Web-sovelluksen kehittäminen ja julkaisu

Juuso Kokkonen, Mikko Nurmi

Haaga-Helia ammattikorkeakoulu
Tradenomi
Opinnäytetyö
2023

Tekijä(t)
Juuso Kokkonen, Mikko Nurmi

Tutkinto
Tradenomi

Raportin/Opinnäytetyön nimi
Web-sovelluksen kehittäminen ja julkaisu

Sivu- ja liitesivumäärä
29 + 3

Tämä on toiminnallinen opinnäytetyö, jonka tarkoituksena on käydä läpi web-sovelluksen kehittämisprosessi ja julkaisu alusta loppuun. Opinnäytetyön toiminnallisessa osassa kehitetään web-palvelu, joka käsittelee Riot Gamesin julkaiseman League of Legends -videopelin otteludataa ja esittää sitä palvelun käyttäjille. Otteludataa sovellus tulee saamaan Riot Gamesin tarjoamasta ohjelmointirajapinnasta. Web-sovelluksemme koostuu selainpuolesta, palvelinpuolesta sekä tietokannasta. Palvelun selainpuoli, eli front-end, toteutetaan käyttämällä React.js -JavaScript-kirjastoa. Palvelun palvelinpuoli, joka on vastuussa tiedon siirtämisestä tietokannan, palvelun selainpuolen ja Riot Gamesin ohjelmointirajapinnan välillä, rakennetaan Java-ohjelmointikielellä käyttäen Spring Boot -viitekehystä. Lisäksi palvelu tarvitsee tiedon säilyttämiseen relaatiotietokannan, joka toteutetaan käyttäen SQL-pohjaista PostreSQL-tietokannanhallintajärjestelmää.

Opinnäytetyön kirjallisessa osuudessa kuvataan web-sovelluksen kehittämisprosessia ja sen vaiheita. Kirjallinen osa kuvaa web-sovelluksen kehittämistä yleisellä tasolla ja sen rinnalla kulkee opinnäytetyön toiminnallisen osan kuvaus, jossa muun muassa perustellaan valintoja ja päätöksiä, joita olemme tehneet toiminnallisen osan kehityksen aikana.

Työn toiminnallisen osan sovellus julkaistaan projektin päätteeksi internettiin palvelun käyttäjien saataville. Palvelun eri osat tulevat pyörimään Linux-palvelimella ja palvelulle laaditaan oma verkkotunnus, jolla sen käyttäjät pääsevät vaivattomasti käyttämään palvelua omalta selaimeltaan.

Asiasanat
Web-sovellus, Ohjelmistokehitys, Java, React, Linux-palvelin

Sisällys

1 Johdanto 1
1.1 Sovelluksen määrittely 2
1.2 Keskeiset käsitteet 2
2 Sovelluksen suunnittelu 6
2.1 ARAM Challengen suunnittelu 6
2.2 Teknologioiden valinta 7
3 Sovelluksen kehitys 9
3.1 Sprint 1 9
3.1.1 Sovelluksen osien luonti 10
3.1.2 Kehityspalvelin 10
3.2 Sprint 2 12
3.2.1 Back-end kehitys 14
3.2.2 Front-end kehitys 17
3.3 Sprint 3 18
3.3.1 Julkaisun suunnittelu 18
3.3.2 Tuotantopalvelimen asennus 19
4 Web-sovelluksen julkaisu yleisesti 20
4.1 Verkkoisännöinti 20
4.2 Verkkotunnus 21
5 Pohdinta 23
5.1 Oppimisen pohdintaa 24
5.2 Jatkokehitys 24
Lähteet 26
Liitteet 27

1 Johdanto

Sovelluskehitys on muuttunut viime vuosina huimasti. Noin kymmenen vuotta sitten koko sovellusalalla alkoi tapahtua käänne käyttäjän laitteella ajettavista sovelluksista kohti suoraan verkkopalvelimilla toimivia ohjelmia. Nykyisin lähes jokaisesta käyttäjän laitteella ajettavasta sovelluksesta löytyy verkkopohjainen versio, oli kyse sitten videoiden tai kuvien editoinnista, 3D-mallien luomisesta, keskustelusta chat-huoneissa tai oman suosikkipelisi ottelutulosten tarkastelusta kolmannen osapuolen kehittämällä haastetaululla.

Tämän opinnäytetyön tavoitteena on kuvata web-sovelluksen kehitysprosessi sen alkumetreiltä aina valmiin palvelun julkaisemiseen asti. Projektin tuloksena syntyy web-sovellus, joka tullaan julkaisemaan internettiin käyttäjien saataville, sekä kirjallinen raportti, jossa kuvataan sovelluksen kehitysvaiheita. Toiminnallisen osan sovelluksen tarkoituksena on automatisoida käyttäjien pelaamien suositun videopelin League of Legendsin otteluiden seuranta ja tarjota palvelu, jota pelin julkaisija Riot Games ei käyttäjilleen tarjoa. Sovelluksemme on rajattu pelin sisäisen ARAM pelimuodon otteluiden seurantaan. ARAM eroaa pelimuotona pelin yleisimmästä pelimuodosta muun muassa siten, että ennen ARAM ottelun alkua jokaiselle pelaajalle arvotaan satunnaisesti yksi pelihahmo pelin päälle 160 hahmosta pelattavaksi. Arvottuja pelihahmoja voi kuitenkin vaihtaa viiden hengen joukkueen pelaajien välillä ja tietyin edellytyksin hahmon voi arpoa itselleen uudestaan, tosin maksimissaan kaksi kertaa per ottelu per pelaaja. Lisäksi kyseisen pelimuodon pelialue tai “kartta” on normaalia pienempi ja yksinkertaisempi, sekä otteluiden kesto on yleensä lyhyempi kuin perinteisen pelimuodon. ARAM otteluissa kaksi viidestä pelaajasta koostuvaa joukkuetta yrittää tuhota vastustavan joukkueen tukikohdan. Ottelun voittaa joukkue, joka tuhoaa vastapuolen tukikohdan ensimmäisenä. Sovelluksemme käy läpi käyttäjien pelattuja ARAM otteluita ja tallentaa tiedot siitä, millä hahmoilla pelaaja on otteluita voittanut tai hävinnyt.

Opinnäytetyön raportissa kuvataan web-sovelluskehityksen vaiheita ja käytäntöjä yleisellä tasolla, sekä siinä ohessa kehittämämme sovelluksen kehittämisprosessia. Raportissa käydään lyhyesti myös läpi päätöksiä, joita teimme sovelluksen kehitykseen liittyen, kuten käytettävien teknologioiden valintaa ja kehitysmenetelmiä. Toivomme, että tämä raportti voisi auttaa myös muita web-sovelluskehittäjiä kehittämään omia sovelluksiaan muun muassa kuvaamalla kehityksen vaiheita ja sovelluksen julkaisua. Raportin tarkoituksena ei kuitenkaan ole toimia yksityiskohtaisina ohjeina aloitteleville web-sovelluskehittäjille.

1.1 Sovelluksen määrittely

Sovelluksen perustana on leikkimielinen haaste, jossa pelaajan tavoitteena on voittaa vähintään yksi ottelu jokaisella pelin pelihahmolla. Tämän vuoksi sovellus sai nimen “ARAM Challenge” (ARAM-haaste). ARAM Challenge toimii käytännössä erillisenä “pelinä” League of Legends pelin rinnalla. Käyttäjä näkee ARAM Challengen avulla millä hahmoilla hän ei vielä ole voittanut League of Legendsin ARAM ottelua ja yrittää voittaa kyseisillä hahmoilla. Haaste katsotaan suoritetuksi, kun käyttäjä on voittanut jokaisella hahmolla vähintään yhden ottelun.

Sovellus tulee myös tarjoamaan tilastodataa niistä pelaajista, jotka ovat haasteen suorittaneet. Pelaajat voivat kilpailla keskenään siitä, kuka on suorittanut haasteen vähimmällä määrällä pelattuja otteluita tai nopeimmin haasteen aloitusajankohdasta laskettuna. Sovellus palvelee myös käyttäjiä, jotka eivät välttämättä halua kilpailla muiden pelaajien kanssa, vaan haluavat helpon ja nopean tavan seurata omia otteluitaan.

1.2 Keskeiset käsitteet

ARAM — Yksi League of Legends pelin pelimuoto. Tämän opinnäytetyön toiminnallinen osa on rajattu käsittelemään vain ARAM pelimuodon otteluita. Lyhenne “ARAM” tulee sanoista All Random All Mid, jotka viittaavat pelimuodon normaalista poikkeaviin ominaisuuksiin, eli tavallista pienempään pelialueeseen ja pelaajille satunnaisesti arvottaviin pelihahmoihin.

ARAM Challenge — Tämän opinnäytetyön toiminnallisen osan tuotoksena valmistuvan sovelluksen nimi. Nimi tulee pelimuodosta, johon sovellus on rajattu, sekä leikkimielisestä “haasteesta” voittaa vähintään yksi ottelu pelin jokaisella hahmolla.

Back-end — Back-end sisältää kaiken palvelimella suoritettavan sovelluksen tai verkkosivuston osaset, johon kuuluvat esimerkiksi yhteydet tietokantoihin, lomakkeiden ja tiedostojen käsittelyt, sovellukseen tai sivustolle kirjautuminen sekä salasanojen tarkistaminen. Back-end siis käsittelee, siirtää ja vastaanottaa verkkosivuston tai sovelluksen tietoja (ise s.a).

