Ville Lahtinen

GameSparks-pilvipalvelu

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Ohjelmistotekniikka

Insinöörityö

11.4.2016

Metropolia

Tekijä(t) Otsikko Sivumäärä Aika	Ville Lahtinen GameSparks-pilvipalvelu 32 sivua 11.4.2016
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Tietotekniikka
Suuntautumisvaihtoehto	Ohjelmistotekniikka
Ohjaaja(t)	Lehtori Miikka Mäki-Uuro

GameSparks on erityisesti mobiilipelejä varten luotu pilvipalvelu, jolle kehittäjät pystyvät rakentamaan kaikki pelinsä palvelinpuolen toiminnot. Työn tavoitteena on tutustuttaa lukija GameSparks-pilvipalveluun ja siihen, minkälaisia mahdollisuuksia se tarjoaa kehittäjälle.

Insinöörityö on jaettu kolmeen päälukuun. Ensimmäisessä luvussa tutustutaan yleisesti pilvipalveluihin ja siihen, miten ne jaotellaan tarjoamiensa palveluiden ja jakelumallien perusteella eri kategorioihin. Työn toisessa luvussa tutustutaan GameSparksiin ja sen pääominaisuuksiin. Viimeisessä luvussa on esitelty muutama havainnollistava esimerkki siitä, minkälaisia ratkaisuja GameSparksilla on mahdollista toteuttaa.

GameSparks osoittautui ominaisuuksiensa ja helppokäyttöisyytensä puolesta tehokkaaksi ratkaisuksi pelin backendin luomista varten. Täydellinen se ei kuitenkaan ole vaan parannettavaa löytyy usealta osa-alueelta, erityisesti dokumentointi on joidenkin ominaisuuksien osalta hyvin puutteellinen. Puutteistaan huolimatta on GameSparksia helppo suositella, jos etsii toimivaa ratkaisua pelin backendin luomiseen.

Avainsanat	GameSparks, pilvipalvelut, backend as a service

Author(s) Title Number of Pages Date	Ville Lahtinen GameSparks Cloud Service 32 pages 11 April 2016
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Specialisation option	Software Engineering
Instructor(s)	Miikka Mäki-Uuro, Senior Lecturer

GameSparks is a cloud-based development platform on which developers can build all the server-side components of their game. The aim of thesis is to introduce the reader to GameSparks and the opportunities it offers to game developers.

The thesis is divided into three main chapters. The first chapter serves as an introduction to cloud services in general and how they are classified into different categories by the services they offer and their development models. The second chapter focuses on GameSparks and its main functions. The last chapter of the thesis introduces a few demonstrative examples of implementations that can be built by using GameSparks.

GameSparks proved to be an effective yet somewhat flawed solution for creating backend functionality for a game. While its functions are varied and easy to use, they are often lacking in documentation leaving it up to the developer to figure how the function exactly works. Despite its shortcomings GameSparks is a noteworthy competitor when looking for a platform on which to build backend for a game.

Keywords	Cloud services, GameSparks, backend as as service

Sisällys

Lyhenteet

API	Application Programming Interface. Ohjelmointirajapinta, jonka kautta ohjelmat pystyvät vaihtamaan tietoa keskenään.
BaaS	Backend as a Service. Backendin tarjoaminen palveluna.
IaaS	Infrastructure as a Service. Infrastruktuurin tarjoaminen palveluna.
JSON	JavaScript Object Notation. Avoimen standardin tiedostomuoto, jota käytetään tiedonvälitykseen.
PaaS	Platform as a Service. Sovellusalustan tarjoaminen palveluna.
SDK	Software Development Kit. Sovelluskehitykseen tarkoitettuja työkaluja tietylle alustalle
SaaS	Software as a Service. Sovelluksen tarjoaminen palveluna.

1 Johdanto

Erilaisten verkkoon kytkettävien mobiililaitteiden yleistyminen aiheutti päänsärkyä mobiilikehittäjille, koska sovelluksen tulisi toimia sujuvasti laitteesta ja alustasta riippumatta. Etsittiin ratkaisuja, jolla sovelluksen kehitykseen käytettävä työaika voitaisiin maksimoida. Yhtenä vaihtoehtona oli backendin, eli sovelluksen taustapalveluiden, kuten tietokantojen ja palvelimen, ulkoistaminen jollekin kolmannelle osapuolelle. Sen aikaisilla pilvipalveluilla ei tämä kuitenkaan luontevasti onnistunut. Vuonna 2011 asiaan saatiin kuitenkin ratkaisu, kun maailman ensimmäinen BaaS-tyypin pilvipalvelu Kinvey avattiin. [1.]

BaaS-tyypin pilvipalvelut tarjoavat kehittäjille yksinkertaisen tavan sovelluksensa backendin luomiseen. Palvelussa kehittäjät linkittävät sovelluksensa palveluun, joka tarjoaa kehittäjälle pilvestä tallennustilaa sovelluksen tietojen hallinnoimista varten. Palvelut tarjoavat myös useita ominaisuuksia kehittäjän käyttöön kuten käyttäjien hallinnan ja integraation sosiaalisten medioiden kanssa. Palvelun ja sovelluksen väliset toiminnot rakennetaan palveluntarjoajan tarjoamilla työkaluilla. Työkalut on räätälöity palvelua varten, ja ne ovat saatavilla usealle eri alustalle. [2.]

Useimmat BaaS-tyypin pilvipalvelut käyttävät palvelunsa rahoittamiseen niin sanottua freemium-hinnoittelumallia. Freemium-hinnoittelumallilla tarkoitetaan jotakin tuotetta tai palvelua, joka on saatavilla ilmaiseksi, mutta rahaa veloitetaan osista sen ominaisuuksista tai virtuaalisista tuotteista. BaaS-tyypin palveluissa tämä ilmenee esimerkiksi siten, että palvelu tarjoaa kehittäjän sovellukselle tuen 1000 käyttäjälle kuussa. Jokaisesta käyttäjästä, joka tämän rajan ylityksen jälkeen sovellukseen kirjautuu, peritään kehittäjältä pieni rahasumma. Osa palveluista tarjoaa kuitenkin mahdollisuuden kiinteään kuukausittaiseen maksuun, jotta kulut olisivat ennakoitavissa.

Insinöörityön pääaiheena on BaaS-pilvipalvelu GameSparks ja sillä rakennetut ratkaisut. Työ on jaettu kolmeen eri lukuun. Työn ensimmäisessä luvussa tutustutaan yleisesti pilvipalveluihin ja siihen, millaisiin eri tyyppeihin ja jakelumalleihin ne voidaan jaotella. Toisessa luvussa tutustutaan GameSparksiin ja sen ominaisuuksiin. Viimeisessä luvussa esitellään muutama GameSparksilla rakennettu ratkaisu.

2 Pilvipalvelut

Pilvipalvelu on laaja käsite, joka kattaa laajan valikoiman eri palveluja. Yksinkertaisesti selitettynä pilvipalvelu on verkossa, eli pilvessä tarjottava palvelu. Palvelu on siis riippumaton käyttäjän sijainnista, koska kaikki tarvittavat tiedot ja suoritettavat toiminnot suoritetaan verkossa, eli riittää, että käyttäjällä on internetyhteys. [3.]

Pilvipalveluiden käyttöönotto tarjoaa ilmeisiä etuja. Pilven käyttöönotto on useimmiten nopeampaa, halvempaa ja joustavampaa kuin omien järjestelmien rakentaminen. Etujen mukana tulee kuitenkin myös riskejä. Käyttäjien tiedot sijaitsevat palveluntarjoajien palvelimilla, joiden sijainnista käyttäjillä ei ole tietoa. Tiedot ovat alttiina väärinkäytöksille, jos palveluntarjoajan tietoturva ei ole kunnossa. [4.]

2.1 Palvelutyypit

Pilvipalvelut luokitellaan eri palvelutyyppeihin niiden tarjoamien palveluiden perusteella. Suurin osa palveluista jakautuu kolmeen pääkategoriaan, jotka ovat: Software as a Service (SaaS), Platform as a Service (PaaS) sekä Infrastructure as a Service (IaaS). [5.]

Vaikka suurin osa palveluista lukeutuukin näihin palvelutyyppeihin, on olemassa vähemmän tunnettuja ja käytössä olevia palvelutyyppejä, kuten Marketplace as a Service (MaaS) ja Backend as a Service (BaaS).

2.1.1 Software as a Service

Software as a Service -mallin pilvipalveluilla tarkoitetaan sellaisia pilvipalveluita, jotka tarjoavat perinteisiä sovelluksia asiakkaiden käyttöön internetin välityksellä. Toisin kuin perinteisten ohjelmistojen kanssa, jotka käyttäjä joutuu asentamaan itse päätelaitteelleen, sovellusta pääsee käyttämään internetselaimen kautta. Käyttäjä ei joudu siis huolehtimaan sovelluksen asentamisesta, päivittämisestä tai sen toimivuudesta, vaan nämä kaikki ovat palveluntarjoajan vastuulla. [6.]

Useimmat palvelut eivät velota käyttäjältä mitään tiettyä kiinteää rahasummaa, vaan kuukausimaksun, jonka hinta mukautuu käyttäjän valitsemien ominaisuuksia mukaan. Käyttäjä ei siis maksa itse ohjelmasta vaan sen käyttöoikeudesta. [6.]

Tunnetuimpia esimerkkejä SaaS-tyypin palveluista ovat esimerkiksi Google Apps, Salesforce.com ja Microsoft Office 365.

