan reaaliaikaisuus ja integrointi, kuitenkin on olemassa [kuten esimerkiksi 72 ja 73]. ChatGPT:tä hyödyntäviä VSCoden laajennuksia puolestaan löytyy VSCoden kaupasta (*marketplace*) [74] jo lukuisia.

Mainittujen tekoälyratkaisujen hyödyntäminen ei kuitenkaan lähtökohtaisesti ole ilmaista. ChatGPT:n hyödyntäminen VSCodessa edellyttää OpenAI:n API-rajapinnan käyttämistä, mistä OpenAI perii maksua. Samaten digitaalisten assistenttivideoiden generoiminen on myös maksullista.

6 Yhteenveto

Tässä työssä kehitettiin verkko-oppimisympäristöä, jonka tarkoituksena on tarjota erityisesti aloitteleville opiskelijoille vakioitu ympäristö ohjelmoinnin opettelemiseen. Opiskelijat säästävät yhden alkuvaiheen, kun heidän ei ole tarpeen asentaa ohjelmointiympäristöä omalle koneelleen. Samalla vältytään mahdollisilta versioristiriidoilta ja yhteensopivuusongelmilta. Lisäksi vakioidun verkko-oppimisympäristön käyttö madaltaa kynnystä laitteistovaatimusten osalta: riittää, että opiskelijalla on tietokone, joka kykenee suorittamaan modernia nettiselainta.

Kehitettyyn verkko-oppimisympäristöön kuuluu oppimisanalytiikan mahdollisuus. Mikäli opiskelija luo käyttäjätunnuksen Wakapi-ohjelmistoon, jota suoritetaan samalla palvelimella kuin verkko-oppimisympäristöäkin, aletaan opiskelijan ajankäytöstä oppimisympäristössä kerätä dataa. Tätä dataa pääsevät katsomaan sekä opiskelija itse että opettaja. Toiveena on, että opiskelijalle oman ajankäyttödatan näkeminen on oivalluksia synnyttävää ja motivoivaa. Opettajan kannalta datan toivotaan auttavan häntä kehittämään kurssia ja kiinnittämään huomiota opiskelijoiden oppimisen kannalta keskeisiin asioihin.

Kun työ aloitettiin, verkko-oppimisympäristöstä oli kehitetty alustava versio. Tämän työn kannalta keskeiseksi muodostui ensinnäkin tehdä jo olemassa oleva ympäristö näkyväksi, eli kuvata sen toiminta ja tämän jälkeen päivittää ympäristö käyttämään ohjelmistojen tuoreita versioita ja korjata ympäristöön liittyvät mahdolliset virheet.

Ympäristön lähtötilanne on kuvattu tässä raportissa ja päivittäminen ja korjaukset ovat toteutettu onnistuneesti. Verkko-oppimisympäristön ohjelmointieditori VSCode aukeaa opiskelijalle suoraan, kun hän syöttää nettiselaimeensa verkko-oppimisympäristön osoitteen ja kirjautuu sisään Kajaanin ammattikorkeakoulun GitLab-tunnuksilla. Taustalla suoritetaan päivitettyjä ohjelmistoja.

Lisäksi työssä oli tarkoituksena antaa ideoita verkko-oppimisympäristön edelleen kehittämiseksi ja mahdollisuuksien mukaan käytännössä toteuttaa nämä ideat. Erilaisia ideoita ja ympäristöön liittyviä havaintoja syntyi työn aikana paljonkin. Osalle ideoista annettiin konkreettinen toteutustapaehdotus tai käytäntöön viety toteutus.

Esimerkiksi yksi toteutustapaehdotus tehtiin lisädatan keräämisestä. Luvussa *5.3.1 Lisädatan kerääminen* esitettiin menetelmä, jolla voitaisiin kerätä dataa kurssitehtävien suoritusyrityskerroista lokitietoja analysoimalla. Menetelmään sisältyi kuitenkin muutama puute. Koska dataa olisi

kerätty Wakapista erillisenä menetelmänä, ei opiskelija olisi voinut tarkastella dataa Wakapin visualisointinäkymässä. Lisäksi opiskelijalle ei olisi välttämättä ollut selvää, että datan kerääminen menee ”päälle”, kun lokitiedon keräämistaso nostetaan korkeaksi. Lopulta suoritusyrityskertojen määrän laskemista ei toteutettu käytännössä.

