Suvi Hämäläinen

Verkkopohjaisen pistetaulukon kehittäminen mobiilipeliin

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintätekniikan tutkinto-ohjelma

Insinöörityö

10.9.2024

Tiivistelmä

Tekijä:	Suvi Hämäläinen
Otsikko:	Verkkopohjaisen pistetaulukon kehittäminen mobiilipeliin
Sivumäärä:	45 sivua
Aika:	10.9.2024
Tutkinto:	Insinööri (AMK)
Tutkinto-ohjelma:	Tieto- ja viestintätekniikka
Ohjaajat:	Osaamisaluejohtaja Janne Salonen

Insinöörityön tavoitteena oli suunnitella ja toteuttaa verkkopohjainen pistetaulukko Unity-pelimoottorilla kehitettyyn The Ultimate Rock & Metal Quiz -mobiilipeliin. Työn alussa esiteltiin pelin paikallinen pistetaulukko ja käytiin läpi verkkopohjaiseen pistetaulukkoon siirtymisen hyötyjä. Lisäksi tutkittiin pistetaulukon mahdollisia toteutustapoja sekä mobiilipelialan yleisiä käytäntöjä.

Projekti sisälsi verkkopohjaisen pistetaulukon suunnittelun ja toteutuksen Unityssa. Työhön kuului olennaisena osana tietokantapalvelun valinta ja käyttöönotto sekä sen integroiminen peliin. Myös pelin käyttöliittymää päivitettiin pistetaulukon lisäämisen yhteydessä.

Työssä vertailtiin eri pilvipalveluja, joista valittiin Firebase sen helppokäyttöisyyden, luotettavuuden ja skaalautuvuuden takia. Työn edetessä todettiin, että Firebase oli hyvä valinta ja sen tarjoamat palvelut ja toiminnallisuudet sopivat hyvin mobiilipelien kehitykseen.

Projektissa kiinnitettiin huomiota myös mahdollisiin riskeihin, joita käyttäjien syöttämä tieto voi aiheuttaa. Peliin lisättiin rajoituksia ja käyttäjien syöttämien nimimerkkien tarkistus riskien minimoimiseksi.

Insinöörityön lopputuloksena syntyi toimiva verkkopohjainen pistetaulukko, joka tarjoaa käyttäjille mahdollisuuden kilpailla muiden pelaajien kanssa. Pistetaulukko parantaa pelikokemusta, mikä voi lisätä The Ultimate Rock & Metal Quiz -pelin käyttäjämääriä ja sitä kautta myös mainostuloja.

Avainsanat: mobiilipeli, pelinkehitys, Unity, Firebase

Abstract

Author:	Suvi Hämäläinen
Title:	Developing an Online Leaderboard for a Mobile Game
Number of Pages:	45 pages
Date:	10 September 2024
Degree:	Bachelor of Engineering
Degree Programme:	Information and Communication Technology
Supervisors:	Janne Salonen, Director of school

The objective of this thesis project was to design and develop an online leaderboard for a mobile game called The Ultimate Rock & Metal Quiz. The game development work was done using the Unity game engine. The thesis began by introducing the local leaderboard of the game and exploring the benefits of updating it to an online leaderboard. Potential implementation methods and common practices and solutions used in the mobile game industry were also discussed.

The project involved the planning and implementation of the online leaderboard in Unity. Selecting and deploying a database service and integrating it into the game was a key part of the work. The game's user interface was also updated when adding the leaderboard.

After comparing various cloud services, Firebase was chosen due to its ease of use, reliability, and scalability. As the project progressed, Firebase proved to be a good choice because it offers services and functionalities suitable for mobile game development.

The project also addressed potential risks related to user inputs. Restrictions and checks were added to validate usernames to minimize these risks.

A functional online leaderboard was created and added to the game as a result of this thesis, allowing players to compete against each other. The leaderboard offers an improvement to the game experience, which may increase the user base of The Ultimate Rock & Metal Quiz and, as a result, the ad revenue of the game.

Keywords: mobile game, game development, Unity, Firebase

Sisällys

Lyhenteet

UI: *User interface*. Käyttöliittymä.

