OULUN AMMATTIKORKEAKOULU

Jani Mustonen

Harjoittelutoiminnon siirto Asiosta Peppiin

Harjoittelutoiminnon siirto Asiosta Peppiin

Jani Mustonen
Opinnäytetyö
Syksy 2017
Tietojenkäsittelyn ko
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietojenkäsittely, tietohallinnon ja verkkopalveluiden koulutusohjelma

---

Tekijä(t): Jani Mustonen
Opinnäytetyön nimi: Harjoittelutoiminnon siirto Asiosta Peppiin
Työn ohjaaja: Teppo Räisänen
Työn valmistumislukukausi ja -vuosi: Syksy 2017 Sivumäärä: 29

Opinnäytetyön aiheena oli harjoittelutoiminnon siirto Asiosta uuteen perusrekisteriin eli Peppiin. Toiminnallinen opinnäytetyö rajattiin itse harjoittelutoiminnon tekemiseen ja sen prosessin käsittelyyn. Toimeksiantajana toimi Oulun ammattikorkeakoulu. Harjoittelutoiminnon tavoitteena oli yhtenäistää Oulun ammattikorkeakoulun eri yksiköiden harjoitteluprosessia ja mahdollistaa parempi tilastollinen seuranta opiskelijoiden tekemistä työharjoitteluista. Harjoittelutoiminnon tarkoituksena oli myös helpottaa opiskelijan ja opettajan työskentelyä ammattiharjoittelutoteutuksilla. Lisäksi tavoitteena oli tehdä hyvä ja kattava opinnäytetyö.

Tietoperustassa käsiteltiin tietokantaa ja sen suunnittelua. Lisäksi tietoperustassa käsiteltiin erilaisia ohjelmointikieliä, kuten HTML, PHP, JSON, MySQL ja CSS. Harjoittelutoiminto tehtiin kesällä 2017 Oamkin tiloissa. Asiossa ollut harjoittelutoiminto oli tehty PL/SQL -kielellä ja uusi perusrekisteri Peppi ei tue PL/SQL -kieltä. Tässä opinnäytetyössä tehty harjoittelutoiminnon käyttöliittymä on toteutettu HTML ja PHP -kielillä, ja taustalla toimiva tietokanta on MySQL -kanta.

Harjoittelutoiminto valmistui ajallaan elokuun alkupuolella 2017. Harjoittelutoiminnossa opiskelija tekee sähköisesti harjoittelusuunnitelman, jonka opettaja pystyy tarkistamaan ja hyväksymään tai palauttamaan sen opiskelijalle. Opiskelija tekee myös harjoittelun jälkeen harjoitteluraportin harjoittelutoiminnon kautta ja opettaja pystyy käsittelemään rapotin samalla tavalla kuin suunnitelman. Harjoittelutoimintoa tullaa jatkokehittämään myöhemmin tehtävän vaatimusmäärittelyn tuloksien mukaisesti.

Asiasanat: ohjelmointikielet, tietokanta, relaatiomalli, tietokantasuunnittelu, toiminnallinen opinnäytetyö, harjoittelutoiminto

ABSTRACT

Oulu University of Applied Sciences
Degree Programme in Business Information Systems, Information Management and Internet Services

---

Author(s): Jani Mustonen
Title of thesis: Transfer training function move from Asio to Peppi
Supervisor(s): Teppo Räisänen
Term and year when the thesis was submitted: 2017 Number of pages: 29

The topic of this thesis was transfering training function move from Asio to Peppi. This project-based functional thesis was limited to creating the training function and the process itself. The commissioner of this thesis was Oulu University of Applied Sciences. The objective of this thesis was the unifying Oulu University of Applied Sciences different units training process and allow statistical tracking of job trainings. The purpose training function was also collaboration between student and teacher during professional training. In additional the objective was to make a good and comprehensive thesis.

The theoretical background consisted of database, its design and facilitate programming languages such as HTML, PHP, JSON, MySQL and CSS. The training function was made in summer 2017 at OUAS premises. The old training in Asio was manufactured using PL/SQL language which Peppi does not support. In this thesis training function is implemented by using HTML and PHP languages. Database working in the background is MySQL database.

The training function was the completed-on time in the beginning of August 2017. In the training function a student makes the training plan which teacher can check and accept or return it back to student. After the professional training period report of that the student submit the report to teacher via training function. The training function will be further developed after requirement specifications are complete.

Keywords: programming language, database, relational model, database design, training function, project-base

SISÄLLYS

1 JOHDANTO

Oulun ammattikorkeakoulu on siirtynyt käyttämään syksyllä 2017 uutta perusrekisteriä, josta käytetään myös nimeä Peppi. Tämän johdosta vanhoja ohjelmia täytyy kirjoittaa uusiksi, sillä uusi perusrekisteri ei ymmärrä vanhoja ammattikorkeakoulun käytössä olevia ohjelmia, jotka on tehty PL/SQL-kielellä. Lisäksi vanhan perusrekisterin ohjelmat toimivat PL/SQL -kielellä Oracle -palvelimilla, josta Oamk haluaa luopua.

Tämän toiminnallisen opinnäytetyön tarkoituksena oli luoda uusi harjoittelutoiminto vanhan Asiossa olleen toiminnon pohjalta ja samalla tehdä parannuksia itse ohjelmaan. Uusi harjoittelutoiminto tehtiin PHP ja HTML -kielillä ja tietokantana oli MySQL. Ohjelma rakennetiin valmiiksi tehdyn sivustopohjan päälle, jonka on luonut Oulun ammattikorkeakoulun tietohallinnossa työskentelevä pääsuunnitelija Ville Valtonen. Lisäksi harjoittelutoiminnossa hyödynnettiin Valtosen tekemiä PHP Classeja mm. ohjelmaan kirjautumisessa ja lokitietojen arkistoinnissa. Opinnäytetyön tietoperustassa käsitellään tietokantaa ja sen suunnittelua sekä ohjelmassa käytettyjä ohjelmointikieliä PHP, HTML ja CSS sekä MySQL -tietokantaa.

1.1 Oulun ammattikorkeakoulu

Oulun ammattikorkeakoulu eli Oamk on Oulun alueella toimiva ammattikorkeakoulu, jossa voi opiskella kulttuuria, liiketaloutta, luonnonvara-alaa, sosiaali- ja terveysalaa sekä tekniikkaa. Koulutustarjontaa on sekä suomen että englanninkielisenä ja Oamk:ssa on mahdollista opiskella myös ylempi ammattikorkeakoulututkinto (Master-tutkinto). Vuonna 2016 Oamk:ssa oli n. 9000 opiskelijaa ja henkilökuntaa (sisältäen sivutoimiset) n. 640 henkilöä. Oamk:n liikevaihto oli vuonna 2016 n. 56 M euroa ja toimitusjohtajana sekä rehtorina toimii Jouko Paaso. Oulun ammattikorkeakoulun visio on olla pohjoisen Suomen johtava, monialainen ja

kansainvälinen ammattikorkeakoulu. Oamk:n arvoja ovat yhteisöllisyys, työelämäkumppanuus, kehittymishalukkuus ja tuloksellisuus. (OAMK 2017, viitattu 14.10.2017.)

1.2 Peppi (Perusrekisteri)

Peppi-tietojärjestelmä on uusi tapa yhtenäistää yliopistojen, ammattikorkeakoulujen ja toisen asteen koulujen tietojärjestelmiä. Järjestelmä sisältää paljon toimintoja, jotka on aikaisemmin tehty eri järjestelmillä ja näin Pepin tarkoituksena on yhtenäistää monia järjestelmiä yhdeksi isommaksi kokonaisuudeksi. Peppikonsortio on yhteenliittymä eri oppilaitoksille ja järjestelmäntoimittajille, joka sekä hallinnoi että kehittää Peppi-järjestelmäkokonaisuutta eteenpäin. Perusrekisteri otettiin ensimmäisen kerran käyttöön vuonna 2015 Metropoliassa. (Metropolia 2015, viitattu 23.7.2017.) Peppi-järjestelmä sisältää erilaisia toimintoja opiskelijoille ja opettajille. Opiskelijoille järjestelmästä löytyy oma työalusta, jossa he voivat suunnitella oman henkilökohtaisen opintosuunnitelman eli HOPS:n. Lisäksi Peppiin pystyy tekemään liitännäisiä, jotka hyödyntävät Pepin kattavia rajapintoja. (Eduix Oy 2017, viitattu 23.7.2017.)

1.3 Työntausta ja tavoitteet

Ohjelma tehtiin Oulun ammattikorkeakoululle ja toimeksiantajana toimi Oulun ammattikorkeakoulu. Yhteyshenkilönä ammattikorkeakoulussa oli tietohallinnonpäällikkö Samuli Malinen. Toimeksiantajan teknisenä ohjaajana toimi Ville Valtonen, keneltä saatiin harjoittelutoiminnon tekniset vaatimukset ja keneltä pystyttiin pyytämään teknistä tukea opinnäytetyön etenemisessä. Opinnäytetyö tehtiin Oamk:n tiloissa kesällä 2017.