2.1.2 Platform as a Service

PaaS eli sovellusalusta palveluna tarkoittaa palvelualustan ulkoistamista. Palveluntarjoaja tarjoaa käyttäjien käytettäväksi alustan, jolla kehittää, ajaa ja hallita web-sovelluksia. PaaS-palveluntarjoaja ylläpitää laitteistot ja ohjelmistot omassa infrastruktuurissaan. Käyttäjien ei siis tarvitse asentaa erillisiä laitteistoja tai ohjelmistoja omassa päässään aloittaakseen uuden sovelluksen kehittämistä. [7.]

Palveluun luotua sovellusta käyttäjät pääsevät käyttämään selaimen kautta. Palvelun ylläpitäjät veloittavat kehittäjiltä maksua sovelluksen käyttäjämäärien mukaan. Jotkut palvelut tarjoavat kuitenkin mahdollisuuden tilata palvelun itselleen kuukausittaisella maksulla. [7.]

PaaS-palvelun käyttöönotto tuo kehittäjille säästöjä laitteiston sekä ohjelmistojen osalta ja jättää enemmän aikaa sovelluksen kehitykseen, kun näistä ei tarvitse huolehtia. Toisaalta sovelluksen kehittämistä rajoittavat palvelun tarjoamat työkalut. Mitä jos palveluntarjoaja lopettaakin jonkun ohjelmointikielen tukemisen? Tässä tapauksessa kehittäjän tulee kääntää sovellus toiselle ohjelmointikielelle tai vaihtaa palveluntarjoajaa. [7.]

Tunnettuja PaaS-palveluntarjoajia ovat mm. Heroku, Amazon Web Services, Google App Engine ja Force.com.

2.1.3 Infrastructure as a Service

IaaS-tyypin pilvipalvelut tarjoavat asiakkaiden käyttöön samat ominaisuudet kuin fyysinen konesalikin, mutta vain virtualisoituna. Virtuaalisena tarjottavat resurssit ovat ominaisuuksiltaan samanlaisia kuin fyysiset resurssit, mutta niiden hallinnointi on huomattavasti helpompaa ja nopeampaa. [8.]

IaaS-palveluntarjoaja ylläpitää virtuaalista konesalia, mutta ei rajoita tai ohjaa palveluidensa käyttöä. Palvelun käyttäjä voi muokata vuokraamastaan virtuaalikoneesta omanlaisensa, aina käyttöliittymästä lähtien. [8.]

Toisin kuin SaaS- ja PaaS-tyyppisten pilvipalveluiden kanssa, ei IaaS-tyypin pilvipalveluista veloiteta kiinteää kuukausimaksua. Palvelun käytöstä maksettava summa kasvaa tunneittain käytettyjen resurssien mukaan, ja se veloitetaan kuukausittain. [8.]

Tunnetuimpia IaaS-palveluita ovat Microsoft Azure, Amazon Web Services ja Google Compute Engine.

2.2 Jakelumallit

Pilvipalvelut lajitellaan laitteiston omistajuuden ja hallinnoijien perusteella eri jakelumalleihin. Nämä jakelumallit ovat julkinen, yksityinen, yhteisöllinen ja hybridi-pilvi.

Julkinen pilvi on julkisesti käytettävissä ja sen omistaa kolmannen osapuolen palveluntarjoaja. Palveluntarjoaja omistaa pilven ja tarjoaa palvelunsa asiakkaiden käyttöön joko maksua vastaan tai ilmaiseksi, saaden tulonsa mainosten kautta. Palveluntarjoaja on vastuussa palvelun laitteistojen ja sovelluksien hallinnasta. [9.]

Yhteisöllinen pilvi on hyvin samankaltainen julkisen pilven kanssa, mutta siihen on pääsy vain jonkun tietyn yhteisön jäsenillä. Pilvi voi olla tämän yhteisön jäsenten yhteisomistuksessa tai jokin palveluntarjoaja tarjoaa yhteisölle käyttöön julkisen pilven, jonka käyttö on rajattu yhteisön jäsenille. Yhteisön jäsenet yhdessä huolehtivat pilven toiminnasta. [10.]

Yksityinen pilvi on yritykseen käyttöön tarkoitettu yksityinen pilvi. Useimmiten yritys käyttää pilveä IT-resurssien keskittämiseen yrityksen eri toimistojen ja osastojen välillä. Pilvi voi olla itse yrityksen hallitsema ja omistama, tai se voi olla hankittu yrityksen käyttöön palveluntarjoajalta. [11.]

Hybridi pilvi on aikaisemmin mainittujen mallien yhdistelmä. Esimerkiksi pilvipalvelua käyttävä yritys säilyttää sille tärkeitä tietoja yksityisessä pilvessä ja muita tietoja julkisessa pilvessä. [12.]

3 Gamesparks

GameSparks on vuoden 2012 elokuussa avattu BaaS-tyypin pilvipalvelu, joka on luotu erityisesti mobiilipelejä varten. Pelinkehittäjät pystyvät rakentamaan kaikki palvelinpuolen toiminnot Gamesparksin tarjoamalla alustalle, eikä tarvetta hankkia omia servereitä tai tietokantoja synny. Palvelulla on asiakkaina yli 200 eri studiota ympäri maailmaa, ja se voitti vuonna 2014 parhaan teknisen innovaation palkinnon. [13.]

GameSparks tarjoaa käyttöön useita SDK:ta, jotka on räätälöity eri pelimoottoreille ja alustoille. GameSparks tukee mm. Unitya ja Unreal Enginea.

3.1 Hinnoittelu

GameSparks tarjoaa kolme eri vaihtoehtoa palvelunsa käyttämiseen. Näistä ensimmäinen on vapaa lisenssi, joka toimii tavallaan palvelun kokeiluversiona. Vapaan lisenssin käyttöönotto ei maksa käyttäjälle mitään, vaan vaatii pelkän tunnuksen luonnin palveluun. Vapaa lisenssi tarjoaa käyttöön kaikki GameSparksin ominaisuudet sekä tuen 1000 käyttäjälle kuussa.

Toisena vaihtoehtona GameSparks tarjoaa yrityspakkauksia. Nämä ovat erityisesti tarkoitettu jo luoduille ja pidemmän aikaa toimineille yrityksille. Yrityspakkausta ei voi suoraan tilata, vaan yritysten tulee ottaa GameSparksiin yhteys ja ilmoittaa halukkuutensa ottaa palvelu käyttöön heidän projektissaan. Yhteydenoton jälkeen räätälöidään pakkaus yrityksen projektin tarpeiden mukaan. Lopullinen hinta muokkautuu valittujen ominaisuuksien ja pelin kuukausittaisen käyttäjämäärien mukaan.

Viimeisenä vaihtoehtona tarjotaan erityisesti opiskelijoille ja indie-kehittäjille suunnattua pakettia. Pakettia ei siis tarjota aivan kenelle tahansa, vaan sen voi tilata vain, jos on pelinkehityksen kurssia käyvä opiskelija tai kurssia pitävä opettaja, maksimissaan kolmen hengen indie-peliyritys tai pelinkehityksen harrastaja.

Paketin tilaamisesta ei tarvitse etukäteen maksaa mitään. Maksua ruvetaan perimään vasta, kun pelin kuukausittaisten käyttäjien lukumäärä ylittää 100 000 käyttäjää. Tämän jälkeen ruvetaan perimään 2 senttiä per pelaaja, joka peliin tuon rajan ylityksen jälkeen

kirjautuu. Käytännössä siis GameSparksin käyttäminen on ilmaista, ellei pelistä tule hyvin suosittua. [14.]

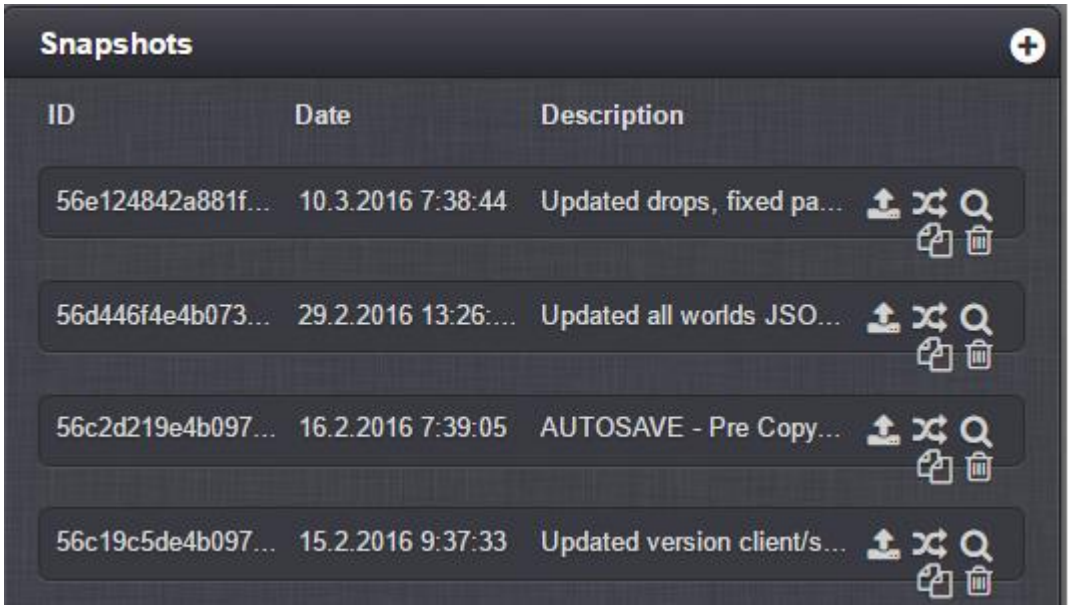
3.2 GameSparks-portaalin käyttöönotto

Kaikki pelin kehittämiseen ja hallinnoimiseen liittyvät asiat hoidetaan GameSparks-portaalin kautta. Ensimmäisen kerran portaalin kirjautuessaan käyttäjän pitää luoda kehitysympäristö ensimmäiselle pelilleen. Kehitysympäristön luonti tapahtuu yksinkertaisesti. Käyttäjän tulee antaa pelilleen vain nimi ja kuvaus, jonka jälkeen kehitysympäristö on valmis. Tulee ottaa kuitenkin huomioon, että kehitysympäristöt voivat olla sidoksissa vain yhteen peliin ja käyttäjän tulee luoda uusi kehitysympäristö jokaista peliä kohden.