Front-end — Sovelluksen selainpuoli. Front-end on se verkkosivuston tai sovelluksen

	osa, jonka kanssa sen käyttäjä on tekemisissä. Se sisältää siis sivuston tai sovelluksen käyttöliittymän (ise s.a).
GIT	Git on hajautettu versiohallintajärjestelmä, joka on suunniteltu tallentamaan ja seuraamaan koodin muutoksia sekä helpottamaan yhteistyötä kehittäjien välillä. Git mahdollistaa koodin versionhallinnan, haarojen luomisen, muutosten yhdistämisen ja monia muita toimintoja. Sitä käytetään yleisesti ohjelmistokehityksessä ja muiden tekstipohjaisten tiedostojen hallinnassa.
GitHub	GitHub on pilvipohjainen palvelu, joka tarjoaa Git-versionhallinnan keskitetyn palvelimen. GitHub helpottaa projektien jakamista ja yhteistyötä kehittäjien välillä.
Java	Java on yleiskäyttöinen oliopohjainen ohjelmointikieli, jonka avulla ohjelmoijat voivat kirjoittaa koodia, joka kääntäessä toimii kaikilla Javaa tukevilla alustoilla ilman tarvetta kääntää koodia uudelleen.
JavaScript	Pääasiassa verkkoympäristössä käytettävä ohjelmointikieli. JavaScriptillä toteutetaan muun muassa toiminnallisuus, kuten erilaiset valikot ja muut dynaamiset elementit verkkosovelluksiin.
JSON	Yksinkertainen ja kevyt tiedostomuoto. JSON on ohjelmointikielestä riippumaton ja sitä käytetään datan siirtämiseen ja tallentamiseen eri sovellusten ja palveluiden välillä. “JSON” lyhenne tulee sanoista JavaScript Object Notation.
League of Legends	Riot Gamesin kehittämä ja julkaisema tietokoneella pelattava strategiapeli. Pelissä pelataan otteluita, jossa kaksi viiden pelaajan joukkuetta yrittää tuhota vastustajajoukkueen tukikohdan. Ottelun voittaa joukkue, joka onnistuu tuhoamaan vastustajan tukikohdan ensimmäisenä.
Node.js	Node.js on avoimen lähdekoodin JavaScript-ympäristö, joka suorittaa JavaScript-koodia palvelimen puolella.
Ohjelmointirajapinta	Rajapinta, jolla kaksi tai useampi ohjelmaa pystyy vaihtamaan dataa keskenään. Tunnetaan myös nimellä API (Application Programming Interface).

PostgreSQL PostgreSQL on avoimen lähdekoodin, relationaalinen tietokantajärjestelmä (ORDBMS), joka tarjoaa tehokkaan, luotettavan ja laajennettavan tavan tallentaa ja hallita tietoja.

React.js React.js on JavaScript-kirjasto, jonka avulla voidaan luoda käyttöliittymiä verkkosovelluksiin. Se on kehitetty Facebookin toimesta ja sitä käytetään laajalti modernissa web-kehityksessä.

REST-rajapinta REST-rajapinta (Representational State Transfer) on arkkitehtuurinen tyyli, jota käytetään web-sovellusten ja palvelinten välisessä kommunikoinnissa. REST-rajapinta käyttää yleensä JSON- tai XML-formaattia tiedon siirtoon.

Riot Games Amerikkalainen pelikehitykseen ja julkaisuun keskittynyt yritys, jonka tunnetuin tuote on League of Legends videopeli. Riot Games tarjoaa myös ohjelmointirajapinnan, josta kehittämämme sovellus saa otteludataa pelatuista League of Legends otteluista.

Scrum Scrum on ketterä projektinhallintakehys, joka auttaa tiimejä jäsentämään ja hallitsemaan työtään arvojen, periaatteiden ja käytäntöjen avulla (Atlassian s.a). Scrumia käytetään yleisimmin ohjelmistoprojekteissa ja sen yleisiin käytäntöihin kuuluu muun muassa kokonaisprojektin jakaminen pienempiin osiin ja päivittäiset lyhyet palaverit, joissa käydään läpi projektin etenemistä.

Spring Boot Suosittu viitekehys tuotantotason Java web-sovellusten kehittämiseen. Spring Boot helpottaa Java Spring sovellusten luomista muun muassa yksinkertaistamalla sovelluksen ominaisuuksien määrittämisen ja luomalla heti toimivan itsenäisen sovelluksen, jota kehittäjä voi alkaa muokkaamaan tarpeisiinsa sopivaksi (ibm s.a).

Sprint Scrum kehitysmallissa käytetty aikajakso, jonka aikana kehitetään pieni osa kokonaisprojektista. Ennen jokaisen sprintin alkua työn tavoitteet määritellään projektin työryhmän kokouksessa.

SSH SSH (Secure Shell) on tietoverkon protokolla, jonka avulla voidaan muodostaa salattu ja turvallinen yhteys kahden tietokoneen välille. SSH-yhteyden avulla käyttäjät pystyvät etäkäyttämään toisen

tietokoneen komentoriviä ja sitä käytetään laajasti muun muassa etäpalvelinten hallintaan.

Verkkotunnus Verkkosivun helposti ihmisluettava osoite, joka viittaa palvelimen IP-osoitteeseen. Verkkotunnuksen avulla voidaan muuttaa esimerkiksi IP-osoite “12.34.56.789” helposti luettavaan muotoon “www.esimmerkki.com”.

2 Sovelluksen suunnittelu

Ennen varsinaisen sovelluksen ohjelmoinnin aloittamista tarvitaan suunnitelma. Projektin laajuudesta ja sen tekijöiden lukumäärästä riippuen suunnitelma voi olla hyvinkin laaja ja yksityiskohtainen tai vain suullinen ja suuntaa antava. Suunnitteluvaiheessa on hyvä määrittää muun muassa seuraavat asiat: Mihin tarkoitukseen sovellusta tultaisiin käyttämään ja mitkä ovat sen toiminnallisuudet? Ketkä ovat sovelluksen kohderyhmää, eli ketkä tulisivat sovellusta mahdollisesti käyttämään?

Lisäksi on hyvä kartoittaa sovelluksen ominaisuudet ja toiminnallisuus mahdollisimman tarkasti. Ominaisuuksien kartoittamisella ehkäistään niin sanotun ”scope creepin” syntymistä, joka tarkoittaa sovelluksen laajuuden paisumista liian suuriin mittoihin kehityksen aikana. Jatkuvasti paisuva projekti saattaa tehdä tuotteesta liian monimutkaisen käytettäväksi ja uudet ominaisuudet hidastavat kehitystä, jolloin projektille määritetyt aikarajat saattavat ylittyä mahdollisesti vakavine seurauksineen.

Sovelluskehityksen suunnitteluvaiheessa on myös tärkeää määrittää mitä osia kehitettävä sovellus tulee tarvitsemaan toimiakseen ja miten eri osat tulevat toimimaan keskenään. Kun sovellukseen vaadittavat osat ovat tiedossa, on valittava niiden rakentamiseen käytettävät teknologiat. Esimerkiksi millä ohjelmointikielillä ja viitekehyksillä projektin osat tullaan rakentamaan ja mikä ohjelmointikielen versio soveltuu parhaiten projektin kullekin osalle. Lisäksi projektin etenemisen seurantaan ja projektin aikaiseen kommunikaatioon on hyvä määrittää ratkaisut suunnitteluvaiheessa.

2.1 ARAM Challengen suunnittelu

ARAM Challengen suunnitteluvaiheessa päätettiin, että kehitettävä sovellus tulee koostumaan selainpuolesta (front-end), palvelinpuolesta (back-end) ja erillisestä relaatiotietokannasta. Sovelluksen päätarkoitus on yksinkertaisesti hakea dataa pelatuista League of Legends pelin otteluista pelin julkaisijan Riot Gamesin ohjelmointirajapinnasta ja esittää sitä tietyssä muodossa sovelluksen käyttäjille. Sovelluksen odotettu käyttäjäkunta muodostuu siis League of Legends pelin pelaajista ja vielä tarkemmin “ARAM” pelimuodon pelaajista, jonka otteludataan sovellus keskittyy.

Projektissa oli mukana kaksi tekijää. Päätimme käyttää projektinhallintaan ketterää Scrum-viitekehystä, jota sovelsimme tarpeidemme mukaan. Määritimme sovelluksen kehitysprosessin koostuvan kolmesta eri vaiheesta, eli sprintistä. Jokaiselle sprintille määritettiin omat tavoitteet ja vaatimukset ennen sprintin alkua. Tavoitteet ja vaatimukset kirjattiin pieniksi erillisiksi tehtäviksi ja niiden seurantaan sekä hallintaan käytettiin Microsoftin Teams palvelun tarjoamaa tehtävälistaa. Yksi iso

projekti siis jaettiin kolmeen pienempään osaan, joka teki projektin etenemisen seurannasta helppoa.

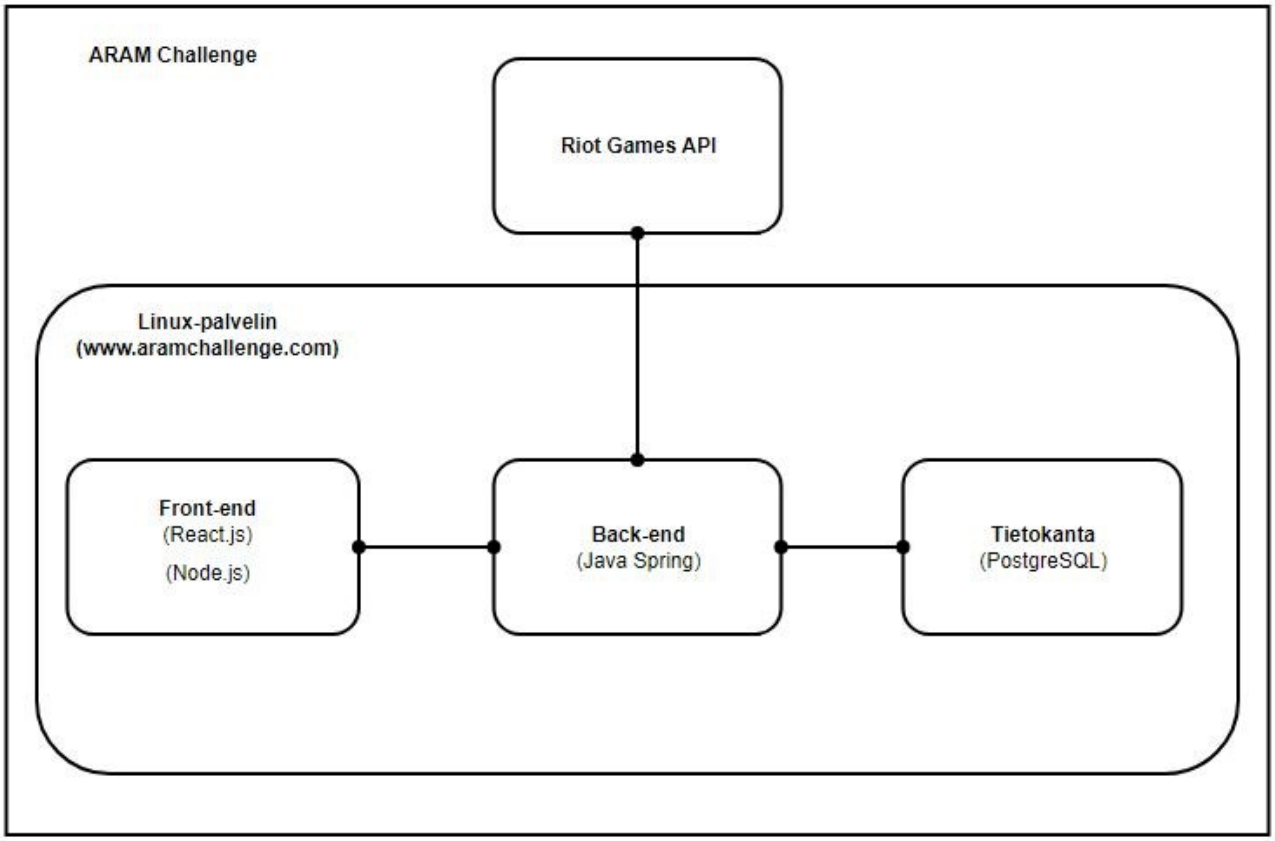
Perinteisestä Scrum projektinhallinnan menetelmistä poiketen emme pitäneet päivittäisiä palavereja työn edistymisestä. Näimme sen tarpeettomaksi kahden hengen projektille ja keskustelimme projektin etenemisestä kuitenkin Teamsin ja Discordin välityksellä lähes päivittäin. Kuitenkin säännöllisiä tapaamisia järjestettiin projektin edetessä ja varsinkin sprinttien suunnittelupalaverit käytiin puheyhteyden välityksellä.

