



OPAS YHTEISÖLLISEEN KEHITTÄMISEEN

Projektina web-sovellus

Pekka Piispanen

Opinnäytetyö
Joulukuu 2014
Tietojenkäsittelyn koulutusohjelma
Ohjelmistotuotannon suuntautumisvaihtoehto

TAMPEREEN AMMATTIKORKEAKOULU
Tampere University of Applied Sciences

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietojenkäsittelyn koulutusohjelma
Ohjelmistotuotannon suuntautumisvaihtoehto

PIISPANEN, PEKKA:
Opas yhteisölliseen kehittämiseen
Projektina web-sovellus

Opinnäytetyö 32 sivua
Joulukuu 2014

Opinnäytetyössä käsiteltiin ison web-sovelluksen suunnittelua ja kehittämistä. Opinnäytetyön idean synnytti vuonna 2009 aloitettu innovaatiopankkisovellus, jota kehitettiin ympäri Suomen projektissa mukana olleiden ammattikorkeakoulujen kanssa. Isoimmat haasteet ja ongelmat projektissa olivat yhteisöllisen kehittämisen saaminen toimimaan saumattomasti sekä valtavien käyttäjämäärien käyttöön tarkoitetun sovelluksen rakentaminen huolellisesti ja sen saanti toimimaan vakaasti. Opinnäytetyön tavoitteena oli tehdä opas, joka antaisi apua ja ohjeistusta vastaavanlaista projektia aloittavalle.

Työssä oli tarkoitus käsitellä yleisesti yhteisöllisen kehittämisen sekä ison web-sovelluksen tekemisen kanssa kohdattavia haasteita. Yksi isoista haasteista ja tärkeistä asioista oli kommunikointi, varsinkin opinnäytetyöni taustalla olevan projektin tapauksessa, kun projektin jäseniä oli joka puolelta Suomea. Lisäksi tärkeää oli pohtia ja valita huolella kehittämiseen tarvittavat työkalut, jotka olisivat laadukkaita tällaista valtavaa projektia varten ja silti myös projektin kehittäjille helposti saatavilla ja hyvin omaksuttavissa.

Lopputuloksena on opas yhteisölliseen kehittämiseen, jossa käydään pääpiirteittäin läpi ison web-sovellusprojektin onnistumiseen vaadittavia asioita. Oppaan on tarkoitus antaa hyvät perustiedot web-sovelluskehittäjälle ja näin säästää aikaa projektin aloittamisvaiheessa.

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Degree Programme in Business Information Systems
Option of Software Engineering

PIISPANEN, PEKKA:
A Guide Book to Community Development
Web Application as a Project

Bachelor's thesis 32 pages
December 2014

This thesis was about designing and developing large scale web application. The idea for this thesis came up when I started developing an open innovation banking system early 2009. The project was developed with universities of applied sciences located everywhere in Finland. The greatest challenges and troubles faced were to get community development working seamlessly and to build web application for a large number of users. The web application must be designed thoroughly and it must be stable to accommodate web traffic when there are lots of people visiting the website simultaneously. The aim of this thesis was to create a guide book for someone, who is going to work on a similar project.

The purpose of the thesis was to give a general overview on community development and developing a large scale website. One of the biggest challenges -and the most important aspects- was communication; especially when project members are located all around the country. And just as important as choosing the development tools; they had to be good enough for a huge project like this and would have to be easy for project members to learn and embrace.

The result was a guide book to community development, which was about general things to pay attention to when developing a large scale web application. The purpose of the guide book is to give basic knowledge for web developers and save some time when starting a new project.

Key words: web software, community development, guide book

SISÄLLYS

| | | |
|-----|-----------------------------------|----|
| 1 | JOHDANTO..... | 5 |
| 2 | YHTEISÖLLINEN KEHITTÄMINEN | 6 |
| 2.1 | Hyödyt | 6 |
| 2.2 | Haasteet..... | 7 |
| 3 | PROJEKTIN INFRASTRUKTUURI | 8 |
| 3.1 | Kehittämisympäristö | 8 |
| 3.2 | Kommunikointi | 9 |
| 4 | KEHITTÄMISTYÖKALUJEN VALINTA..... | 12 |
| 4.1 | HTTP-palvelin | 12 |
| 4.2 | Tietokanta | 14 |
| 4.3 | Versionhallinta..... | 16 |
| 4.4 | Ohjelmointikieli | 20 |
| 4.5 | Ohjelmistokehys | 23 |
| 5 | YHTEENVETO | 27 |
| | LÄHTEET | 31 |

1 JOHDANTO

Opinnäytetyöni käsittelee yhteisöllistä kehittämistä ja toimii oppaana kyseisellä kehittämistavalla toteutettavalle web-sovellukselle. Vuoden 2009 aikana olin TAMKilla suorittamassa harjoittelua ja työskentelin ison web-sovellusprojektin parissa. Projektina oli ideapankkisovellus ja mukana oli ammattikorkeakouluja ympäri Suomea. TAMK oli valittu kouluista vetovastuuseen, ja tätä vetovastuuta hoidin luokkakaverini Joel Peltosen kanssa kahdestaan. Meidän tuli siis tehdä päätöksiä muun muassa projektin kehittämiseen käytettävistä tekniikoista sekä yhteydenpitomenetelmistä. Projektin työnimi oli sitä tehdessämme Open Innovation Banking System eli OIBS.

Projektilla oli jo pohja, sillä sitä olivat aloittaneet kaksi muuta ohjelmoijaa ennen minua ja Joelia. Hyvin alkukantaisessa vaiheessa se kuitenkin vielä oli, mutta projektista sai silti hieman käsitystä ja osa kehittämistyökaluista oli jo valittu. Nämä jo tehdyt valinnat olivat hyödyksi, kun projektia lähdimme Joelin kanssa suunnittelemaan lisää. Vaikka valitut työkalut eivät välttämättä jäisikään käyttöön, niin niitä voitiin pitää lähtökohtina tutkiessamme eri tekniikoita ja tapoja työskennellä.

Projektin alussa kului kuitenkin toista kuukautta aikaa pelkästään tiedon etsimiseen ja tutustuttaessa uusiin tekniikoihin. Määränpäinä oli valita työkalut laajan sovelluksen rakentamiseen verkossa siten, että yhteistyö olisi sujuvaa ja ongelmaton. Haasteena oli löytää sellaiset työskentelytavat, jotka olisivat mahdollisimman monelle uudelle projektin jäsenelle helpot omaksua, mutta myös sellaiset, jotka olisivat riittävän tehokkaat viemään tällaista projektia eteenpäin. Projekti oli tarkoitus tehdä avoimen lähdekoodin periaatteella, ja avoin lähdekoodi on myös tässä oppaassa pääosin lähtökohtana.

Tekniikoita etsittäessä ja vertailtaessa kävi useaan otteeseen mielessä, että eiköhän tällaisen projektin tekemiseen löydy jonkinlaista ohjeistusta Internetistä. Kuitenkaan mitään suoranaista opasta ei etsinnöistä huolimatta löytynyt. Tietoa löytyy joidenkin projektin tekemisessä käytettyjen sovellusten ja tekniikoiden osalta, mutta ei kuitenkaan juuri sillä tasolla ja niin laajasti kuin tätä projektia varten olisi ollut tarve. Tästä sitten syntyi idea oppaasta, joka on keskittynyt yhteisöllisesti kehitettävän web-sovelluksen rakentamiseen. Oppaasta voi saada myös vinkkejä muunlaisien projektien kehittämiseen, mutta pääpaino on kuitenkin web-sovelluksella.

2 YHTEISÖLLINEN KEHITTÄMINEN

Yhteisöllinen kehittäminen on kehittämistapa, jossa ryhmä yhdessä kehittää kohdetta, jonka käyttäjiä he myös yleensä ovat. Tässä tapauksessa kehitettävä kohde siis oli ideapankkisovellus, ja yhteisön muodostivat mukana olleet ammattikorkeakoulut ympäri Suomea, jolloin yhteistyötä tekivät koulujen kehittäjät ja opiskelijat. Kehittäminen eteni yhteisön jäsenien palautteen ja toiveiden perusteella.

2.1 Hyödyt

Mitä isompi projektissa mukana oleva yhteisö on kehittäjien osalta, sitä laajemmin voi löytyä eri osa-alueiden osaajia. Kehittäminen helpottuu, kun mukana on laajasti eri osa-alueiden osaajia, ja näin voi myös syntyä jokaiselle omat roolit osaamisalueiden mukaan. Jokainen voi siis keskittyä kehittämään niitä asioita, jotka aidosti hallitsee parhaiten.

Jossakin kohtaa projektia voi tulla tarve muuttaa tai päivittää siihen liittyviä perusasioita, kuten esimerkiksi vaihtaa versionhallintajärjestelmää parempaan tai ottaa mukaan kokonaan uusia kehittämistapoja, esimerkiksi otetaan käyttöön uusi ohjelmointikieli. Tällaiset isot projektiin liittyvät päätökset on selkeämpää tehdä useamman henkilön voimin. Lisäksi tämänkaltaisia muutoksiin liittyviä tarpeita voi tulla esiin vasta isommassa projektiryhmässä, esimerkiksi pidettäessä ryhmän kesken palavereita.

Ryhmässä saa myös erilaisia näkökulmia asioiden toteuttamiseen, eikä yhden henkilön tarvitse välttämättä pohtia kaikkia toteuttamistapoja itsekseen. Yksin miettiessä ei välttämättä edes tulisi kaikki mahdolliset toteuttamistavat mieleen, joten ryhmän kesken pohdittaessa tulee ennen kehittämisen aloittamista paremmin mietittyä, mitä pitäisi tehdä ja miten. Myös ongelmanratkaisu käy helpommaksi, kun on useampi henkilö ratkomaan sitä. Jollekin toiselle hyvin ilmiselvä ratkaisu ongelmaan voi toiselle olla hyvin vaikeasti hahmotettavissa, ja yhdessä siitäkin selviää nopeammin.

2.2 Haasteet

Silloin kun on iso ryhmä työskentelemässä projektin parissa, pitää ennen projektin alkua miettiä esimerkiksi toimintatavat ja -menetelmät, sovellukset joita käytetään ja miten kommunikointi hoidetaan projektin aikana. Pitää siis löytää oikeat tavat, että projekti toimisi ja että kaikille projektin jäsenille olisi helppoa tulla projektiin mukaan ja omak-sua kaikki päätetyt toimintatavat.

Haasteita tuo se, että väistämättä kehittämässä on osaamistasoltaan erilaisia ihmisiä. Eteneminen voi siis olla välillä hidastakin esimerkiksi projektin jäsenen vasta ottaessa haltuun uusia ohjelmointitapoja, mutta toisaalta tällä tavalla kehitettäessä päämääränä ei olekaan saada projektia valmiiksi mahdollisimman nopeasti. Kehittäminen ei muuten-kaan ole isolla ryhmällä välttämättä yksinkertaista, jos esimerkiksi samaa työtehtävää tekee epähuomiossa kaksi ihmistä yhtä aikaa. Tällaisessa tapauksessa versionhallinta auttaa jonkin verran, sillä se ilmoittaa aina käyttäjille, mikäli jossakin tiedostossa esiin-tyy konflikteja. Tämä tarkoittaa siis tilannetta, jolloin kaksi ihmistä on muokannut sa-man tiedoston samaa kohtaa. Vaikka konflikti on selvitettävissä versionhallintaa apuna käyttäen, on paljon nopeampaa, jos tällaisia tilanteita ei synny ollenkaan.

Kuten aiemmin mainittiinkin, on tärkeää, että kommunikointi on sujuvaa ja nopeaa. Varsinkin edellisessä kappaleessa mainittuja konfliktitilanteita ajatellen olisi parasta, mikäli toiseen ongelmatilanteeseen liittyvään henkilöön saadaan nopeasti yhteys. Tie-tysti muutenkin kommunikoinnin on sujuttava hyvin, sillä projektissa voi milloin tahan-sa tulla tilanteita, jolloin jokin asia on saatava nopeasti kaikkien muiden tietoon.