Kun pelin kehitysympäristö on luotu, tulee käyttäjän ladata GameSparksin SDK ja plugin haluamalleen kehitysalustalle. Tällä hetkellä GameSparksin tukemat kehitysalustat ovat Unity, Unreal Engine ja ActionScript. Kehitysympäristö ja -alusta linkitetään toisiinsa käyttäen API-keyta ja API-secretia. Nämä kaksi ovat kehitysympäristön luonnin yhteydessä generoituja merkkijonoja, joita käytetään kehitysympäristön ja alustan linkittämiseen keskenään.

Kun peliympäristö luodaan, sillä kehitettävä peli asetetaan automaattisesti preview-tilaan. Preview-tila on tarkoitettu kehityksessä olevia pelejä varten. Preview-tilan päätarkoituksena on kehityksessä olevan pelin testaaminen pienillä pelaajamäärillä. Preview-tilassa pelissä pystyy olemaan samanaikaisesti kirjautuneena vain 100 pelaajaa.

Kun peli on valmis julkaistavaksi, tulee siitä aluksi luoda uusi Snapshot, joka luo tallennuspisteen pelin sen hetkisestä tilasta. GameSparks pitää listaa luoduista SnapShoteista (kts. kuva 1). Kehittäjä voi tarkastella SnapShotissa tapahtuneita muutoksia ja palauttaa pelin halutessaan aikaisemman SnapShotin tilaan, jos esimerkiksi uuden päivityksen kanssa esiintyy ongelmia. [15.]



Kuva 1. Lista luoduista snapshoteista

3.3 Eventit

Eventit ovat tapahtumia, jotka toimivat rajapintana pelin ja GameSparks-alustan välillä. Tapahtumat on luokiteltu kahteen eri luokkaan: System Events ja User Events.

System Events -tapahtumat ovat GameSparksin luomia tapahtumia, jotka ovat kaikkien kehittäjien kutsuttavissa. Nämä tapahtumat eivät ole kehittäjien muokattavissa, eivätkä ne näy käyttäjien luomien tapahtumien listassa. Tapahtumat liittyvät GameSparksin valmiiksi rakentamiin ratkaisuihin. Esimerkkinä FindChallengeRequest-tapahtuma etsii ja palauttaa pelaajalle kaikki tälle lähetetyt haasteet. [16.]

User Events -tapahtumat ovat kehittäjien itsensä luomia tapahtumia, jotka ovat sidoksissa kehittäjän luomaan peliin. Tällaisissa tapahtumissa käsitellään useimmiten peliin liittyvää logiikkaa ja tietorakenteita.

Eventin luominen tapahtuu GameSparks-palvelussa. Eventin luodakseen tarvitaan täyttää seuraavat kentät:

- Short Code
    - määritetään millä nimellä Eventiä pystytään kutsumaan API:n kautta

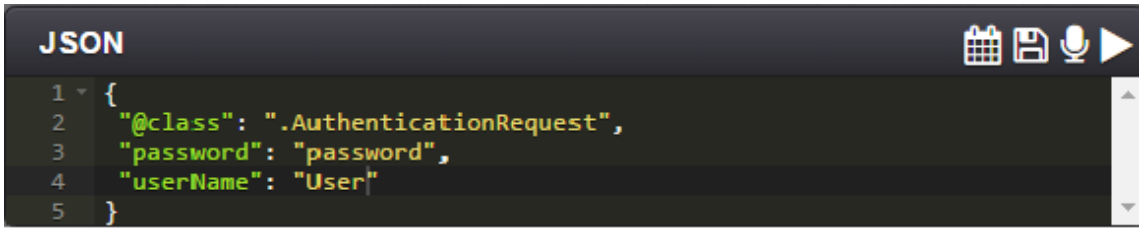
- Name
    - eventin nimi, käytetään Eventin tunnistamiseen ja etsimiseen GameSparks-alustassa
- Description
    - lyhyt kuvaus Eventin toiminnasta.

Event voi vastaanottaa myös yhden tai useamman parametrin. Parametreille tulee antaa seuraavat tiedot:

- Name
    - parametrin nimi
- Short Code
    - tätä nimeä käytetään koodissa, kun tälle parametrille halutaan asettaa arvo
- Data Type
    - parametrin datatyyppi, parametri voi olla tyypiltään joko int, string tai JSON
- Default Value
    - parametrin oletusarvo
- Default Calc
    - parametrin käyttötarkoitus. Esim. parametria tullaa käyttämään koodissa.

Luotujen tapahtumien logiikka ja kaikki muu pelin palvelinpuolen toimintoihin liittyvä logiikka kirjoitetaan CloudCodessa. CloudCode on rakennettu JavaScript SDK:n päälle, eli kaiken CloudCodessa kirjoitettavan koodin tulee noudattaa JavaScriptin syntaksia.

Käyttäjän luomien ja GameSparksin omien tapahtumien toimintaa pystyy testaamaan Test Harness -välilehdellä. Test Harness mahdollistaa API-pyyntöjen lähettämisen GameSparks-alustalle JSON-muodossa (kts. kuva 2). Käyttäjä pystyy asettamaan pyynnölle haluamansa parametrit.



Kuva 2. JSON-lähete

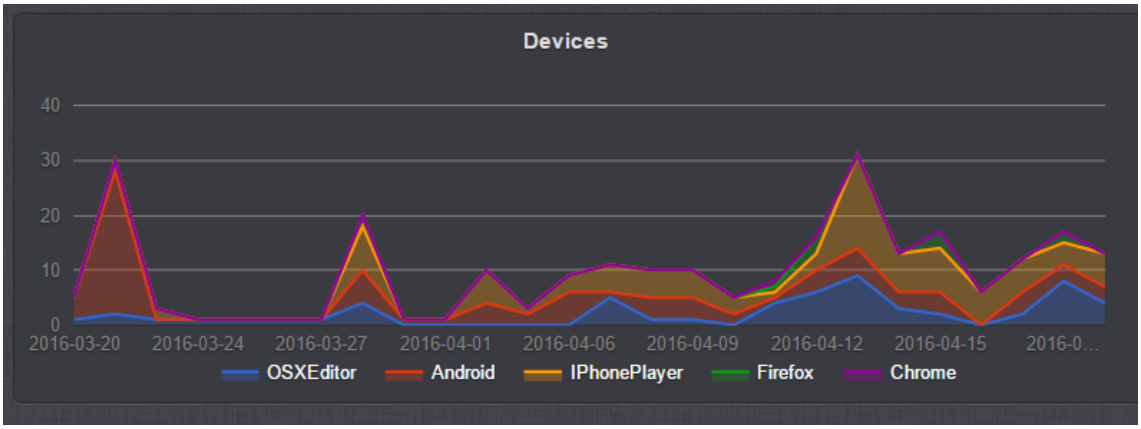
Pyynnön lähetettyään käyttäjälle aukeaa uusi ikkuna, jossa näkyy suoritettavan tapahtuman koodi. Käyttäjällä on mahdollisuus käydä tapahtuman koodi läpi rivi kerrallaan tai hypätä suoraan tarkastelemaan palvelimen pyyntöön palauttamaa vastausta. Palvelimen palauttamassa vastauksessa pelaajan lähettämä pyyntö on merkitty vihreällä tekstillä ja palvelimen lähettämä vastaus sinisellä tekstillä (kts. kuva 3).

Kuva 3. Palvelimelta pyyntöön saatu vastaus

3.4 Analytics

Analytics-välilehdellä kehittäjä pystyy seuraamaan pelinsä kasvua. Välilehdelle on kerätty tietoa pelin käyttäjistä ja luotu näistä havainnollistavia kaavioita. Tietoa löytyy mm. siitä, kuinka monta uutta käyttäjää peliin on rekisteröitynyt kuukauden aikana ja kuinka

peliä pelaavat käyttäjät jakautuvat laiteittain (kts. kuva 4). Käyttäjä pystyy itse määrittelemään ajankohdan ja pelin version, jolta hän haluaa tietoja tarkastella. Käyttäjällä on myös mahdollisuus luoda omia kaavioitaan, käyttäen GameSparksin keräämiä tietoja. [17.]



Kuva 4. Analytics-välilehti

3.5 Tietojen hallinta

Pelin tietoja pystyy hallitsemaan NoSQL-välilehdellä. Välilehdellä käyttäjän on mahdollista tutkia kaikkea pelistä kerättyä dataa ja luoda uusia kokoelmia. Tämä toimii siis pelin tietokantana ja sisältää kaikki ominaisuudet mitä tietokannalta odottaisi. Tiedon hallintaan käytetään MongoDB:ta ja tiedon hallinnointiin liittyvät kyselyt suoritetaan NoSQL-notaatioilla. Käyttäjällä on myös mahdollista tallentaa haluamansa tiedot JSON-muodossa muuhun käyttöön. [18.]