2.2 Teknologioiden valinta

Kehitettävä sovellus määritettiin koostumaan selainosasta, palvelinosasta ja tietokannasta. Sovellus vaatii toimiakseen myös datan lähteen, joka meidän tapauksessamme on Riot Gamesin tarjoama ohjelmointirajapinta (API). Sovelluksen julkaisuun vaaditaan vielä erillinen Linux-palvelin ja verkkoalusta, johon palvelun osat voidaan laittaa käyntiin ja käyttäjien saataville.

Selainosan päätimme lähteä toteuttamaan käyttäen Node.js-ympäristössä toimivaa React.js-JavaScript sovellusta. Node.js on avoimen lähdekoodin ajonaikainen ympäristö JavaScript-koodin suorittamiseen (Sheldon, Denman 2022). Node toimii alustana React-sovelluksellemme, ja ne yhdessä muodostavat ARAM Challengen front-endin. Reactilla ja Nodella rakennettu front-end tulee toimimaan palvelumme käyttöliittymänä jokaisen käyttäjän nettiselaimessa. Päädyimme käyttämään kyseisiä teknologioita, koska kummallakin projektin tekijällä oli aikaisempaa kokemusta niiden käytöstä niin koulun kursseilta kuin omista henkilökohtaisista projekteistakin.

Hyvin samoista syistä päätimme toteuttaa sovelluksen “back-endin” käyttämällä Java-ohjelmointikieltä ja Spring Boot -viitekehystä, joka soveltuu erinomaisesti web-palveluiden kehittämiseen. Back-end toimii nimensä mukaisesti palvelun taustalla ja on vastuussa muun muassa tiedon siirtämisestä ja muokkaamisesta sovelluksen eri osien välillä. Esimerkiksi jos käyttöliittymä tarvitsee dataa tietokannasta, pyytää se ensin tarvitsemaansa dataa back-endiltä, joka hakee datan tietokannasta ja palauttaa sen käyttöliittymälle käyttäjän saataville. Datan siirtoa varten kehitämme REST-rajapinnan, jonka tarkoituksena on siirtää dataa yksinkertaisessa JSON-muodossa. Kuvassa 1 kuvataan ARAM Challengen muodostavia osia ja niiden yhteyksiä toisiinsa.



Kuva 1. ARAM Challengen muodostavat osat

3 Sovelluksen kehitys

Sovelluksemme kehitys jaettiin kolmeen eri Scrum projektinhallinnan viitekehyksen käytäntöjen mukaiseen osaan, eli sprinttiin. Ennen jokaisen sprintin alkua kävimme keskustelun siitä, mitä sprintin aikana tultaisiin tekemään ja määrittelimme sprintin aikaiset tehtävät. Tehtävät oli kirjattu Teamsin tarjoamaan tehtävätauluun, josta niitä pystyi seuraamaan helposti ja näki mitkä tehtävät oli tehty ja mitkä vielä tekemättä.

Sprinttien pituudet ja tärkeimmät tehtävät määrittyivät seuraavalla tavalla. Ensimmäisessä sprintissä keskityttiin projektin alustamiseen. Projektin eri osien alkuversiot, joita myöhemmin oli tarkoitus kehittää projektin tarpeisiin sopiviksi, luotiin ja ne lisättiin versionhallintaan helppoa hallintaa varten. Ensimmäisen sprintin kestoksi määritettiin noin kolme viikkoa.

Toisen sprintin aikana sovelluksen eri osat kehitettiin toimimaan halutulla tavalla. Tämän sprintin kestoksi määriteltiin noin viisi viikkoa, joka tekee siitä hieman muita sprinttejä pidemmän. Sprintille varattiin reilusti aikaa, koska projektin suurin kehitysosa tapahtuisi tässä sprintissä ja ohjelmistokehityksen aikana syntyy usein yllättäviä ja aikaa vieviä ongelmia.

Kolmannen ja viimeisen sprintin tavoitteeksi määritettiin sovelluksen julkaisu. Tavoitteena oli sprintin loppuun mennessä saada valmis sovellus näkyville internettiin sen käyttäjiä varten. Sprintin kestoksi määritettiin noin kolme viikkoa.

3.1 Sprint 1

Projektin ensimmäisessä kehitysvaiheessa lähdimme tekemään varsinaista ohjelmistokehitystä edeltäviä toimenpiteitä. Ensimmäisen sprintin tavoitteiksi oli määritetty muun muassa sovelluksen front-endin ja back-endin ensimmäisten versioiden luominen ja niiden arkistointi versionhallintaan. Versionhallintaan käytimme Git-versionhallintajärjestelmää, joka tallentaa projektin tiedostot GitHub-palvelun pilvipalvelimelle. Git-versionhallinta helpottaa projektiin tehtyjen muutosten seurantaa ja mahdollistaa projektin samanaikaisen kehityksen usean kehittäjän osalta.

Ensimmäisen sprintin tavoitteena oli myös pystyttää kehityspalvelin sovelluksen eri osien testaamista varten. Sprintin aikana kartoitettiin ja dokumentoitiin Riot Gamesin tarjoaman ohjelmointirajapinnan osoitteet, joista sovelluksemme tulee saamaan tarvitsemansa datan. ARAM Challenge tarvitsee otteludataa pelatuista otteluista, julkista dataa pelin käyttäjistä sekä dataa pelin eri pelihahmoista saavuttaakseen tarkoituksenmukaisen toiminnallisuuden. Riot Gamesin tarjoama ohjelmointirajapinta, eli API, on saatavilla ilmaiseksi kaikille halukkaille käyttäjille. API:n käyttämiseen vaaditaan kuitenkin erityinen avain. Riot Games rajoittaa palvelun satunnaisia käyttäjiä tarjoamalle yksityisille käyttäjille vain väliaikaisen avaimen, joka vanhenee 24 tunnin kuluttua avaimen

luonnista ja on aina generoitava uudestaan edellisen vanhentuessa. Pysyvän avaimen saamiseksi tuotantotarkoituksia varten on rekisteröitävä sovellus, jossa rajapinnan dataa aikoo käyttää, Riot Gamesin API-palveluun. Sprintin aikana rekisteröimme sovelluksemme ja saimme pysyvän avaimen sovelluksen käyttöön.

3.1.1 Sovelluksen osien luonti

Sovelluksemme määritettiin koostumaan kolmesta eri osasta. Web-käyttöliittymästä, tietokannasta ja back-end palvelinosasta. Lisäksi datan lähde, eli Riot Gamesin ohjelmointirajapinta on periaatteessa yksi sovelluksen osa mutta sen toimintaan emme pysty vaikuttamaan.

Back-endin luomiseen käytimme internetissä saatavilla olevaa “Spring Initializr” palvelua. Palvelussa määritetään Java Spring Boot sovellukselle halutut ominaisuudet, jonka jälkeen sivulta pystyy lataamaan uuden tyhjän Spring Boot sovelluksen määritettyine ominaisuuksineen. Luodun sovelluksen latasimme Git versionhallintaan, josta sen jatkokehittäminen halutun toiminnallisuuden mukaiseksi olisi helppoa.

Sovelluksemme käyttöliittymän sekä tietokannan luominen sujui ongelmitta. JavaScript-koodin suorittamiseen tarkoitettu ajoympäristö Node.js loi uuden tyhjän React.js-käyttöliittymäsovelluksen yhdellä komennolla. Kuten back-endin kanssa, myös tyhjä käyttöliittymäsovellus tallennettiin omaan Git-repositorioonsa. Tietokantaa varten asensimme PostgreSQL-tietokannanhallintajärjestelmän sekä pgAdmin 4-nimisen ohjelman, joka tarjoaa graafisen käyttöliittymän tietokannan hallintaan.

Ensimmäisen sprintin tavoitteena oli myös saada sovelluksen osat kommunikoimaan keskenään. Sprintin loppupuolella kehitimme muutaman testaukseen tarkoitettuja toimintoja sekä back-endiin että front-endiin. Näiden avulla saimme lähetettyä yksinkertaista dataa käyttöliittymästä back-endin kautta tietokantaan ja haettua käskystä tietokannasta dataa takaisin käyttöliittymälle. Sovellus toimi sekä lokaalissa ympäristössä että kehitystä varten pystyttämällämme kehityspalvelimella.

3.1.2 Kehityspalvelin

Sovelluksen kehittämisen ajaksi asensimme valmiiksi olemassa olevaan henkilökohtaiseen palvelimeen käyttöjärjestelmän ja sovelluksemme ajamiseen vaadittavat ohjelmat. Tämän ansiosta pystyimme testaamaan sovelluksen toimintoja ympäristössä, joka oli mahdollisimman lähellä lopullisen tuotantopalvelimen kokoonpanoa.

