Matti Passoja

WEB-OHJELMOINTI

Web-ohjelmointipolku kesäharjoittelijoille

Opinnäytetyö
CENTRIA-AMMATTIKORKEAKOULU
Tietotekniikan koulutusohjelma
Kesäkuu 2018

TIIVISTELMÄ OPINNÄYTETYÖSTÄ

Centria- **ammattikorkeakoulu**	**Aika** Kesäkuu 2018	**Tekijä/tekijät** Matti Passoja
Koulutusohjelma Tietotekniikka		
Työn nimi Web-ohjelmointipolku		
Työn ohjaaja Sakari Männistö		**Sivumäärä** 27
Työelämäohjaaja Sakari Männistö		
Opinnäytetyössäni pidin kesäkurssia web-ohjelmoinnista opiskelijoille, jotka eivät ole saaneet alaan kuuluvaa kesätyötä. Tavoitteeni oli opettaa kaikki vaiheet, joita ohjelmoija tarvitsee verkkosivujen luontiin, ja dokumentoida saavutukseni tähän opinnäytetyöhöni. Kesäkurssilla keskityttiin projekti-muotoiseen opiskeluun. Kurssin aikana opetin virtuaalipalvelimen luonnin ja tarvittavat toimenpiteet, että opiskelija saa luomansa verkkosivun Internettiin. Opinnäytetyössäni haastattelin Kokkolan alueen IT-yrityksiä ja käytin sieltä saamiani neuvoja kurssini kulussa. Opiskelijoissa huomasin motivaatiota, kun projektit olivat semmoisia, joita itse haluaisin tehdä.		

Asiasanat Git, HTML, Javascript, Kesäkurssi, Linux, Ohjelmointi, PHP, Python, Palvelin, Web-ohjelmointi

Centria University **of Applied Sciences**	**Date** June 2018	**Author** Matti Passoja
Degree programme IT		
Name of thesis Web-programming path		
Instructor First name Last name		**Pages** 27
Supervisor First name Last name		

My thesis is summer school course about web-programming for students that don't have job for the summer. My plan was to give extensive course on web-programming from start to finish and document my findings. Summer school was based on project-oriented studying. It starts from setting up a virtual-server and teaching basics of programming languages used in Kokkola area and ending with knowledge about how to release your project to the Internet. Teaching is done using projects that I find interesting and have used in my own life.

Key words

Git, HTML, Javascript, Linux, PHP, Programming, Python, Server, Summer course, Web-programming

KÄSITTEIDEN MÄÄRITTELY

Callback – Funktio, joka kutsutaan toisen funktion sisällä.

CMS Content Management System – Sisällönhallintajärjestelmä, joka hallitsee koko palvelun sisältöä.

Dedicated server – Asiakkaalle varattu fyysinen palvelin palveluntarjoajan tiloissa

Frontend framework – Kehys verkkosivun ulkoasuun.

Full-stack ohjelmoija – Ohjelmoija, jolla on valmiudet ohjelmoida palvelinpuolen ja selainpuolen ohjelmistoja.

HTML – Hypertext Markup Language

Kirjasto – Kokoelma valmiiksi ohjelmoituja toimintoja.

Responsiivinen – Eri päätelaitteisiin mukautuva www-sivu

Serveri – Tietokone tai ohjelma, joka jakaa käyttäjille yhteisiä resursseja tai palveluita.

Skriptikieli – Ohjelmointikieli, joka muuttaa lähdekoodin tietokonekoodiksi käytön aikana.

Stacktrace – Raportti, joka antaa tietoa ohjelman aliohjelmista. Yleensä käytetään vian etsinnässä.

VPS Virtual Private Server – Virtuaalipalvelin

TIIVISTELMÄ

ABSTRACT

KÄSITTEIDEN MÄÄRITTELY

SISÄLLYS

1 JOHDANTO

Opinnäytetyöni tavoitteena on suunnitella web-ohjelmointiin kesäkurssi ottaen huomioon Kokkolan alueen IT-yritysten tarpeet. Olen harrastanut web-ohjelmointia vuodesta 2010 lähtien, ohjannut SQL-tools ohjelmiston luontia Centria-ammattikorkeakoululle ja kolme kesää johtanut opiskelijoiden kesäopiskelua. Kesäopiskelu on hyvä tapa oppia ohjelmointia käytännön avulla. Kesäopiskelu pohjautuu projektien tekoon ryhmissä, taikka yksin. Projektiluontoinen opiskelutapa pitää opiskelijan mielenkiinnon ohjelmoinnissa, jonka johdosta innostus ohjelmointiin ja luomiseen kasvaa. Tähän oppimistapaan on hyvä yhdistää Kokkolan alueen IT-yritysten tarpeet ja luoda semmoinen kurssi, joka antaa opiskelijalle valmiudet hyödyntää opittuaan työelämässä. Kesä on lyhyt aika, joten kurssin pitäisi keskittyä kaikista olennaisimpaan asiaan ja oppimistahti on nopea.

Opinnäytetyössä olen yhteydessä paikallisiin yrityksiin, haastattelen heitä ja kirjaan ylös heidän käyttämät teknologiat. Kokoan nämä teknologiat, suunnittelen niiden ympärille kurssin, joka koskee niitä osa-alueita. Kurssin tulisi olla joustava ja teknologian muuttuminen tulee ottaa huomioon.

2 KESÄHARJOITTELU KOULUSSA

Centria-ammattikorkeakoulu järjestää kesäisin tietotekniikan opiskelijoille, jotka eivät ole saaneet töitä, mahdollisuuden opiskella kesäharjoittelussa. Tämä kurssi on opiskelijoiden pitämä ja siitä voi kerryttää opintopisteitä harjoitteluun. Kesäharjoitteluun voi ilmoittautua minä opiskeluvuonna hyvänsä, ja kurssia valvoo kokeneempi opiskelija. Harjoittelu keskittyy projektitöihin. Edellisinä vuosina olemme opetelleet Unityn käyttöä ja C#-ohjelmointia. Vuonna 2016 kävimme läpi NodeJS:n, PHP:n, HTML:n ja CSS:n. Opetin myös Linux palvelimen asentamista.

Kesäharjoittelu kestää noin puolitoista kuukautta, jolloin opiskelija on viisi kertaa viikossa ja kahdeksan tuntia päivässä koululla opiskelemassa. Tämä tahti mahdollistaa tiiviin ja tehokkaan kurssin toteutuksen. Kesäharjoitteluun osallistuneet ovat näyttäneet korkeampaa innostusta ja osaamista muihin opiskelijoihin verrattuna.

3 WEB-OHJELMOINTIPOLKU

Kurssi aloitetaan palvelimen luonnilla. Ensimmäisten päivien tarkoituksena on näyttää, miten Linux palvelin asennetaan. Opiskelijoiden tulisi asentaa virtuaaliset palvelimet omille koneilleen ja asentaa perus LAMP-ohjelmistokokonaisuuden (Linux, Apache, MySQL, PHP). Tässä vaiheessa on hyvä kerrata Linuxin peruskomentoja ja käytäntöjä. Tämän jälkeen ohjaajan tulisi asentaa pilveen oma palvelin ja konfiguroida se opiskelijoiden käyttöä varten. On hyvä käydä läpi tietoturvallisia käytäntöjä kuten SSH-avainten käyttö, porttien avaaminen ja estäminen. Opiskelijoille kannattaa myös kertoa mahdollisuudesta hyödyntää hosting-palveluja, sekä esitellä muutamia esimerkkejä näistä.

Tunnusten luonnin jälkeen opiskelijan tulee luoda GitHub-tili ja opetella perusteiden käyttö Gitistä. Kaikki projektit tulisi lähettää Githubiin, jotta opiskelijoiden portfolio laajenisi. Web-ohjelmointi osuudessa ohjaaja aloittaa HTML-perusteista ja opettaa opiskelijoita luomaan staattisia verkkosivuja. On hyvä valita jo alussa jokin projekti, jota kehittää kurssin edetessä. HTML:n perusteisiin kannattaa käyttää noin viiko. Tämän jälkeen opiskelijat tulee harjoitella erilaisten frontend frameworkien käyttöä, kuten Bootstrap ja Foundation. Frameworkien yhteydessä kannattaa päivittää omat staattiset sivut käyttämään kyseistä frameworkia. Opetuksessa on hyvä käyttää Internetistä löytyviä opetusvideoita ja lähiopetusta. Perusteiden jälkeen on hyvä aloittaa palvelinpuolen ohjelmointikielen opettaminen. Palvelinpuolen ohjelmoinnissa perusteet on hyvä aloittaa PHP-kielellä. Kun opiskelijat ymmärtävät perusteet palvelinpuolelta, he voivat lisätä logiikkaa heidän staattisiin verkkosivuihinsa. JavaScriptiin kannattaa varata muita enemmän aikaa. JavaScriptissä tulisi käydä läpi käyttäjäpuolen ohjelmointia ja palvelinpuolen ohjelmointia. PHP:llä kannattaa kumminkin suorittaa suurin osa palvelinpuolen ohjelmoinnista, mutta taitavammat opiskelijat voivat luoda valmiita verkkosivuratkaisuja käyttäen Javascriptiä.

Kurssin tavoitteena on saada luotua oma web-sovellus alusta loppuun asti. Tämä sovellus voi olla jokin harjoitustyö yritykselle tai omaan käyttöön. Projektin on hyvä olla jokin kiinnostava ja käyttää hyödyksi kaikkia kurssin osa-alueita. Hyviä esimerkkejä on jokin yksin- tai moninpeli javascriptiä hyödyntäen, oma portfolio-verkkosivu käyttäen uusia teknologioita, taikka jokin omaa elämää helpottava sovellus.

3.1 Suunnittelu

Kurssi jaetaan neljään pääalueeseen: serverin luomiseen ja ymmärtämiseen, palvelinpuolen ohjelmointiin, käyttäjäpuolen ohjelmointiin ja JavaScriptiin. Monissa opintojaksoissa käydään yksittäisiä osia läpi,

mutta web-ohjelman luonti tyhjästä valmiiksi palveluksi jää usein opiskelijan itse selvitettäväksi. On tärkeää, että opiskelija ymmärtää mistä ja miten heidän luomansa sovellus on tullut. Palvelinpuolen ohjelmoinnissa tulisi keskittyä pääosin PHP:hen ja käydä päällisin puolin läpi muita ohjelmointikieliä.