3.6 Sosiaaliset ominaisuudet

GameSparks sisältää useita sosiaalisia ominaisuuksia, jotka ovat pääasiassa tarkoitettu pelaajien väliseen kanssakäymiseen.

Joukkueita käytetään pelaajien ryhmittämiseen keskenään. Määrittääkseen, millaisia joukkueita pelissä pystytään luomaan, täytyy kehittäjän luoda näistä mallit. Joukkueita voi olla kahdenlaisia pysyviä ja hetkellisiä. Pysyvät joukkueet ovat pelin käyttäjien luomia ja näiden johtaja toimii yksi tai useampi käyttäjä. Joukkueen johtajilla on mahdollisuus

lisätä ja poistaa muita pelaajia joukkueesta. Hetkelliset joukkueet eivät ole pysyviä, vaan ne luodaan jotain tiettyä tarkoitusta varten. Esimerkiksi moninpelissä, jossa kaksi joukkuetta ottelevat toisiaan vastaan, luodaan kaksi hetkellistä joukkuetta, johon pelaajat kiinnitetään. Ottelun päätyttyä hetkelliset joukkueet poistetaan.

Kehittäjällä on mahdollisuus liittää pelinsä tapahtumiin tulostaulukkoja. Tulostaulukot pitävät tietoa yllä tuloksista ja järjestävät nämä kehittäjän haluamaan järjestykseen. Kehittäjän on mahdollista asettaa liittää palkintoja tulostaulukkoon, jotka ansaitsevat tulostaulukossa tietyillä sijoilla olevat pelaajat.

Kehittäjä pystyy määrittelemään haasteita, joita pelaajat voivat lähettää toisilleen. Haasteet voivat olla yhden tai useamman pelaajan välisiä ja niille voi asettaa aloitus- ja lopetusajankohdan. Haasteisiin voidaan liittää myös panos, minkä haasteen voittaja itselleen saa.

Kehittäjillä on mahdollisuus luoda peliin virtuaalisia tavaroita, joita pelaajat voivat ansaita tai ostaa. Tavarat on mahdollistaa linkittää pelin ulkopuolisiin kauppoihin, kuten AppStoreen, ja täten on mahdollista myydä virtuaalisia tavaroita oikeaa rahaa vastaan.

GameSparks on sidottu myös vahvasti useihin sosiaalisiin medioihin. Kehittäjillä onkin mahdollisuus liittää pelinsä esimerkiksi Facebookiin, joka mahdollistaa pelaajien kirjautumisen peliin sen kautta.

4 GameSparksilla toteutetut ominaisuudet

Insinöörityön kirjoittaja toimi 2015 harjoittelijana Marimo Games Oy -nimisessä pelialan mikroyrityksessä. Yritys oli perustettu vuoden 2014 loppupuolella, ja se kehitti ensimmäistä peliään. Työn kirjoittajan vastuulla oli pelin toimintojen rakentaminen GameSparksin puolella. Tässä luvussa tullaan esittelemään muutama ratkaisu, jotka työn kirjoittaja GameSparksilla harjoittelun aikana kehitti.

4.1 Kaverilistat

Gamesparks sisältää valmiit ratkaisut Facebook- ja Google+ kavereiden lisäämiseen käyttäjille. Tavoitteena oli kuitenkin, että pelaajilla olisi myös mahdollisuus lisätä sosiaalisen median ulkopuolisia käyttäjiä kavereiksi (kts kuva 5).

Kuva 5. Kaverinpyynnön lähetystä havainnollistava kaavakuva

4.1.1 Kaverilistan luominen

Gamesparks ei sisällä valmiiksi rakennettua kaverilistaa, mutta se sisältää tuen pelaajien välisille joukkueille. Käyttäjän rekisteröityessä luodaan uusi joukkue, jonka jäseneksi ja johtajaksi rekisteröitynyt käyttäjä asetetaan. Johtajan asemassa käyttäjällä on oikeus lisätä ja poistaa käyttäjiä joukkueesta, eli joukkue toimii kaverilistan korvikkeena. Jotta joukkue ja sen omistava pelaaja eivät sekoittuisi keskenään, kun tietokannasta haetaan tietoja id:n perusteella, luodaan joukkueelle uniikki id (kts. kuva 6).

```
var friendsList = Spark.sendRequest({
     "@class": ".CreateTeamRequest",
     "teamId": Spark.getPlayer().getPlayerId() + "friends",
     "teamType": "friendsList",
     "teamName": "Friends List"
});
```

Kuva 6. Rekisteröinnin yhteydessä ajettava koodi, joka luo käyttäjälle joukkueen

4.1.2 Muiden pelaajien haku

Jotta käyttäjät pystyisivät lisäämään muita pelaajia heidän kaverilistaansa, nämä täytyisi myös löytää jollain tavalla. GameSparks ei itsestään kuitenkaan ylläpidä listaa peliin rekisteröityneistä käyttäjistä. Käyttäjän rekisteröinnin yhteydessä haetaan kokoelma rekisteröityneistä pelaajista muuttujaan, jonka jälkeen pelaaja lisätään tähän kokoelmaan. Jos kokoelmaa ei ole olemassa sitä haettaessa, sellainen luodaan ja asetetaan pelaaja tämän kokoelman ensimmäiseksi tietueeksi.

Muiden pelaajien etsimistä varten luodaan uusi tapahtuma. Tapahtuma vastaanottaa kolme parametria. Tapahtuman vastaanottamat parametrit ovat:

- displayName
    - etsittävän pelaajan nimi. Saadaan pelaajan pelin puolella antamasta syötteestä
- entryCount
    - määrittää kuinka monta pelaajaa tapahtuma maksimissaan palauttaa. Oletusarvoksi asetettu 10. Jos löydettyjen pelaajien määrä ylittää annetun oletusarvon, palautetaan ne 10 pelaajaa, joiden nimet vastasivat lähinnä pelaajan antamaa syötettä

- offset
    - ei käytetä tässä tapahtumassa varsinaisesti mihinkään, mutta funktio jota käytetään pelaajien etsimiseen, tarvitsee tämän toimiakseen.

Tapahtumassa haetaan lista kaikista rekisteröityneistä pelaajista muuttujaan. Listasta etsitään enintään 10 pelaajaa, joiden nimet ovat lähinnä pelaajan antamaa syötettä. Tämän jälkeen käydään läpi nämä löydetyt pelaajat ja poistetaan heidän tiedoistaan playerId (kts. kuva 7). Lopuksi löydettyjen pelaajien nimet lisätään uuteen tauluun, joka palautetaan pelin puolelle (kts. kuva 8).

```
var playerCollection = Spark.runtimeCollection("playerDirectory");
var playerList = playerCollection.find({playerName : {$regex : "(?i)^"
+
Spark.getData().displayName}}).limit(Spark.data.en-
tryCount).skip(Spark.data.offset);

var players = [];
while(playerList.hasNext())
{
    var player = playerList.next();
    delete player["_id"];
    players.push(player);
}

Spark.setScriptData("players", players);
```

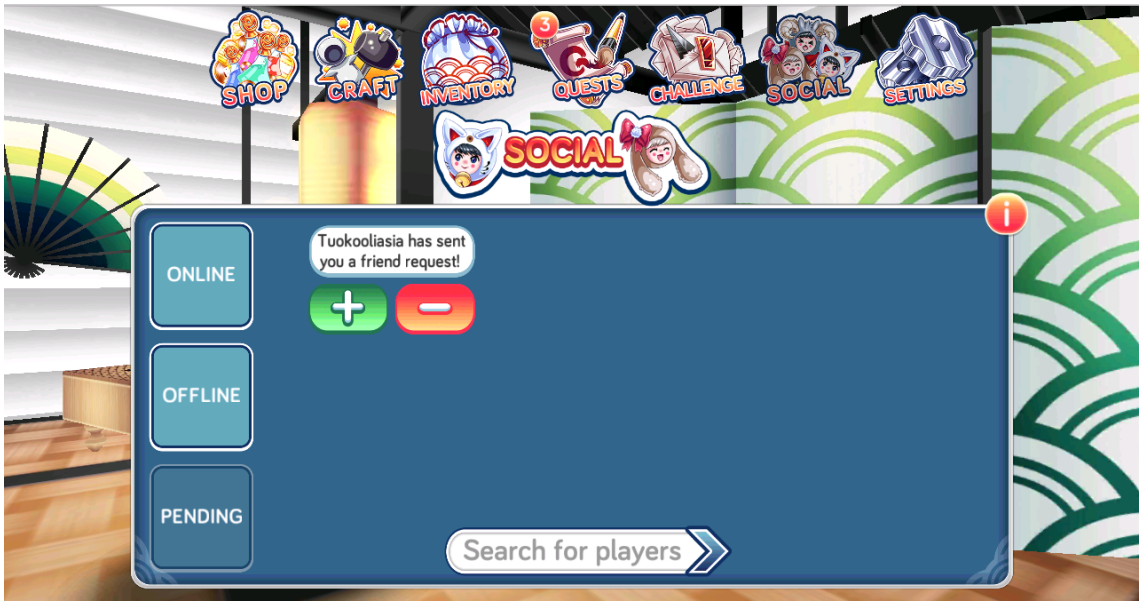
Kuva 7. Pelaajien hakuun käytettävän tapahtuman koodi

Kuva 8. Pelaajan haulla löydetyt pelaajalla

4.1.3 Kaveripyynnön lähettäminen

Kaveripyyntönä käytetään haasteita. Haasteet ovat normaalisti tarkoitettu muiden pelaajien haastamiseen, johonkin pelaajien väliseen kilpailuun. Tässä tapauksessa haastetta käytetäänkin kaveripyynnön sijaisena ja haasteen hyväksyessään pelaajat lisätään toistensa kaverilistoihin.