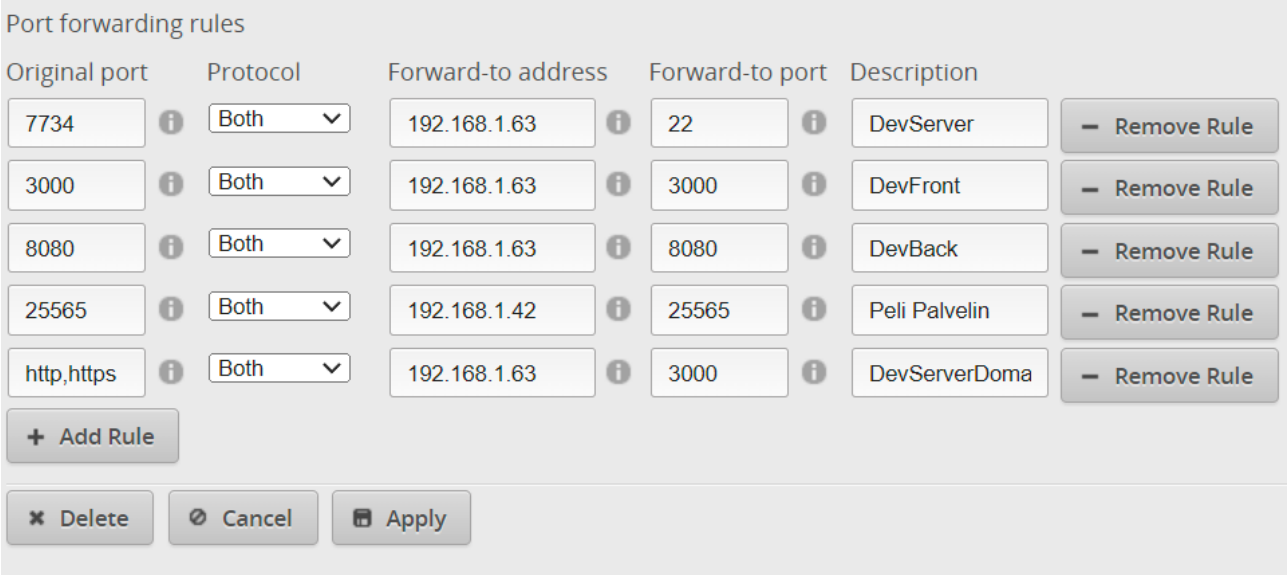
Palvelimen käyttöjärjestelmäksi valitsimme Debian GNU/Linux 11:n. Valitsimme kyseisen jakelupaketin, koska kummallakin meistä on aikaisempaa kokemusta Linux-pohjaisten käyttöjärjestelmien käyttämisestä komentoriviltä. Aikaisempi kokemus käyttöjärjestelmästä nopeutti sen käyttöönottoa

ja minimoi ongelmat tarvittavia ohjelmia asennettaessa sekä järjestelmän edellyttävien konfiguraatioiden asettamisessa.

Testipalvelimelle asensimme myös SSH-palvelinsovelluksen ja UFW-palomuurin. Asennus tapahtui komennolla “Sudo apt-get install ssh-server ufw –y”. Komento asensi pääkäyttäjän oikeuksilla SSH-palvelinsovelluksen, joka mahdollistaa palvelimeen hallinnan etäyhteyden välityksellä. UFW on yksinkertainen palomuurin hallintasovellus, joka otetaan käyttöön komennolla “Sudo ufw enable”.

Kun palomuuri on päällä, palvelin ei salli muiden tietokoneiden muodostaa verkkoyhteyksiä palomuurin suojaamalle palvelimelle. Palomuuriin on kuitenkin avattava sovelluksen toiminnalle oleelliset portit. Valitsemamme front-end ja back-end teknologiat käyttävät oletuksena portteja 3000 ja 8080. Myös SSH-yhteys tarvitsee vapaan portin, joka on oletuksena portti 22. Tarvittavien porttien avaaminen tapahtui komennolla “Sudo ufw allow 3000 8080 22”. Lisäksi avasimme sovelluksemme tietokannanhallintajärjestelmän PostgreSQL 13:n oletuksena käyttämän portin 5432. Avoimet portit ja palomuurin toiminnallisuuden voi tarkistaa komennolla: “Sudo ufw status”. Yleisien verkkoporttien, kuten SSH:n käyttämän portin 22, avaaminen kotiverkon ulkopuoliselle verkkoliikenteelle on huomioitava verkkoturvallisuusriski.

Porttiohjaukset luodaan kirjautumalla kotiverkon ja ulkoisen verkon välissä olevalle reitittimelle ja määrittämällä säännöt, joiden perusteella reititin ohjaa verkkoliikenteen halutulle laitteelle. Esimerkiksi kuvassa 2 porteille 7734, 3000 ja 8080 kohdistuva verkkoliikenne ohjataan testipalvelimelle sisäverkon IP-osoitteeseen 192.168.1.63.



Kuva 2. Verkkoporttien hallintanäkymä

PostgreSQL-tietokannan pystyttämisessä kehityspalvelimelle esiintyi haasteita. Tietokantasovelluksessa on sisäänrakennettuja palomuurin kaltaisia ominaisuuksia, jotka estivät verkkoyhteyden muodostamisen tietokantaan. Tietokannan verkkoyhteyttä ja toiminnallisia ominaisuuksia säätelevät tiedostot löytyvät pyytämällä niiden sijaintia suoraan ohjelmalta itse komennolla “sudo psql -U postgres -c 'SHOW config_file’”. Komento palauttaa tiedoston sijainnin, josta tiedosto löytyy muokkaamista varten. Tietokannan asennuksen yhteydessä loimme myös käyttäjän tietokannalle, jonka tunnuksia vaaditaan tietokannan yhdistämisessä sovelluksemme back-end osaan.

Kehityspalvelimelle asensimme myös pakettihallintaohjelman npm (Node Package Manager). Tätä paketinhallintasovellusta käytetään Node.js-palvelinohjelmiston toimintaan sekä React.js pakettien lataamiseen.

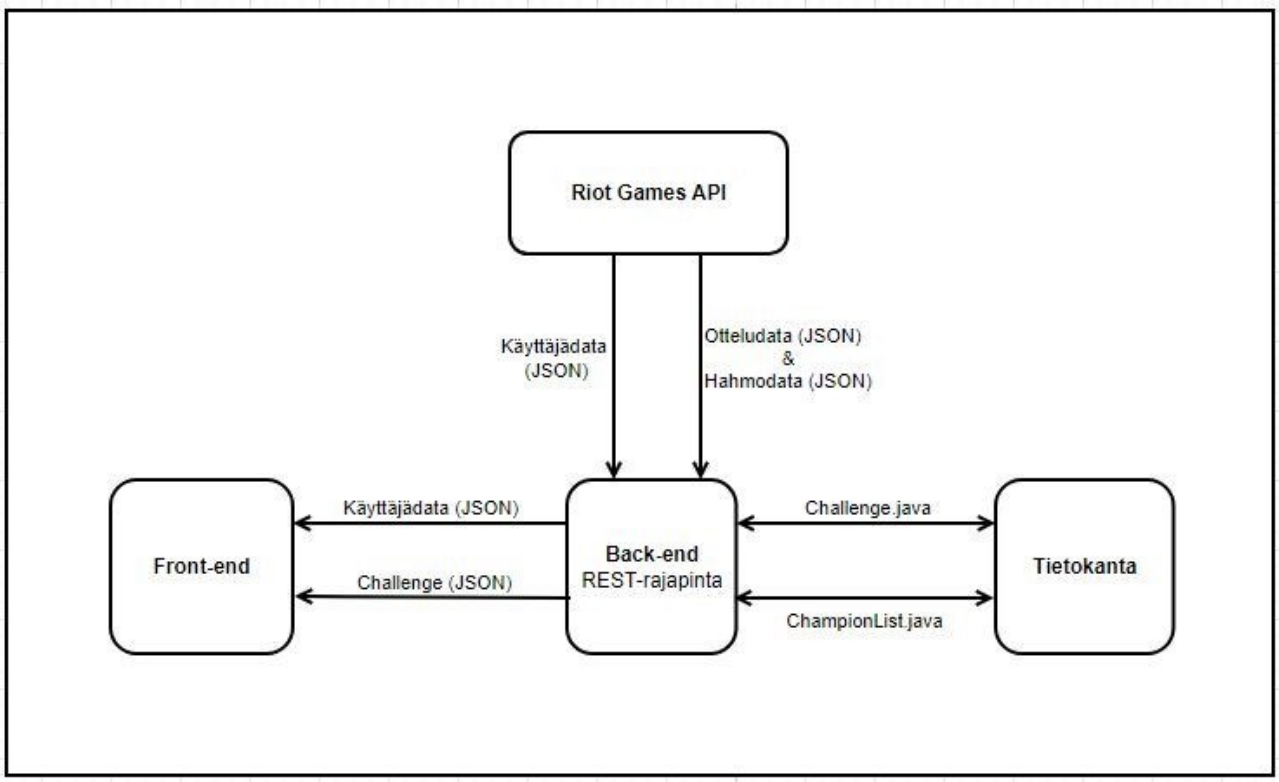
Back-endimme suorittamiseen vaadittavan Java JDK:n asentamisessa kehityspalvelimelle esiintyi ongelmia, sillä sen käyttämä Java-versio ei ole suoraan tuettu Linux 11 -järjestelmissä. Tarvittavan Java-version toimintaan saamiseksi, piti se ladata manuaalisesti ja muuttaa Javan oletusasetuksia (Datastax 2023).

3.2 Sprint 2

Toisen sprintin teemana oli jatkokehittää ensimmäisessä sprintissä luotuja sovelluksen osia. Tarkoituksena oli saada osat käyttämään haluttua dataa ja kehittää ne toimimaan keskenään sovellukselle määritetyillä tavoilla. Tämän sprintin jälkeen sovelluksen tulisi olla siinä kunnossa, että sen kaikki määritellyt toiminnot toimivat odotetulla tavalla ja sen voisi julkaista käyttäjien saataville.

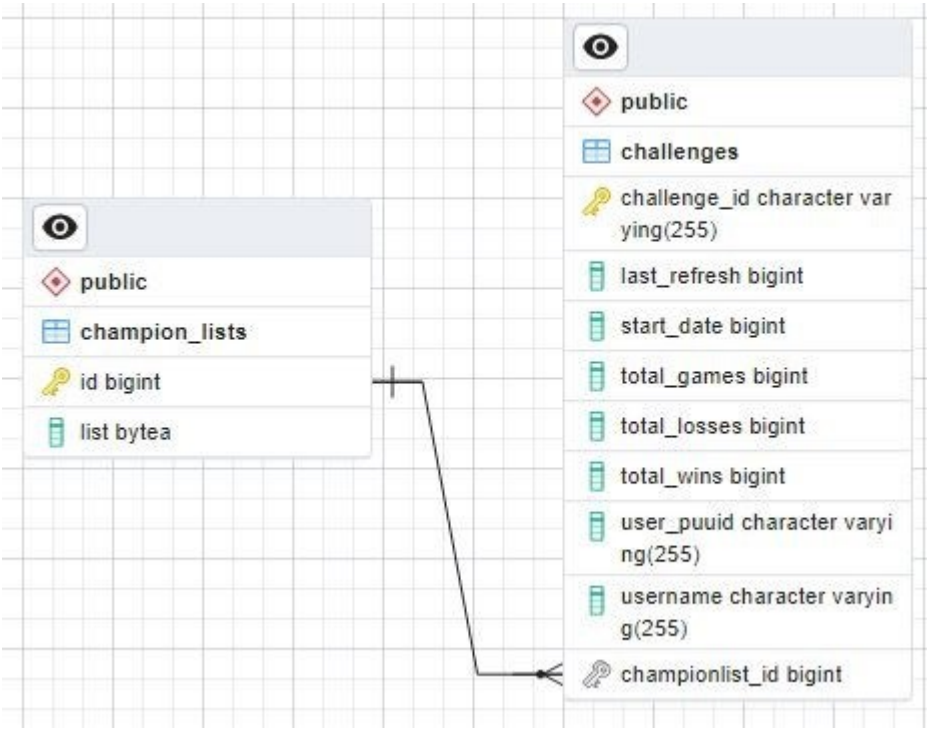
Datan kulku sovelluksen eri osien välillä, jonka toteutimme tämän sprintin aikana, määriteltiin seuraavanlaisesti. Kaikki sovelluksen dataliikenne kehitettiin kulkemaan back-endin kautta (kuva 3). Riot Gamesin ohjelmointirajapinnasta sovelluksemme hakee käyttäjädataa, otteludataa pelatuista otteluista ja hahmodataa pelin tämänhetkisistä pelattavista pelihahmoista. Riot Gamesin tarjoama data on aina JSON-muotoista.