Kurssilla tehtävät tulisi suorittaa versionhallintaa käyttäen. Opiskelijat voivat lähettää työnsä erillisiin versionhallintapalveluihin kuten Bitbucket ja GitHub. Haastattelemani yritykset käyttävät git-versionhallintaohjelmistoa. Gitin käyttö kaikissa projekteissa totuttaa opiskelijan käyttämään sitä ja laajentaa portfoliota.

3.1.1 Palvelimet

Centria-ammattikorkeakoululla on tarjolla Linux-opintojakso, jolla käydään läpi Linuxin asentamista ja Linuxin käyttöä. Tämä kurssi on hyvä pohja Linux-käyttöjärjestelmän ymmärtämiseen.
Serverit-osiossa tulisi käsitellä, mitä webhotellit, VPS ja dedikoidut palvelimet ovat ja mistä niitä voidaan hankkia. Osion alussa opiskelijoiden tulisi luoda oma virtuaali-serveri, johon he asentaisivat perus-LAMP stackin ja kävisivät läpi, miten se toimii. Perusasennuksen jälkeen, kurssin pitäjä voisi näyttää, miten virtuaali- serveri asennetaan pilvipalveluun. Palvelimelle tulisi antaa tunnukset jokaiselle opiskelijalle. Palvelimella pitäisi olla perus web-serverin ominaisuudet ja jokaisen opiskelijan pitää saada omassa hallinnassa oleva hakemistopolku palvelimelle näkyviin. On tärkeää opettaa opiskelijalle, miten Apache- ja Nginx-palvelinohjelmistoilla luodaan virtuaalisia www-palvelimia ja miten ne konfiguroidaan konfigurointitiedostojen avulla. Config-tiedostojen perusteisiin kuuluu rakenteen/moduulien ymmärtämien ja sääntöjen luominen. Opiskelijan on hyvä osata luoda turvallinen verkkosivuympäristö käyttäen oikeuksia ja todennusta.

Linuxin perusteita on hyvä kerrata tämän osion aikana. SSH-yhteys, SFTP, crontab, tiedoston oikeudet, kopiointi, siirtäminen ja purkaminen on hyvä hallita. Opiskelijan on hyvä antaa vapaasti liikkua palvelimella ja tutkia, miten mikäkin toimii. Jokaiselle käyttäjälle tulisi luoda oma tietokanta ja käyttäjätunnus, jolla on täydet oikeudet omaan tietokantaansa. Oma kotikansio ja tietokanta luo hänelle tilan, missä hän voi vapaasti opetella verkkosivujen luontia.

Palvelinosuuden tavoite on tutustuttaa opiskelijat siihen, miten palvelin luodaan ja toimii. Tämän osuuden lopussa opiskelija tulisi ymmärtää miten webhotellit, VPS ja dedikoidun palvelimen erottaa toisistaan, miten LAMP-stack luodaan ja miten opiskelija pystyy liikkumaan luomallaan palvelimella. Opiskelijan on myös ymmärrettävä, miten domain toimii ja miten se liitetään IP-osoitteeseen.

3.1.2 Palvelinohjelmointi

Backend-osuudessa opiskelija oppii eri palvelinpuolen ohjelmointikieliä. Pääosainen ohjelmointi tulisi keskittyä PHP:hen ja sen sovelluskehityksiin. Frameworkien opetuksessa opiskelija tulisi ymmärtää perusteet miksi frameworkejä käytetään ja miten ne toimivat. Frameworkeissä voidaan käyttää seminaarikurssia hyödyksi ja antaa jokaiselle opiskelijalle tehtäväksi tutkia yhtä uutta frameworkiä ja tehdä siitä seminaariesitys, jossa käydään läpi kyseisen frameworkin erikoisuudet ja hyödyt. Tämä mahdollistaa monen frameworkin läpikäynnin ja opiskelijat saavat laajan listan eri frameworkien hyödyistä ja haitoista.
Opintojaksolla tulee käydä läpi eri ohjelmointikielien eroavaisuuksia. Opiskelijan tulisi ymmärtää missä tilanteessa eri ratkaisut ovat hyviä. Ohjelmointikielen perusteisiin kuuluu ymmärrys kyseisen kielen hyvistä ja huonoista puolista, kielen rakenteesta ja opiskelijalla tulee olla valmiudet opiskella kieltä pidemmälle.
Backend-ohjelmoinnissa opetus aloitetaan yksinkertaisista ”Hello World” tehtävistä ja tietokannan yhdistämisestä ohjelmaan. Centria-ammattikorkeakoulussa on relaatiotietokannat kurssi, joten tietokantakyselyiden luonti pitäisi olla tuttua. Perusrakenteen jälkeen opiskelijoiden tulisi laajentaa tekemiään staattisia sivuja sisältämään logiikkaa ja tietokannan hyödyntämistä www-sivujen sisässä.
Datan luomisen yhteydessä on hyvä käydä läpi erilaisten rajapintojen luontia ja ymmärrystä. Hyvä harjoitus on luoda sovelluksia, jotka käyttävät jonkin kolmannen osapuolen rajapintaa. Perus JSON ja XML datan ymmärtäminen ja luominen on hyvä opetella jo varhain. Tämä mahdollistaa sivujen integrointia ja laajentaa mahdollisten projektien määrää.
Backend-ohjelmointiosuuden tavoitteena on saada opiskelija ymmärtämään ohjelmoinnin perusteet ja vähintään yhden backend kielen käytön. Opiskelijan tulisi ymmärtää, miten tietokannat toimivat ja miten niitä voidaan käyttää hyödyksi.

3.1.3 Selainohjelmointi

Frontend-puolella on hyvä aloittaa opetus HTML perusteista. Opiskelijoiden tulisi pystyä rakentamaan toimivat staattiset sivut käyttäen HTML:ää ja CSS:ää. Perusteiden jälkeen opiskelijat tulisi tutustuttaa eri frameworkien käyttöön. Hyviä esimerkkejä ovat Bootstrap ja Foundation. Frameworkien rakenne ja hyöty on hyvä osoittaa nopeasti sen jälkeen, kun opiskelijat osaavat luoda omia verkkosivuja. Kun opiskelijat ymmärtävät frameworkin toimintatavan, on hyvä antaa oikean elämän esimerkkejä sivuista, jotka

käyttävät niitä. Opiskelijoille voi antaa tehtäväksi replikoida valmiita verkkosivuja. Tämä auttaa opiskelijoita hahmottamaan, miten kyseisellä frameworkilla saa luotua hyvännäköisiä sivuja.

3.1.4 Javascript

Perusteiden jälkeen on hyvä alkaa käydä läpi Javascriptiä. Javascriptiä käyttäen opiskelijat voivat lisätä toimintoja verkkosivuilleen ja tehdä niistä interaktiivisia. Hyvää Javascriptosaamista tarvitaan monissa Kokkolankin alueen IT-yrityksissä, joten on hyvä käyttää paljon resursseja sen oppimiseen.

3.2 Ohjelmointikielet

Web-ohjelmoija tulee osata monia eri ohjelmointikieliä. Jokaisella kielellä on hyvät ja huonot puolensa.

3.2.1 PHP

PHP on yleisin palvelinpuolen ohjelmointikieli. Sitä käyttää yli 80% sivuista. (W3techs 2018.)
PHP ei ole kielenä parhain. PHP alkoi Rasmus Lerdorfin ”työkalupakkina”, jonka avulla hän pystyi helpottamaan omaa verkkosivujen luontia. PHP ei pyri olemaan yleinen skriptikieli, vaan PHP:n idea on löytää lyhin reitti ongelman ratkaisuun. (Yank, K 2002.)

PHP on helppo oppia, ja se on yleensä asennettuna kaikissa verkkosivujen isännöintipalveluissa. Tämän takia PHP on tärkeä kieli ymmärtää ja osata, vaikkei sitä paljoa käyttäisi. PHP:ta käytetään monissa CMS-ohjelmissa, esim. WordPress, Drupal ja Joomla. PHP on suosittu. PHP:hen löytyy laaja yhteisö, jota pystyy auttamaan ongelmissa ja luo omia sovelluksia käyttäen tätä kieltä. Dokumentointi on myös hyvin laaja.

PHP:n huonot puolet pohjautuu yleensä kielen alkuperäisiin komentoihin ja ratkaisuihin. Pitkät funktiot kuten mysql_real_escape_string tekevät koodaamisesta inhottavaa. PHP ei myöskään anna selvää stacktracee virheestä, joten virheen löytäminen voi olla hankalaa. Aloituskynnys PHP-kielisten sovellusten kehittämiseen on pieni, minkä vuoksi alalla on monen tasoista ohjelmoijaa. Halvin ohjelmistontoimittaja voi osoittautua kalliiksi esimerkiksi kehitetyn sovelluksen heikkojen tietoturvaominaisuuksien johdosta.

3.2.2 Javascript

Javascript on pääosin selainpään skriptauskieli. Javascriptillä on kumminkin toteutettu erilaisia palvelinpuolen ratkaisuja. Käytettäessä Javascriptiä myös palvelinohjelmointiin, saadaan hyödynnettyä ohjelmoijan mahdollinen hyvä Javascript osaaminen. Hyvä esimerkki palvelinpuolen Javascript-ajoympäristö on NodeJS.

