TAMPEREEN
AMMATTIKORKEAKOULU

GAMESPARKS-PALVELUN KÄYTTÖ ROLLER CRASH -PELISSÄ

Tomi Pulliainen

Opinnäytetyö
Toukokuu 2018
Tietojenkäsittely
Pelituotanto

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietojenkäsittely
Pelituotanto

PULLIAINEN, TOMI:
GameSparks-palvelun käyttö Roller Crash -pelissä

Opinnäytetyö 43 sivua
Toukokuu 2018

Tässä opinnäytetyössä tarkastellaan case studyn kautta GameSparks-palvelua ja sen tarjoamia ominaisuuksia Roller Crash -mobiilipelin näkökulmasta. GameSparks on backend-palveluja tarjoava yritys, jota lukuisat eri pelit käyttävät. Roller Crash on Pelite Productions Oy:n kehittämä mobiilipeli, jonka ominaisuudet vaativat backend-palvelua. Olin itse vastuussa pelin backendin toteuttamisesta.

Opinnäytetyössä tutkittiin, mitä ominaisuuksia GameSparks tarjoaa käyttäjilleen ja miten niitä voi hyödyntää Roller Crash -mobiilipelin tarpeisiin. Opinnäytetyössä käsitellään pelille asetetut tarpeet sekä dokumentoidaan niiden toteuttaminen GameSparksin tarjoamien ominaisuuksien ja palvelujen avulla.

GameSparks vastasi hyvin pelin backend-tarpeisiin ja sen avulla saatiin toteutettua kaikki halutut ominaisuudet niille ennalta asetettujen vaatimusten mukaan. Haasteita tuottivat palvelun vähäinen dokumentaatio ja reilun käytön rajat, jotka rajoittavat pelaajakohtaista tiedonsiirtoa. Palvelun käyttö vaati paneutumista ja opettelua, mutta niiden avulla palvelun käytön oppiminen oli mahdollista ohjelmoijalle, jolla ei ollut aiempaa kokemusta pelien backendistä.

Asiasanat: backend, games-as-a-service, gaas, backend-as-a-service, baas

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Business Information Systems
Game Development

PULLIAINEN, TOMI:
GameSparks Service in Roller Crash Game

Bachelor's thesis 43 pages
May 2018

The purpose of thesis was to examine features of a backend service provider GameSparks as a case study. The subject of the case study was a mobile game Roller Crash developed by Pelite Productions Oy.

This study aims to examine what features GameSparks provides for its users and how to fulfil the needs of Roller Crash with these features. The thesis covers the features made for the game by using GameSparks services and documents the implementation process.

The case study concludes that GameSparks was suitable for the needs of the game and it was possible to implement all features wanted as they were designed. Low documentation about the subject and fair usage limits of GameSparks made the progress slightly challenging but as a result, the service was possible to be learned by a user who did not have prior experience from backend services.

Key words: backend, games-as-a-service, gaas, backend-as-a-service, baas

SISÄLLYS

ERITYISSANASTO

APNs	Applen Push-ilmoitus palvelu (Apple Push Notification service)
BaaS	Pilvessä toimiva backend-palvelu (Backend-as-a-Service)
Backend	Ohjelman taustalla toimiva järjestelmä
Backend-as-a-Service	Pilvessä toimiva backend-palvelu
C#	Ohjelmointikieli
FCM	Googlen Push-ilmoitus palvelu (Firebase Cloud Messaging)
GaaS	Pelit palveluna (Games-as-a-Service)
Games-as-a-Service	Pelit palveluna
GSML	GameSparks-palvelussa hallintapaneeleiden tekemiseen käytettävä kuvauskieli (GameSparks Markup Language)
HTML	Webissä käytetty kuvauskieli (Hypertext Markup Language)
Javascript	Ohjelmointikieli
JSON	Tiedostomuoto tiedonvälitykseen (JavaScript Object Notation)
MAU	Kuukausittainen käyttäjä (Monthly Active User)
Monetisaatio	Ansaintamalli
NoSQL	Tietokanta, joka ei käytä relaatiota (Not Only SQL)
SDK	Ohjelmistokehityspaketti (Software Development Kit)
SQL	Kyselikieli relaatiotietokantaan (Structured Query Language)
Unity	Pelimoottori

1 JOHDANTO

Nykypäivän pelit ovat entistä enemmän palveluja. Pelaajat haluavat, että pelissä on moninpelielementtejä ja muuttuvaa sisältöä. Pelinkehittäjät taas haluavat parempaa analytiikkaa pelaajien käyttäytymisestä ja muokata pelin sisältöä helposti ja nopeasti. Peleistä palveluna puhutaan termein ”Games-as-a-service” tai lyhyemmin ”GaaS”.

Vanhoina päivinä pelinkehittäjät tekivät pelin, laittoivat sen levylle, antoivat sen julkaisijalle ja siirtyivät seuraavan projektin pariin. Nykypäivänä pelinkehittäjien pitää saada tehtyä peli, pelaajat pelaamaan peliä ja rakentaa yhteys pelin pelaajien kanssa. (Griffin 2014.) Todd Harrisin (Co-founder & COO, Hi-Rez Studios) mukaan pelit palveluna merkitsee juuri sitä, että peli ei ole koskaan valmis. Julkaisu on vasta lähtöviiva eikä maaliviiva. (Wong 2017.)

Jotta pelistä voisi tehdä palvelun, tarvitaan pelille jonkinlainen backend-ratkaisu. Pelin backendillä tarkoitetaan pelin taustalla toimivaa järjestelmää. Tähän järjestelmään voivat olla liitettynä esimerkiksi pelaajien profiilit, pelin tiedot, sosiaaliset elementit, pistetilastot, saavutukset ja analytiikka. Myös pelin logiikka voi toimia backendin kautta.

Pelinkehityksessä aika on rahaa ja pelin backendin rakentamiseen voi upota paljon aikaa, jos sitä alkaa tekemään alusta asti itse. Varsinkaan pienillä pelistudioilla ei ole aikaa eikä resursseja rakentaa alusta asti omaa backendiä. Ulkopuolisen tahon tarjoamat pilvessä toimivat backend-palvelut tuovat ratkaisun näihin ongelmiin, koska ne sisältävät tarvittavat kehitystyökalut jo valmiiksi rakennettuina. Myöskään backendin ylläpitoratkaisuja ei tarvitse näin itse miettiä, koska ne on jo hoidettu valmiiksi.

Aikaisemmin pelinkehittäjät ovat toteuttaneet pelin backendin omien palvelinratkaisujen kautta, mutta nykyään rinnalle ovat tulleet ulkoiset pilvessä toimivat palveluntarjoajat, jotka tarjoavat omia palveluja backendin kehittämiseen. Palveluja kutsutaan nimellä ”Backend-as-a-Service” tai lyhyemmin ”BaaS”. GameSparks on yksi näistä pilvessä toimivista BaaS-palveluiden tarjoajista.

Opinnäytetyöni tarkoituksena on tarkastella GameSparks-palvelua yleisesti ja sen tarjoamia ominaisuuksia Roller Crash -mobiilipelin näkökulmasta. Toimin kyseisen pelin teknisenä johtajana ja olin vastuussa pelin backendin toteuttamisesta. Käytin backendin toteuttamiseen GameSparksin palveluja ja toteuttamisen kautta opin paljon kyseisen palvelun käytöstä. Samalla huomasin, että palvelun käyttämisestä ei löytynyt paljoa dokumentointia. Halusinkin opinnäytetyöni kautta jakaa tietoa asioista, joita opin projektin aikana. Pelit palveluna on myös hyvin ajankohtainen aihe, koska peleiltä vaaditaan entistä enemmän ominaisuuksia, jotka vaativat palveluita taustalle toimiakseen.

Tämä opinnäytetyö on tarkoitettu pelien pilvessä toimivista backend-palveluista kiinnostuneille ja niille, jotka aloittelevat GameSparksin käyttämistä. Opinnäytetyössäni etsitään ratkaisuja Roller Crash -mobiilipelin tarpeisiin ja sen kautta lukija pystyy tutkimaan olisiko samankaltaisista ratkaisuista hänelle hyötyä. Opinnäytetyössäni tutkin, millaisia ominaisuuksia GameSparks tarjoaa käyttäjilleen ja miten niitä voi hyödyntää. Kerron myös, miten Roller Crashin ominaisuudet toteutettiin palvelua käyttäen. Opinnäytetyön lukijan olisi hyvä hallita perusasiat ohjelmoinnista, mutta se ei ole välttämätöntä.

2 SOPIVAN BACKEND-PALVELUN VALINTA

Pilvessä toimivia backend-palveluiden tarjoajia on nykypäivänä monia. Nämä tarjoavat samankaltaisia palveluja, mutta hinnoittelussa voi olla suuriakin eroja. Yleensä palvelut tarjoavat kehittäjälle ainakin tietokannan, pilvikoodin, ja analytiikan, mutta jokainen palvelu tarjoaa lisenssikohtaisia lisäominaisuuksia. Palveluntarjoajan valinta riippuukin täysin siitä mitä tarpeita pelin backendille on ja mikä on pelin budjetti.

2.1 Ehdokkaat Roller Crash -pelin backend-palveluksi

Roller Crashin backendiksi harkitsimme GameSparksin, Playfabin ja Googlen Firebasen palveluja, joita vertailimme keskenään hinnoittelun ja tarjottujen palvelujen pohjalta. Nämä palvelut kuuluvat suosituimpiin pilvessä toimiviin backend-palveluihin.

2.1.1 GameSparks

GameSparks palvelu tarjoaa käyttäjilleen kolmea erilaista lisenssiä:

- Evaluation & Prototyping
- Indie & Student Programme
- Enterprise.

(GameSparks Pricing 2018.)

Evaluation & Prototyping -lisenssin saa itselleen ilmaiseksi rekisteröitymällä palveluun. Tällä lisenssillä saa käyttöön kaikki Gamesparksin tarjoamat ominaisuudet, mutta käyttäjä voi toimia vain Preview-alustassa, joka on tarkoitettu testikäyttöön. Preview-alustaan voi olla yhteydessä samanaikaisesti ainoastaan 100 pelaajaa, joka tekee lisenssin käyttämisestä mahdotonta julkaistavassa pelissä.

