jamk.fi

PDX-ohjelmiston modernisointi

Juho Hautamäki

Opinnäytetyö
Toukokuu 2020
Tekniikan ala
Insinööri (AMK), tietotekniikka
Tietoverkkotekniikka

Jyväskylän ammattikorkeakoulu
JAMK University of Applied Sciences

jamk.fi

Kuvailulehti

Tekijä(t)	Julkaisun laji	Päivämäärä
Juho Hautamäki	Opinnäytetyö	Toukokuu, 2020
	Sivumäärä 41	Julkaisun kieli Suomi
		Verkkojulkaisulupa myönnetty: x

Työn nimi
PDX-ohjelmiston modernisointi

Tutkinto-ohjelma
Tieto- ja viestintätekniikka

Työn ohjaaja(t)
Ari Rantala

Toimeksiantaja(t)
Miro Sundholm

Tiivistelmä

Opinnäytetyön tavoitteena oli seurata PDX-ohjelmiston kehitystä kohti modernimpaa rajapintakeskeistä toteutusta sekä selvittää uuden toteutuksen etuja verrattuna vanhaan ratkaisuun. Arvioinnin seurauksena haluttiin ymmärtää nykyaikaisemman modulaarisen ohjelmistosuunnittelun etuja verrattuna vanhanaikaisempaan monoliittiseen toteutukseen.

Eroja toteutustyylien välillä ei arvioitu ainoastaan ohjelmistokehityksen kannalta vaan tavoitteena oli saada laajempi kuva myös eduista, jotka voidaan saavuttaa taustajärjestelmien, käyttäjäkokemuksen ja kehitysprosessin puolella. Opinnäytetyössä projektia seurattiin kehitystiimin jäsenen näkökulmasta, joka osallistui projektin suunnitteluun ja toteutukseen osana pientä kehitystiimiä.

Työn aikana tarkastellut kehitysprojektit keskittyivät PDX-ohjelmiston uuden käyttöliittymän julkaisuun, sekä sen vaatimaan rajapintakehitykseen. Ratkaisujen suunnittelua ja toteutusta käydään opinnäytetyössä läpi ainoastaan pintapuolisesti ja tutustutaan ohjelmiston uudistuksen kannalta keskeisiin tekniikoihin ja ratkaisuihin.

Uuden käyttöliittymän ensimmäinen julkaisu koski ainoastaan ohjelmiston useimmin käytettyjä ominaisuuksia, mutta rajapintakehitystä lähestyttiin tulevaisuuden tarpeiden näkökulmasta. Uusi käyttöliittymän ensimmäinen osa julkaistiin ohjelmiston käyttäjille marraskuussa 2019.

Tuloksien valossa voitiin todeta uudemmalla lähestymistavalla olevan useita merkittäviä etuja. Toteutuksen aikana havainnoitiin, että ohjelmistonrakenteen muutos voi auttaa myös kehittämään kehitys – ja julkaisuprosesseja. Päätelmissä voidaankin todeta uuden toteutustavan johtaneen lukuisiin parannuksiin PDX-kokonaisuudessa.

Avainsanat (asiasanat)

Rajapintakehitys, Sovelluksen modernisointi, Ohjelmistokehitys

Muut tiedot

jamk.fi

Description

Author(s) Juho Hautamäki	Type of publication Project Report	Date May, 2020
		Language of publication: Finnish
	Number of pages 41	Permission for web publication: x

Title of publication
Modernization of PDX software

Degree programme
Information Technology

Supervisor(s)
Ari Rantala

Assigned by
Miro Sundholm

Abstract

The purpose of this project report was to follow the development of the PDX system towards a more modern interface-based approach and to evaluate the advantages of this new implementation.

The goal of the evaluation was to understand the advantages of a modern modular implementation as compared to an outdated monolithic approach. The evaluation does not concern only software development process but also attempts to include user-experience and back end systems. The evaluation is carried out from the perspective of a member of the development team who participated in the planning and implementation of the changes.

The development projects observed during the evaluation of the modernization project focused on the development of a new user interface for the PDX system. The thesis observes the chosen technologies and solutions on the surface level and attempts to explain the technologies used.

The first release of the user interface project contained only the most commonly used parts of the system. The API was developed with the goal of replacing the entire logic of the program and moving it to the API.

The first parts of the user interface were released to customers in November of 2019.

The report found significant advantages in the new approach. During the project, it was observed that the changes to the program structure also helped with the development of new development and release practices. In conclusion, it can be asserted that the new implementation led to numerous improvements in the PDX system.

Keywords/tags (subjectshttp://vesa.lib.helsinki.fi/))

API development, Software modernization, software development

Miscellaneous

Sisältö

Kuviot

Lyhenteet

AD	Active Directory
API	Application programming interface
AWS	Amazon Web Services
CLI	Command Line Interface
CSS	Cascading Style Sheets
DNS	Domain Name System
DOM	Document Object Model
HTML	Hyper Text Markup Language
IDE	Integrated development environment
IRC	Internet Relay Chat
MVC	Model–view–controller
ORM	Object-relational mapping
URI	Uniform Resource Identifier
VPN	Virtual Private Network
XML	Extensible Markup Language

1 Työn lähtökohdat

1.1 Tausta

Opinnäytetyö sai alkunsa kesällä 2017, kun minut rekrytoitiin LKI-asiantuntijapalvelut-nimisen yrityksen palvelukseen ohjelmistokehittäjänä. Toimenkuvani koostui ensisijaisesti REST-rajapintakehityksestä PDX-ohjelmistolle.

Työtä suunniteltiin toteutettavaksi osana PDX-REST-rajapinnan kehitystä muun työn ohella. Arvioitiin, että työn sisältö voisi koostua suhteellisen pienestä rajapinnan laajennusprojektista. PDX-REST-rajapinnan uudistusprojekti oli tällöin vielä projekti, jolle ei ollut suunniteltu aikataulua tai selkeitä tavoitteita. REST-rajapintaan oli tarkoitus kehittää valmiutta käyttöliittymäuudistukselle, muun kehityksen ohella. Rajapintakehityksen pääpaino yrityksessä oli tällöin lomakepohjaprojektissa, joka oli aloitettu vuonna 2015 erään suuren asiakkaan kanssa.

Yrityskaupan myötä PDX:n kehityssuunnitelmat muuttuivat sisältämään laajamittaisen rajapinnan kehityksen, jonka seurauksena kehityksestä ei voinut enää eriyttää yksittäistä projektia opinnäytetyöksi. Tämän muutoksen seurauksena työn aihetta muokattiin keskittymään toteutustapojen eroihin ja uudistuksessa tehtyihin rakenteellisiin muutoksiin ja niiden etuihin. Muutoksen ansiosta raportissa voidaan ottaa huomioon myös muiden kehittäjien osuus projektissa. Painopistettä voidaan täten siirtää poispäin ohjelmoinnista ja otetaan mukaan myös verkkotekniikan elementtejä.

Koska PDX on tuotantokäytössä oleva kaupallinen ohjelmisto, ohjelmakoodin osien liittäminen työhön olisi ollut hankalaa, joten näiden liittämisen välttämiseksi ratkaisuja käsitellään mahdollisimman pintapuolisesti.

Opinnäytetyössä käsitellään REST Application programming interface (API) -kehitystä, PDX:n käyttöliittymän siirtymistä käyttämään rajapintaa, kehitysputken muutoksia, sekä uuden lähestymistavan etuja verrattuna aiempaan toteutukseen.

1.2 Toimeksiantaja ja yhteistyökumppanit

Opinnäytetyön toteutuksen aikana toimeksiantajana toimineeseen yritykseen LKI-asiantuntijapalvelut, kohdistui yrityskauppa. Lokakuussa 2017 Sanoma- konsernin alaisuudessa toimiva Oikotie Asunnot Oy osti LKI:n PDX-liiketoiminnan (Oikotie Asunnot Oy ostaa liiketoimintakaupalla LKI-Asiantuntijapalvelut Oy:n liiketoiminnan 2017).

Kaupan seurauksena ja siitä aiheutuneiden muutosten ja viivästysten vuoksi sovittiin PDX-kehitystiimin johtajan kanssa, että hän toimisi henkilökohtaisesti toimeksiantajana.

LKI-asiantuntijapalvelut

LKI-asiantuntijapalvelut on vuonna 2001 perustettu pienyritys, jonka liiketoiminta koostui PDX+ ja PDX Construction ohjelmistojen kehityksestä ja myynnistä. Yritys työllisti kahdeksan henkilöä yrityskaupan aikaan. PDX-kehitystiimi koostui neljästä henkilöstä.

Sanoma Oyj

Sanoma Oyj on suuri suomalainen mediakonserni, joka on listattu Helsingin pörssissä. Konsernilla on toimintaa useammassa maassa, ja se työllistää arviolta 4400 henkilöä. Konserniin liikevaihto oli vuonna 2018 noin 1,3 miljardia euroa. (Sanoma vuosikatsaus 2018.)

Oikotie Oy

Oikotie Asunnot Oy oli Oikotie Oy:n alaisuudessa toimiva yritys, joka ylläpitää Oikotie asunnonvälitysportaalia. Työn toteutuksen aikana Oikotie Asunnot Oy fuusioitiin osaksi Oikotie Oy:tä. Oikotie Oy:n liikevaihto oli noin 11,2 miljoonaa euroa vuonna 2018. (Oikotie Oy n.d.).

1.3 Opinnäytetyön tavoitteet

Opinnäytetyön tavoitteena oli seurata PDX-ohjelmiston kehitystä kohti rajapintamallista toteutusta ja dokumentoida uudella toteutustavalla saavutettuja etuja. Työssä tutustutaan ohjelmistoon tehtäviin rakenteellisiin muutoksiin ja niiden vaikutuksiin

ohjelmiston toiminnassa sekä taustajärjestelmissä. Työssä käsitellään lyhyesti myös muutosten vaikutusta kehitystyöhön. Ohjelmiston kehitystä seurataan kehitystiimin jäsenen näkökulmasta.

1.4 PDX-ohjelmisto

PDX on selainpohjainen kiinteistönvälitysjärjestelmä, joka on jaettu kiinteistönvälittäjille kohdistettuun PDX+:aan ja rakennusliikkeille kohdistettuun PDX Constructioniin. PDX-ohjelmistoa voisi luonnehtia alansa markkinajohtajaksi Suomessa lähinnä vähäisen kilpailun vuoksi (PDX n.d). PDX:n suurin kilpailija on Etuovi-portaalisivuston ylläpitämä KIVI. Etuovi-palvelun omistaa Alma Media -konserni.

PDX-ohjelmistoja on kehitetty yli 20 vuotta, ja sen toteutus on tänä aikana muuttunut kokonaan vain kerran, kun ohjelmisto muuttui työpöytäsovelluksesta web-pohjaiseksi. Ohjelmiston alkuperäinen kehittäjä on edelleen mukana PDX-tiimissä, mutta ei osallistu enää ohjelmiston kehitykseen. Nykyinen kehitys repositorio perustettiin vuonna 2006, jolloin PDX-projekti siirtyi käyttämään git-versionhallintaa.

Kiinteistönvälitysala on Suomessa vahvasti säännöstelty, minkä vuoksi PDX:n kehityksessä on otettava huomioon välittäjille lain asettamat vaatimukset. Ohjelmiston tekniseen toteutukseen vaikuttavat eritoten dokumenttien säilytystä koskevat säädökset.

2 Teknologiat

2.1 Käytetyt teknologiat

2.1.1 Yleistä

Projektissa käytetyt teknologiat määräytyvät pitkälti PDX:n jo olemassa olevan toteutuksen mukaan. Uusia teknologioita kumminkin otetaan projektissa käyttöön tarpeen mukaan. Työ käsittelee pääosin PDX-käyttöliittymäprojektin osana tai seurauksena tehtyjä muutoksia ohjelmistoon, joten tässä osiossa keskitytään pääasiallisesti teknologioihin, jotka liittyvät näihin toteutuksiin.

Luvussa 2.2 käsitellään lyhyesti myös projektissa käytettyjä kehitys - ja hallintatyökaluja.

2.1.2 PHP

PHP on rekursiivinen akronyymi sanoille PHP: Hypertext Preprocessor. Se on laajasti käytetty palvelinpuolinen ohjelmointikieli, joka soveltuu erityisesti web-kehitykseen.

PHP-koodi voidaan upottaa osaksi Hyper Text Markup Language (HTML) sivua, jossa se prosessoidaan sivun näytön yhteydessä. PHP eroaa JavaScript-kielestä merkittävästi siinä, että sen tulkinta toteutetaan palvelimen puolella ja tulokset lähetetään asiakasohjelmalle (What is PHP? n.d.)

Rasmus Lerdorf kehitti PHP-kielen vuonna 1995. Versiosta 5 alkaen kieli on tukenut olio-ohjelmointia (History of PHP n.d.)

PHP-kieli on helppo oppia, mutta se tarjoaa monipuoliset kehitysmahdollisuudet.

2.1.3 JavaScript

JavaScript on alkujaan web-sivujen toiminnallisuuden laajentamiseen kehitetty kevyt ohjelmointikieli. JavaScript tulkitaan suoritushetkellä käyttäjän selainohjelmassa, minkä vuoksi JavaScript-koodi on luettavissa käyttäjälle. JavaScriptin ominaisuudet määritetään ECMAScript-standardissa, jota tukevat kaikki nykyaikaiset selaimet. JavaScript voidaan nykyään suorittaa myös selaimen ulkopuolella esimerkiksi Node.js- tai Adobe Acrobat -ohjelmassa. (An Introduction to JavaScript 2020.)

