Haaga-Helia
ammattikorkeakoulu Oy

Palvelinten tilan raportointinäkymä

Niko Virolainen

Opinnäytetyö
Tietojenkäsittelyn
koulutusohjelma
2020

Haaga-Helia
ammattikorkeakoulu Oy

Tiivistelmä

Tekijä
Niko Virolainen

Koulutusohjelma
Tietojenkäsittelyn koulutusohjelma

Raportin/Opinnäytetyön nimi
Palvelinten tilan raportointinäkymä

Sivu- ja liitesivumäärä
29 + 0

Opinnäytetyössä rakennettiin toimeksiantona Kelan uusien verkkosivujen palvelinten tiloja tarkkaileva raportointinäkymä. Raportointinäkymän rakennuksen lisäksi työssä perehdytään liiketoiminnassa ja sovelluskehityksessä käytettävien raportointinäkymien tarkoituksiin ja eroihin.

Tavoitteena oli rakentaa Kelan uusien verkkosivujen kehitysryhmälle raportointinäkymä, joka tarkistaa tietyin väliajoin kehityksessä, testauksessa ja tuotannossa käytettävien palvelinten tiloja. Raportointinäkymän haluttiin olevan visuaalisesti selkeä, jotta sitä voisi hyödyntää kehityksen tukena nopeasti tulkitsemaan palvelimista muodostuvien palveluiden, eli verkkosivujen tiloja. Raportointinäkymässä haluttiin myös esittää palvelimilla pyörivän sisällön versiotietoja.

Tuloksena syntyi kokonaan JavaScript-ohjelmointikielellä rakennettu raportointinäkymä, jossa on omat välilehdet jokaisen ympäristön palvelimille. Palvelinten tiloja korostetaan kuvakkeilla ja selkeällä väriteemalla, jossa vihreä tarkoittaa toimintakunnossa olevaa palvelinta ja punainen virhetilassa olevaa palvelinta. Jokaisella välilehdellä on useita palvelinkortteja, joista yhtä palvelinta edustaa yksi palvelinkortti. Palvelimilla pyörivien sisältöjen versiot esitetään palvelinkorttien alareunassa.

Työ aloitettiin vuoden 2020 helmikuussa ja raportointinäkymä otettiin käyttöön kehityksen tueksi vuoden 2020 marraskuussa. Työtä edistettiin Scrum-työskentelymallin mukaan osana kehitysryhmää.

Asiasanat
Palvelimet, palvelinvalvonta, monitorointi, raportointi, JavaScript

Sisällys

1 Johdanto

Tämän opinnäytetyön tarkoituksena oli rakentaa palvelinten tilaa tarkkaileva ja esittävä raportointinäkymä. Projektissa sidosryhminä toimi Kelan Digitaalisten Palvelujen IT-yksikön sisäiset kehitystyöryhmät. Pääsidosryhmänä toimi kehitystyöryhmistä nimettömänä pysyvä ryhmä, jonka tehtäviin kuuluu Kelan uusien verkkosivujen kehittäminen. Kun opinnäytetyössä viitataan ryhmään, kehitysryhmään tai työryhmään, puhutaan juuri tästä kyseisestä pääsidosryhmästä. Kaikki päätökset raportointinäkymän vaatimuksia, ominaisuuksia ja ulkoasua koskien toteutettiin kyseisessä ryhmässä.

Raportointinäkymän tarkoitus on toimia pääsidosryhmän tukena Kelan uusien verkkosivujen kehittämisessä. Tavoitteena on se, että raportointinäkymä on kehittäjällä selaimessa auki luonnollisesti kehittämisen aikana, ja että se on ensimmäinen paikka, minkä kehittäjä tarkistaa kohdatessaan ongelman, jonka hän olettaa liittyvän palvelimiin. Opinnäytetyössä myös selvitetään erilaisten liiketoiminnassa ja sovelluskehityksessä käytettävien raportointinäkymien ominaisuuseroja. Raportointinäkymän tarkkailemat palvelimet ovat Kelan uusien verkkosivujen kehitys-, testaus- ja tuotantopalvelimia. Raportointinäkymää rakentaessa tarkoituksena oli luoda juuri Kelan verkkosivujen työryhmälle spesialisoitu raportointinäkymä, mutta niissä puitteissa, että sitä on mahdollista tietyllä dokumentoiduilla muokkauksilla soveltaa muidenkin Kelan työryhmien käyttöön.

Teknologiat-osiossa kuvaillaan teknisiä asioita, joilla raportointinäkymä on rakennettu, ja avataan opinnäytetyössä käytettävää terminologiaa. Raportointinäkymä-osiossa kuvataan erilaisia näkymien käyttötarkoituksia, ja avataan ”raportointinäkymä”-sanan taustoja liittyen tähän opinnäytetyöhön. Raportointinäkymän toteutusratkaisut-osio syventyy toteutustapaan ja tyylittelyratkaisuihin. Raportointinäkymän tarve-osiossa kerrotaan, miksi kyseinen projekti on käynnistetty. Tulokset ja pohdinta-osiossa kerrotaan raportointinäkymän käytön tuloksista opinnäytetyön kirjoittamiseen mennessä, arvioidaan tulevaisuuden käytettävyyttä, mietitään sekä käytettyjä että mahdollisia ratkaisuita, sekä pohditaan raportointinäkymän rakennuksessa onnistuneita ja epäonnistuneita asioita.

2 Teknologiat & työskentelytavat

Kun opinnäytetyössä viitataan frontend-kehitykseen, tarkoitetaan sillä raportointinäkymän palvelinten tilojen esittämispaikan kehitystä, kun taas backend-kehityksestä puhuttaessa tarkoitetaan palvelinten tilojen hakemisen ja tarkistamisen logiikkaa. Raportointinäkymä on kokonaisuudessaan rakennettu JavaScript-ohjelmointikielellä.

Palvelimella opinnäytetyössä tarkoitetaan laitteistoa, joka ottaa vastaan internetin yli välitettyjä pyyntöjä, ja lähettää pyynnön ollessa oikein muotoiltu vastauksena kutsujalle sen pyytämää dataa. Tyypillisesti palvelimella viitataan tietokonejärjestelmään, joka vastaanottaa pyyntöjä verkkosivuista ja palauttaa niitä. (Computerhope 2020.)

Kehitys-, testaus- ja tuotantoympäristöt toimivat esimerkiksi verkkosivuja rakentaessa tärkeässä roolissa verkkosivujen edistämisprosessissa. Tuotantoympäristössä on asiakkaille näkyvät valmiit verkkosivut, jotka oletettavasti toimivat toivotulla tavalla. Kehitysympäristössä sen sijaan on vain kehittäjille näkyvää sisältöä, ja vaikka sen yleensä toivottaisiin muistuttavan mahdollisimman paljon tuotantoympäristön sisältöä, voi tällä tasolla vielä olla virheellistä toiminnallisuutta. Testausympäristö on kahden edellisen välimalli. Testausympäristöön asetetaan sisältöä kehitysympäristöstä, ja tavoitteena on, että se toimii mahdollisimman virheettömästi, jotta se voidaan siirtää eteenpäin tuotantoon. (James 9.5.2016.) Kaikille ympäristöille on tässä opinnäytetyössä omat palvelimet.

2.1 JavaScript ja JS-kirjastot

JavaScript on ohjelmointikieli, jolla rakennetaan pääosin toiminnallisuutta verkkosivuille ja sen vahvoja puolia on se, että sillä voi tehdä dynaamisesti päivittyvää sisältöä ja vaikkapa animoida verkkosivun kuvia (Mozilla Developer 2020). Tässä opinnäytetyössä käytetään kahta JavaScriptille rakennettua kirjastoa, Reactia ja Nodea, jotka helpottavat kehittämistä valmiiksi sisäänrakennetulla toiminnallisuudellaan.

ReactJS on JavaScript-kirjasto, jonka primääritarkoitus on helpottaa kehittäjien työtä tarjoamalla helposti implementoitavia palasia interaktiivisen käyttöliittymän rakennukseen (ReactJS 2020). Nodekin on JavaScript-kirjasto, mutta sen tarkoituksena on tarjota kehittäjille mahdollisuudet rakentaa palvelimilla tapahtuvaa sovelluksen tai verkkosivujen logiikkaa (W3Schools 2020). Node on käytännössä se työkalu, joka käsittelee palvelimelle saapuvia verkkosivu-pyyntöjä ja tarjoaa vastauksia pyydetyssä muodossa.

2.2 Termit, teknologiat ja työskentelytavat

Termi	Selitys
JSON	JSON muodostuu sanoista ”JavaScript Object Notation” ja se on yleensä JavaScriptissä tiedonvälitykseen käytetty tiedostomuoto (W3Schools 2020).
API	API muodostuu sanoista ”Application Programming Interface” ja se mahdollistaa sovellusten välisen kommunikoinnin. Esimerkiksi sijainnin tarkistaminen puhelimen karttasovelluksella käyttää APIa (Mulesoft 2020).
NPM	NPM eli “Node Package Manager” on verkosta löytyvä NodeJS-projektien säilytyspaikka, joita kehittäjät voivat implementoida sovelluksiinsa, ja jota käytetään yleensä tietokoneen terminaalin kautta (NodeJS 2020).
PROXY	Proxy Server eli tuttavallisemmin pelkkä proxy, toimii porttina käyttäjän ja verkkosivun välissä ja sen tarkoitus on sekä lisätä käyttäjän turvallisuutta, että parantaa suorituskykyä (Petters 10.6.2020).
CORS	CORS (Cross-Origin Resource Sharing) on verkkosivuilla käytetty mekaniikka, joka hallitsee sekä sitä, että kenellä on lupa vierailla verkkosivulla ja sitä millä tavoin vierailija saa verkkosivuilla vierailla (Codeacademy 2020).
HTTP	HTTP eli ”Hyper Text Transfer Protocol” tarkoittaa protokollaa, jota käytetään tiedonsiirtoon selainten ja palvelimien välillä. HTTP:seen kuuluu statuskoodeja, jotka indikoivat palvelimen tilaa. (W3schools 2020)

Taulukko 1. Opinnäytetyössä käytettyjä relevantteja termejä.

OpenShift on RedHatin tuottama kehitysalusta, jonka tarkoituksena on jakaa osia sovelluksista omiin tiloihin, joita kutsutaan konteiksi. Koska kontteja voi monistaa niin samoista verkkosivuista voi tehdä monta samanlaista konttia, joka mahdollistavat verkkosivujen käyttäjien tuottaman kuorman jakamista konttien kesken, joka taas parantaa verkkosivujen suorituskykyä (Hälikkä 1.11.2017). Opinnäytetyössä rakennettavassa raportointinäkymässä hyödynnetään OpenShiftiä.

Olennainen osa raportointinäkymän kehitysputkea on OpenShiftin lisäksi Jenkins. Martin Heller kuvailee artikkelissaan (What is Jenkins? The CI server explained. 2020) Jenkinsin olevan automaatiotyökalu, joka tarjoaa helpon tavan pystyttää jatkuvaan integraatioon perustuvan ympäristön. Jatkuvalla integraatiolla viitataan kehittäjien tekemien päivittäisten muutosten vaikutukseen IT-projektissa. Jenkins otettiin käyttöön opinnäytetyössä OpenShiftin avuksi, ja se on helpottanut projektin edistämistä rakentamalla tuotantoval-

miin version raportointinäkymästä tasaisin väliajoin, eli yleensä päivittäin. Päivittäisiä versioita tutkimalla on saattanut huomata raportointinäkymää rikkovia muutoksia, jotka ovat kehitysvaiheessa jääneet havaitsematta.

Raportointinäkymän koodia säilytetään versionhallintaan tarkoitetussa sivustossa, Bitbucketissa. Bitbucket on Atlassianin kehittämä työkalu, jonka tarkoitus on säilyttää rakennettavan sovelluksen tai verkkosivun koodia, johon pääsee käsiksi työtilaan lisätyt kehittäjät, ja jossa he voivat vertailla, kommentoida ja hyväksyä toistensa tekemiä muutoksia projektiin (Bitbucket 2020). Bitbucket on ollut raportointinäkymän kehityksessä käytössä alusta asti, ja sitä hyödynnetään myös Kelan uusien verkkosivujen rakennusprojektissa. Bitbucketissa on mahdollista luoda omia tehtävätauluja, mutta raportointinäkymän tehtävien jaottelemiseen käytettiin Atlassianin toista tuotetta, Jiraa. Jira on projekteille tarkoitettu työtila, jossa erilaisia työtehtäviä voi pilkkoa pienempiin osiin. Osia kutsutaan tiketeiksi ja tiketeillä on erilaisia tilannekuvauksia, kuten ”kehityksessä” tai ”valmis”. Tiketeistä muodostuvaa kokonaisuutta kutsutaan työtauluksi, josta on helppo tarkkailla projektiin liittyvien tehtävien tilaa.