NodeJS käyttää Googlen V8 javascript moottoria. Tämä tekee sovelluksista nopean, koska V8 kääntää javascriptin konekoodiksi. NodeJS käyttö mahdollistaa helpon tiedonsiirron palvelimen ja käyttäjän välillä, koska tiedon muuntamista ei tarvitse tehdä. Käyttämällä NodeJS:ää ohjelmoijalla on laaja kirjasto erilaisia lisäosia käytettävissä NPM(Node.js Package Manager) avulla. Lisäosat helpottavat puhtaan ja toimivan koodin tuottoa. On kuitenkin tärkeää valita lisäosat tarkkaan tuen ja käytettävyyden perusteella. NodeJS on asynkroninen ohjelmanajoympäristö. Normaalisti kirjoitus- ja lukutapahtumat tapahtuvat synkronisesti, tämä pakottaa ohjelman odottamaan, kunnes edellinen komento on suoritettu. Asynkroninen toimintamalli tekee NodeJS sovelluksista hyvin nopean.
NodeJS ei sisällä kääntäessä minkäänlaista tarkistusta, joten kaikkia virheitä ei välttämättä huomata heti. NodeJS soveltuu nopeaan ja ketterään ohjelmistokehitykseen.

```
var express = require("express");
var app = express();
var server = require("http").createServer(app);
var io = require("socket.io").listen(server);
var game = false;
users = [];
connections = [];

server.listen(8000);
console.log("Server running...");

app.get("/", function(req,res){
        res.sendFile(__dirname + "/index.html");

});

io.sockets.on('connection', function(socket){
        connections.push(socket);
        console.log("Connected: %s sockets connected", connections.length);

        socket.on("disconnect",function(data){

                if(socket.username){
                users.splice(users.indexOf(socket.username),1);
                updateUsernames();}
                        //disconnect
        connections.splice(connections.indexOf(socket), 1);
        console.log("Disconnected: %s sockets connected",connections.length);
                });

        socket.on("send message", function(data){

                io.sockets.emit('new message', {msg: data});
                console.log(data);
        });
```

KUVA 1. Esimerkki NodeJS:ssä ajettavaksi kirjoitettua Javascript ohjelmakoodia

NodeJS on suunniteltu skaalautuville verkkosovelluksille. NodeJs käyttää vähän tehoja, koska käyttämättömän ajan se nukkuu. (About Node.js)

NodeJS on ” Non-blocking” kieli. Tämä tarkoittaa sitä, että suurin osa komennoista ei pysäytä ohjelman kulkua. Blocking operaatioita ovat esimerkiksi tiedostoon kirjoittaminen ja lukeminen. Näihin komentoihin NodeJS antaa non-blocking mahdollisuudet, jotka hyödyntävät callback funktioita. Jotkin komennot antavat mahdollisuuden blocking komennoille, mutta niitä kutsutaan Sync komennoiksi. Kuvassa (2 ja 3) on esimerkki estävästä ja ei-estävästä koodista.

```
const fs = require('fs');
const dataa = fs.readFileSync('/tiedosto.md'); //estää ohjelman jatkumisen, kunnes tiedosto on luettu
console.log(dataa);
```

KUVA 2. Estävä koodi

```
const fs = require('fs');
fs.readFile('/tiedosto.md', (err, dataa) => {
    if (err) throw err;
    console.log(dataa);
});
```

KUVA 3. Ei-estävä koodi

Ensimmäisessä kuvassa ohjelma estää ohjelman jatkumisen, kunnes tiedosto on luettu. Tämä pysäyttää kaikki mahdolliset toimenpiteet ohjelmalta. Seuraavassa kuvassa ohjelma pystyy jatkamaan suoritusta samalla, kun tiedostoa luetaan. Console.log-komento tapahtuu vasta tiedoston luvun suorituksen jälkeen. (Overview of blocking vs non-blocking.)

3.2.3 Python

Python on helppo ohjelmointikieli. Koodi rakenne on helposti ymmärrettävissä, koska pakolliset sisennykset ja yleiset komentonimet tekevät siitä helposti luettavaa ja siten ymmärrettävää. Pythonilla voi tehdä paljon asioita ja web-sivujen luonti on yksi niistä. Web-sivujen luontiin on hyvä kumminkin hankkia oikeat välineet. Pieniin projekteihin käy hyvin Flask miniframework. Flask ei sisällä mitään ylimääräistä, se on tarkoitettu nopeaan verkkosivun luontiin.

Kattavampiin frameworkeihin sisältyy Django ja Pyramid. Nämä sisältävät valmiit käyttäjän tarkistukset, turvallisuuden, tietokantojen hallinnan ja paljon muuta. Puhdasta Python koodia voi myös käyttää, mutta on hyvä osata valmiiden frameworkien käyttöä.

3.3 Frameworkit

Frameworkin tehtävä on luoda sovellukselle yhdenmukainen rakenne, joka helpottaa ohjelmointia ja ylläpitoa. Frameworki käyttää pohjana haluttua kieltä ja sen tehtävä on tuoda komentoja ja ohjelmointitapoja, jotka luovat helpommin ymmärrettävää ja käytännöllisempää koodia. Framework sisältää yleensä jonkinlaisia suojausmenetelmiä ja valmiita todennuksen ja käyttäjähallinnan ratkaisuja. On hyvä osata pohjakieli ennen frameworkin käyttöä.

3.3.1 Teoria

Frameworkin tarkoitus on helpottaa ohjelmoijan työtä. Kun tavoitteena on ohjelmoida puhdasta koodia, on hyvä tapa opetella kieli ja ymmärtää sitä, mutta tällaisessa ohjelmoinnissa ohjelmoijan itse pitää tehdä suurin osa työstä. Frameworkit on suunniteltu helpottamaan ohjelmoijan työtä siten, että normaalit toimenpiteet, joita ohjelmoija tarvitsee, on tehty ohjelmoijalle valmiiksi. Frameworkin toiminnot liittyvät haluttuun tehtävään. Front-end frameworkeissä on esimerkiksi valmiit ratkaisut haluttuun ulkoasuun ja erilaisiin käyttöliittymän osiin, tällöin ohjelmoijan itse ei tarvitse luoda niitä.

Frameworkit on luotu halutun kielen päälle. Ne yhdistävät normaaleja komentoja siten, että haluttu toiminto saadaan suoritettua mahdollisimman helposti. Frameworkit ovat yleensä avointa lähdekoodia, joten kuka tahansa voi tutkia sen toimintatapaa ja löytää haavoittuvuuksia. Nämä haavoittuvuudet voidaan korjata ja tulos on turvallinen framework-ohjelmoijan käytettäväksi.

3.3.2 Bootstrap

Bootstrap on Twitterin työntekijöiden suunnittelema front-end framework. Bootstrap on yksi yleisimmistä HTML, CSS ja JS frameworkeista, joka on responsiivinen ja mobiililaitteille tarkoitettu.
Bootstrap suunniteltiin vuonna 2011 Twitterin ohjelmointitiimille. He tarvitsivat koodirakenteen, jota voi käyttää kaikkialla. Bootstrapin suunnittelivat entinen Twitterin insinööri Mark Otto ja Jacob Thorton. Heidän tavoitteena oli saada ohjelmointitiimin koodin yhtenäiseksi. Mark ja Jacob huomasi, että heidän ratkaisu oli tosi hyvä ja elokuussa 2011 he julkaisivat Bootstrapin avoimen lähdekoodin projektina Githubiin. Muutamassa kuukaudessa Bootstrapista tuli aktiivisin avoimen lähdekoodin projekti koko maailmassa ja nyt se on yksi käytetyimmistä front-end frameworkeistä. (Utterback 2014.)

3.3.3 ProcessWire

ProcessWire on Content Management System (CMS) ja framework, joka on luotu PHPn päälle. ProcessWire:n tavoite on luoda helppo, mutta monipuolinen paketti verkkosivujen luontiin. ProcessWire on inspiroitu jQuerystä. PHP-taito auttaa sen ymmärtämisessä, mutta sen osaaminen ei ole vaatimus ProcessWire sivujen luontiin. (What differentiates ProcessWire.) ProcessWire on luotu yleiseksi työkaluksi verkkosivujen luontiin.

3.4 Versionhallinta

Versionhallinnalla pidetään tallessa tiedostojen muutoshistoriaa niin, että haluttaessa voidaan palata tiettyyn tiedostoversioon tai useampien tiedostojen tilaan tiettynä ajankohtana. Versionhallinta tarjoaa myös ratkaisuja tiedostojen hallittuun muuttamiseen monikäyttäjäympäristössä. Kun samaa tiedostoa käyttää monta ihmistä, siitä normaalisti luodaan monta eri kopiota ja tallentaessa edellinen tiedosto korvataan kokonaan käyttäjän versiolla. Tämä aiheuttaa ongelman, kun samaa tiedostoa muokataan monessa paikkaa samaan aikaan. Versionhallinnan käyttöönotossa jokainen käyttäjä luo oman kopion tiedostaan ja tallennusvaiheessa version hallinta muistaa muutokset mitä käyttäjä on tehnyt. Tallennusvaiheessa muutokset liitetään vanhaan versioon ja tallennus tapahtuu vain niissä kohdin, missä muutosta on tapahtunut ja näistä muutoksista luodaan uusi tiedosto. On olemassa erilaisia versionhallintatapoja, mutta peruslogiikka on yleensä sama. (Christensson 2011.)

3.4.1 Toimintaperiaate

Hyvässä versionhallinnassa on tiedostorakenne, joka tallentaa kaikki modifikaatiot tiedoston luonnista asti. Kun moni ohjelmoija muokkaa samaa projektia, on iso mahdollisuus, että kaksi ihmistä muokkaa samaa kohtaa tiedostossa. Tämän takia on hyvä jaottaa projekti moneen pieneen haaraan ja tehdä muokkaukset omassa haarassaan. Kun ohjelmoija on valmis omissa muokkauksissaan, käyttäjä lisää muokkauksensa päähaaraan (yleensä nimike ”Master”) joka päivitetään käyttäjän tekemillä muutoksilla. Ohjelmoijat voivat ottaa haluamansa version ohjelmistosta tarkasteltavakseen checkout-toiminnolla. (Raymond 2009.)

3.4.2 Hyödyt

Versionhallinnan päähyöty on sovelluksen tiedostojen muutoshistorian säilyttäminen. Kun käyttää jotain käytetyimmistä version hallinta ohjelmista kuten Git, käyttäjän on annettava kommentti muokkauksistaan joka kerta, kun hän lisää muutoksensa ohjelmistoon. Tämä kannustaa käyttäjää kommentoimaan koodiaan ja kertomaan, mitä hän on tehnyt sille. Muutosten kommentointi helpottaa virheiden löytämistä ja muistamaan mitä käyttäjä on tehnyt koodilleen. Kun sitä käytetään useiden henkilöiden projekteissa, versionhallinta mahdollistaa projektin haaroittamisen jokaiselle käyttäjälle ja näin he voivat omistaa oman version pääprojektista. Kun joku yhdistää omaan ohjelmistohaaraansa tehdyt muutokset päähaaraan, niin hänen täytyy tässä yhteydessä mahdollisesti ratkaista yhdistämisessä syntyvät ristiriitaisuudet,

joita versionhallintaohjelma ei osaa automaattisesti ratkaista. Tämä mahdollistaa aina toimivan koodin ja jos jokin menee pieleen, commitin pystytään aina palauttamaan edelliseen tilaan. (Raymond, E)

3.4.3 Local Data Model

Local Data Model (LDM) on versionhallinnanmalli, jossa tiedostot ja muutoshistoria tallennetaan vain paikallisesti. Se mahdollistaa offline työskentelyn, mutta rajoittaa saatavuutta, koska käyttäjillä on oltava pääsy yhteiseen tiedostojärjestelmään. Local data model käy hyvin projekteihin, joita ei ole tarkoitettu jaettaviksi. Esimerkki local data modelista on Source Code Control System (SCCS), jota käytetään Unix-käyttöjärjestelmissä. Tätä ei kumminkaan enää aktiivisesti päivitetä.