Harjoittelutoiminnon tarkoituksena oli helpottaa harjoittelusuunnitelmien ja raporttien käsittelyä sekä yhtenäistää harjoitteluopintojaksojen käytännön prosesseja.

Lisäksi harjoittelutoiminto mahdollisti tilastollisen seurannan opiskelijoiden tekemistä työharjoitteluista. Tällä harjoittelutoiminnolla oli kiireellinen aikataulu, koska vanha järjestelmä poistui käytöstä uuden lukukauden alkaessa syksyllä 2017.

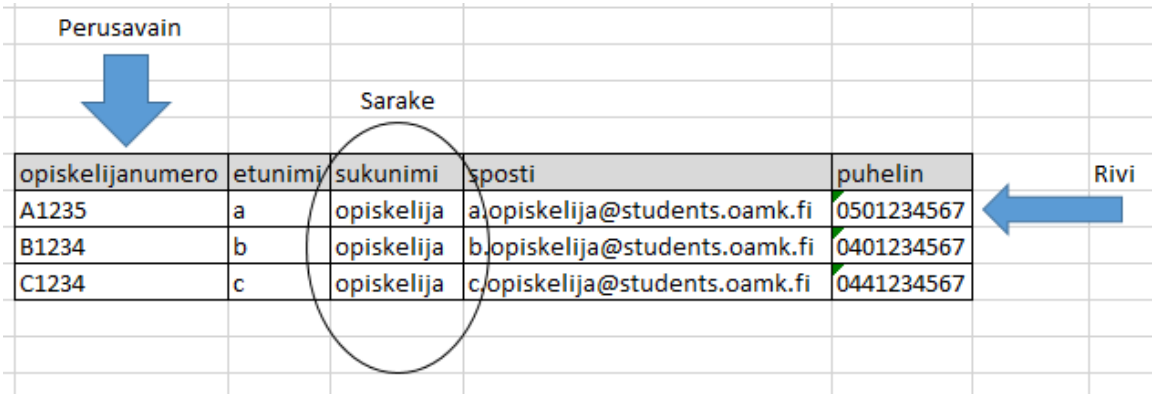
Harjoittelutoiminnon tavoitteena oli saada toimiva, selkeä ja käyttäjäystävällinen ohjelma. Tavoitteena oli myös, että Oulun ammattikorkeakoulu pystyy hyödyntämään harjoittelutoimintoa tulevilla lukukausilla sekä tilastollisessa seurannassa. Tekijän henkilökohtaisena tavoitteena oli lisäksi tehdä hyvä opinnäytetyö, jonka lopputulokseen sekä opinnäytetyöntekijä että toimeksiantaja olisivat olla tyytyväisiä.

2 TIETOKANTA JA SEN SUUNNITTELU

Tietokannalla tarkoitetaan loogisesti yhteenkuuluvien, tallennettujen tietojen joukkoa, jota voidaan käsitellä tietokantakielellä, kuten SQL:llä. Tietokannassa olevia tietoja hallinnoi tietokannan hallintajärjestelmä, esimerkiksi MySQL. Tietokannan hallintajärjestelmän avulla tietokanta voi olla eheä, ajantasainen, yhteiskäyttöinen ja ei-toisteinen. Tietokantoja on eri tyyppisiä, kuten verkkomallisia, hierarkkisia, relaatiotietokantoja ja oliotietokantoja. Suurin osa nykyisistä tietokannan hallintajärjestelmistä ovat SQL -pohjaisia relaatiotietokantoja. Tämä johtunee siitä, että relaatiotietokannat ovat helpompi käyttää ja muuttaa kuin perinteisemmät hierarkkiset ja verkkomalliset tietokannat. (Hovi, Huostari & Lahdenmäki, 2005, 4 – 6.)

2.1 Relaatiomalli

E.F. Coddin julkaisi vuonna 1970 relaatiomallin, joka määrittelee relaatiotietokannan teoreettisen pohjan. Malli perustuu joukko-oppiin, matematiikkaan ja predikaattiin, mutta siinä ei oteta kantaa relaatiokannan fyysiseen toteutustapaan. Relaatiomalli jaetaan kolmeen osaan: rakenteeseen, käsittelyyn ja eheyssääntöihin. (Hovi, Huostari & Lahdenmäki, 2005, 7 – 12.)



opiskelijanumero	etunimi	sukunimi	sposti	puhelin
A1235	a	opiskelija	a.opiskelija@students.oamk.fi	0501234567
B1234	b	opiskelija	b.opiskelija@students.oamk.fi	0401234567
C1234	c	opiskelija	c.opiskelija@students.oamk.fi	0441234567

Kuva 1. Tietokantataulu esimerkki.

Tietokannan peruselementti on taulu (Kuva 1), jossa on sekä sarakkeita että rivejä. Taulun sisällä sarakkeilla on toisistaan poikkeavat nimet ja yksilöllisesti määritetyt tietotyypit. Esimerkkikuvassa puhelin sarakkeeseen on määritetty tietotyypiksi numero ja maksimi pituudeksi 12. Jokaisessa taulussa on tunnisteena perusavain, jonka on oltava yksilöivä eli uniikki. Toisin sanoen sarakkeessa ei voi olla kahdella eri rivillä samaa arvoa. Esimerkkikuvassa perusavaimena toimii opiskelijanumero. Tietokannassa viiteavaimella tarkoitetaan linkitettyä tietoa toiseen tauluun ja viiteavain määritellään sarake kohtaisesti. (Hovi, Huostari & Lahdenmäki, 2005, 7 – 12.)

Relaatiomallissa tietoja käsitellään joukko-opillisesti ja se toteutetaan SQL -kielellä. Taulu muodostuu joukosta rivejä, joihin voi kohdistaa joukko-operaatioita. Esimerkkikuvassa tietoja voidaan hakea haulla ”kaikki rivit, joiden sukunimi on opiskelija”. Relaatiomalli ottaa kantaa tietokannan eheyteen. Eheydellä tarkoitetaan sitä, että tietokannan tiedot ovat oikein, ristiriidattomia ja ne vastaavat reaalimaailmaa. Relaatiomallissa on tiettyjä eheysrajoitteita, kuten avaineheys. Avaineheydellä tarkoitetaan sitä, ettei perusavaimen arvo saa olla tyhjä. Toisin sanoen perusavaimen arvo on pakollinen. (Hovi, Huostari & Lahdenmäki, 2005, 7 – 12.)

2.2 Tietokannan suunnittelu

Tietokannan mallinnus (Database modelling) on tietokannan kuvaamista jollakin kuvaustekniikalla. Mallinnusta laajempi käsite on tietokannan suunnittelu (Database desing), joka käsittää laajan kirjon asioita vaatimusmäärittelystä tietokannan mallinnukseen ja fyysiseen suunnitteluun. Hyvän tietokannan rakenteen keskeisiä ominaisuuksia ovat kattavuus, selkeys ja ymmärrettävyys, muutosjoustavuus, yleiskäyttöisyys, eheys, ohjelmointimukavuus ja tehokkuus. Tietokannan rakenteen tulee olla siis yksinkertainen ja siinä on selkeät tietorakenteet. Samalla siitä löytyy kaikki tarvittavat tiedot sekä yhteydet ja lisäksi se soveltuu eri ympäristöihin (Hovi, Huostari & Lahdenmäki. 2005, 20 – 22.)

Tietokannat voidaan jakaa räätälöityihin operatiivisin järjestelmiin, valmisohjelmistoihin ja tietovarastoihin. Tehdessä räätälöityä järjestelmää on tavoitteena se, että tietokannan rakenne on selkeä ja tarkasti tarpeisiin sovitettu. Taulujen ja tietojen nimet ovat omia termejä ja sarakkeet tarkoittavat pääsääntöisesti yhtä asiaa. Tietokannan muutosjoustavuus on hyvä ja siihen voi lisätä uusia tauluja ja sarakkeita koskematta olevassa oleviin ohjelmiin. Räätälöidyssä järjestelmässä pyritään välttämään taulujen pilkkomista ja yhdistämistä, sillä ne johtavat useasti nykyohjelmien muutoksiin. Tietokannan rakenne pyritään pitämään suhteellisen yksinkertaisena, jotta sen suorituskyky on hyvä (Hovi, Huostari & Lahdenmäki. 2005, 20 – 22.)

Valmisohjelman tietokannan tavoitteena on, että sen rakenne on hyvin yleiskäyttöinen, jotta samaa tietokantarakennetta voidaan monistaa kaikille asiakkaille. Valmisohjelma sovitetaan kuhunkin ympäristöön erilaisilla säädettävillä parametreilla. Valmisohjelmistossa tietokannan rakenne on selkeä järjestelmää ylläpitäville spesialisteille. Sarakkeet nimetään yleisnimillä, jolloin ne voivat tarkoittaa eri asioita eri ympäristöissä. Valmisohjelmistojen suorituskyky voi olla toisinaan heikko, koska tietokannan rakenne on monimutkainen johtuen liitoksista. Tietokannan virittäminen saattaa olla myös vaikeaa, koska ohjelmistoa käytettään eri lailla eri ympäristöissä (Hovi, Huostari & Lahdenmäki. 2005, 20 – 22.)