Käytännössä sen sijaan toteutettiin erillisen visualisointiratkaisun luominen. Wakapin omat visualisointinäkymät koettiin rajallisiksi eivätkä ne tarjonneet opettajalle mahdollisuutta tarkastella visualisointeja koostetusti. Tämän myötä arkkitehtuuriin lisättiin komponentiksi Grafana, jolle määriteltiin datalähteeksi ympäristön PostgreSQL-tietokanta. Data haettiin tietokannasta Grafanan kautta räätälöidyin SQL-kyselyin. Kyselyjen tuloksista muodostettiin paneeleita, joita yhdistelemällä rakennettiin koontinäkymä, jota opettaja pääsee tarkastelemaan.

Seuraavassa kehitysvaiheessa opettajan Grafana-koontinäyttöjä voisi laajentaa. Niihin voisi lisätä erityisesti kurssien tehtäväkohtaista ajankäyttöä opiskelijoittain kuvaavia visualisointeja. Olisi hyvä, mikäli yksi koontinäyttö antaisi yleiskuvan yhdestä kurssista sen eri tehtävineen niin, että mahdolliset erityistä huomiota vaativat osa-alueet olisivat havaittavissa yhdellä vilkaisulla.

Tämän lisäksi koontinäyttöjen katselua voisi sujuvoittaa luomalla niitä varten erillisen nettisivun, johon koontinäytöt upotettaisiin. Tällä tavoin opettajan ei tarvitsisi erikseen kirjautua Grafanaan koontinäyttöjä katsellessaan, ja Grafana voitaisiin jättää etupalvelimen ulottumattomiin. Tarkoitukseen pyhitetyllä nettisivulla koontinäytöt lukittaisiin muokkaamiselta ja niihin pääsy suojattaisiin salasanalla.

Lisäksi seuraavassa kehitysvaiheessa olisi hyvä miettiä, olisiko MOOC.fi-kurssien tehtävänannot ja teoriaosuudet integroitavissa verkko-oppimisympäristön sisään. Tällä tavoin opiskelijan ei tarvitsisi pitää auki erillistä nettisivua tehtäviä lukiessaan ja myös tehtävien ja teorian lukemiseen käytetty aika kertyisi osaksi oppimisanalytiikkaa.

TMC-laajennuksen osalta voisi tarkastella sitä, olisiko Jupyter-ympäristö määriteltävissä niin, että käyttäjän ei tarvitse kirjautua laajennukseen uudelleen hänen oltuaan käyttämättä verkko-oppimisympäristöä riittävän pitkän ajan. Vähintään voitaisiin selvittää tarkemmin, olisiko ympäristö määriteltävissä niin, ettei käyttäjän tarvitsisi aktivoida suorittamiaan kursseja uudelleen TMC:n valikon kautta, vaikka uudelleenkirjautuminen laajennukseen vaadittaisiinkin.

Myös lisädatan keräämistä olisi hyvä miettiä. Voisiko olla muita oppimisanalytiikan kannalta merkityksellisiä datalähteitä kuin ajankäyttötilastot? Esimerkiksi tieto siitä, kuinka monta kertaa ku-

takin tehtävää on yritetty ennen onnistumista, voisi olla arvokasta. Tähän luotiinkin ratkaisuehdotus tämän työn puitteissa, mutta ehdotuksessa oli muutamia puutteita. Voisiko tämän työn yhteydessä esitettyä ratkaisua jalostaa tai löytyisikö ongelmaan ratkaisu muuta kautta?