Aina haastetta lähetettäessä suoritetaan CreateChallengeResponse-tapahtuma. Tapahtuma on GameSparksin luoma ja on oletuksena tyhjä. Tapahtuma tietää, että jokin haaste on lähetetty, mutta se ei kuitenkaan tiedä, minkä tyypin haaste on kyseessä. Onkin siis tarkastettava, että lähetetty haaste on varmasti kaveripyyntö. Lopuksi haastajan nimi lähetetään takaisin pelin puolelle ja haastetulle pelaajalle lähetetään viesti kaveripyynnön saapumisesta (kts. kuva 9).



Kuva 9. Vastaanotettu kaveripyyntö

4.1.4 Kaveripyynnön hyväksyminen ja hylkäys

Kaveripyynnön hyväksymiseen ja hylkäykseen käytettävät tapahtumat ovat identtisiä. Molemmat vastaanottavat yhden parametrin, joka on lähetetyn kaveripyynnön id.

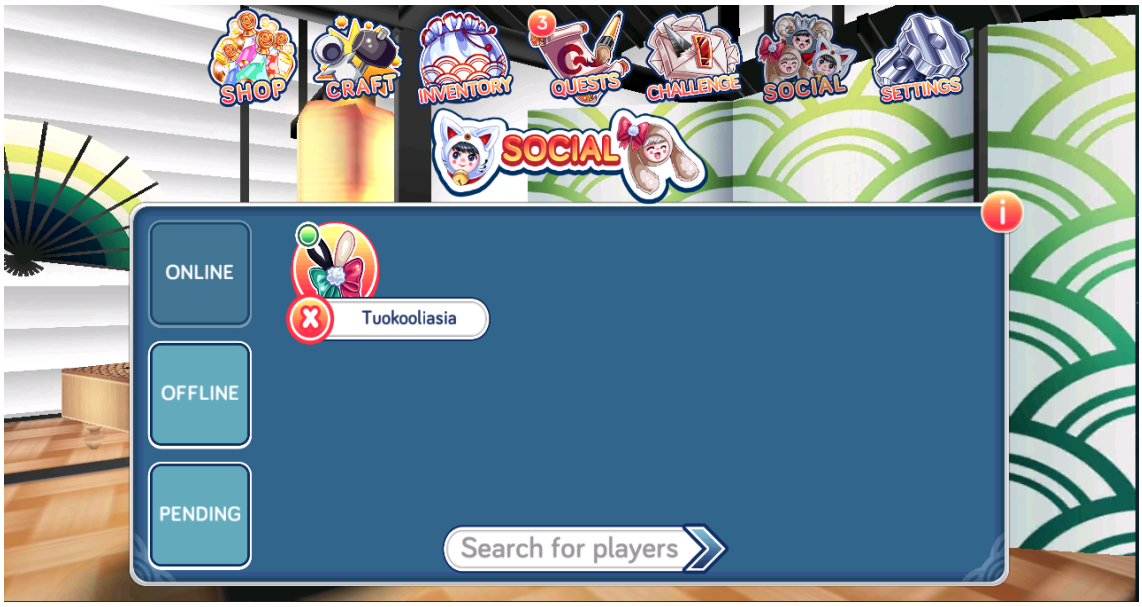
Kaveripyynnön lähettäjä ja sen vastaanottaja lisätään toistensa kaverilistoihin. Lähettäjä saa viestin, jossa kerrotaan, että hänen kaveripyyntönsä on hyväksytty. Koska haastetta ei varsinaisesti kuitenkaan haluta hyväksyä, jolloin siitä tulisi aktiivinen haaste niin kauaksi aikaa, kunnes toinen pelaajista julistettaisiin haasteen voittajaksi, vedetään haaste takaisin.

Tapauksessa, jossa kaveripyynnön vastaanottaja ei halua lisätä kaveripyynnön lähettäjää kaverikseen, kaveripyyntö eli haaste, hylätään. Useimmissa sosiaalisissa medioissa ei ole tapana ilmoittaa käyttäjälle siitä, jos hänen kaveripyyntönsä hylätään. Näin toimitaan tässäkin tapauksessa.

4.1.5 Kavereiden listaus

Kavereiden listausta varten luotu tapahtuma vastaanottaa yhden parametrin, joka on käyttäjän id. Tapahtumassa haetaan pelaajan inventaario muuttujaan. Inventaario toimii

nimensä mukaisesti varastona, jossa säilytetään kaikkia pelaajaan liittyviä tietoja. Inventaariosta haetaan pelaajan kaverilista ja lähetetään se takaisin pelin puolelle (kts. kuva 10).



Kuva 10. Kaverilista pelin puolella

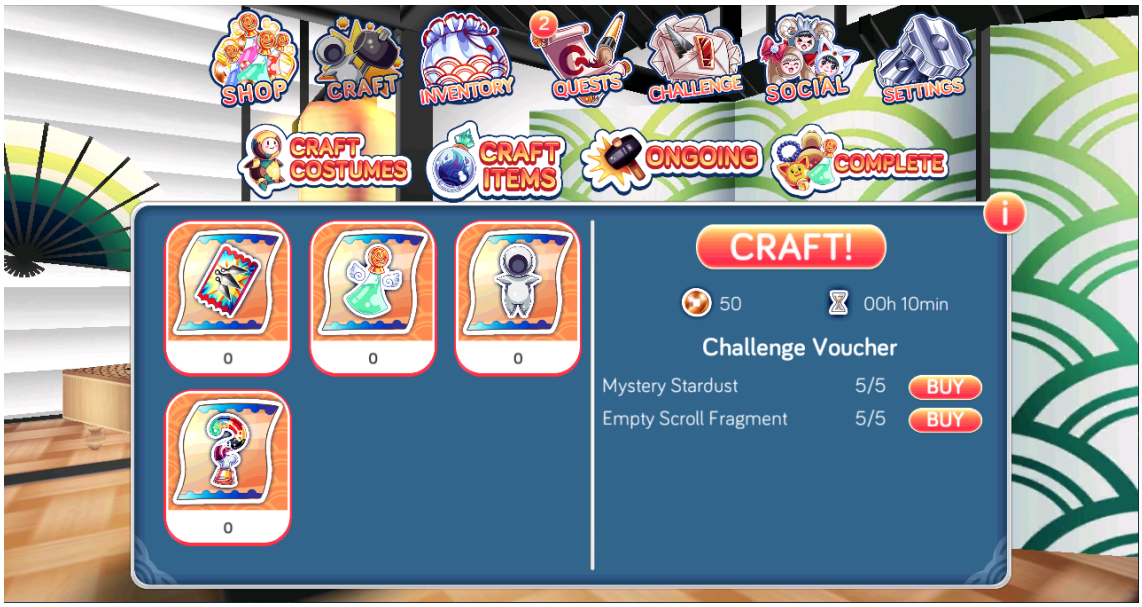
4.1.6 Kaverin poistaminen

Kaverin poistamista varten luotiin removeFriend-niminen tapahtuma. Tapahtuma vastaanottaa yhden parametrin, joka on kaverilistalta poistettavan kaverin id. Tapahtumassa haetaan muuttujaan poistettavan kaverin id. Tämän jälkeen poistetaan kaveri käyttäjän kaverilistasta ja käyttäjä poistettavan kaverin kaverilistasta. Tässäkään tapauksessa kaverilistasta poistettavalle käyttäjälle ei ilmoiteta, että käyttäjä on poistanut hänet kaverilistaltaan.