Raportointinäkymää rakentaessani toimin myös kehittäjänä Kelan uusien verkkosivujen kehitysryhmässä, ja kaikki raportointinäkymään liittyvät tiketit sijaitsivat kyseisen kehitysryhmän Jira-taululla, jotta raportointinäkymän edistystä pystyttiin seuraamaan mahdollisimman tarkasti. Ryhmässä toimitaan hyödyntäen ketterän ohjelmistokehityksen Scrum-mallia. Scrum kuvaillaan Ken Schwaberin ja Jeff Sutherlandin oppaassa (Scrum-opas 2017) viitekehykseksi, jonka avulla voidaan luovasti ratkaista monimutkaisia ongelmia liittyen suuria lisäarvoja tuoviin projekteihin eli käytännössä Scrumilla hallitaan monia työn sisällä hyödynnettäviä eri prosesseja ja tekniikoita.

3 Raportointinäkymä

> ”*Dashboard*-sana on hankala suomennettava: sen lisäksi, että se on irtautunut alkuperäisestä merkityksestään (ajoneuvon fyysinen kojetaulu), sen käyttö tietotekniikankin kielessä on hyvin laveaa. Siksi ei pärjätä yhdellä sanalla, vaan suomenkielinen vastine on mietittävä tapauskohtaisesti. Esimerkiksi sisällönhallintaohjelmiston käyttöliittymässä *dashboard* on suomennettu *ohjausnäkymäksi*. Muita mahdollisia vastineita ovat *ohjaus*- tai *hallintapaneeli* ja *ohjaus*- tai *hallintatyökalu*. *Dashboard*-sanan tuorein käyttötapa taitaa liittyä liiketoimintatiedon esittämiseen. Tällöin viitataan visuaalisesti toteutettuun näkymään, jossa organisaation toimintaan liittyviä tietoja ja lukuja esitetään pelkistetyssä muodossa. Tällaisesta on käytetty ainakin nimityksiä *visuaalinen mittaristo*, *visuaalinen raportointityökalu* ja *raportointinäkymä*.” (Kielitohtori 2020.)

Yllä olevassa Kielitohtori-sivulta otetussa lainauksessa tiivistetään hyvin *dashboard*-sanan suomentamisen vaikeus, ja syyt siihen, miksi tässä työssä käytetään sanaa raportointinäkymä. Sanana raportointinäkyvä kuvaa projektia parhaiten, sillä toimeksiannon ydin oli luoda näkymä, joka näyttää palvelinten tilaa ja virhetilanteissa vain informoi käyttäjälle virheen ytimen, mutta ei tarjoa mahdollisia toimintoja virheen korjaamiseen.

Sekä monitorointinäkymä, että ohjausnäkymä viittaavat näkymän toiminnalliseen tarkoitukseen, ja molemmat niistä niminä vihjaavat kehittäjälle, että jotain toimintoja olisi itse näkymässäkin tehtävissä. Näiden syiden vuoksi päädyttiin sanaan raportointi, joka Suomisanakirjan mukaan määritellään mm. tiedotukseksi, tiedonannoksi ja tilannekatsaukseksi, joista kaikki kuvaavat hyvin projektin tarkoitusperää.

3.1 Raportointinäkymien käyttötarkoitukset

Raportointinäkymä on yleensä sovelluskehityksessä tai markkinoinnissa käytetty työkalu, joka varastoi, järjestää ja visualisoi jonkinlaista hyödyllistä dataa. Koska raportointinäkymiä on rakennettu paljon, niin luonnollisesti niille löytyy useanlaisia erilaisia käyttötarkoituksiakin. Yleisesti niitä hyödynnetään ainakin liiketoiminnassa ja sovelluskehityksessä, joskin molemmissa hieman eri tarpeisiin. Tämän raportointinäkymän pääkäyttötarkoituksena on valvoa palvelimia, jotka sisältävät uusien Kelan verkkosivujen eri kehitys-, testaus- ja tuotantovaiheessa olevaa sisältöä.

Raportointinäkymät voidaan eritellä neljään isoon eri kategoriaan: Business Dashboard, Executive Dashboard, KPI (Key Performance Indicators) Dashboard ja Project Dashboard. Yleensä kaikenlaiset liiketoiminnassa hyödynnetyt raportointinäkymät luokitellaan Business Dashboard-kategoriaan, koska niiden päätarkoituksena on tuottaa informaatiota yrityksen tärkeistä liiketoiminnan metriikoista osakkeenomistajille tai Executive Dashboard-kategoriaan, koska niiden tarkoituksena on visualisoida yrityksen osakkeenomistajille elintärkeätä yrityksen pyörittämiseen käytettävää informaatiota. Executive Dashboard eroaa Business Dashboardista informaation kohteissa; toinen (Executive

Dashboard) näyttää yrityksen työntekijöihin ja osastojen tapaamisiin kohdistuvaa dataa ja toinen (Business Dashboard) yrityksen myynteihin ja isoihin suuntaa antaviin päätöksiin kohdistuvaa dataa. KPI Dashboardin päätarkoitus on seurata yrityksen osastoille asetettavia tavoitteita ja tarjota niiden edistymiseen liittyvää monitorointia. Project Dashboardien ideana on seurata koko projektin valmistumisaikataulua ja yrittää arvioida projektin työntekijöiden taakkoja projektin suhteen, eli esimerkiksi että onko tietyillä työntekijöillä liikaa työtehtäviä suhteutettuna projektin aikatauluun. (iDashboards 2020.)

Margaret-Anne Storey ja Christoph Treude (Software Engineering Dashboards: Types, Risks and Future 2019) kuvailevat sovelluskehityksessä raportointinäkymiä yleensä käytettävän tuottamaan informaatiota keskeneräisestä tuotteesta näkymien käyttäjille tai kehittäjille, sekä kehitysprosessin apuna näyttämään kehitysprosessia tukevia tai vastustavia analyysejä, kun taas Klipfolion raportointinäkymäesimerkkien mukaan liiketoiminnassa raportointinäkymät yleensä tarjoavat niin kutsutun ”at-a-glance” eli yhden vilkaisun ratkaisuita, jotka tarjoavat metriikkaa esimerkiksi yrityksen myyjien tilastoihin tai myytävän tuotteen menestykseen liittyen.

Tämän projektin puitteissa rakennetun raportointinäkymän tarkoitus on tarkistaa ja esitellä erilaisten kehityksessä hyödynnettävien palvelinten tilaa, joten sen voisi kuvailla tuottavan informaatiota keskeneräisen tuotteen *palvelimista,* ei niinkään tuotteesta itsessään, koska tuote on ikään kuin kaikkien palvelinten summa sen ollessa tässä tapauksessa verkkosivu. Raportointinäkymä näin ollen siis yhdistelee sekä sovelluskehityksessä, että liiketoiminnassa käytettävien raportointinäkymien eri ominaisuuksia, kuten esimerkiksi yhden vilkaisun ratkaisun, sekä palvelinten informaation näyttämisen.

Maailmalta löytyy varsin kattava kirjo kaikenlaisessa toiminnassa hyödynnettäviä raportointinäkymämalleja ja tätä projektia suunnitellessa tutkittiin mahdollisuuksia hyödyntää niin jo valmiiksi tehtyjä kokonaisuuksia, kuin pelkkiä tyylittelymallinteitakin. Erilaisia ratkaisuita on sekä maksullisia, että maksuttomia, joista kumpiakin ratkaisuita mietittiin, mutta lopputulemaksi muodostui kuitenkin raportointinäkymän alusta alkaen itse rakentaminen perinpohjaisesti.

Suurin haaste maailmalta löytyvissä raportointinäkymissä on se, että ratkaisuissa on usein paljon ylimääräisiä ominaisuuksia, joita ei nähty ollenkaan tarpeellisiksi tässä projektissa. Ominaisuuksien karsiminen jo valmiista näkymistä osoittautui jo vähintäänkin yhtä työlääksi, kuin näkymän omakätinen rakentaminen. Projektin raportointinäkymän tarkoituksena ei ole esitellä yrityksen taloudellista metriikkaa tai myyntiä, eikä monitoroida palvelun

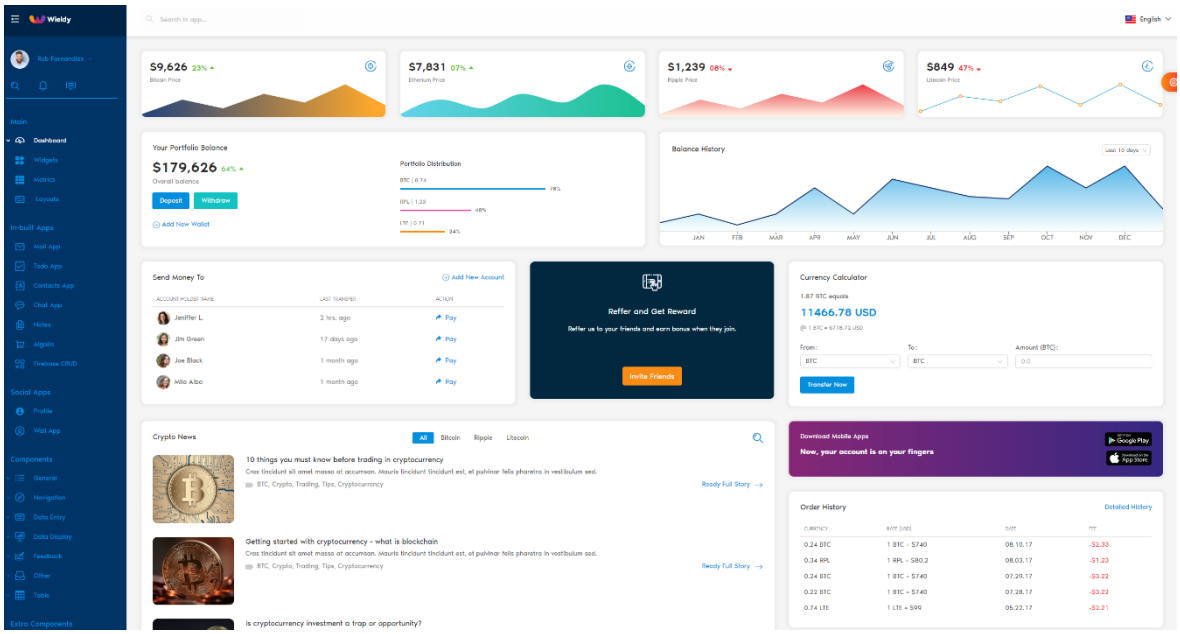
edistymistä, vaan informoida kehittäjiä palvelun palvelinten tiloista. Jokainen tutkittu ratkaisu sisälsi jollain tavoin toiminnallista logiikkaa liittyen tilastollisiin edistymisiin tai myyntilukuihin. Näiden mainittujen asioiden lisäksi oltiin tyytyväisiä ajatukseen, että raportointinäkymän kehityksen edistyessä toteutustapaa voitaisiin rauhassa muokata palvelinten kutsunnan mukaan. Mahdollisten valmiiden ratkaisuiden hyödyntämisessä oli riskinä se, että palvelimien kutsumis-, tai ulostuontiratkaisuita olisi pitänyt sovittaa valmiin ratkaisuin toiminnallisuuden puitteisiin.