Varmuuskopiointi on myös tärkeässä asemassa, kun projektiryhmässä on paljon henki-löitä. Mitä enemmän projektissa on väkeä, sitä suuremmalla todennäköisyydellä sattuu huolimattomuusvirheitä, ja esimerkiksi kaikki tiedostot voivat hävitä. Versionhallinta on toki tässä kohtaa myös apuna, mutta kaikki projektin tiedostot eivät projektista riip-puen välttämättä kuulu versionhallintaan. Tietokannasta varsinkin kannattaa ottaa varmuuskopioita usein, jotta aina olisi mahdollisimman tuore versio tallessa, jos jotakin sattuu. Tietokannan varmuuskopioita voi myös lähettää versionhallintaan talteen, mutta usein niiden koko on niin suuri, että niiden varmuuskopioimisessa ja muiden hakiessa niitä omille tietokoneilleen kestäisi aivan liian kauan.

3 PROJEKTIN INFRASTRUKTUURI

Projektin alussa ensimmäiset mietittävät asiat olivat kehittämissympäristö sekä kommunikointi. Piti siis selvittää, millä käyttöjärjestelmillä kukin projektin jäsen voisi olla mukana kehittämässä ja löytyykö samoja tai edes yhteensopivia ohjelmistoja niin Windows-, Linux- kuin Applen OS X -käyttöjärjestelmällekkin. Ja kuten aiemmin mainittiinkin, yhteydenpito tulisi olemaan tärkeää, kun projektin jäsenten välimatkat olivat suuria.

3.1 Kehittämissympäristö

Web-sovellusta kehittäessä hyvä asia on se, että ainakin yleisimmät kehitykseen tarvittavat ohjelmistot löytyvät myös kaikille yleisimmille käyttöjärjestelmille. Muun muassa HTTP-palvelimet Apache ja nginx, tietokantaohjelmistot MySQL ja SQLite ja versionhallintaohjelmistot Subversion ja Git löytyvät niin Windowsille, Linuxille kuin Applen OS X -käyttöjärjestelmällekkin. Näin ollen itse web-sovellus voi sijaita minkä tahansa käyttöjärjestelmän sisältävässä palvelimessa ja web-sovellusta voi myös kehittää millä käyttöjärjestelmällä haluaa. Avoimen lähdekoodin ohjelmistoissa pitäydyttäessä ei tietystikään Windows ja OS X tule kysymykseen, mutta esimerkiksi Linux-jakelut ovat pääasiassa kaikki avoimen lähdekoodin käyttöjärjestelmiä.

Projektimme web-sovellus sijaitsi TAMKIn Linux-palvelimella, jossa oli asennettuna HTTP-palvelinohjelmisto Apache ja tietokantaohjelmisto MySQL. Versionhallinta sijaitsi SourceForge.net-sivuston versionhallintajärjestelmässä. SourceForge.net on Internetin suurin avoimen lähdekoodin kehittämiseen liittyvä sivu, ja sivustolta löytyy yli 430 000 projektia (SourceForge n.d.). Projektia varten oli itse asiassa kaksi sijaintia, oibs.projects.tamk.fi, jota käytettiin lähinnä projektin esittelyyn muille ja jonka sisältämiä web-sovellusta ei muutettu tai kehitetty lainkaan. Toinen sijainti oli oibs2.projects.tamk.fi, jonne varsinainen kehitys tehtiin.

Minun ja Joelin osalta kehitys tapahtui TAMKIn Windows-koneilla, joihin oli asennettu web-sovelluksemme ympäristö. Koneilta löytyi siis myös Apache ja MySQL ja versionhallinnan käyttöä varten ohjelma TortoiseSVN. Tyypillinen kehittämisspäivä alkoi sillä, että ensi töiksemme haimme TortoiseSVN-ohjelmalla viimeisimmät muutokset SourceForgen versionhallinnasta Windows-koneillemme. Tämän jälkeen kaikki kehit-

täminen tapahtui paikallisesti omilla koneillamme. Kehitystä pystyi siis tekemään huolelta, kun muutokset eivät menneet reaaliaikaisesti kaikkien nähtäville. Uusia ominaisuuksia sai siis rauhassa testata toimiviksi, ja tämän jälkeen pystyi halutessaan lähettämään tehdyt muutokset SourceForgen versionhallintaan. Yleensä jokaista pientä yksittäistä muutosta ei kuitenkaan lähetetty erikseen, vaan isommissa palasissa, esimerkiksi päivän lopussa. Muutoksia lähetettäessä pystyi myös kirjoittamaan kommentteja tekemistään muutoksista, mikä oli erittäin toivottavaakin. Näin muut kehittäjät näkevät nopealla vilkaisulla, mitä kaikkea muutospaketin lähettänyt kehittäjä on tehnyt viimeksi.

3.2 Kommunikointi

Isossa projektiryhmässä hyvä kommunikointi on tietysti tärkeää. Erityisen tärkeää kommunikointi on tällaisessa projektissa kuin mitä meillä TAMKilla oli, kun projektin jäseniä on pääkaupunkiseudulta Rovaniemelle. Pitkien välimatkojen takia ei ole mahdollista kovin usein tavata kasvotusten, eikä se ole kovin järkevääkään. Kuitenkin useimmiten kommunikoinnin laatu on sellaista, ettei kasvotusten tapaaminen ole välttämätöntä. Toki muutaman kerran projektin jäseniä nähtiin kasvokkain.

Sähköposti on hyvä tapa viestittää projektin jäsenten kesken, varsinkin jos vastauksella ei ole tulenpalava kiire. Sähköposti on muutenkin luonteeltaan sellainen, että sen kautta voi helposti hieman pitemmän kirjoitelman lähettää eteenpäin, toisin kuin esimerkiksi jotakin pikaviestintäohjelmaa käytettäessä. Puhelimella soittaminen on myös hyvä vaihtoehto, varsinkin kun ryhmäpuhelut ovat mahdollisia. Voi kuitenkin olla kätevämpää asentaa kaikille projektin jäsenille Skype, joka on nykyisin Microsoftin omistama pikaviestintäohjelma. Skypen avulla voi soittaa video- ja äänipuheluita Internet-yhteyden yli. Projektin jäsenten kesken voitaisiin muodostaa Skype-ryhmäpuhelu ja samalla kaikki voivat olla tietokoneidensa ääressä ja projektitiedostot ovat heti käsillä.

Google Docs on myös hyvä väline yhteydenpitoon. Se on ilmainen, Internet-selaimessa toimiva Microsoft Officen kaltainen toimisto-ohjelmakokonaisuus. Jos esimerkiksi projektissa mukana olevien työskentelyajat eivät kohtaa, voi Google Docsista olla iso hyöty. Esimerkiksi projektin tehtävälista voisi olla siellä, ja jokaiselle projektin jäsenelle on annettu tähän tiedostoon oikeudet. Tehtävälistaan voi käydä omaan tahtiin merkkäämassa, minkä tehtävän ottaa työn alle ja myös käydä merkkäämassa, kun sen on saanut tehtyä. Tai Google Docsissa voisi olla tiedosto, jossa esimerkiksi käsitellään projektiin

tulevia muutoksia tulevaisuudessa ja pyydetään kommentteja projektin jäseniltä. Helppointa on käydä tuohon yhteen tiedostoon kommentoimassa sitten, kun itselle sopii. Google Docsissa on myös chat-ominaisuus, joten dokumenttia lukevat voivat myös viestitellä keskenään ikkunan alalaitaan aukeavassa chat-ikkunassa.

Yksi maailman vierailuimmista web-sivustoista on yhteisöpalvelu Facebook. Facebookissa voi luoda projektiryhmälle oman ryhmäsivun, tai vain luoda yksinkertaisen ryhmäkeskustelun, jossa kaikki ovat mukana. Ryhmäkeskustelusta tekee helpompaa se, että kaikki projektin jäsenet ovat Facebook-kavereita keskenään, sillä käyttäjien yksityisyysasetukset eivät välttämättä salli yksityisviestejä sellaisten henkilöiden kesken, jotka eivät ole keskenään kavereita. Projektin jäsenet eivät välttämättä ole kuitenkaan halukkaita tätä asetusta vaihtamaan. Eivätkä kaikki välttämättä edes halua henkilökohtaisen Facebook-tilinsä olevan millään tavalla kytköksissä tällaisiin projekteihin tai muuhunkaan työhön liittyvään, vaan haluavat pitää profiilin omana ja vain vapaa-aikaan liittyvänä. Lisäksi kaikki eivät välttämättä edes ole luoneet tunnusta Facebookiin.

Älypuhelinien yleistyessä on tekstiviestittelyn rinnalle tullut muita käteviä viestittelysovelluksia, kuten WhatsApp, Kik Messenger tai ChatOn. Esimerkiksi WhatsApp toimii niin, että kunhan toisen henkilön puhelinnumero löytyy yhteystiedoista, voi alkaa keskustelemaan. Keskustelu ja viestit kulkevat Internet-yhteyden yli. Tällaisen pikaviestimen avulla on näppärä luoda ryhmäkeskustelu kaikkien projektiryhmäläisten kesken. Tämä on suhteellisen vaivaton tapa kommunikoida, tarvitaan siis vain yksi sovellus ja puhelinnumerot yhteystietoluetteloon.

Monien uusien pikaviestimien takia nykyisin vähemmälle huomiolle jäänyt kommunikointiväline on IRC. IRC on suomalaisen Jarkko Oikarisen vuonna 1988 kehittämä tekstipohjainen pikaviestinpalvelu. IRCin suosio on laskenut tasaisesti vuoden 2003 jälkeen, pientä nousua vuonna 2005 lukuun ottamatta (Pingdom 2012). Sen lisäksi, että pikaviestimien saralle on ilmaantunut paljon uusia tulokkaita - esimerkiksi jo kuollut MSN Messenger - on käyttäjämääriä saattanut laskea myös IRCin vaikeampi käyttö. Lisäksi IRC-sovellukset ovat tavallisesti hyvin pelkistetyn näköisiä, joten voi olla houkuttelevampaa alkaa käyttämään jotakin kiinnostavamman näköistä ohjelmaa. IRC koostuu keskusteluhuoneista, joita kutsutaan kanaviksi. Käyttäjät voivat olla useilla kanavilla yhtä aikaa, ja kanaville ja käyttäjille voidaan asettaa eri asetuksia ja oikeuksia.

IRC ei kuitenkaan edusta pelkästään pelkkää laskusuuntaa tai tuulen huminaa tyhjiudessa. Hyvänä esimerkkinä on Freenode-verkko, joka on ollut kasvussa vuodesta 2001 lähtien. Freenode on myös siitä erityinen IRC-verkko, että sen tukena ovat palveluntarjoajat, yliopistot ja muut järjestöt tarjoamalla palvelimia ja kaistaa liikennöintiä varten (IRC-Junkie.org 2012). Tästä syystä Freenodesta löytyy kanavia eli keskustelualueita lähes aiheeseen kuin aiheeseen liittyen. Esimerkiksi PHP:tä, MySQL:ää ja HTTP-palvelin Apachea varten on omat kanavansa. Kanavilla on runsaasti aktiivisia käyttäjiä ja keskustelu on vilkasta. Ongelmatapauksissa vastauksen saaminen on myös erittäin todennäköistä ja nopeaa.

IRC on siis yksi hyvä vaihtoehto kommunikoinnille, kunhan pieni säätö käytön aloittamisessa ei pelota. Samalla kun oma kanava on projektia varten luotu ja yhteydenpito on siellä helppoa ja nopeaa, voi myös liittyä projektin kannalta hyödyllisille muille kanaville, joista voi saada apua tarvitessa. Samassa paikassa voi siis pitää reaaliaikaisesti yhteyttä projektin jäsenten kanssa ja pitää keskustelukanavat auki esimerkiksi ohjelmointikielen yhteisöön.