2.1.4 React

React on JavaScript-kirjasto, jota kehittää ja ylläpitää Facebook. Se on tarkoitettu käyttöliittymän rakentamiseen. React-kehityksen sydämessä ovat komponentit. Komponentti on itsenäinen moduuli, joka tuottaa jonkin tulosteen. React-komponentit voivat sisältää toisia React-komponentteja tulosteessaan.

Toisin kuin monet aiemmat JavaScript-kirjastot, React ei käsittele dataa suoraan selaimen Document Object Modelissa (DOM) vaan sen virtuaalisessa vastineessa. Reactin virtuaalinen DOM on olemassa kokonaisuudessaan muistissa ja esittää selaimen

DOMia. Virtuaalisen DOMin päivityksen jälkeen React päättelee, mitä muutoksia on tehtävä selaimen DOMiin. (Lerner n.d.)

2.1.5 HTTP-metodit

HTTP-protokolla (Hypertext Transfer Protocol) määrittelee yhdeksän yleistä metodia käytettäväksi HTTP-kutsuissa. REST-rajapinnat tukevat näistä usein vain viittä. Metodilla määritellään, mitä pyynnöllä halutaan tehdä. Viisi tyypillisesti REST-rajapinnoissa käytettyä metodia tai verbiä ovat

1. GET – On tiedon hakemiseen tarkoitettu metodi, jota käytetään jonkin tietyn resurssin hakemiseen palvelimelta. Operaatiota pidetään turvallisena, sillä se on idempotentti. Tämä tarkoittaa, että kutsun tulisi palauttaa aina sama tulos, vaikka sen tekisi monta kertaa.
2. POST – Metodia käytetään luomaan uusi resurssi palvelimella. Sen käyttötarkoitukset ovat tulkinnanvaraisempia kuin muiden metodien. POST-metodi on HTTP-protokollan ainoa metodi, joka ei ole idempotentti.
3. PUT – On tiedon tallentamiseen ja päivittämiseen tarkoitettu metodi, jota käytetään tallentamaan pyynnön sisältämä resurssi osoitteeseen, johon pyyntö on tehty. Myös PUT-metodi on idempotentti, ja sen tulisi johtaa aina samaan tulokseen.
4. DELETE – Kutsu on tarkoitettu nimensä mukaan resurssien poistamiseen palvelimelta.
5. PATCH – metodi on tarkoitettu resurssien päivittämiseen ja toimii samankaltaisesti PUT-metodin kanssa. Toisin kuin PUT-metodissa käyttäjä määrittelee ainoastaan muutettavat attribuutit resurssista. Kaikki selaimet eivät tue PATCH-metodia. (HTTP Methods n.d.)

Näitä viittä metodia nimitetään usein yhdessä CRUD- operaatioiksi (create, retrieve, update, delete). Muita usein käytettyjä metodeja ovat OPTIONS ja HEAD. (HTTP Methods n.d.)

2.1.6 REST

REST on lyhenne sanoista Representational State Transfer. Se ei ole teknologia vaan kokoelma sääntöjä, jotka määrittävät, miltä API näyttää. Rajapinnan voidaan sanoa olevan RESTful, jos nämä 6 periaatetta täyttyvät:

1. Asiakas – Palvelin Rajapinta erottaa käyttöliittymän data säilöstä. Tämä mahdollistaa eri käyttöliittymä toteutuksen useammalla alustalla ja yksinkertaistaa palvelin komponentteja.
2. Tilaton – Asiakkaan lähettämät pyynnöt sisältävät kaiken tiedon, jota tarvitaan pyynnön toteuttamiseen.
3. Pyynnön vastauksen täytyy, määritellä onko se tallennettavissa välimuistiin.

4. Yhtenäinen rajapinta – Rajapinnat on suunniteltava yhtenäisiksi, jotta koko järjestelmä pysyy selkeänä.
5. Monikerroksinen järjestelmä – Asiakkaan ei tarvitse tietää mihin osaan tai kerrokseen ohjelmaa se tekee pyyntöjä.
6. (vapaaehtoinen) Rajapinnan kautta voidaan tarvittaessa siirtää ohjelmakoodia. (What is REST n.d.)

RESTin pääasiallinen yksikkö on resurssi. Uniform Resource Identifier (URI) on merkkijono, joka osoittaa resurssin sijainnin tai nimen. Mikä vain informaatio, joka voidaan nimetä, voi olla resurssi. REST-periaatteiden mukaan resurssin tulee olla itsemäärittelevä eli asiakas ohjelman ei pitäisi tarvita tietää, onko resurssi esimerkiksi henkilö tai laite.

Rajapinnalta voidaan tyypillisesti kysyä resurssin nykyistä tilaa käyttäen HTTP-protokollan GET-metodia, mutta REST ei vaadi HTTP-metodien käyttöä. Se vaatii vain, että rajapinta on yhdenmukainen. Toisin sanoen PUT-metodin ei ole pakko tarkoittaa päivitystä, kunhan se vain aina tarkoittaa samaa toimenpidettä.

Tyypillisesti REST-rajapinnat käyttävät HTTP-protokollan eri metodeja. Niiden käyttö ei kumminkaan ole pakollista, jotta rajapinta olisi RESTful. (What is REST n.d.)

2.1.7 MVC

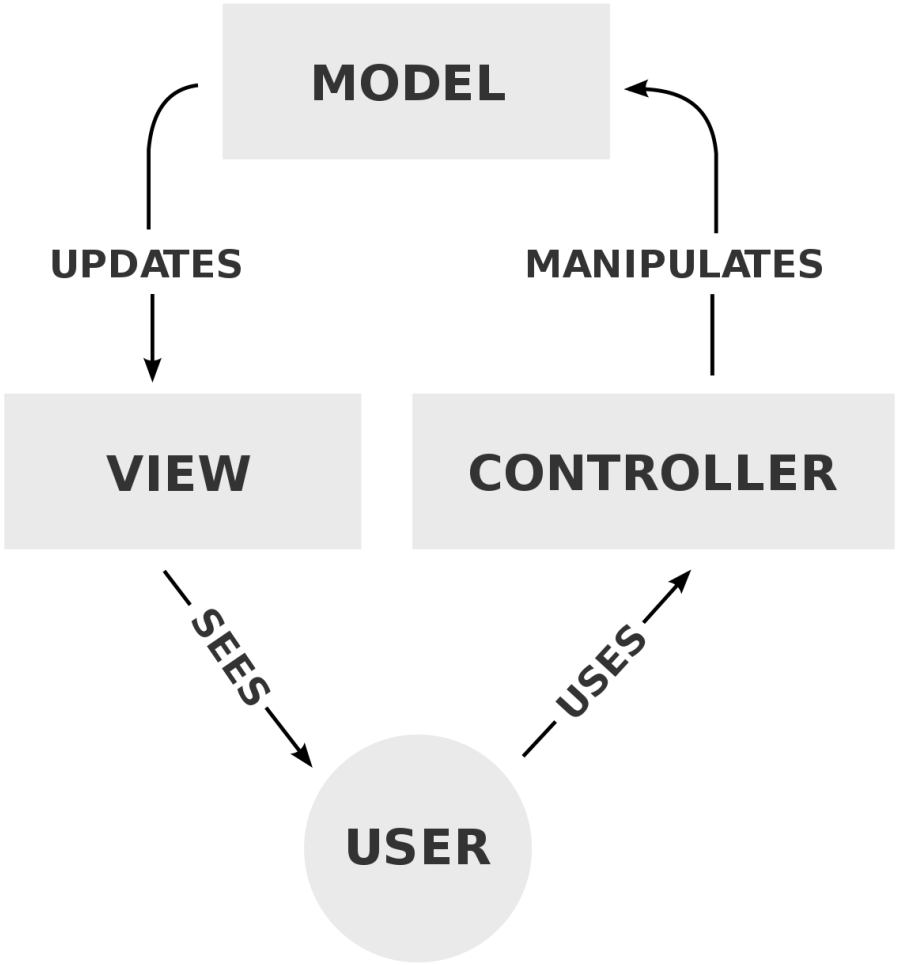
Model-View-Controller (MVC) on ohjelmiston toteutustapa, joka erottaa ohjelman kolmeen loogiseen pääkomponenttiin. Tämän tarkoitus on, erottaa miten informaatio esitetään ohjelman sisällä, siitä miten se esitetään käyttäjälle. Tyypillisesti mallia on seurattu käyttöliittymällisissä työpöytäohjelmissa, mutta sen käyttö saanut suosiota web-ohjelmien myötä.

Model eli suomeksi malli, vastaa ohjelmiston sisäisestä tietorakenteesta. Se muodostaa ohjelman tiedonkäsittelyyn liittyvät loogiset operaatiot. Malli sisältää ohjelman datan, logiikan ja säännöt.

View tai näkymä käsittää ohjelman tuottaman tulosteen. Tämä tuloste voi koostua mistä vain datasta. Ohjelma voi sisältää useita näkymiä.

Controller tai ohjain ottaa vastaan syötteet ja käsittelee ne komennoiksi. Se toimii rajapintana mallin ja näkymän välillä.

Kuviossa 1 esitetään MVC-malli visuaalisesti. (Anshul 2018)



Kuvio 1. MVC malli kuvattuna

2.1.8 ORM

Object-relational mapping (ORM) on tekniikka, jolla käännetään yhteensopimatonta dataa erityyppisten järjestelmien välillä käyttäen olio-ohjelmointia. Tekniikan tyypillisin toteutus käsittelee loogisen olion sisältämää dataa muotoon, jossa se voidaan tallentaa esimerkiksi tietokantaan. Tietokannat eivät tyypillisesti kykene tallentamaan monimutkaisia datatyyppejä, jonka vuoksi tällainen data on purettava tiekannalle sopiviin osiin.

ORM-kirjastoa käytetään usein käsittelemään datan muunnoksen ohjelman olioiden ja tietokannan välillä. Tavoitteena on siis käsitellä tietokannan dataa käyttäen valittua ohjelmistokieltä Structured Query Language (SQL) kielen sijaan. (Hoyos 2018)

2.2 Työkalut

2.2.1 Yleistä

Projektin toteutuksessa käytetyt työkalut määrittyvät pääsääntöisesti kehitystiimin valitsemien ohjelmistojen ja ratkaisujen mukaan. Lisäksi ratkaisujen valintaan vaikuttaa Sanoma Oyj -konsernin valitsemat ratkaisut ja vaatimukset.

Projektin aikana toteutin työtäni pääasiallisesti etätyönä, jolloin Virtual Private Network (VPN) -ohjelman käyttö oli välttämätöntä. Yrityksessä tapahtuneiden muutosten vuoksi päädyin käyttämään useaa eri VPN-sovellusta toteutuksen aikana. Tämän vuoksi en esittele tiettyä ohjelmaa. Tunnistautuminen erinäisiin yrityksen tarjoamiin palveluihin tapahtui Active Directory (AD) tunnusten kautta.

2.2.2 Active Directory

AD on Microsoftin kehittämä hakemistopalvelu. Se on hajautettu tietokanta, joka ylläpitää tietoa verkkoon liitetyistä laitteista, käyttäjäprofiileista ja resursseista. Se mahdollistaa käyttäjien ja verkkoresurssien hallinnan hierarkkisesta rakenteesta.

AD mahdollistaa käyttäjien oikeuksien hallinnan eri verkkoresursseihin keskitetysti käyttäjän tunnusten perusteella. (Active Directory Domain Services Overview 2017.)

2.2.3 Virtual Private Network

VPN on palvelu, joka ohjaa käyttäjän internet liikenteen salatun kanavan läpi. VPN-verkkoa voidaan käyttää salaamaan käyttäjän arkaluontoinen liikenne tai säilyttämään käyttäjän yksityisyys internettiä selatessa. (What is a VPN? n.d)

Yrityskäytössä VPN-ohjelman käyttötarkoitus on tyypillisesti yhdistää maantieteellisesti erossa olevia toimipisteitä yhdeksi loogiseksi sisäverkoksi, jolloin VPN-käyttäjät voivat käyttää sisäverkon resursseja, ilman että näitä tarvitsee altistaa julkiseen internettiin. VPN on usein välttämätön työkalu etätyön toteutuksessa.

2.2.4 Slack

Slack on kommunikaatiotyökalu, joka on kehitetty erityisesti työyhteisöjen käyttöön. Se on toiminnaltaan hyvin samankaltainen kuin Internet Relay Chat (IRC), joka oli pitkään suosittu yhteydenpitotapa internetissä. Sen toiminta koostu pääasiallisesti kanavista, joihin käyttävät liittyvät. Kanaville liitytään usein toisen käyttäjän lähettämästä kutsusta.

Slack tarjoaa IRC:hen verrattua useita kehittyneitä ominaisuuksia, kuten haettavan viestihistorian ja mahdollisuuden yhdistää ulkoisia ohjelmia kanaville. Nämä ohjelmat voivat raportoida kanavalla ajankohtaisista tapahtumista, kuten palvelimen tilasta tai koodi repositorion päivityksestä.

Slack tarjoaa myös mahdollisuuden ääni - ja videopuheluihin kanavan jäsenten välillä. Videopuhelussa myös ruudun jakaminen on mahdollista. Slack voi täten korvata monessa tilanteessa useamman eri kommunikaatiotyökalun kuten Skype tai Teamviewer. (What is Slack? n.d.)