Tavoitteena rakentaessa tietovarastokantaa on, että sen rakenne tukee helppoja kyselyitä ilman varsinaista ohjelmointia. Näin ollen tietovarastojen tulee olla selkeitä, helppokäyttöisiä ja ymmärrettäviä. Tietovarastot rakennetaan räätälöidysti ja tiedon toisteisuutta ei vältetä, vaan sitä suositaan kyselyjen nopeuttamiseksi. Tietovastoissa säilytetään useasti monen vuoden historiaa, jolloin se vie enemmän levytilaa kuin operatiivisen kannat. (Hovi, Huostari & Lahdenmäki. 2005, 20 – 22.)

3 KÄYTETYT OHJELMOINTIKIELET

3.1 HTML

HTML eli Hypertext Markup Language tehtiin alun perin tieteellisten dokumenttien esittämiseen. HTML: ää kehitettiin alun perin CERNissä ja ensimmäinen kielen julkinen määrittely IETF:n HTML 2.0 laadittiin vuonna 1995. Tässä vaiheessa määrittelystä jäi pois Section -elementti, joka oli ollut mukana varhaisissa suunnitelmissa. Section -elementti tuli kuitenkin määrittelyyn ja selaimiin HTML5:ssä. Ensimmäisen julkisenmäärittelyn jälkeen selainten tekijät jatkoivat omien laajennustensa lisäämistä ja samalla kehitettiin teoreettiselta pohjalta HTML:n uutta versiota. HTML 3.0 jäi kuitenkin luonnos asteelle vuonna 1995. (Korpela. 2014, 28.)

HTML 3.2 julkaistiin vuonna 1997, mutta uusi versio oli enemminkin selaimiin jo tehtyjen laajennusten julkaisemista. HTML 4.0 määriteltiin vuotta myöhemmin ja vaikka se oli lähinnä tosiasiallisten laajennusten julkaisemista, oli siinä lisäksi teoreettisia uutuuksia, kuten object -elementti. Vuonna 1999 julkaistiin HTML 4.01 versio, josta puhutaan vieläkin niin sanottuna ”virallisena HTML:nä”. Edelliseen määrittelyyn verrattuna versiota oli hyvin vähän muokattu, mutta uusi versio on vielä nykyisinkin selainten yleisesti tukema. Klassisen HTML:n kehittäminen lopetettiin vuonna 1998. (Korpela. 2014, 28.)

Ensimmäinen W3C:n (Word Wide Web Consontiun) HTML5 -luonnos julkistettiin vuonna 2008. (sivu29) Uusin HTML -versio HTML 5.1.2 julkaistiin 3 lokakuuta 2017, joka on toinen versio viidennestä pääversiosta. Ensimmäinen versio HTML 5:stä oli HTML 5.1. Lisäksi versio päivityksessä lisättiin, poistettiin ja muutettiin joitakin ominaisuuksia. (WC3 2014, viitattu 29.10.2017.)

```
<!DOCTYPE html>
<html>
  <head>
    <title>Sample page</title>
  </head>
  <body>
    <h1>Sample page</h1>
    <p>This is a <a href="demo.html">simple</a> sample.</p>
    <!-- this is a comment -->
  </body>
</html>
```

Kuva 2. A basic HTML document (WC3 2017, viitattu 29.10.2017)

Esimerkissä (kuva 2) käytetään muutamaa HTML -koodin perustagia. HTML -koodin pohjana on <html> tagi, jonka sisälle tulee kaikki sivustolla näkyviin tulevat HTML -koodit. <head> tagin sisään tulee metadataa, esimerkiksi sivun otsikko <title>, joka tulostaa selaimen välilehteen nimen. Lisäksi <head> tagin sisään tulee hakukoneoptimointi ja jos sivustolle viitataan tyyli- tai kielitiedostoja, niin ne myös lisätään tässä osiossa. <body> tagien sisään tulee sivuston sisältö, joka tässä tapauksessa sisältää otsikon, joka on <h1> tagin sisällä. Lisäksi on tekstiä sekä johon on upotettu linkin. Teksti kirjoitetaan <p> tagin sisään ja linkki tulee <a> tagin sisään.

3.2 PHP

PHP (Hypertext preprocessor) on yleiskäyttöinen ohjelmointikieli, joka on käytännössä erikoistunut www -käyttöön. Alun perin PHP oli kokoelma rutiineja, joilla helpotettiin www -pohjaisten sovellusten tekemistä. PHP on tulkattava kieli eli PHP -koodi ajetaan aina palvelimella juuri ennen, kun se lähetetään selaimelle. Toisin sanoen PHP suoritetaan palvelimen päässä. PHP:n vahvuutena on samankaltaisuus muiden ohjelmointikielien kanssa. Lisäksi PHP:ssä on valmiina suuri määrä eri toimintoja. (Heinosuo & Rauta, 2007, 12 – 13, 26.)

PHP on yksi maailman yleisimmistä avoimeen lähdekoodiin perustuva web -kehityskielistä. PHP -koodi on erittäin sulautuvaa ja siitä löytyy todella paljon valmiita ominaisuuksia, joita monet ammattikoodarit käyttävät jatkuvasti. PHP:tä pystyy sulauttamaan HTML -koodiin. PHP -koodia käytetään yleisesti HTML -sivuilla erilaisten ehtojen luomiseen. (The PHP Group. 2017. viitattu 16.10.2017.)

```
1  <!DOCTYPE HTML>
2  <html>
3      <head>
4          <title>Example</title>
5      </head>
6      <body>
7          <?php
8              if($_POST['moi']){
9                  echo "Moi";
10             }
11             if($_POST['ei']){
12                 echo "Ei";
13             }
14         ?>
15     </body>
16 </html>
```

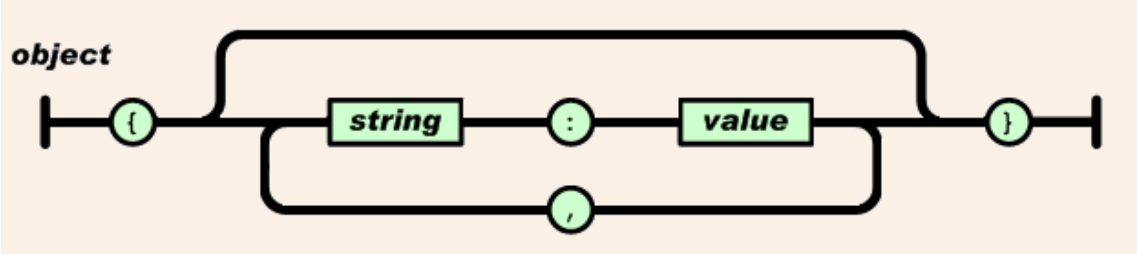
Kuva 3. Perus PHP -koodi esimerkki

Kuvan 3 esimerkissä ollaan sulautettu PHP -koodia HTML -koodin sisälle ja PHP -koodilla tehdään ehto. PHP tarkkailee edelliseltä sivulta lähetettyä dataa: Jos sivustolle tulee ”Moi”, tulostetaan Moi. Jos taas tulee ”ei”, tulostetaan Ei. Nämä ehdot eivät näy millään tavalla sivuston käyttäjälle, koska ehto suoritetaan palvelimen päässä ennen sivun lähettämistä selaimelle.

3.3 JSON

JSON eli JavaScript Object Notation on kevyt tiedonsiirtomuoto. JSON on helppo luettava ja kirjoitettava tekstipohjainen tiedostomuoto. JSON perustuu JavaScript -ohjelmointikielien osa-alueeseen ECMA-262.3, joka julkaistiin joulukuussa 1999. JSON voi sisältää seuraavia tiedostotyyppejä: objekti, lista, numero, merkkijono,

tosi, epätosi tai nolla-arvo. Vaikka JSON pohjautuu JavaScriptiin, niin se on täysin riippumaton siitä. JSON sisältää vain pienen määrän rakenteellisia sääntöjä tietojen esitykselle. (JSON. 2017 viitattu 24.10.2017) (Ecma International. 2013. viitattu 24.10.2017).



Kuva 4. JSON object (JSON 2017. viitattu 24.10.2017)

Objekti on järjestelemätön listaus nimiä ja niiden arvopareja. Objekti alkaa vasemmalta aaltosulkujen sisältä ja päättyy oikealla aaltosulkuihin (kuva 4). Jokaista nimeä seuraa kaksoispiste, jonka jälkeen tulee nimi ja arvopari, jotka erotetaan pilkulla.

```
{
   opiskelija : {
      perustiedot {

         etunimi : "Jani",
         sukunimi : "Mustonen"

               },
         ryhma : {

               tunnus : "TIK3KA",
               opintoala : "Tietojenkäsittely"

            }
      }
}
```

Kuva 5. Esimerkki JSON -tiedoston sisällöstä

Esimerkkikuvassa 5 on JSON -tiedosto, joka sisältää opiskelija objectin. Objectin sisällä on listattu tietoja etunimi ja sukunimi. Lisäksi listalta löytyy toinen objecti “ryhma”, jonka sisällä on toinen lista, josta löytyy tunnus ja opintoala.