3.4.4 Server-Client Data Model

Server-Client Data Model on versionhallintamalli, jossa ohjelmatiedostot ovat versionhallintapalvelin-ohjelmiston hallinnassa. Käyttäjät voivat asiakasohjelmien avulla kirjata tiedostoja ulos ja sisään palvelimelta.

3.4.5 Distributed Data Model

Distributed Data Model on yhdistelmä palvelin- ja lokaali-mallia. Tässä mallissa on keskitetty pääversio ja jokainen käyttäjä kopioi tämän version heidän omille koneilleen. Käyttäjät muuttavat paikallista kopiotaan ja voivat sitten siirtää muutokset yhteiseen pääversioon. Haaroittuminen pää versiosta on normaalia ja jokaisella on heidän oma versio koodistaan. Hajotettu malli mahdollistaa jokaisen itsenäisen työskentelyn myös silloin kun tietoliikenneyhteys palvelimeen ei ole esimerkiksi tilapäisesti saatavilla. Käyttäjät tekevät muutoksensa omaan haaraan ja sen jälkeen heillä on mahdollisuus yhdistää oma haaransa päähaaraan. Tämän takia käyttäjien on helppo ylläpitää monta haaraa ja muokata heidän omaa versiotansa ilman, että pääversio vahingossa muuttuu.

3.4.6 Git

Git-versionhallintajärjestelmän suunnitteli Linus Torvads vuonna 2005. Git on moderni ja nopea järjestelmä joka käyttää hyödyksi tehokkaita algoritmeja ja tietorakenteita. Git ei välitä tiedoston nimistä, vaan keskittyy tiedoston sisältöön. (What is git.)

Git on turvallinen, koska se käyttää crypto algoritmejä tiedon salaamiseen. Tämä suojaa projektia vahingolliselta ja tahallisilta historian muutoksilta. Tämä mahdollistaa tiedoston historian täyden seuraamisen. (What is git.)

3.5 Hostauspalvelut

Kun web-sovellus on tehty, on se hyvä saada näkyviin muille käyttäjille Internetissä. On erilaisia tapoja julkaista sivut Internetiin. Oman kotipalvelin ylläpito on yksi tapa päästä alkuun. Normaalit websovellukset eivät vie paljoa tehoja tietokoneelta, joten RaspberryPi voisi olla edullinen vaihtoehto aina päällä olevaksi pienimuotoiseksi palvelimeksi. Yleensä kumminkin kannattaa hankkia kolmannen osapuolen palvelin ja pitää sivut siellä. Webhotellit, VPS, dedikoidut palvelimet ovat hyviä vaihtoehtoja tähän, mutta jokaisessa on omat hyvät ja huonot puolensa. Jokaisessa ratkaisussa on hyvä varmistaa, kuinka paljon sovellus tarvitsee tiedonsiirtokapasiteettiä. Ostetuilla palveluilla voi olla tiedonsiirtorajoituksia.

3.5.1 Oma palvelin

Oman palvelimen ylläpito on aloittavalle web-ohjelmoijalle hyvä tapa kokeilla ja opetella verkkosivujen luontia, mutta on tärkeää tutkia Internet palvelun tarjoajan käyttöehdoista, saako niin tehdä. Hyvä puoli oman palvelimen ylläpidossa on se, että voit käyttää omaa tietokonettasi, taikka ostaa vaikka RaspberryPi:n. Kotitalouden tiedonsiirtoyhteys on usein asymmetrinen, eli nopeudet verkkoon päin voivat olla huomattavasti pienemmät kuin verkosta koneeseen päin. Myös yhteydenottojen salliminen verkosta kotiverkkoon lisäävät huomattavasti tietoturvariskejä.

3.5.2 Webhotelli

Webhotelli on nopea tapa saada helpot www-sivut tarjolle. Webhotellin hintaan yleensä sisältyy domain nimi, sähköpostiosoitteita, tietokanta, FTP yhteys palvelimelle ja mahdollisuus palvelinpään ohjelmiin esimerkiksi PHP-kielellä. Nämä ratkaisut ovat hyvä niille käyttäjille, jotka eivät halua itse hoitaa palvelimessa mitään ja haluavat vain luoda verkkosivuja. Webhotellissa asiakas on rajoittunut vain palvelun-tarjoajan antamiin rajoihin ja lisäominaisuuksien hankkiminen voi olla hankalaa.

3.5.3 Virtual Private Server

VPS on virtuaalinen kone, jossa käyttäjä vuokraa kapasiteettia palvelinkoneesta. Koko käyttöjärjestelmä on virtuaalisesti luotu ja käyttäjällä on pääsy vain omaan virtuaalikoneympäristöön. Virtuaalisissa servereissä kaikki käyttäjät jakavat samoja fyysisen koneen resursseja, joten on tärkeää varmistaa, että palvelinkone on tarpeeksi tehokas tarvittavaan työhön. Virtuaali-serverit ovat yleensä halpoja ja niiden yleinen laskutustapa on tuntihinta, eikä kuukausihinta. Tämä tekee virtuaali-serveristä halvan erilaisiin kokeiluihin. Virtuaali- serverissä käyttäjä pitää itse asentaa tarvittavat ohjelmat ja suojaukset. Virtuaali-serverin hintaan kuuluu yleensä oma IP-osoite ja itse palvelin. On tärkeää tarkistaa, että IP-osoite on oma, eikä jaettu muiden kanssa. Tällöin käyttäjä saa vain tietyt portit kyseisestä IP-osoitteesta.

3.5.4 Dedikoitu palvelin

Dedikoitu palvelin on asiakkaan yksinomaiseen käyttöön varattu fyysinen palvelin. Dedikoitu palvelin on kalliimpia muihin ratkaisuihin verrattuna, mutta silloin käyttäjällä on täysi kontrolli palvelimestaan. Dedikoidun palvelimen hintaan yleensä kuuluu oma IP-osoite. Dedikoidulla palvelimella käyttäjä itse on vastuussa turvallisuudesta ja ohjelmien asentamisesta. Tämä ratkaisu on hyvä, kun käyttäjällä on isommat vaatimukset sovellukseltaan ja sovellus vaatii ison osan tietokoneen tehoista.

3.6 Haastattelut

Opinnäytetyön käytännön osuutena olen haastatellut yrityksiä. Tavoitteena on saada Kokkolassa toimivilta IT-yrityksiltä tietoa heidän tarpeistaan ja käyttämistään järjestelmistä.

3.6.1 Preludi

Preludi on Kokkolan keskustassa toimiva mainostoimisto. Preludilla on kokemusta alalta jo 20 vuotta. Preludi keskittyy yritysten business to business verkkosivujen suunnitteluun ja luontiin. He ovat erikoistuneet verkkosivuihin ja sisällön tuottamiseen. Preludi etsii pääosin full-stack ohjelmoijia ja he ottavat opiskelijoita harjoitteluun ja mahdollisuudet työllistymiseen tulee tämän kautta. Pääosin verkkosivut ohjelmoidaan PHP-kielellä, ja heidän tämänhetkinen CMS:nsä on Processwire.
Heidän tarpeisiin kehittäjällä on tärkeää ymmärtää tiedon liikkuvuus ja relaatiot. Verkkosivujen luonnissa kehittäjän tulisi osata luoda layoutista toimiva responsiivinen sivu. Vaikkakin Processwire hoitaa suurimman osuuden tietokannoista itse, on tärkeää tietää, miten tietokannat toimivat. Javascriptin osaaminen on myös tärkeää.

Palkkaamisen kannalta näyttö on tärkeää. Palkattavalle on hyödyksi omistaa esim. GitHub tunnus.

3.6.2 Livion

Livion on Kokkolan keskustassa toimiva ohjelmointitalo. Livion keskittyy uusimpiin teknologioihin ja suurin osa ohjelmistoista luodaan Javascriptillä. Livion ottaa aktiivisesti opiskelijoita harjoitteluun. Keskustelussa heidän kanssa kävi ilmi, kuinka tärkeää Javascriptin ymmärtäminen on. Heidän mukaan opiskelijan kannattaa opetella Javascriptiä ja sen frameworkejä. Tärkeitä frameworkejä on React.js ja Bootstrap.