2.2.5 GIT

Git on moderni laajasti käytetty versionhallinta työkalu, jonka kehitti alun perin Linus Torvalds vuonna 2005. Git toimii hajautetusti, jolloin jokaisen kehittäjän työstämä kopio projektista on myös repositorio, joka voi sisältää täyden historian muutoksista.

Toisin kuin moni versiohallintajärjestelmä Git ei hämmenny tiedostonimien muutoksista, sillä se keskittyy tiedostojen sisältöön.

Git repositoriot ovat suojattuja SHA1-hash-algoritmilla, joka varmistaa, että projektin koko historia on seurattavissa. Git on nykyään käytännön standardi ohjelmistokehityksessä ja on sen vuoksi laajasti tuettu erilaisten työkalujen kuten Integrated development environment (IDE) puolesta. Tämä tekee Gitin käyttöönotosta projektissa helppoa. (What is Git n.d.)

Git on avoimenlähdekoodin projekti, joten se on laajasti saatavilla ja aktiivisesti ylläpidetty.

2.2.6 Jira

Jira on Atlasianin kehittämä projektinhallintatyökalu. Se on kehitetty alkujaan ongelmien seurantaan, mutta on kehittynyt tehokkaaksi hallintatyökaluksi. Se soveltuu erityisesti erilaisten ketterien kehitystyylien kuten scrum ja kanban seurantaan. Kanban tarjoaa taulunäkymän tikettien seurantaan ja mahdollistaa tikettien seurannan helposti koko tiketin elinkaaren läpi.

Jira sisältää myös tehokkaat seuranta työkalut projektin etenemisen seuraamiseen. Tämä mahdollistaa tehokkuuden arvioinnin tiimeille. Jira tarjoaa mahdollisuudet eri tiimien työn seurantaan samassa järjestelmässä oikeuksien avulla. (What is Jira used for? n.d.)

2.2.7 Confluence

Confluence on Atlasianin kehittämä tiedonjakoympäristö. Sitä voidaan käyttää moniin eri käyttötarkoituksiin, mutta se on pääasiallisesti suunniteltu projekti dokumentoinnin säilyttämiseen ja indeksointiin.

Se tarjoaa käyttäjille helpot ja monipuoliset työkalut artikkelien kirjoittamiseen ja muokkaamiseen. Artikkelit voivat sisältää kuvia tai muita multimediatiedostoja. Artikkelien käyttöoikeuksia voidaan hallinta, jolloin voidaan varmistaa, että käyttäjä pääsee käsiksi vain sellaiseen materiaaliin, johon hänellä on oikeudet.

Confluence tarjoaa myös integrointi mahdollisuuksia muiden Atlasianin tuotteiden kuten Jira:n kanssa. (Confluence: What is Confluence? n.d.)

2.2.8 PHPstorm

PHPstorm on jetbrains yrityksen kehittämä IDE, joka tarjoaa monipuolisen kehitysympäristön web-kehittäjille. PHPstrom nimestä huolimatta IDE ei ole rajoittunut PHP-kehitykseen vaan tukee useita kieliä.

Ohjelma tarjoaa kaikki tavalliset IDE toiminnallisuudet kuten koodin refaktoroinnin ja syntaxin värityksen. Se tarjoaa myös automaattisen koodin formatoinnin, joka helpottaa pitämään koodin paremmin luettavana. (PhpStorm n.d.)

Tavanomaisten ominaisuuksien lisäksi ohjelma tarjoaa ominaisuuksia, jotka ovat erityisen hyödyllisiä web-kehityksessä. Näihin kuuluvat muun muassa mahdollisuus lähettää koodi tallennettaessa kehityspalvelimille. Tämä ominaisuus mahdollistaa muutosten nopean testaamisen kehitysympäristössä.

2.2.9 Amazon Web Services

Amazon Web Services (AWS) on verkkokaupastaan tunnetun Amazon-yhtiön tarjoama pilvipalvelualusta. Se tarjoaa suuren määrän erilaisia verkkoteknologiapalveluja ja se on maailman laajin ja käytetyin pilvipalvelu. (The leading cloud platform n.d.)

AWS tarjoaa asiakkailleen tehokkaan Command Line Interface (CLI) -työkalun, joka mahdollistaa skriptien käytön AWS-palvelujen hallinnassa. Hyvän CLI-työkalun olemassaolo on tärkeää ylläpidon kannalta, sillä laajamittaisten palvelujen hallinta web-käyttöliittymien kautta on usein hidasta ja työlästä.

Projektin kannalta tärkeimmät AWS-tuotteet ovat Cloudfront Content Delivery Network (CDN) ja S3 sisällön hostaus palvelu.

3 Sovelluksen nykytila

3.1 Projektin aloitus

Projektin alkutilanteena pidetään PDX-ohjelmiston ja rajapintojen tilaa tammikuussa 2018. Ajankohta valittiin, koska tällöin uudesta käyttöliittymä projektista oli päätetty ja sen käytännön toteutusta aloitettiin konkreettisesti suunnittelemaan. Tässä projektin vaiheessa oli päätetty uuden käyttöliittymän toteutustavoista ja siinä käytettävistä teknologioista.

Käyttöliittymäprojektin aikataulun vakiintuminen johti myös toteutukseen tarvittavien rajapintamuutosten aikataulun ja tarpeiden määrittelemisen. Ennen käyttöliittymäprojektin aloitusta rajapintauudistukselle ei ollut varattu omaa kehitysaikaa, vaan rajapinnan kehitys oli tapahtunut muiden projektien taustalla ajan salliessa.

3.2 PDX

PDX-ohjelmisto on toteutettu pääosin PHP-kielellä ja noudattaa löyhästi ORM-tyylistä toteutusta. PDX-ohjelmisto toimi tähän aikaan vielä tyypillisen LAMP-kokonaisuuden päällä.

Itse PDX-ohjelmiston koodissa ei ollut merkityksellistä erotusta käyttöliittymän ja sovelluslogiikan välillä. Ohjelmistossa datan käsittelyä hoidettiin kaikilla toteutuksen tasoilla ja ohjelma lähetti kutsuja tietokannalle, sekä ulkopuolisille integraatiolle tarvittaessa suoraan käyttöliittymään liitetyillä skripteillä. Käytännön näkökulmasta ohjelmistolla ei ollut erillistä front - tai back endiä. Front endillä tarkoitetaan tässä kontekstissa ohjelman käyttöliittymää ja back endillä ohjelman sovelluslogiikkaa, sekä taustajärjestelmiä. Tämä tilanne koski eritoten PDX-repositorion koodia, joka käsitti varsinaisen PDX-ohjelman. Reposition tilanne ei kumminkaan anna hyvää kokonaiskuvaa PDX-tuotteen koodikannasta, sillä PDX-kokonaisuuteen kuului myös paljon uudempia komponentteja. Useat näistä komponenttiohjelmista tuotettiin eri palvelimilla ja niiden kehitys tapahtui erillisissä projekteissa. Komponenttiohjelmat olivat kumminkin toiminnallisia osia PDX-kokonaisuutta ja niiden toiminta oli osa PDX-tuotetta. Ohjelmiston käyttäjän näkökulmasta PDX oli yksi yhtenäinen kokonaisuus, jota käytettiin yhden käyttöliittymän kautta.

Monet näistä uudemmista projekteista käyttivät PDX-REST-rajapintaa, mutta eivät pääosin koskeneet PDX:n ydintoiminnallisuuteen.

PDX-ohjelmisto tuotti käyttöliittymänäkymän käyttäen erilaisia .tpl- ja .htm-tarkenteisia template-tiedostoja. Erot tiedostojen välillä olivat suuria, sillä käyttöliittymän eri välilehdillä oli käytetty eri toteutustapoja. Esimerkkinä vuokratoimeksiannoista vastaavat sivut oli toteutettu käyttäen Smarty-kirjastoa, joka käytti omaa .tpl-formaattia sivun perusrakenteeseen. Suurin osa PDX-välilehdistä kumminkin tuotettiin eri tekniikalla, joka käytti .htm-tiedostoja pohjanaan.

Molempien toteutustapojen template-tiedostot sisälsivät HTML-, PHP-, JavaScript- ja JQuery -koodia, sekä suuren määrän kirjastokohtaisia tekniikoita. Templateihin sisällytetyt skriptit tekivät suoria tietokanta kutsuja niin datan lähetykseen kuin noutoon.

Eri toteutustapojen käyttö aiheutti myös suuria ongelmia käyttöliittymän yhtenäisen ulkonäön ylläpidossa. Toteutusten erilainen tapa tuottaa HTML-elementtejä johti tilanteeseen, jossa näiden elementtien ulkonäön asetukset jouduttiin usein asettamaan yksilöllisesti. Tämä puolestaan johti tilanteeseen, jossa lähes jokainen uusi visuaalinen elementti hajotti sivun ulkoasun ja vaatii useiden elementtien määritysten muokkausta.

PDX:n epäjohdonmukainen rakenne aiheutti kehityksessä haasteita, jotka johtivat uusien ominaisuuksien hitaaseen kehitykseen, sekä bugeihin. Näiden ongelmien lisäksi satunnaiset tietokantakyselyt kuormittivat palvelimia ja hidastivat ohjelman toimintaa. Kyselyjen suuri määrä koodissa teki myös niiden laajamittaisesta optimoinnista vaikeaa.

3.3 REST

3.3.1 Tausta

PDX-REST-projektin kehitystavoitteena on aloituksesta asti ollut vanhan PDX-ohjelma-pinon korvaus. Alkuperäisiin kehitystavoitteisiin kuului luoda mahdollisuus asiakkaalle kehittää oma käyttöliittymänsä.

PDX-tiimillä oli ollut hyvin rajallisesti mahdollisuuksia rajapintakehitykseen ja tämän vuoksi rajapintaa oli kehitetty pääosin liiketoiminnan tarpeiden mukaan. Rajapintakehityksen painopisteet ovat täten olleet asiakasprojekteissa, jonka vuoksi todellinen kehitys käyttöliittymärajapintaa kohti oli mahdollista vasta yrityskaupan jälkeen.

REST-rajapinnassa oli tarkastellun projektin aloitushetkellä, pääasiassa valmiudet ainoastaan tiedon hakuun eli GET-metodille. Suurin REST-rajapintaa käyttävä ominaisuus oli eräälle suurelle asiakkaalle toteutettu lomakepalvelu, joka käytti omia endpointtejaan ja ei merkittävästi käsitellyt PDX-ohjelman datamalleihin tallennettua tietoa. Endpointilla tarkoitetaan osoitetta, johon rajapintakyselyt lähetään.

REST-rajapinta on toteutettu PHP-kielellä ja sen kehitys oli aloitettu Luracast Restler -sovelluskehyksen päälle. Alusta oli vuosien varrella todettu toimivaksi ja tarjosi kaikki projektin tarvitsemat ominaisuudet, joten alustan vaihtoa ei tarvinnut harkita. REST-

kehitys projekti oli aloitettu käyttäen Composer-nimistä, PHP:lle suunniteltua paketinhallintaratkaisua, joka huolehtii kirjastojen päivityksestä ja yhteensopivuudesta.

REST-toteutuksen voidaan sanoa noudattavan ORM- ja MVC-kehitysmalleja.

3.3.2 Composer

Rajapinnan käyttämät kirjastot oli asennettu käyttäen Composer-paketinhallintatyökalua.

Composer huolehtii kirjastojen vaatimusten täyttämisestä sekä päivityksestä. Se toimii projekti pohjaisesti ja hankkii ainoastaan projektiin määritellyt paketit, sekä niiden riippuvuudet. (Coposer: Introduction n.d.)

3.3.3 Luracast Restler

Alustaksi oli valittu Luracast Restler sen keveyden ja helppokäyttöisyyden vuoksi. Alustan valinta perustui silloisen kehittäjän suositukseen ja havaintoon sen vakaudesta. Vaihtoehtona oli harkittu Zend Framework 2:sta, jota oli käytetty yrityksen muissa projekteissa, mutta sitä pidettiin tarpeettoman monimutkaisena ja raskaana projektin tarkoituksiin.

Restler on ilmainen avoimen lähdekoodin RESTful-rajapinnan toteuttava sovelluskehys, joka on kehitetty olemaan joustava, helppokäyttöinen ja kevyt. (Restler n.d.)

3.4 Integraatiot

3.4.1 Tausta

PDX-ohjelmistojen pääasiallinen tavoite on helpottaa ja nopeuttaa kiinteistövälittäjien työtä. Tähän tarkoitukseen PDX sisältää integraatioita moniin kiinteistövälityksessä tarvittaviin palveluihin kuten web-portaaleihin. Näiden lisäksi PDX sisältää integraatioita esimerkiksi Canonin monitoimilaitteisiin, jotka mahdollistavat asiakirjojen lukemisen PDX:ään suoraan laitteen käyttöliittymästä.

Nämä integraatiot oltiin suurimmaksi osaksi toteutettu rajapintoihin lähetettävien XML-tiedostojen (Extensible Markup Language) muodossa. Työn kattaman jakson alkupuolella uusimmat integraatiot toimivat jo REST-rajapinnan kautta. Näiden lisäksi

oli olemassa vanha PDX API-rajapinta, jonka käyttöä on pyritty jatkuvasti vähentämään.