Kuten alussa mainittiinkin, oli hyvien kommunikointivälineiden löytäminen tärkeää projektillämme. Usealle projektimme jäsenelle IRC oli jo ennestään tuttu, joten sitä päätimme alkaa käyttää saman tien. Koska muutamille IRCiä käyttämättömälle ei ollut ongelma alkaa sitä käyttämään, saimme yhteiselle IRC-kanavalle hyvin projektin jäseniä. IRC-kanavamme sijaitsi aiemmassa kappaleessakin mainitussa Freenode-verkossa. Oli erittäin kätevää, että projektiryhmän kanssa yhteyttä pidettäessä pääsi myös reaaliaikaiseen keskusteluun asiantuntevan väen joukkoon, lähes aiheesta kuin aiheesta. Samalla kun oli projektin keskustelukanava IRC-ohjelmassa auki, niin olivat myös usein ikkunat auki muun muassa PHP- ja MySQL-kanaville.

Muista tässä alaluvussa mainituista kommunikointivälineistä käytettiin lähinnä sähköpostia. Vaikka Facebook oli projektimme aikaan vuonna 2009 ollut jo jonkin aikaa tuttu sosiaalisen median paikka meille suomalaisillekin, ei sen suosio ollut silti nykyisen kaltainen. Itsekin liityin vasta samana vuonna käyttämään Facebookia. Sen käyttäminen projektin viestintävälineenä ei oikein missään kohtaa tullut edes esille, eikä sitä koskaan otettukaan käyttöön.

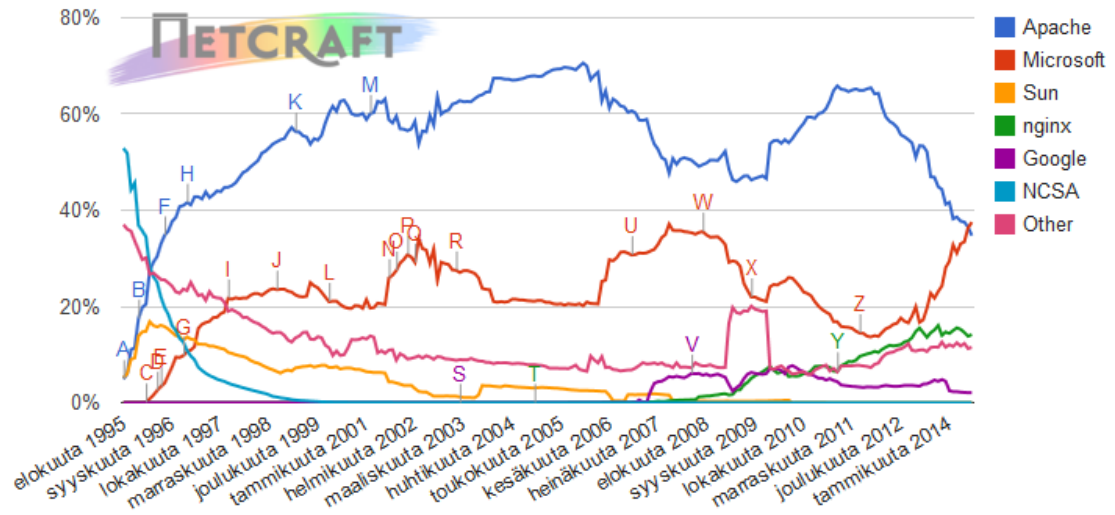
4 KEHITTÄMISTYÖKALUJEN VALINTA

Kun projektissa oli tutustuttu siihen, millaisissa kehittämissympäristöissä toteutus tulisi tapahtumaan ja oli tehty päätöksiä kommunikoinnin suhteen, oli projektin alkamiselle hyvä perusta. Seuraavaksi piti valita, millaisilla työkaluilla web-sovellusta lähdettäisiin rakentamaan. Työkalujen piti olla laadukkaita, mielellään yleisesti paljon käytettyjä sekä aktiivisesti edelleen kehitettäviä. Pohdittavat työkalut muodostaisivat siis käytännössä koko projektin perustan, ja niihin kuuluvat tärkeimpinä HTTP-palvelin, tietokantaohjelmisto ja versionhallintaohjelmisto. Lisäksi ohjelmointikieli ja siihen liittyvä mahdollinen ohjelmistokehys olivat työkaluja, joita piti projektia varten miettiä.

4.1 HTTP-palvelin

HTTP-palvelinohjelma on ohjelma, jonka avulla web-sovellukset näkyvät käyttäjille. Se on siis Internet-sivuston ydin, jonka ympärille kaikki rakentuu. Käyttäjä selailee sivustoa jonkin selainohjelman avulla, joka lähettää pyyntöjä HTTP-palvelimelle. Ja tätä kautta haetaan tietoa esimerkiksi tietokannasta, joka sitten näytetään käyttäjän ruudulle selaimessa.

HTTP-palvelimen valintaa pohdittaessa on hyvä tutustua ensimmäiseksi todellisiin tilastoihin suosituimmuudesta. Käytetyimmät kolme palvelinohjelmistoa erottuvat selvästi muista, Apachen ollessa käytännössä käytetyin palvelin. Tässä kohtaa "käytännössä" tarkoittaa sitä, että käytetyimmän palvelinohjelmiston piikki paikka riippuu siitä, mitä tilastoa tutkii. HTTP-palvelimista tilastoja pitävän Netcraft-sivuston (2014) mukaan Microsoftin palvelinohjelmisto IIS on käytetyin, kun tarkastellaan asiaa rekisteröityjen www-osoitteiden määrän kannalta. Heinäkuu 2014 oli käännekohta tätä tilastoa ajatellen, sillä Microsoft syrjäytti Apachen käytetyimpänä ohjelmistona (Kuva 1).

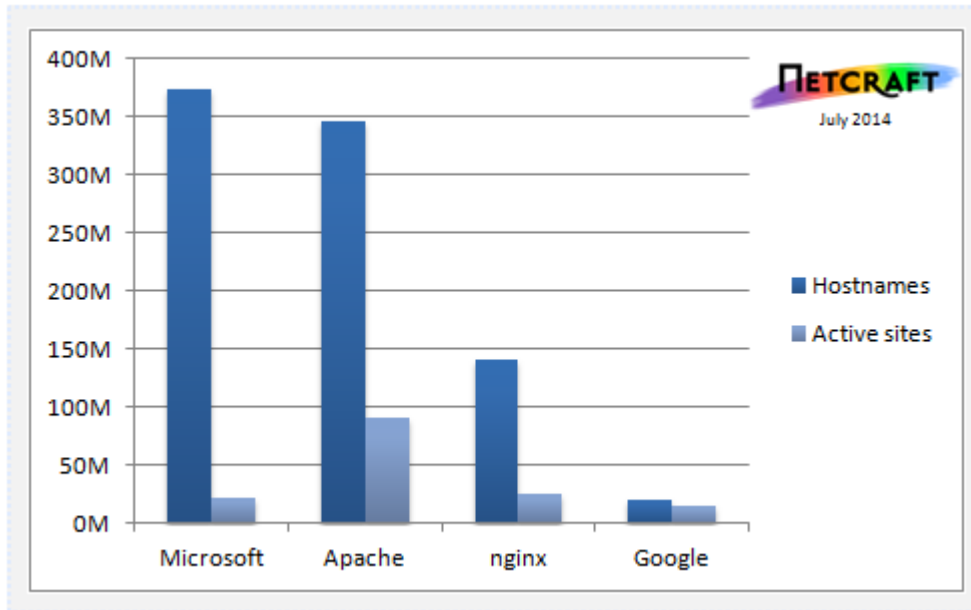


KUVA 1. Käytetyimmät palvelinohjelmistot (Netcraft 2014)

Kaikkien rekisteröityjen www-osoitteiden mukaan tehtyä tilastoa katsottaessa, on huomattavissa aika suuriakin nousuja ja laskuja ajan mittaan. Tämä tilasto kuitenkin sisältää paljon turhia www-sivustoja, jotka ovat vain esimerkiksi mainostamiseen tai uhkapeleihin liittyviä (Netcraft 2014).

Merkittävämpi tilasto on kuitenkin se, jossa mitataan suosituimmuutta käyttäen aidosti aktiivisia sivustoja mittapuuna. Rekisteröityjen sivustojen määrä on moninkertainen aktiivisten sivujen määrään verrattuna, joten turhia sivustoja on maailmalla hyvin paljon (Netcraft 2014). Aktiivisia sivustoja laskettiin ennen uniikkien IP-osoitteiden perusteella, mutta nykyisillä tekniikoilla samasta IP-osoitteesta voi toimia moniakin todellisia eri sivustoja. Näin ollen tilastointitekniikkaa piti muuttaa, jotta saataisiin luotettavampia lukemia. Nykyisin käydään läpi saman IP-osoitteen eri sivustoja, ja verrataan niiden etusivuja toisiinsa. Täysin samanlaista sisältöä olevat sivustot lisäävät siis sivustojen kokonaismäärää vain yhdellä (Netcraft n.d.).

Kun tarkastelussa ovat vain aktiiviset sivustot, kärkipaikkaa pitää Apache (Kuva 2). Apachen kokonaissivustomäärästä noin neljännes on aktiivisia, kun Microsoftilla aktiivisia sivustoja on vain viisi prosenttia. Kaikkien sivustojen tilastojen valossa jäi HTTP-palvelin nginx pienelle huomiolle kolmannella sijallaan ja huomattavasti pienemmällä prosentiosuudellaan, mutta pelkästään aktiivisten sivujen määrää vertailemalla se onkin yllättäen toiseksi käytetyin HTTP-palvelinohjelmisto (Netcraft 2014).



KUVA 2. Käytetyimmät palvelinohjelmistot, aktiiviset sivut (Netcraft 2014)

Palvelinohjelmiston valinnan tullessa pohdittavaksi ideapankkisovellusta kehittäessämme oli Apache HTTP Server ensimmäisenä mielessä. Se oli myös valittu käyttöön jo tätä projektia luotaessa. Lisäksi Apachen nimi oli tullut tutuksi monella kotisivupalvelimella, joita olin käyttänyt vapaa-ajalla joihinkin omiin ohjelmistoprojekteihin. Siitä tuntui myös olevan saatavilla hyvä dokumentaatio sekä hyvin paljon keskusteluita ja foorumeita ongelmatilanteiden selvittelyyn. Jos siis jonkin asian kanssa meni sormi suuhun, oli sama ongelma todennäköisesti ollut jo jollakin toisella, ja asia oli saatettu jo ratkaistakin. Vähintään keskustelua aiheesta oli ollut ja aina löysi joitakin vinkkejä, miten edetä asian kanssa. Asiaan perehtymisen jälkeen päätimme pysyä jo käytössä olleessa Apache HTTP Serverissä, se vaikutti hyvältä projektiamme varten.

4.2 Tietokanta

Tietokantaohjelmisto on ohjelma, johon varastoidaan tietoa. Tietotekniikassa käytettävän tietokannan ei tarvitse välttämättä olla erillinen ohjelma, sillä esimerkiksi tekstitiedostoja apuna käyttäen voi syöttää, muokata ja poistaa tarvittavia tietoja. Hyvä esimerkiksi tietokannan sisällöstä on esimerkiksi verkkokaupan tuotevalikoima ja verkkokaupan asiakkaiden käyttäjätiedot ja tilaushistoriat.