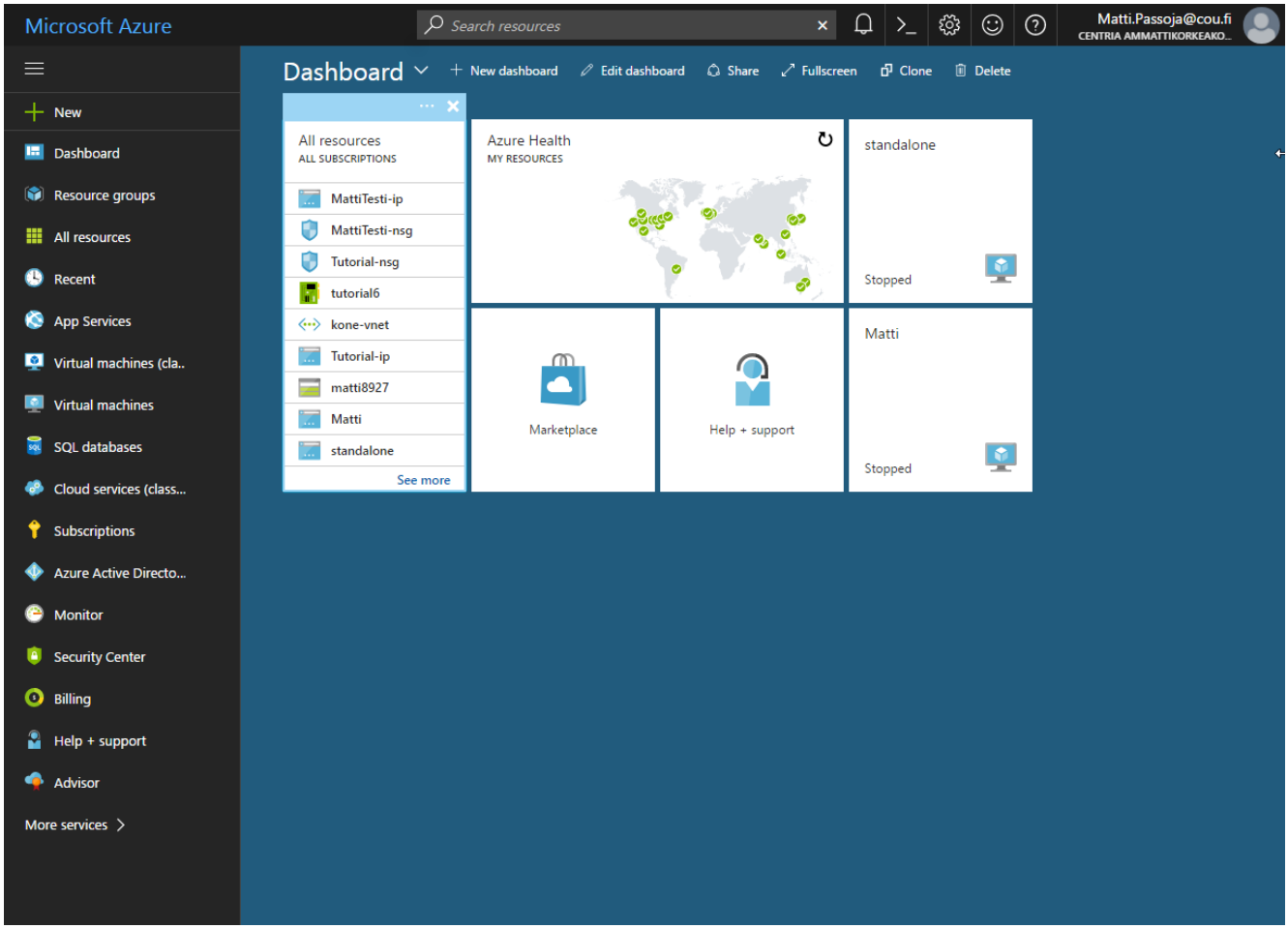
Livionin mielestä myös GitHub-tili näyttää työnantajalle, että työnhakija on kiinnostunut ohjelmoinnista ja että hän ohjelmoi omalla ajallaan. Livion käyttää pilvipalveluita ja niiden ymmärtäminen on tärkeää. Linuxin perusteiden osaaminen on hyödyksi. Opiskelijan tulisi pystyä liikkua Linux-terminaalilla sujuvasti ja osata käyttää Gittiä. Livionille Javascript on hyvin tärkeää ja sitä pitäisi painottaa. PHP on heidän mielestä vanhaa teknologiaa ja sen opetus pitäisi jättää taka-alalle.

3.7 Webserverin asennus Azure pilvipalveluun

Kesäkurssilla opiskelijat käyttävät Azuren pilvipalvelussa ylläpidettyä palvelinta. Tässä osiossa käyn läpi, miten asennetaan yksinkertainen ja turvallinen webserveri opiskelijoiden käyttöön.

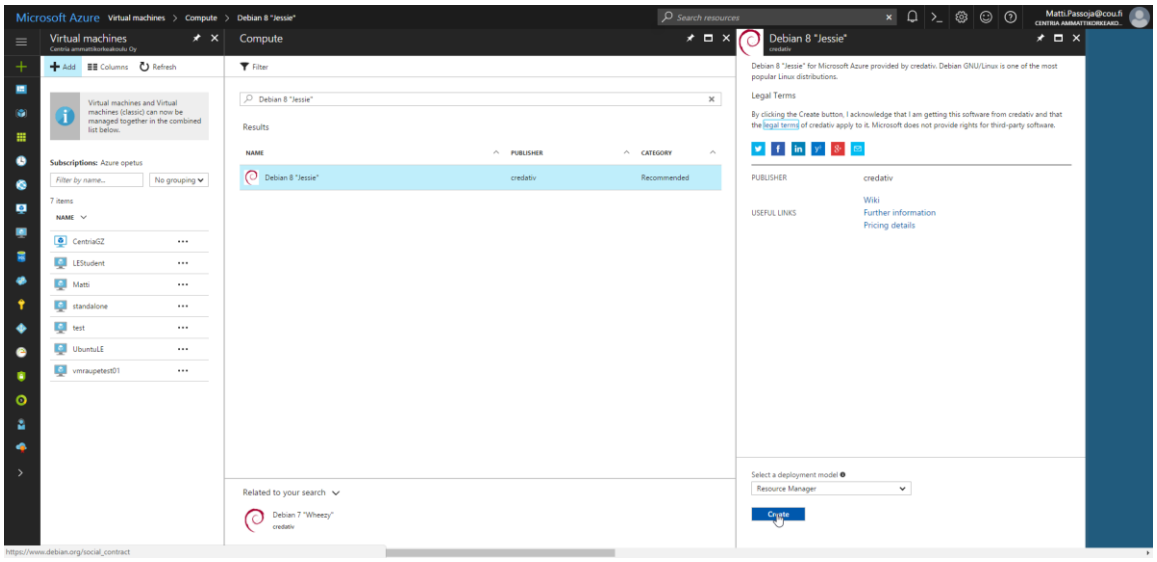
3.7.1 Azure

Azure on Microsoftin omistama pilvipalvelu. Azuressa käyttäjä voi luoda erilaisia pilviratkaisuja kuten virtuaali-tietokoneita, SQL-tietokantoja ja IOT-sovelluksia. Azure on vahvasti yhdistetty Microsoftin tuotteisiin ja palveluihin ja tämä mahdollistaa Active Directoryn käytön eri sovelluksissa.



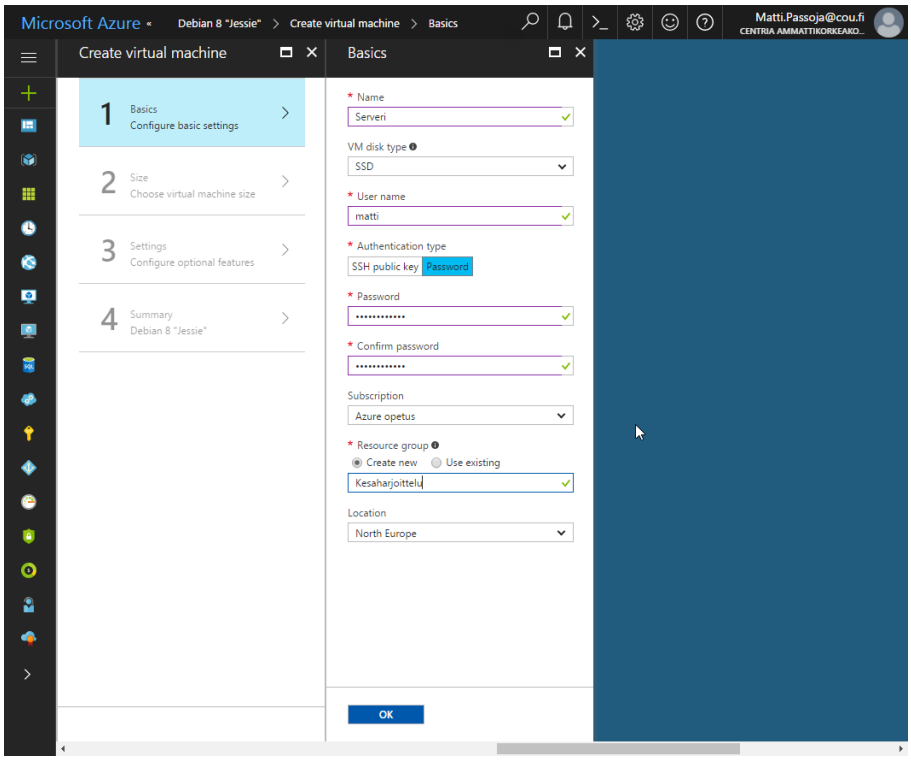
KUVA 4. Azure pääikkuna

Azuren pääikkuna näyttää palvelut ja käytettävät resurssit. Vasemmassa reunassa on lista kaikista mahdollisista palveluista, mitä Azure tarjoaa.



KUVA 5. Virtuaaliserverin luonti

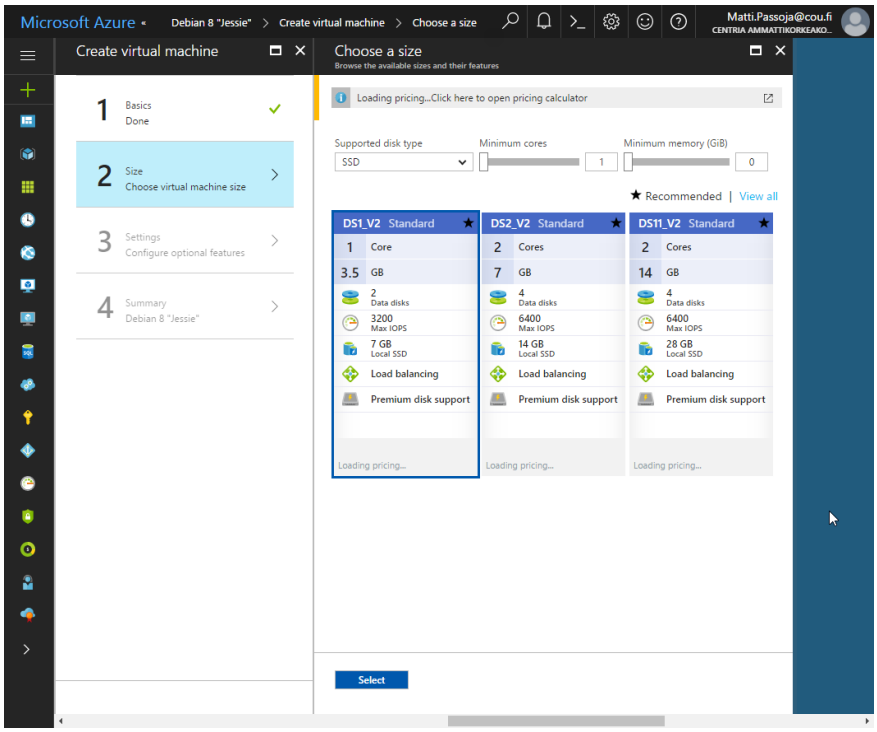
Valitse vasemmasta reunasta virtuaali-kone ja lisää uusi. Voit valita haluamasi käyttöjärjestelmän listalta. Esimerkki koneeseen valitsin Debian 8 ”Jessie” käyttöjärjestelmän. Luo virtuaali-kone painamalla ”Create”.