Indie & Student Programme -lisenssi täytyy hakea ja hakijan täytyy olla jokin seuraavista:

- ammattimainen indie-pelikehittäjä jolla on pieni kehitystiimi
- opiskelija joka opiskelee peleihin liittyvää kurssia
- koulutushenkilöstöön kuuluva joka on sidoksissa peleihin liittyvään kurssiin
- harrastelijapelikehittäjä.

(Indie & Student Programme FAQ 2018.)

Indie- ja opiskelijalisenssillä saa käyttöön kaikki GameSparksin ominaisuudet ja käyttäjä voi myös julkaista pelinsä Live-ympäristöön, jossa ei ole mitään rajoituksia. Lisenssi on täysin ilmainen, kunnes peli saavuttaa 100 000 aktiivisen kuukausittaisen käyttäjän rajan (MAU). Tämän jälkeen käyttäjä joutuu maksamaan 0,02$ per MAU 100 000 yli menevistä käyttäjistä. Lisenssi vaatii myös, että pelin alussa näkyy GameSparks-logo.

Enterprise-lisenssit räätälöidään aina asiakkaan tarpeisiin sopiviksi ja ne neuvotellaan erikseen GameSparksin kanssa.

2.1.2 PlayFab

PlayFab palvelu tarjoaa käyttäjilleen kolmea erilaista lisenssiä:

- Essentials
- Professional
- Enterprise.

(PlayFab Pricing 2018.)

Essentials-lisenssi on ilmainen ja sillä saa käyttöönsä kaiken perustoiminnallisuuden. Lisenssillä pelaajia voi olla rajattomasti, mutta jotkin ominaisuudet on rajattu pois. Näitä rajattuja ominaisuuksia on muun muassa Scheduled tasks -ominaisuus, jolla pilven tapahtumia voi ajastaa ja Advanced analytics -ominaisuus, jolla pelistä saa parempaa analytiikkaa.

Professional-lisenssi maksaa 0,008$ per MAU, mutta vähintään 299$ kuukaudessa. Tämä lisenssi avaa kaikki palvelun ominaisuudet. Lisenssi alkaa maksamaan vasta, kun peli on julkaistu. Julkaistulla pelillä tarkoitetaan peliä, joka on saavuttanut 1000 MAU-rajan tai peliä, jonka julkaisusta on ilmoitettu PlayFabille.

Enterprise-lisenssit räätälöidään aina asiakkaan tarpeisiin sopiviksi ja ne neuvotellaan erikseen PlayFabin kanssa.

2.1.3 Firebase

Googlen Firebase palvelu tarjoaa käyttäjälle kolmea erilaista lisenssiä:

- Spark Plan
- Flame Plan
- Blaze Plan.

(Firebase Pricing Plans 2018.)

Spark Plan -lisenssi on ilmainen, mutta sisältää käyttörajoituksia. Käyttörajoitukset ovat tiedonsiirron ja tietokantojen kokojen rajoituksia. Yksi näistä rajoituksista muun muassa on, että yhtäaikaisia tietokantayhteyksiä voi olla vain 100. Tämä rajaa lisenssin käyttämisen julkaistavassa pelissä pois, koska tietokantayhteyksiä tarvitaan enemmän.

Flame Plan -lisenssi maksaa 25$ kuukaudessa ja käyttörajoitukset eivät ole niin tiukat kuin ilmaisessa lisenssissä. Esimerkiksi yhtäaikaisten tietokantayhteyksien määrä nousee 100 000 samanaikaiseen yhteyteen.

Blaze Plan -lisenssissä käyttäjä maksaa siitä kuinka paljon käyttää resursseja. Näillä resursseilla tarkoitetaan muun muassa tiedonsiirtoa pelin ja Firebasen välillä sekä tietokantaan tallennetun datan kokoa. Firebasen sivuilta löytyvän laskurin avulla käyttäjä pystyy arvioimaan tarvitsemiensa resurssien hinnan.

2.2 Miksi GameSparks valittiin

GameSparks soveltui tarpeisiimme tarjoamiensa ominaisuuksien puolesta. Indie & Student Programme -lisenssillä saa kaikki ominaisuudet käyttöön ja käyttäjän tarvitsee maksaa vain, jos peli menestyy eli saa yli 100 000 MAU. Se myös sisältää tarvitsemamme Scheduled tasks -ominaisuuden mitä PlayFabin ilmaislisenssiin ei sisälly. Se on myös sallivampi käyttörajoitusten kanssa, kuin Firebase, koska kaikki rajat perustuvat kuukausittaisten aktiivisten käyttäjien määrään.

3 GAMESPARKS

GameSparks on BaaS-palveluja tarjoava yritys, jonka perustivat vuonna 2013 Gabriel Page ja John Griffin. Myöhemmin mukaan liittyi myös Griffin Parry, joka toimi yrityksen toimitusjohtajana. Vuonna 2017 GameSparks siirtyi Amazonin omistukseen. (Trish 2014; GameSparks Joins the Amazon Family 2018.)

Yrityksen tavoitteena on tarjota BaaS-palveluja erityisesti pelinkehittäjien tarpeisiin. John Griffinin mukaan yrityksen vahvuutena on kyetä tarjoamaan pelistudioille työkaluja, joilla pelien palvelinpuolen komponenttien rakentaminen ja käsittely ovat helppoa. (Trish 2014.) Nykypäivänä yritys onkin yksi suosituimmista BaaS-palveluntarjoajista ja sitä käyttää yli 200 pelistudiota (GameSparks FAQ 2018). Yrityksen palveluissa pelinkehittäjiä houkuttelee erityisesti sen tarjoamat monipuoliset ominaisuudet, skaalautuvuus ja reilu hinnoittelu.

GameSparksia on käytetty useiden suosittujen pelien backendin toteuttamiseen:

- 7 Days to Die (PC, PS4, Xbox One)
- Pac-Man Pop (Android, iOS)
- Lara Croft: Relic Run (Android, iOS)
- PewDiePie’s Tuber Simulator (Android, iOS)
- Carmageddon (Android, iOS).

(GameSparks Showcase 2018.)

4 GAMESPARKSIN TARJOAMAT OMINAISUUDET

GameSparks tarjoaa paljon erilaisia ominaisuuksia, joilla pelikehittäjä voi luoda peliinsä backend-ominaisuuksia. Palvelussa on monia valmiita työkaluja, joilla kehittäjän haluamat ominaisuudet saadaan pelissä käyttöön helposti ja nopeasti, sekä työkaluja, joilla ominaisuudet voidaan tehdä täysin pelin omien tarpeiden mukaan.

KUVA 1. GameSparksin ominaisuudet (GameSparks FAQ 2018)

4.1 GameSparks-ympäristön alustat

Käyttääkseen GameSparksia täytyy ymmärtää, miten eri alustat GameSparks-pelin ympäristössä toimivat. Pelin ympäristössä on kaksi erilaista alustaa: preview ja live. Preview-ympäristö on suunniteltu kehittämiseen ja mahdollistaa 100 pelaajan yhtäaikaisen yhteydessä olemisen. Live-ympäristö on taas tarkoitettu julkaistulle pelille ja sillä ei ole mitään rajoituksia. Nämä alustat ovat täysin eristettyjä toisistaan, joka mahdollistaa pelin kehittämisen ilman, että se häiritsee julkaistun pelin toimintaa.

Preview-alustan muutosten siirto Live-alustaan tapahtuu snapshottien avulla. Pelin alustasta voidaan ottaa snapshot milloin tahansa ja se sisältää alustan sen hetkisen konfiguraation. Tämä snapshot voidaan sitten julkaista toiseen alustaan. Snapshottien avulla on myös virhetilanteissa helppo palata viimeksi toimivaan konfiguraatioon.

4.2 Pelaajan autentikoiminen

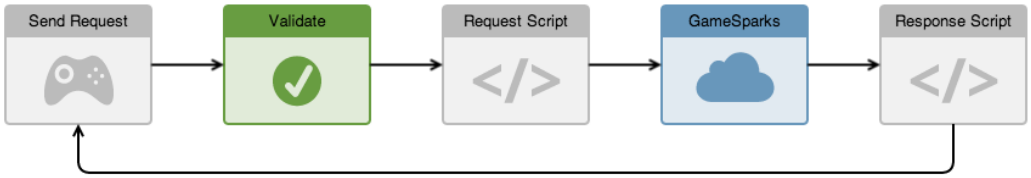
Autentikoimisella (*authentication*) tarkoitetaan pelaajan kirjautumista pelin GameSparks-alustaan ja alustaan liitettyyn pelaajatiliin. Tämä vaaditaan jokaiselta pelaajalta, jotta hän pystyy käyttämään pelissä GameSparksin tarjoamia ominaisuuksia.

GameSparks tukee useita autentikoimistapoja, jotka sopivat eri tilanteisiin. Pelaaja voidaan autentikoida perinteisellä käyttäjänimellä ja salasanalla, joka tarkoittaa, että pelaajan tulee syöttää hänen käyttäjänimensä ja salasana aina pelin alussa. Autentikoiminen voidaan myös suorittaa jotain kolmannen osapuolen tiliä käyttäen. Näitä voivat olla esimerkiksi Facebook-tili tai PlayStation-pelin yhteydessä PSN-tili. Jos autentikoiminen halutaan tehdä mahdollisimman helpoksi, voidaan käyttää laitekohtaista autentikoimista, jossa pelaaja autentikoidaan hänen laitteensa uniikilla tunnisteella. Tällöin pelaajan tili on laitekohtainen, mutta hänen ei tarvitse erikseen kirjautua millään tilillä.

4.3 Pilvikoodi

Pilvikoodi (*cloud code*) on yksi GameSparksin toiminnan peruspilareista. Yksinkertaisesti sillä tarkoitetaan pilvessä olevaa koodia, jota suoritetaan joko heti tai ajastetusti. Pilvikoodin avulla pelilogiikkaa voidaan siirtää pilveen, mikä lisää pelin tietoturvaa ja huijauksenestoa, koska pelaajalla ei ole mahdollisuutta muuttaa logiikkaa tilanteiden välissä. Pelilogiikkaa on myös näin helppo ja nopea muuttaa muokkaamalla koodia pilvessä eikä pelaajan täydy ladata erillisiä päivityksiä peliin. Pilvikoodissa käytetään JavaScript-kieltä.