UX: *User experience*. Käyttökokemus.

AWS: *Amazon Web Services.* Amazonin pilvipalvelualusta.

SDK: *Software development kit.* Ohjelmistokehityspaketti.

API: *Application programming interface.* Ohjelmointirajapinta.

JSON: *JavaScript Object Notation.* Tiedostomuoto, joka perustuu avain-arvo-pareihin.

Skripti: Ohjelmointikielellä kirjoitettu tiedosto.

Scene: Unityssa käytettävä ympäristöobjekti, joka sisältää kaikki pelinäkymän objektit.

ID: *Identifier.* Tunniste.

UID: *Unique identifier.* Uniikki yksilöintitunniste.

Prefab: Unityssa käytettävä esimääritelty objekti.

1 Johdanto

Olen kehittänyt ja julkaissut Google Play Storessa mobiilipelin, ja tämän insinöörityön tavoitteena on suunnitella ja toteuttaa siihen "online leaderboard" systeemi – eli reaaliaikaisesti päivittyvä verkkopohjainen pistetaulukko, jossa kukin pelaaja näkee oman sijoituksensa suhteessa muiden pelaajien parhaisiin pisteisiin.

Tällä hetkellä pelaajien pisteet tallennetaan paikallisesti laitteen muistiin, mutta tarkoituksena on tehdä verkon kautta toimiva pistetilasto, jossa on kunkin pelaajan paras tulos, nimimerkki ja sijoitus. Pelaajat voivat katsella tätä pistetaulukkoa, omia ja muiden parhaita tuloksia ja tietysti omaa sijoitustaan. Työn lopputuloksena peliin syntyy hyvin suunniteltu ja toteutettu verkkopohjainen pistetaulukko, johon jokaisen pelaajan parhaat pisteet päivittyvät.

Peli on Android-pohjaisilla mobiililaitteilla toimiva “The Ultimate Rock & Metal Quiz”, jonka olen julkaissut itse Googlen sovelluskaupassa. Pelissä on mainoksia, joista kertyy tuloja. Insinöörityön tuloksena syntyvän uuden pistetaulukko-ominaisuuden on tarkoitus lisätä kilpailuhenkeä ja tarjota pelaajille hauskempi ja mielenkiintoisempi pelikokemus. Kehittäjän näkökulmasta tavoitteena on myös motivoida pelaajia pelaamaan enemmän ja samalla katsomaan enemmän mainoksia, mikä puolestaan lisää pelin tuottoa.

2 Mobiilipelien pistetaulukot

Pistetaulukot ovat erittäin yleisiä nykyaikaisissa mobiilipeleissä. Ne tuovat peliin lisäarvoa sekä kehittäjän että pelaajan näkökulmasta. Hyvin suunniteltu ja toteutettu pistetaulukko voi parantaa pelikokemusta merkittävästi ja edesauttaa pelaajien sitoutumista [1].

Pelien pistetaulukot ovat joko paikallisia tai verkkopohjaisia. Paikallinen pistetaulukko on usein hyvin yksinkertainen ja turvallinen ratkaisu, sillä pelaajan

pisteet tallennetaan suoraan laitteelle. Tämän vuoksi paikallisessa pistetaulukossa on rajoituksia – käyttäjä näkee ainoastaan omat tuloksensa eikä siten voi kilpailla muita pelaajia vastaan.

Verkkopohjainen pistetaulukko mahdollistaa pelaajien välisen kilpailun. Pelaajat voivat vertailla suorituksiaan muiden kanssa, mikä luo peliin kilpailullisuutta ja voi myös kannustaa käyttäjiä suosittelemaan peliä ystävilleen. Verkkopohjaisen pistetaulukon lisääminen peliin vaatii enemmän työtä ja teknisiä resursseja, mutta hyvin toteutettuna se voi luoda huomattavasti paremman pelikokemuksen ja sitä kautta kasvattaa pelin suosiota ja tuottoa.