Lopputuloksena on kuitenkin jo nyt kompakti paketti, joka toteuttaa tarkoituksensa: Se toimii vakioituna verkko-oppimisympäristönä ohjelmointia opetteleville opiskelijoille ja mahdollistaa myös oppimisanalytiikan. Verkko-oppimisympäristön kautta voi suorittaa ohjelmointikursseja ja samalla opiskelija pääsee seuraamaan omaa ajankäyttöään kurssien parissa. Opettajalla on puolestaan käytettävissään ajankäytön koontinäkymä. Luotu verkko-oppimisympäristö nojaa yksinkertaiseen ideaan, jota on kuitenkin tähän saakka käytetty vain hyvin vähän: ympäristö hyödyntää taustalla JupyterHubin teknologiaa, mutta avaakin opiskelijalle täysiverisen VSCode-ohjelmointieditorin tarvittavine liitännäisineen.

Lähteet

1 Opetushallitus. Perusopetuksen opetussuunnitelman perusteet. 2014 [viitattu 4.4.2023]. Saatavilla: https://www.oph.fi/sites/default/files/documents/perusopetuksen_opetussuunnitelman_perusteet_2014.pdf

2 Opetus- ja kulttuuriministeriö. Koulutuksen tietoyhteiskuntakehittäminen 2020. Parempaa laatua, tehokkaampaa yhteistyötä ja avoimempaa vuorovaikutusta. 2010. http://urn.fi/URN:ISBN:978-952-485-950-9

3 Moodle. About Moodle FAQ. 2023 [viitattu 5.4.2023]. Saatavilla: https://docs.moodle.org/401/en/About_Moodle_FAQ

4 Peda.net. Tietoa Peda.netistä. 2023 [viitattu 5.4.2023]. Saatavilla: https://peda.net/info

5 ViLLE. Esittely. 2023 [viitattu 5.4.2023]. Saatavilla: https://www.oppimisanalytiikka.fi/ville/

6 Google for Education. Google Classroomin suomenkielinen etusivu. 2023 [viitattu 5.4.2023]. Saatavilla: https://edu.google.com/intl/ALL_fi/workspace-for-education/classroom/

7 Panda, Y. Jupyter-videochat documentation. 2023 [viitattu 6.4.2023]. Saatavilla: https://jupyter-videochat.readthedocs.io/

8 Jupyter. Jupyter Project Documentation. 2023 [viitattu 5.4.2023]. Saatavilla: https://docs.jupyter.org/en/latest/

9 JupyterLab. Overview. 2023 [viitattu 5.4.2023]. Saatavilla: https://jupyterlab.readthedocs.io/en/stable/getting_started/overview.html

10 JupyterHub. JupyterHubin dokumentaation etusivu. 2023 [viitattu 5.4.2023]. Saatavilla: https://jupyterhub.readthedocs.io/en/stable/

11 Coder/GitHub. Coder-server-tietovarasto. 2023 [viitattu 25.4.2023]. Saatavilla: https://github.com/coder/code-server

12 CSC. What is Pouta. 2023 [viitattu 5.4.2023]. Saatavilla: https://docs.csc.fi/cloud/pouta/pouta-what-is/

13 Kubernetes. Overview. 2023 [viitattu 5.4.2023]. Saatavilla: https://kubernetes.io/docs/concepts/overview/

14 JupyterHub. OAuthenticator. 2023 [viitattu 5.4.2023]. Saatavilla: https://oauthenticator.readthedocs.io/en/latest/index.html

15 Visual Studio. Visual Studio Coden etusivu. 2023 [viitattu 3.4.2023]. Saatavilla: https://code.visualstudio.com

16 Coder. Code-serverin dokumentaation etusivu. 2023 [viitattu 3.4.2023]. Saatavilla: https://coder.com/docs/code-server/latest

17 GNU. GCC:n etusivu. 2023 [viitattu 6.4.2023]. Saatavilla: https://gcc.gnu.org/

18 GitHub/rage. TestMyCode for Visual Studio Code -tietovarasto. 2023 [viitattu 6.4.2023]. Saatavilla: https://github.com/rage/tmc-vscode