Pilvikoodin toiminta tapahtuu lähettämällä pyyntö (*request*) GameSparksiin. Pyyntö sisältää tarvittavat parametrit halutun toiminnon suorittamiseen. Suorittamisen jälkeen peliin lähetetään vastaus (*response*), joka sisältää suorittamisesta saadut tiedot. Unity-pelimoottoria ja C#-kieltä käyttämällä oman pyynnön lähettäminen ja vastauksen prosessoiminen tapahtuvat yksinkertaisimmillaan kuvan 3 funktiolla. Kuvassa 4 näkyy pyynnön suoritus ja vastauksen lähettäminen pilvikoodilla.



KUVA 2. Pilvikoodin suorittaminen (Cloud Code 2018)

```
public void SendRequest()
{
    int numberOne = 1;
    int numberTwo = 2;

    new LogEventRequest().SetEventKey("COUNTNUMBERS")
        .SetEventAttribute("NUMBERONE", numberOne)
        .SetEventAttribute("NUMBERTWO", numberTwo)
        .Send((response) =>
        {
            if (!response.HasErrors)
            {
                int result = (int)response.ScriptData.GetInt("RESULT");

                Debug.Log(numberOne + " + " + numberTwo + " = " + result);
            }
            else
            {
                Debug.LogError("Something went wrong.");
            }
        }
    );
}
```

KUVA 3. Pyynnön lähetys ja vastauksen prosessoiminen Unityssä C#-kielellä

```
var numberOne = Spark.getData().NUMBERONE;
var numberTwo = Spark.getData().NUMBERTWO;

var result = numberOne + numberTwo;

Spark.setScriptData("RESULT", result);
```

KUVA 4. Pyynnön suorittaminen ja vastauksen lähettäminen pilvikoodilla

4.4 Reaaliaikainen moninpeli

Reaaliaikainen moninpeli (*realtime multiplayer*) voidaan toteuttaa monella eri tavalla, mutta esimerkiksi huijaamisen estämiseksi olisi tärkeää, että pelaajien välissä on kaiken loppupelissä päättävä palvelin. Tässä tapauksessa pelaajat lähettävät palvelimelle tietoja omista tekemisistään ja pelaaja saa palvelimelta vastauksena tietoja muiden pelaajien tekemisistä. Tämä pitää tietenkin tapahtua mahdollisimman pienellä viiveellä, jotta pelikokemus säilyisi hyvänä.

GameSparks pystyy toimimaan edellä mainittuna palvelimena pelaajien välissä. Pilvikoodista löytyy reaaliaikaisen moninpelin pystyttämiseen omat skriptit, joilla pelinkehittäjä voi rakentaa pelinsä ehdoilla toimivan reaaliaikaisen moninpeliympäristön.

4.5 MongoDB-tietokanta

Pelin backend vaatii yleensä tietokannan (*database*) tietojen tallennusta varten. Tietokannan avulla voidaan tallentaa kaikkien pelaajien edistyminen pelissä pilveen. Näin pelaajan tiedot ovat turvassa, vaikka pelaajan laitteelta häviäisi kaikki tiedot. Tietokantaan voidaan myös tallentaa kaikki peliin liittyvä informaatio, esimerkiksi pelin nopeus ja pelissä olevien taitojen eri tasot. Tämä mahdollistaa pelin muokkaamisen helposti ja nopeasti eikä pelaajien tarvitse ladata uutta peliversiota.

GameSparksissa käytetään MongoDB-tietokantaa. MongoDB on NoSQL-tietokanta, joka tarkoittaa sitä, että se ei käytä perinteistä SQL:n relaatiomallia. Relaatiomallissa tieto on jaettu taulukoista löytyviin riveihin ja näissä riveissä on kolumneja. Kolumnit voivat olla vanhempi-lapsi (*parent-child*) relaatiossa toisiinsa. Johdonmukaisuuden ollessa yksi kriittisistä tekijöistä tietokannan leveyssuunnassa skaalautuminen eli kolumnien lisääminen käy vaikeaksi, ellei mahdottomaksi. (Vaish 2013.) MongoDB:n dokumentit perustuvat JSON-fomaattiin ja tekee tietokannan skaalautumisen helpoksi, koska dokumenttien kenttiä voi vapaasti lisätä ilman, että se vaikuttaa toisiin kenttiin.

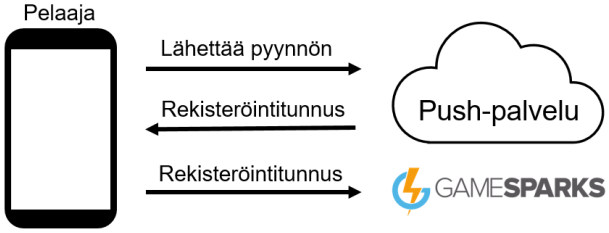
GameSparksissa tietokannan kokoelmat (*collections*) on jaettu kahteen eri luokkaan: meta- ja runtime-kokoelmiin. Meta-kokoelmat ovat suunniteltu muuttumattomaan tietoon ja niitä voidaan käyttää esimerkiksi peliasetusten säilömiseen. Runtime-kokoelmat ovat taas suunniteltu nopeasti muuttuvaan tietoon kuten pelaajien profiilit. Meta-kokoelmat siirtyvät aina snapshotin mukana toiseen alustaan, mutta runtime-kokoelmat ovat alusta-riippuvaisia ja eivät siirry.

4.6 Push-ilmoitukset

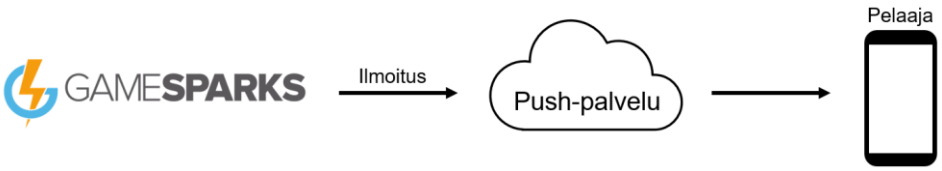
Pelaajalle on joskus tarpeellista ilmoittaa pelissä tapahtuvista muutoksista, vaikka hänellä ei olisi juuri silloin peli auki laitteellaan. Tällaiset muistutukset saavat myös pelaajat palaamaan peliin ja pelaamaan enemmän. Pelaajaa voidaan informoida esimerkiksi, että jokin pelin aikaa vievä prosessi on valmistunut tai joku on ohittanut pelaajan pistetilastoissa.

Mobiilialustoilla tämä onnistuu GameSparksissa Push-ilmoituksilla (*Push-notifications*). Push-ilmoitukset tarvitsevat ulkopuolisia Push-palveluntarjoajia välittäjiksi ilmoitusten jakeluun ja laitteen rekisteröimiseksi. Näitä ulkopuolisia palveluntarjoajia ovat muun muassa Firebase Cloud Messaging (FCM), joka toimii Android- ja iOS-laitteilla sekä Apple Push Notification Service (APNs), joka toimii iOS-laitteilla. Käytetyt palvelut tulee integroida GameSparksiin.

Push-ilmoitusten toimimiseen laite tulee rekisteröidä Push-palveluun ja lähettää rekisteröintitunnus GameSparksiin. Näin GameSparks osaa lähettää ilmoitukset oikeille käyttäjille. Pelaajan rekisteröidyttyä Push-palveluun ja GameSparksin tietäessä tämän rekisteröinti osoitteen pystytään Push-ilmoitukset lähettämään GameSparksista. GameSparks lähettää viestit Push-palveluihin, joka jakelee ne pelaajille.



KUVA 5. Pelaajan rekisteröiminen Push-palveluun



KUVA 6. Push-ilmoituksen lähettäminen pelaajalle

4.7 Hallintapaneelit

Pelin ja pelaajien tietoja on pelinkehittäjien hyvä päästä hallinnoimaan pilvestä käsin. Näitä tietoja voivat olla muun muassa peliasetukset, pelaajan profiili ja pelin eri tavarat. Tietojen muuttamisen pitäisi olla myös helppoa, koska kehitystiimissä voi olla esimerkiksi graafikkoja, jotka eivät ymmärrä miten MongoDB-tietokanta toimii.

GameSparksissa on monia valmiiksi rakennettuja hallintapaneeleita tietojen hallinnoimiseen. Näillä pääsee peruskäytössä jo pitkälle, mutta yleensä pelin edetessä tarvitaan tehdä omia hallintapaneeleja tiettyihin muutostarpeisiin. GameSparks mahdollistaa omien hallintapaneelien tekemisen käyttäen HTML- ja GameSparksin omaa GSML-kieltä. Hallintapaneelien toiminnallisuuden toteuttamiseen käytetään JavaScriptiä.

4.8 Analytiikka

Analytiikan saaminen ja sen seuraaminen ovat nykypeleissä tärkeää. Pelaajien käyttäytymistä pitää päästä seuraamaan, jotta nähdään mikä pelissä pelaajien mielestä toimii ja mikä ei. Analytiikka mahdollistaa järkevän jatkokehityksen pelille, koska päivityksiä ei tehdä laput silmillä. Analytiikan avulla voidaan myös tutkia pelaajien ostokäyttäytymistä ja saada pelillä enemmän rahaa.

GameSparksiin on sisäänrakennettuna monia valmiita analytiikkapaneeleja. Niitä voi myös tehdä itse ilman koodaamista GameSparksin omalla työkalulla. Jos valmis työkalu ei kuitenkaan riitä, analytiikkapaneelit voi tehdä itse koodaamalla hallintapaneelien tavoin.