2.1 Insinöörityön tavoitteet ja aiheen rajaus

Tämän insinöörityön tavoitteena on suunnitella ja toteuttaa verkkopohjainen pistetaulukko "The Ultimate Rock & Metal Quiz" -mobiilipeliin, jossa on tällä hetkellä paikallinen pistetaulukko. Paikallisesti tallennettavan pistetaulukon muuttaminen verkkopohjaiseksi on merkittävä parannus, joka hyödyttää sekä pelaajia että pelinkehittäjää.

Tavoitteena on kehittää verkkopohjainen ratkaisu, joka mahdollistaa reaaliaikaisesti päivittyvän pistetilaston luomisen ja ylläpidon. Järjestelmän ansiosta pelaajat voivat nähdä oman sijoituksensa suhteessa kaikkien muiden pelaajien parhaisiin pisteisiin.

Insinöörityö koostuu käytettävän pilvipalvelun valinnasta, käyttöönotosta, integroinnista ja testaamisesta. Työn tuloksena mobiilipeliin tulee konkreettinen parannus.

2.2 Insinöörityön merkitys

Kehittäjän kannalta yksi merkityksellisimmistä hyödyistä on mahdollisuus lisätä pelin näkyvyyttä ja tuottoa. Parempi pelikokemus voi saada pelaajat viettämään enemmän aikaa pelin parissa ja suosittelemaan sitä muille. Verkkopohjainen pistetaulukko motivoi pelaajia palaamaan pelin pariin ja parantamaan

sijoitustaan. Tämä lisää pelin uudelleenpelattavuutta ja tekee siitä hauskemman.

Koska pelissä on integroituja banneri- ja videomainoksia, pelaajien sitoutumisen lisääntyminen voi kasvattaa mainostuloja ja johtaa siten parempiin ansaintamahdollisuuksiin kehittäjälle.

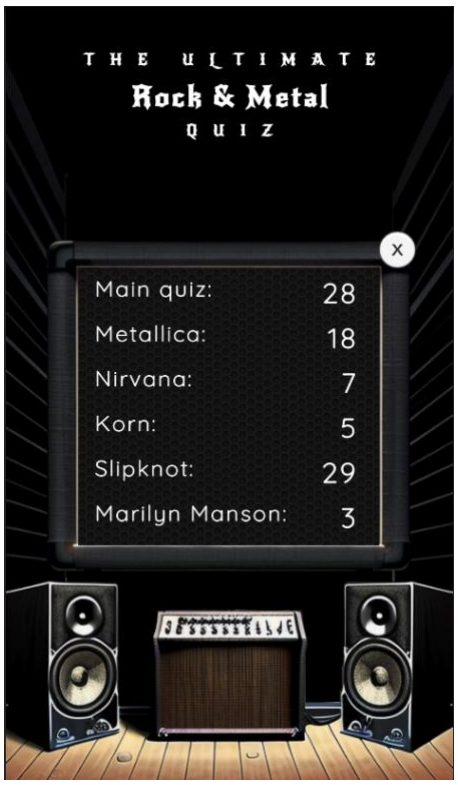
Lisäksi verkkopohjaisen pistetaulukon toteuttaminen vaatii edistyneempiä teknisiä ratkaisuja, kuten tietokantojen ja pilvipalveluiden hallintaa sekä reaaliaikaista datan käsittelyä. Tämä antaa kehittäjälle mahdollisuuden hyödyntää ja kehittää omia teknisiä taitojaan, mikä on hyödyllistä tulevaisuuden projekteissa ja alan työtehtävissä.

2.3 The Ultimate Rock & Metal Quiz -pelin esittely

The Ultimate Rock & Metal Quiz on Android-pohjaisilla mobiililaitteilla toimiva tietovisailupeli, jonka aihealueena on rock- ja metallimusiikki ja -kulttuuri. Ilmaiseksi ladattava free-to-play peli on julkaistu Googlen Play Storessa, ja sen ansaintamallina on Unity Ads -mainonta.

Peli on kehitetty Unity 3D -pelimoottoria ja C#-ohjelmointikieltä käyttäen, ja kehitystyössä on hyödynnetty Git-versionhallintaa. Insinöörityön tekijä on vastannut kaikesta ohjelmointi- ja kehitystyöstä.