3.4.2 Kotisivut

PDX tarjoaa asiakkailleen mahdollisuuden käyttää ohjelmaan syötettyä dataa kotisivujensa sisällön tuottamisessa. Tämä tapahtui työn aloituksen aikana PDX-ohjelmiston tuottamien XML-viestien kautta. Kotisivu XML-viestien sisältö koostuu elementeissä, joihin kohteen tiedot asetetaan. Näiden elementtien rakenne ja sisältö kuvaillaan sanomakuvauksessa, joka on jaettu käyttäjille. Sanomakuvauksen ylläpito kuului PDX-tiimin tehtäviin.

3.4.3 Portaalit

Tärkeä osa PDX:n toiminnallisuutta on ohjelman kyky lähettää asiakkaan kohteet kolmansien osapuolien ylläpitämille web-portaalisivustoille. Tärkeimmät näistä ovat Oikotie- ja Etuovi-asunnonvälitysportaalit.

PDX tuottaa asiakkaan tietokannan muutosten perusteella, portaalille räätälöidyn XML-viestin. Tämä XML-tiedosto muodostetaan portaalin toimittaman sanomakuvauksen mukaisesti.

3.5 Taustajärjestelmät

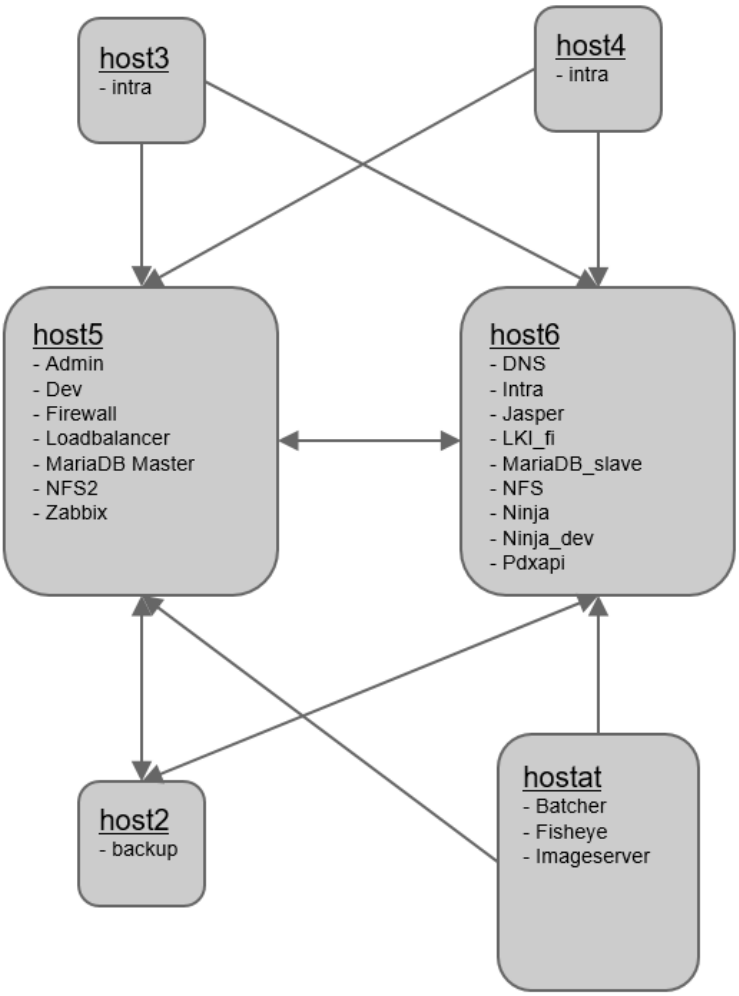
3.5.1 Palvelimet

PDX-ohjelmistoja palvellaan yrityksen omistamilta palvelimilta Nebula nimisen yrityksen konesalissa. PDX-sisäverkko koostuu kuudesta fyysisestä palvelimesta. Kuviossa 2 voidaan nähdä palvelinten väliset fyysiset yhteydet, sekä niillä toimivat virtuaaliset palvelimet. PDX-ohjelma palvellaan Host3- ja Host4- laitteilta.

Käyttäjän näkymä PDX-ohjelmistoon muodostuu kuitenkin useamman eri palvelimen tuottamista komponenteista, jotka yhdistetään PDX käyttöliittymän kautta yhdeksi kokonaisuudeksi. PDX-ohjelmiston eri ikäiset komponentit tarvitsevat erilaiset palve-

linympäristöt, jonka vuoksi ne on tuotettava eri palvelimilla. PDX-sisäverkossa toimivat myös eri komponenttien kehityspalvelimet, sekä jotkin ohjelmiston kehitystä tukevat palvelut. Kaikki ulkoverkon yhteydet kuljetetaan palomuurikoneen kautta.

Yksi rajapintakehityksen päämääristä on mahdollistaa PDX-palvelujen vaiheittainen siirtyminen AWS-ympäristöön.



Kuvio 2. Sisäverkon palvelimet sekä niiden fyysiset yhteydet

3.5.2 Virtualisointi

Virtualisointi on tekniikka, joka mahdollistaa usean virtuaalisen tietokoneen käyttämisen yhdellä fyysisellä laitteella. Virtualisointi parantaa palvelujen skaalattavuutta ja

mahdollistaa ongelmien nopean korjaamisen ja käyttöjärjestelmä, sekä ympäristö päivitysten testaamisen ennen käyttöönottoa.

Virtualisointialustana PDX:n palvelimilla käytetään Xen hypervisoria. PDX-palvelimet on kehitetty palvelemaan pääsääntöisesti yhtä käyttötarkoitusta.

3.5.3 Tietokanta

PDX:n toiminta perustuu pääasiassa datan organisointiin ja esittämiseen asiakkaalle helposti käytettävässä muodossa. Datan säilytykseen PDX käyttää Mysql-sovellukseen perustuvaa MariaDB-tietokantaa. MariaDB on suosittu tietokantaratkaisu, joka tukee SQL-kieltä kyselyjen muodostamiseen.

Kiinteistövälitysala on Suomessa hyvin säädeltyä ja PDX:llä on tämän vuoksi lakisääteisiä velvoitteita säilyttää asiakkaan data- ja muutoslogit useiden vuosien ajan. Ohjelman tietokannasta on riitatilanteessa pystyttävä selvittämään kauppaan liittyvien tietojen tila kauppahetkellä. Tämän vuoksi PDX-ohjelmiston tietokanta on epätavallisen suuri ja kasvaa jatkuvasti. Ohjelma käyttää yli 300: taa asiakastietokantaa, joiden koko vaihtelee huomattavasti. Suurempien asiakkaiden tapauksessa, kanta sisältää useita miljoonia rivejä. Tietokannasta on myös pystyttävä pitämään ajantasaisia varmuuskopioita. Tiimi säilyttää jokaisen asiakkaan tietokannasta On - ja Offsite-varmuuskopioita.

Tietokannan koko asettaa erityisiä vaatimuksia tietokantasovellukselle ja näiden vaatimusten valossa MariaDB on selkeä valinta. MariaDB tarjoaa merkittävän parannuksen suorituskyvyssä verrattua Mysql:ään. Se tarjoaa useita suorituskykyä parantavia ominaisuuksia, kuten rinnakkaisten kyselyjen toteutuksen eri palvelimilla. MariaDB tarjoaa myös paremman skaalautuvuuden suurille tietokannoille. (Aman 2020.)

3.6 Kehitysprosessi

Projektin alussa tiimi käytti kanbaniin perustuvaa ketterää kehitystapaa, joka soveltui hyvin, pienen yrityksen alati muuttuviin prioriteetteihin. Kehitysprosessi perustui tiimijohtajan jakamiin tiketteihin, joita kehittäjät toteuttivat, pääosin haluamassaan järjestyksessä. Prosessin suurimpia heikkouksia oli sen vaatima aika tiimijohtajalta.

Yrityskaupan myötä tiimin kehitys tavoitteet muuttuivat ja vanha kehitysprosessi alkoi jäämään riittämättömäksi uusissa projekteissa.

4 Järjestelmän kehittämiskohteet

4.1 Kehittämiskokonaisuudet

Kehitysprojektin ensisijainen tavoite oli tuottaa PDX-ohjelmistolle uusi käyttöliittymä. Tätä käyttöliittymää lähdettiin kehittämään hyödyntäen PDX REST-rajapintaa. Rajapinnan käyttö loisi selkeän erotuksen PDX:n käyttöliittymän ja tiedonkäsittelyn välille.

Uusi käyttöliittymä haluttiin saada asiakkaiden käyttöön mahdollisen nopeasti. Tämän vuoksi sitä kehitettäisiin vaiheissa, jolloin asiakkaille voitaisiin julkaista pienempiä osakokonaisuuksia. Kehitystapa antaisi tiimille myös mahdollisuuden korjata asiakkaiden löytämiä ongelmia nopeammin.

Yrityskauppa ja sen jälkeinen organisaation järjestely antoivat PDX tiimille hyvän tilaisuuden kehittää uusia prosesseja. Tiimissä oli jo aiemmin huomattu projektinhallinta ja tuotantoprosesseissa puutteita, jotka olivat peräisin LKI ajan lyhyen tähtäimen kehityksestä.

Kehitystä on opinnäytetyössä käsiteltävä mahdollisimman abstraktisti, jotta työstä ei selviäisi yksityiskohtia PDX:n toiminnasta. Näitä voitaisiin esimerkiksi käyttää palvelun häiritsemiseen.

4.2 PDX

PDX ei ollut varsinaisen tuotekehityksen kohde kehitysprojektissa. Rajapinnan laajentaminen uuden käyttöliittymän tarpeisiin johtaisi kuitenkin tilanteeseen, jossa tiimin on ylläpidettävä kahta versiota ohjelmasta. Ylläpitoon käytettävän ajan vähentämiseksi siirtymäaikana, päätettiin PDX muuttaa käyttämään rajapintaa niin paljon kuin on mahdollista.

4.2.1 Suunnitelma

Tavoitteena vanhan PDX:n kehityksessä oli muokata ohjelmiston käyttämiä datamalleja siten, että ne käyttäisivät REST-rajapintaa. Tämä vähentäisi tulevaisuudessa ylläpitotarvetta vanhan käyttöliittymän puolella, jolloin uusien ominaisuuksien kehitys siirtymäaikana olisi nopeampaa.

Kehitys aloitettiin selvittämällä millaisia muutoksia datamallit tarvitsisivat, jotta ne voitaisiin muokata käyttämään rajapintaa. Selvityksen tarkoituksena oli myös arvioida millainen työmäärä tähän päivitykseen liittyisi. Arvion perustella voitaisiin päättää olisiko päivitys ajankäytön kannalta järkevää.

PDX oli selvityksen aloitus vaiheessa hyvin vanha ohjelmisto, jossa uusia ominaisuuksia ja korjauksia oli toteutettu epäjohdonmukaisesti. Ohjelmiston alkuperäinen toimintamalli oli tallentaa lomakkeen tiedot siihen liitetyn datamallin kautta. Vuosien varrella ohjelmistoon oli kumminkin lisätty monia toimintoja, jotka käyttivät tietokantaa suoraan erilaisten skriptien avulla.

PDX:n datamallien tiedonhaku rutiini osoittautui tutkinnassa niin hankalaksi korvata rajapinnalla, että tästä osuudesta projektia päätettiin luopua. Kohteiden luonti - ja päivitysoperaatiot sen sijaan katsottiin olevan päivitystyöhön käytettävän ajan arvoisia.

Ratkaisua lähdettiin kehittämään mahdollisimman modulaarisena, jotta voitaisiin säästää aikaa sen implementoinnissa datamalleihin.

4.2.2 Toteutus

Ratkaisuksi valittiin toteutus, jossa lomakkeessa muuttuneiden kenttien arvot kerätään ja lähetetään rajapinnalle tätä varten kehitetyn luokan avulla.

Kehitys aloitettiin rajapinta luokasta, jonka tarkoitus olisi hoitaa kaikille rajapintakutsulle yhteiset toiminnot. Yhteisiä toimintoja olivat, merkistön muutos ISO-standardista rajapinnan odottamaan UTF-8 standardiin, yksinkertainen virheenkäsittely toiminnallisuus, sekä rajapintakutsujen headerien rakentaminen muun muassa autentikointi tarkoituksiin. Tätä luokkaa käytettäisiin hoitamaan kaikki PDX:n rajapintakutsut.

Toteutusvaiheessa suurin haaste oli PDX-lomakekenttien datan formatointi, jota toteutettiin ohjelmassa monella tasolla. PDX-ohjelmisto muokkasi käyttäjälle näkyvän tiedon muotoa tehdäkseen sen helpommin ymmärrettäväksi.

Esimerkkinä tästä käyttöliittymässä esitettäisiin valuutta arvo muodossa ”100.000,75”, joka on helppolukuinen. Tämä arvo on kumminkin tietokannassa tallennettuna desimaaliarvona 100000,75. Ennen lähetystä rajapinnalle arvo on muokattava takaisin tietokannan mukaiseen muotoonsa.

Toteutuksen haaste muodostuu siitä, että lomakekenttien tiedoista ei voisi vain summittaisesti poistaa joitakin merkkejä. Toimintoa ei voisi toteuttaa myöskään yksilöllisesti jokaiselle kentälle, sillä PDX-ohjelmisto sisältää useita satoja lomakekenttiä. Luontaisin tapa olisi selvittää PDX:n tekemät muokkaukset ja peruuttaa ne ennen tallennusta. Tämä ei kuitenkaan ole käytännöllistä, sillä PDX tekee näitä muutoksia toteutuksen jokaisessa kerroksessa ja kaikkien näiden funktioiden etsimisen koodista veisi liikaa aikaa. Muokkauksen poisto oli täten toteutettava siten, että se ymmärtäisi lomake kentän tyypin ja vaikuttaisi ainoastaan juuri sen tyyppisten kenttien arvoihin.