KUVA 6. Virtuaaliserverin konfiguraatio

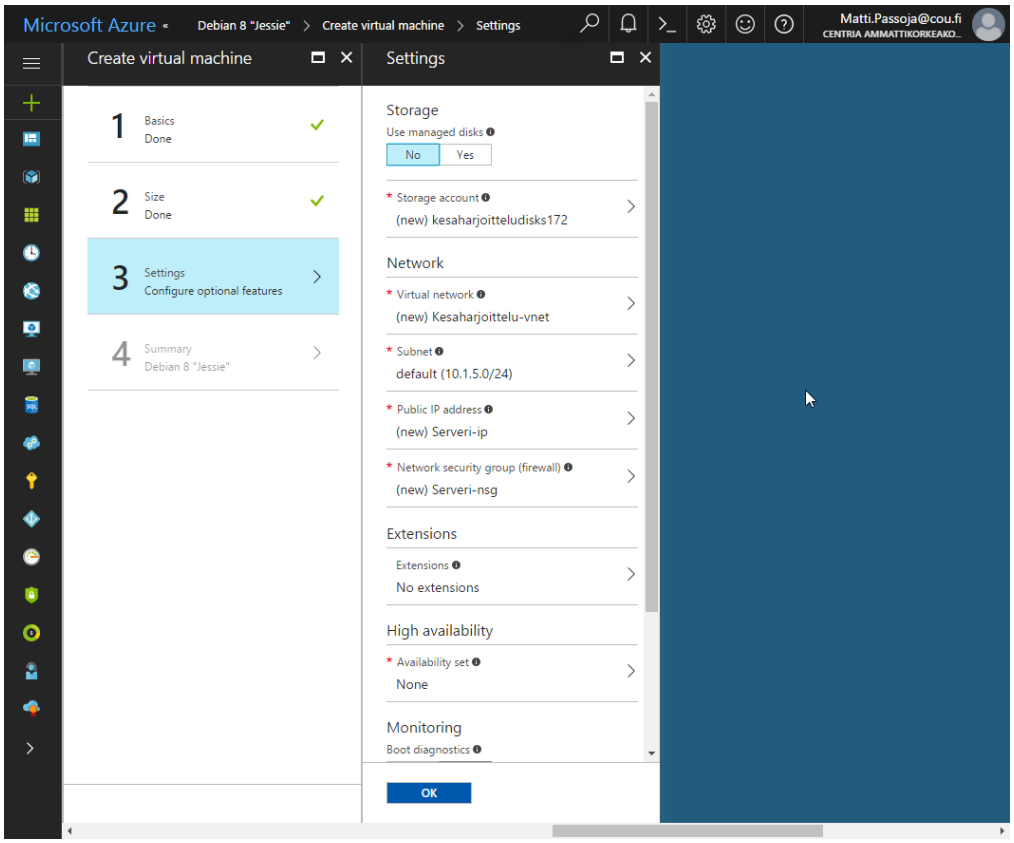
Ensimmäisessä ruudussa tulee valita palvelimelle nimi ja käyttäjän nimi. Todentamiseen voidaan käyttää SSH-avainta, tai salasanaa. On suositeltavaa käyttää SSH-avainta.

Resurssiryhmäksi loin uuden kesäharjoitteluryhmän.



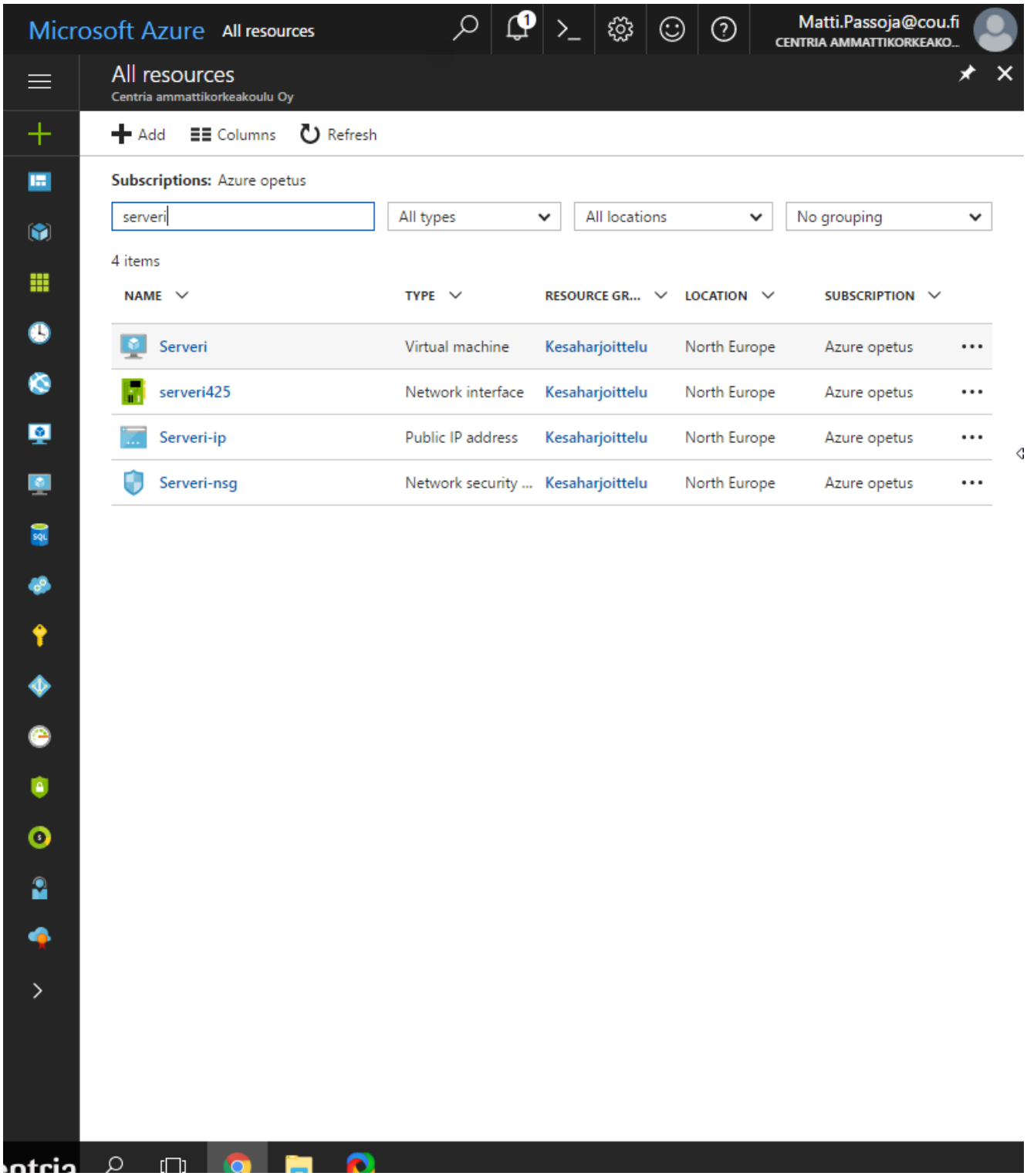
KUVA 7. Virtuaaliserverin koon valinta

Seuraavassa ruudussa käyttäjä voi valita haluamansa kokoisen palvelimen. Opiskelijoiden käytössä halvin mahdollinen palvelin on tarpeeksi hyvä.



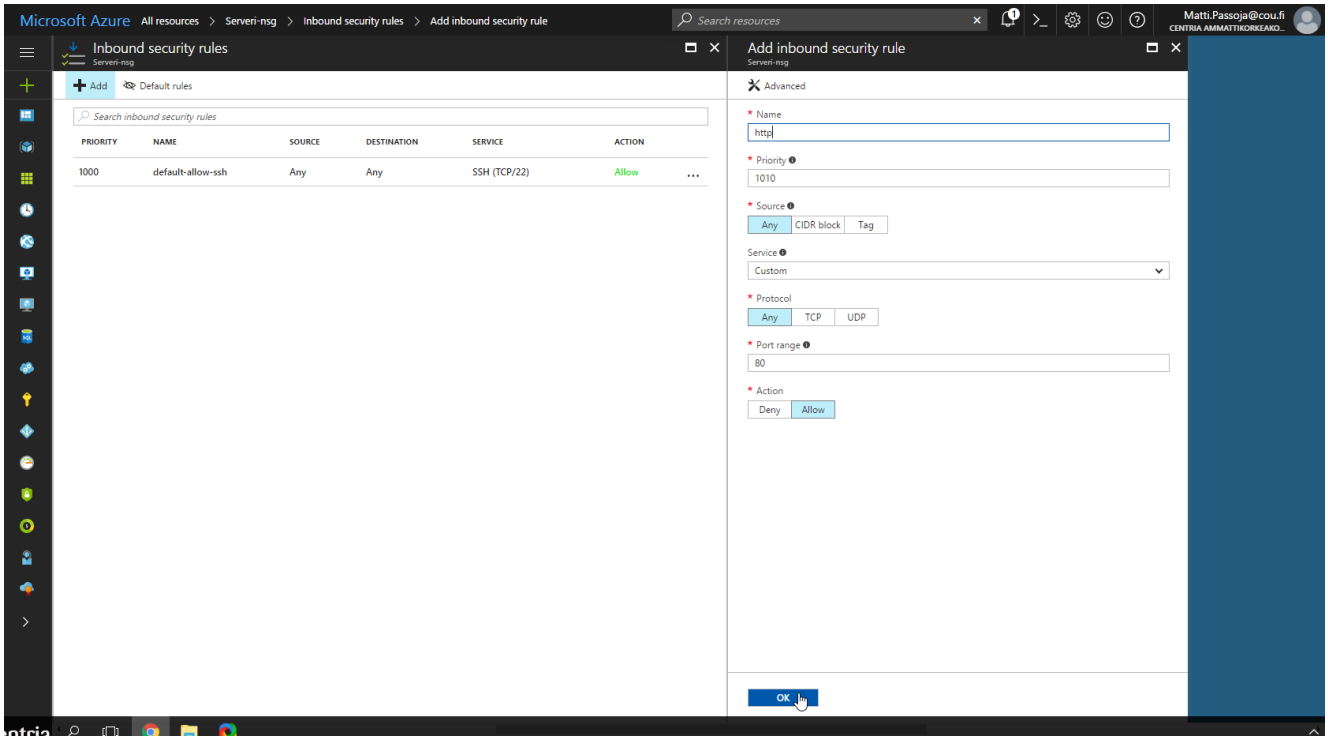
KUVA 8. Virtuaaliserverin valinnaisten asetusten konfiguraatio

Settings-ruudussa pystytään valitsemaan haluamansa asetukset edellä valitun ryhmän resursseista. Tässä ikkunassa käyttäjä voi ottaa käyttöön vanhan kovalevyn, taikka liittää virtuaali-serverin haluamaansa virtuaali-verkkoon.