3.4 MySQL

MySQL on relaatiotietokantoja käsittelevä tietokannanhallintajärjestelmä. MySQL oli alun perin tarkoitettu ruotsalaisen MySQL AB konsultointiyrityksen sisäiseen käyttöön (Heinosuo & Rauta. 2007, 37 – 38.) MySQL nimestä SQL on lyhenne sanoista ”Structured Query Language”. SQL on yleisesti käytetty ja myös standardoitu kieli, jota tietokannat ymmärtävät. MySQL lyhenteestä osa ”My” tulee perustajan Monty Wideniuksen tyttären nimen mukaan. (Oracle corporation. 2017. viitattu 29.10.2017).

Tietokanta on jäsennelty tietojoukko, joka voi olla yksinkertaisesta ostoslistasta ison yrityksen valtavaan tietokantaan. Jotta pystytään hallinnoimaan tietokannan tietoja, tarvitaan tietokannan hallintajärjestelmä, kuten MySQL Server. MySQL Server on yleensä jollain keskeisellä palvelimella ja toiset ohjelman käyttävät MySQL -tietokantaa tallentaakseen tarvittavia tietoja. MySQL tietokantaan pystyy tallentamaan ainakin 200 000 eri taulua ja 5 miljardia riviä tietoa. Yksi MySQL tietokannan taulu tukee yhteen 64 indeksiä. Jokainen indeksi voi sisältää yhteensä 16 saraketta tai sarakkeen osaa. MySQL -tietokannan taulu tukee viittatoista eri tiedostotyyppiä ja tietokannan rivien tietoihin voi vaikuttaa neljällä erikomennolla: DELETE poistaa tietokannasta halutun rivin ja sen tiedot, INSERT lisää tietokantaan rivin tietoa, REPLACE korvaa tietokannassa olevan rivin tiedot ja UPDATE päivittää tietokannassa olevan rivin jonkin kentän tietoa tai vaikka koko rivin. (Oracle corporation. 2017. viitattu 29.10.2017).

```
SELECT * FROM harjoittelu
    WHERE sukunimi = 'Mustonen'
    AND etunimi = 'Jani'
    AND ryhma = 'TIK3KA'
```

Kuva 6. Perus SQL -haku

SQL:n esimerkissä (kuva 6) on perus SQL -haku, joka hakee tietyn opiskelijan tiedot taulusta. Haussa on kumminkin muutamia ehtoja, jotka pitää täyttyä muun muassa opiskelijan sukunimi pitää olla ”Mustonen”, etunimi pitää olla ”Jani” ja opiskelijan ryhmä pitää olla ”TIK3KA”. Haku palauttaa kyseisen opiskelijan kaikki

tiedot, jos haun kohdetaulusta löytyy kyseinen opiskelija haussa annetuilla ehdoilla.

3.5 CSS

CSS eli Cascading Style Sheets on Web-sivuston merkintäjärjestelmän määritystiedosto. CSS -tiedosto antaa Web -sivustolle muotoiluohjeet, jotta sivusto käyttää selaimet oletuksien tilalta. Selainten oletustyylejä pidetään yleisesti huonoina typografisesti. Yleisesti selainten oletusfonttina on Times New Roman, joka ei sovi kuvaruudulta luettavaksi. Lisäksi sen esittelyasu Times New Romanissa on yleensä huono. CSS -tyylitiedosto sisältää eräänlaisen listan HTML tagien tyyleistä, joille annetaan tyylimäärittelyt. Tyylitiedosto pitää ladata jokaisen sivun alussa olevassa <meta> tagin sisälle, jotta tyylit tulevat käyttöön sivustolla. Tyylitiedosto ladataan <link> tagillä, jolla sivusto linkittyy tyylitiedostoon ja lataa sen jokaisen sivunlatauskerran alussa (WC3 2017, viitattu 7.11.2017.)

3.6 Bootstrap

Bootstrap luotiin Twitterissä vuoden 2010 puolessa välissä ennen avoimen lähdekoodin luomista. Bootstrap tunnetaan myös nimellä Twitter Blueprint. Muutaman kuukauden kehitystyön jälkeen Twitter järjesti Hack-viikon, jossa Bootstrap hanke kasvoi räjähdysmäisesti, koska kaikki osaamistason kehittäjät hyppäsivät mukaan ilman ulkoista ohjaamista (Bootstrap-verkkosivu 2017, viitattu 9.11.2017.)

Bootstrap julistettiin 19.8.2011 ja sen jälkeen on ollut yli 20 julkaisua, mukaan lukien kaksi suurta uudelleenkirjoitusta v2 ja v3. Nyt on kehitteillä v4, josta on julkaistu beta 2 versio. Bootstrap 3.3.7 on tällä hetkellä viimeisin stabiili versio, joka on myös kirjoitettu siinä mielessä, että se tukee mobiilia (Bootstrap-verkkosivu 2017, viitattu 9.11.2017.)

3.7 PL/SQL

PL/SQL on yhdistelmä SQL ja ohjelmointikielen prosessuaalisia ominaisuuksia. Se kehitettiin 1990-luvun alussa Oracle Corporationin toimesta parantaakseen SQL:n ominaisuuksia. PL/SQL on yksi kolmesta kielestä, jotka ovat upotettu Oracle -tietokantaohjelmaan. PL/SQL:n lisäksi Oracle -tietokanta sisältää SQL ja Java -ohjelmointikieliä (Oracle Corporation 2017, viitattu 9.11.2017.)

```
DECLARE
   -- variable declaration
   message  varchar2(20):= 'Hello, World!';
BEGIN
   /*
   *  PL/SQL executable statement(s)
   */
   dbms_output.put_line(message);
END;
/
```

Kuva 7. PL/SQL esimerkki

PL/SQL esimerkkikuvassa (kuva 7) on ohjelmointikielen peruskomento ja kommentointitapoja. PL/SQL -kielessä pitää aina julistaa muuttujat komennolla Declare ja aloittaa ohjelma komennolla Begin. Esimerkissä julistetaan ”message” muuttuja, jonka maksimi pituudeksi annetaan 20 varchar2 merkkiä. Varchar2 on tietotyyppi, joka tarkoittaa merkkijonoa ja joka voi sisältää mitä vain merkkejä. Message muuttujan arvoksi annetaan ”Hello World!”. Muuttujien julistamisen jälkeen aloitetaan koodin. Esimerkissä on käytetty kahta eri kommentointi menetelmää: perus yhden rivin kommentointitapa sekä /* */ (keno tähti ja tähti keno), jonka väliin voidaan kirjoittaa useita rivejä kommenttia. Seuraavaksi esimerkissä on koodi bms_output.put_line(message), joka mahdollistaa message muuttujan tulostamisen käyttöliittymään ja koodin lopuksi on END, joka kertoo ohjelmalle, missä koodi loppuu (Oracle Corporation 2017, viitattu 9.11.2017.)

3.8 JavaScript

JavaScript on dynaaminen ohjelmointikieli, joka soveltuu HTML -dokumentointiin ja voi tarjota dynaamista vuorovaikutteisuutta verkkosivustoilla. JavaScript tai lyhemmin tunnettu JS tunnettiin ensin nimellä LiveScript. Vuonna 1995 Netscape aloitti Brenda Eich kehittämän ohjelmiston LiveScript kehittämisen, joka haluttiin yhdistää LiveWire-palvelinpuolen kanssa. Tavoitteena oli tehdä tehokas online -työkalu. Netscape aloitti yhteistyön Sun Micosystemsin kanssa toteuttaakseen yhdistämisen. Yhdistämisen jälkeen LiveScript nimi muuttui JavaScriptiksi. (Brenkoweb 2017, viitattu 9.11.2017.)

JavaScript on monipuolinen ohjelmointikieli, joka voi laittaa selaimen sivustot kuvat vaihtelemaan tai karusellit pyörimään. Lisäksi JavaScriptiä voi käyttää pelien ja animoitujen 2 ja 3D grafiikkaan. JavaScript on itsestään melko kompakti, mutta erittäin joustava ohjelmointikieli. Se kehittäjät ovat tehneet laajan valikoina erilaisia työkaluja sen päälle. Tämä antaa kehittäjille valtavan määrän erilaisia ohjelmia käyttöön pienellä vaivalla (Brenkoweb 2017, viitattu 9.11.2017.)

3.9 Lightweight Directory Access Protocol

LDAP tai Lightweight Directory Access Protocol on Internet-protokola, jota sähköpostit ja muut ohjelmat käyttävät etsiessään tietoa palvelimelta (GracionSoftware 2017, viitattu 10.11.2017). Ensimmäinen versio LDAP tuli vuonna heinäkuussa 1993 RFC 1487, jonka julkaisi Internet Engineering Task Force (IETF). Tämän jälkeen IETF on julkaissut 13 LDAPiin liittyvää RFC:tä vuoteen 1997 mennessä ja senkin jälkeen julkaisuja on ollut useita. Uusimpana kesäkuussa 2006 julkaistu RFC 4511 (LinuxJuornal 2017, viitattu 10.11.2017).