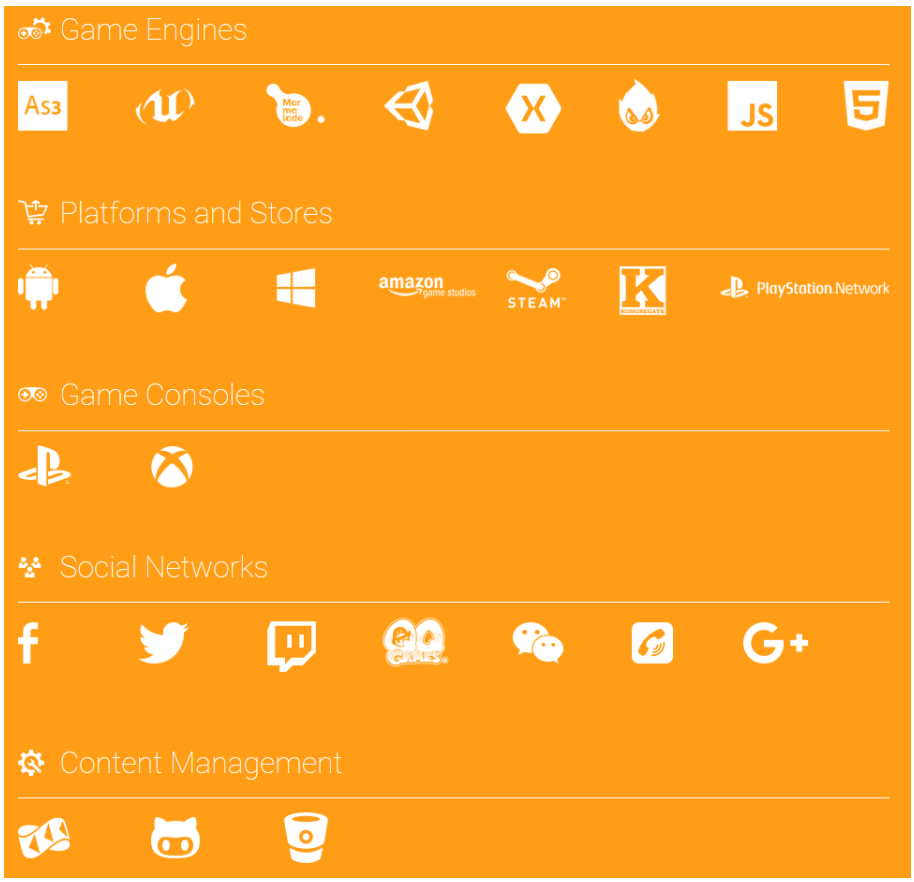
4.9 Pelin sisäiset ostot

Nykyisin suosittu monetisaatiomalli, varsinkin mobiilipeleille, on freemium-malli. Mallissa itse peli on ilmainen, mutta pelaajat voivat ostaa itselleen erilaisia parannuksia oikealla rahalla. Tällaiset parannukset voivat olla vaikkapa pelin resurssit (esimerkiksi raha, kultaharkot ja timantit), aseet, asut ja tavarat. Pelin sisäisten ostojen kuitit tulisi varmistaa ennen ostosten hyväksymistä. Tämä varmistaa, että pelinkehittäjä on saanut rahat ja eikä pelaaja voi huijata.

Mobiililaitteilla on myös eri alustoja ja ostojen hallinnointi voi muuttua hankalaksi. GameSparksin avulla pelin sisäisiä ostoja voidaan hallita yhdestä paikasta. GameSparks sisältää Digital Goods -työkalun, johan syötetään pelin Google Play- ja App Store -tuotteiden tiedot. Pelaajan ostaessa tuotteen GameSparks tarkistaa kauppapaikoilta pelaajan kuitin ja kuitin ollessa kunnossa tallentaa tuotteen pelaajan profiiliin.

4.10 Yhteensopivuus ja integraatio

Yksi GameSparksin suurimmista vahvuuksista on sen yhteensopivuus ja integrointimahdollisuudet. Palvelusta löytyy valmis SDK eli ohjelmistokehityspaketti useimpiin pelimoottoreihin ja kehitysalustoihin. Useiden sosiaalisten medioiden palvelut on myös mahdollista integroida GameSparksiin ja käyttää näitä esimerkiksi pelaajan autentikoimiseen. Pilvikoodin kanssa työskentelyn helpottamiseksi GameSparksiin on myös mahdollista integroida versionhallintapalveluja (GitHub or Bitbucket Synchronization with Cloud Code Import / Export 2018).



KUVA 7. Gamesparksin yhteensopivuus (GameSparks Integrations 2018)

5 ROLLER CRASH

Roller Crash on Unity-pelimoottorilla toteutettu, toiminnantäytteinen mobiilipeli Android- ja iOS-alustoille. Roller Crashin tekeminen aloitettiin syyskuussa 2016 ja se julkaistiin testiin Suomessa ja Kanadassa 3.2.2018 ja maailmanlaajuisesti 20.4.2018. Peli alkoi ensiksi harrasteprojektina, mutta myöhemmin perustimme Pelite Productions Oy -nimisen yrityksen pelin ympärille.

Peli on genreltään endless runner, mikä tarkoittaa, että peli jatkuu, kunnes pelaaja törmää esteeseen. Pelin perusideana on pelata rullaluistelevalla tytöllä, joka luistelee pitkin katuja väistellen ja tuhoten esteitä. Pelissä pelaajalla on valittavana monia eri taitoja, joilla hän voi tuhota tieltään esteitä, suojata itseään tai hidastaa aikaa. Pelaajalla voi olla käytössä kerralla kolme eri taitoa, jolloin hän voi kokeilla erilaisia yhdistelmiä ja löytää niistä omaan pelityyliin sopivimmat taidot.

Peliin haluttiin monenlaista kerättävää, jotta pelaajat jaksaisivat pelata sitä pitkään. Pelissä pelaaja voikin kerätä edellä mainittuja taitoja ja lisäksi asuja hahmolleen. He voivat myös edetä pelissä keräämällä kuuluisuuspisteitä, joilla he pääsevät etenemään kuuluisuustasoilla.

Myös sosiaalisuus oli tärkeä elementti, jonka halusimme mukaan. Pelaajat pystyvät liittymään klubeihin ja perustamaan omia. Joka viikko klubi saa yhteisen haasteen, johon osallistuneet klubilaiset palkitaan haasteen valmistuttua. Pelaajat pääsevät pelaamaan toisiaan vastaan myös viikoittaisissa turnajaisissa, joissa pelaajat yrittävät saada aikaiseksi viikon parhaan tuloksen.

Pelaajalla haluttiin olevan tekemistä joka päivä. Klubihaasteen lisäksi peli tarjoaa myös pelaajakohtaisia haasteita, joita voi tehdä useamman päivässä haasteiden vaikeusasteesta riippuen. Pelissä on joka päivä vaihtuvat päivittäiset tarjoukset, jotka pelaaja voi käydä tarkistamassa päivittäin esimerkiksi asujen tai hyvien taitojen toivossa. Heillä on myös mahdollista pyörittää muutaman tunnin välein onnenpyörää, josta voi saada monenlaisia palkintoja aina kokemuspisteistä pelirahaan.

KUVA 8. Pelikuva Roller Crashista

6 GAMESPARKSIN KÄYTTÖ ROLLER CRASHISSA

Suurin osa Roller Crash -pelin toiminnasta kytkeytyy GameSparksiin. Pelin eri ominaisuudet toimivat pilvikoodilla ja pelaajien profiilit on tallennettu tietokantaan, joten pelaajien eri komennot kulkevat Gamesparksin kautta. GameSparks myös lähettää aina pelaajille tietoa pelissä tapahtuneista muutoksista.

Seuraavaksi käsitellään, miten halutut ominaisuudet on toteutettu. Ensiksi syvennytään pelaajan autentikointiin, profiilin tallentamiseen, pelaajan nimeämiseen sekä pelin dataan pilvessä. Tämän jälkeen tarkastellaan peliominaisuuksia, kuten kuuluisuustasot sekä pelaajahaasteet, ja sosiaalisia elementtejä, kuten klubit, pistetaulukot ja viikoittainen turnaus. Viimeisenä käydään läpi pelaajaa palkitseva onnenpyörä sekä päivittäin vaihtuvat tarjoukset.

6.1 Pelaajan autentikointi

Pelaajan autentikoinnin haluttiin olevan helppo ja nopea prosessi, jotta pelin aloittamiseen ei menisi liikaa aikaa eikä peli jäisi pelaajalta pelaamatta sen vuoksi, että peli kysyy liikaa pelaajan tietoja. Helppouden lisäksi haluttiin, että pelaaja voi myös pelata peliä samalla profiililla usealla eri laitteella.

Päädyimme käyttämään laitekohtaista sekä Facebook-tilillä toimivaa autentikointia. Laitekohtaisessa autentikoinnissa pelaajan tili linkitetään mobiililaitteesta löytyvään uniikkiin tunnisteeseen. Tämä tapa on pelaajalle kaikista helpoin ja nopein, mutta pelaaja ei pysty käyttämään tiliään muilla laitteilla. Mikäli pelaaja hävittää tai myy puhelimensa, menee hänen pelitilinsä myös siinä mukana. Facebook-tilillä autentikointi vaatii pelaajan kirjautumisen omalla Facebook-tilillään. Tämä tapa on monen laitteen alustan peleissä kannattavaa, koska Facebook-tili ei ole rajoitettu mihinkään yksittäiseen alustaan. Pelitili myös säilyy, vaikka pelaajan laite vaihtuisi.

Pelaajan käynnistäessä pelin ensimmäistä kertaa hän voi valita laitekohtaisen tai Facebook-kirjautumisen. Laitekohtainen autentikointi tarkoittaa, että pelaaja painaa Start-painiketta. Facebook-tilillä autentikointi vaatii, että pelaaja painaa Facebook-logolla varustettua Login-painiketta. Laitekohtaisesti autentikoitu pelaaja voi jälkikäteen autentikoitua pelin asetusten kautta. Tällöin tarkistetaan, onko pelaajan kirjautumalla Facebook-tilillä jo luotua tiliä. Jos luotu tili löytyy, niin pelaaja voi valita haluaako hän kirjautua aikaisemmin luodulla tilillä vai jatkaa peliä laitekohtaisella tilillä. Jos tiliä ei löydy, pelaajan laitekohtaisesti autentikoitu tili muutetaan Facebook-tilillä autentikoiduksi tiliksi. Pelaajan autentikoimisen onnistuessa laitteelle tallennetaan lokaalisti autentikoimistapa, jotta seuraavan kerran pelaajan avatessa pelin autentikoidaan suoraan käytetyllä tavalla.