Raportointinäkymässä esiteltävät palvelimet on luokiteltu niiden käyttötarkoitusten mukaan. Ensimmäiseltä välilehdeltä löytyy kehitykseen käytettävän verkkosivun palvelimet, toiselta sivulta löytyy testaukseen käytettävän verkkosivun palvelimet, ja kolmannelta sivulta löytyy tuotannossa olevan verkkosivun palvelimet. Neljäs välilehti on varattu kolmelle eri palvelimelle, jotka kuuluvat samaan verkkosivuun, mutta joita hyödynnetään sekä kehityksessä, testauksessa, että tuotannossa. Tämä luokittelu koettiin mahdollisista luokittelutavoista selkeimmäksi kehittäjien näkökulmasta katsottuna. Toinen luokittelutapa olisi ollut asetella palvelimet välilehdille sen mukaan, missä kohdissa verkkosivua palvelimia hyödynnetään, eli esimerkiksi verkkosivulla olevaan hakuun liittyvät palvelimet tietyllä välilehdellä ja artikkeleihin liittyvät palvelimet omallaan, mutta kehittämisen kannalta selkein tapa on luokitella palvelimet kehitysympäristön mukaan, koska silloin ei tarvitse kehityksen aikana navigoida välilehdeltä toiselle, kun kehitykseen liittyvät palvelimet sijaitsevat samassa paikassa.

3.2 Miten raportointinäkymiä hyödynnetään maailmalla

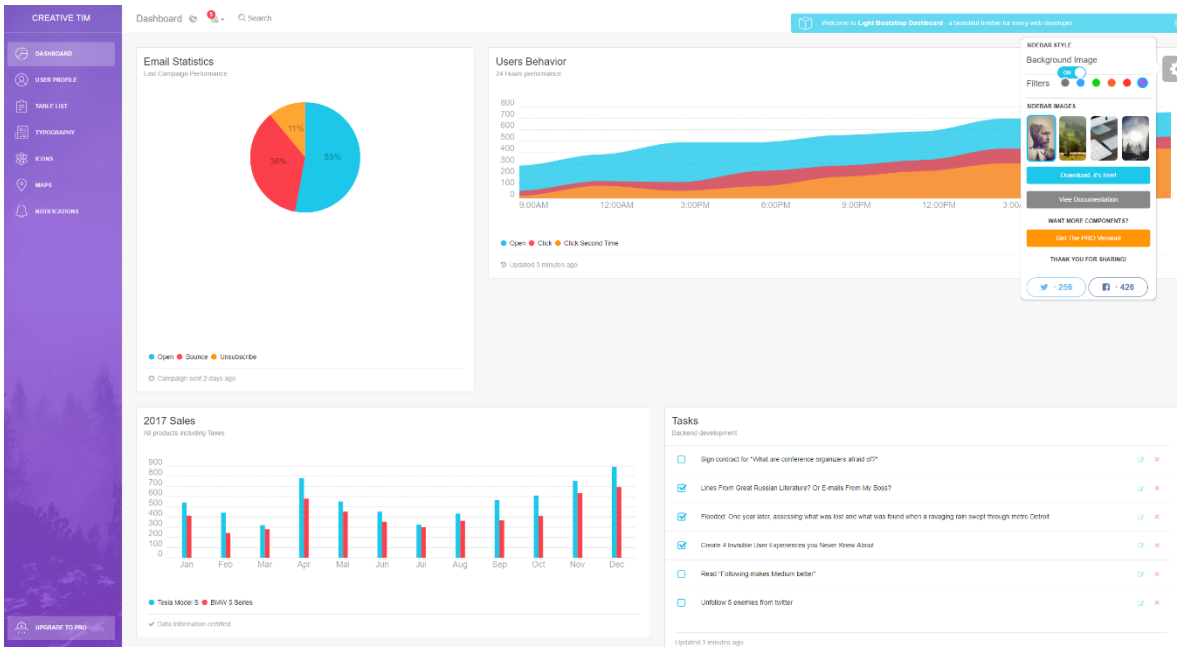
Maailmalla on paljon erilaisia raportointinäkymiä ja raportointinäkymän rakentamiseen hyödynnettäviä malleja, joita projektin alussa tutkittiin. Jo nopealla google-haulla kuvauksia erilaisista näkymistä löytyy huomattavan paljon, ja erilaisia reittejä toteuttaa projekti on useita.

Yksi mahdollisuus raportointinäkymän rakennukseen oli hyödyntää eri yritysten ja henkilöiden jo valmiiksi rakentamia sapluunoita. Themeforest-nettisivun kirjastosta löytyy kaikkiaan noin 29 000 erilaista mallia raportointinäkymän rakentamiselle, hintojen vaihdellessa välillä $5-$999. Hakua rajaamalla tämän projektin vaatimuksiin, joista pakollisena on vain React-kirjaston hyödyntäminen, edes millään tavoin käytettäviä sapluunoita löytyykin vain enää 54, joiden hintaluokka on $5-$69. Näistä malleista lähimpänä projektiin soveltuvuutta oli Wieldy-niminen raportointinäkymä, jonka lisenssihinnaksi olisi muodostunut 24 dollaria ja ylläpitohinnaksi 7 dollaria kuussa (Themeforest 2020).



Kuva 1. Wieldyn Dashboard-sapluuna (Themeforest).

Toinen varteenotettava toteutusvaihtoehto oli maksuttomien mallien etsiminen ja vaihtoehtojen tutkimisen jälkeen päädyttiin Creative Tim-yrityksen Light Bootstrap Dashboard-nimiseen sapluunaan.



Kuva 2. Light Bootstrap Dashboard-sapluuna (Creative Tim).

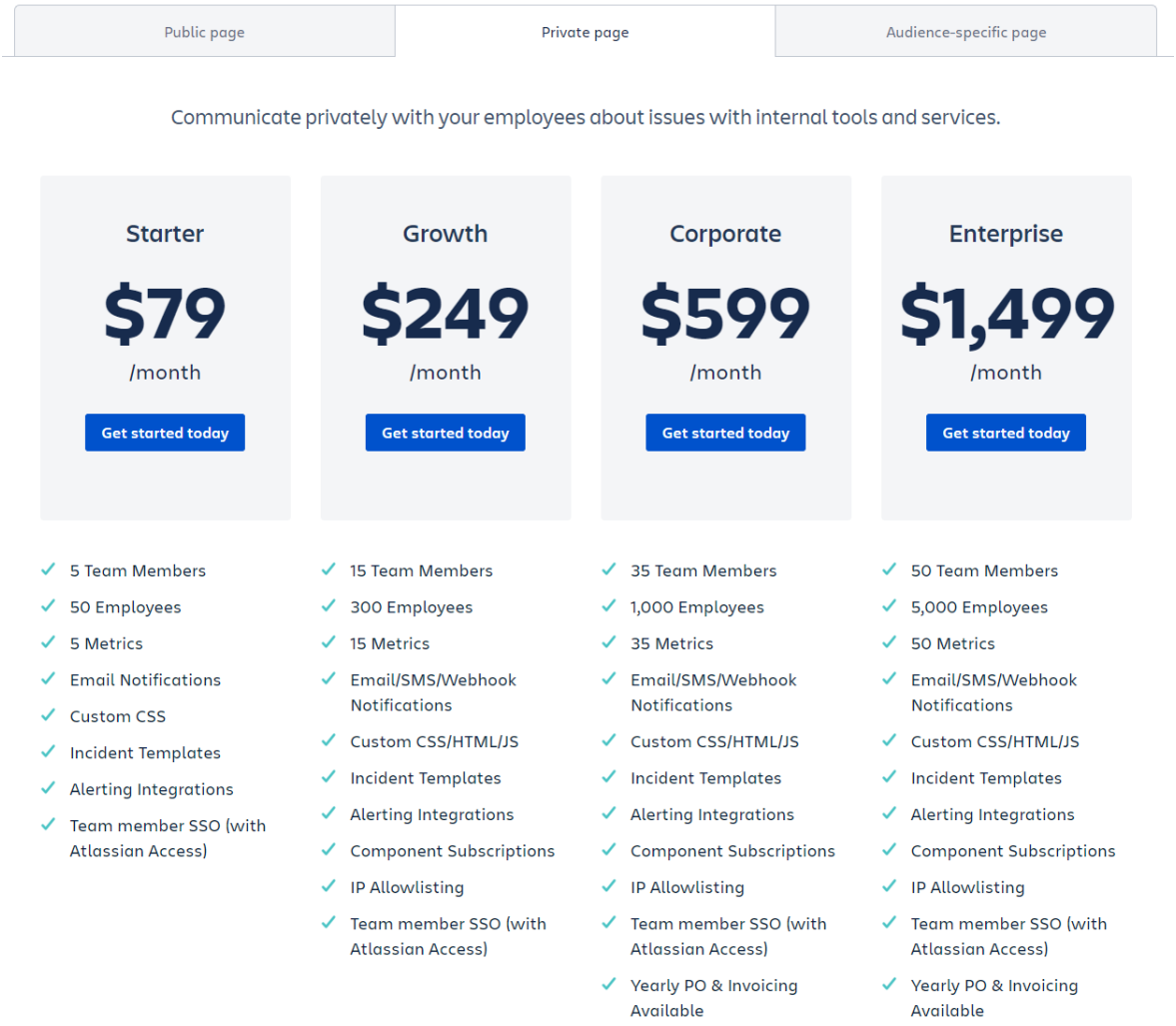
Sekä kuvaa 1, että kuvaa 2 tutkimalla voi huomata esimerkiksi vasemmalla olevan navigaatiopalkin, josta kaikki siinä ilmenevät ominaisuudet ovat tämän projektin puitteissa tarpeettomia. Light Bootstrap Dashboardin suurin ongelma on toimintojen eksessiivisyys. Haluttavien ominaisuuksien valossa sapluunasta olisi pitänyt riisua noin 80% sen mukana

tulevasta koodista, joka olikin ensimmäinen asia, jota yritettiin. Toiminnallisuus ja näkyvät osat olivat sekoitettu Light Bootstrap Dashboardissa keskenään, joka teki pelkän toiminnallisuuden tai pelkän esitettävän asian poistamisesta ilman koko raportointinäkymään vaikuttavaa ketjureaktiota miltei mahdotonta. Koodin riisuminen ja sen myötä eri ominaisuuksien hajoileminen tuotti niin paljon vaikeuksia, että tässä vaiheessa ryhmän yhteinen päätös oli aloittaa raportointinäkymän rakennus alusta asti ilman minkäänlaista sapluunaa. Päätös ei ollut vaikea, sillä tässä vaiheessa projekti oli vielä niin alussa, että turhaa työtä ei oltu tehty lähes ollenkaan.

Yhtenä raportointinäkymän toimeksiantajan vaatimuksista oli se, että näkymä rakennettaisiin On-premises-teknologialla. On-premises-termillä tarkoitetaan ratkaisumallia, jossa rakennettu tai ostettu tuote sijaitsee ostajan tai kehittäjän omilla laitteistoilla pilvipalveluratkaisuiden sijaan (Webopedia 2020). SaaS, eli Software as a Service taasen on palvelumalli, jonka kautta ohjelmiston tilaava yhtiö, yritys tai henkilö saa tilattavan tuotteen jatkuvana maksettavana palveluna pilven kautta, sen sijaan, että tilaaja maksaisi kerralla koko tilattavan tuotteen ja asentaisi sen omalle tietokoneelleen (Pulkkinen 14.12.2017).

On-premises-ratkaisuissa on etuna se, että tuotteen ostajalla on täysi hallinta tuotteesta ostohetkestä eteenpäin, ja ostavan yhtiön kehittäjillä on täydet oikeudet tehdä haluttuja muutoksia tuotteeseen tilanteen vaatiessa. SaaS-ratkaisuissa pitää ottaa huomioon ratkaisua ylläpitävät henkilöt ja kehitysideoiden käsittelyaika, joka muodostuu siitä, kun ratkaisua tarjoavan yhtiön kehittäjiin otetaan yhteyttä ja avataan keskustelua. Kelassa toimitaan pääosin On-premises-mallin mukaan, ja mahdollisen SaaS-mallisen ratkaisun hankkiminen vaatii tiukkojen perusteluiden lisäksi määrittelemättömän määrän aikaa, ehdotuksen mentäessä hyväksyttäväksi eri tahoille. Tämä ei tarkoita SaaS-mallisten ratkaisuiden täystyrmäystä mahdollisessa tulevaisuuden jatkokehityksessä, mutta kun SaaS-malleja harkitaan, tulee raportointinäkymä-ratkaisuun laskea mukaan sen ylläpitokustannuksien lisäksi mahdollisesti pitkäkin ehdotuksen hyväksymisaika. SaaS-palvelumalleista lähimpänä projektin haluttua ratkaisua on Atlassianin Statuspage-niminen raportointinäkymän vuokrausratkaisu.