Tarpeettoman datan lähettämisen välttämiseksi oli vielä määritettävä mitä kenttiä käyttäjä oli muokannut, jotta voitaisiin rajapintakutsussa lähettää ainoastaan nämä muutokset.

4.3 REST

REST-kehityksen pääpainopiste projektin aikana oli saattaa rajapinta sellaiseen valmiuteen, että se kykenisi palvelemaan kaikkia uuden käyttöliittymän tarpeita.

Rajapintakehityksen olisi siis toteutettava datamallit ja reitit kaikille PDX:n sivuille, jotka korvataan uudella käyttöliittymällä. Käyttöliittymä projektin ensimmäinen julkaisu sisältäisi vain muutaman yleisimmin käytetyn sivun PDX:stä. Näiden sivujen toiminnan takana olisi kumminkin useampi PDX:n datamalli ja tämän vuoksi kehitystä alettiin edistämään lähtökohdasta, jossa kaikki PDX:n sovelluslogiikka siirrettäisiin rajapintaan.

4.3.1 Suunnitelma

Rajapinnan laajentaminen sisältämään PDX:n sovelluslogiikan tarkoitti, että paljon vanhan PDX:n koodia kopioitaisiin rajapintaan. Vanhan PDX:n koodia ei kumminkaan kannattaisi siirtää sellaisenaan. Tämä koodi oli monesti toteutukseltaan vanhanaikaista ja joissain tapauksissa toteutettu kovassa kiireessä. Toisin sanoen vanha koodi sisälsi paljon niin kutsuttuja ”häkkejä”, joista olisi päästävä eroon.

Vanhassa PDX:ssä koodi sisälsi paljon vertailuja, jotka käyttivät niin sanottuja ”Magic number”-numeroita eli nk. taikanumeroita. Nämä ovat numeerisia arvoja, joiden merkitystä ei pysty päättelemään ympäröivästä koodista. Uudessa toteutuksessa kaikki tällaiset numerot korvattaisiin konstantteilla, joiden nimestä voisi päätellä niiden merkityksen. Tämä parantaisi koodin luettavuutta merkittävästi vanhasta PDX:stä.

Suunnittelussa kiinnitettiin huomioita myös projektin tiedostojen nimeämiseen, jotta tulevaisuudessa toiminnallisuuksien löytäminen olisi mahdollisimman helppoa. Datamallit nimettäisiin niiden kattaman tietokanta taulun mukaan ja datamallia vastaavat reitit puolestaan toteutettaisiin tiedostoon, jonka nimi olisi datamallin nimi monikossa. Esimerkiksi jos datamalli nimettäisiin ”toimeksianto”, tähän datamalliin liittyvät reitit olisi määritetty tiedostoon nimeltä ”toimeksiannot”.

4.3.2 Toteutus

Kehityksessä hyödynnettiin olio-ohjelmoinnille tyypillistä tapaa rakentaa datamalliluokat yhteisen perusluokan päälle, joka hoitaisi malleille yhteiset toiminnot. Datamalliluokka korvasi tarpeen mukaan perittävän luokan perustoimintoja. Nämä korvaaja funktiot kutsuisivat luokkaan määriteltyjä yksityisiä funktioita, jotka manipuloisivat dataa ennen kuin funktio palauttaisi manipuloidun datan isäntä luokan funktiolle tallennusta varten.

Rajapinnan kehitys olisi aloitettava GET-tyyppisten reittien toteutuksesta, jotta uuden käyttöliittymän kehittäjät voisivat käyttää PDX:ään tallennettua dataa käyttöliittymäsivujen toiminnallisuuden testaamisessa. Näiden reittien tahdottiin palauttavan tietokannan data mahdollisimman suoraan, jotta tulevissa toteutuksissa ei tarvitsisi

harkita paljonko palautettua dataa on manipuloitu. Tapauksissa, joissa käyttöliittymään oli saatava käsiteltyä dataa, kehitettiin sille oma näkymä reittinsä. Tällaisten näkymä reittien käyttö rajoitettaisiin vain sisäisille kutsuille. Sisäinen kutsu tarkoittaa tässä kontekstissa sellaista kutsua, joka on tehty käyttäen sisäiseksi määriteltyä API-avainta.

PUT- ja POST-metodien toteutus oli rajapinnan kannalta paljon monimutkaisempi prosessi. Näiden reittien olisi varmistettava, että käyttöliittymän lähettämä data formatoitaisiin juuri oikeaan muotoon tallennusta varten. Tämän lisäksi reittien olisi varmistettava, että kaikki linkitetty data pysyisi ajan tasalla. Koska PDX oli koko kehitys prosessin ajan kaupallisessa käytössä, olisi varmistettava, että käyttäjille ei aiheutuisi ongelmia siirryttäessä uuteen käyttöliittymään. Kaikkien PDX:ään tallennettujen tietojen tulisi siirtymän jälkeen olla käytettävissä ja muuttumattomia.

4.3.3 Cache

Rajapinnan toteutuksessa on hyödynnetty laajasti cacheja, jotka auttavat parantamaan rajapinnan suorituskykyä ja vähentämään tietokannan kuormaa. REST-rajapinta päivittää cache-objektit osana päivityskutsuja, jolloin voidaan varmistaa, että cacheissa on aina saatavilla viimeisimmät tietueet.

Cachejen toteutukseen käytettiin Memcached-ohjelmaa, joka on alun perin suunniteltu dynaamisten web-palvelujen nopeuttamiseen laskemalla tietokannan kuormaa. Memcached antaa palvelimille mahdollisuuden jakaa yhteinen virtuaalinen muisti, joka tarkoittaa, että resurssi on aina tallennettu ja palveltu samasta sijainnista palvelinklusterilla. Tämä varmistaa, että cache on aina yhdenmukainen ja vähentää työn määrää. (About Memcached n.d.)

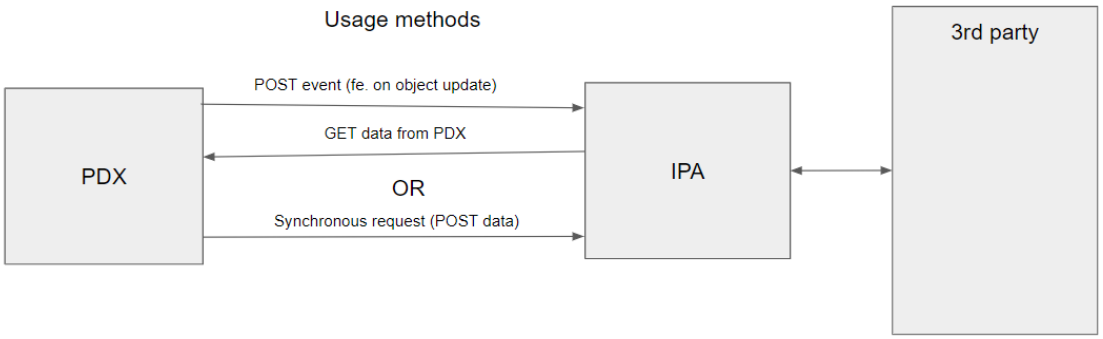
4.4 Integraatiot

Käyttöliittymäprojektin ohessa kehitettiin REST-rajapintaan laajempi tuki integraatioille. Näistä merkittävin kehitysprojekti oli endpointin rakentaminen Oikotien sisäiselle integraatioalustalle. Integraatioalustan kehityksestä vastasi erillinen integraatiotiimi, joka kehittää integraatioita yrityksen eri tuotteiden välillä.

Ennen integraation kehitystä PDX käytti XML-pohjaista aineistonsiirtoa Oikotie-portaalin kanssa. Tämä aineistonsiirto metodi on kumminkin suhteellisen hidas reagoimaan välittäjän tekemiin muutoksiin kohteessa.

Uuden integraation ajatus oli lähettää integraatioalustalle tallennuksen yhteydessä REST-rajapinnalta POST-tyyppinen viesti, joka informoisi integraatioalustaa tapahtuneesta muutoksesta. Tämä toteutus muuttaa portaalin ja PDX:n välisen tiedonjaon lähes tosiaikaiseksi.

Uuden integraatio rajapinnan toiminta on kuvattuna kuviossa 3.



Kuvio 3. Oikotie integraatioalusta

4.5 Uusi käyttöliittymä

PDX:n uusi käyttöliittymä toteutettiin käyttäen React-kirjastoa. PDX-tiimissä ei projektin aloituksen aikaan ollut kokemusta React-kehittämisestä. Projektin onnistumisen varmistamiseksi, kehitystä vetämään palkattiin React - ja käyttöliittymäkehitykseen erikoistunut konsultti. Tämän lisäksi PDX-tiimiin liittyi osa-aikaisesti Oikotie web-portaalin käyttöliittymäsuunnittelija.

En osallistunut suoraan uuden käyttöliittymän toteutukseen. Roolini tässä projektissa keskittyi taustajärjestelmien kehittämiseen ja ylläpitoon. Vastuualueeni projektissa oli AWS-palvelujen pystytys. Uusi käyttöliittymä palveltaisiin käyttäjille S3 bucketista Cloudfrontin kautta.

4.5.1 S3 ja Cloudfront

AWS-palvelun käyttöönotto oli tärkeä osa rajapinta - ja käyttöliittymäprojektin tavoitteita. Aiemmin palvelun käyttöönoton oli estänyt PDX-ohjelmiston tiivis integraatio tietokannan kanssa. Rajapinnan käyttö eristämään käyttöliittymä tietokannasta helpottaa palvelujen toteuttamista pilvessä.

Ensimmäinen toimenpide näiden palvelujen käyttöönotossa oli S3-buckettien luonti siirrettäville palveluille, sekä niiden kehityspalvelimille. S3-bucketin luonti on varsin yksinkertainen prosessi, jonka haasteet koostuvat pääosin sopivien tietoturva-asetusten löytämisestä. PDX-palvelut tarjotaan tietoturvasyistä käyttäen HTTPS-protokollaa, jonka vuoksi palveluille on hankittava sertifikaatti, jota käytetään varmistamaan palvelua tarjoavan palvelimen identiteetti. Nämä sertifikaatit hankittiin AWS:stä tähän tarkoitetun palvelun kautta.

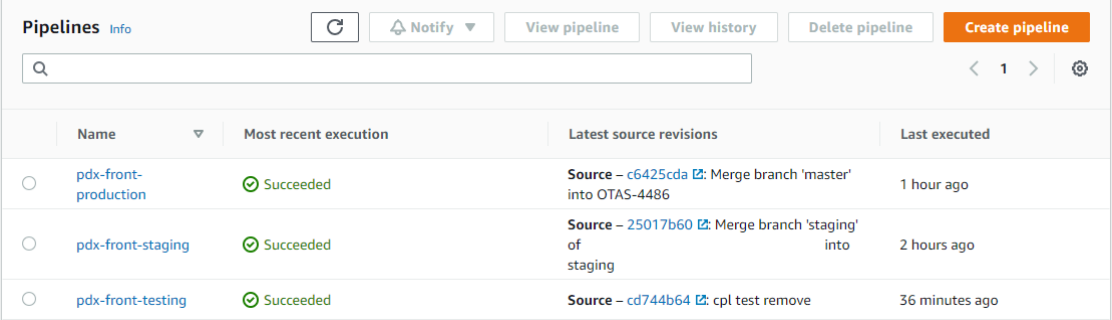
Verkkopalvelujen tarjonta on mahdollista suoraan S3:sta, mutta palvelua käytetään usein yhdessä Cloudfront-palvelun kanssa. Cloudfront-palvelun käyttöönotto on varsin suoraviivainen prosessi, jossa luotu Cloudfront instanssi osoitetaan S3 buckettiin, joka sisältää tarjottavan palvelun sisällön. Cloudfront palvelun käyttäminen vaatii joidenkin asetuksien tekemistä, jotka määrittävät erinäisiä palvelun tarjoamiseen liittyviä toimintoja kuten mihin HTTP kutsut ohjataan tilanteessa, jossa sisältöä ei löydy. Palvelujen siirron viimeisenä toimenpiteenä oli, siirrettävien PDX-palvelujen kanssa käytettyjen DNS-osoitteiden (Domain Name System) muuttaminen osoittamaan Cloudfront-palvelun tarjoamiin osoitteisiin.

4.5.2 Code Pipeline

Uuden käyttöliittymän julkaisua varten luotiin AWS:ään kolme julkaisu pipelineä. Pipelinet luotiin Testing-, Staging- ja Production-asennuksille. Kaikki nämä pipelinet ovat toiminnallisesti samanlaisia ja eroavat toisistaan lähde repositorioiden, sekä kohde-bukettien suhteen.

Pipelinet koostuvat kolmesta vaiheesta, jotka ovat source, build ja deploy. Source tai lähde vaiheessa havaitaan repositorioon tehty muutos. Tämä vaihe laukaisee Build vaiheen, jossa repositoriosta ladattu lähdekoodi käännetään julkaisua varten. Build

vaiheessa käydään läpi myös koodia varten kehitetyt laadunvarmennus testit. Viimeisessä deploy vaiheessa, käännetty koodi lähetetään pipelinelle määritettyyn S3 buckettiin. Kuviossa 4 voidaan nähdä Amazonin hallintanäkymä määritetyille codepipelineille.