19 WakaTime. WakaTimen etusivu. 2023 [viitattu 3.4.2023]. Saatavilla: https://wakatime.com

20 PostgreSQL. About. 2023 [viitattu 4.4.2023]. Saatavilla: https://www.postgresql.org/about/

21 Helsingin yliopisto. TestMyCoden etusivu. 2023 [viitattu 5.4.2023]. Saatavilla: https://tmc.mooc.fi

22 OAuth. OAuth 2.0. 2023 [viitattu 3.4.2023]. Saatavilla: https://oauth.net/2/

23 GitLab. Configure GitLab as an OAuth 2.0 authentication identity provider. 2023 [viitattu 3.4.2023]. Saatavilla: https://docs.gitlab.com/ee/integration/oauth_provider.html#configure-gitlab-as-an-oauth-20-authentication-identity-provider

24 KajakDC. KajakDC Gitlab Ultimaten etusivu. 2023 [viitattu 3.4.2023]. Saatavilla: https://gitlab.dclabra.fi/

25 GitLab. GitLab for the education industry. 2023 [viitattu 3.4.2023]. Saatavilla: https://about.gitlab.com/solutions/education/

26 Helsingin yliopisto. Avoimet verkkokurssit eli MOOCit. 2023 [viitattu 8.4.2023]. Saatavilla: https://www.helsinki.fi/fi/hakeminen-ja-opetus/avoin-yliopisto/avoimet-verkkokurssit-eli-moocit

27 Agile Education Research -tutkimusryhmä. Ohjelmoinnin perusteet ja jatkokurssi 2023. 2023 [viitattu 3.4.2023]. Saatavilla: https://ohjelmointi-23.mooc.fi/

28 Leppisaari I, Puomio H. Itsearviointia oppimisanalytiikan tuella – miten virtuaaliryhmässä opitaan 21. vuosisadan taitoja? Teoksessa R. Kleimola, A. Pulkkinen & H-R. Aho (toim.) Analytiikan aalloilla, oppimisen jäljillä. Centria-ammattikorkeakoulu, 2021. (20–31). http://www.urn.fi/URN:NBN:fi-fe2021061036464

29 Elo, A. Oppimisanalytiikan hyödyntäminen kansainvälisessä ja monialaisessa YAMK-opintojaksossa. Teoksessa K. Lempinen, V. Suominen & M. Nevaranta (toim.) Lähtökohtia oppimisanalytiikkaan Satakunnan ammattikorkeakoulussa. Satakunnan ammattikorkeakoulu, 2021. (16–21). http://www.urn.fi/URN:NBN:fi-fe2021063040656

30 Nisula, L. Oppimisanalytiikka jatkuvan oppimisen tukena. Lumen – Lapin ammattikorkeakoulun verkkolehti 1/2022 [viitattu 7.4.2023]. Saatavilla: https://www.lapinamk.fi/loader.aspx?id=88b7240c-c80a-416e-8b19-a95d6be7c1c1

31 WakaTime. WakaTime for VS Code. 2023 [viitattu 3.4.2023]. Saatavilla: https://wakatime.com/vs-code

32 WakaTime. Introduction. 2023 [viitattu 3.4.2023]. Saatavilla: https://wakatime.com/developers

33 GitHub/muety. Wakapi-tietovarasto. 2023 [viitattu 3.4.2023]. Saatavilla: https://github.com/muety/wakapi

34 Mütsch F. Advanced Setup: Client side proxy. 4.4.2021 [viitattu 3.4.2023]. Saatavilla: https://github.com/muety/wakapi/wiki/Advanced-Setup:-Client-side-proxy

35 GitHub/mujx. Hakatime-tietovarasto. 2023 [viitattu 8.4.2023]. Saatavilla: https://github.com/mujx/hakatime

36 Siemens G. Call for Papers. [blogikirjoitus]. 22.7.2010 [viitattu 6.4.2023]. Saatavilla: https://web.archive.org/web/20101213140954/https://tekri.athabascau.ca/analytics/call-papers

37 Journal of Learning Analytics. About the Journal. 2023 [viitattu 6.4.2023]. Saatavilla: https://learning-analytics.info/index.php/JLA