Atlassianin Statuspage on projekteihin implementoitava ja loppukäyttäjiä opastava verkkosivujen ja palvelujen kokonaisratkaisu, joka toimii sekä tukena verkkosivujen kehittäjille tuottaen tietoa erilaisista virheistä, että informaation tuottajana loppukäyttäjille mahdollisista käyttöön liittyvistä katkoista ja ongelmista. (Atlassian, 2019.)



Kuva 3. Atlassianin Statuspagen yksityisen käytön hinnoittelu-taulukko.

Kuvassa 3 esitellään Statuspagen hinnoittelu yksityiseen käyttöön. Tämän projektin puitteissa alkuperäisen, pelkkään yhden ryhmän käyttöön, sovellettavan ratkaisuun olisi mahdollisesti kehitysryhmän jäsenmäärän puolesta riittänyt Starter-nimellä kulkeva paketti, mutta Starter-paketti ei anna vuokraavalle ryhmälle oikeuksia muokata esimerkiksi JavaScriptillä Statuspagen ominaisuuksia, joka voisi nousta ongelmaksi esimerkiksi kehittäjien halutessa lisätä palvelinten tilan näyttöpaikkaan palvelimella pyörivän sisällön versiotietoja, tai rakennettavien verkkosivujen hakuun liittyvän palvelimen indeksointiaikoja.

Growth-niminen paketti puolestaan tarjoaa mahdollisuuden ominaisuuksienkin muokkaamiseen ja tarjoaa potentiaalisesti kolmellesadalle kehittäjälle mahdollisuuksia käyttää, muokata ja tarkastella raportointinäkymää. Puitteidensa puolesta juuri Growth-paketti olisikin paketeista tähän projektiin kaikista sopivin, ja vaikka 300 kehittäjää onkin yli sen määrän, jota raportointinäkymää samanaikaisesti kehittää, ei ylimääräisistä kehittäjätunnuksista olisi haittaakaan projektille. Statuspagen esittelyn mukaan se kuitenkin on tähdätty enemmänkin asiakaslähtöisille ja *valmiille* projekteille, joka tässä tapauksessa tarkoittaisi

tilannetta, jossa oltaisiin kokonaan tuotannossa, ja verkkosivun valvottavat palvelimet olisivat pääosin tuotantopalvelimia.

Statuspagen keskinäisin ajatus on asiakkaille kommunikointi mahdollisista palvelimilla tai verkkosivuilla esiintyvistä ongelmista, ja toimeksiannon tarkoitus oli rakentaa raportointinäkymä kehittäjille ilman asiakaskosketusta. Statuspage vaikuttaa kyllä projektiin nähden potentiaaliselta jatkokehitysratkaisulta, jos SaaS-ratkaisuita tulevaisuudessa Kelaan implementoidaan, mutta nykyisillä menetelmillä ja kehityksen ollessa vielä pääosassa palvelee itse toteutettu raportointinäkymä paremmin nykytarvetta.

4 Raportointinäkymän toteutusratkaisut

Itse raportointinäkymän käyttöliittymän rakennukseen on käytetty JavaScriptin ReactJS-kirjastoa, ja sen alla oleva logiikka on toteutettu NodeJS-suoritusympäristöä hyödyntäen. Raportointinäkymän ympäristöjen pystytykseen ja ylläpitoon valikoitui Kelan sisällä jo muutenkin rakennukseen ja testaukseen käytetty automaatiotyökalu Jenkins, sekä OpenShift, joka on pilvipohjainen kehitysalusta projektille.

Raportointinäkymässä käytetty toteutusratkaisu johtuu CORS eli ”Cross-Origin Resource Sharing”-rajoitteista, jotka estävät suorien pyyntöjen toteuttamisen frontendistä palvelimille. Ratkaisuna tähän on käytetty proxy-serverinä NodeJS-pohjaista backendiä, joka pystyy kutsumaan palvelimia, kunhan kutsuissa lähetetään mukana lupa hake dataa palvelimelta. Projektikansion package.json-tiedosto sisältää erilaisia projektin käynnistykseen ja kehitykseen liittyviä asetuksia, ja yksi niistä on frontendille annettu lupa kutsua dataa backendistä, jolla puolestaan on lupa kutsua dataa palvelimilta.

4.1 Teknologiat ja toteutus

```
58  app.get("[redacted]", (req, res) => {
59    fetch("[redacted]", options)
60      .then((res) => res.text())
61      .then((res) => res.search("[redacted]"))
62      .then((res) => JSON.stringify(res))
63      .then((data) => {
64        res.send({ data });
65      })
66      .catch((err) => {
67        res.send(err.message);
68      });
69  });
70
71  app.get("[redacted]", (req, res) => {
72    fetch(
73      "[redacted]",
74      options
75    )
76      .then((res) => res.json())
77      .then((res) => res.response.[redacted])
78      .then((data) => {
79        res.send({ data });
80      })
81      .catch((err) => {
82        res.send(err.message);
83      });
84  });
85
```

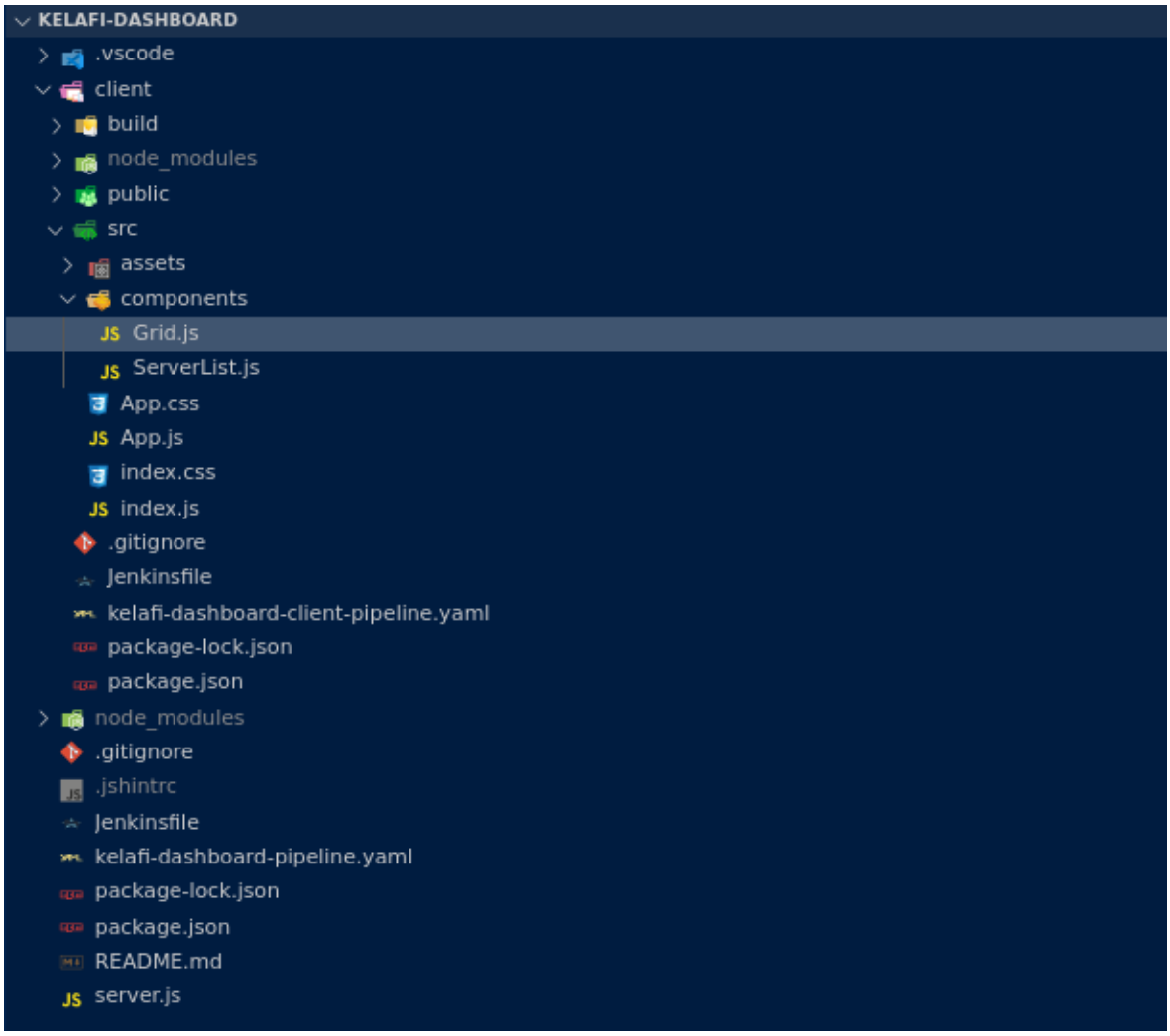
Kuva 4. Kaksi raportointinäkymän backendin palvelinkutsua.

Backendin tarkoitus projektissa on rakentaa logiikka, jossa palvelimia kutsutaan ja niiden tilat määritellään ja tallennetaan APIin, jotta frontendissä kyetään hakemaan yksittäisten

palvelinten tiloja ja esittämään niitä. Palvelimien kutsumistapoja on JavaScriptissä erilaisia ja raportointinäkymän palvelinten kutsumisissa todettiin, että pelkän 200-statuskoodin saaminen palvelimelta ei riitä, koska se ei vielä kerro onko palvelimella olemassa haluttua sisältöä, joten backendiä kehitettäessä selkein ja helpoin tapa rakentaa palvelinkutsuja oli luoda palvelimelle lähetettävä pyyntö, jonka vastauksena palautetaan kutsun lähettäjälle palvelimella pyörivä sisältö. Sisällöstä poimitaan olennainen iso otsikko tai tekstinkappale ja sen sijainti palvelimella tallennetaan JSON-muotoisena APIin. Kuvaa 4 tutkimalla voi nähdä palvelinkutsusta esimerkin. Jos haettava sisältö löytyy, tallennetaan sen sijainti positiivisena numerona, kun taas jos sisältöä ja näin ollen sen sijaintia ei palvelimelta löydy, tallennetaan sijainniksi numero -1.

Frontendin tehtävänä on esittää palvelinten tiloja eri palvelimille määritetyissä paikoissa välilehtiryhmittäin, viitaten kehitys-, testaus- ja tuotantopalvelimiin. Kuvat 6 ja 7 havainnollistavat niiden esitystapaa. Frontendissä kutsutaan APIa, johon on tallennettu palvelinten tilat, ja tarkistetaan sieltä löytyvä numeroarvo. Numeroarvon ollessa positiivinen, todetaan palvelimen toimivan halutulla tavalla. Positiivinen numeroarvo löytyy vain tilanteessa, jos palvelimelta haettava sana vastaa sisällöstä löytyvää sanaa. Jos numeroarvo on -1, palvelin ei esimerkiksi aikakatkaisun takia vastaa, tai palvelin vastaa palauttamalla jonkinlaisen virhetilan, huomauttaa raportointinäkymä palvelimella olevan virheen, ja esittää sen raportointinäkymässä kuvasta 9 löytyvän ”Error:”-tekstin jälkeisen virheilmoituksen tulostamisen sijaan lisäämällä suoraan näkymään klikattavan linkin, joka johtaa kyseisen palvelimen virheenkäsittelysivulle. Jatkokehitysideana voisi todeta, että vaikka se ei raportointinäkymän vaatimuksena ollutkaan, niin ehkä silti koko palvelimella saatavan virheen voisi myös tulostaa virhesivulle ilman sen muotoilemista, jotta sieltä näkisi sekä sen, että palvelin ei ole toimintakunnossa, että palvelimella tapahtuneen kyseisen virheen kokonaisuudessaan.

Raportointinäkymän halutaan luonnollisesti pysyvän ajan tasalla palvelimilla tapahtuvista muutoksista, joten raportointinäkymään on rakennettu ReactJS-kirjaston sisäänrakennetulla ”setTimeout”-komponentilla ajastin, jonka tarkoituksenomaisesti kutsuu viidentoista sekunnin välein palvelimia uudestaan. Ajastimen asettamisessa haluttiin huomioida palvelinten rasitukseen liittyvät kysymykset, ja ei haluttu ottaa riskiä palvelinten kuormittumisesta esimerkiksi sekunnin välein palvelimia kutsumalla. Kehitysryhmän kesken pohdittiin yhteisesti aikamääre, jonka voi vielä laskea olevan lähes välitöntä palautetta aiheuttava ilman, että se aiheuttaa liian isoa kuormitusta palvelimille, ja päädyttiin lopputulokseen, että palvelimia on hyvä kutsua neljä kertaa minuutissa.