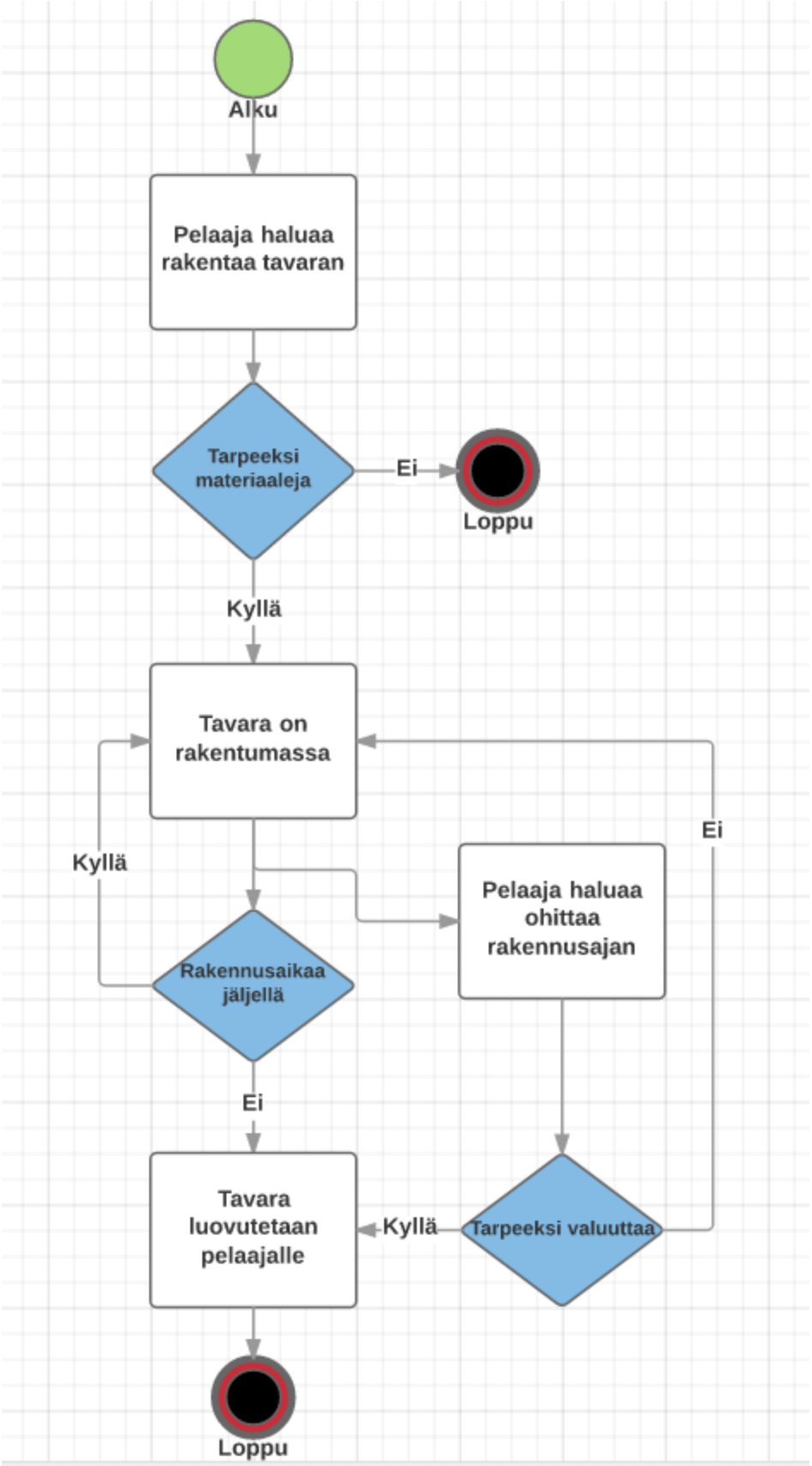
4.2 Tavaroiden rakentaminen

Peliin oli tarkoitus luoda systeemi, jonka avulla pelaajat pystyisivät rakentamaan erilaisia heitä hyödyttäviä tavaroita (kts. kuva 11). Tavaroiden rakennukseen käytettäisiin materiaaleja, joita pelaajat saisivat käyttöönsä tehtävien suorittamisesta. Tavaroiden rakentamiseen kuluisi aikaa minuuteista päiviin riippuen rakennettavasta tavarasta. Pelaajalla

olisi myös mahdollisuus ohittaa tämä rakennusaika käyttämällä pelinsisäistä valuuttaa (kts. kuva 12).



Kuva 11. Tavaran rakennuksen aloittaminen



Kuva 12. Tavaroiden rakennusta havainnollistava kaavakuva

4.2.1 SparkScheduler

Gamesparks sisältää valmiin moduulin nimeltä SparkScheduler, jonka avulla pystyy ajastamaan tapahtumia suoritettavaksi tietyn ajan kuluttua. Moduuli suoritti ajastetut tehtävät ajallaan, mutta sen tietoihin ei päästy ollenkaan käsiksi. Tämä aiheutti sen, että pelaajille ei saatu minkäänlaista tietoa siitä, kauanko heidän tavaransa rakentamisessa vielä menisi.

4.2.2 Craft Item

Tavaroiden rakentamista varten luotiin Craft Item -niminen tapahtuma. Tapahtumaa kutsutaan, kun pelaaja laittaa minkä tahansa tavaran rakentumaan. Tapahtuma vastaanottaa kolme parametria, jotka ovat:

- Item Short Code
  - rakennettava tavara
- Craft Time
  - tavaroiden rakentamiseen kuluva aika minuutteina
- Craft Quantity
  - Rakennettavien tavaroiden kappalemäärä.

Tapahtumassa haetaan playerCraftCollection-kokoelma muuttujaan. Kokoelma sisältää kaikkien pelaajien tällä hetkellä rakennuksessa olevat tavarat. Tapahtumassa luodaan kokoelmaan lisättävälle tietueelle uniikki id ja lisätään tämä kokoelmaan (kts. kuva 13).

```
var itemShortCode = Spark.getData().itemShortCode;
var craftTime = Spark.getData().craftTime;
var craftQuantity = Spark.getData().craftQuantity;

var playerCraftCollection = Spark.runtimeCollection("playerCraftCollec-
tion");

var itemIndex = Date.now() + Spark.getPlayer().getPlayerId() +
itemShortCode

playerCraftCollection.insert({"itemIndex" : itemIndex, "playerId" :
Spark.getPlayer().getPlayerId(),"craftQuantity" : craftQuantity,
itemShortCode" : itemShortCode, "craftTime" : craftTime});
```

Kuva 13. Tavara asetetaan rakentumaan

4.2.3 Rakennusajan seuranta

SparkScheduler-moduulia ei siis voitu käyttää tavaroiden rakentamisen ajastamiseen, koska siitä ei saatu haluttuja tietoja ulos. GameSparks sisältää kuitenkin kolme valmista tietyn väliajoin ajettavaa tapahtumaa. Tapahtumat suoritetaan minuutin, tunnin ja päivän välein. Oletukseltaan tapahtumat ovat tyhjiä luokkia. Käytämme rakennusajan päivittämiseen minuutin välein ajettavaa tapahtumaa.

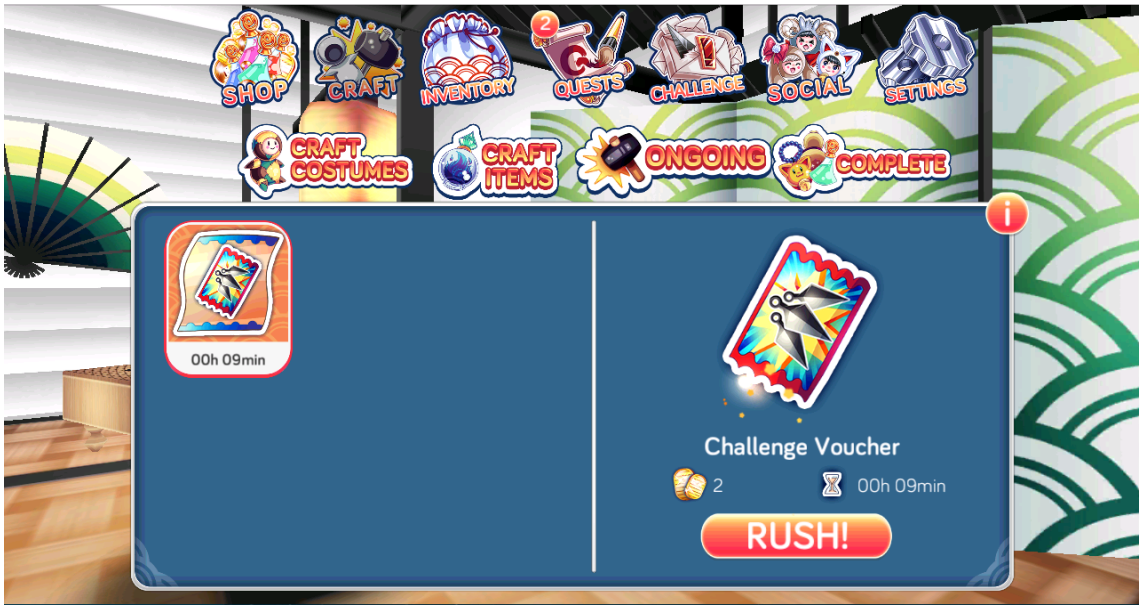
Tapahtumassa käydään jokainen playerCraftCollection-kokoelmassa rakenteilla oleva tavara läpi ja vähennetään näiden rakennusaikaa yhdellä minuutilla. Tämän jälkeen tarkastetaan jäljellä oleva rakennusaika. Jos rakennusaikaa ei ole jäljellä, lisätään tavara pelaajalle ja poistetaan tietue kokoelmasta. Jos rakennusaikaa on vielä jäljellä, päivitetään tietueen rakennusaika vähentämällä sitä minuutilla.

4.2.4 Rakennusajan ohittaminen

Pelaajalla tulee olla mahdollisuus ohittaa rakennukseen käytettävä aika ja saada tavara itselleen heti. Tämä kuitenkin maksaa jonkin verran pelinsisäistä valuuttaa, riippuen siitä, kauanko rakennusaikaa on vielä jäljellä. Rakennusajan ohittamiseen käytettävä tapahtuma vastaanottaa kolme parametria, jotka ovat:

- itemIndex
  - tietueen sijainti kokoelmassa
- itemShortCode
  - rakennettava tavara
- currencyCost
  - rakennusajan ohittamisen hinta.

Tapahtumassa haetaan playerCraftCollection-kokoelmasta se tavara, jonka rakennusaika halutaan ohittaa (kts. kuva 14). Lisätään halutta tavara pelaajalle ja veloitetaan häneltä maksua rakennusajan ohittamisesta. Rakennuksessa olleen tavaran tietue poistetaan kokoelmasta ja päivitetään pelaajan tietoihin rakennettujen tavaroiden lukumäärä (kts. kuva 15).