38 Itä-Suomen yliopisto. Oppimisanalytiikan hyödyntäminen itseohjatun oppimisen tukemisessa koulutuspolun eri vaiheissa -tutkimushankkeen etusivu. 2023 [viitattu 6.4.2023]. Saatavilla: https://sites.uef.fi/oahot/etusivu/tutkimushanke/?lang=fi

39 Tampereen ammattikorkeakoulu. Oppimisanalytiikka – avain parempaan oppimiseen AMKeissa -hankkeen etusivu. 2023 [viitattu 6.4.2023]. Saatavilla: https://projects.tuni.fi/apoa/

40 OA-hanke. Ammatillisen koulutuksen oppimisanalytiikan kehittäminen -hankkeen etusivu. 2023 [viitattu 6.4.2023]. Saatavilla: https://oahanke.fi/

41 Helsingin yliopiston Koulutus- ja kehittämispalvelut HY+. Tiedosta tukea -hankkeen etusivu. 2023 [viitattu 6.4.2023]. Saatavilla: https://tiedostatukea.fi/

42 Suomen eOppimiskeskus ja Turun yliopiston Oppimisanalytiikan keskus. Analytiikka opetuksessa -hankkeen etusivu. 2023 [viitattu 6.4.2023]. Saatavilla: https://analytiikkaopetuksessa.fi/

43 Oppimisanalytiikka avuksi oppimiseen -hanke. Tietoja hankkeesta. 2023 [viitattu 6.4.2023]. Saatavilla: https://koulusta.fi/tietoja-hankkeesta/

44 Kajasilta H, Christopoulos C, Laakso M-J ja Nurmi J. Oppimisanalytiikan käsikirja. Turun yliopisto: Oppimisanalytiikan keskus, 2021 [viitattu 6.4.2023]. Saatavilla: https://projects.tuni.fi/uploads/2021/10/a607d223-oppimisanalytiikan-kasikirja-apoa.pdf

45 Suomen eOppimiskeskus. Hakutulokset: analytiikka. 2023 [viitattu 6.4.2023]. Saatavilla: https://eoppimiskeskus.fi/?s=analytiikka

46 Oulun yliopisto. AVAIN – opintopolkujen monitieteistä tutkimusta. 2023 [viitattu 6.4.2023]. Saatavilla: https://www.oulu.fi/fi/tutkimusryhmat/avain-opintopolkujen-monitieteista-tutkimusta

47 Auvinen A-M. Oppimisanalytiikka tulee – oletko valmis? Poluttamo, 2017 [viitattu 6.4.2023]. Saatavilla: https://poluttamo.fi/2017/08/02/oppimisanalytiikka-tulee-oletko-valmis/

48 Valtakunnallinen DigiErko-verkosto. Hakutulokset kohteelle: analytiikka. 2023 [viitattu 6.4.2023]. Saatavilla: https://digierko.fi/?s=analytiikka

49 Digikilta-verkosto. Haku: analytiikka. 2023 [viitattu 6.4.2023]. Saatavilla: https://www.digikilta.fi/?s=analytiikka

50 Suomen eOppimiskeskus. Oppimisanalytiikkaverkoston etusivu. 2023 [viitattu 6.4.2023]. Saatavilla: https://oppimisanalytiikkaverkosto.fi/

51 Turun yliopisto. Oppimisanalytiikan tutkimusinstituutin etusivu. 2023 [viitattu 6.4.2023]. Saatavilla: https://www.oppimisanalytiikka.fi/

52 Oppimisanalytiikkajaosto. Oppimisanalytiikan viitekehys. Hyvät käytännöt oppimisanalytiikan käyttöönotossa ja hyödyntämisessä. Helsinki: Opetus- ja kulttuuriministeriö, 2021. http://urn.fi/URN:ISBN:978-952-263-842-7

53 Chatti M.A, Dyckhoff A.L, Schroeder U, Thüs H. A reference model for learning analytics. International Journal of Technology Enhanced Learning 2013;4(5-6):318–331. Saatavilla: https://doi.org/10.1504/IJTEL.2012.051815