DB-Engines-sivuston (2014) pitämän tilaston mukaan MySQL on toiseksi suosituin tietokantaohjelmisto (Kuva 3). Tätä tilastoa päivitetään joka kuukausi. Kärkipaikkaa

pitää Oracle, joka tosin nykyisin omistaa MySQL:n, joten kaksi suosituinta tietokantaohjelmistoa ovat saman yrityksen tuotteita. Kolmantena on Microsoft SQL Server, ja nämä kolmea kärkipaikkaa pitävät ovat ylivoimaisesti suosituimpia.

Tilastointi perustuu yleiseen suosituimmuuden tutkimiseen, jossa apuna käytetään muun muassa Google Trendsia, jonka avulla voi vertailla hakusanojen suosiota. Tämän lisäksi apuna käytetään tunnettujen ongelmanratkaisuun liittyvien sivustojen sisältöjä kartoittaen, kuinka suuri osa on mitään tietokantaohjelmistoa koskevia sekä Googlen ja Bingin hakukoneiden hakutulospäämiä tietokantaohjelmistoihin liittyen. Tietokantojen yleisyys sosiaalisessa mediassa on myös yksi tilastointikohde ja sekä ohjelmistojen esiintyvyys avoimissa työpaikkailmoituksissa (DB-Engines n.d.).

223 systems in ranking, September 2014

| Rank | Last Month | DBMS | Database Model | Score | Changes |
|------|------------|----------------------|-------------------|---------|---------|
| 1. | 1. | Oracle | Relational DBMS | 1466.91 | -3.95 |
| 2. | 2. | MySQL | Relational DBMS | 1297.14 | +15.92 |
| 3. | 3. | Microsoft SQL Server | Relational DBMS | 1208.87 | -33.62 |
| 4. | 4. | PostgreSQL | Relational DBMS | 255.79 | +5.94 |
| 5. | 5. | MongoDB | Document store | 240.98 | +3.63 |
| 6. | 6. | DB2 | Relational DBMS | 197.03 | -9.39 |
| 7. | 7. | Microsoft Access | Relational DBMS | 140.48 | +0.86 |
| 8. | 8. | SQLite | Relational DBMS | 92.61 | +3.74 |
| 9. | ↑ | 10. Cassandra | Wide column store | 87.86 | +5.96 |
| 10. | ↓ | 9. Sybase ASE | Relational DBMS | 85.42 | -0.75 |

KUVA 3. Tietokantaohjelmistojen suosituimmuusjärjestys (DB-Engines 2014)

Listalla kahdeksantena oleva SQLite todellisuudessa kiilaisi kärkipaikoille suosituimmuudessa, sillä tämä tietokantaohjelmisto löytyy jokaisesta Applen iPhonesta ja Googlen Android-käyttöjärjestelmää käyttävästä älypuhelimesta sekä myös muun muassa Googlen Chrome-selaimesta (SQLite n.d.). Android-käyttöjärjestelmää käyttäviä älypuhelimia oli heinäkuuhun 2014 mennessä myyty lähes 1,2 miljardia kappaletta, ja samaan ajankohtaan mennessä Applen iPhone 5 -puhelinta kaikki eri versiot huomioon ottaen hieman yli 100 miljoonaa laitetta. Kaiken kaikkiaan iPhone-puhelinta on myyty vuodesta 2007 lähtien yli 500 miljoonaa kappaletta (Statistic Brain 2014a; Statistic Brain 2014b; Rogowsky 2014).

Tietokantaohjelmistoja on hieman hankala verrata suorituskyvylisesti, sillä ne ovat hyvin erilaisia. Ohjelmoinnin näkökulmasta tietokannan kanssa työskentely on hyvinkin

samanlaista esimerkiksi SQL-kielen syntaksin osalta, mutta toteutustekniikka eroaa paljon. Esimerkiksi SQLite tallentaa kaiken tiedon vain yhteen tiedostoon ja on siitä syystä myös erittäin helposti siirrettävä toisiin alustoihin tai järjestelmiin. MySQL-tietokannassa tieto taas on omassa formaatissaan, joten sisältöä ei suoraan voi siirtää toisaalle eikä tietokanta ole muiden tietokantaohjelmistojen luettavissa. MySQL:stä löytyy kuitenkin vientitoiminto, jolla tietokannan saa varmuuskopioitua yhteen tiedostoon. Tiedon tallennustavan myötä myös nämä kaksi tietokantaohjelmistoa eroavat tietoturvan osalta. SQLitessa ei ole minkäänlaista suojausta tietokannan tietoihin, tietokantatiedoston löytäessään voi kuka tahansa avata sen. Tällöin tietysti SQLiteä käyttävässä ohjelmistossa on syytä olla kirjautuminen vaatimuksena tietokannan muokkaamiseen (Chipkin n.d.).

Lisäksi SQLite ei tue monia yhtäaikaisia tietokantaan tallennuksia: vain yksi tiedonmuutos kerrallaan on mahdollista. Tämä voi olla ongelma suurien käyttäjämäärien sivustoilla. Hyvä esimerkki on verkkokauppa, jossa tuhannet ihmiset tekevät yhtä aikaa tilauksia sekä luovat tai muokkaavat tunnuksiaan. Isossa mittakaavassa tämä saattaa tuoda selkeää hidastumista. MySQL:ssä monien yhtäaikaisten tietokantausekkeiden suorittaminen on mahdollista (Chipkin n.d.).

Kun projektissamme aloimme pohtia, mikä tietokantaohjelmisto pitäisi valita käyttöön, oli mielipide selkeä. MySQL oli ainoa jota olimme Joelin kanssa käyttäneet aiemmin, ja se oli myös jo valittu käyttöön tähän projektipohjaan aiempien ohjelmoijien toimesta. Vaikka meillä kummallakin tietokantojen ohjelmointikokemusta, emme olleet koskaan muita käyttäneet tai edes tehneet vertailua muihin tietokantaohjelmistoihin. MySQL oli helppo valinta toki siksi, että se oli molemmille jo aiemmastaan hyvin tuttu, mutta myös siksi, että se on hyvin yleinen tietokantaohjelmisto ja tämän takia Internetissä on valtavasti materiaalia, esimerkkejä ja vastauksia ongelmatilanteisiin. Lisäksi MySQL on käytössä myös suurien liikennöintimäärien sivustoilla, kuten Wikipedia ja Facebook, joten luotettavuuttakin löytyy (MySQL n.d.).

4.3 Versionhallinta

Versionhallinta on järjestelmä, joka tallentaa yhden tai useamman tiedoston muutoksia. Se on siis periaatteessa varmuuskopio projektin tiedostoista, sillä se voi sisältää esimerkiksi kokonaisen ohjelmiston kaikki tiedostot lähdekoodeineen. Versionhallinta pitää

kirjaa jokaisesta pienestäkin muutoksesta, joka tapahtuu tiedostoihin, joten mikä tahansa tiedoston aiemmista tiloista on palautettavissa. Ja kun versionhallintaan on jokaiselle käyttäjälle omat tunnuksensa, kirjautuu myös jokaisen muutoksen lisäksi hallintaan tiedot siitä, kuka muutoksen teki, miksi teki, ja esimerkiksi viittauksia mitkä siihenastisista ongelmista korjaantuvat tällä muutoksella. Versionhallinta on varsinkin silloin erittäin hyödyllinen, kun kyseessä on iso projekti ja suuri määrä tiedostoja, ja kun on paljon projektiin osallistuvaa väkeä. Kukin projektin ohjelmoija voi tahollaan tehdä muutoksia ohjelmiston koodiin, ja muutoksien ollessa valmiita ohjelmoija lähettää ne versionhallintaan. Kun muut käyttäjät lähettävät omia päivitettyjä tiedostojaan versionhallintaan, saavat he samalla nämä muiden tekemät muutokset myös itselleen. Tuoreimman version tiedostoista voi hakea milloin vain lähettämättä itse mitään muutoksia.

Versionhallintajärjestelmistä on olemassa kolmea erilaista tyyppiä. Niitä ovat paikalliset, keskitetyt ja hajautetut versionhallintajärjestelmät. Paikallinen versionhallinta sijaitsee nimensä mukaisesti paikallisesti käyttäjän tietokoneella. Yksi suosituimmista paikallisista versionhallintajärjestelmistä oli RCS, joka löytyy nykyäänkin muun muassa Mac OS X -käyttöjärjestelmästä kehittäjätyökalujen asentamisen jälkeen. Tällainen versionhallintajärjestelmä toimii siten, että se tallentaa kiintolevyllä muistiin jokaisen tiedostoon tapahtuneen muutoksen. Näitä muutostietoja hyväksi käyttäen voidaan tiedostosta palauttaa minkä tahansa ajankohdan versio (Git n.d.).

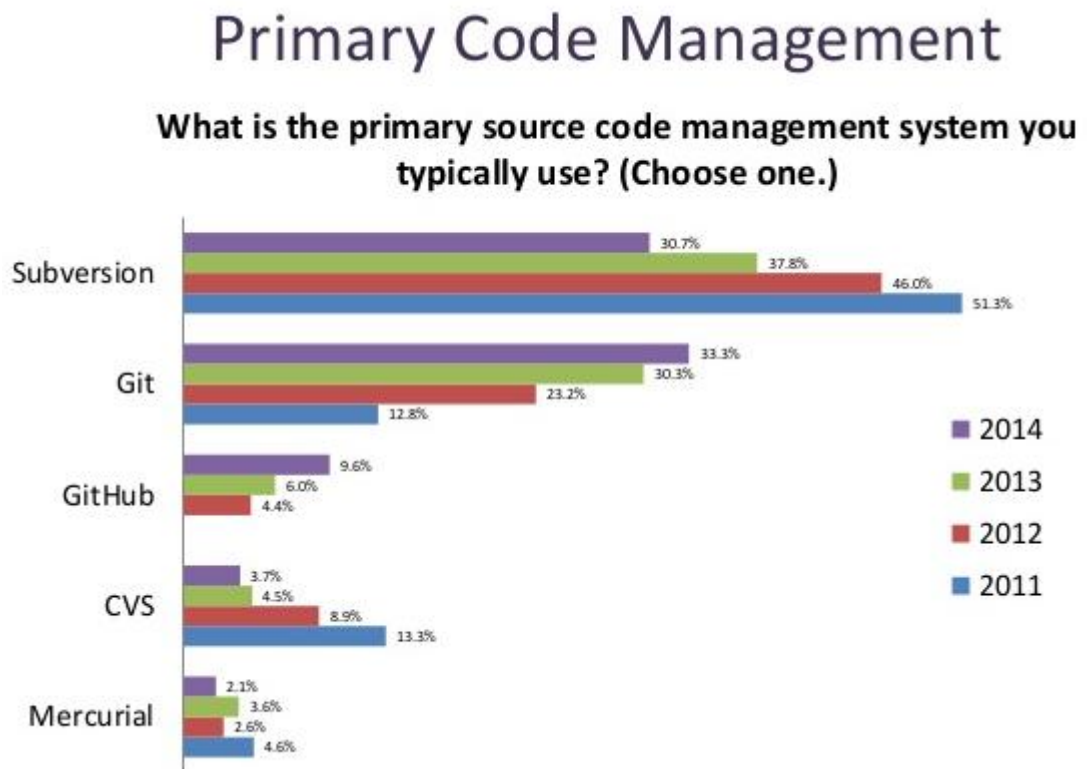
Keskitetyn versionhallinnan tapauksessa versionhallintajärjestelmällä on yksi palvelin, joka sisältää kaikki tiedostot kaikkine versioineen. Kaikki tällaisen järjestelmän käyttäjät hakevat tiedostot yhdestä samasta paikasta, jonne myös jokaisen käyttäjän tekemä jokainen muutoskin tallentuu. Keskitetyn versionhallinnan ehkä keskeisin ongelma on se, että ainoa täysi kopio tiedostoista on ainoastaan yhdessä paikassa, eli tällä järjestelmän keskuspalvelimella. Jos palvelin on esimerkiksi huoltotoimenpiteiden takia jonkin aikaa pois käytöstä, eivät kenenkään tekemät muutokset kulkeudu eteenpäin ennen kuin palvelin palaa linjoille. Tätä pahempi tilanne on vielä esimerkiksi se, että palvelin hajoaa fyysisesti ja siellä ollut tieto on lopullisesti saavuttamattomissa. Hyvällä onnella jollakin versionhallinnan käyttäjistä saattaa olla koko projekti viimeisimpine päivityksineen, mutta tilanne voi olla myös se, että jäljellä on enää vain palasia projektista. Näin pahan katastrofin voi välttää, kunhan versionhallintapalvelimen säännöllinen varmuuskopiointi on kunnossa. Keskitettyihin versionhallintajärjestelmiin kuuluu mm. Subversion (Git n.d.).