Kuva 5. Projektin rakenne projektikansiosta tarkasteltuna.

Raportointinäkymän projektikansiota esittävää kuvaa 5 tutkimalla voi löytää backendin sisältävän tiedoston server.js ja frontendin sisältävän tiedoston App.js. Raportointinäkymän käynnistäminen lokaalisti kehittäjän tietokoneella tapahtuu navigoimalla kyseisen tietokoneen käyttöjärjestelmän terminaalissa raportointinäkymän projektikansioon, ja ajamalla siellä terminaalille komennon ”npm run dev”. Käynnistämisessä on hyödynnetty npm-pakettia ”concurrently”, joka antaa projektille mahdollisuuden ajaa yhdellä komennolla sekä frontendiä, että backendiä samanaikaisesti. Navigoimalla selaimelle osoitteeseen ”http://localhost:3000” pääsee kehittäjä tutkimaan raportointinäkymää, kun taas osoitteesta ”http://localhost:3001/X”, jossa X edustaa halutun palvelimen nimeä, pääsee kehittäjä näkemään backend-APIn, joka palauttaa palvelimilta haetun sisällön sijainnin.

Projektille luotiin OpenShift-kehitysputki, joka tarkoittaa sitä, että raportointinäkymä on kehittäjille saatavilla Kelan sisäverkossa myös tietystä osoitteesta. Koska vallitseva pandemia esti kehitystiimin toimistotyöskentelyn ja pakotti kaikki työntekijät varatoimisesti etä-

työskentelyyn ja näin ollen raportointinäkymää ei alkuperäisen suunnitelman mukaan voinut työpäivän aluksia vain avata työtiloihin yhteiselle näytölle, luotiin sen vuoksi kehitysputki, että kehittäjän, joka raportointinäkymää haluaa hyödyntää, ei joka kerta tarvitsisi käydä läpi tätä terminaalissa navigoinnin prosessia toimiessaan etätöissä. Projektin kehitysputki on toteutettu hyödyntäen Jenkins-automaatiotyökalua ja OpenShift-kehitysalustaa. Kehitysryhmän Kelan uusien verkkosivujen kehitysprojektissa on hyödynnetty jo valmiiksi paljon OpenShift ja Jenkins-kombinaatiolla varustettuja kehitysputkia, joten valmiin kehitysputken kopiointi ja raportointinäkymään implementointi niiden pohjista ei tuottanut vaikeuksia.

4.2 Ratkaisun konfigurointi muille ryhmille ja raportointinäkymän muokkaus

Raportointinäkymän muutosten tekeminen vaatii lähdekoodin muokkaamista ja sinne uusien palvelinten lisäämistä ja tilojen hakemisen konfiguroimista. Raportointinäkymän rakennuksen aikana oli jo selvillä lähes kaikki palvelimet, joista tiloja halutaan hakea, ja suunnitteilla ei ole ottaa laajamittaisesti uusia palvelimia kehityskäyttöön. Jos uusia palvelimia kuitenkin kehityksen tai tuotannon varmistamisen takia halutaan ottaa käyttöön, on kehitystiimin työtilaan dokumentoitu ohjeet kehittäjälle, joka muutoksia alkaa lähdekoodiin tekemään. Ohjeet on kirjoitettu niin, että taustatietoa palvelimista ei tarvitse omata, ja mahdollinen uusikin kehitysryhmän työntekijä pystyy palvelinten lisäämisen suorittamaan. On oletettavaa, että jokaiseen ympäristöön on tulossa korkeintaan kaksi palvelinta lisää Kelan uusien verkkosivujen kehitystyön aikana.

Tilanteessa, jossa muut Kelan sisäiset kehitystyöryhmät haluavat ottaa raportointinäkymän oman ryhmänsä käyttöön valvomaan ryhmän omia palvelimia, lisätään kehitysryhmien kehittäjiä Bitbucket-työtilaan ja annetaan heille oikeudet kopioida raportointinäkymän pohja. Tällöin työtilassa olevat dokumentoidut ohjeet auttavat kehittäjiä lisäämään haluttavat palvelimet kyseisen ryhmän omaan raportointinäkymään. Kehitysputken kopioinnista on koko Kelan IT-osastolla yhteinen dokumentaatio-sivu, jota voi ja kannattaa hyödyntää mahdollisen kehitysputken rakennuksessa kopioitaviin raportointinäkymiin.

5 Raportointinäkymän tarve

Toimeksiantajan kehitystiimin tarve palvelimet yhdistävälle raportointinäkymälle oli mittava, sillä vaikka hälytysmallisia ratkaisuita oli eri palvelimille olemassa, eivät ne vastanneet haluttuun tilaan. Rakentamalla kaikille palvelimille yhteinen raportointinäkymä saatiin aikaan myös yksi uusi tärkeä valvonnan aihe; palvelimien muodostamien *palveluiden* valvonta. Palvelulla tässä kontekstissa viitataan esimerkiksi kehitysympäristön palvelinten valvontaan. Yhdessä ympäristön palvelimet muodostavat yhden palvelun, eli tässä tapauksessa verkkosivun.

5.1 Hälytinmalliset ratkaisut Kelassa

Nykyisissä raportointinäkymäratkaisuissa on puutteita, jotka mielessä pitäen tätä kyseistä toteutusta lähdettiin suunnittelemaan. Päätarkoituksena tässä työssä oli täyttää aukkoja, joita jo olemassa olevat ratkaisut muodostavat. Olemassa olevia ratkaisuita voisi kuvailla enemmänkin hälytinmallisiksi ratkaisuiksi, kuin raportointinäkymiksi, koska niiden toimintatarkoitus on enemmänkin toimia hälyttiminä hätätilanteissa, kuin jokapäiväisessä työssä hyödynnettävinä työkaluina kehityksessä, jonka vuoksi tässä toteutuksessa keskityttiin juuri simppelin raportointinäkymän kehittämiseen. Näkymää suunnitellessa todettiin yhteisesti nykyratkaisuiden kahdeksi suurimmaksi haasteeksi nousevan sekä ajankäytön, kun kaikkien palvelinten omia näkymiä piti erikseen käydä tutkimassa, jos halusi kokonaiskuvaa kaikkien palvelinten tiloista, että palvelinten irtonaisen seurannan. Hälytinratkaisut eivät anna konkreettista kuvaa ympäristöjen verkkosivujen toiminnasta, vaan kehittäjä joutuu itse päättelemään yksittäisten palvelinten toimintaa tutkimalla ympäristöissä pyörivien palveluiden toiminnasta.

Nykyinen hälytinmallinen ratkaisu on toteutettu IBM Tivoli Monitoring-ohjelmistolla. IBM Tivoli Monitoring on yrityksille tarkoitettu spesialisoitava ohjelmisto, joka tarjoaa tavan tarkistaa infrastruktuurin suorituskykyyn liittyviä tekijöitä ja niiden historiatietoja (Gune 2.12.2014). Sen toimintamalliin kuuluu erityisesti valvoa portteja, joissa palvelimet pyörivät, ja lähettää valikoiduille kehittäjille sähköpostia, jos palvelin ei ole jostain syystä päällä. Tivoli Monitoring ei kuitenkaan kerro sähköpostissa palvelimen kaatumisen syytä, tai anna minkäänlaista yleiskuvaa siitä, kuinka monta palvelinta on tietyssä palvelussa toimintakunnossa. Vahvoja puolia siinä on mahdollisuus tarkistaa tiettyjen palvelinten historiatietoja sen suorituskyvystä, käytettävyydestä ja terveydestä. Nämä tekijät omalta osaltaan vaikuttivat raportointinäkymän ominaisuuksien priorisointiin niin, että raportointinäkymään ei koettu tarpeelliseksi lisätä palvelinten historiatietoja, koska ne ovat tilanteen vaatiessa helppo tarkistaa IBM Tivoli Monitoring-ohjelmistosta. IBM Tivoli Monitoring on yksi harvoja Kelassa käytettyjä Software as a Service-tyyppisiä palvelumalleja, ja tästä syystä Kelan

kehittäjät eivät pääse käsiksi ohjelmistoon tekemään mahdollisia haluttuja muutoksia, vaan niistä pitää keskustella ohjelmiston ylläpidon kanssa. Raportointinäkymän rakennuksessa yksi halutuimmista asioista oli kehittäjien mahdollisuus vaikuttaa nopeasti valvottaviin palvelimiin tai palvelinten valvontatapoihin.

5.2 Käyttötapaukset

Pääkäyttötapaus raportointinäkymälle oli ennen kirjoitushetkellä vallitsevaa pandemiaa se, että näkymän voisi toimistolle saapuessa avata jollekin toimistolla sijaitsevista näytöistä, jolloin se tukisi kehitystyötä ollessaan koko ajan auki tekemisen taustalla. Pandemian puhjettua ja kehittäjien siirryttyä etätöihin käyttötapauskin muuttui pilvipohjaista kehitysalustaa (OpenShift) hyödyntäen lokaalista sovelluksesta ulkoistetuksi sovellukseksi, joka toi oman työmäärän lisäyksensä projektiin.

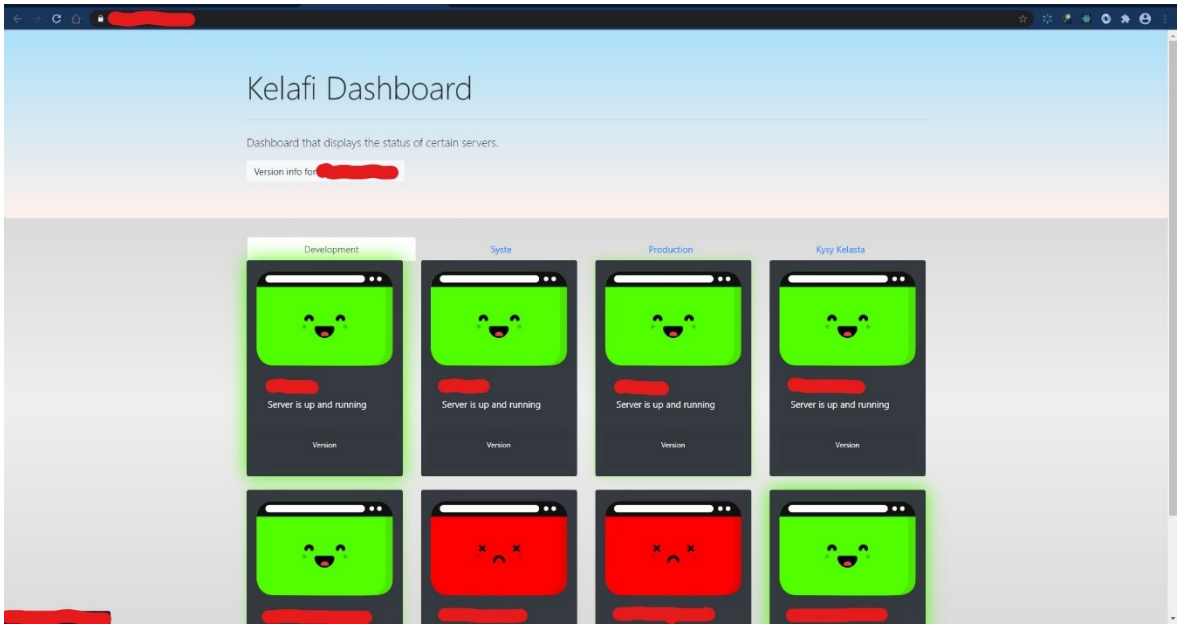
Nykyisessä tilassaan raportointinäkymää voi hyödyntää mahdollisien toimistopäivien aikana alkuperäisesti tarkoitetulla tavalla, mutta sen lisäksi se toimii kitkattomasti myös kehittäjän selainvälilehdellä aukinaisena palveluna, josta voi tarpeen tullen nopeasti tarkistaa kehitys-, testaus- ja tuotantopalvelinten tilan.