Kuvio 4. AWS codepipeline success

4.6 Kehitysprosessi

4.6.1 Kehitystavat

Uudistukset eivät ole rajoittuneet ainoastaan ohjelmiston puolelle vaan uusien projektion myötä on PDX-tiimi kehittänyt myös uudet kehitysprosessit.

Uusi prosessi pohjautuu edelleen vahvasti kanban-kehitysmalliin, mutta siihen on otettu mukaan myös scrum-mallin kaltainen viikoittainen kehityspalaveri.

Prosessissa koko kehitystiimi osallistuu ominaisuuksien kehitykseen tarvittavan työn arviontiin. Tämän jälkeen kehitys tiketti valitaan kehitettäväksi ja lisätään kehitysjonoon prioriteetin mukaan. Yksi uuden kehitysprosessin tavoitteista on kehittää tiiminjäsenten ymmärrystä toistensa erikoisalueista. Prosessijonossa olevan kehitystiketin voi valita kehitettäväksi kuka vain tiimin jäsen, jolloin lokeroitumista ei synny helposti.

Myös työn toteutus on jaettu entistä useampaan vaiheeseen, jossa tiimi arvioi tuotetun koodin laadun ennen sen omaksumista testi asennukseen. Hyväksytty koodi otetaan käyttöön testi asennuksessa, jossa tiketin alkuperäinen luoja voi käydä arvioimassa vastaako toteutus toivomuksia ennen koodin julkaisua.

4.6.2 Tuotantoputki

Uuden käyttöliittymän, sekä REST:in kehitykseen otettiin käyttöön uudet tuotantoputket. Aiemmin PDX kehityksessä kehittäjä loisi uuden kehityshaaran ominaisuudelle ja testaisi koodin toimivuuden, jonka jälkeen yksikönjohtaja ajoittain mergeisi uudet kehityshaarat tuotantohaaraan ja julkaisi ne.

Uusi prosessi antaisi kehittäjille mahdollisuuden luoda pull reqestejä kehityshaaroilleen ja näiden kautta muutokset voitaisiin mergetä tuotantohaaraan ja julkaista. Merge on prosessi, jossa kehityshaarassa olevat muutokset yhdistetään toiseen haaraan. Yksi prosessin tavoitteista oli vähentää yksikönjohtajan työtaakkaa julkaisuprosessissa.

Prosessin toteutukseen käytettiin Github-palvelua, joka tarjoaa koodi repositorion hallintaan tarvittavia työkaluja.

5 Tulokset

5.1 Julkaisu

Käyttöliittymäprojektin ensimmäinen vaihe julkaistiin asiakkaille marraskuussa 2019. Vaihe uudisti PDX:n toimeksiantonäkymän, sekä toimeksiantohaun. Nämä uudistukset kattavat vielä vain pienen osan käyttöliittymästä, mutta kehityksen yhteydessä tehdyt valmiudet REST-rajapintaan ja vanhaan käyttöliittymään mahdollistavat nopeamman julkaisutahdin tulevaisuudessa.

Projektin toinen vaihe, joka korvaa käyttöliittymän työpöytänäkymän, on jo käytössä testauspalvelimilla. Se julkaistaan toukokuussa 2020.

Uudistusten tavoitteena oli modernisoida ympäristöä ja luoda mahdollisuus tiedonkäsittelyyn pienemmissä kokonaisuuksissa kuin PDX-ohjelman alkuperäinen toteutus tarjosi.

Kehitystiimi on pitänyt projektin toteutusta onnistuneena.

5.2 PDX

Käyttöliittymäprojektin aikana PDX-ohjelman asema kehityksessä on muuttunut pääasiallisesta kehityskohteesta, legacy repositorioksi. Ohjelman kehitystavoitteet ovat siirtyneet uusista ominaisuuksista ylläpitotehtäviin, joiden tavoitteena on ylläpitää niitä ohjelmiston toimintoja, joita ei ole vielä korvattu uudessa käyttöliittymässä. Projektin alkuperäiseen suunnitelmaan kuulunut PDX:n toiminnan siirtäminen käyttämään rajapintaan onnistui vain osittain ohjelmanrakenteen vuoksi. Projektin tärkeimmät tavoitteet PDX-koodin ylläpidon vähentämisestä kummin saavutettiin.

PDX on niin laaja ohjelmisto, että sen korvaaminen tulee viemään useita vuosia. Joitakin sen osia todennäköisesti käytetään vielä vuosikymmenenkin päästä. Tästä huolimatta projektin tavoitteet on laajamittaisesti saavutettu. Ohjelmiston käytetyimmät toiminnot, sekä viimeiset käyttöliittymä sivut korvataan todennäköisesti muutaman vuoden sisällä.

5.3 REST

REST-rajapinta on projektin aikana kehittynyt kattamaan lähes koko PDX-ohjelmiston sovelluslogiikan. Rajapinnan laajennus on mahdollistanut myös uusien integraatio projektien kehittämisen, sekä vanhojen integraatioiden vaiheittaisen siirtämisen käyttämään rajapintaa.

Rajapinta toteutus on ollut tiimin havaintojen mukaan varsin menestykäs ja tulee olemaan tulevien PDX-kehitysprojektien selkäranka.

5.4 Integraatiot

Projektin aikana saatiin siirrettyä yhä suurempi osa PDX-integraatioista käyttämään REST-rajapintaa. Näistä suurin muutos oli Oikotie web-portaalin siirto käyttämään tosiaikaista päivitys strategiaa. Tämä muutos vähensi huomattavasti XML-tiedostojen luonnin tarvetta ja täten auttoi myös nopeuttamaan toiminnon käyttöä niille palveluille, jotka edelleen tarvitsivat sitä.

Oikotie integraatio projektin pääasiallinen hyöty oli kumminkin lähes tosiaikaiset päivitykset PDX:stä Oikotien web-portaaliin.

5.5 Käyttöliittymä

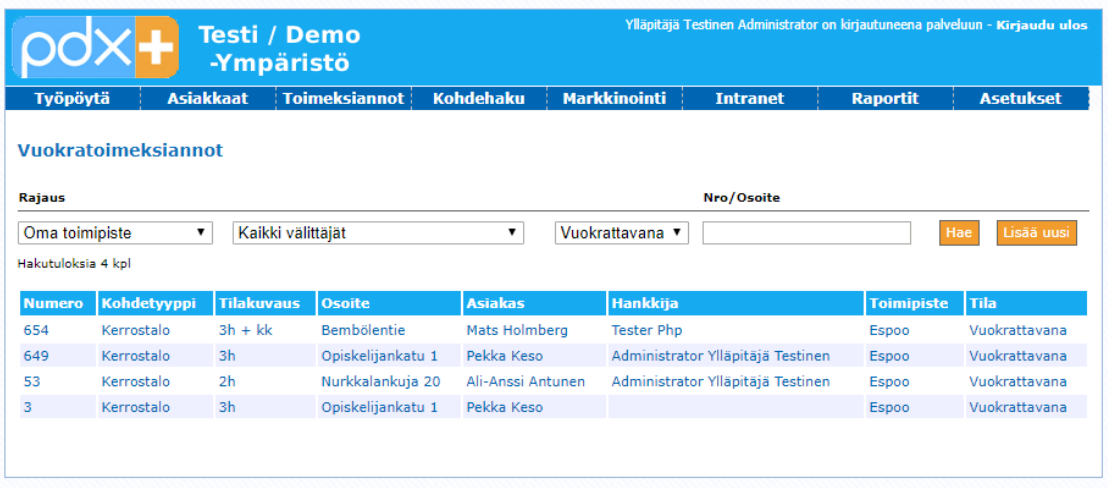
Erot käyttöliittymien välillä ovat mittavat niin käyttökokemuksen kuin toteutuksen kannalta. Vanha PDX-käyttöliittymä on verrattain hidas ja ikääntyneen näköinen. Vaikka vanhasta käyttöliittymästä voi havaita, että sen organisoinnissa on ajateltu käyttäjän kulkua ohjelman toimintojen läpi, on selvää että se on ikääntynyt huonosti.

5.5.1 Vanha käyttöliittymä

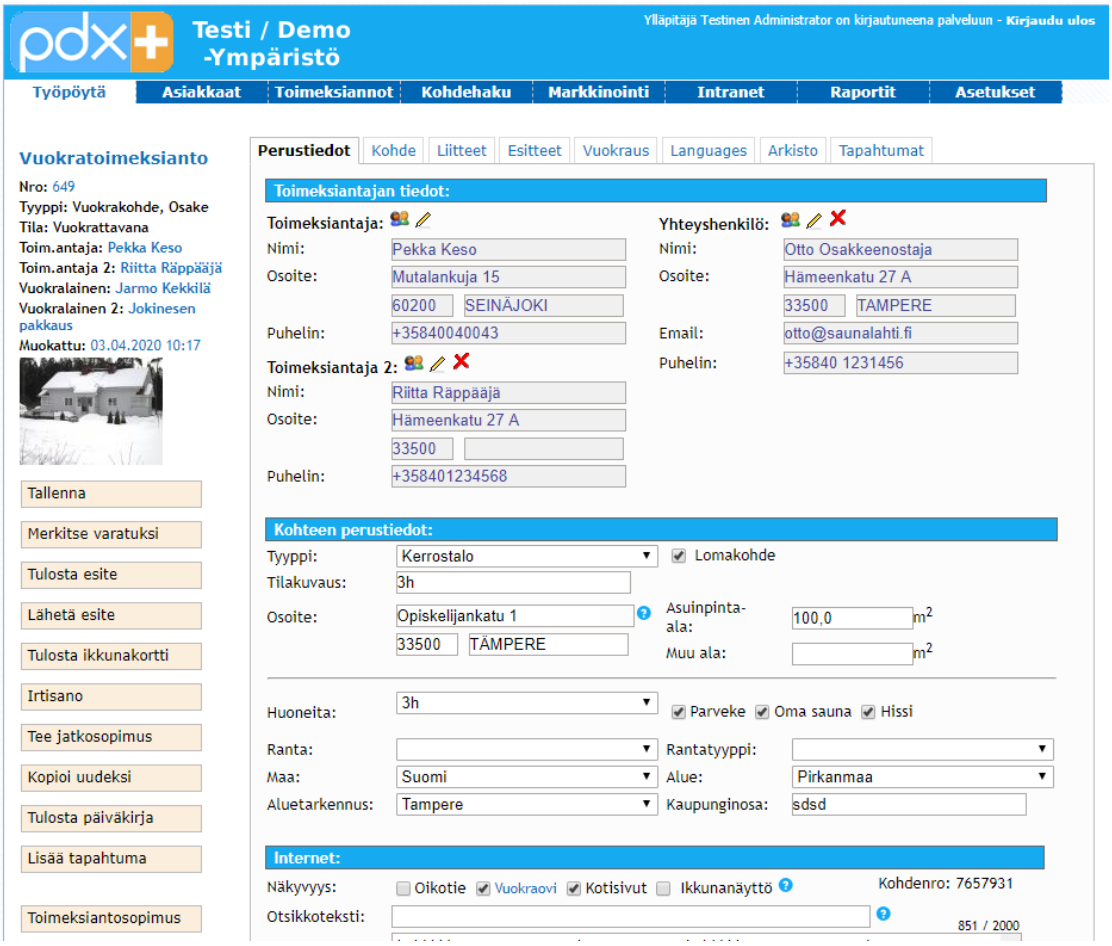
Vanhan käyttöliittymän datamallit uudistettiin käyttämään REST-rajapintaa, suorien tietokanta yhteyksien sijaan. Näiden muutosten myötä toteutus on lähempänä tilannetta, jossa ohjelmalla olisi erillinen front - ja back end. Nämä muutokset eivät ole kuitenkaan tuoneet käyttöliittymään vasteaikoihin huomattavia parannuksia. Vanha PDX-ohjelmisto ei muutosten jälkeenkään noudata MVC-kehitysmallin mukaista toteutusta mittavissa määrin.

Muutosten ensisijainen tavoite oli poistaa tarvetta vanhojen datamallien ylläpitoon ja tässä tavoitteessa on onnistuttu. Datamallien ylläpidon taakka on käytännössä siirretty rajapintakehityksen puolelle, josta se palvelee niin uutta kuin vanhaa käyttöliittymää. Tämä on puolestaan onnistuneesti vähentänyt vanhan käyttöliittymän ylläpidon aiheuttamaa kuormaa siirtymisvaiheen aikana.

Vanhan PDX:n kautta palveltavien sivujen ulkonäkö ei projektin aikana muuttunut, mutta niiden käyttämä taustajärjestelmä vaihtui osittain. Suurin osa vanhan PDX:än useimmin käytetyistä sivuista on muutettu käyttämään REST-rajapintaa tallentamisen osalta. Kuvioissa 5 on kuvattuna vanhan käyttöliittymän hakunäkymä ja kuviossa 6 toimeksiantonäkymä.