Hajautettu versionhallinta toimii niin, että toisin kuin keskitetyssä versionhallinnassa, käyttäjillä ei ole vain jotakin tiettyä tiedostoa tai pienempää osaa projektin tiedostoista, vaan jokaisella käyttäjällä on oma täydellinen kopionsa projektista koneellaan. Projektin tiedostoja päivitettäessä versionhallinnasta päivittyy siis joka kerta kaikki sisältö. Hajautetun versionhallinnan suurimpana etuna on se, että versionhallinnassa ei ole yhtä heikkoa lenkkiä, jonka täysi hajoaminen tai hetkellinen käyttökatko tuottaisi ongelmia projektin kehittämiselle. Kun jokaisella käyttäjällä on käytännössä täysi varmuuskopio projektista, ei yhden käyttäjän poistuminen projektista tuota ongelmia. Lisäksi etuna on se, että omien muutoksien lähettäminen tai projektin tiedostojen päivittäminen omaan kehitysympäristöön on nopeampaa, kun kaikkien käyttäjien toiminta ei kohdistu yhteen keskuspalvelimeen. Tiedostoja jaetaan vertaisverkon tapaan käyttäjältä käyttäjälle. Hajautettuja versionhallintajärjestelmiä ovat muun muassa Bazaar, Git ja Mercurial (Git n.d.).

Eclipse Foundation on yleishyödyllinen, voittoa tavoittelematon yhteisö ihmisille ja järjestöille, jotka haluavat toimia avoimen lähdekoodin ohjelmistojen parissa. Ehkä tunnetuin Eclipsen projekteista on Eclipse-ohjelmointiympäristö, jonka avulla voi ohjelmoida käyttäen esimerkiksi C++-, Java- ja PHP-ohjelmointikieliä (Eclipse Foundation n.d.). Eclipse Foundation on järjestänyt monena vuotena peräkkäin yhteisölleen kyselyn, jonka tarkoituksena on tutkia muun muassa millaisia ohjelmistoja kehittäjät ovat tekemässä sekä mitä eri ohjelmia kehittämisessä käytetään.

Tässä monivuotisessa Eclipse Surveyssä yhtenä kyselyn osa-alueena on koodin hallinta, jossa vertaillaan muutamien versionhallintajärjestelmien suosituimmuutta. Vuosista 2012 ja 2013 poiketen viimeisimmässä vuoden 2014 kyselyssä ei enää ole versionhallinnan osalta mukana IBM:n kahta versionhallintajärjestelmää. Syykin on selvä, aiempina vuosina kyselyn tuloksissa ne pitivät jo muutenkin häntäpäätä suosituimmuudessa hyvin marginaalisilla osuuksilla. Vuoden 2012 tilastoissa oli jo näkyvissä tuleva kehitys versionhallintajärjestelmien suosituimmuuden osalta. Subversionin pitäessä selkeää kärkipaikkaa hieman alle 50 prosentin osuudella, Git ja GitHub nousivat vuoden takaisesta tilastosta 14 prosenttiyksikköä pitäen siis 27 % osuutta yhdessä (Skerrett 2012). GitHub on palvelu projekteille, joissa käytetään Gitä versionhallintajärjestelmänä, ja niiden osuus tilastoitiin yhteenlaskettuna vuoden 2011 tilastoihin asti (Eclipse Foundation 2011).

Vuonna 2013 suunta oli sama, ja voimasuhteet olivat lähes tasoittuneet. Vuoden aikana Gitin ja GitHubin yhteenlaskettu osuus versionhallintajärjestelmistä oli jo hieman yli 36 prosenttia, kun samalla Subversion jatkoi laskuaan ja oli niukasti suosituin hieman vajaalla 38 prosentin osuudellaan (Skerrett 2013). Seuraavana vuonna ei suunnassa ollut muutosta, Git ja GitHub jatkoivat kasvamistaan ja Subversionin suosio jatkoi laskemistaan. Muutos oli jopa niin suurta, että Git nousi suosituimmaksi versionhallintajärjestelmäksi omallakin osuudellaan mitattuna, sen ollessa hieman yli 33 prosenttia. Subversionin osuus laski 30 prosenttiin, ja kun GitHubin osuuskin oli jo lähes 10 prosenttia, voidaan sanoa, että Subversionin aiempien vuosien dominointi on kääntynyt täysin päälleen (Kuva 4).



KUVA 4. Käytetyimmät versionhallintajärjestelmät (Skerrett 2014b)

Projektiin lähdeittäessä en ollut käyttänyt versionhallintaa lainkaan, vaikkakin tietämystä aiheesta olikin jonkun verran. Versionhallintaa piti siis lähteä miettimään omalta osalta melko tyhjästä. Projektipohjassa oli jo kuitenkin käytössä versionhallintaohjelmisto Subversion, joten se oli tietysti hyvä lähtökohta, varsinkin kun mistään muusta ei ollut aikaisempaa kokemusta. Silloinen ohjelmointiympäristömme oli Windowsissa, ja Subversionin käyttöön tarkoitettu asiakasohjelma TortoiseSVN oli asennettuna ja käytössä. Se vaikutti hyvin sopivalta, ja päätimme siis pitäytyä Subversionin käytössä, joka myös

vasta näin jälkikäteen tilastoja tutkien osoittautui myös hyväksi vaihtoehdoksi. Vuonna 2009 Subversionin osuus oli lähes 60 prosenttia, kun Gitin ja GitHubin yhteenlaskettu osuus vain hieman yli kaksi prosenttia (Eclipse Foundation 2009).

4.4 Ohjelmointikieli

Ohjelmointi on ohjeiden ja käskyjen luomista, minkä perusteella tietokone osaa tehdä jonkin asian ohjelmoijan haluamalla tavalla. Esimerkiksi verkkopankkiin kirjautuessa tietokone on ohjelmoitu tarkistamaan käyttäjätunnuksen ja tunnusluvun oikeellisuus. Ja olivatpa käyttäjätiedot oikein tai väärin, löytyy ohjelmasta ohjelmoijan tekemät toimenpiteet kumpaakin tilannetta varten. Ohjelmointia varten on olemassa hyvin monia eri ohjelmointikieliä, jotka eroavat toisistaan lauseopillisesti eli niiden syntaksi on erilainen. Hyvän ohjelmoijan on kuitenkin helppo oppia uusi ohjelmointikieli, sillä ohjelmoinnin peruslogiikka pysyy samana.

Ohjelmointikielet voidaan jakaa kahteen pääryhmään: asiakaspuolen ohjelmointikielet ja palvelinpuolen ohjelmointikielet. Asiakkaalla tarkoitetaan yleensä tietotekniseen tyyliin jotakin käyttäjän käyttämää ohjelmaa, eli web-sovelluksen tapauksessa siis Internet-selainta.

Asiakaspuolen ohjelmointikieliä ovat esimerkiksi HTML, CSS ja JavaScript. Näistä HTML ja CSS vastaavat sivuston staattisesta rakenteesta ja ulkoasusta. Käyttäjän saapuessa sivustolle Internet-selain lataa muistiin sivustoon liittyvän ohjelmakoodin palvelimelta, ja muodostaa sivuston ulkoasun HTML- ja CSS-määritysten mukaan. JavaScriptin avulla sivustolle saadaan dynaamisia toimintoja. Esimerkiksi jos verkkokaupan tuotesivulla on jokaisesta tuotteesta pitkä kuvausteksti, voi kuvaustekstistä olla aluksi näkyvissä vain ensimmäinen kappale. Kappaleen jälkeen voi olla linkki tai nappi, jota painamalla saa näkyviin loput tekstistä. Nämä kaikki asiakaspuolen ohjelmointikoodit ovat selaimen muistissa sivun latauduttua, joten sivusto näyttää ja toimii JavaScriptin dynaamisten toimintojenkin osalta ihan oikein, vaikka Internet-yhteys katkeaisi.

Huomionarvoinen seikka tosin on se, että JavaScriptiä voi myös käyttää liitettynä palvelinpuolen ohjelmointikieliin, jolloin edellä mainitussa yhteydettömässä tilassa sellaiset toiminnot eivät tietenkään toimi. Esimerkki tällaisesta voisi olla verkkokaupan asiakkaan tilaushistoria. Käyttäjä näkee ensin allekkain omat tilausnumeronsa linkkeinä, ja

linkkiä klikatessa JavaScript kutsuu palvelinpuolen ohjelmointikieltä, joka sitten hakee tarvittavat tiedot ruudulle. Tämän hyöty on se, että käyttäjän ei tarvitse ladata selaamaansa sivua uudelleen, vaan tiedot latautuvat reaaliaikaisesti ruudulle.

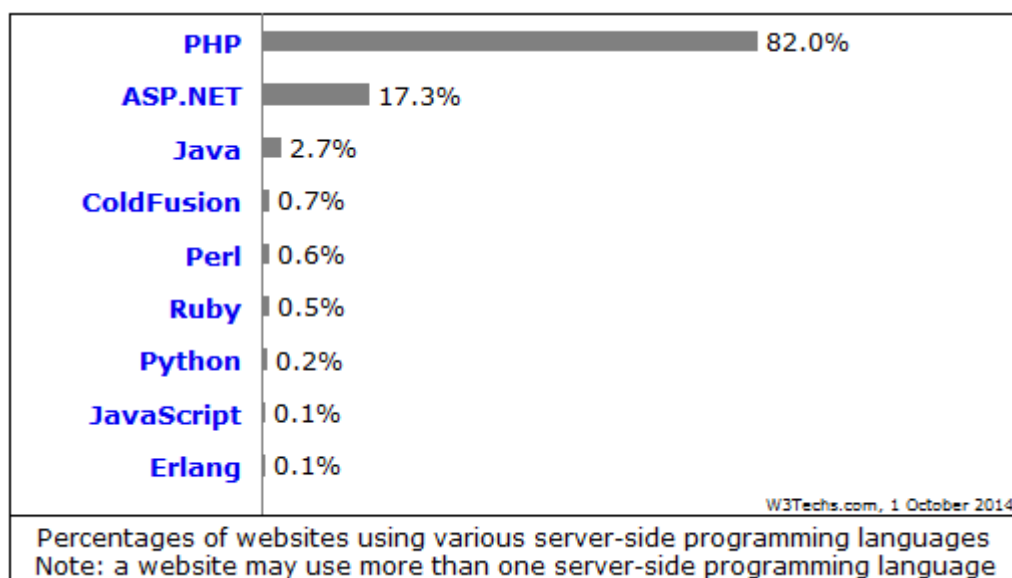
Palvelinpuolen ohjelmointikieliä ovat esimerkiksi PHP, Java ja Python. Selaimesta lähetetyt tiedot menevät näillä kielillä tehdyn ohjelmointikoodin tarkasteltavaksi. Esimerkiksi käyttäjän rekisteröityessä web-sivustolle voivat käyttäjätiedot mennä PHP-koodin kautta tietokantaan. Palvelinpuolen kielien tarkoitus on myös suorittaa ohjelmia, jotka sijaitsevat palvelimella, kuten juuri edellisessä virkkeessä mainittua tietokantaohjelmaa. Lisäksi tietoturvan kannalta on parempi, että esimerkiksi käyttäjätietojen kaltaiset tiedot ovat palvelimella, eivätkä ladattuna jokaisen käyttäjän tietokoneelle.