Isoin riski raportointinäkymän käyttötapauksissa on itse palvelimelta saatavan sisällön validoinnin vikatila. Raportointinäkymä on pyritty rakentamaan niin, että ainoastaan oikeanlainen ja toimiva kutsu yhdistettynä paikkansapitävään sisältöön aiheuttavat raportointinäkymän käyttöliittymässä palvelimen tilaksi oikein toimivan vihreän kuvakkeen. Koska backend kutsuu palvelinta ja määrittelee frontendille lähetettäväksi tilaksi joko positiivisen numeroarvon sisällön löytyessä, tai negatiivisen numeron käytännössä missä tahansa muussa tapauksessa, on mahdotonta, että käyttöliittymässä näkyisi joskus oikein toimiva palvelin sisällön ollessa puutteellista. Käytännössä kaikissa mahdollisissa virhetilanteissa, kuten kutsuessa virhetilassa olevaa palvelinta, sisällön puuttumisessa palvelimelta tai palvelinkutsun kestämisessä näytetään käyttöliittymässä palvelimen kohdalla punainen virheilmoitus ja linkki backendiin, joka kertoo sisällön puuttuvan tai mahdollisen 504-statuskoodin, joka indikoi proxy-palvelimen kutsun aikakatkaisua. Tapaus, jossa raportointinäkymän koodin logiikkaa olisi käyty muuttamassa, on käytännössä raportointinäkyvän toimivuuden kannalta ainoa riskialtis tila, ja tämän vuoksi koodia säilytetään kehitysryhmän Bitbucketissa, johon vain valitut kehittäjät pääsevät käsiksi. Koodimuutosten takia raportointinäkymän rikkoutumisenkin voi hätätilanteessa korjata vaihtamalla raportointinäkyvän versiota viimeisimpään toimivaan versioon.

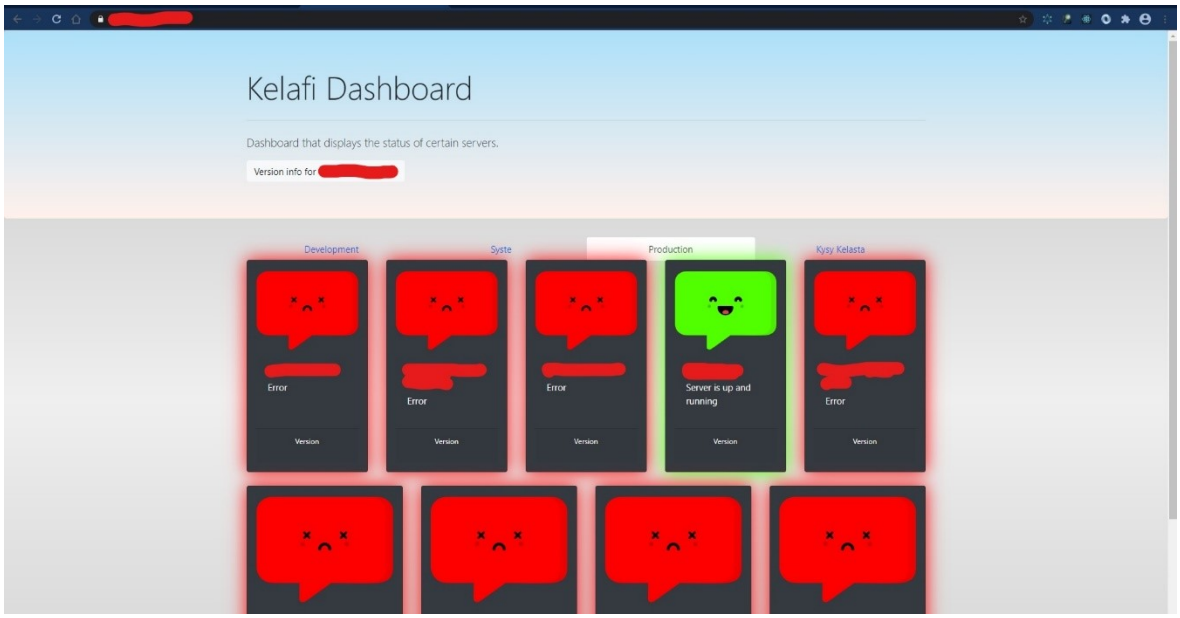
Raportointinäkymien yleisiä heikkouksia on sekä se, että jos raportointinäkymässä yritetään antaa liikaa informaatiota, saattaa relevanttia dataa hukkua sen käyttäjältä informaatiotulvan alle, tai se, että raportointinäkymät eivät yleensä selitä esittämäänsä dataa tarpeeksi hyvin (Storey & Treude 2019, 185-187.). Molemmat näistä ongelmista kuuluvat enemmänkin liiketoiminnassa hyödynnettyihin raportointinäkymiin, ja koska tämän projektin raportointinäkymän tarkoitus on esittää vain relevanttia dataa ja mahdollisimman selkeästi, ei näiden seikkojen voi olettaa olevan kovin potentiaalisia riskejä. Raportointinäkymän käyttämisen vaikeusastettakaan ei voi mieltää kovin isoksi riskiksi, sillä raportointinäkymää käyttää vain kehittäjät, jotka tietävät kuinka se toimii, ja jos tulee tilanne missä vaikka kehitysryhmään tulee uusia jäseniä, on raportointinäkymän käytöstä luotu laaja dokumentaatio-sivu kehitysryhmän työtilaan.

5.3 Haluttu tila

Tätä toimeksiantoa suunnitellessa päätettiin, että vanhojen monitorointiratkaisuiden annetaan jäädä hälytinmallisiksi toimintavälineiksi, sillä siihen ne parhaiten soveltuvat. Ne myös tukevat uuden yhtenäisen raportointinäkymän käyttöä tarjoten kriittisten tilanteiden varalta suoran kaistan virheilmoituksista kehittäjille.



Kuva 6. Raportointinäkymän ensimmäinen välilehti.



Kuva 7. Raportointinäkymän kolmas välilehti. Kuva otettu ennen virheilmoitusten toteuttamista.

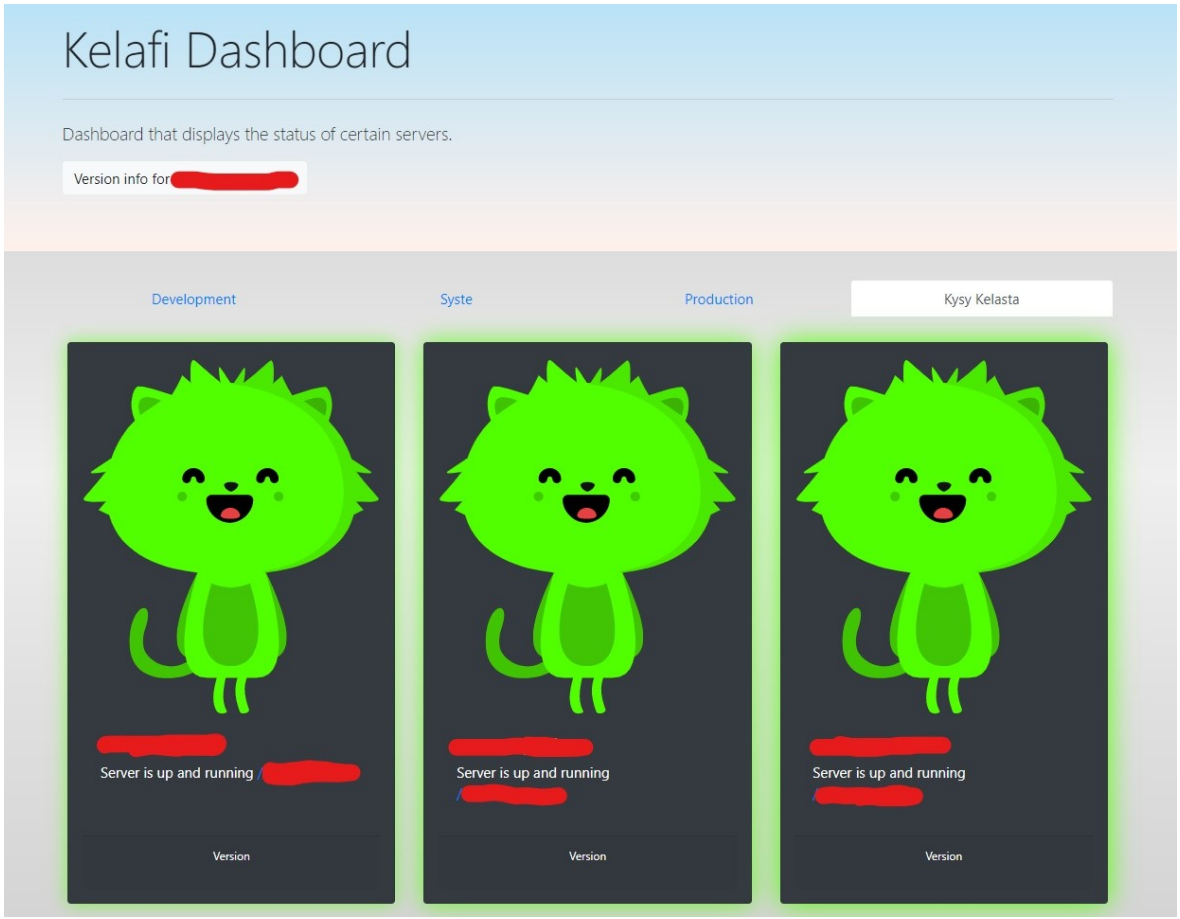
Yhtenäistä raportointiratkaisua suunnitellessa haluttiin, että samojen ympäristöjen palvelinten tilan voisi nopeasti tarkistaa samasta paikasta, jolloin päädyttiin välilehtimallin hyödyntämiseen. Kuvia 6 ja 7 tutkimalla voi huomata palvelinten nimien, tilojen ja määrien vaihtuvan ruudun keskellä. Molemmissa kuvissa on tietoturvasyistä punaisella värillä peitetty sekä osoite, että palvelinten nimet. Yksi välilehti edustaa käytännössä yhtä palvelimista muodostuvaa palvelua.

5.4 Raportointinäkymän Design-ratkaisut

Designin taustalla pääprioriteettina oli raportointinäkymän selkeys. Tavoitteena oli, että jo nopealla kunkin navigaatiovälilehden vilkaisulla saisi selville palvelinten nykyisen tilan. Raportointinäkymä tyyliteltiin aluksi tummanpuhuvalla sävyllä, mutta sittemmin värimaailmaksi valittiin vihreä väri edustamaan toimintakunnossa olevia palvelimia ja punainen väri edustamaan jonkinlaisen virhetilan omaavia palvelimia. Taustan värimaailma myös vaihdettiin tummansinisestä vaaleaksi korostamaan palvelinkorttien näkyvyyttä raportointinäkymässä.

Kehitysryhmän yhteinen päätös oli, että raportointinäkymän käyttö tulee olemaan vain kehityksen aikaista työkalun hyödyntämistä eikä ulkoiselle käyttäjälle näkyvää, joten hieman kevytmielisempi toteutus kuvakkeissa oli paikallaan. Palvelinten kuvaukseksi valikoitui työryhmän suosikiksi muodostunut ”react-kawaii”-niminen npm-kirjasto, josta löytyy muutamia erilaisia kuvakevaihtoehtoja, joita itse näkymässä hyödynnetään. Kuvakkeita kirjastossa on yhteensä 11, mutta niistä vain neljää erilaista käytetään raportointinäkymässä.

Kuvia 6 ja 7 tutkimalla voi huomata, kuinka kuvakkeita raportointinäkymässä käytetään. Kaikkiin kuvakkeisiin on mahdollista valita joko iloiset tai surulliset kasvot, ja tätä hyödynnetään helpottamaan palvelinten tilan esittämistä. Kuvakkeessa näkyvät iloiset kasvot, kun palvelimella on haluttu tila, kun taas surulliset kasvot näkyvät palvelimen ollessa virhetilassa. Kasvojen vaihtuvuus yhdistettynä värimaailmaan luo tarkoituksellisesti positiivisen tai negatiivisen tunnelman jo nopealla palvelinten tilan vilkaisulla, joka edistää raportointinäkymän tarkoituksenmukaisuutta toimia vauhdikkaasti kehityksen tukena.