54 WakaTime. Frequently Asked Questions – What data is sent to your servers? 2023 [viitattu 8.4.2023]. Saatavilla: https://wakatime.com/faq#data-collected

55 CSC. Snapshots. 24.3.2020 [viitattu 11.4.2023]. Saatavilla: https://docs.csc.fi/cloud/pouta/snapshots/

56 Docker Hub. Jupyter/minimal-notebook. 2023 [viitattu 12.4.2023]. Saatavilla: https://hub.docker.com/r/jupyter/minimal-notebook/tags?page=1&name=ubuntu

57 Coder/GitHub. Releases. 2023 [viitattu 12.4.2023]. Saatavilla: https://github.com/coder/code-server/releases

58 MDXDave/GitHub. Version 2.3.7 shows version 2.3.5. Asia (issue) tietovarastossa muety/wakapi. 4.7.2022 [viitattu 12.4.2023]. Saatavilla: https://github.com/muety/wakapi/issues/386

59 Xgugeng/GitHub. Version 2.3.8 shows version release-arch-fix-test. Asia (issue) tietovarastossa muety/wakapi. 23.7.2022 [viitattu 12.4.2023]. Saatavilla: https://github.com/muety/wakapi/issues/409

60 JupyterHub. Capacity planning. 2023 [viitattu 12.4.2023]. Saatavilla: https://jupyterhub.readthedocs.io/en/stable/admin/capacity-planning.html

61 CSC. Virtual machine flavors and billing unit rates. 8.3.2023 [viitattu 12.4.2023]. Saatavilla: https://docs.csc.fi/cloud/pouta/vm-flavors-and-billing/#cpouta-flavors

62 CSC. Haka-käyttäjätunnistusjärjestelmä. 2023 [viitattu 12.4.2023]. Saatavilla: https://www.csc.fi/haka-kayttajatunnistusjarjestelma

63 Jupyterhub/GitHub. LTI JupyterHub Authenticator -tietovarasto. 2023 [viitattu 12.4.2023]. Saatavilla: https://github.com/jupyterhub/ltiauthenticator

64 Rage/GitHub. Ohjelmointi-23-tietovarasto. 2023 [viitattu 14.4.2023]. Saatavilla: https://github.com/rage/ohjelmointi-23

65 Selenium. Seleniumin etusivu. 2023 [viitattu 12.4.2023]. Saatavilla: https://www.selenium.dev/

66 MacroPower/GitHub. Wakatime Exporter -tietovarasto. 2023 [viitattu 13.4.2023]. Saatavilla: https://github.com/MacroPower/wakatime_exporter

67 Prometheus. Prometheuksen etusivu. 2023 [viitattu 13.4.2023]. Saatavilla: https://prometheus.io/

68 Grafana Labs. What is Grafana? 2023 [viitattu 13.4.2023]. Saatavilla: https://grafana.com/oss/grafana/

69 Grafana Labs. Provision the data source. 2023 [viitattu 19.4.2023]. Saatavilla: https://grafana.com/docs/grafana/latest/datasources/postgres/#provision-the-data-source