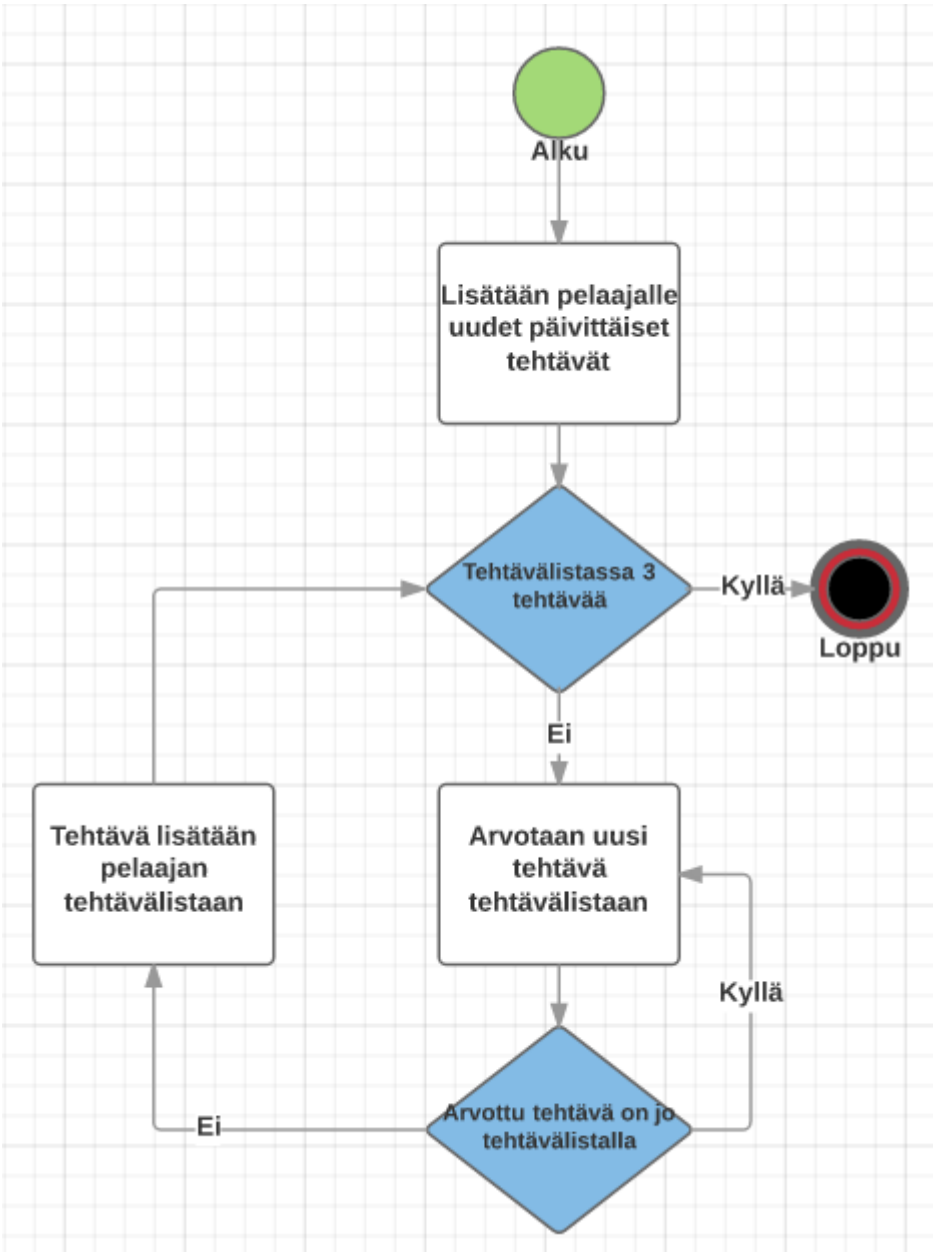
Kuva 14. Rakennusajan ohittaminen

```
var playerCraftCollection =
Spark.runtimeCollection("playerCraftCollection");
var craftableItem = playerCraftCollection.findOne({"itemIndex" :
Spark.getData().itemIndex});

var currentPlayer = Spark.loadPlayer(craftableItem.playerId);

currentPlayer.addVGood(Spark.getData().itemShortCode, 1);
currentPlayer.debit2(Spark.getData().currencyCost);

playerCraftCollection.findAndRemove({"itemIndex" :
Spark.getData().itemIndex});

var playersDirectoryCollection =
Spark.runtimeCollection("playerDirectory");

playersDirectoryCollection.update({"playerId" : craftableItem.playerId},
{"$inc" : {"generalInfo.totalItemsCrafted" : 1}}, true, false);
```

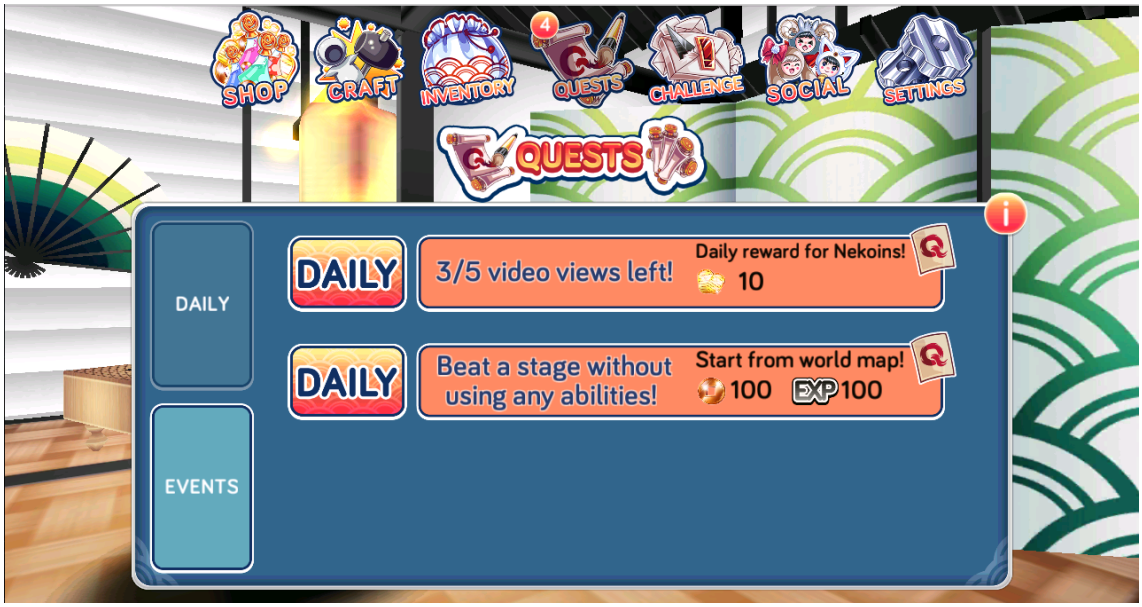
Kuva 15. Rakennusajan ohitus

4.3 Päivittäiset tehtävät

Normaalien tehtävien lisäksi pelissä on myös päivittäisiä tehtäviä (kts. kuva 17). Pelaajalla voi olla maksimissaan kolme päivittäistä tehtävää ja näiden suorittamisesta pelaajaa palkitaan erilaisilla materiaaleilla. Tehtävät arvotaan summittaisesti JSON-tiedostosta, joka sisältää listan kaikista päivittäisistä tehtävistä. Tehtävät arvotaan vain suoritettujen tehtävien tilalle, eli jos käyttäjä ei ole vielä suorittanut vanhoja tehtäviään, ei näiden tilalle arvota uusia. Myös duplikaatteja ei lisätä pelaajan tehtävälistaan, eli sama tehtävä ei voi esiintyä useampaan kertaan pelaajan tehtävälistassa (kts. kuva 16).



Kuva 16. Päivittäisten tehtävien toimintaa havainnollistava kaavakuva



Kuva 17. Pelaajan päivittäiset tehtävät

4.3.1 Tehtävien lisääminen

Tehtävien arpominen pelaajan tehtävälistaan suoritetaan GameSparksin luomassa päivittäin ajettavassa tapahtumassa.

Tapahtuma käy läpi kaikki rekisteröityneet pelaajat. Jos käyttäjällä ei ole yhtään päivittäistä tehtävää listassaan, arvotaan tälle listaan kolme satunnaista tehtävä. Jos tehtäviä on listassa alle kolme, arvotaan puuttuva määrä tehtäviä listaan, mutta ei kuitenkaan samoja tehtäviä, jotka jo listassa ovat (kts. kuva 18).

```
while(playerList.hasNext())
{
  var currentPlayer = playerList.next();

  if(currentPlayer["playerId"] !== null &&
currentPlayer["playerId"] !== undefined)
  {
       var currentLoadedPlayer =
       Spark.loadPlayer(currentPlayer["playerId"]);
       var remainingDailyQuests = currentPlayer.remainingDailyQuests;
       var combinedQuestsArray = dailyQuests;

       playerCollection.update({"playerId" :
       currentPlayer["playerId"]}, {"$set": {"hasReceivedDailyReward":
       false}}, true, false);

       if(remainingDailyQuests < 3 || remainingDailyQuests == null ||
       remainingDailyQuests == undefined)
       {
         var dailyQuests = currentPlayer.dailyQuests;
         var quests = shuffleArray(combinedQuestsArray).slice(0, 4);


         if(Array.isArray(dailyQuests))
         {
              quests = quests.concat(dailyQuests);

              if(indexOfObject(dailyQuests,quests[0]) === -1)
              {
                   dailyQuests.push(quests[0]);
              }
              else if(indexOfObject(dailyQuests,quests[1])  === -1)
              {
                   dailyQuests.push(quests[1]);
              }
              else if(indexOfObject(dailyQuests,quests[2])  === -1)
              {
                   dailyQuests.push(quests[2]);
              }
              playerCollection.update({"playerId" :
              currentPlayer["playerId"]}, {"$set":
              {"remainingDailyQuests": dailyQuests.length,
              "dailyQuests": dailyQuests}}, true, false);

         }
         else
         {
              quests = quests.slice(0, 1);
              playerCollection.update({"playerId" :
              currentPlayer["playerId"]}, {"$set":
              {"remainingDailyQuests": quests.length, "dailyQuests":
              quests}}, true, false);
         }


       }
  }
}
```

Kuva 18. Päivittäisten tehtävien lisäys pelaajalle

4.3.2 Tehtävien palautus

Päivittäisten tehtävien palautusta varten luotu tapahtuma vastaanottaa kolme parametria, jotka ovat

- User Id
  - käyttäjän id
- Quest Name
  - palautettavan tehtävän nimi
- Multip
  - kasvava kerroin, jolla kerrotaan pelaajalle tehtävästä annettava palkkio. Kerrointa kasvatetaan, kun pelaaja kirjautuu peliin peräkkäisinä päivinä.