Autentikointi toteutettiin käyttämällä GameSparksin valmiita autentikoimispyyntöjä. Facebook-tilillä autentikoimista jouduttiin muokkaamaan, jotta se osaisi asetusten kautta kirjautuessa tarkistaa, onko pelaajalla jo aikaisempi tili olemassa. Facebook-tilillä kirjautumista varten peliin tuli myös integroida Facebookin oma SDK eli ohjelmistokehityspaketti, jonka avulla Facebookin ominaisuuksia voidaan käyttää. Tätä ohjelmistokehityspakettia käyttäen Facebookin oma kirjautumisikkuna saatiin pelaajalle näkyviin pelissä.

6.2 Pelaajan profiilin tallennus

Kaikki pelaajan tieto haluttiin tallentaa pilveen. Pelimme ominaisuudet myös vaativat pelaajan tietojen muokkaamista pilvikoodin avulla, ja jotta pilvikoodi pääsisi näihin tietoihin käsiksi, pitää niiden olla tallennettuna GameSparksiin. Näin pelaajan tiedot ovat aina turvassa pilvessä.

Pelaajasta pitää tallentaa muun muassa seuraavat tiedot:

- nimi ja tunniste
- rahamäärä ja kultaharkkomäärä
- kerätyt taidot
- kerättyjen taitojen tasot ja kokemuspisteet
- käytössä olevat taidot
- kaikki kerätyt asusteet
- käytössä olevat asusteet

- pelaajahaasteen edistyminen
- kuuluisuustaso, kuuluisuuspisteet ja kerätyt kuuluisuuspalkinnot
- onnenpyörän tiedot
- klubitiedot
- paras pelitulos
- viikottaisen turnauksen tulos
- ostetut päivittäiset tarjoukset
- ostetut pelinsisäiset ostokset
- statistiikat analytiikkaa varten
- Push-rekisteröinti.

Tietojen tallentamiseen päädyttiin käyttämään GameSparksin MongoDB-tietokantaa. Tietokannasta on helppo hakea, lisätä, muokata ja poistaa tietoa pilvikoodin kautta. Kaikkien tapahtumien mennessä pilvikoodin kautta estetään huijaaminen ja pelaajan tiedot ovat aina samat pilvessä kuin pelaajan omalla laitteella.

Pelaajan profiilin tallennus toteutettiin luomalla tietokantaan oma kokoelma, johon pelaajien profiilit tallennetaan omia dokumentteinaan. Pelaajan tehdessä jonkun toiminnon pelissä, se käsitellään aina pilvikoodissa ja pelaajan profiilia muokataan, jos on tarve. Muokkaamisen jälkeen peliin lähetetään tieto, kuinka pelaajan profiili muuttui.

6.3 Pelaajan nimi

Halusimme, että jokaisella pelaajalla on itse annettu uniikki nimimerkki ja että pelin aloittaminen olisi helppoa ja nopeaa. Monia pelaajia harmittaa pelin alussa kokeilla montaa eri nimeä, koska haluttu nimi on jo käytössä toisella pelaajalla. Tätä haluttiin välttää, koska pelaaja voisi mahdollisesti turhautua ja lopettaa pelaamisen, kun ei saa haluamaansa nimeä.

Ongelma päädyttiin ratkaisemaan sallimalla pelaajien antaa itselleen mikä tahansa nimi ja asettamalla pelaajan valitseman nimen loppuun nelinumeroinen tunniste. Tällä menetelmällä pelaaja voi antaa nimekseen ”Matti” ja hänen pelinimekseen tulee esimerkiksi ”Matti#1357”. Näin samannimisiä pelaajia voi olla pelissä 9999 kappaletta ennen kuin pelaajaa vaaditaan valitsemaan joku toinen nimi.

Ominaisuus toteutettiin hakemalla tietokannassa samannimiset pelaajat kuin pelaajan haluama nimi. Haun jälkeen saadaan tietoon mitkä lopputunnisteet ovat jo käytössä ja tämän jälkeen pelaajan nimeen arvotaan lopputunniste, joka ei ole vielä varattu.

6.4 Pelin data pilvessä

Pelin data haluttiin pilveen, jotta sitä olisi nopeampi muokata ja jotta sen käyttäminen pilvikoodissa olisi helpompaa. Pelin datan muokkaukseen tarvittiin helposti käytettävät hallintapaneelit, jotta kaikki kehitystiimistä pystyvät muokkaamaan pelin dataa.

Pelin data tuli myös saada ladattua helposti pelaajien laitteille, koska sitä tarvitaan pelissä. Dataa ei voida kuitenkaan joka kerta pelin käynnistäessä ladata laitteelle: tämä vie liikaa kaistaa mobiililaitteilla ja käyttää turhaan resursseja GameSparksissa. Uusissa peliversioissa voi myös olla dataa, jota aiemmissa versioissa ei ollut. Dataa voidaan myös käyttää niissä eri tavalla, joten pelidata tulee palauttaa peliversion mukaan. Pelin tulee myös toimia monella peliversiolla samaan aikaan live-alustalla ja ymmärrettävä, milloin dataa on muokattu ja uuden datan lataaminen on tarpeellista.

Seuraavat pelin datat piti laittaa pilveen:

- taidot
- asusteet
- kulutustavarat (esim. kuuluisuusjuomat)
- arkut
- pelaajien haasteet
- klubien haasteet
- haasteiden tehtävät
- kuuluisuustasot ja -palkinnot
- viikoittaisten turnausten palkinnot sijoituksen mukaan
- onnenpyörien palkinnot
- estekartat (käytetään pelissä esteiden luomiseen)
- asetukset (peli-, kauppa- ja yleiset asetukset)
- pelin sisäiset ostokset ja muut tuotteet.

Jokaiselle datatyypille päädyttiin tekemään tietokantaan oma kokoelma, johon datat tallennetaan dokumentteina. Kaikkien kokoelmien, jossa on pelidataa, tuli olla meta-kokoelmia, jotta ne siirtyisivät snapshottien mukana alustalta toiselle. Muutoksien helpottamiseksi jokaisella datatyypillä on erikseen luotu hallintapaneeli, joka tekee niiden muokkaamisesta helppoa ja nopeaa.

```
{
    "_id": {
        "$oid": "xxxxxxxxxxxxxxxxxxxxxxxx"
    },
    "id": "helmet_bunny",
    "clientVersion": 108,
    "name": "Bunny Ears",
    "type": "helmet",
    "icon": "helmet_bunny",
    "prefabs": [
        {
            "prefab": "helmet_bunny"
        },
        {
            "prefab": "hair_02"
        }
    ]
}
```

KUVA 9. Esimerkki pelin datasta dokumenttina MongoDB-tietokannassa

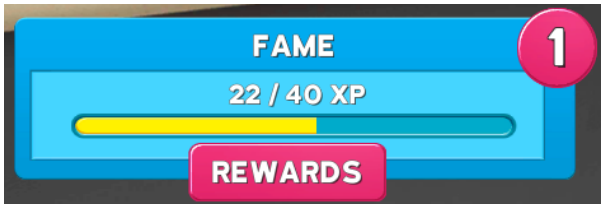
Datojen versiointiin suunniteltiin systeemi, jossa jokaisella pelin datalla on oma peliversio (*client version*), jotta tiedetään mihin peliversioon ne kuuluvat. Datatyypeillä on sitten peliversioittain dataversio (*data version*) omassa tietokannassaan. Hallintapaneelit ovat toteutettu niin, että dataversiota kasvatetaan aina automaattisesti, kun niiden kautta tehdään muutoksia dataan. Tästä jää samalla lokimerkintä, joka helpottaa muutosten seurannassa. Tällä systeemillä peli osaa verrata lokaalin datan versioita pilvessä oleviin ja päivittää ne, jos pilviversio on uudempi kuin lokaaliversio. Pelin alkaessa peli lähettää pyynnön GameSparksiin, joka tarkistaa lokaalidatojen tilan. Peli lataa myös aina vain oman peliversion datat laitteelle.

Aina uuden peliversion kehityksen alkaessa täytyy versiolle luoda omat datat. Tämä tarkoittaa edellisen peliversion datojen kopioimista uuteen peliversioon. Näin eri peliversioiden datat ovat täysin eristetyt toisistaan eikä datan muokkaaminen vaikuta, kuin yhden peliversion toimintaan. Peliversioiden datojen luonti ja poistaminen sekä yleinen datan hallinta onnistuvat sille erikseen tehdystä hallintapaneelista. Datojen muuttuessa ilmoitetaan kaikille pelissä oleville laitteille muutoksista ja lokaalidatat päivitetään sen mukaan.

6.5 Kuuluisuustasot

Halusimme, että pelaaja tuntee etenevänsä koko ajan pelatessaan peliä. Tästä johtuen kehitimme peliin kuuluisuustasot (*fame levels*). Kuuluisuustasojen saamiseen pelaajan tarvitsee saada kuuluisuuspisteitä, joita saa pelaamalla peliä ja kuuluisuusjuomilla. Jokaisesta tasosta pelaaja saa palkinnon.

Ominaisuus toteutettiin tallentamalla pelaajan kuuluisuustasot, -pisteet ja lunastetut palkinnot pelaajan profiiliin. Kuuluisuustasoja päivitetään aina pelaajan suorittaessa pelisession loppuun. Pelaajan lunastaessa palkinnon tallennetaan hänen profiiliin tieto lunastetusta palkinnosta. Pelaajan kuuluisuuspisteitä päivitetään aina pelisession päättyessä ja pelaajan tasoa kasvatetaan, mikäli pisteet ylittävät vaadittavan määrän. Kuuluisuustasojen pistevaatimuksia pidetään omassa tietokannassaan.