Myös ohjelmointikielen valinnassa on hyvä tutkia niiden yleisyyttä ja suosituimmuutta. Mitä suositumpi ja käytetympi ohjelmointikieli on, sitä enemmän siitä puhutaan keskustelupalstoilla ja sitä todennäköisemmin kohtaamasi ongelma on jollakin toisella tullut jo eteen. Lisäksi suosittua ohjelmointikieltä varten löytyy varmemmin erilaisia oppaita ja harjoitteita, joita tekemällä harjaantuu kielen ohjelmoinnissa.

Isoa web-sovellusta suunniteltaessa ei asiakaspuolen ohjelmointikielillä ole kovin suurta merkitystä. On myös hyvin sivustokohtaista, mille kaikille niistä on tarvetta, mutta lopputulokseen ei niillä ole kuitenkaan suuria vaikutuksia. Siksi seuraavissa tilastoissa ja vertailuissa kohdistankin huomion lähinnä palvelinpuolen ohjelmointikieliin.

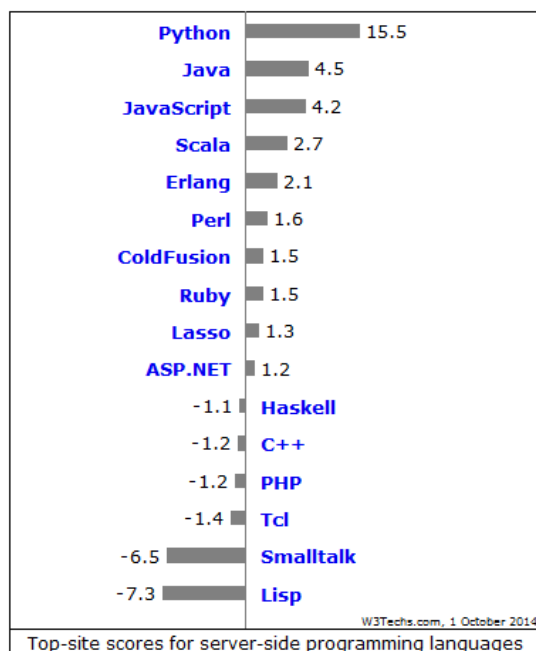
W3Techs-sivusto pitää yllä tilastoa eri tekniikoiden suosituimmuudesta, ja muun muassa siis tilastoa palvelinpuolen ohjelmointikielistä. Yksittäisellä sivustolla voi olla käytössä useampi ohjelmointikieli, ja tilastointiin riittää, että yksikin sivuston yksittäisistä sivuista käyttää kyseistä kieltä. Palvelinpuolen ohjelmointikieliä koskevia tilastoja on kaksi. Toinen koskee kaikkia web-sivustoja ja toinen vain suosituimpia. Kaikkia web-sivustoja käsittelevä tilasto on rajattu käsittämään maailman 10 miljoonaa suosituinta sivustoa, ja suosituimpia sivustoja koskeva tilasto käsittää maailman 1000 suosituinta sivustoa. Näitä tilastoja varten tutkittavat sivustot valitaan yhdysvaltalaisen yrityksen Alexa Internet Inc. ylläpitämän Alexa.com-seurantaohjelmiston tilastojen mukaan. Alexa.com seuraa verkkoliikenteen määrää ja pitää tilastoja suosituimmista web-sivustoista (W3Techs 2014b).

W3Techs-sivuston pitämän tilaston mukaan ylivoimaisesti suosituin ohjelmointikieli on PHP (Kuva 5). Seuraavaksi suosituin vajaalla viidenneksen osuudella on ASP.NET, joka oikeammin on Microsoftin kehittämä ohjelmistokehys, ja käsittää yleisesti joukon .NET-kieliä. Näitä .NET-kieliä ovat esimerkiksi C#, C++ ja Visual Basic .NET. Tilastossa muut kielet ovatkin erittäin pienellä osuudella, Javan hieman kuitenkin erottuessa joukosta.



KUVA 5. Käytetyimmät palvelinpuolen ohjelmointikielät (W3Techs 2014c)

Tilasto muuttuu kuitenkin täysin, kun otetaan huomioon vain 1000 suosituinta sivustoa maailmassa. Näitä sivustoja tutkittaessa selkeästi käytetyimmäksi nousee Python, joita seuraa melko tasaisilla osuuksilla olevat Java ja JavaScript (Kuva 6). Kuvassa olevaa pylväsdiagrammia luetaan siten, että luvut kertovat 1000 suosituimman sivuston ja kaikkien sivustojen suhdetta. Eli, 1000 suosituimman sivuston joukossa Python on 15,5 kertaa yleisempi, kuin kaikkien sivustojen joukossa. Ja näin ollen PHP on 1,2 kertaa suosituimpi kaikkien sivustojen joukossa, kuin 1000 suosituimman sivuston joukossa.



KUVA 6. Käytetyimmät palvelinpuolen ohjelmointikielät, yleisimmät sivustot (W3Techs 2014a)

Ohjelmointikielen valinta projektiämme varten oli melko selkeää, samoin kuin tietokantaohjelmistoakin valitessa. Tosin tässä tapauksessa minulla ja Joelilla oli aiempaa kokemusta useammastakin ohjelmointikielestä, mutta selvästi eniten olimme tehneet PHP:llä ohjelmointia. Javakin oli kyllä tuttu kieli, ja itselleni myös Python vähäisesti, mutta näiden kielten tuntemus ja osaaminen oli aivan liian pientä tämänkaltaista projektia varten kuin mikä meillä oli edessämme. Samalla tuli säästettyä aikaa, kun pystyi heti alkamaan töihin eikä tarvinnut ensin opetella kieltä, josta oli vähäinen tuntemus pohjalta. Olisihan Javaa oppinut toki tekemällä ja ehkä suhteellisen nopeastikin, mutta kun alla oli niin vankka osaaminen PHP:stä, niin se oli järkevämpi ehdokas projektiimme. Lisäksi aiemmat ohjelmoijat olivat valinneet projektin ohjelmistokehykseksi Zend Frameworkin, joka on toteutettu PHP:llä. Näin ollen PHP:n pitäminen ohjelmointikielenä oli enemmänkin kuin järkevää, kun paras osaaminen kuitenkin on kyseiseen kieleen ja projektin perusrunkokin on sillä toteutettu.

4.5 Ohjelmistokehys

Ohjelmistokehys on web-sovellusta varten rakennettu apuväline. Se toimii projektin runkona sekä alustana ja sisältää apukirjastoja tärkeimpiä ja useimmiten käytettyjä ominaisuuksia varten. Tällaisia voivat olla esimerkiksi tietokantaan liittyvät toiminnallisuudet, eli tiedon hakeminen ja kirjoittaminen tietokantaan. Isoimmat hyödyt ohjelmistoke-

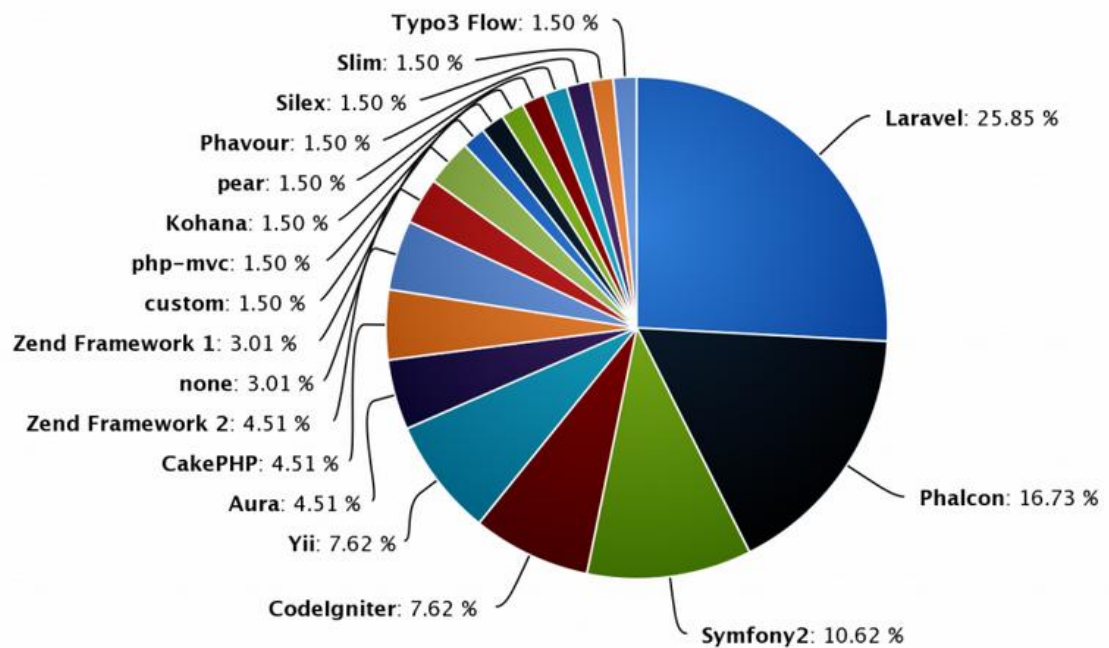
hyksen käytössä ovat siis sovelluksen nopeampi kehittäminen ja saman ohjelmointikoodin uudelleenkirjoituksen vähentäminen. Usein ohjelmistokehykset ovat olio-ohjelmointipohjaisia. Olio-ohjelmointi koostuu luokista ja olioista ja olio-ohjelmoinnin tavoite on ohjelmakoodin uudelleenkäytettävyys. Seuraavassa kappaleessa avataan olio-ohjelmointia hieman.

Otetaan esimerkiksi tilanne, jossa web-sovelluksena on verkkokauppa, ja tietokantaan voi tallentua monenlaisia tietoja: asiakkaan asiakastilin tietoja, asiakkaan tilaushistorian tiedot ja ylläpidon lisäykset ja poistot tuotetietokantaan. Ilman tällaista ohjelmistokehystä tai muuta apukirjastoa näiden jokaisen tietokantatallennuksen yhteydessä pitäisi joka kerta kirjoittaa tietokantayhteyden luontia varten sama koodinpätkä tietokannan käyttäjätunnukseen. Ja samalla joka kerta pitäisi kirjoittaa tietokantaan syöttöä varten SQL-lauseke ja ajaa se. Ohjelmistokehystä käyttäessä taas riittää, kun luodaan halutusta tietokantataulusta uusi olio, annetaan sille tiedot ja tallennetaan se. Oliota luotaessa ja tallennettaessa huolehditaan siis automaattisesti tietokantayhteyksistäkin tietokantatunnukseen, ja tieto tallentuu tietokantaan ilman erikseen kirjoitettuja SQL-lausekkeita.

Ohjelmistokehyksen valintaa pohdittaessa lähtökohdaksi voisi ensin ottaa ohjelmointikielen, millä se on kirjoitettu. Ohjelmistokehyksiä on paljon jo pelkästään web-sovelluksiakin varten, joten on välttämätöntä tehdä rajausta. Builtwith-sivuston (2014) pitämän tilaston mukaan ohjelmistokehyksiä käyttävien web-sivustojen ohjelmistokehyksistä suurin osa on kirjoitettu PHP:llä. Tilastosta voi valita 10 000, 100 000 tai miljoonaa suosituinta sivustoa koskevan tilaston, tai sitten koko Internetin kaikkia sivustoja koskevan tilaston. PHP:n asema ei muutu käytännössä juuri minkään tilaston kohdalla, vaan osuus on noin 40 prosentin luokkaa kaikissa. Selvää kakkospaikkaa pitävä ohjelmointikehyks ASP.NET MVC hallitsee kaikissa muissa tilastoissa noin viidenneksen osuutta, mutta koko Internetin ollessa kyseessä nousee ASP.NETin osuus noin kolmannekseen kaikista.