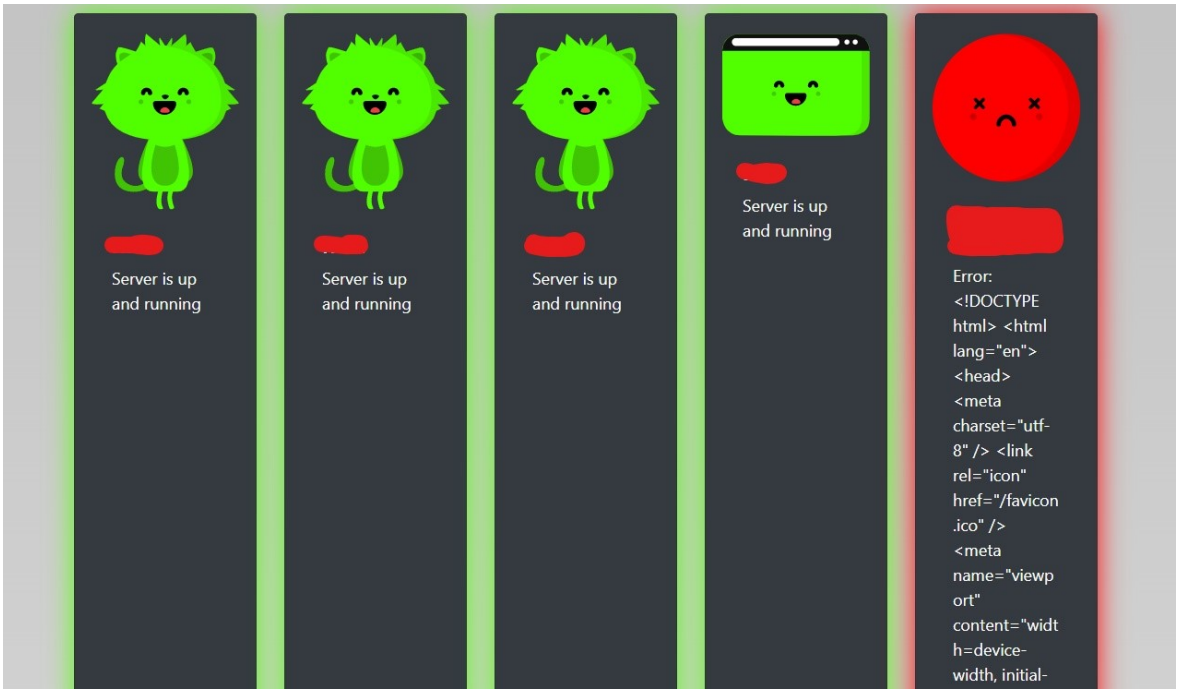


Kuva 8. Raportointinäkymän neljäs välilehti.

Raportointinäkymän palvelinten esittämiskohdat skaalautuvat sen mukaan, kuinka monta palvelinta kullakin välilehdellä on. Vertaamalla kuvaa 8 kuviin 6 ja 7, voi huomata kuinka palvelinkortit kasvavat, jos palvelimia on vähän ja kutistuvat, jos palvelimia on paljon. Myös kuvakkeet palvelinkorteissa vaihtuvat ympäristöjen mukaan. Kaikilla ympäristöjen välilehdillä käytetään eri kuvakkeita, jotta välilehdet olisivat nopeasti erotettavissa toisistaan.

5.5 Vaatimusmääritykset ja käyttökohderyhmät

Raportointinäkymän toiminnallisuudesta haluttiin, että palvelinten näkymä luokiteltaisiin käyttöympäristöjen perusteella. Eri ympäristöihin kuuluu kehitys-, testaus-, ja tuotantoympäristöt. Ympäristöjen luokittelu helpottaa ongelmien havainnollistamista ympäristökohtaisiksi ja näin ollen antaa hyvän kuvan siitä, miten vakavasta mahdollisesta ongelmasta palvelussa sijaitsevilla palvelimilla on kyse.



Kuva 9. Raportointinäkymän ensimmäinen virheilmoituksen esitystapa.

Taustalla tapahtuvista palvelinkutsuista määrityksenä oli tietyn avainsanan sijainnin haku palvelimelta, jotta voidaan tarkistaa sekä palvelimen tila, että sinne sijoitetun sisällön paikkansapitävyys. Joissakin artikkeleita säilövissä palvelimissa avainsanan sijaan haetaan dokumenttimäärä, jonka palvelin palauttaa kutsussaan. Kuvassa 9 demonstroituna nähdään, että jos nämä kriteerit täyttyvät, esitetään palvelinkohtaisesti sille varattuun kohtaan näkymässä sekä toimivaa palvelinta kuvaava teksti ”Server is up and running”, sekä selkeä vihreä värimaailma avustamaan kehittäjää tarkistamaan nopeasti palvelimen tilan.

Toimeksiannon alussa suunniteltiin, että mahdollisen virheilmoituksen voisi tulostaa suoraan sivulla näkyvän palvelimen tilalle varattuun laatikkoon, mutta raportointinäkymän edetessä paremmaksi ratkaisuksi osoittautui suora linkki backendin luomaan virheilmoitukseen sen ollessa tyylillisesti huomattavasti helppolukuisempaa. Kuvan 9 tapaan virheilmoituksen sattuessa ja koko virheen tulostuessa palvelimelle varattuun esitystilaan, laajeni sen pituus epäkäytännöllisen pitkäksi ja vaikealukuiseksi. Kuten esimerkiksi kuvista 6,

7 ja 9 voi huomata, virheellistä palvelimen tilaa kuvastaa myös kirkkaanpunainen värimaailma, joka on helppo erottaa neutraalista taustasta ja mahdollisista toimivista vihreistä palvelimista.

Pääkäyttökohderyhmänä raportointinäkymä on käytössä ryhmällä, jonka palvelinten ympärille työ rakennettiin. Raportointinäkymä kuitenkin on rakennettu niin, että tarvittaessa myös muut Kelan sisäiset ryhmät voivat ottaa sen käyttöön kopioimalla raportointinäkymän lähdekoodin Bitbucketista ja lisäämällä omat tutkittavat palvelimet nykyisten tilalle. On myös mahdollista muokata palvelinten hakuun liittyvää logiikkaa tarvittaessa ja siitä on kirjoitettu ohjeet raportointinäkymän työtilan dokumentaatioon, mutta lopullinen logiikan muokkaaminen jää sen ryhmän jäsenten vastuulle, jotka haluavat ottaa raportointinäkymän käyttöön oman ryhmänsä sisällä.

6 Tulokset ja pohdinta

Tavoitteena opinnäytetyössä oli tuottaa palvelinten tilan raportointisovellus, eli raportointinäkymä, jota olisi mahdollista hyödyntää verkkosivujen kehityksen tukena. Raportointinäkymän tarkoitus oli tuottaa visuaalisesti helppolukuista informaatiota sen käyttäjille palvelinten nykytilasta, sekä tarjota tietoa palvelimilla esiintyvistä mahdollisista ongelmatilanteista. Opinnäytetyö toteutettiin toimeksiantona, mutta toimeksiantaja ei antanut tiukkoja teknisiä kriteereitä raportointinäkymän kehitykselle. Tiivistettynä yhteen lauseeseen raportointinäkymän tavoitteen voisi kuvailla näin: haluttiin rakentaa sovellus, jonka tavoitteena on olla ensimmäinen paikka, johon verkkosivun kehittäjät katsoisivat kohdatessaan palvelinkohtaisen ongelman, ja jota hyödyntäessään kyseistä ongelmaa ei tarvitsisi palvelimella asti koskaan kohdata, koska hyödynnettäessä kehitystyössä raportointinäkymä olisi informoinut jo ongelmasta ja antanut suuntaa ongelman laadusta.

Raportointinäkymän alustava suunnittelu sekä kehitys aloitettiin vuoden 2020 helmikuussa, ja se otettiin verkkosivujen kehityksen tueksi käyttöön vuoden 2020 lokakuussa. Raportointinäkymää kehitettiin osa-aikaisesti kevään ja kesän aikana noin kahden työpäivän verran viikossa, ja syksyllä noin viiden työpäivän verran viikossa. Raportointinäkymän on rakentanut kokonaisuudessaan yksi henkilö, eli tämän opinnäytetyön kirjoittaja. Raportointinäkymän vaatimuksia on määritellyt ja ulkoasua ideoinut kaksi toimeksiantajalla Kelan uusien verkkosivujen työryhmässä työskentelevää kehittäjää.

Valmistunut tuote on JavaScript-ohjelmointikielellä rakennettu pilvipalvelualusta OpenShiftissä pyörivä verkkosivu, johon pääsee käsiksi vain toimeksiantajan sisäverkosta, jolla varmistetaan sen olevan suljettu ulkopuolisilta käyttäjiltä. Navigoimalla verkkosivun osoitteeseen sen käyttäjä näkee listan Kelan uusia verkkosivuja rakentavan kehitysryhmän palvelimista, sekä tiedon niiden nykyisistä tiloista. Mahdollisia tiloja on kaksi; palvelin toimii halutulla tavalla, tai palvelimella on havaittu käyttöä estävä virhe. Sivulta löytyvät neljä eri välilehteä sisältävät jokainen eri määrän eri ympäristöjen palvelimia, ja ympäristöt on lajiteltu niiden käyttötarkoitusten mukaan. ”Development”-välilehdeltä löytyy kehityksessä käytettävän verkkosivun palvelimet, ”Syste”-välilehdeltä puolestaan testauksessa käytettävän verkkosivun palvelimet. ”Production”-välilehdeltä löytyy aktiivisten Kelan verkkosivujen palvelimet. Neljänneltä välilehdeltä löytyy yhden verkkosivun kaikkien ympäristöjen palvelimet, yksi kustakin ympäristöstä (kehitys, testaus, tuotanto). Jokaista palvelinta edustaa yksi kortti, josta löytyy palvelinkohtaisten tietojen lisäksi kuvake, joka muuttuu palvelimen tilan mukaan. Palvelinkorttien alareunassa lukee palvelimella pyörivästä sisällöstä riippuen joko sisällön versionumero, tai tietyissä tapauksissa sisällön indeksointiaika. Valmistunut tuote vastaa toimeksiantajan vaatimuksia raportointinäkymästä.

Raportointinäkymä on ollut Tulokset-luvun kirjoituksen aikaan käytössä neljällä kehittäjällä noin kaksi viikkoa. Paremman kuvan raportointinäkymän hyödyllisyydestä saa varmasti vasta useamman kuukauden käytön ja useamman ongelmatilanteen jälkeen, mutta jo nyt näkymää on käytetty hyödyksi kehityksen tukena. Esimerkkitilanne: yhdeltä rakennettavan verkkosivun palvelimelta puuttui työpäivän alkaessa indeksoinnissa tapahtuneen virheen vuoksi sisältöä. Nopealla raportointinäkymän vilkaisulla kehittäjä paikallisti palvelimen, jossa virhe oli tapahtunut, navigoi virhesivulle, ja huomasi sisällön puuttuvan. Ongelmatilanne ratkesi indeksoimalla palvelimen sisällöt uudelleen, ja näin ollen kehittäjä sai virhetilanteen korjattua noin tunnissa. Ilman raportointinäkymän hyödyntämistä kehittäjällä olisi voinut kulua päiviäkin aikaa huomata sisältövirhe, riippuen sisällön kehityskohteesta.

Kehittäjiltä käytön perusteella saadun palautteen mukaan kaikista onnistuneimpia osia raportointinäkymässä on sen selkeys ja käytettävyys. Palvelinkorteista näkee nopeasti palvelinten tilan, ja välilehtinäkymä-ratkaisun vuoksi kehittäjä voi tehdä jo välilehtiä silmäilemällä nopean arvion palvelinten muodostamien palveluiden tilasta. Virhetilanteen sattuessa kehittäjä saa tarpeellista informaatiota virhetilanteeseen liittyen. Palvelinkorteissa olevat versiotiedot ovat nopeuttaneet myös kehitystä, sillä nyt ei tarvitse yksitellen navigoida palvelinten ominaisuuksiin ja etsiä sisällön versiotietoja sieltä, koska ne ovat palvelinkohtaisesti tulostettuina raportointinäkymään.

Kehittäjien palautteen mukaan raportointinäkymässä epäonnistuneita asioita on muuan muassa opinnäytetyössä tietoturvallisista syistä piilotettuna olevat palvelinten nimet ja nimeämisratkaisut. Jopa niiden kehittäjien, jotka muistavat palvelinten osoitteet, on hankala päätellä mitkä palvelinkortit kuuluvat millekin palvelimelle. Toki jo viikon käytön jälkeen nimet alkavat tulla tutuiksi, ja niitä varten on dokumentaatiosivusto, josta kaikki nimet ja niille kuuluvat palvelimet voi tarkistaa, mutta siitä huolimatta raportointinäkymässä valvottavien palvelinten nimet tullaan muuttamaan selkeämmiksi. Nimeämiskäytäntöön on tähän mennessä kuulunut nimetä palvelinkortit palvelimen osoitteen muotoisiksi. Esimerkiksi jos palvelimen osoite olisi ”https://kehitysympäristö-verkkosivu-x.com”, niin palvelinkortissa kyseinen palvelin olisi nykynimeämiskäytännöillä nimetty ”keh-verk-x”:ksi. Palvelinkortteja tullaan muokkaamaan vähintäänkin niin, että nimi sisältää kokonaisia sanoja eikä lyhenteitä.