LDAP on lähinnä keskisuurien ja suurten organisaatioiden käytössä. Organisaatioilla, joilla LDAP on yleensä käytössä oma LDAP-palvelin, jonka kautta pystyy etsimään esimerkiksi yhteystietoja. LDAP ei vain rajoitu yhteystietojen etsimiseen, vaan sitä voidaan käyttää salausvarmenteiden, tulostimien ja muiden verkossa olevien palveluiden osoittamiseen ja ”yhden kirjautumisen” luomiseen,

jossa käyttäjän salasana jaetaan useiden palveluiden kesken (GracionSoftware 2017, viitattu 10.11.2017.)

4 HARJOITTELUTOIMINTO

Harjoittelutoiminnon tekeminen aloitettiin kesäkuussa 2017. Projektin aluksi luotiin toimeksiantajan teknisen ohjaajan Ville Valtosen kanssa tietokantahahmotelma siitä, mitä tietoja harjoittelutoiminnon täytyy pitää sisällään. Tämän jälkeen suunnittelua jatkettiin käyttöliittymän suunnittelulla, joka tehtiin alustavasti paperille. Nopeasti kuitenkin siirryttiin käyttämään HTML -koodia ja selainta. Projekti eteni tämän jälkeen nopeammin, kun oli alustavat tietokantakuvat ja suunnitelma siitä, minkä näköinen sivustosta tulee. Toki projektin etenemisen aikana suunnitelmat menivät melkein kokonaan uusiksi ja sivusto kasvoi paljon aluksi suunniteltua isommaksi. Harjoittelutoiminto valmistui kuitenkin ajallaan elokuun 2017 alkupuolella.

Harjoittelutoiminnon tarkoituksena on helpottaa harjoittelusuunnitelmien ja raporttien käsittelyä sekä yhtenäistää käytäntöjä. Lisäksi harjoittelutoiminnon tarkoituksena on mahdollistaa työharjoitteluiden tilastollinen seuranta Oulun ammattikorkeakoulussa, esim. tilastoseuranta siitä, missä ja milloin opiskelijoita on ollut työharjoittelussa.

4.1 Tietokantasuunnittelu

Tietokanta on suunniteltu relaatiomallin mukaisesti, jotta siitä löytyviä tauluja voi tarvittaessa käyttää toisessakin ohjelmassa. Niiden taulujen, joita ohjelma käyttää opiskelijatietojen tallentamiseen tai tietojen hakemiseen, perusavaimena toimii opiskelijanumero, joka yksilöi jokaisen opiskelijan. Tietokanta on tehty räätälöidysti kyseisen ohjelman tarpeiden mukaisesti, jotta sen suorituskyky pysyy hyvänä ja rakenne selkeänä (tietokantakaavio LIITE 1). Lisäksi tietokannan rakenne tukee muutosjoustavuutta ja muita ohjelmistoja. Tietokanta on tehnyt MySQL:llä ja se toimii koulun sisäisellä Linux -pohjaisella palvelimella, johon on rajattu pääsy. Tietokannan tyypiksi valittiin räätälöity versio, koska tietokannasta halut-

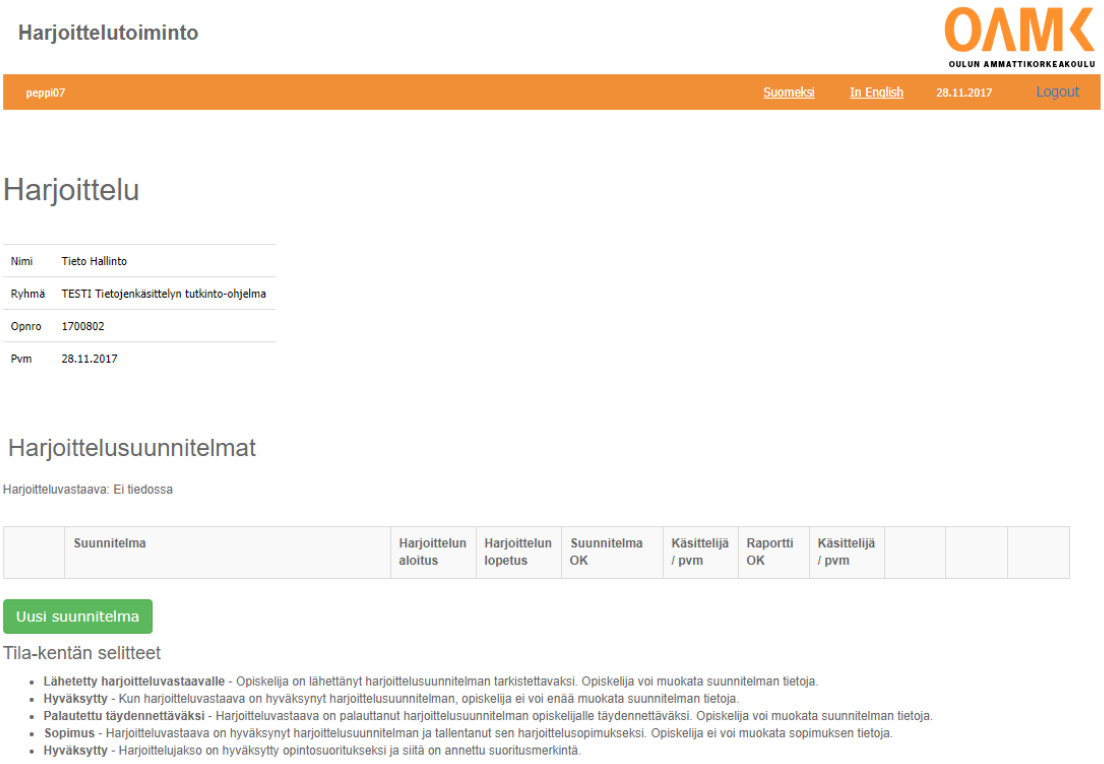
tiin juuri tietynlainen, eikä valmisohjelman hankintaan ollut erikseen varattuja resursseja. Lisäksi itse tehty tietokanta on helposti muokattavissa ja jatkokehittämistä ajatellen on helpompi käyttää itse tehtyä tietokantaa.

Tietokannan suunnitteluvaiheessa ei tehty kirjallista vaatimusmäärittelyä, koska opinnäytetyöstä olisi tullut liian laaja. Lisäksi ohjelman esikoisversio piti saada erittäin nopealla aikataululla valmiiksi toimeksiantajan tarpeiden mukaisesti, joten vaatimusmäärittelyn tekemiseen ei käytetty resursseja tässä vaiheessa. Ohjelmiston teko aloitettiin kesälomakauden aikana, jolloin suurin osa loppukäyttäjistä oli lomalla, joten vaatimusmäärittelyn tekeminen olisi ollut melkein mahdotonta kyseisenä aikana.

4.2 Harjoittelutoimintoprosessin yleiskuvaus

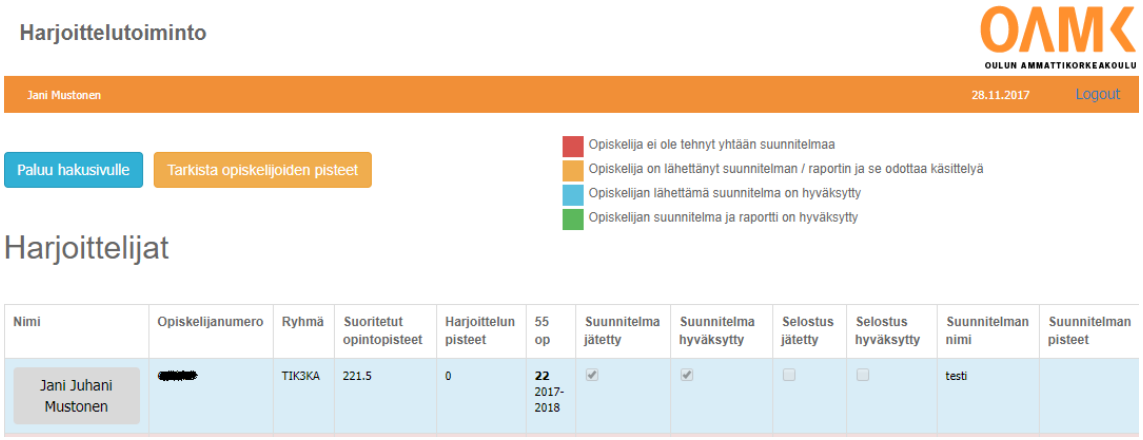
Harjoittelutoiminnosta löytyy erilaiset näkymät opettajille ja opiskelijoille. Harjoittelutoiminto on ohjelma, jonka avulla opiskelija tekee harjoittelusuunnitelmia ja raportteja, jotka opettaja tarkistaa. Oppilas voi tehdä suunnitelmia ennakkoon tai vasta harjoittelun jälkeen. Opettaja käsittelee oppilaan tekemät suunnitelmat ja raportit. Raportin opiskelija voi tehdä vain hyväksyttyyn harjoittelusuunnitelmaan (Kuva 8).