Tilastojen perusteella on PHP siis käytetyin ohjelmointikieli ohjelmistokehysten joukossa, ja kuten sanottua, pelkästään PHP:llä kirjoitettuja kehyksiäkin on erittäin paljon tarjolla. Niitä myös syntyy uusia jatkuvasti, sillä kaikilla on omat näkemyksensä siitä, millainen hyvän ohjelmistokehyksen pitäisi olla ja moni haluaa tehdä mieluummin omanlaisensa. Voi myös olla, että esimerkiksi yksinkertaisesti olemassa olevien kehysten suorituskyky ei ole tarpeeksi hyvä, ja halutaan tehdä itse tehokkaampi. Web-

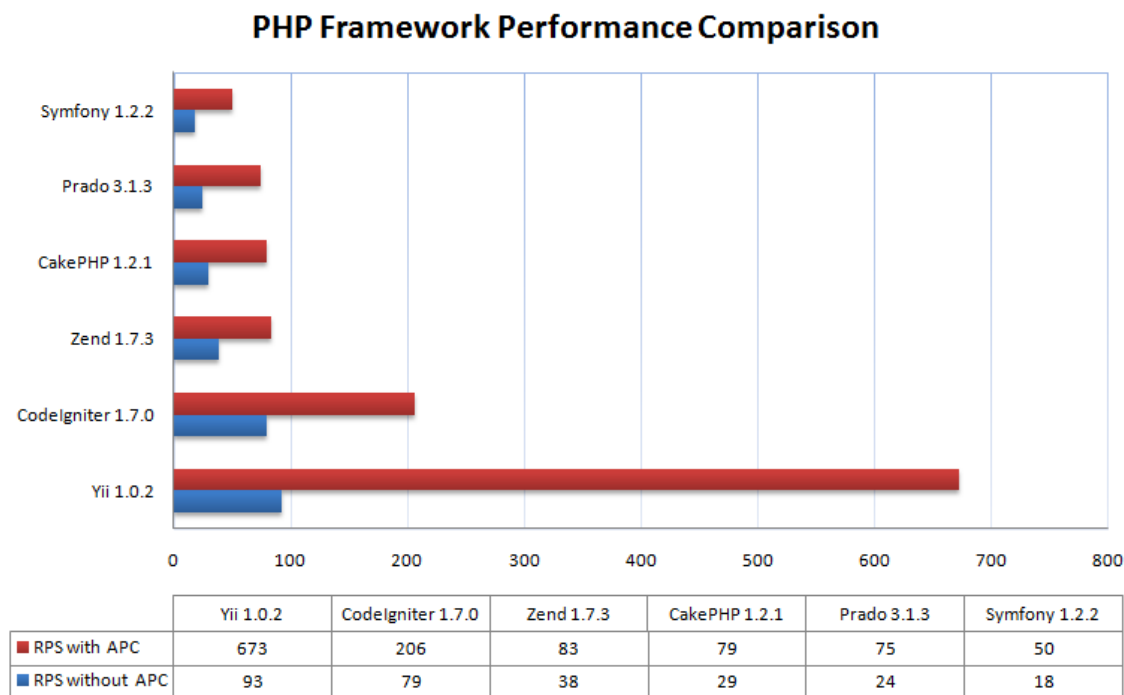
kehitykseen liittyvää sisältöä tuottava sivusto SitePoint teki vuoden 2013 lopulla kyselyn, jolla selvitettiin PHP-ohjelmistokehysten suosituimmuutta. Eri kehymiä oli kyselyn vastauksissa lähes parikymmentä, joka vahvistaa aiemmin mainitun tosiasian, että ohjelmistokehymiä esimerkiksi pelkästään PHP:lle on paljon (Kuva 7). Selvää kärki-paikkaa neljäsosan osuudella pitää Laravel ja sitä seuraavat Phalcon ja Symfony2. Näiden kolmen jälkeen tulevat tasoissa olevat CodeIgniter ja Yii kumpikin vajaalla kahdeksan prosentin osuudella. Näiden viiden kärki-ohjelmistokehymisen jälkeen onkin sitten melkoisen tasaista.



KUVA 7. Ohjelmistokehysten suosituimmuus, loppuvuosi 2013 (SitePoint 2013)

Kun ohjelmointikieli on rajattu ja suosituimmuustilastojakin tarkasteltu, on tietenkin yksi tärkeä vertailukohta suorituskyky. Suurien käyttäjämäärien sivustoilla on tärkeää, miten tehokkaasti ohjelmistokehys pystyy toimimaan. Ohjelmistokehystilaston kärki-viisikossa olleen Yii:n sivustolta löytyy suorituskykyvertailu, jossa on vertailussa muutama suosittu ohjelmistokehys (Kuva 8). Vertailussa olevista kehyksistä tosin Prado ei esiintynyt ollenkaan SitePointin kyselyssä, ja tässä vertailussa oleva Symfony on versio 1 kun SitePointin kyselyssä Symfony edusti versiolla 2. Nämäkin seikat kertovat siitä, että ohjelmistokehymiä todellakin on runsaasti, ja eri kyselyissä ja vertailuissa ei aina ole välttämättä kaikkia samoja kehymiä. Yii:n suorituskykytestissä on testattu kuutta ohjelmistokehystä kahdella eri tavalla, joko APC-laajennuksen kanssa tai ilman. APC on suorituskykyä parantava laajennus koodin optimointiin (PHP n.d.). Testin tulokset

esittävässä pylväsiagrammissa RPS tarkoittaa pyyntöjä per sekunti, eli kuinka monta ohjelmiston tekemää pyyntöä ohjelmistokehys kykenee sekunnissa suorittamaan. Yii oli suorituskyvyltään testin paras, ja APC-laajennuksen ollessa käytössä Yii oli suorastaan ylivoimainen.



KUVA 8. PHP-ohjelmistokehysten suorituskykyvertailu (Yii n.d.)

Ennen projektia en ollut käyttänyt ohjelmistokehystä lainkaan. Jonkin verran ohjelmistokehystistä kuitenkin oli tietoa. Hieman tässä tilanteessa tulikin hankittua tietoa eri kehystistä, mutta lopullista valintaa helpotti kuitenkin se, että projektiin oli jo valittuna pohjalle Zend Framework. Se oli jo tietoa hankittaessakin tuntunut yhdeltä hyvältä vaihtoehdolta ja vaikka se ei ollutkaan nopein ja suorituskykyisin, löytyi siitä silti kasapäin ominaisuuksia. Siihen tutustuminen oli kuitenkin ehkä pitkäkestoisin prosessi ennen projektin kunnollista aloittamista, sillä varsinkin Zend Framework on erittäin laaja ohjelmistokehys ja se sisältää todella paljon ominaisuuksia. Osan ominaisuuksista oppiikin vasta matkan varrella kehystä käyttäessä. Hyvä harjoitus itselle oli erään ylläpitämäni web-sivuston ohjelmoiminen alusta asti uudelleen käyttämään Zend Frameworkia. Paljon siinä tuli opittua, mutta kuten sanottua, paljon opin myös projektin aikana.

5 YHTEENVETO

Palvelinohjelmiston valinnassa kannattaa ottaa yhdeksi tärkeäksi lähtökohdaksi ohjelmiston yleisyys ja helppo tiedon saatavuus ongelmatilanteissa. Jos projektissa kohtaakin jonkin ongelman ohjelmiston käytössä, tai haluaisi esimerkiksi konfiguroida jonkin uuden ominaisuuden käyttöön, on tietysti hyödyllistä, että apu on nopeasti saatavilla. Tätä ajatellen Apache HTTP Server on hyvä vaihtoehto. Vaikkakin se oli projektissa valmiina jo käytössä, ja siitä itsellänikin on eniten kokemusta, niin tilastot puhuvat myös puolestaan. Aktiivisia sivustoja, joissa on Apache käytössä, on noin neljä kertaa enemmän kuin niitä, joissa on Microsoftin IIS tai Nginx. Microsoftin IIS ei itsessään ole maksullinen, mutta tulee Windows-käyttöjärjestelmän mukana, joka vaatii lisenssin, joten avoimen lähdekoodin projekteihin se ei sovellu. Apachen suosituimmuus tuo mukanaan niin ison käyttäjäjyhteisön ja laajat keskustelusivustot aiheesta, että tukea on saatavilla reilusti.

Tietokantaohjelmiston valinnassa on toki myös yhtenä tärkeänä lähtökohtana ohjelmiston yleisyys ja suosituimmuus. Kehittäminen on nopeampaa ja sujuvampaa, kun minäkään ongelman takia ei tarvitse keskeyttää tekemisiä kovin pitkäksi aikaa, vaan apu on lähellä. Lisäksi varsinkin käyttäjämääriltään suuressa projektissa kannattaa ottaa huomioon tietokantaohjelmiston suorituskyky ja selvittää, mitä isot ja suositut sivustot maailmalla käyttävät tietokantaohjelmistonaan. Lisäksi suurten käyttäjämäärien projektia varten valittavan tietokannan ominaisuuksilla on merkitystä, kuten aiemmassa tietokantaluvussa tuli esille. Esimerkiksi SQLite saattaa muodostua pullonkaulaksi, kun se ei tue yhtäaikaista tietokantatalennuksia. Näitä mainittuja seikkoja tarkasteltaessa, on MySQL helppo valinta. Kuten tietokantaluvussakin tuli todettua, se on toiseksi yleisin tietokanta ja käytössä isoilla sivustoilla, kuten muun muassa Wikipediassa ja Facebookissa.

Versionhallintaa suunniteltaessa ei ole mitään yhtä ja oikeaa valintaa olemassa. Se toki on selvää, että paikallinen versionhallinta on käytännössä vain pieniin ja yhden ihmisen projekteihin soveltuva. Trendi kuitenkin on projektimme ajoista asti ollut se, että keskitetyn versionhallinnan suosio on laskusuunnassa ja vastaavasti hajautettu versionhallinta on vain lisäämässä suosiotaan. Syykin on selvä, kun pohditaan näiden kahden välisiä hyöty- ja haittapuolia. Näiden perusteella hajautettu versionhallinta on juuri sellainen, kuin versionhallinnan tuleekin olla. Hajautetun versionhallinnan käyttämisessä katastro-

fimaiset ongelmatilanteet vähenevät ja sen käyttäminenkin on keskitettyä versionhallintaa nopeampaa sen toimiessa vertaisverkkojen tapaan.

Ohjelmointikielen valintaan ei ole, kuten ei ole versionhallinnan valintaankaan, yhtä parasta vaihtoehtoa. Paljon riippuu ensinnäkin ohjelmoijan osaamisesta ja kokemuksesta ohjelmointikielten parissa. Jos jokin kieli on erittäin hyvin hallussa, miksi lähteä jostakin toista opettelemaan tyhjästä. Toisaalta taas, kuten ohjelmointikieltä koskevassa luvussakin tuli esille, hyvä ohjelmoija omaksuu uudenkin kielen suhteellisen nopeasti. Tällaisessa tilanteessa ei tietenkään ole haitaksi ottaa käyttöön itselle uutta ohjelmointikieltä. Palvelinympäristö voi tietysti myös olla rajoittavana tekijä. Ennen ohjelmointikielen valintaa pitää selvittää mitä kaikkea palvelin tukee, voiko kyseisellä palvelimella ohjelmoida esimerkiksi Javaa tai PHP:tä käyttäen. Jos tutkii tilastojen valossa suosituimpia ohjelmointikieliä, parhaiten web-sovelluksen kehittämiseen soveltuvia ovat ASP.NETin kielet ja PHP. Kuitenkaan mitään absoluuttista totuutta ei näidenkään kahden välillä valitsemiseen ole. Molemmilla kielillä voi tehdä pitkälti samoja asioita ja molempia kieliä voi käyttää ilmaiseksikin. Internetin keskustelupalstoja lukiessa huomaa hyvin, kuinka ohjelmointikielten puoltajat ovat pääasiassa kyseisen ohjelmointikielen taitajia. Eli pääasialliset PHP-ohjelmoijat ovat ehdottomasti PHP:n valinnan kannalla ja sama taas ASP.NET-osaajien kohdalla. Ohjelmointikielen valinnassa kannattaa siis ihan ensimmäiseksi tarkastella kielen soveltuvuutta palvelinympäristöön, ja vasta sen jälkeen valita omalle osaamiselle sopiva ohjelmointikieli.