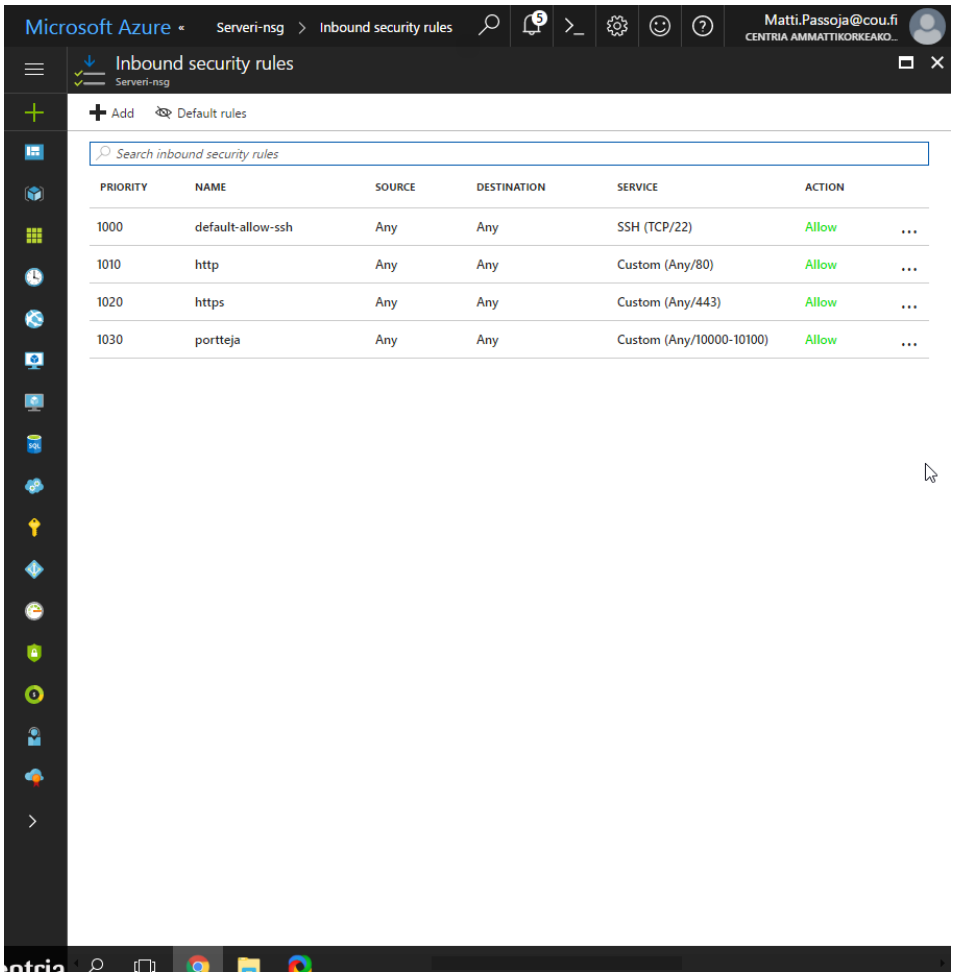
KUVA 9. Virtuaaliserverin osien etsiminen

Virtuaalikoneen käynnistyksessä kestää jonkin aikaa, ja sen jälkeen voidaan katsoa luotuja komponentteja ”All resources”-välilehdestä. Virtuaalikone sisältää itse virtuaalikoneen, IP-osoitteen, verkkokortin ja verkkoturvallisuusryhmän. Verkkoturvallisuusryhmässä pystyt hallitsemaan verkkoliikenteen sääntöjä, kuten portteja.



KUVA 10. Palomuuriasetusten konfigurointi

Lisätäksesi portin, käyttäjän tulee painaa Verkko turvallisuus ryhmää ja ”Add” painiketta. Tässä käyttäjä voi nimetä säännön, antaa sille prioriteetin ja halutut arvot. Oheisessa kuvassa sallitaan http-liikenne. Lähde saa olla mikä tahansa, protokolla mikä tahansa ja portti 80.



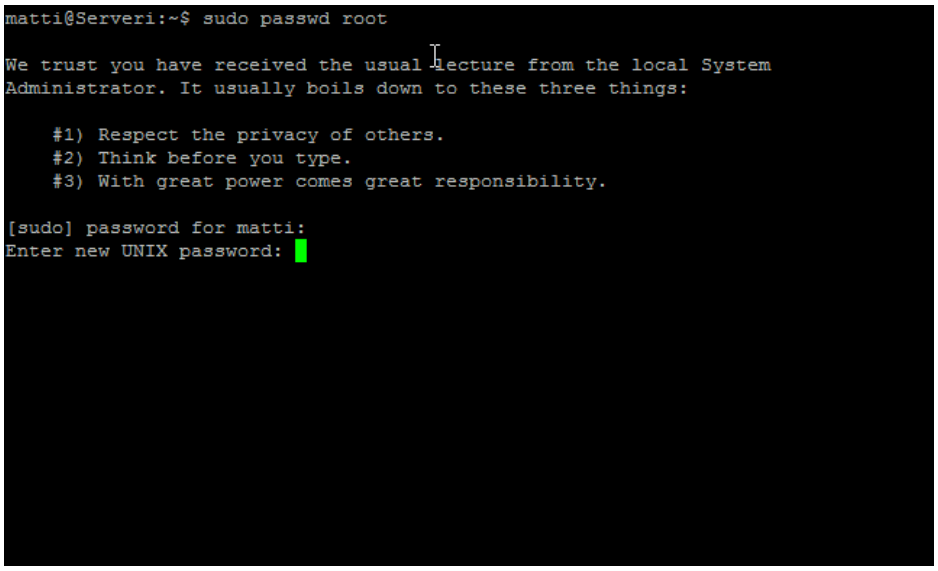
KUVA 11. Palomuuriasetusten selaaminen

Kesäharjoittelun palvelimella on hyvä olla avoinna vain SSH-, HTTP- ja HTTPS-portti. Lisäsin myös 100 porttia väliltä 10000 ja 10100. Näillä porteilla opiskelijat voivat käynnistää omia verkko sovelluksia.

3.7.2 Linux

Käyttöjärjestelmään tulee ottaa SSH-yhteys. Windows-koneilla tämä onnistuu esimerkiksi Putty sovelluksella. Linux-koneilla käyttäjä voi ottaa yhteyden terminaalin kautta käyttäen komentoa *ssh käyttäjä@iposoite*

3.7.2.1 Perusasennus



KUVA 12. Käyttäjän salasanan vaihto

Ensimmäisenä käyttäjän kannattaa lisätä salasana root käyttäjälle.

sudo passwd root

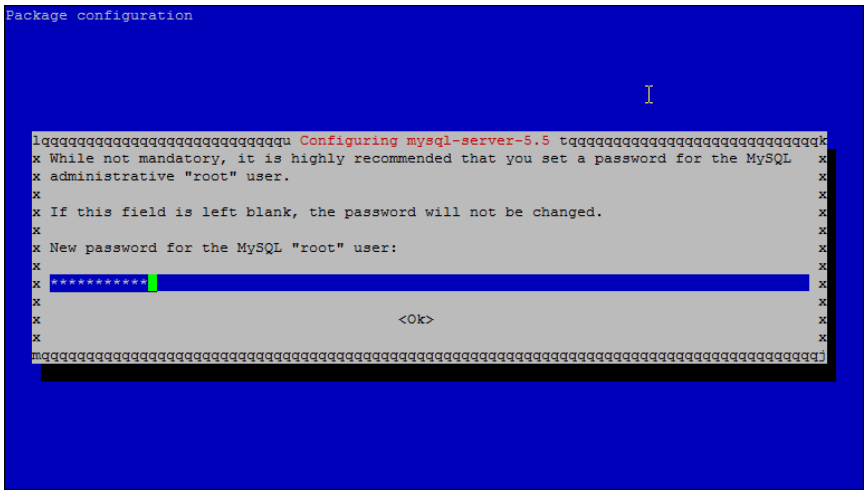
sudo mahdollistaa komennon suorituksen pääkäyttäjänä

passwd root vaihtaa salasanan käyttäjälle root

Tämän jälkeen käyttäjän tulisi kirjautua root-käyttäjänä. *Su*-komennolla käyttäjä voi kirjautua roottina käyttäen root käyttäjän salasanaa.

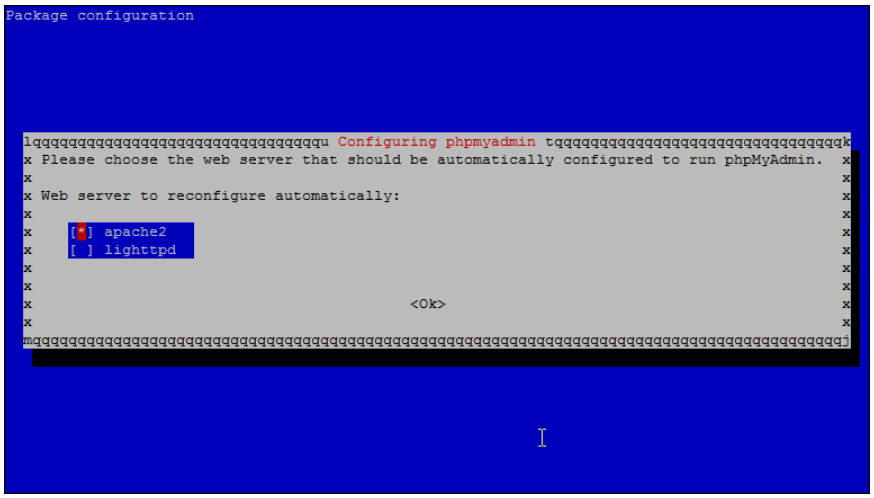
Seuraavaksi on hyvä päivittää käyttöjärjestelmä ja asennetut sovellukset käyttämällä koemntoja *apt-get update* ja *apt-get upgrade*. Päivityksen jälkeen käyttäjän tulee asentaa perus LAMP stack-ohjelmat kuten apache2, PHP ja MYSQL serveri. Suositeltavaa on myös jonkinlaisen MYSQL tietokanta hallintasovelluksen hankkiminen, kuten phpmyadmin.