Front-endin ja back-endin välillä datan kulku on yksisuuntaista. Front-end saa käyttäjälle esitettävää dataa back-endiltä, mutta ei varsinaisesti lähetä uutta dataa muokattavaksi tai tallennettavaksi back-endin suuntaan. Käyttöliittymän saama data back-endistä on JSON-muotoista, joka tekee sen hallinasta ja esittämisestä helppoa palvelumme front-end sovelluksessamme.



Kuva 3. Datan kulku sovelluksen osien välillä

Dataa, jota back-end lähettää pyynnöstä käyttöliittymälle, noudetaan lähes poikkeuksetta tietokannasta. Sovelluksemme tallentaa tietokantaan kahdenlaista dataa. ChampionList-objekteja ja Challenge-objekteja. Tietokannassa on siis kaksi tietokantataulua, joihin edellä mainittuja objekteja tallennetaan tietueiksi (kuva 4). Jokaisella Challenge-objektilla on viittaus yhteen ChampionList-objektiin.



Kuva 4. Tietokantataulut, joihin sovelluksen dataa tallennetaan

3.2.1 Back-end kehitys

Sovelluksemme back-end toimii muiden osien välillä ja on vastuussa datan noutamisesta, lähettämisestä ja sen muokkaamisesta. Tähän tarkoitukseen kehitimme sovelluksemme palvelinosalle niin sanotun “REST-rajapinnan”. Rajapinnan toimintaperiaate on käytännössä täysin sama kuin datalähteemme Riot Gamesin palvelulla, joka palauttaa pyynnöstä tiettyä dataa käyttäjälleen JSON-muodossa.

Sovelluksemme back-endille koodattiin osoitteita, joita kutsumalla back-end vastasi lähettämällä osoitteen kutsulle määritettyä dataa takaisin JSON-muotoisena. Sovelluksemme back-end haki dataa aina joko tietokannasta tai kutsui itse Riot Gamesin ohjelmointirajapintaa ja lähetti sieltä vastauksena saatua dataa eteenpäin käyttöliittymälle.

Sovelluksemme päätoimintaperiaate on suunniteltu seuraavanlaisesti. Palvelun käyttäjä antaa käyttöliittymään League of Legends pelin käyttäjänimensä. Käyttöliittymä kutsuu palvelinta, joka luo uuden “challenge”-objektin käyttäen käyttäjän käyttöliittymälle syöttämää käyttäjänimeä. Uuden challengen luonnin yhteydessä back-end kutsuu Riot Gamesin ohjelmointirajapintaa ja saa vastaukseksi listan tämänhetkisistä pelattavista pelihahmoista. Tämä vaihe on tärkeä sovelluksen toiminnan kannalta siten, että kaikki senhetkiset pelattavat hahmot tulevat varmasti näkyvillle käyttäjälle. Uusia hahmoja lisätään peliin säännöllisin väliajoin, joten staattinen valmis lista ei välttämättä sisällä kaikkia hahmoja ja se vaatisi manuaalista päivittämistä aina uuden hahmon julkaisun

yhteydessä. Uudesta hahmodatasta luodaan “ChampionList”-Java-objekti ja se tallennetaan tietokantaan. Kuvassa 5 esitetään ChampionList-objektin sisältämää dataa.


```
[
  {
    "id": 40,
    "list": [
      {
        "id": "Aatrox",
        "name": "Aatrox",
        "wins": 0,
        "losses": 0
      },
      {
        "id": "Ahri",
        "name": "Ahri",
        "wins": 0,
        "losses": 0
      },
      {
        "id": "Akali",
        "name": "Akali",
        "wins": 0,
        "losses": 0
      },
      {
        "id": "Akshan",
        "name": "Akshan",
        "wins": 0,
        "losses": 0
      },
```


Kuva 5. Osa ChampionList-objektista JSON-muodossa

Challenge-objektille tallennetaan viittaus juuri luotuun ChampionList-objektiin objektin uniikin id:n perusteella. Lopulta uusi challenge-objekti tallennetaan omaan tietokantatauluunsa ja käyttöliittymään lähetetään vastauksena Challengen oma uniikki id, jonka avulla käyttäjä pääsee käsiksi luomaansa dataan myöhemin (kuva 6).

```
{
  "challenge_id": "03bb1bf5-d097-491d-a00e-ad2ed9f569b4",
  "username": "ambaal",
  "user_puuid": "cfQsh74gH1Nzt0xNZ15VoOT_v1d5r1yOmxeDwPr8z4EpSnicdwJHSU2uXgSSyRU7HuZMDqJisziFJQ",
  "totalGames": 0,
  "totalWins": 0,
  "totalLosses": 0,
  "startDate": 1683858101669,
  "lastRefresh": 1683858101669,
  "championlist": {
    "id": 42,
    "list": [
      {
        "id": "Aatrox",
        "name": "Aatrox",
        "wins": 0,
        "losses": 0
      },
      {
        "id": "Ahri",
        "name": "Ahri",
        "wins": 0,
        "losses": 0
      },
      {
        "id": "Akali",
        "name": "Akali",
        "wins": 0,
```


Kuva 6. Käyttäjälle "ambaal" luotu Challenge-objekti JSON-muodossa

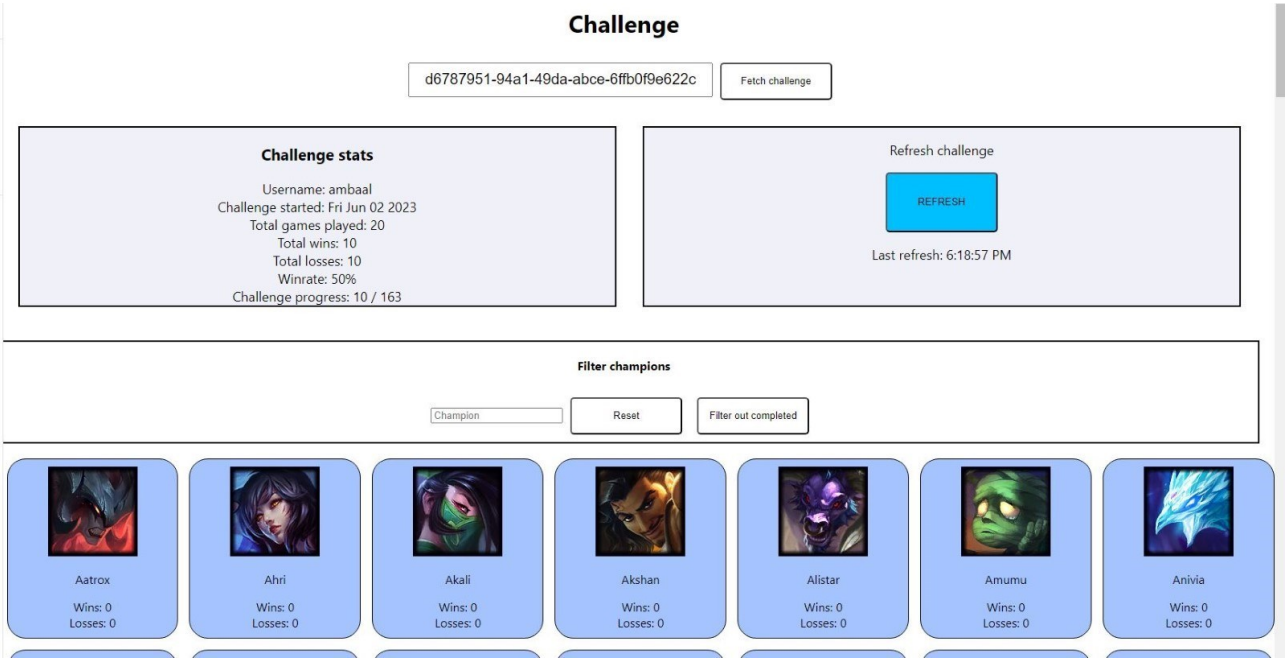
Sovelluksemme tarkoitus on myös seurata pelaajien pelaamien otteluiden tuloksia ja tallentaa niitä. Kyseinen toiminnallisuus on toteutettu challengen päivitystoiminnolla. Kun käyttäjä on pelannut otteluita ja haluaa seurata edistymistään, hakee hän oman challengensa tiedot sovelluksen käyttöliittymästä käyttäen challengen id:tä. Back-end vastaanottaa id:n käyttöliittymältä ja etsii annetun id:n omaavan challengen tietokannasta ja palauttaa sen käyttöliittymään. Käyttäjä näkee nyt oman challengensa tiedot ja voi laukaista challengensa päivityksen, jolloin back-end hakee listan challengeen liittyvän pelaajan käyttäjänimen perusteella listan otteluista Riot Gamesin palvelusta. Saatuaan vastaukseksi listan pelatuista ARAM-otteluista viimeisimmän päivityskerran jälkeen, selvittää back-end otteludatasta, millä hahmoilla käyttäjä on otteluita pelannut ja jakaa voittoja ja häviöitä challenge-olion “championlist”-osan hahmoille. Tämän jälkeen challengeen tehdyt muutokset tallennetaan tietokantaan, ja uusi päivitetty versio käyttäjän challengesta lähetetään takaisin käyttöliittymään käyttäjän näkyville.