Toiseksi epäonnistumiseksi raportointinäkymä-projektissa voi nostaa aikataulun venähtämisen. Helmikuussa aloitettu projekti olisi mieluusti otettu kehityksen tueksi jo ennen kesää, mutta muiden kehitystöiden korkeampi priorisointi laski raportointinäkymään jäävää kehityspanosta kevään ja kesän osalta pienemmäksi. Tuloksienkin kannalta kehityksen

painottaminen kevääseen olisi ollut hedelmällisempää, kun testausdataa raportointinäkymästä olisi saatu laajemmalta aikaväliltä. Virheilmoituksien suunnittelun voisi myös nostaa epäonnistumiseksi, sillä alun perin suunniteltua virheilmoitusten näyttämistä itse palvelinkorteissa ei voitukaan toteuttaa, mutta lopulta nyt toteutettu ratkaisu eli virheilmoitus-sivujen linkkien lisääminen kortteihin osoittautuikin kehittäjien mielestä paremmaksi ratkaisuksi sekä selvyyden, että designin puolesta, joten ehkä projektissa osoitetun joustavuuden ratkaisuiden suhteen voi nostaa onnistumiseksi.

Raportointinäkymälle ei ollut tarvetta määritellä erityisiä tietoturvavaatimuksia, sillä siihen pääsee käsiksi vain yrityksen sisäverkosta, ja sisäverkkoon voi muodostaa yhteyden vain kehittäjätunnuksilla. Raportointinäkymän koodia säilytetään kehitysryhmän Bitbucketissa, ja siihen pääsee käsiksi vain kehitysryhmän työntekijät. Opinnäytetyön kannalta tietoturvavaatimuksiksi voi laskea palvelinten osoitteiden peittämisen, ja sitä on noudatettu tarkasti. Ympäristöjen nimet saavat opinnäytetyössä näkyä. Ainoat suorituskykyvaatimukset raportointinäkymässä olivat se, että se ei tuki palvelimia pyyntömäärillä, ja se, että avaamalla raportointinäkymän selaimessa pyörähtää palvelinten tilojen haku ensimmäisen viidentoista sekunnin aikana läpi. Molemmat suorituskykyvaatimukset saavutettiin rajoittamalla palvelinkutsujen määrää neljään kertaan minuutissa viidentoista sekunnin välein.

Jos raportointinäkymän rakennus aloitettaisiin nyt, niin tekniset ratkaisut olisivat pitkälti hyvin samanlaiset. JavaScript toimii hyvin raportointinäkymän kaltaisissa pienemmän luokan projekteissa, ja missään vaiheessa raportointinäkymän kehitystä ei noussut esiin komplikaatioita, joita jokin toinen ohjelmointikieli olisi tehnyt paremmin. Myös toteutusjärjestys oli hyvin suunniteltu; ensin rakennetiin backend ja sinne muutama palvelinkutsu, sitten frontend ja palvelinkutsujen validointi ja näyttäminen, jonka jälkeen rakennettiin backendiin muut palvelinkutsut ja viimeiseksi lisättiin palvelimiin versiotiedot ja indeksointiajat. Isoin asia, mitä tekisin toisin olisi ehkä raportointinäkymän kehityksen tärkeyden priorisointi keväälle, joka tosin on toki lopulta täysin tuoteomistajan ja toimeksiantajan päätettävissä.

Jatkokehitysideoita raportointinäkymään on jo suunnitteilla muutama. Yhtenä isona asiana kehittäjiltä saadussa palautteessa oli ilmoitukset. Toivottiin, että tulevaisuudessa voitaisiin implementoida systeemi, jossa raportointinäkymä antaisi ilmoituksen selaimessa, jos palvelinten tiloissa tapahtuu muutoksia. Ilmoituksilla tässä tapauksessa viitataan siihen, että esimerkiksi jos kehittäjällä on raportointinäkymä auki selaimessa epäaktiivisella välilehdellä, voisi raportointinäkymän välilehti ilmoittaa visuaalisesti tilan muutoksesta kehittäjälle, vaikkapa Thoriq Firdauksen (How To Display Notifications in the Browser Tab 2015) artikkelin mukaisesti. Myös potentiaalinen virheiden historiatieto nousi raportointinäkymän kehityksessä esiin. Kehittäjät saisivat hyödyllistä dataa virheiden yleisyydestä ja niiden

syistä, jolla pystyttäisiin mahdollisesti tunnistamaan palvelinta hajottavia asioita ja eliminoimaan niitä. Virheiden historiatiedon säilyttämiseen tosin ei olla keksitty vielä mitään konkreettista vaihtoehtoa. Yksi jo suunnitteluvaiheessa oleva jatkokehitysidea on palvelinten suodatus. Yhden raportointinäkymää jo kehityksessä hyödyntäneen kehittäjän ajatus oli, että raportointinäkymään voisi lisätä suodatus-kentän, jossa palvelimen nimen hakusanalla voisi hakea kaikilta välilehdiltä vain tietyn avainsanan sisältäviä palvelinten nimiä. Näin saataisiin mahdollisesti vaikkapa kaikkien eri ympäristöjen verkkosivujen hakupalvelimet esiteltyä vierekkäin, jos sellainen tarve kehityksessä syntyy. Suodatuksen rakentamiseenkin on jo löytynyt lupaavan näköinen artikkeli: Quick and Simple Search Filter Using Vanilla JavaScript (Justin Rexroad 2019).

Viimeiseksi asiaksi opinnäytetyössä voisi todeta, että johdannossakin sivuttu kehittäjien toive siitä, että raportointinäkymä olisi ensimmäinen paikka, jota tutkittaisiin, kun kehityksen aikana tapahtuu odottamattomia virheitä näyttää jo nyt toteutuneen perustuen kehittäjiltä saatuun palautteeseen raportointinäkymän käyttökelpoisuudesta, vaikka raportointinäkymä ei olekaan ollut pitkään kehityskäytössä. Kaikkiaan raportointinäkymän rakennusprojektin voi siis laskea onnistuneeksi ja tuotteen käyttökelpoiseksi, vaikka aikataulu kehityksessä venyikin pitemmäksi, kuin oli tarkoitus. Aikataulun venymisen syyn voidaan kuitenkin perustella olevan asiallinen, kun kyse oli muun tärkeämmän kehityksen priorisoinnista sen edelle koko kehitysryhmän toimesta.

Lähteet

Atlassian 2019. Statuspage. Luettavissa: https://www.atlassian.com/software/statuspage?tab=private. Luettu 29.9.2020

Bitbucket 2020. What is Bitbucket? Luettavissa: https://bitbucket.org/ Luettu 7.10.2020

Codeacademy 2020. What is CORS? Luettavissa: https://www.codecademy.com/articles/what-is-cors. Luettu 11.11.2020

Computerhope 8.2.2020. Server. Luettavissa: https://www.computerhope.com/jargon/s/server.htm. Luettu 27.9.2020

Creative Tim 2020. Light Bootstrap Dashboard. Luettavissa: https://www.creative-tim.com/product/light-bootstrap-dashboard. Luettu 17.4.2020

Firdaus, T. 18.3.2015. How to Display Update Notifications in the Browser Tab. Luettavissa: https://webdesign.tutsplus.com/tutorials/how-to-display-update-notifications-in-the-browser-tab--cms-23458. Luettu 20.9.2020

Gune, A. 2.12.2014. IBM Tivoli software for enterprise system management. Luettavissa: https://searchitoperations.techtarget.com/feature/IBM-Tivoli-software-for-enterprise-system-management. Luettu 21.10.2020

Heller, M. 9.3.2020. What is Jenkins? The CI server explained. Luettavissa: https://www.infoworld.com/article/3239666/what-is-jenkins-the-ci-server-explained.html. Luettu 20.3.2020

Hälikkä, S. 1.11.2017. OpenShift haltuun – vinkkejä sovelluskehitykseen. Luettavissa: https://www.ambientia.fi/blogi/openshift-haltuun-vinkkeja-sovelluskehitykseen/. Luettu 20.3.2020

iDashboards 2020. What is a Dashboard? Luettavissa: https://www.idashboards.com/guides/what-is-a-dashboard/ Luettu 27.10.2020

James, S. 9.5.2016. Effective Development Environments – Development, Test, Staging/Pre-production and Production Environments. Luettavissa: https://medium.com/venture-garden-group-technology-blog/effective-development-environments-development-

test-staging-pre-production-and-production-environmen-a3a85cd349b2. Luettu 18.10.2020

Kielitohtori 2020. Dashboard-termin suomennos. Luettavissa: http://www.kielitohtori.fi/suomen-kielenhuollon-kysymys/mit%C3%A4-dashboard-suomeksi. Luettu 29.9.2020

Klipfolio 2020. Sales Dashboards. Luettavissa: https://www.klipfolio.com/resources/dashboard-examples/sales. Luettu 29.9.2020

Mozilla Developer 2020. What is JavaScript. Luettavissa: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript. Luettu 5.11.2020

Mulesoft 2020. What is an API? (Application Programming Interface). Luettavissa: https://www.mulesoft.com/resources/api/what-is-an-api. Luettu 5.11.2020

NodeJS 2020. What is npm? Luettavissa: https://nodejs.org/en/knowledge/getting-started/npm/what-is-npm/. Luettu 5.11.2020

Petters, J. 10.6.2020. What is a Proxy Server and How Does it Work? Luettavissa: https://www.varonis.com/blog/what-is-a-proxy-server/. Luettu 5.11.2020

Pulkkinen, V. 14.12.2017. Mitä tarkoittaa SaaS? Luettavissa: https://www.inderes.fi/fi/mita-tarkoittaa-saas. Luettu 19.10.2020

ReactJS 2020. A JavaScript library for building user interfaces. Luettavissa: https://reactjs.org/. Luettu 21.10.2020

React Kawaii 2020. Components. Luettavissa: https://react-kawaii.now.sh/#/Components. Luettu 20.4.2020

Rexroad, J. 30.4.2019. Quick and Simple Search Filter Using Vanilla JavaScript. Luettavissa: https://scotch.io/tutorials/quick-and-simple-search-filter-using-vanilla-javascript. Luettu 8.11.2020

Salesforce 2020. The Perfect Sales Dashboard Should Have These 12 Sales Metrics. Luettavissa: https://www.salesforce.com/products/sales-cloud/resources/sales-dashboard-tips/. Luettu 5.10.2020

Schwaber, K. & Sutherland, J. 2017. Scrum-opas. Luettavissa: https://scrum-guides.org/docs/scrumguide/v2017/2017-Scrum-Guide-Finnish.pdf. Luettu 26.10.2020

Storey MA., Treude C. 2019. Software Engineering Dashboards: Types, Risks, and Future. Luettavissa: https://link.springer.com/chapter/10.1007/978-1-4842-4221-6_16. Luettu 7.10.2020

Suomisanakirja 2020. Raportti-sanan määritelmä. Luettavissa: https://www.suomisana-kirja.fi/raportti. Luettu 25.9.2020

Themeforest 2020. Wieldy – React Admin Template Ant Design and Redux. Luettavissa: https://themeforest.net/item/wieldy-react-redux-ant-design-admin-template/22719616. Luettu 27.3.2020

W3Schools 2020. Node.js Introduction. Luettavissa: https://www.w3schools.com/nodejs/nodejs_intro.asp. Luettu 5.11.2020

W3Schools 2020. What is JSON? Luettavissa: https://www.w3schools.com/whatis/whatis_json.asp. Luettu 5.11.2020

W3Schools 2020. What is HTTP? Luettavissa: https://www.w3schools.com/whatis/whatis_http.asp. Luettu 5.11.2020

Webopedia 2020. On-premises. Luettavissa: https://www.webopedia.com/TERM/O/on-premises.html. Luettu 19.10.2020