Komento näiden asennukseen on *apt-get install apache2 php5 mysql-server phpmyadmin*



KUVA 13. MySQL pääkäyttäjän salasanan-valinta

Asennettaessa MySQL-serveriä, asennus kysyy root käyttäjälle salasanaa. MySQL-serverin pääkäyttäjällä on hyvä olla vahva salasana.



KUVA 14. PHPmyadmin automaattisen konfiguraation valinta

Phpmyadminin asennuksessa pystytään automaattisesti konfiguroimaan web serveri. Korosta nuolinäppäimillä käyttämäsi sovellus ja välilyöntiä painamalla pystyt valitsemaan sen. Asennuksen yhteydessä pitää myös antaa MySQL käyttäjän salasana. Phpmyadmin käyttää tätä salasanaa luodakseen tarvittavat tietokannat sovelluksen käyttämiseen.

Oppilailla tulisi olla oma osio verkko-osoitteessa. Tämä saadaan aikaan käyttämällä apachen moduulia userdir. Userdir moduulin saadaan käyttöön komennolla *a2enmod userdir*. Komennolla a2enmod voi-

daan käynnistää apachen eri moduuleja. Oletuksena userdir moduulin polut eivät hyväksy PHP:n käyttöä, tämän voidaan kumminkin sallia kommentoimalla /etc/apache2/mods-available/php5.conf tiedostosta rivit, jotka koskevat userdir moduulia.

```
</FilesMatch>
<FilesMatch ".+\.phps$">
    SetHandler application/x-httpd-php-source
    # Deny access to raw php sources by default
    # To re-enable it's recommended to enable access to the files
    # only in specific virtual host or directory
    Require all denied
</FilesMatch>
# Deny access to files without filename (e.g. '.php')
<FilesMatch "^\.ph(p[345]?|t|tml|ps)$">
    Require all denied
</FilesMatch>

# Running PHP scripts in user directories is disabled by default
#
# To re-enable PHP in user directories comment the following lines
# (from <IfModule ...> to </IfModule>.) Do NOT set it to On as it
# prevents .htaccess files from disabling it.
#<IfModule mod_userdir.c>
#    <Directory /home/*/public_html>
#        php_admin_flag engine Off
#    </Directory>
#</IfModule>
                                                          25,1          Bot
```

KUVA 15. Kommentoitu PHP esto userdir kansioista.

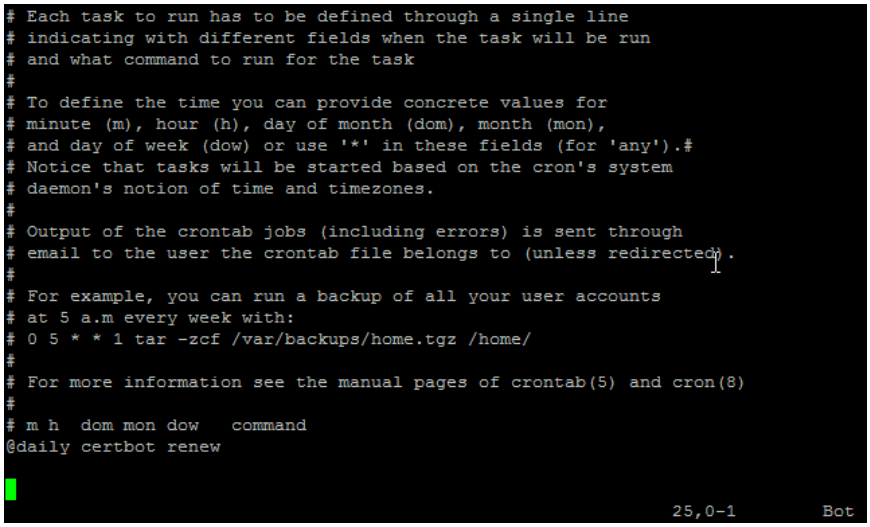
Näiden muokkausten jälkeen voidaan käynnistää apache2 uusiksi komennolla *service apache2 restart*. Jokaisella tulevalla käyttäjällä pitäisi olla public_html kansio kotihakemistossaan. Tämän kansion voidaan lisätä polkuun /etc/skel/. Jokainen uusi käyttäjä saa pohjan kotihakemistolleen /etc/skel-hakemiston mallin mukaisesti. Komennoilla *mkdir /etc/skel/public_html* ja *touch /etc/skel/public_html/index.html* pystytään luomaan kansiorakenne, joka vastaa haluttuja tarpeita. Jokainen käyttäjä, joka luodaan käyttämällä komentoa *adduser nimi* saa tarvittavat kansiot. Tätä komentoa tulisi käyttää luotaessa uusia käyttäjiä.

3.7.2.2 Lisätoimenpiteet

Perusasennuksen tapauksessa opiskelijat ottavat yhteyden palvelimelle IP-osoitteen kautta. On helpompaa ottaa yhteys domain-nimen avulla. Domain-nimi helpottaa osoitteen muistamista ja mahdollistaa salatun yhteyden palvelimelle. Jos domain-nimi on hankittu ja yhdistetty palvelimen IP-osoitteeseen, olisi hyvä asentaa jonkinlainen salattu yhteys. Harjoituksessa käytän Letsencrypt ilmaista sertifikaattia.

Sertifikaatin asentamiseen tarvitaan sovellus ”certbot”. Tämän sovelluksen voidaan asentaa komennolla *apt-get install python-certbot-apache -t jessie-backports*. Certbot tunnistaa automaattisesti käytetyt domain nimet apache-kansiosta, joten tiedostoon /etc/apache2/sites-available/000-default.conf tulisi lisätä ServerName www.domainnimitähän.fi

Certbot-komento tuo interaktiivisen ohjeistuksen, miten sertifikaatti asennetaan. Asennuksen jälkeen on hyvä lisätä komennolla *crontab -e* cronjobiin automaattinen sertifikaatin päivitys.



KUVA 16. Crontab-asetusten lisääminen

4 KOKEMUKSIA TOTEUTUKSESTA

Lopputulos projektista oli myönteinen. Opiskelijat oppivat liikkumaan oikealla palvelimella oletusten mukaan ja muutama opiskelija hankki oman palvelimen, jota he ylläpitävät omissa projekteissaan. Tietenkään kaikkien opiskelijoiden huomiota ei saatu pidettyä yllä koko kesän aikaa, mutta ne jotka olivat kiinnostuneet ohjelmoinnista, olivat huomattavasti kehittynyt. Suurin motivaatio mielestäni oli projektit ja pienet koodin pätkät, joista on hyötyä omissa projekteissa. Isoin osa kesästä meni PHP:n kanssa. Jälkeenpäin mietittynä, JavaScriptiin olisi kannattanut käyttää enemmän aikaa. Opiskelijat esittivät muutaman viikon välein heidän saavutuksiaan koko luokalle. Esityksissä opiskelijat saivat kysyä kysymyksiä ja esittäjän tuli vastata niihin parhaansa mukaan. Kesän aikana opiskelijat oppivat myös ryhmätyö-, esiintymis- ja ajanhallintataitoja. Opiskelijat työskentelivät antamissani projekteissa ja oppivat oikean työelämän taitoja samalla.

Valvojana opin tärkeitä taitoja opetuksesta. Opiskelijoiden taso vaihtelee, ja tämän ansiosta suunnitellut tunnit voivat kestää huomattavasti kauemmin kuin, mitä alun perin on suunniteltu. Parhaimman hyödyn opiskelijat saavat siitä, kun valvoja on aktiivinen ja käy auttamassa heitä. Harva opiskelija kysyy itse apua. Valvojana oppii myös esiintymis- ja puhumistaitoja. Tämmöinen projektipohjainen harjoittelu edistää kaikkea. Opiskelijat saavat kokemusta projektityöskentelystä, valvoja pystyy kehittämään taitoja johtamisessa ja esiintymisessä, ja koulu saa opiskelijoille opintopisteitä.

LÄHTEET

About Node.js. Saatavissa: https://nodejs.org/en/about/. Viitattu: 24.5.2018

Branch-Per-Feature Source Control. Saatavissa: http://lostechies.com/derickbailey/2009/07/21/branch-per-feature-source-control-part-2-how-theory. Viitattu: 24.5.2018

Christensson. 2011. Version Control Definition. Saatavissa: http://www.techterms.com/definition/version_control. Viitattu: 24.5.2018

December 2016 Web Server Survey. Saatavissa: https://news.netcraft.com/archives/2016/12/21/december-2016-web-server-survey.html. Viitattu: 24.5.2018

Detail in error messages. Saatavissa: http://phpsadness.com/sad/44. Viitattu: 24.5.2018

Overview of Blocking vs Non-Blocking. Saatavissa: https://nodejs.org/en/docs/guides/blocking-vs-non-blocking/. Viitattu: 24.5.2018

Raymond, E. Understanding Version-Control Systems. Saatavissa: http://www.catb.org/~esr/writings/version-control/version-control.html. Viitattu: 24.5.2018

Usage of server-side programming languages for websites. Saatavissa: https://w3techs.com/technologies/overview/programming_language/all. Viitattu: 24.5.2018

Utterback. 2014. What is Bootstrap? – The History and the Hype: Part 1 of 2. Saatavissa: https://www.prestashop.com/en/blog/what-is-bootstrap. Viitattu: 24.5.2018

What differentiates ProcessWire. Saatavissa: https://processwire.com/about/what/. Viitattu: 24.5.2018

What is git. Saatavissa: https://www.atlassian.com/git/tutorials/what-is-git. Viitattu: 24.5.2018

Yank, K. 2002. Interview – PHP's Creator, Rasmus Lerdorf. Saatavissa: https://www.sitepoint.com/phps-creator-rasmus-lerdorf/. Viitattu 24.5.2018