3.2.2 Front-end kehitys

Tämän sprintin aikana toteutimme sovelluksemme käyttöliittymäosan ominaisuudet, joilla front-endiin syötettyä dataa pystytään järkevästi tallentamaan back-endin REST-rajapintoja hyödyntäen tietokantaan, sekä tuomaan tietokannasta ohjelmallisesti helposti luettavaa dataa JSON muodossa. Käyttöliittymä näyttää käyttäjälle tietokannasta haetun datan selkeästi taulukkomuodossa.

Front-endin kautta käyttäjät ohjaavat sovellusta. On siis tärkeää, että käyttöliittymä toimii moitteettomasti ja selkeästi. Samalla sen olisi myös tarkoitus näyttää visuaalisesti hyvältä parhaan mahdollisen käyttäjäkokemuksen saavuttamiseksi.

Käyttöliittymämme tärkein käyttötarkoitus on kutsua back-endistä challenge-objektia käyttäjän syöttämällä koodilla ja esittää back-endiltä saatua dataa käyttäjälle. Kyseinen toimenpide on toteutettu käyttäen “Axios”-kirjastoa. Sen avulla kutsutaan back-endin ohjelmointirajapintaa, ja sieltä vastaanotetusta datasta muodostetaa taulukko. Taulukko esitetään käyttäjälle sovelluksen challenge-sivulla, josta näkee taulukon lisäksi tilastodataa haasteen edistymisestä, kuten pelattujen otteluiden määrän ja voittoprosentin (kuva 7). Taulukko esittää hahmot, joilla käyttäjä on voittanut vähintään yhden ottelun harmaalla. Tämä tekee haasteen edistymisen seuraamisesta helppoa ja nopeaa (kuva 8). Muita käyttöliittymän sivuja ja ominaisuuksia kuvataan liitteissä 1–3.



Kuva 7. Challenge-sivun näkymä, kun challenge on noudettu käyttöliittymään tietokannasta

Kuva 8. Hahmojen erottelu voittojen perustella

3.3 Sprint 3

Viimeisessä sprintissä suunnittelimme julkaisun yksityiskohtia ja toteuttimme ARAM Challengen julkaisun. Sprintin aikana pohdimme erilaisia verkkoisännöintiratkaisuja ja vuokrasimme sovelluksellemme oman verkkotunnuksen. Sprintin päätteeksi ARAM Challenge saatiin julkaistua näkyville internettiin ja käyttäjät pystyvät löytämään palvelun sille määritetyllä verkkotunnuksella.

Pyrimme toimimaan julkaisun aikana parhaiden tietoturvamenetelmien mukaisesti ja varmistamaan, ettemme jätä huomiotta turhia tietoturva-aukkoja, jotka voisivat vaarantaa sovellustamme tai palvelintamme. Palvelumme ei käsittele arkaluontoisia tietoja, kuten henkilö- tai käyttäjätietoja, mutta mahdollisiin palvelunestohyökkäyksiin on hyvä aina varautua asianmukaisin menetelmin.

3.3.1 Julkaisun suunnittelu

Suunnittelimme aluksi projektimme julkaisua verkkoisännöintipalveluita tarjoavan DigialOcean palvelun kautta. Vuokrasimme palvelimen, mutta julkaisun aikana kuitenkin huomasimme, että vuokraamamme palvelimen suorituskyky ei riittänyt sovelluksemme front- ja back-end osien samanaikaiseen ajamiseen. Tiukkojen opiskelijabudjettien takia tehokkaamman palvelimen vuokraaminen

ei ollut kustannustehokas ratkaisu projektin julkaisuun. Päädyimme hyödyntämään projektin testipalvelimena toiminutta laitteistoa myös palvelun varsinaiseen julkaisemiseen. (DigitalOcean 2023)

Verkkoyhteyksien avaaminen yksityiseen kotiverkkoon on aina tietoturvariski, joten on tärkeää noudattaa oikeaoppisia tietoturvatoimenpiteitä. Vuokrasimme sovelluksellemme verkkotunnuksen “aramchallenge.com”. Tunnuksen hinta on 8 € vuodessa Namecheap-palvelusta. (Namecheap 2023).

3.3.2 Tuotantopalvelimen asennus

Tuotantopalvelimen asennus oli käytännössä sama kuin testipalvelimen asennus, mutta tuotantopalvelimen asennuksessa täytyi ottaa huomioon tietoturvan kannalta tärkeitä seikkoja. Aluksi asensimme palvelimelle käyttöjärjestelmän uudestaan. Tällä varmistuimme, että palvelimella ei ole asennettuna ylimääräisiä sovelluksia tai kehitysvaiheessa muokattuja tietoliikennemäärityksiä. Käyttöjärjestelmänä käytimme samaa Debian 11 -käyttöjärjestelmää, jota käytimme sovelluksen kehitysvaiheessa.

Seuraavaksi palvelimelle asennettiin sovelluksen vaatimat ohjelmat sekä palomuuri. Asennus tapahtui komennolla ”Sudo apt-get install npm nodejs react postgre-13 git ufw –y”. Javan version 1.8 asentaminen sujui tuotantopalvelimelle helpommin kuin testiympäristössä, eikä palvelimelle tarvinnut asentaa mitään ylimääräistä tai tehdä turhia muutoksia järjestelmän toimimaan saamiseksi. (Datastax 2023)

Asennuksien jälkeen avasimme palomuurista sovelluksen käytössä olevat portit ja estimme ulkoverkosta tulevan verkkoliikenteen porttiin 22. Näin palvelin kieltäytyy muodostamasta SSH-yhteyksiä kyseenalaisilta tahoilta tulevaan liikenteeseen, mutta sisäverkosta on silti mahdollisuus hallita palvelinta komentorivin välityksellä. Viimeiseksi palvelimen pääkäyttäjä eli root-käyttäjä lakkautettiin. Pääkäyttäjätilin puuttuessa ei palvelimella pysty aiheuttamaan katastrofista tuhoa, vaikka palvelimelle saisi muodostettua luvattoman yhteyden.

Jos palvelimella olisi useampia ylläpitäjiä, tai siihen olisi tarpeellista pystyä muodostaan etäyhteys ulkoverkosta, olisi ratkaisu siihen SSH-avain. SSH-avaimet ovat tiedostoja, joilla varmennetaan käyttäjän lupa muodostaa etäyhteys laitteisiin ilman salasanaa. Useista yrityksistä huolimatta emme saaneet SSH-avainta toimimaan tuotantopalvelimella, mutta palomuurimääritysten ja lakkautetun root-käyttäjän ansiosta palvelimen tietoturva on riittävällä tasolla sovelluksemme tallentamien tietojen puitteissa.

4 Web-sovelluksen julkaisu yleisesti

Web-sovelluksen julkaisu on tärkeä osa sen elinkaarta. Toimiva julkaisu auttaa sovellusta menestymään ja parantaa käyttäjäkokemusta. Jos sovellus hidastelee palvelimen verkkoyhteyden takia tai sovelluksen verkkotunnus on epäselvä, ei loppukäyttäjä välttämättä tahdo käyttää sovellusta vaan siirtyy käyttämään toista vastaavanlaista palvelua.

Verkkoisännöintiratkaisuun ja sovelluksen näkyvyyteen verkkotunnuksen avulla kannattaa kiinnittää erityistä huomiota sovellusta julkaistaessa. Oikea verkkoisännöintiratkaisu on kustannustehokas ja luotettava, sekä tarjoaa muun muassa tietoturvaa sovellukselle. Asianmukainen verkkotunnus tuo web-sovellukselle uskottavuutta, sekä auttaa käyttäjiä löytämään sovelluksen paremmin internetistä. Hyvä verkkotunnus on lyhyt, ytimekäs ja se kuvaa sovelluksen tai web-sivun sisältöä hyvin.

4.1 Verkkoisännöinti

Verkkoisännöintiin on olemassa useita eri tapoja. Jokaisella ratkaisulla on omat vahvuutensa ja heikkoutensa. Kolme yleisintä ratkaisua ovat self-hosting, virtual dedicated server ja cloud-hosting.

Self-hosting, suomeksi “itseisännöinti” tarkoittaa kotona tai muussa yksityisessä tilassa olevaa fyysistä tietokone tai palvelinlaitetta, jossa on asennettuna halutut ohjelmistot ja tarvittavat palomuuri määritykset, jotta palvelin on näkyvissä halutuille tahoille. Self-hosting on suosittu vaihtoehto harrastajapiireissä, sillä palvelimiksi kelpaa vanhat tietokoneet ja laitteiston kokoonpanosta saa juuri haluamansa. Huonona puolena tässä ratkaisussa on palvelimen asennukseen ja ylläpitoon kuluva aika. Monet verkkoisännöintipalvelut tarjoavat maksutonta neuvontaa ja tukea vuokraamilleen palvelimille, mutta tätä etua ei ole itseisännöidyillä palvelimilla. On myös hyvä huomioida, että jotkin verkko-operaattorit kieltävät käyttöehdoissaan suurten tai yrityskäyttöön tarkoitettujen palvelimien isännöinnin kotien laajakaistayhteyksillä. Itseisännöidyt palvelimet ja siten myös verkot, joissa palvelimet sijaitsevat, ovat alttiita erilaisille kyberhyökkäyksille. (Self-hosting-Guide)

Virtual Dedicated Server, joskus myös Virtual Private Server (VPS), on yleisin ja yksinkertaisin tapa saada palvelin verkkoon. Tämä tapahtuu vuokraamalla palvelintilaa joltain sitä tarjoavalta yritykseltä. Tämä ratkaisu toimii hyvin kevyessä yrityskäytössä, sekä yksityiskäytössä sellaisissa tilanteissa, jossa oman palvelimen käyttöönotto ei ole mahdollista tilan tai laitteiston puutteessa. Yksityiskäytössä tämä ei välttämättä ole kustannustehokkain ratkaisu, sillä halvimmissa vuokrapalvelimissa on yleensä alhainen tallennustilan määrä ja heikko suorituskyky. Suurempihintaiset palvelimet taas maksavat useita kymmeniä euroja kuukaudessa, jolloin itseisännöitypalvelin tulisi edullisemmaksi ratkaisuksi pidempiaikaisessa käytössä. (Tuomanen 2018 13)

Palvelimen fyysisen sijainnin olisi hyvä olla fyysisesti mahdollisimman lähellä palvelimen arvioitua käyttäjäkuntaa parhaan mahdollisen käyttäjäkokemuksen takaamiseksi. Mikäli arvioitu käyttäjäkunta on jakautunut tasaisesti ympäri maailmaa, olisi hyvä valita palvelin joko läheltä suurinta arvioitua käyttäjien keskittymää tai alueelta, jossa internet yhteydet ja sähköverkko ovat luotettavia.