Kuvio 5. Vanhan käyttöliittymän hakunäkymä



Kuvio 6. Vanhan käyttöliittymän toimeksiantonäkymä

5.5.2 Uusi käyttöliittymä

Uuden käyttöliittymän toteutus selkeisiin back - ja front end -komponentteihin on saavuttanut kehitykselle asetetut tavoitteet erinomaisesti. Toiminnallisuuden selkeän

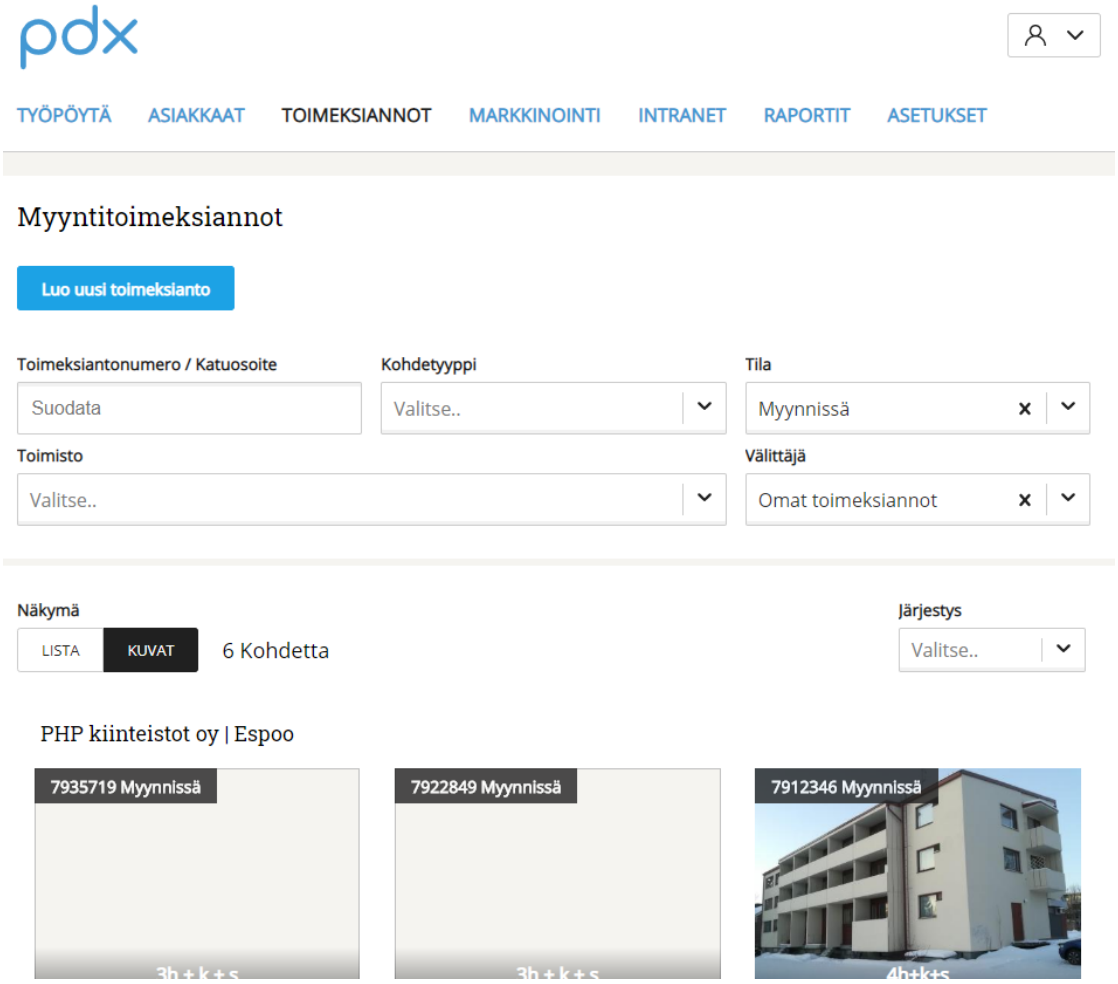
erotuksen vuoksi käyttöliittymä voidaan palvella AWS-pilvipalvelun kautta, joka tarjoaa paremmat mahdollisuudet skaalautumiselle. Palvelun nopeutta ja tietokannalle aiheuttamaa kuormaa auttavat myös REST-rajapinnalle kehitetyt cache-metodit.

Uuden käyttöliittymän kehityksessä käytetään automaattisia unit-testejä, jotka toteutetaan osana uutta codepipelinea. Nämä testit toteutetaan automaattisesti koodin buildauksen yhteydessä ennen ohjelman lähetystä S3-buckettiin.

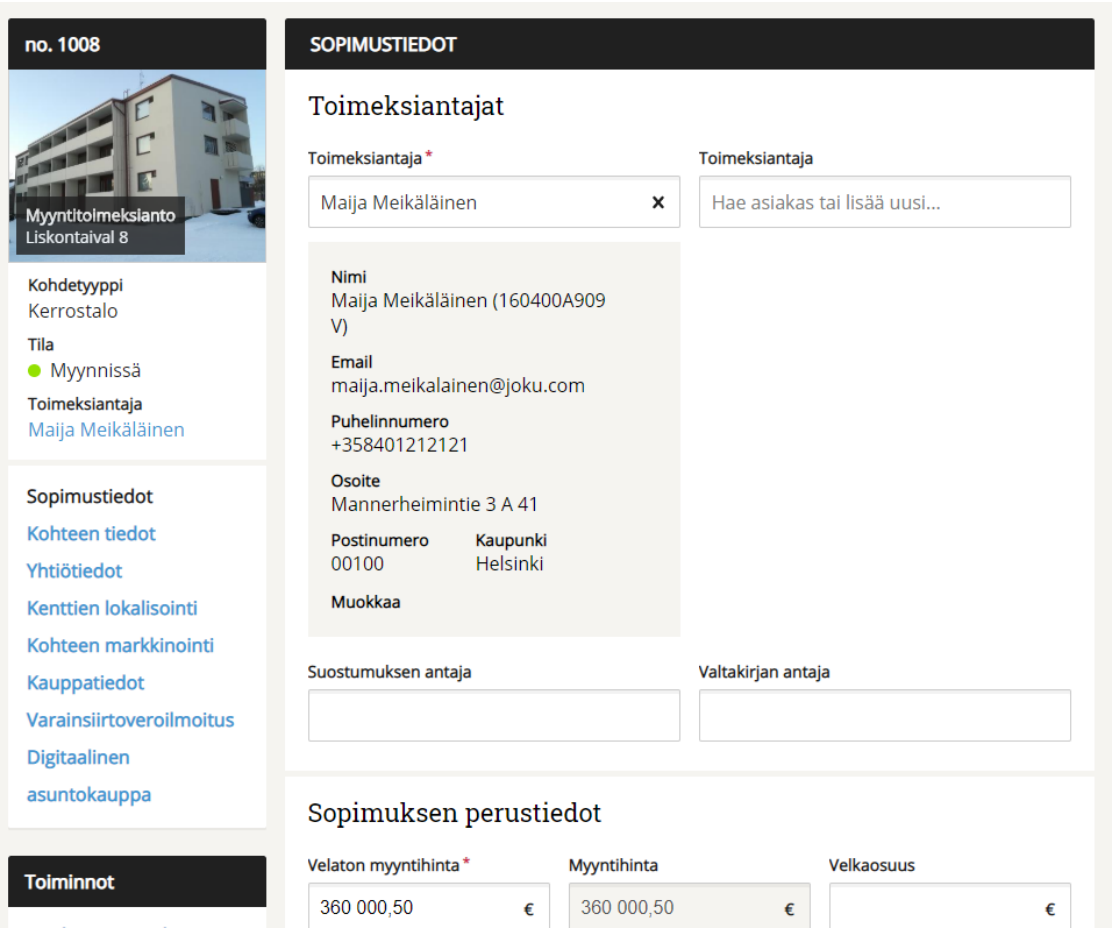
Julkaisu marraskuussa 2019 onnistui hyvin. Julkaisu tehtiin kohtuullisen lähellä joululomien alkua, joka tarkoittaisi, että suurin osa tiimistä pitäisi lomiaan pian julkaisun jälkeen. Tämän vuoksi mahdollisten ongelmien korjaus joutuisi odottamaan. Ajankohta olikin valittu osaksi juuri lomien vuoksi, jotta käyttäjäkunta olisi normaalia pienempi julkaisu ajankohtana. Tämä antaisi tiimille mahdollisuuden viilata pieniä ongelmia, jotka eivät paljastuneet pilotoinnin aikana.

Strategiaan liittyi myös pieni riski, sillä jos suuria ongelmia ilmenisi lomien aikana niin niitä ei välttämättä pystyttäisi korjaamaan riittävän nopeasti. Palvelu voitaisiin jopa joutua sulkemaan lyhyeksi aikaa. Tällaisia ongelmia ei kumminkaan onneksi ilmenny ja julkaisu sujui varsin rauhallisesti.

Kuviossa 7 on kuvattuna uuden käyttöliittymän hakunäkymä ja kuviossa 8 toimeksiantonäkymä.



Kuvio 7. Uuden käyttöliittymän hakunäkymä



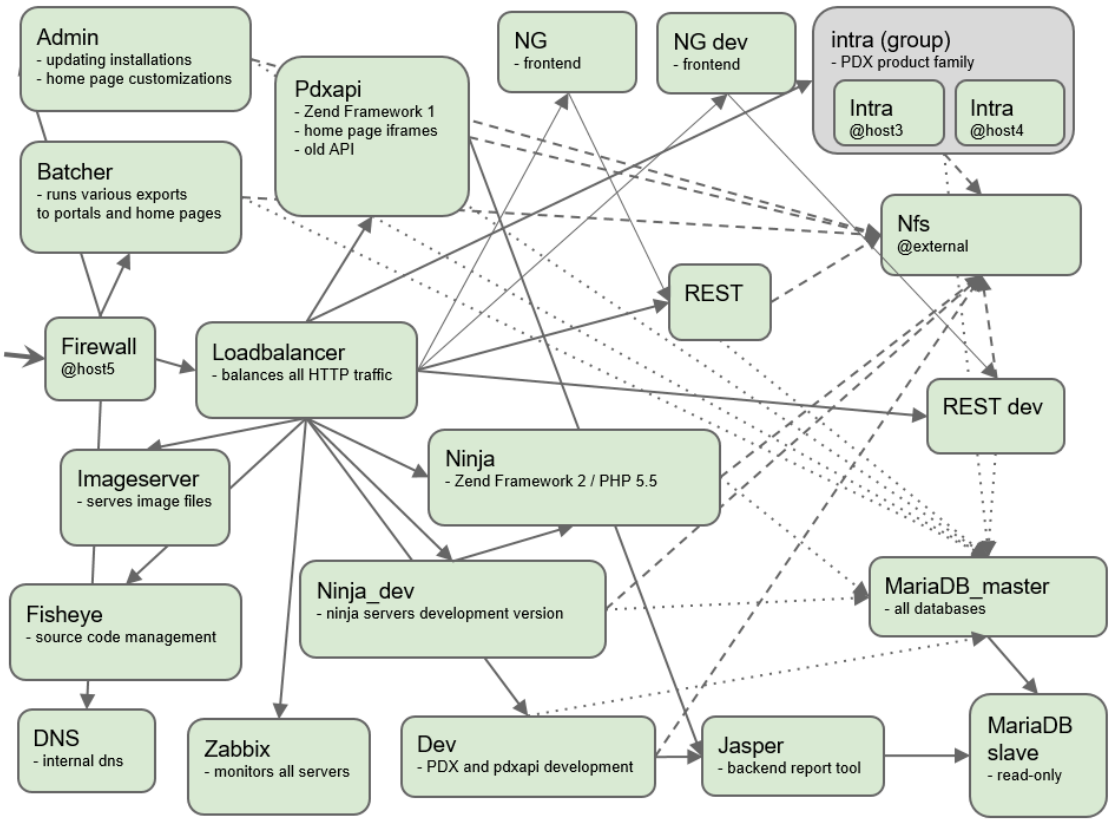
Kuvio 8. Uuden käyttöliittymän toimeksiantonäkymä

5.6 Taustajärjestelmät

Projektin aikana myös PDX:n taustajärjestelmät ovat kehittyneet. Palvelujen rakenne on kumminkin muuttunut projektin aikana vähän. Palvelun rakenteen muutos on edelleen kehityksessä oleva osa projektia, jonka tavoitteena on siirtää suurin osa palveluista Amazonin AWS-pilvipalveluun.

Uuden käyttöliittymän sivut palvellaan Cloudfront-palvelujen kautta. Koska PDX:n uusi käyttöliittymä ei kata kaikkia PDX:n toimintoja, palvellaan nämä toiminnot edelleen yrityksen omilta laitteilta. Uuden käyttöliittymän lisäksi PDX-Extranet palvelu tuotetaan Amazonin palvelimilla.

Kuviossa 9 on nähtävissä virtuaalisten palvelujen suhteet toisiinsa. Palvelujen määrän ja yhteyksien vuoksi niiden asettaminen yhteen loogiseen kuvaan oli jokseenkin haastavaa.



Kuvio 9. PDX palvelujen loogiset suhteet

5.7 Kehitysprosessi

Uusi kehitysprosessi on toiminut tiimille hyvin. Se on saavuttanut alkuperäisen tavoitteensa laadun parantamisessa ja tiiminjohtajan työtaakan vähentämisessä.

Testausvastuun siirtäminen tiketin alkuperäiselle raportoijalle on auttanut varmistamaan, että kehitetyt ratkaisut vastaavat paremmin alkuperäistä tavoitetta. Uudet koodin julkaisuputket ovat puolestaan nopeuttaneet koodin eteenpäinvientiä.