Tapahtumassa käydään läpi pelaajan tehtävälistalla olevat tehtävät. Jos tehtävä on suoritettu, annetaan pelaajalle tehtävästä saatavat palkinnot. Tehtävä poistetaan pelaajan tehtävälistasta ja vähennetään pelaajan jäljellä olevien tehtävien määrää yhdellä (kts. kuva 26). Pelaajan ei tarvitse käydä itse erikseen palauttamassa tehtävää, vaan tapahtumaa kutsutaan automaattisesti, kun tehtävän asettamat vaatimukset täyttyvät.

```
var playerDir = Spark.runtimeCollection("playerDirectory");
var playerId = Spark.getData().userId;
var player = Spark.loadPlayer(playerId);
var multip = Spark.data.multip;

var playerInventory = playerDir.findOne({"playerId" : playerId});

for (var i = 0; i < playerInventory.remainingDailyQuests; i++)
{
    var quest = playerInventory.dailyQuests[i];

    if(quest.name == Spark.getData().questName)
    {

        for(var j = 0; j < quest.rewards.length; j++)
        {
            var reward = quest.rewards[j];

            switch(reward.shortCode)
            {
                case "currency2":
                    player.credit2(reward.amount*multip);
                    break;
                case "currency3":
                    player.credit3(reward.amount*multip);
                    break;
                default:
                    player.addVGood(reward.shortCode,
                  reward.amount*multip);
                    break;
            }
        }
            playerDir.update({"playerId" : playerId}, {"$pull":
            {"dailyQuests": {shortCode : quest.shortCode}}}, true,
            false);

            playerDir.update({"playerId" : playerId}, {"$inc":
            {"remainingDailyQuests": -1}}, true, false);
    }
}
```

Kuva 19. Päivittäisen tehtävän palautus

5 Yhteenveto

Insinöörityön pääaiheina olivat GameSparks-pilvipalvelu ja sillä harjoitteluni aikana rakentamani ratkaisut. Sen ohessa oli tarkoitus tutustua yleisesti siihen, mitä termillä pilvipalvelu tarkoitetaan.

Työskennellessäni GameSparksin kanssa noin vuosi sitten oli se vielä betassa, minkä johdosta osa toiminnoista ja niiden dokumentoinneista oli vielä vaiheessa. Dokumentointien keskeneräisyys aiheutti ongelmia pariinkin otteeseen.

Esimerkiksi haasteiden yhteydessä pelaajat voivat lyödä vetoa keskenään pelinsisäistä valuuttaa käyttäen. Tiesimme, että panos asetetaan haastetta lähettäessä, mutta missään ei mainittu, milloin tuo kyseinen summa veloitetaan molemmilta pelaajilta. Mitä jos pelin hävitessään pelaajalla ei olekaan tarpeeksi rahaa maksaa vastustajalle asetettua panosta?

Ehdimme rakentaa peliin jo valmiin ratkaisun, joka ottaa kyseisen summan talteen kummaltakin pelaajalta, kun haastettu hyväksyy haasteen. Testausvaiheessa kuitenkin selvisi, että GameSparksilla olikin jo valmiiksi rakennettu ratkaisu, joka toimi samalla tavalla kuin omammekin. Kehitysaikaa olisi siis säästynyt useampikin tunti, jos tästä olisi vain mainittu dokumentoinnissa tai jossakin esimerkissä.

Suurin osa muista BaaS-tyypin palveluista eivät ole räätälöityjä vain pelejä varten. GameSparksin keskittyminen vain peleihin on samalla sen suurin vahvuus, mutta myös sen suurin heikkous. GameSparksin hinnoittelu tarjoaa oivan mahdollisuuden harrastelijoille ja indie-kehittäjille lähteä kokeilemaan onneaan pelimarkkinoilla, kun palvelinpuolen ominaisuudet saa käyttöönsä ilmaiseksi.

GameSparks rupeaa veloittamaan palvelun käytöstä vasta, kun kehittäjän pelin kuukausittaisten käyttäjien määrä ylittää 100 000 kävijän rajapyykin. On vaikea nähdä, miten tällaisella hinnoittelulla pystytään pitämään palvelua yllä, kun mobiilipelimarkkinoilla menestyvän pelin luominen on melkeinpä yhtä todennäköistä kuin lottovoitto.

GameSparks mainostaa kuitenkin, että heidän palveluaan käyttää muutama pelinkehityksen suurnimistä, kuten Square Enix ja Ubisoft. On hyvinkin mahdollista, että GameSparks rahoittaakin toimintansa näiden jättien kanssa tehtyjen räätälöityjen sopimusten kautta.

GameSparksin asiakaspalvelusta pitää antaa kuitenkin positiivista palautetta. Palvelu oli nopeaa ja ystävällistä, pisin aika jonka odotimme apua, oli vain noin 10 minuutta.

Rakentamani ratkaisut ovat vielä käytössä yrityksen luomassa pelissä ja olen tyytyväinen niihin, ottaen huomioon, että olimme kaikki aloittelijoita GameSparksin käytössä. Ratkaisut kävivätkin läpi useita iteraatioita, aina kun uusia ominaisuuksia ilmeni ja saimme neuvoja GameSparksin tuesta.

GameSparksista kirjoittaminen aiheutti hankaluuksia, koska siitä löytyy todella niukasti tietoa. Viimeisimmät löytämäni palveluun liittyvät haastattelut ja artikkelit olivat vuodelta 2014. Turvauduinkin lähinnä GameSparksista kirjottaessa heidän omaan blogiinsa ja dokumentointiin.

Loppujen lopuksi GameSparksista jäi positiivinen mielikuva. Sen ominaisuudet toimivat kuten pitivätkin, vaikkakin dokumentoinneissa oli välillä puutteita. Sillä sai mielestäni rakennettua toimivia, vaikkei ehkä kaikkein sulavimpia ratkaisuja. Suosittelisin kokeilemaan GameSparksia, jos etsitään peliin palvelinpuolen ratkaisua. Siinä ei ainakaan häviä mitään.

Lähteet

1. Mobile cloud backend as a Service Ecosystem Map – All roads lead to BaaS, http://www.kinvey.com/blog/65/mobile-cloud-backend-as-a-service-ecosystem-map-8211-all-roads-lead-to-baas, 16.3.2016.

2. What is backend as a service, http://www.baasbox.com/what-is-backend-as-a-service/, 17.4.2016.

3. Cloud computing introduction, http://www.explainthatstuff.com/cloud-computing-introduction.html, 16.3.2016.

4. Increased Security Vulnerabilities, http://whatiscloud.com/risks_and_challenges/increased_security_vulnerabilities, 16.3.2016.

5. Understanding the Cloud Computing Stack: SaaS, PaaS, IaaS, https://support.rackspace.com/white-paper/understanding-the-cloud-computing-stack-saas-paas-iaas/, 20.3.2016.

6. What is SaaS, http://www.interoute.com/what-saas, 20.3.2016.

7. Platform as a Service (PaaS), http://searchcloudcomputing.techtarget.com/definition/Platform-as-a-Service-PaaS], 20.3.2016.

8. Infrastructure as a service, http://whatiscloud.com/cloud_delivery_models/infrastructure_as_a_service., 22.3.2016.

9. Public clouds, http://whatiscloud.com/cloud_deployment_models/public_clouds, 22.3.2016.

10. Community clouds, http://whatiscloud.com/cloud_deployment_models/community_clouds, 22.3.2016.

11. Private clouds, http://whatiscloud.com/cloud_deployment_models/private_clouds, 22.3.2016.

12. Hybrid Clouds, http://whatiscloud.com/cloud_deployment_models/hybrid_clouds, 22.3.2016.

13. 12 reasons to use gamesparks, http://www.pocketgamer.biz/feature/59149/12-reasons-to-use-gamesparks/], 24.3.2016.

14. Indie student programme faq https://www.gamesparks.com/indie-student-programme-faq/, 24.3.2016.

15. Gamesparks docs, https://docs.gamesparks.net/tutorials/snapshots, 26.3.2016.

16. Gamesparks docs, https://docs.gamesparks.net/documentation/request-api/multiplayer-request-api/findchallengereques, 26.3.2016.

17. GameSparks docs, https://docs.gamesparks.net/developer-portal/analytics, 26.3.2016.

18. GameSparks docs, https://docs.gamesparks.net/developer-portal/nosql, 26.3.2016.