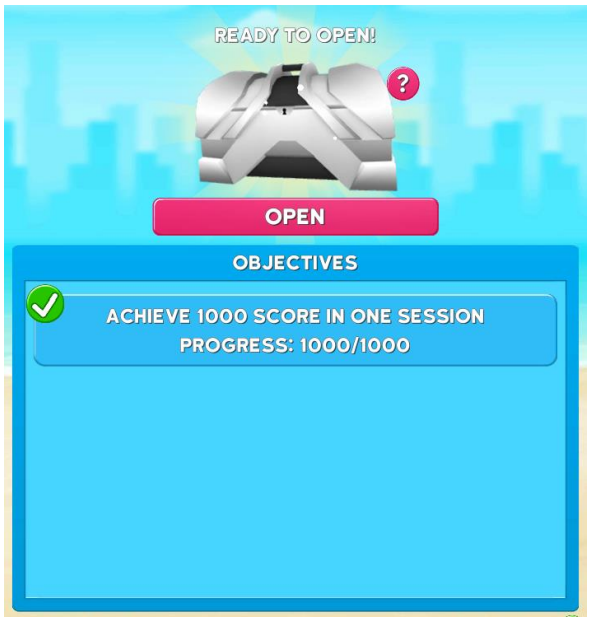
KUVA 10. Kuuluisuustasot

6.6 Pelaajahaasteet

Pelaajalla tulee olla tehtävää ja syy palata peliin joka päivä. Tämän ajatuksen pohjalta syntyi pelaajahaasteet (*player challenges*), jossa pelaajan pitää esimerkiksi tuhota tietty määrä esteitä. Haasteita on kolmea eri tasoa ja ne voivat sisältää useita eri tehtäviä. Pelaajan saadessa lisää kuuluisuustasoja hän avaa mahdollisuuden uusiin haasteisiin.

Haasteen suorittaminen palkitaan arkulla. Mitä vaikeampi haaste ja useampi tehtävä, sitä paremman arkun pelaaja saa, mutta sen avaamisaika myös pitenee. Arkuissa on aikarajoitus, jotta pelaaja ei avaa kaikkea pelisisältöä liian nopeasti ja hänellä on syy palata peliin myöhemmin takaisin. Pelaaja voi halutessaan lyhentää arkun avaamisaikaa katsomalla mainoksen tai maksamalla pelin sisäistä valuuttaa. Mainoksen katsomista tai pelin sisäistä valuuttaa voidaan käyttää myös haasteen vaihtamiseen, jos pelaaja kokee sen liian haastavaksi tai ei vaan halua tehdä kyseistä haastetta.

Ominaisuus toteutettiin tallentamalla pelaajan haasteen tiedot pelaajan profiiliin tietokannassa. Haasteen etenemistä päivitetään aina pelaajan pelatessa yhden pelisession loppuun. Pelaajan suoritettua kaikki haasteen tehtävät ajoitetaan palkintoarkun avaus GameSparksin Scheduler-funktiolla. Arkun ollessa valmis avaamiseen pelaajalle lähetetään siitä tieto ja arkun avautuessa pelaaja saa uuden haasteen. Arkun avaamisaikaa lyhennettäessä arkun avaamisen suorittava moduuli uudelleen ajastetaan 30 minuuttia lähemmäksi.



KUVA 11. Pelaajahaasteet

6.7 Pelaajien klubit

Halusimme peliin klubit, joiden avulla pelaajat voivat olla sosiaalisessa vuorovaikutuksessa keskenään. Pelaajat voivat esimerkiksi yrittää saada toinen toistaan parempia pelituloksia ja kiivetä klubin palkintopalleille tai tehdä yhdessä klubihaasteita ja saada niistä palkintoja. Klubeja tuli olla sekä julkisia että yksityisiä.

Klubien nimiin ei tule lopputunnisteita vaan ne ovat uniikkeja, jotta klubit olisi helpommin erotettavissa. Päätimme myös, että yksityisiin klubeihin liitytään omistajan asettamalla salasanalla. Salasana on helppo antaa sitten pelaajille, jotka klubiin halutaan. Riskinä on tietysti, että salasana päätyy sellaisten ihmisten käsiin joita klubiin ei haluta, mutta tällaisissa tilanteissa salasana voidaan vaihtaa ja klubista potkia pelaajat, joita sinne ei haluta.

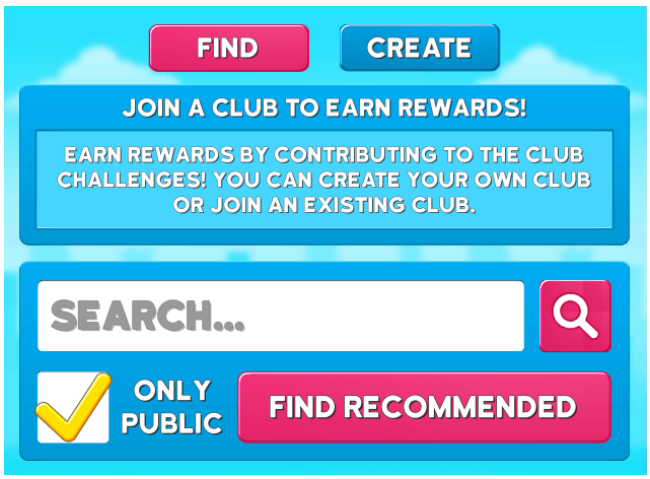
GameSparksissa on valmis Teams-työkalu klubien tapaisten ominaisuuksien toteutukseen, mutta klubiin haluttiin paljon enemmän toimintoja mitä Teams-työkalu tarjosi. Tästä syystä päädyttiin tekemään oma klubijärjestelmä pilvikoodia käyttäen. Klubeille tehtiin oma runtime-kokoelma, jonne klubit tallennetaan dokumentteina. Klubin dokumentti sisältää klubin tiedot ja jäsenluettelon.

KUVA 12. Klubinäkymä pelissä

6.7.1 Klubiin liittyminen

Klubiin liittyminen haluttiin tehdä mahdollisimman helpoksi lisäämällä ominaisuus, joka suosittelee pelaajalle sopivia klubeja. Pelaajalle suositellaan klubeja vertaamalla pelaajan parasta tulosta klubijäsenten keskiarvotulokseen. Näin pelaaja saa listan klubeista, joissa on samantasoisia pelaajia ja jotka ovat julkisia, eli niihin voi liittyä ilman salasanaa.

Hakiessa suositeltuja klubeja etsitään klubeja, joissa jäsenten keskimääräinen paras tulos on lähimpänä pelaajan omaa parasta tulosta. Tämän jälkeen löydetyt klubit järjestetään pelaajan tuloksen mukaan ja ne lähetetään takaisin peliin. Haun jälkeen pelaaja voi tarkastella näitä löydettyjä klubeja ja halutessaan liittyä johonkin niistä.



KUVA 13. Klubisivu pelissä, kun ei pelaaja ei ole vielä klubissa

```
function SearchRecommendedClubs(playerData, maxNumberOfReturnedClubs) {
  var clubsWithGreaterAverageScore = Spark.runtimeCollection("clubs").find(
    {
      "averageMemberScore" : { "$gte": Number(playerData.bestScore) },
      "publicAccess": true
    }
  )
  .sort({"averageMemberScore":1})
  .limit(Number(maxNumberOfReturnedClubs))
  .toArray();

  var clubsWithLowerAverageScore = Spark.runtimeCollection("clubs").find(
    {
      "averageMemberScore" : { "$lt": Number(playerData.bestScore) },
      "publicAccess": true
    }
  )
  .sort({"averageMemberScore":1})
  .limit(Number(maxNumberOfReturnedClubs))
  .toArray();

  var allRecommendedClubs = [];
  for(var i = 0; i < clubsWithGreaterAverageScore.length; i++){
    allRecommendedClubs.push(clubsWithGreaterAverageScore[i]);
  }

  for(var i = 0; i < clubsWithLowerAverageScore.length; i++){
    allRecommendedClubs.push(clubsWithLowerAverageScore[i]);
  }

  allRecommendedClubs.sort(
    function(a, b)
    {
      return Math.abs(playerData.bestScore-a.averageMemberScore) -
        Math.abs(playerData.bestScore-b.averageMemberScore);
    }
  );

  return allRecommendedClubs;
}
```

KUVA 14. Pilvikoodissa oleva funktio, joka hakee suositellut klubit pelaajalle

Halusimme, että klubeja on mahdollista hakea myös nimen perusteella. Tätä voidaan käyttää esimerkiksi silloin, kun pelaajan kaverit ovat tehneet klubin johon pelaaja haluaisi liittyä. Nimellä hakemista on myös pakko käyttää, jos haluaa nähdä yksityiset klubit. Haun pitäisi myös löytää klubi vajaalla hakusanalla, jolloin riittää, että pelaaja muistaa vain osan nimestä.

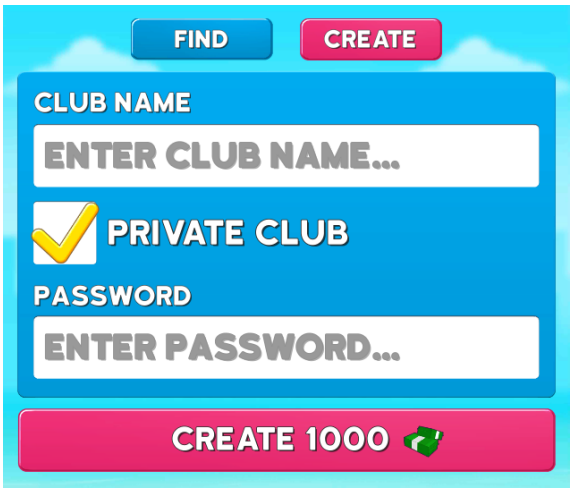
Klubeja hakiessa tarkistetaan ensin, onko pelaajan käyttämä hakusana oikeanmuotoinen ja virhetilanteessa lähetetään vastaus, kuinka korjata hakusanaa. Hakusanan tulee olla 2–20 merkkiä pitkä ja se ei saa sisältää erikoismerkkejä. Hakusanan ollessa kelvollinen haetaan maksimissaan 10 klubia, jotka hakusanalla löydetään.

```
function SearchClubs(searchTerms, onlyPublicClubs) {
  var result = {};

  if(searchTerms.length > 1){
    if(searchTerms.length < 21) {
      var letterNumber = /^[0-9a-zA-Z]+$/;
      if(searchTerms.match(letterNumber)) {
        result.foundClubDatas = Spark.runtimeCollection("clubs").find(
          {
            "name" : {"$regex": String(searchTerms), "$options": "i"},
            "publicAccess": Boolean(onlyPublicClubs)
          }
        ).limit(10).toArray();
        result.success = true;
      }
      else {
        result.error = "That name contains special characters.";
        result.success = false;
      }
    }
    else {
      result.error = "Too long search terms. (Max 20 characters)";
      result.success = false;
    }
  }
  else {
    result.error = "Too short search terms. (Min 2 characters)":
    result.success = false;
  }

  return result;
}
```

KUVA 15. Pilvikoodissa oleva funktio, joka hakee klubin nimellä

6.7.2 Klubin perustaminen

Haluttiin, että kuka tahansa pelaaja pystyy halutessaan luomaan oman klubinsa. Klubin perustaminen maksaa pelaajalle pelin sisäistä valuuttaa. Klubia perustaessa tarkistetaan, että samannimistä klubia ei vielä ole olemassa, nimi ja salasana ovat oikean pituisia eikä niissä ole erikoismerkkejä sekä pelaajalla on tarpeeksi pelin sisäistä valuuttaa luoda klubi. Ehtojen täyttyessä klubi luodaan ja pelaaja siirretään klubinäkymään, mutta jos nämä ehdot eivät täyty, niin pelaajalle ilmoitetaan mikä meni vikaan. Luonnin jälkeen klubiin voi liittyä muut pelaajat.