Harjoittelutoiminnon prosessi lähtee liikkeelle, kun opiskelija kirjautuu ja lähettää harjoittelusuunnitelman. Kun opiskelija lähettää suunnitelman eteenpäin tai vain tallentaa suunnitelman myöhempää muokkaamista varten, hänen tietonsa tallentuvat tietokantaan. Kun opiskelija on lähettänyt suunnitelman, se menee lukittuun tilaan eikä opiskelija voi muokata sitä.



Kuva 8. Opiskelijanäkymä

Opettaja hakee opiskelijan joko nimellä tai luokan perusteella (Kuva8). Luokkahaku tuotetaan näkymässä, jossa on kaikki luokalla läsnäolevat opiskelijat. Luokkanäkymään tulee opettajille tarpeellisia tietoja, kuten opiskelijoiden opintojen etenemisestä. Opettajat näkevät näkymässä opiskelijoiden koko opintopistekertymän ja kuluvan lukukauden opintopistekertymän. Nimellä haetun opiskelijan kautta opettaja ei näe suoraan opiskelijan pistekertymiä, mutta opettaja pääsee opiskelijan tietojen kautta kätevästi luokkanäkymään, josta löytyy pistekertymät (Kuva 9).



Kuva 9. Ryhmänäkymä

Opettaja näkee luokkanäkymässä, jos joku opiskelija on lähettänyt suunnitelman. Opettaja pääsee lukemaan suunnitelman, hyväksymään sen tai palauttaa sen opiskelijalle. Jos opettaja palauttaa suunnitelman opiskelijalle, palautuu se opiskelijalle muokkaustilaan. Opettaja voi tarvittaessa palauttaa laittaa viestiä opiskelijalle, miksi suunnitelma/raportti on palautettu. Jos opettaja hyväksyy suunnitelman, opiskelijalle tulee näkyviin hyväksyntä merkki, päivä ja kuka hyväksynnän teki.

Tämän jälkeen opiskelija voi alkaa tekemään suunnitelman mukaisesti suoritetun harjoittelun raporttia. Raportin tekeminen toimii samalla tavalla kuin suunnitelman, mutta poikkeuksena on, että raporttiin opiskelija voi lisätä liitteitä, joita harjoittelun suorittamiseen tarvitaan. Opettaja voi toimia rapotin kohdalla samalla tavalla kuin suunnitelman: kun opettaja hyväksyy raportin, tarkoittaa tämä, että yksi harjoittelu on suoritettu suunnitelman mukaisesti ja se on hyväksytysti läpi.

Hyväksyttyihin suunnitelmiin pystyy kohdistamaan joko toteutuksen osia tai koko toteutuksia. Jotta opettaja pystyy kohdistamaan suunnitelmille pisteitä, täytyy opiskelijalla olla harjoitteluopintojakso Pepissä, josta harjoittelutoiminto ottaa toteutuksen ja sen osat. Tämä tieto ei mene Peppiin, vaan se jää järjestelmän sisälle ja auttaa opettajaa näkemään, kuinka paljon opiskelijalle on työharjoittelusta annettu opintopisteitä. Pisteet tulevat näkyviin luokkanäkymässä jokaisen suunnitelman kohdalle ja kertymä tulee näkymään opiskelija kohtaisesti.

Raportin hyväksyntä myös lukitsee kyseisen harjoitteluraportin ja sen suunnitelman niin, että opettaja ei voi enää palauttaa tai kohdistaa siihen pisteitä. Lukko pystytään tarvittaessa avaamaan, jos harjoittelun pisteytyksessä on tullut jokin virhe (Prosessikaavio LIITE 2).

4.3 Käyttöoikeudet

Harjoittelutoiminnossa on sisäinen lista, millä oikeudella siihen pääse kirjautumaan. Käyttäjän käyttöoikeudet tarkistetaan kirjautumisen yhteydessä Pepistä.

Pepissä jokainen käyttäjä kuuluu johonkin käyttöoikeusryhmään, jota harjoittelutoiminto katsoo. Vain osalla henkilökuntaa on pääsy harjoittelutoimintoon ja tietokantaan. Käyttöoikeudet tarkastetaan Pepistä rajapintatyökalua käyttäen. Rajapinta käyttää käyttäjätunnusta roolikyselyyn Pepistä: jos henkilöltä löytyy oikea rooli, hän pääsee kirjautumaan sisään. Tietokantaan pääsee vain tällä hetkellä Oamk:n tietohallinnosta vain muutama henkilö.

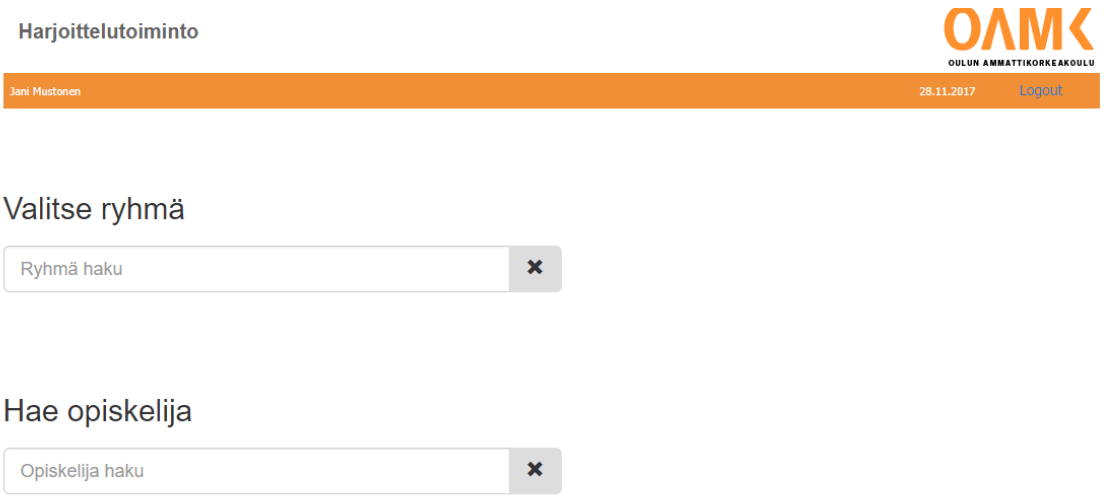
4.4 Opiskelijan näkymä

Opiskelija kirjautuu Peppiin ja jos opiskelijalta löytyy useampi voimassa oleva opinto-oikeus, ohjataan opiskelija opinto-oikeuden valintasivulle. Opintojen valinta sivulla tulee lista opiskelijan opinto-oikeuksista ja opiskelijan tulee valita, mitä opinto-oikeutta käytetään. Opiskelija kirjautuu sisään harjoittelutoimintoon ja tekee harjoittelusuunnitelman, johon opiskelija täyttää työnantajan yhteystiedot ja harjoittelun ajankohdan. Lisäksi opiskelijan tulee täyttää suunnitelmaan omat työtehtävät sekä harjoittelun tavoitteet. Opiskelija voi tallentaa suunnitelman itselleen myöhempää muokkaamista varten tai lähettää suunnitelman opettajalla hyväksyttäväksi. Suunnitelma tulee näkyviin opiskelijan aloitussivulle, josta opiskelija voi seurata suunnitelman käsittelyn etenemistä. Kun opiskelija on suorittanut harjoittelun suunnitelman mukaisesti, hän voi tehdä kyseiseen harjoitteluun raportin, jonka hän tekee samalla tavalla kuin suunnitelman.

4.5 Opettajan näkymä

Opettaja kirjautuu järjestelmään perusrekisterin kautta tai suoraa linkkiä käyttämällä. Kun opettaja on kirjautunut ohjelmaan, hänen ensimmäinen tehtävä on hakea opiskelija. Opettaja voi hakea yksittäisen opiskelijan nimellä tai koko opiskeluryhmän. Ryhmä-näkymässä opettaja saa ryhmästä tehdyn taulukon, josta hän pystyy tarkistamaan opiskelijoiden kokonaispistemäärän ja seuraamaan opiskelijan lukukauden opintopistekertymää. Lisäksi opettaja näkee, onko joku opiskelija tehnyt suunnitelmia ja vaatiko tämä opettajalta toimenpiteitä. Opettajan ei voi käsitellä opiskelijan suunnitelmia ryhmätasolla, vaan hänen pitää mennä

aina jokaisen opiskelijan tietoihin ja lukea suunnitelma / raportti läpi, jotta hän voi hyväksyä tai palauttaa sen (Kuva 10).