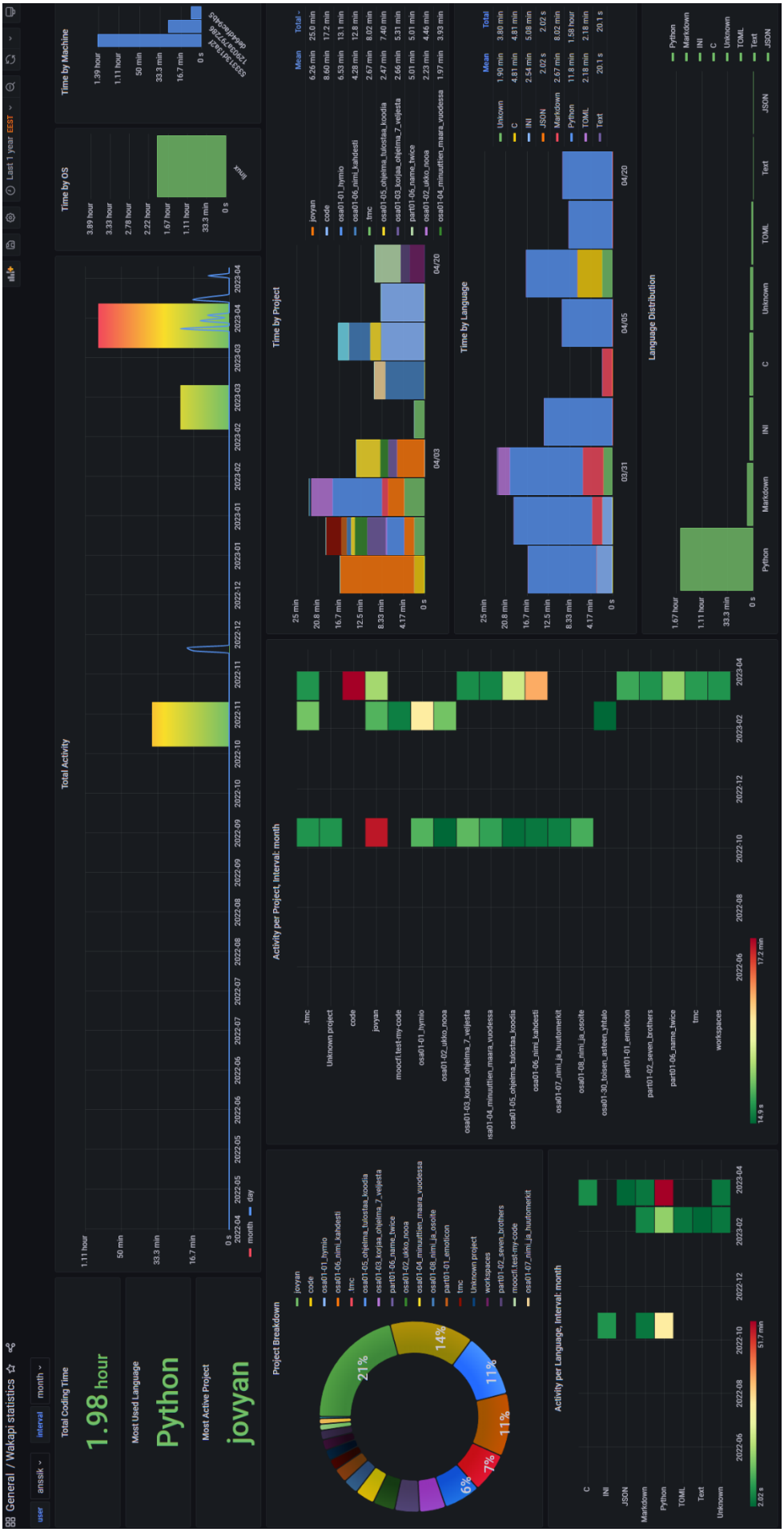
70 Muety/GitHub. Aggregate_durations.sql-tiedoston lähdekoodi muety/wakapi-tietovarastossa. 2023 [viitattu 19.4.2023]. Saatavilla: https://github.com/muety/wakapi/blob/master/scripts/aggregate_durations.sql

71 OpenAI. Introducing ChatGPT. 30.11.2022 [viitattu 14.4.2023]. Saatavilla: https://openai.com/blog/chatgpt

72 D-ID. D-ID:n etusivu. 2023 [viitattu 14.4.2023]. Saatavilla: https://www.d-id.com/

73 Elai. Elai:n etusivu. 2023 [viitattu 14.4.2023]. Saatavilla: https://elai.io/

74 Visual Studio. Visual Studio Coden laajennuskaupan etusivu. 2023 [viitattu 14.4.2023]. Saatavilla: https://marketplace.visualstudio.com/


General / Wakapi statistics
user
ansaik
interval
month
Last 1 year
Total Coding Time
1.98 hour
Most Used Language
Python
Most Active Project
jovyan
Total Activity
Time by OS
Time by Machine
Time by Project
Time by Language
Language Distribution
Project Breakdown
Activity per Project, Interval: month
Activity per Language, Interval: month