5.8 PDX:n uusi toimintamalli

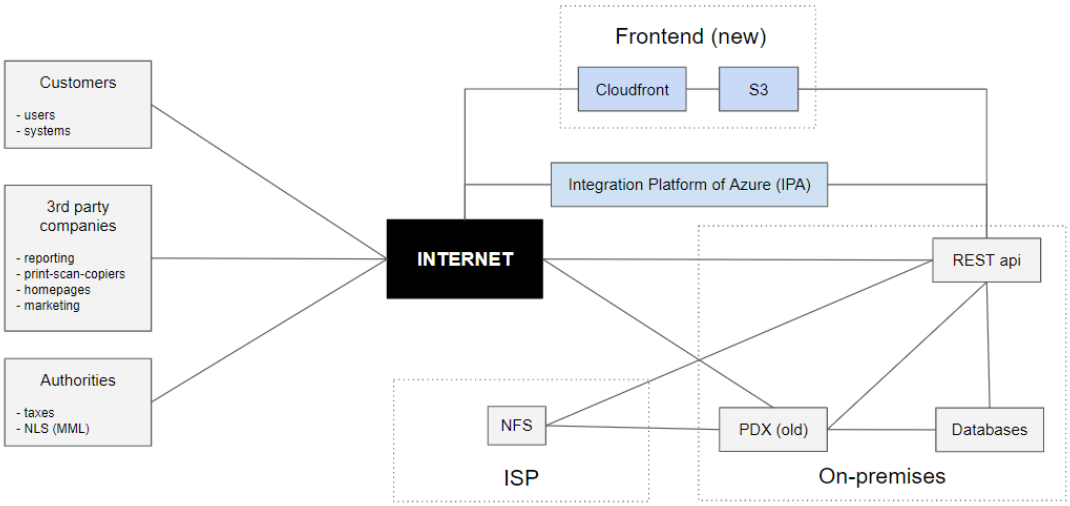
PDX:n uusi toimintamalli pyrkii käyttämään lähes kaikkiin tietokanta yhteyksiin REST-rajapintaa. Tämä malli parantaa tietoturvaa siirtämällä kaikki käyttäjän autentikoinnit saman järjestelmän alaisuuteen. Rinnakkaisten kirjautumismallien poistuessa ohjelmiston ylläpito taakka vähenee ja mahdollisten heikkouksien havaitseminen, sekä korjaus helpottuu.

REST-rajapinnan käyttö tietokantayhteyksissä helpottaa tietokantahakujen optimointia ja auttaa vähentämään hakujen määrää välimuistien (cache) avulla. Rajapinta tehostaa täten ohjelmiston toimintaa jakamalla kuormaa ohjelmiston eri komponenttien välillä.

Uudet rajapinnan kautta toimivat integraatiot mahdollistavat nopeammat päivitykset asiakkaiden järjestelmiin ja vähentävät tarpeettoman datan lähetystä. XML-tiedostojen luomisen vähentyessä myös tietokannan kuorma laskee.

Rajapintamalliin siirtymien on mahdollistanut joidenkin palvelujen siirtämisen AWS-pilvipalveluihin. Siirtyminen AWS:n palveluihin mahdollistaa palvelulle paremman skaalautuvuuden, joka parantaa vasteaikoja ja täten asiakkaan käyttökokemusta.

Kuviossa 10 on hahmoteltuna PDX palvelujen uusi looginen toimintamalli.



Kuvio 10. PDX uusi toimintamalli

Ohjelmiston uuden toimintamallin edut vanhaan toteutukseen verrattuna ovat merkittävät niin palvelun laadun kuin kehityksen kannalta.

6 Johtopäätökset ja pohdinta

Työn tavoitteet ovat toteutuksen aikana muuttuneet merkittävästi ja alkuperäistä opinnäytetyöprojektia ei lopulta päästy toteuttamaan suunnitelman mukaisesti.

Uusi aihe ei soveltunut mielestäni hyvin opinnäytetyöksi, mutta yrityksellä ei ollut tarjota sopivaa tarpeeksi rajattua projektia. Aiheen etsintä yrityksen ulkopuolelta olisi ollut hankalaa ajankäytön kannalta, koska olin vakituisessa työsuhteessa ohjelmistokehittäjänä yrityksessä. Tämän lisäksi en halunnut hylätä tekemääni työtä projektin eteen, joten päätin arvioida työn aihetta uudelleen.

Toisin kuin tyypillisessä opinnäytetyöprojektissa, minulla ei ollut hyvin määriteltyä ja rajattua osuutta toteutetusta kokonaisuudesta. Vaikka osallistuin komponenttien suunnitteluun ja toteutukseen, ei voi sanoa, että olisin suunnitellut tai toteuttanut niistä ainuttakaan. Tämän seurauksena opinnäytetyö raportin kirjoittaminen osoittautuikin odotettua vaikeammaksi ja työstä muodostui lopulta eräänlainen seurantaraportti käytännön ohjelmistokehityksestä 3 vuoden ajalta.

Päätin aloittaa raportin kirjoittamisen uudelleen käyttöliittymäprojektin ensimmäisen vaiheen julkaisun jälkeen, jotta voisin yrittää tarkastella projektia yhtenä kokonaisuutena. Tästä huolimatta suurin haaste raportin kirjoittamisessa oli kokonaisuuden rajaaminen, jonka seurauksena raportista tuli jokseenkin hajanainen. Halusin alkujaan sisällyttää raporttiin enemmän yksityiskohtia rajapinnan ja vanhan PDX:n toteutuksesta, mutta monien yksityiskohtien lisääminen olisi vaatinut niiden salaamista julkaisussa ja en tahtonut aiheuttaa työn toimeksiantajalle ylimääräistä työtä raportin tarkastamisessa.

Kehitysprojekti on tuonut selkeästi esiin ongelmia, joihin ohjelmistokehittäjät törmäävät vanhan ohjelmiston uudistuksessa. Vanhan PDX:n koodi on vuosien varrella kehittynyt erittäin vaikealukuiseksi. Ongelman syntymiseen ovat vaikuttaneet osaksi vanhentuneet kirjastot ja toteutustavat, mutta näitä suurempi tekijä on kehittäjät. Ohjelman elinkaaren aikana sen kehittäjät ovat vaihtuneet moneen kertaan, ja he ovat jokainen jättäneet oman tyylinsä aikanaan tekemiinsä toteutuksiin. Kehittäjien määrän vuoksi ohjelman koodilla ei enää ole yhtä yhtenäistä toteutustapaa, ja tämä tekee koodin tulkitsemisen erittäin vaikeaksi.

PDX-ohjelmassa tätä ongelmaa kärjistää sen monoliittinen toteutus. Työskentely vanhan PDX:än kanssa onkin opettanut, että jatkuvassa kehityksessä oleva ohjelmisto kehittyy väistämättä vuosien varrella ennakoimattomiin suuntiin. Riittävän ajan ku-

luttua ohjelman toteutukseen on osallistunut niin moni henkilö, että kokonaisuus alkaa muuttua epäjohdonmukaiseksi ja sen kehityksestä tulee vaikeaa. Tämän ongelman minimoimiseksi onkin tärkeää, että ohjelma olisi kehitetty mahdollisimman selkeisiin ja toisistaan eriytettyihin komponentteihin. Ohjelman jakaminen tällä tavoin auttaa kehittäjiä tulevaisuudessa uusimaan ohjelmaa pienemmissä osissa. Pienemmät osakokonaisuudet on helpompi pitää toteutukseltaan yhdenmukaisina ja tarpeen tullen korvata.

Käyttöliittymäprojektin ensimmäisessä julkaisussa nähtiin uuden toteutuksen edut, ongelmien vähäisenä määränä. Ennen julkaisua tiimissä oli varauduttu valitusten tulvaan, mutta odotusten vastaisesti julkaisu sujui ilman suuria ongelmia. Julkaisussa ilmenneet ongelmat näyttivätkin kuinka ohjelman jakaminen osiin auttaa eristämään ongelmat, rajoittaen niiden vaikutusta ohjelman toimintaan. Ongelmien rajattu luonne auttoi myös niiden selvittämisessä ja nopeutti ratkaisujen löytämistä.

Tehtyjen muutosten jälkeen kehitysprojekti on jatkunut aikataulussa ja seuraava osio julkaistaan 20 päivä toukokuuta 2020. Tiimi onkin jo aloittanut kolmannen julkaisu kokonaisuuden kehityksen.

Uudistukseen käytetty aika onkin osoittautunut tehokkaasti käytetyksi projektin tulevaisuuden kannalta. Vanhan PDX:n vaiheittainen käytöstä poistuminen säästää kehittäjiltä paljon aikaa. Vaikka tässä vaiheessa ei voi sanoa, että säästetty kehitysaika olisi verrattavissa projektiin käytetyn ajan kanssa, voi tämä tilanne muuttua nopeasti vuosienvarrella.

Lähteet

About Memcached. N.d. Artikkeli memcachedin sivuilla. Viitattu 13.5.2020. https://www.memcached.org/about

Active Directory Domain Services Overview. 2017. Katsaus aktiivihakemistoon Microsofti sivuillla. Viitattu 10.5.2020. https://docs.microsoft.com/en-us/windows-server/identity/ad-ds/get-started/virtual-dc/active-directory-domain-services-overview

An Introduction to JavaScript. 2020. Artikkeli javacript.info-sivustolla. Viitattu 10.5.2020. https://javascript.info/intro

Coposer: Introduction. N.d. Composer-ohjelman esittely. Viitattu 12.5.2020. https://getcomposer.org/doc/00-intro.md

Confluence: What is Confluence? N.d. Confluence esittely Atlasianin verkkosivuilla. Viitattu 10.5.2020. https://confluence.atlassian.com/confeval/confluence-evaluator-resources/confluence-what-is-confluence

Aman, G. 2020. MariaDB vs Mysql. Hackr.oi-sivustolta löytyneestä blogista. Viitattu 10.5.2020. https://hackr.io/blog/mariadb-vs-mysql

History of PHP. N.d. Verkkoartikkeli. Viitattu 10.5.2020. https://www.php.net/manual/en/history.php.php

HTTP Methods. N.d. Verkkoartikkeli. Viitattu 10.5.2020. https://restfulapi.net/http-methods/

Hoyos, M. 2018. What is an ORM and Why You Should Use it. Artikkeli blogista. Viitattu 13.5.2020. https://blog.bitsrc.io/what-is-an-orm-and-why-you-should-use-it-b2b6f75f5e2a

Lerner, A. N.d. What is react. Artikkeli Newline-verkkosivustolla. Viitattu 13.5.2020. https://www.newline.co/fullstack-react/30-days-of-react/day-1/

Oikotie Asunnot Oy ostaa liiketoimintakaupalla LKI-Asiantuntijapalvelut Oy:n liiketoiminnan. 2017. Artikkeli M&M lehden verkkosivulla. Viitattu 10.5.2020. https://www.marmai.fi/uutiset/oikotie-ostaa-asuntomyynnin-ja-markkinoinnin-palveluja-tuottavan-suomalaisyrityksen-asuntokauppa-on-digitalisoitumassa-kovaa-vauhtia/9eea43f4-ccc8-33d8-9880-b5aa27ccf7c7

Oikotie Oy. N.d. Oikotie Oy tiedot finder-palvelussa. Viitattu 10.5.2020. https://www.finder.fi/Internet-palvelut/Oikotie+Oy/Helsinki/yhteystiedot/2026545

PDX. N.d. PDX-tuotteen esittely PDX-ohjelmistojen verkkosivuilla. Viitattu 10.5.2020. https://www.pdx.fi/

PhpStorm. N.d. PhpStorm IDE esittely jetbrains-yrityksen verkkosivulla. Viitattu 10.5.2020. https://www.jetbrains.com/phpstorm/

Restler. N.d. Artikkeli Luracastin verkkosivuilta. Viitattu 10.5.2020. https://www.luracast.com/products/restler/

Sanoma vuosikatsaus. 2018. Sanoma Vuosikatsaus vuodelle 2018. Viitattu 26.5.2020. https://sanoma.com/wp-content/uploads/2019/03/Sanoma_Vuosikatsaus_2018-1.pdf

The leading cloud platform. N.d. AWS-palvelun esittelysivu Amazonin verkkosivuilla. Viitattu 10.5.2020. https://aws.amazon.com/what-is-aws/

Anshul, V. 2018. Artikkeli medium-sivustolla. Viitattu 13.5.2020. https://medium.com/@anshul.vyas380/mvc-pattern-3b5366e60ce4

What is a VPN? N.d VPN:n esittely Mcafee-sivustolla. Viitattu 10.5.2020. https://www.mcafee.com/en-us/vpn.html

What is Git. N.d. Git-työkalun esittely atlassianin verkkosivuilla. Viitattu 10.5.2020.

https://www.atlassian.com/git/tutorials/what-is-git

What is Jira used for? N.d. Jiran esittely Atlassianin verkkosivuilla. Viitattu 10.5.2020. https://www.atlassian.com/software/jira/guides/use-cases/what-is-jira-used-for

What is PHP? N.d. Verkkoartikkeli. Viitattu 10.5.2020. https://www.php.net/manual/en/intro-whatis.php

What is REST. N.d. Verkkoartikkeli. Viitattu 10.5.2020. https://www.smashingmagazine.com/2018/01/understanding-using-rest-api/

What is Slack? N.d. Slack-työkalun esittely slackin verkkosivulla. Viitattu 10.5.2020. https://slack.com/intl/en-fi/help/articles/115004071768-What-is-Slack-