Kuva 10. Opiskelijanhaku näkymä

4.6 Harjoittelutoiminnon tekninen toteutus

Asiossa ollut harjoittelutoiminto oli tehty PL/SQL -kielellä ja se pyöri Oraclen palvelimella. Uusi perusrekisteri Peppi on tehty Javalla eikä se tue PL/SQL -kantaa. Tästä syystä uusi harjoittelutoiminto on tehty pääosin HTML / PHP -kielillä ja taustalla toimii MySQL -kanta. Lisäksi toiminnossa on käytetty JavaScriptiä sekä ulkoasussa käytettiin valmiita Bootstrap 3.3.7 kirjastosta löytyviä classeja ja värejä.

Taustalla käytettävää MySQL -tietokantaa kutsutaan PHP -funktioilla. Tietojen haku toimii joko SQL -kyselynä tai JSON -paketti hakuina, joita käsitellään PHP -funktioilla. SQL -kyselyt menevät harjoittelutoiminnon omaan tietokantaan ja JSON -pakettien kysely menee Peppi -rajapintaan, joten tiedot tulevat täten Pepistä. Käyttöliittymä on tehty Ville Valtosen tekemän sivupohjan päälle. Pohja sisältää Oamk -sivun koko- ja värimäärityksset. Lisäksi ohjelmassa on käytetty muutamaa Valtosen tekemään valmista PHP Class -muuttujaa, jotka toimivat rajapintojen kanssa sekä kirjautumisessa.

Kirjautuminen harjoittelutoimintoon tapahtuu Pepin kautta tai suoraan linkillä. Tietoturvallisesti kirjautuminen on turvallista, koska ensimmäinen vaihe on LDAP kirjautuminen, joka katsoo, että käyttäjä on olemassa ja että käyttäjänimi ja salasana ovat oikein. Tämän jälkeen itse harjoittelutoiminto varmistaa Pepistä, että kyseistä käyttäjätunnusta käyttävä käyttäjä löytyy Pepistä ja että tunnukset ovat voimassa. Jos nämä asiat ovat kunnossa, harjoittelutoiminto ohjaa opiskelijan eteenpäin. Opettajien käyttäjätunnuksesta tarkistetaan, että opettajalta löytyy tiedoista oikea rooli eli opettajan tai pääkäyttäjän rooli, jolloin kirjautuminen sallitaan

4.7 Tietoturva

Harjoittelutoiminto toimii vain Oamk:n sisäisellä palvelimella, johon pääsee käsiksi vai Oamk:n sisäverkosta tai VPN -yhteydellä. Sisäänkirjautuminen toimii LDAP:llä (Lightweight Directory Access Protocol), joka on yleisesti käytössä Oamk:n järjestelmissä. Sisäänkirjautumiseen käytetään koulun AD -tunnuksia, jotka jokainen opiskelija saa aloitettuaan koulun.

Kirjautuminen ohjelmaan tapahtuu perusrekisterin kautta tai suoraan linkillä. Jos ohjelmaan mennään perusrekisterin kautta, siihen ei erikseen tarvitse kirjautua, koska käytössä on single sign in -toiminto. Kyseinen toiminto tallentaa käyttäjän tunnuksen ja salasanan sessiomuuttujaan ja käyttää sitä kirjautumiseen. Tämä toiminto on käytössä kaikissa Peppiä varten uudistetuissa ohjelmistoissa.

4.8 Rajapinta perusrekisteriin

Rajanpinnalla tarkoitetaan yhteyttä harjoittelutoiminnon ulkopuoliseen ohjelmaan. Tässä tapauksessa harjoittelutoiminon ulkopuolinen ohjelma on Peppi. Ulkopuolisesta ohjelmasta haetaan tietoja, joita käytetään tässä harjoittelutoiminnossa. Rajapinta toimii JSON -tiedostopaketti hakuina. Rajapintakyselyitä varten on Valtonen tehnyt oman työkalun, joka hakee perusrekisteristä raportteja annettujen tietojen perusteella. Rajapinta ei palauta mitään sensitiivistä tietoa opiskelijoista tai henkilökunnasta. Tieto, jota se palauttaa, käsitellään palvelimen päässä niin, että vain haluttu tieto näkyy käyttäjälle.

5 POHDINTA

Tämän toiminnallisen opinnäytetyön tarkoituksena oli tehdä harjoittelutoiminto Oulun ammattikorkeakoulun opiskelijoille ja opettajille. Uusi harjoittelutoiminto luotiin vanhan Asiossa olleen toiminnon pohjalta ja samalla siihen tehtiin parannuksia. Harjoittelutoiminto ohjelmointiin PHP ja HTML -kielillä ja tietokantana on MySQL. Ohjelma rakennetiin valmiiksi tehdyn sivustopohjan päälle, jonka on luonut Oulun ammattikorkeakoulun tietohallinnossa työskentelevä pääsuunnittelija Ville Valtonen. Lisäksi harjoittelutoiminnossa hyödyntiin Valtosen tekemiä PHP Classeja mm. ohjelmaan kirjautumisessa ja lokitietojen arkistoinnissa.

Harjoittelutoiminnon tekovaiheessa ei oltu tehty vaatimusmäärittelyä, koska ohjelmasta haluttiin mahdollisimman nopeasti ensimmäinen versio opettajille testattavaksi, joten vaatimusmäärittelyyn ei käytetty resursseja projektin alussa. Lisäksi vaatimusmäärittely on rajattu pois opinnäytetyö alueesta, koska opinnäytetyöstä olisi tullut muuten liian laaja. Vaatimusmäärittely tehdään toiminnon ensimmäisen version jälkeen, jolloin toiminnon toisessa versiossa otetaan huomioon vaatimusmäärittelyn tulokset. Mikäli vaatimusmäärittely olisi tehty projektin alussa, olisi harjoittelutoiminto ollut huomattavasti helpompi tehdä, koska tällöin olisi ollut tiedossa kaikki toimintoon halutut ominaisuudet. Lisäksi toiminnon jatkokehittäminen olisi helpompaa, kun sen perusta olisi ollut valmiiksi jo todella hyvä.

Harjoittelutoiminnon ensimmäisen version tavoitteellinen valmistuminen oli 1.8.2017 ja se tavoite onnistui. Ohjelman lopullisen valmistumisen päivämäärää ei ole vielä tiedossa. Vaatimusmäärittelyn puuttuessa ei oikeasti tiedetty, mitä ohjelmalta halutaan, joten seuraavan version valmistumisen päivämäärä arvio on 1.1.2018, jolloin pyritään saamaan selkeät tavoitteet lopulliselle ohjelmaversiolle.

Tekemäni harjoittelutoiminto oli tarkoitus laittaa Peppiin niin, että oppilaan pääsevät kirjautumaan ja käyttämään sitä. Tämä ajatus ei ole toteutunut tämän opinnäytetyön valmistuessa, koska toimintoon halutaan tehdä vielä muutoksia vaatimusmäärittelyn myötä. Lisäksi toiminnossa olevat ylimääräiset linkit, jotka eivät toimi, aiheuttaisivat vain hämmennystä oppilaiden keskuudessa. Ohjelma tullaan laittamaan Peppiin, kun se on siinä vaiheessa, että oppilaat saavat alkaa käyttämään sitä. Toiminnon jatkokehittäminen tulee olemaan erittäin pitkä prosessi, koska toiminto halutaan käyttöön kaikille Oulun ammattikorkeakoulun yksiköille ja koska ohjelmalla pyritään yhtenäistämään ammattiharjoittelun käytäntöä.

Omasta mielestäni toiminnon ensimmäinen versio tehtiin liian nopealla aikataululla ja liian pienillä alkutiedoilla. Mikäli aikaa olisi ollut enemmän, olisi toiminnosta saatu heti kerralla kattavampi. Mielestäni toiminnon ensimmäisestä versiosta tuli kuitenkin hyvä ottaen huomioon se, etten itse ollut koskaan tehnyt ohjelmaa alusta loppuun enkä käyttänyt PHP- kieltä tässä mittakaavassa.

Opin opinnäytetyön aikana paljon uutta IT -alalla työskentelystä ja muun muassa sen, että aina kaikki ei mene niin kuin suunnitellaan. Suurin oppimisen määrä tuli kuitenkin PHP -kielestä, koska en ollut koskaan aikaisemmin käyttänyt PHP -kieltä tai edes kirjoittanut sitä yhtä komentoa enempää. Suurin osan opinnäytetyöstäni tehtiin PHP kielellä, joten oppimääräni kielestä on kasvanut todella paljon. Lisäksi käytin opinnäytetyössä paljon MySQL:tä ja siitäkin opin paljon uusia ominaisuuksia ja kehityin huomattavasti sen käytössä. Työn alkupuolella minun täytyi myös opetella jonkin verran PL/SQL -kieltä, koska ns. vanha harjoittelutoiminto oli tehty PL/SQL.

Opinnäytetyön ohessa sain osallistua useisiin palavereihin liittyen opinnäytetyöhöni. Myös oman työtavan löytäminen oli alussa haasteellista, koska en ollut koskaan tehnyt kokopäiväisesti istumatyötä avokonttorissa. Haasteita herätti myös oman kalenterin järjestäminen niin, että sain tehtyä kaikki työt ajallaan. Opinnäytetyötä tehdessäni oppimistani asioista on varmasti paljon hyötyä tulevaisuudessa myös työelämässä.