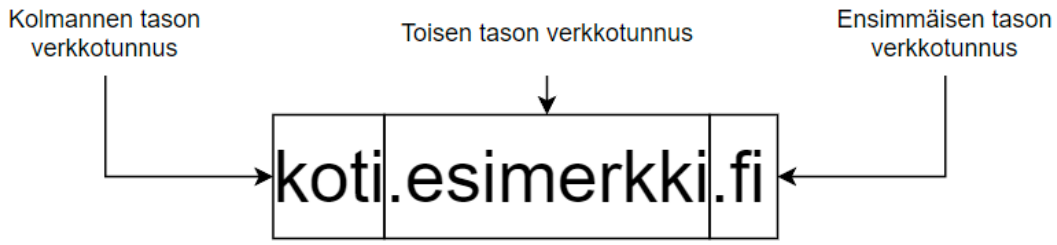
Palvelimenlaitteiston kokoonpanoon on tarpeellista valita tarpeeksi muistia sekä riittävän tehokas suoritin. Sopiva kokoonpano riippui kuitenkin palvelimen käyttötarkoituksesta. Esimerkiksi kevyen verkkosovelluksen pyörittämiseen tarvitaan vähemmän vaativat laitteistovaatimukset. Yksinkertaisin tapa tarkistaa, onko palvelimella tarpeeksi resursseja tarvittavien ohjelmistojen moitteettomaan ajamiseen, on kokeilla sitä. Lisäksi on tärkeää kiinnittää huomiota verkkokaistaan ja sallittuun datan siirtomäärään. Monet tuotantopalvelinvuokraamot asettavat kuukausittaisia tai laskutuskauden mukaan nollautuvia rajoja datansiirtoon. Pelkän tekstin, JSON-datan ja HTTP-kutsujen lähettäminen käyttää hyvin vähän dataa, mutta jos palvelin esimerkiksi käsittelee mediaa, kuten korkealaatuisia kuvia tai videota, datan kulutus nousee nopeasti.

Cloud-hosting on lähivuosien uusi innovaatio verkkoisännöinnissä. Pilvipalvelimen muokkaaminen ja suorituskyvyn päivittäminen sekä uusien palvelimien avaaminen on helppoa ilman palvelimen sammuttamista. Tämä vähentää palvelimen käyttökatkoksia ja varmistaa, että palvelin on aina lähellä käyttäjiä. Tällä hetkellä suurin pilvipalveluita ja -palvelimia vuokraava yritys on Amazon. (Amazon AWS 2023)

4.2 Verkkotunnus

Jokaisella merkittävällä verkkosovelluksella on hyvä olla helposti muistettava ja sopiva verkkotunnus. Verkkotunnuksia on saatavilla monenlaisia. Toisen tason verkkotunnukset saattavat olla hyvinkin kalliita. Hinnat lyhyissä .com ja .org loppuisissa tunnuksissa saattavat nousta jopa kymmeniin tuhansiin euroihin. Suuri osa vapaista tunnuksista ei kuitenkaan maksa tuhansia euroja ja noin viidelläkymmenellä eurolla on vuokrattavissa asianmukaisia tunnuksia.

Mikäli palvelimellasi ei ole tarkoitus tulla laajaan julkiseen käyttöön, on myös mahdollista ostaa tai varata myös ilmainen kolmannen tason verkkotunnus. Kolmannen tason verkkotunnukset eroavat toisen tason tunnuksista, jotka ovat muodossa “esimerkki.fi” olemalla yhden tason alempana, jolloin tunnus tulee jotakuinkin muotoon “kolmannentason.esimerkki.fi” (kuva 9). Yksi kolmannen tason verkko tunnuksia tarjoava yritys on No-IP. No-IP tarjoaa myös maksuttoman DUC palvelun. DUC eli “Dynamic Update Client” on sovellus tai palvelu, joka vahtii dynaamisesti vaihtuvia IP-osoitteita ja päivittää palvelimesi osoitteen verkkotunnus palveluun, jotta palvelinliikenteesi ohjautuu oikeaan osoitteeseen, vaikka verkkopalveluntarjoajasi vaihtaisi IP osoitteesi. (No-IP)



Kuva 9. Verkkotunnuksen muodostavia tasoja

Verkkotunnusta vuokratessasi on kannattavaa, että pystyt luottamaan tahoon, joka sen sinulle vuokraa, sekä valtioon, jonka ensimmäisen tason verkkotunnuksen alta olet tunnuksesi vuokrannut. Harvinaisissa tapauksissa voi tunnusta sinulle vuokraava taho tai verkkotunnuksen omistava valtio lakkauttaa tai muuttaa tunnuksiaan koskevia määräyksiä, joka saattaa estää vuokraamasi tunnuksen käytön.

5 Pohdinta

Opinnäytetyön tarkoituksena oli kehittää ja julkaista web-sovellus, sekä laatia raportti, jossa kuvataan kehittämisprosessia. Pääosin voimme sanoa, että kehittämisprosessin tuloksena syntynyt sovellus täyttää sille asetetut vaatimukset ja sen julkaisuversiossa on suurin osa sille määritetyistä ominaisuuksista. Julkaisun jälkeen tarkoituksenamme olisi pitää sovellus toiminnassa ainakin vuoden verran ja jatkokehittää sovellukselle muutama lisäominaisuus. Sovelluksemme ARAM Challenge on julkaistu internettiin näkyville verkkotunnuksen “www.aramchallenge.com” alle.

Tämän opinnäytetyön suurimmiksi riskeiksi oli määritelty kehitettävän sovelluksen ominaisuuksien paisuminen liian suuriin mittoihin kehityksen aikana, ajankäyttöön liittyvät ongelmat ja projektin heikko suunnittelu. Yksi edellä mainituista riskeistä ei toteutunut ollenkaan, yksi toteutui osittain ja yksi ilmeni etenkin projektin loppupuolella.

Opinnäytetyön toiminnallisen osan julkaisuversion valmistuttua voimme todeta, että sovelluksemme ominaisuuksien määrittelyssä ja rajauksessa onnistuttiin hyvin. Ennen projektin alkua meillä oli visio web-sovelluksesta ja sen toiminnallisuudesta, jota lähdimme sitten toteuttamaan. ARAM Challenge oli rajattu sopivan kokoiseksi ja sen kehittämisen aikana ei tullut uusia ideoita tai ominaisuuksia, joita olisi lähdetty kehittämään paisuttaen sovelluksen kokoa. Niin sanotulta “scope creepiltä” siis vältyttiin.

Yhdeksi riskiksi oli määritetty projektin heikko suunnittelu. Mielestämme itse sovellus ja sen ominaisuudet sekä toiminnallisuus oli suunniteltu hyvin. Ongelmia kuitenkin syntyi sovelluksen julkaisuvaiheessa, kun lähdimme selvittämään mahdollista julkaisualustaa sovelluksellemme. Olimme projektin suunnitteluvaiheessa päätyneet käyttämään yrityksen X tarjoamaa verkkoisännöintipalvelua, mutta sovellusta julkaistaessa kävi ilmi, että kyseisen palvelun tarjoaman palvelimen teho ei yksinkertaisesti riittänyt pyörittämään sovelluksemme eri osia samanaikaisesti. Jouduimme projektin viime hetkillä etsimään uutta julkaisualustaa ja suoritimme siitä syystä ARAM Challengen julkaisun toistaiseksi omalle henkilökohtaiselle palvelimelle, mikä ei ole pidemmällä aikatähtäimellä kannattava ratkaisu. Ongelmaa ei olisi syntynyt, jos julkaisua olisi suunniteltu paremmin ja esimerkiksi selvitetty mikä palvelu tarjoaisi toimivan julkaisuratkaisun.

Suurin toteutunut riski projektissa oli ajankäyttöön liittyvät ongelmat. Suurin osa suunnitellusta sovelluskehityksestä tuli valmiiksi ajallaan ilman sen suurempia ongelmia. Ajankäytöllisiä ongelmia alkoi kuitenkin ilmenemään, kun aloimme keskittyä raportin kirjoittamiseen. Olimme aliarvioineet täysin raportin laatimiseen vaadittavan ajan ja se näkyi etenkin projektin loppupuolella, kun opinnäytetyölle varattu aika alkoi käydä vähiin. Käytännössä se tarkoitti sitä, että iso osa raportista laadittiin kiireellä projektin viimeisinä viikkoina, kun sen olisi voinut kirjoittaa pienissä osissa pitkin

projektia. Asiaa ei auttanut se, että kummallakin tämän opinnäytetyön tekijöistä on ikävä piirre jättää asioita viime tinkaan.

5.1 Oppimisen pohdintaa

Projektin aikana opimme molemmat paljon sovelluskehityksestä. Kehittämämme sovellus oli molemmille opinnäytetyön tekijöille suurin yksittäinen sovellus, jota olimme toistaiseksi kehittäneet itsenäisesti alusta loppuun. Sovelluksemme osien käyttämiksi teknologioiksi valikoitui pääosin sellaisia, joista kummallakin oli edes hieman aikaisempaa kokemusta, mutta sovelluskehityksen aikana törmäsimme tilanteisiin, joissa jouduimme opettelemaan täysin uusia menetelmän. Tämä kehitti osaamistamme muun muassa Java-ohjelmoinnista ja Reactista. Myös tietokannan käyttö datan tallentamiseen oli yksi osa-alue, josta opimme paljon uutta. Käyttämämme PostgreSQL-tietokannanhallintajärjestelmä ja sen käyttäminen tuli kummallekin tutuksi kehityksen aikana.

Sovelluksen julkaisu käyttäjien saataville oli uusi prosessi molemmille projektiin osallistuneille. Olimme kyllä pyörittäneet erilaisia sovelluksia omilla lähiverkoillamme, mutta vastaavan sovelluksen saaminen näkyville mahdollisille käyttäjille ympäri maailmaa oli täysin erilainen prosessi. Sovellusta julkaistaessa jouduimme pohtimaan asioita, kuten verkkotunnuksia, verkkopalveluntarjoajia ja palvelimia sekä niihin liittyviä kustannuksia.

5.2 Jatkokehitys

League of Legends on yksi maailman tämän hetken pelatuimmista videopeleistä (Baumgartl 2023). Sen suosiota selittää se, että peli on pelattavissa täysin ilmaiseksi ja sen alhaiset systeemivaatimukset mahdollistavat pelin pelaamisen myös vanhemmilla tietokoneilla. Pelin kehittäjä ja julkaisija Riot Games ei virallisesti julkaise pelin tarkkoja käyttäjämääriä mutta kuukausittaisia pelaajamääriä pystytään arvioimaan tarkastelemalla pelattuja otteluita. Kamberovic (2023) kirjoittaa artikkelissan pelillä olevan noin 180 miljoonaa aktiivista kuukausittaista käyttäjää vuonna 2023. Pelin pelaajakunta on jakautunut alueittain eri palvelimille, joista suurin on Kiinan palvelin, jolla on noin 75 miljoonaa kuukausittaista käyttäjää.