KUVA 16. Oman klubin luonti

6.7.3 Klubin hallinta

Klubeista haluttiin helppoja hallita, joten klubijäsenille kehitettiin arvot, joilla on eri oikeuksia klubin hallintaan. Klubin jäsenellä voi olla yksi kolmesta eri arvosta: omistaja, moderaattori ja jäsen.

Päätettiin, että omistaja pystyy tuhoamaan klubin, muuttamaan klubin asetuksia, vaihtamaan pelaajien arvoja sekä potkimaan kaiken arvoisia pelaajia klubista. Moderaattori pystyy näkemään yksityisen klubin salasanan ja potkimaan jäsen-arvoisia pelaajia klubista. Jäsenellä ei ole hallintaoikeuksia klubissa.

Ominaisuus toteutettiin lisäämällä tietokannassa klubidokumentin jäsenluetteloon jäsenen arvo. Hallintaominaisuuksia käytettäessä tarkastetaan aina ensiksi pelaajan arvo klubissa ennen toiminnan suoritusta.

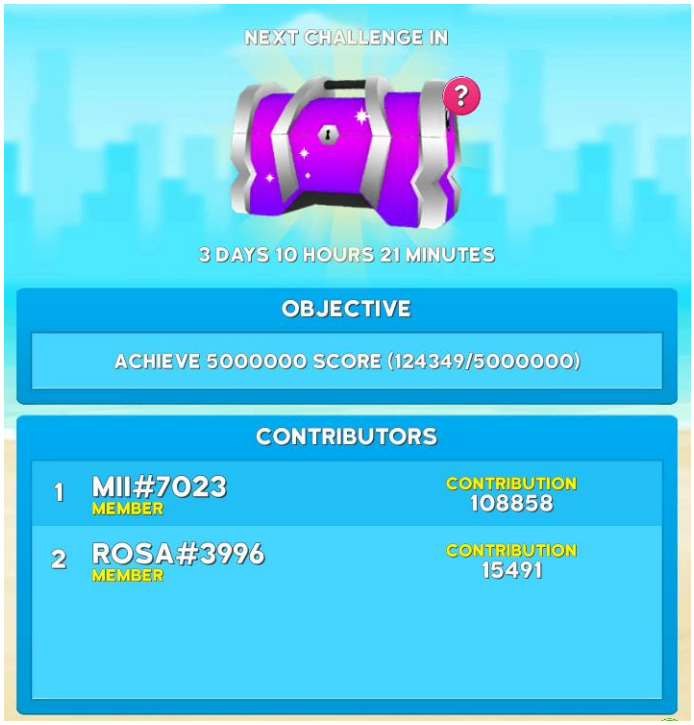
KUVA 17. Klubin hallinta klubin omistajan näkökulmasta

6.7.4 Klubihaasteet

Halusimme, että klubin jäsenet pystyvät työskentelemään yhteisen tavoitteen eteen. Tämän vuoksi kehitettiin klubihaaste (*club challenge*), jossa klubin jäsenillä on viikko aikaa suorittaa tehtävä. Klubihaasteen tehtävä on paljon suurempi kuin pelaajahaasteessa ja tarvitsee useamman osallistujan. Kaikki klubihaasteeseen osallistuneet palkitaan arkulla, jos tehtävä suoritetaan annetun ajan sisällä. Pelaajan on lunastettava palkinto ennen uuden haasteen alkua.

Klubihaasteen yhteydessä näkyy lista niistä pelaajista, jotka ovat osallistuneet tehtävän suorittamiseen. Klubin jäsen ei ole automaattisesti mukana tehtävässä vaan hänen on osallistuttava siihen ennen tehtävän valmistumista. Viikon haaste on joka klubilla sama ja pelaaja voi lunastaa vain yhden klubihaastepalkinnon viikossa, vaikka hän osallistuisi usean klubin haasteen suorittamiseen. Ainoastaan haasteeseen osallistuneet klubin jäsenet voivat lunastaa arkun haasteen valmistuttua.

Ominaisuus toteutettiin tallentamalla klubin haasteen tiedot ja eteneminen klubin profiiliin. Klubin haasteen eteneminen saadaan laskemalla osallistujien kontribuutiomäärät yhteen. Pelaajan tehdessä jotain klubihaasteen eteen hänen kontribuutionsa tallennetaan klubihaasteen osallistujalistaan. Kun kontribuutioiden yhteenlaskettu määrä ylittää haasteen vaatimuksen, haaste merkitään suoritetuksi ja klubin jäsenille ilmoitetaan haasteen valmistumisesta Push-ilmoituksella.



KUVA 18. Klubihaasteet

6.8 Pistetaulukot

Halusimme peliimme pistetaulukot (*high score*) pelaajille ja klubeille. Pelaajien pistetaulukossa näytetään paremmuusjärjestyksessä 100 pelaajaa, jotka ovat tehneet parhaat tulokset pelissä. Klubien pistetaulukossa näytetään 100 parasta klubia. Klubien pisteet lasketaan laskemalla jäsenien parhaat tulokset yhteen.

GameSparksissa on sisäänrakennettu Leaderboards-työkalu, jonka avulla pistetaulukoita voidaan tehdä helposti. Oma pistetaulukkosysteemi todettiin kuitenkin tarpeelliseksi, koska pelaajan pisteet lasketaan pilvikoodissa niin, että Leaderboards-työkalun käyttö olisi epäloogista ja raskasta.

Pistetaulukot on tallennettu tietokannassa omaan kokoelmaansa. Kokoelmassa on omat dokumentit pelaajien ja klubien tuloksille. Resurssien säästämiseksi pistetaulukkoa ei päivitetä reaaliajassa vaan se päivitetään omaan tietokantaansa joka minuutti. Tämä tapahtuu ajamalla oma koodimoduuli GameSparksin GS_MINUTE-skriptissä, joka on GameSparksin sisäinen joka minuutti ajettava skripti. Tässä koodimoduulissa haetaan pelaaja- ja klubikokoelmista kaikki dokumentit, jotka sitten järjestetään tulosten mukaan. Suorituksen jälkeen pistetaulukot tietokannassa päivitetään uusilla järjestetyillä listoilla.

KUVA 19. Pistetaulukot

6.9 Viikoittainen turnaus

Peliin haluttiin pelaajien välistä kilpailua, josta pelaajien on mahdollista saada palkintoja. Tähän tarpeeseen keksimme viikoittaisen turnauksen (*weekly tournament*), jossa pelaajat pääsevät kilpailemaan toisiaan vastaan tuloksissa viikoittain. Viikon lopussa pelaaja palkitaan hänen sijoituksensa mukaan erilaisilla arkuilla. Mitä parempi sijoitus, sitä parempi arkku, mikä motivoi pelaajia pyrkimään parhaille sijoille. Turnaus luo myös pelaajien välille sosiaalisen kokemuksen ja he pääsevät näkemään ja olemaan eräänlaisessa kanssakäymisessä myös klubinsa ulkopuolella olevien pelaajien kanssa.

Viikoittaisen turnauksen tiedot ovat tallennettu omaan tietokantaansa, jossa on aktiivinen ja edellinen turnaus omina dokumentteinaan. Aktiivisessa dokumentissa on sen hetkisen turnauksen tiedot kuten pelaajien sijoitukset. Edellisessä dokumentissa on edellisviikon turnauksen tiedot sekä tieto siitä onko sijoittunut pelaaja lunastanut palkintonsa. Pelaajien profiileista löytyy heidän viikoittainen paras tuloksensa, joista päivitetään aktiiviseen dokumenttiin 50 parasta tulosta GS_MINUTE-skriptin avulla. Turnausta ei voi päivittää reaaliajassa samoista syistä kuin pistetaulukoita.

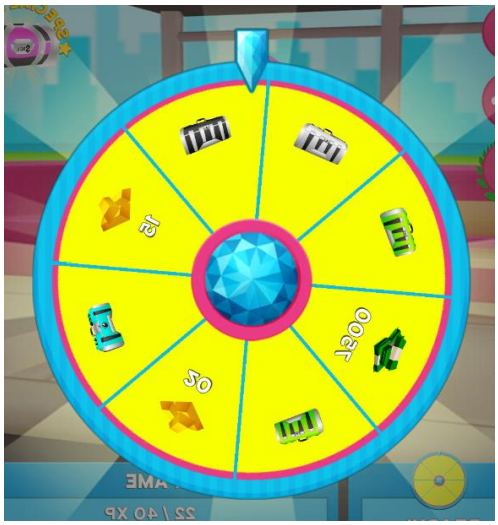
Turnauksen loppuminen tapahtuu GameSparksin Scheduler-funktion avulla. Turnauksen loputtua ajastetaan seuraavan turnauksen lopetusaika ja siirretään aktiiviseen dokumentin tiedot edellisviikon dokumenttiin. Tällöin myös pelaajien profiileista nollataan heidän paras viikoittainen tulos. Kaikille listoille päässeille pelaajille ilmoitetaan palkinnosta Push-ilmoituksella. Mikäli pelaaja ei ole lunastanut edeltävän viikon palkintoa viikko turnajaisten päättymisen jälkeen, se jää häneltä saamatta.