LÄHTEET

Bootstrap-verkkosivu, http://getbootstrap.com/docs/2.2/about/, viitattu 9.11.2017

Brenkoweb. 2017. History of JavaScript. Viitattu 9.11.2017, http://www.brenkoweb.com/articles/world-wide-web/web-design-software/history-of-javascript

Ecma International. 2013. The JSON Data Interchange Format. Viitattu 24.10.2017, http://www.ecma-international.org/publications/files/ECMA-T/ECMA-404.pdf

Eduix 2017. Peppi. Viitattu 23.7.2017, http://www.eduix.fi/peppi

GracionSoftware 2017. What is LDAP. Viitattu 10.11.2017, http://www.gracion.com/server/whatldap.html

Heinosuo, R & Rauta, I. 2007. PHP ja MySQL Tietokantapohjaiset verkkopalvelut. Helsinki: Talentum

Hovi, A & Huostari, J & Lahdenmäki, T. 2005. Tietokantojen suunnittelu & indeksointi. Porvoo: WS Bookwell

JSON. 2017, Introducing JSON. Viitattu 24.10.2017, http://www.json.org/

Korpela, J. 2014. HTML5- käsikirja. Saarijärven Offset Oy

Linux Journal, LDAP Series Part III - The Historical Secrets. Viitattu 11.10.2017, http://www.linuxjournal.com/content/ldap-series-part-iii-historical-secrets

Metropolia 2015. Milloin Perusrekisteri ja Pakki otetaan käyttöön?. Viitattu 23.7.2017, https://wiki.metropolia.fi/pages/viewpage.action?pageId=132383907

OAMK 2017. Tietoa Oamkista. Viitattu 14.10.2017, http://www.oamk.fi/fi/tietoa-oamkista/oamkin-esittely/#strategia

Oracle corporation. 2017. What is MySQL. Viitattu 29.10.2017, https://dev.mysql.com/doc/refman/5.7/en/what-is-mysql.html

Oracle corporation. 2017. What is MySQL. Viitattu 29.10.2017, https://dev.mysql.com/doc/refman/5.7/en/history.html

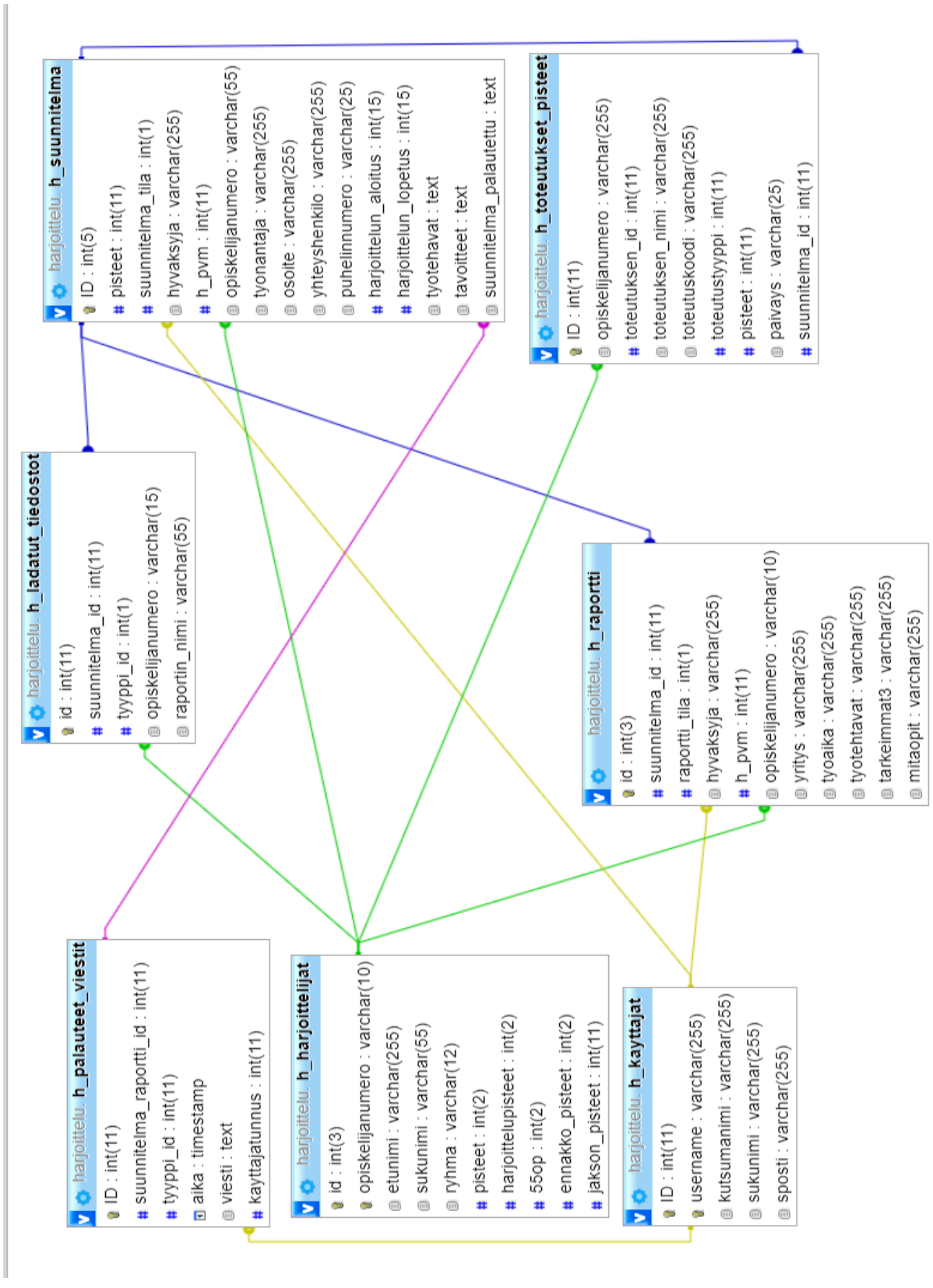
Oracle corporation. 2017. PL/SQL - Basic Syntax. Viitattu 9.11.2017, https://www.tutorialspoint.com/plsql/plsql_basic_syntax.htm

Oracle corporation. 2017. PL/SQL Tutorial. Viitattu 9.11.2017, https://www.tutorialspoint.com/plsql/index.htm

The PHP Group. 2017. What is PHP. Viitattu 16.10.2017, http://php.net/manual/en/intro-whatis.php

WC3 2017, CSS Spanshot 2017. Viitattu 7.11.2017, https://www.w3.org/TR/css-2017/

WC3 2014, HTML5. Viitattu 29.10.2017, https://www.w3.org/TR/html5/


harjoittelu. h_suunnitelma
ID : int(5)
pisteet : int(11)
suunnitelma_tila : int(1)
hyvaksyja : varchar(255)
h_pvm : int(11)
opiskelijanumero : varchar(55)
tyonantaja : varchar(255)
osoite : varchar(255)
yhteyshenkilo : varchar(255)
puhelinnumero : varchar(25)
harjoittelun_aloitus : int(15)
harjoittelun_lopetus : int(15)
tyotehavat : text
tavoitteet : text
suunnitelma_palautettu : text
harjoittelu. h_toteutukset_pisteet
ID : int(11)
opiskelijanumero : varchar(255)
toteutuksen_id : int(11)
toteutuksen_nimi : varchar(255)
toteutuskoodi : varchar(255)
toteutustyyppi : int(11)
pisteet : int(11)
paivays : varchar(25)
suunnitelma_id : int(11)
harjoittelu. h_ladatut_tiedostot
id : int(11)
suunnitelma_id : int(11)
tyyppi_id : int(1)
opiskelijanumero : varchar(15)
raportin_nimi : varchar(55)
harjoittelu. h_raportti
id : int(3)
suunnitelma_id : int(11)
raportti_tila : int(1)
hyvaksyja : varchar(255)
h_pvm : int(11)
opiskelijanumero : varchar(10)
yritys : varchar(255)
tyoaika : varchar(255)
tyotehtavat : varchar(255)
tarkeimmat3 : varchar(255)
mitaopit : varchar(255)
harjoittelu. h_palauteet_viestit
ID : int(11)
suunnitelma_raportti_id : int(11)
tyyppi_id : int(11)
aika : timestamp
viesti : text
kayttajatunnus : int(11)
harjoittelu. h_harjoittelijat
id : int(3)
opiskelijanumero : varchar(10)
etunimi : varchar(255)
sukunimi : varchar(55)
ryhma : varchar(12)
pisteet : int(2)
harjoittelupisteet : int(2)
55op : int(2)
ennakko_pisteet : int(2)
jakson_pisteet : int(11)
harjoittelu. h_kayttajat
ID : int(11)
username : varchar(255)
kutsumanimi : varchar(255)
sukunimi : varchar(255)
sposti : varchar(255)

Prosessikaavio LIITE 2