Ohjelmistokehityksen valinnassa ensimmäinen kysymys on se, onko sellaisella lainkaan tarvetta projektissa. Suurta haittaa ei kehityksen käyttöönotosta tietenkään ole, mutta esimerkiksi pienehkössä projektissa saattaa suurin osa kehityksen tuomista ominaisuuksista jäädä käyttämättä. Jos kuitenkin kyseessä on laajempi projekti ja apukirjastolle näyttäisi olevan tarvetta, kehityksen valintaa helpottaa, mikäli ohjelmointikieli on jo valittu. Ohjelmointikielen valinnalla voi hyvin rajata niiden ohjelmointikehysten määrän, joihin tutustua. Jos ohjelmointikieltä ei ole vielä valittu, niin silloin tutustumisen arvoisia kieliä ja niiden kehityksiä ovat tilastojenkin valossa suosituimmat kielet, eli PHP ja .NET-kielet. Kun on saanut ohjelmointikielen valinnalla rajattua ohjelmistokehityksien määrää, on seuraavaksi hyvä tutustua tarjolla oleviin kehityksiin. Niihin kannattaa tutustua huolellisesti, sillä varsinkin koskaan ohjelmistokehityksiä käyttämättömälle tiedossa on paljon opettelua ja uudenlaisen ohjelmointitavan sisäistämistä. Kun kehityksiin tutustuu, kannattaa ainakin selvittää kuinka yleisesti käytetty kehitys on, kuinka hyvin se on

dokumentoitu ja kehitetäänkö sitä edelleen aktiivisesti. Näiden seikkojen toteutuessa on myös todennäköistä, että kehykseltä löytyy aktiivista käyttäjäkuntaa. Aktiivinen kehittäminen on tärkeää, sillä ohjelmointikielet ja ohjelmistokehykset kehittyvät kuitenkin kaiken aikaa. Järkevämpi on ottaa käyttöön sellainen, joka ei edusta jo vanhentunutta tekniikkaa. Alun käyttöönottovaihetta helpottaa, kun on saatavilla hyvät dokumentaatiot kehyksen toiminnasta, ja jos vielä jotakin jää silti epäselväksi, voi apua löytää kehykseen liittyviltä keskustelualueilta.

Projektissamme olisi ollut iso hyöty, jos sitä suunnittelemaan lähettäessä olisi ollut jonkinlainen opas tai jokin muu hyvä teos siitä, mitä kaikkea tällaisen ison web-sovelluksen kehittäminen vaatii. Sellaista ei kuitenkaan ollut, ja paljon kuluikin aikaa selatessa Internetistä tietoa ja tilastoja kaikesta mahdollisesta. Tästä huolimatta projekti kuitenkin lähti lopulta liikkeelle, vaikkakin hitaasti. Alkuvaiheet menivät siis tutustussa uusiin kehittämistyökaluihin ja niiden käytön opetteluun.

Alun tutkimus- ja opetteluvaiheen jälkeen pidimme Espoossa Laurea-ammattikorkeakoulussa Dev Day -tapahtuman, mihin pyrimme saamaan kaikki ne projektimme jäsenet, jotka vain aikatauluiltaan pääsivät paikalle. Tapahtumassa oli tarkoitus tavata muita kehittäjiä kasvotusten ja sopia sekä lyödä lukkoon kehitysmenetelmät ja -tavat. Tapahtuman jälkeen projekti saatiin ulkomaailmaan projektin jäsenten saataville, kun se laitettiin SourceForgen versionhallintaan ja myös samalla projekti lähti varsinaisesti kunnolla käyntiin. Projekti sujui alusta asti pääasiassa sujuvasti ja ongelmitta. Toki kehittäjillä oli erilaiset lähtökohdat omien taitojensa suhteen, ja jotkut joutuivat ehkä enemmän alussa käyttämään asioiden opetteluun aikaa kuin toiset.

Näin jälkikäteen projektia muistellessa ja itsekkin ohjelmoinnissa kehittyneenä löytyy kyllä asioita, joita tekisi toisin, mikäli tämä sama projekti pitäisi aloittaa nyt tyhjästä. Ensinnäkin olisi ollut alun kannalta järkevämpää, tehokkaampaa ja nopeampaa, jos projektin kehittäjien yleinen kokemustaso ohjelmoinnista olisi ollut korkeampi, varsinkin minulla ja Joelilla, kun kuitenkin olimme vetovastuussa projektista. Toisaalta taas osa projektin tarkoitusta oli kokeilla tällaista yhteisöllistä kehittämistä tutkien miten se sitten toimii käytännössä. Ja alkoihan kehitystyö varsinkin loppupuolella projektia olla jopa sujuvaakin. Dev Day -tapahtumaa nyt järjestäessä toisin enemmän esille muun muassa koodin laaduntarkastukseen liittyviä seikkoja, eli muuttujien ja funktioiden nimeämissääntöjä, koodin kommentointisääntöjä ja olisin myös kiinnittänyt huomiota

dokumentointiin muutenkin enemmän. Testaaminen oli myös melko olematonta: käytännössä kehittäjä testasi itse tuotostaan sitä tehdessään. Ei se projektissa kuitenkaan ongelmaksi muodostunut, varsinkin kun kehittäminen tapahtui paikallisesti kehittäjien omilla koneilla. Näin kehitettävä ominaisuus sai alkuvaiheessa ollakin keskeneräinen ja rikki, kun ei se kuitenkaan vielä siinä vaiheessa näkynyt muille. Pitkällä tähtäimellä tämä ei kuitenkaan ole suinkaan järkevä tapa. Ensinnäkään testaaminen ei ole järkevää vain yhden henkilön tekemänä, koska varmasti jotakin jää huomaamatta, eikä varsinkaan ole hyvää testaamista, jos tämä yksi ja ainoa testaaja on kehittäjä itse.

Oli projektissa paljon hyviäkin asioita. Projektin aikana opittiin valtavasti uutta asiaa ohjelmoinnista, muun muassa olio-ohjelmoinnin taidot paranivat suuresti ja Zend Frameworkin käyttäminen oli itselleni ensimmäinen kokemus ohjelmistokehyksen käytöstä. Vaikka joitakin asioita tekisin ehkä nykyisin toisin, eivät nekään asiat olleet kuitenkaan täysin huonosti tehtyjä. Projekti opetti hyvin perusteita esimerkiksi versionhallinnasta ja varmuuskopioinnista sekä lisäsi yleisesti tietämystä web-sovellusten kehittämisestä, kun asioita joutui todella paljon penkomaan alussa. Edelleen web-sovellusten parissa työskennellessäni uskon tästä projektista olleen paljon hyötyä.

LÄHTEET

Builtwith. 2014. Framework Usage Statistics. Luettu 26.10.2014.
<http://trends.builtwith.com/framework>

Chipkin. n.d. SQLite vs MySQL (short summary). Luettu 24.10.2014.
<http://www.chipkin.com/sqlite-vs-mysql-short-summary/>

DB-Engines. 2014. DB-Engines Ranking. Kuvakaappaus 19.9.2014.
<http://db-engines.com/en/ranking>

DB-Engines. n.d. Method of calculating the scores of the DB-Engines Ranking. Luettu 19.9.2014. http://db-engines.com/en/ranking_definition

Eclipse Foundation. n.d. About the Eclipse Foundation. Luettu 17.10.2014.
<http://www.eclipse.org/org/>

Eclipse Foundation. 2009. Eclipse Survey 2009 report. Luettu 17.10.2014
http://www.eclipse.org/org/press-release/Eclipse_Survey_2009_final.pdf

Eclipse Foundation. 2011. Eclipse Survey 2011 report. Luettu 17.10.2014
http://www.eclipse.org/org/community_survey/Eclipse_Survey_2011_Report.pdf

Git. n.d. About version control. Luettu 14.10.2014.
<http://git-scm.com/book/en/Getting-Started-About-Version-Control>

IRC-Junkie.org. 2012. Freenode is still growing. Luettu 24.10.2014.
<http://www.irc-junkie.org/2012-03-04/freenode-is-still-growing/>

MySQL. n.d. MySQL Customers. Luettu 19.9.2014. <http://www.mysql.com/customers/>

Netcraft. 2014. July 2014 Web Server Survey. Luettu 28.9.2014.
<http://news.netcraft.com/archives/2014/07/31/july-2014-web-server-survey.html>

Netcraft. n.d. How many active sites are there? Luettu 28.9.2014.
<http://www.netcraft.com/active-sites/>

PHP. n.d. Introduction. Luettu 2.11.2014.
<http://php.net/manual/en/intro.apc.php>

Pingdom. 2012. IRC is dead, long live IRC. Luettu 24.10.2014.
<http://royal.pingdom.com/2012/04/24/irc-is-dead-long-live-irc/>

Rogowsky, M. 2014. Without Much Fanfare, Apple Has Sold Its 500 Millionth iPhone. Luettu 28.9.2014.
<http://www.forbes.com/sites/markrogowsky/2014/03/25/without-much-fanfare-apple-has-sold-its-500-millionth-iphone/>

SitePoint. 2013. Best PHP Frameworks for 2014. Kuvakaappaus 2.11.2014.
<http://www.sitepoint.com/best-php-frameworks-2014/>

Skerrett, I. 2012. Eclipse survey 2012 report [final]. Luettu 17.10.2014.
<http://www.slideshare.net/IanSkerrett/eclipse-survey-2012-report-final>

Skerrett, I. 2013. Eclipse Community Survey Report 2013. Luettu 17.10.2014.
<http://www.slideshare.net/IanSkerrett/eclipse-survey-2013-report-final>

Skerrett, I. 2014a. Eclipse community survey 2014 v2. Luettu 17.10.2014.
<http://www.slideshare.net/IanSkerrett/eclipse-community-survey-2014>

Skerrett, I. 2014b. Eclipse community survey 2014 v2. Kuvakaappaus 17.10.2014.
<http://www.slideshare.net/IanSkerrett/eclipse-community-survey-2014/24>

SourfeForge. n.d. About. Luettu 9.11.2014.
<http://sourceforge.net/about>

SQLite. n.d. Well-Known Users of SQLite. Luettu 19.9.2014.
<https://www.sqlite.org/famous.html>

Statistic Brain. 2014a. Android Phone Statistics. Luettu 28.9.2014.
<http://www.statisticbrain.com/android-phone-statistics/>

Statistic Brain. 2014b. iPhone 5 Sales Statistics. Luettu 28.9.2014.
<http://www.statisticbrain.com/iphone-5-sales-statistics/>

W3Techs. 2014a. Server-side programming languages ranked by usage on top websites. Kuvakaappaus 3.10.2014.
http://w3techs.com/technologies/topsite/programming_language

W3Techs. 2014b. Technologies Overview. <http://w3techs.com/technologies>

W3Techs. 2014c. Usage of server-side programming languages for websites. Kuvakaappaus 3.10.2014.
http://w3techs.com/technologies/overview/programming_language/all

Yii Framework. n.d. Performance of Yii. Kuvakaappaus 2.11.2014.
<http://www.yiiframework.com/performance/>