KUVA 20. Viikoittainen turnaus

6.10 Onnenpyörä

Halusimme peliin onnenpyörän (*fortune wheel*), josta pelaaja voi voittaa erilaisia palkintoja. Onnenpyörää voi pyöräyttää katsomalla mainoksen tai pelin sisäistä valuuttaa vastaan. Onnenpyörän palkinnot vaihdetaan päivittäin ja sitä voi pyöräyttää neljän tunnin välein. Aina kun onnenpyörää voi pyöräyttää uudestaan ilmoitetaan siitä pelaajalle Push-ilmoituksella. Onnenpyörän palkinnoilla on eri todennäköisyydet.

Onnenpyörät ja palkintojen todennäköisyydet on tallennettu omaan tietokantaansa. Pyöräyttäessä onnenpyörää haetaan nykyinen onnenpyörä ja sen sisältämien palkintojen todennäköisyydet. Todennäköisyyksillä arvotaan pelaajan saama palkinto. Arvonta suoritetaan pilvikoodissa, joten pelaaja ei pääse vaikuttamaan siihen mitenkään. Arvonnan jälkeen sisältö ja pelaajan saama palkinto palautetaan peliin. Tämän jälkeen pelin onnenpyörä lähtee käyntiin ja pysähtyy pelaajalle arvotun palkinnon kohdalle. Onnenpyörän vaihtuminen on toteutettu Scheduler-funktiota apuna käyttäen.

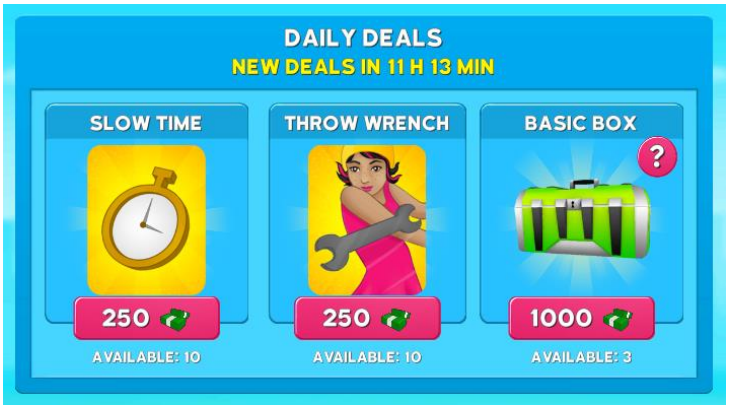


KUVA 21. Onnenpyörä

6.11 Päivittäiset tarjoukset

Halusimme peliin päivittäisen tarjoukset (*daily deals*), jotta pelaaja voi käyttää saamansa pelin sisäistä valuuttaa ostaakseen haluamiaan taitoja, asusteita ja arkkuja. Päivittäisiä tarjouksia on kolme ja ne vaihtuvat joka päivä. Joitain tarjouksia pelaaja voi ostaa useamman kerran, mutta ostokertoja on rajoitettu ja hinta nousee jokaisen oston jälkeen. Päivittäiset tarjoukset ovat jokaiselle pelaajalle aina samat.

Aktiiviset päivittäiset tarjoukset on tallennettu omaan tietokantaansa. Pelaajan päivittäisten tarjousten ostokerrat tallennetaan pelaajan profiiliin, jotta pelaajalle voidaan näyttää oikea hinta. Päivittäiset tarjoukset vaihdetaan Scheduler-funktiota apuna käyttäen ja samalla nollataan pelaajien profiileista ostokerrat. Erilaiset tarjouskokonaisuudet on tallennettu omaan tietokantaansa, joista aktiiviset tarjoukset aina vaihdettaessa arvotaan.



KUVA 22. Päivittäiset tarjoukset

7 POHDINTA

Opinnäytetyön tavoitteena oli etsiä ratkaisuja Roller Crash -mobiilipelin tarpeisiin ja toteuttaa pelin backend GameSparks-palvelun avulla. GameSparks vastasi tarpeitamme Roller Crash -pelin backend-ratkaisuna ja sillä saatiin toteutettua kaikki tarvitsemamme ominaisuudet toimiviksi.

Työskentely Gamesparksin parissa ei ollut aina helppoa. GameSparksissa on reilun käytön rajat, joilla tarkoitetaan, että tiedonsiirtoa ja pyyntöjä ei saa olla liikaa kuukausittaista pelaajaa kohden. Pilvikoodin suorittaminen pitää myös tapahtua tietyn ajan sisällä. Nämä rajat aiheuttivat haasteita, mutta optimoimalla pilvikoodia sekä pelin ja backendin tiedonvaihtoa, niistä selvittiin. Tavallaan nämä rajat myös auttoivat minua tekemään kaikesta mahdollisimman kevyttä ja tehokasta, joka lisäsi suorituskykyä. GameSparksin dokumentaatio oli myös ajoittain todella vähäistä, mikä oli myös yksi syy tämän opinnäytetyön aiheen valitsemiseen. Jouduin selvittämään asioiden toimintaa kokeilemalla ja ottamalla GameSparksiin yhteyttä. Tämä söi kehitysaikaa, mutta loppujen lopuksi kaikkien asioiden toiminta selvisi aina jotakin kautta.

Hyvien ja huonojen puolien yhteenvetona GameSparks-palvelu oli toimiva kokonaisuus, joka tarjosi pelin tarpeille juuri sen mitä siltä pyydettiin. Varsinkin erilaiset ympäristön alustat mahdollistivat sujuvan pelinkehityksen backendissä, kun muutoksia ei tarvinnut tehdä suoraan liveversioon vaan kehitysversio oli oma eristetty alustansa. Palvelulla saatiin myös rakennettua helppokäyttöiset hallintapaneelit, joilla jokainen pelitiimin jäsen pystyy tekemään muutoksia peliin ja sen asetuksiin joutumatta käyttämään erilaisia komentoja esimerkiksi tietokantamuutoksiin.

Roller Crash -peliä kehitettiin 1,5 vuotta ja tämän aikana opin paljon pelin backendin vaatimuksista ja sen toteuttamisesta. Ennen tätä projektia minulla ei ollut kokemusta pelien backendistä, mutta nyt minulla on melko laaja tietämys siitä, miten asiat toimivat. Tämä on taito, mistä on varmasti hyötyä myös tulevaisuudessa ja johon voin paneutua myös entistä syvemmin tulevissa projekteissa.

GameSparksilla on vielä paljon annettavaa eikä sitä ole vielä kulutettu loppuun peliin jo tehtyjen ominaisuuksien kanssa. En ole vielä käyttänyt GameSparksin reaaliaikaista moninpelipilvikoodia. Tämä olisi hyödyllinen tutkimuskohde jatkokehityksen kannalta, koska sen avulla voisi toteuttaa muun muassa erilaisia reaaliaikaisia moninpeliominaisuuksia, jotka voivat olla hyvin tarpeellisia, jos Roller Crash -peliin halutaan tuoda lisää sosiaalisia ominaisuuksia ja mahdollistaa esimerkiksi kilpaileminen toisia pelaajia vastaan reaaliajassa.

Opinnäytetyöstä olisi voinut ottaa tutkivamman otteen, mutta koska lähestymistapa oli case-tutkimus ja aiheeseen liittyvää hyvää materiaalia oli lähes mahdotonta löytää, on työ mielestäni onnistunut. Erilaisiin ominaisuuksiin olisi voinut syventyä enemmänkin, mutta halusin pitää toimintojen esittelyn pintapuolisena sen takia, että tietoa ei voida väärinkäyttää Roller Crash -pelin toiminnan häiritsemiseksi. Ominaisuuksien toteuttamisen syvempi tarkastelu olisi myös laajentanut opinnäytetyötä liikaa. Tätä samaa aihetta voisi käsitellä jatkossa jonkun eri genren pelin näkökulmasta, jossa on erilaiset vaatimukset. Tällainen voisi olla esimerkiksi PC-peli, johon tarvitaan reaaliaikaisia moninpeliominaisuuksia.

LÄHTEET

Cloud Code. Luettu 4.4.2018.
https://docs.gamesparks.com/documentation/key-concepts/cloud-code.html

Firebase Pricing Plans. Luettu 26.4.2018.
https://firebase.google.com/pricing/

GameSparks FAQ. Luettu 4.4.2018.
https://www.gamesparks.com/blog/gamesparks-faq/

GameSparks Integrations. Luettu 4.4.2018.
https://www.gamesparks.com/product/#integrations

GameSparks Joins the Amazon Family. Luettu 4.4.2018.
https://www.gamesparks.com/blog/gamesparks-joins-amazon/

GameSparks Pricing. Luettu 4.4.2018.
https://www.gamesparks.com/pricing/

GameSparks Showcase. Luettu 11.4.2018.
https://www.gamesparks.com/showcase/

GitHub or Bitbucket Synchronization with Cloud Code Import / Export.
Luettu 4.4.2018.
https://docs.gamesparks.com/tutorials/third-party-integrations/synchronizing-cloud-code-with-github-or-bitbucket.html

Griffin, J. 2014. The power of the cloud: Enchancing your game experience with Cloud Services. Seminaaritallenne. Game Dev Camp Portugal 2014. Katsottu 11.4.2018. https://channel9.msdn.com/Events/GameDevCampPT/Game-Dev-Camp-Portugal-2014/The-power-of-the-Cloud-enhancing-your-game-experience-with-Cloud-Services-DEV-Track-1

Indie & Student Programme FAQ. Luettu 4.4.2018.
https://www.gamesparks.com/indie-student-programme-faq/

PlayFab Pricing. Luettu 26.4.2018.
https://playfab.com/pricing/

Trish, D. 2014. Cloud-based sparks let the games begin. Luettu 11.4.2018.
https://www.irishexaminer.com/business/features/cloud-based-sparks-let-the-games-begin-275388.html

Vaish, G. 2013. Getting Started with NoSQL: Your Guide to the World and Technology of NoSQL. Packt Publishing. Birmingham-Mumbai.

Wong, S. 2017. What it means to treat games as a service. Luettu 11.4.2018.
http://www.alistdaily.com/strategy/means-games-service/