Kehittämämme sovellus ARAM Challenge on toistaiseksi rajattu toimimaan vain Euroopan League of Legends palvelimien käyttäjille. Yksi sovelluksen jatkokehitysmahdollisuus olisikin saada palvelu toimimaan Euroopan ulkopuolisille käyttäjille. Esimerkiksi palvelun tarjoaminen Kiinan tai Etelä-Korean käyttäjille toisi suuren määrän uusia käyttäjiä palvelullemme. Sovelluksemme julkaisuversio on englanninkielinen. Englannin kielellä saavutamme suuren käyttäjäkunnan esimerkiksi Euroopassa ja Pohjois-Amerikassa mutta maanosissa, joissa englannin kielen asema on heikko, joutuisimme mahdollisesti tarjoamaan palvelua paikallisilla kielillä. Esimerkiksi jos haluaisimme laajentaa palveluamme Aasiaan, joutuisimme lokalisoimaan palvelun käyttöliittymän ainakin kiinaksi,

koreaksi ja mahdollisesti japaniksi suurimman mahdollisen käyttäjäryhmän tavoittamiseksi. Lisäksi joutuisimme pystyttämään sovelluksen paikallisille palvelimille parhaan mahdollisen käyttäjäkokemuksen takaamiseksi.

Toinen sovelluksemme jatkokehittämismahdollisuus on kehittää sovellus seuraamaan myös muiden pelimuotojen otteluita. ARAM Challenge on toistaiseksi rajattu seuraamaan vain käyttäjien pelaamia ARAM-pelimuodon otteluita. ARAM on League of Legendsin toisiksi suosituin pelimuoto ja sovelluksemme voisi helposti kehittää seuraamaan myös muiden pelimuotoja otteluita. ARAM-pelimuodosta poiketen muissa pelimuodoissa pelaaja valitsee itse ennen ottelun alkua hahmon, jolla haluaa ottelun aikana pelata, eikä pelihahmoa arvota satunnaisesti. ARAM Challengen “haaste” voittaa vähintään yksi ottelu pelin jokaisella pelihahmolla toimisi siis normaalisti, mutta ottelut pelattaisiin suuremmalla pelialueella ja ne kestäisivät keskimääräisesti pidempään.

Sovelluksen julkaisua harkitessamme tutustuimme myös kevyesti Amazonin AWS-pilvipalveluihin yhtenä mahdollisena verkkoisännöintiratkaisuna. Emme kuitenkaan aloittaneet täysin ennestään tuntemattoman palvelun käyttöä sovelluksen alkuperäisessä julkaisuvaiheessa varmistaaksemme, että ehdimme julkaista sovelluksen raportin valmistumiseksi. Myöhemmässä sovelluksen jatkokehitysvaiheessa tutkimme mahdollista julkaisua pilveen.

Lähteet

Atlassian 2023. What is scrum? Luettavissa: https://www.atlassian.com/agile/scrum. Luettu: 24.05.2023.

Amazon AWS 2023. Introduction to AWS. Luettavissa: https://aws.amazon.com/what-is-aws/?nc2=h_ql_le_int. Luettu: 25.05.2023.

Baumgartl, R. 2023. Multiplayer Games with The Highest Player Count. Luettavissa: https://www.thegamer.com/multiplayer-games-with-large-player-base/. Luettu: 25.05.2023.

Datastax. Installing Oracle JRE or JDK 8 on Debian or Ubuntu Systems. Luettavissa: https://docs.datastax.com/en/jdk-install/doc/jdk-install/installOracleJdkDeb.html. Luettu: 25.05.2023.

DigitalOcean. Luettavissa. https://www.digitalocean.com/. Luettu: 25.05.2023.

IBM 2023. What is Java Spring Boot? Luettavissa: https://www.ibm.com/topics/java-spring-boot. Luettu: 23.05.2023.

Ise 2023. Back-end määritelmä. Luettavissa: https://ise.fi/sanastoa/backend/. Luettu 23.05.2023.

Ise 2023. Front-end määritelmä. Luettavissa: https://ise.fi/sanastoa/frontend/. Luettu 23.05.2023.

Kamberovic, R. 2023. League of Legends Player Count: Here Are the Stats. Luettavissa: https://riftfeed.gg/more/player-count. Luettu: 25.05.2023

Namecheap. Luettavissa: https://www.namecheap.com/. Luettu: 25.05.2023.

No-IP. Luettavissa. https://noip.com. Luettu: 25.05.2023.

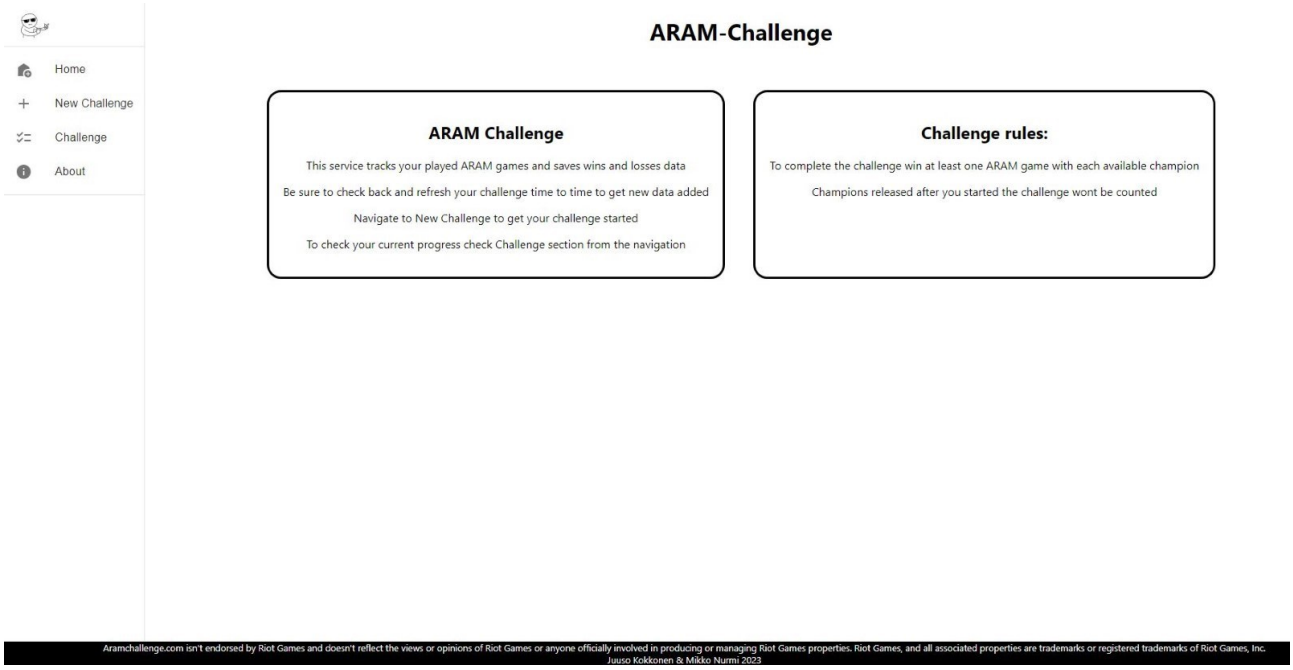
Royal, M. Self-Hosting-Guide. Luettavissa: https://github.com/mikeroyal/Self-Hosting-Guide. Luettu 25.05.2023.

Sheldon, R. & Denman, J. 2022. Definition Node.js (Node). Luettavissa: https://www.tech-target.com/whatis/definition/Nodejs. Luettu: 23.05.2023.

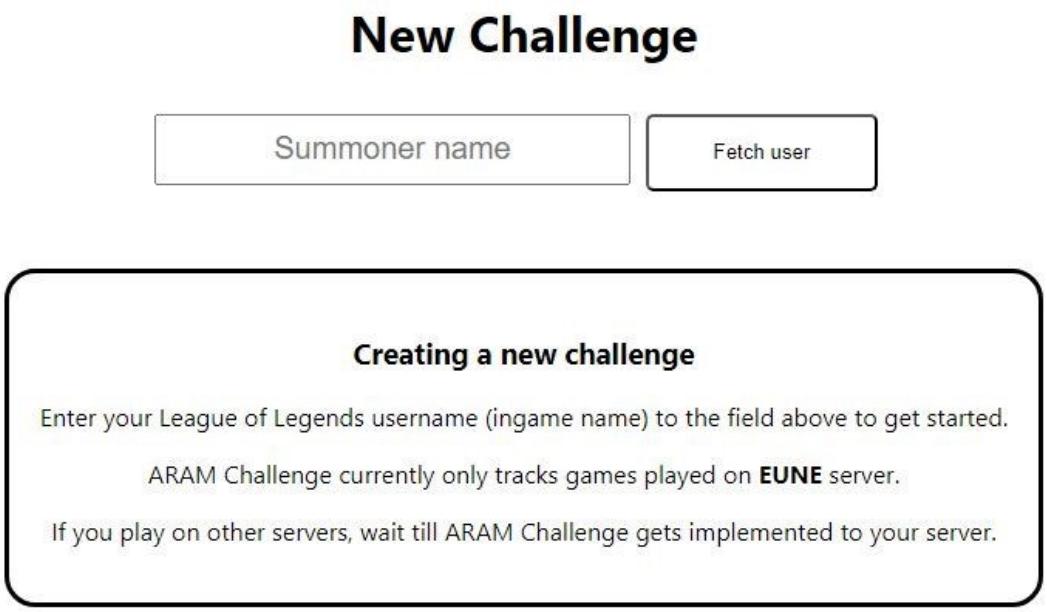
Tuomanen, I. 2018. Pienyrityksen verkkokaupan perustaminen ja ylläpito. Opinnäytetyö. Insinööri (AMK), mediatekniikan koulutusohjelma. Jyväskylän Ammattikorkeakoulu Luettavissa: https://www.theseus.fi/bitstream/handle/10024/151566/Opinnaytetyo_Ilkka_Tuomanen_2018.pdf?sequence=1. Luettu: 25.05.2023.

Liitteet

Liite 1. ARAM Challengen etusivu



Liite 2. Uuden haasteen luomisen sivu



Liite 3. Uuden haasteen luominen ja haasteen id:n palauttaminen käyttäjälle