Pelin tämänhetkisessä versiossa on paikallinen pistetaulukko, johon tallennetaan ainoastaan pelaajan paras tulos. Taulukkoon tallennetaan pelaajan tulokset pelin kaikki kysymykset sisältävästä ”Main Quiz” -versiosta sekä yksittäisiin yhtyeisiin keskittyvistä versioista. Omia tuloksiaan voi katsella valikon kautta [kuva 1].



Kuva 1. The Ultimate Rock & Metal Quiz -pelin paikallinen pistetaulukko.

Uusi verkkopohjainen pistetaulukko tulee korvaamaan tämänhetkisen toteutuksen. Sen kehityksessä käytetään samoja teknologioita, ja niiden lisäksi projektiin integroidaan tarvittavat pilvipalvelut.

2.4 Pistetaulukot menestyneissä mobiilipeleissä

Pistetaulukot ovat hyvin yleisiä nykyaikaisissa mobiilipeleissä ja muissa videopeleissä. Insinöörityön pistetaulukon toiminnan suunnittelua varten tarkastellaan menestyneiden pelien pistetaulukoiden toiminnallisuuksia ja ominaisuuksia. Hyvä pistetaulukko on paitsi visuaalisesti miellyttävä, myös selkeä ja reilu.

Julkaistujen pelien pistetaulukoista voi huomata joitakin yleisiä ratkaisuja, jotka liittyvät esimerkiksi pistetaulukon sijoitusten määrään, pisteiden laskutapaan tai päivittymistiheyteen.

Esimerkiksi nopeasti päivittyvät pisteet ja sijoituksen paraneminen usein on tavallinen ratkaisu varsinkin nopeatempoisissa mobiilipeleissä, niin sanotuissa casual ja hyper casual -peleissä. Tällainen pistetaulukko kannustaa jatkamaan pelaamista ja antaa myös satunnaisille ja aloitteleville pelaajille mahdollisuuden saada nimensä listalle.

Hyper casual -kategoriaan kuuluvassa Puzzle Blocks Classic -mobiilipelissä on nopeasti päivittyvä pistetalukko, jolla on helppo edistyä [kuva 2].

Kuva 2. Puzzle Blocks Classic -mobiilipelin pistetaulukko

Joissakin peleissä pistetaulukko on jaettu niin sanottuihin liigoihin eli pienempiin osiin, joissa kilpaillaan rajatun pelaajajoukon kesken. Tällainen rakenne mahdollistaa sen, että pelaajat voivat kilpailla samantasoisissa ryhmissä ja siten päästä paremmille sijoituksille helpommin [2].

Mikäli pelissä on useita eri pelityyppejä tai kategorioita, myös pistetaulukko on usein jaettu osioihin. Näin eri kategorioita pelaavat pelaajat kilpailevat toisiaan vastaan, ja sama pelaaja voi päästä myös useammalle listalle. Esimerkiksi Age of Empires 2 Definitive Edition -pelin pistetaulukko on jaettu osiin eri pelityyppien mukaan [kuva 3].

Kuva 3. Age of Empires 2 Definitive Edition -pelin pistetaulukko. Kategoriaa voi vaihtaa yläreunan alasvetovalikosta.

Kaikki edellä esitellyt toimintaperiaatteet ovat tavalla tai toisella The Ultimate Rock & Metal Quiz -peliin sopivia. Peli on suhteellisen yksinkertainen ja nopeatempoinen tietovisa, joten pistetaulukon nopea päivittyminen olisi toimiva ratkaisu. Myös pelaajien jakaminen liigoihin voisi toimia, mikäli pelin pelaajamäärä kasvaa tulevaisuudessa. Lisäksi pelin paikallinen pistetaulukko on jo jaoteltu pelityyppien mukaan, joten samanlainen jako toimisi myös verkkopohjaisessa pistetaulukossa.

2.5 Eri tapoja luoda verkkopohjainen pistetaulukko

Verkkopohjaisen pistetaulukon toteuttamiseen on useita eri tapoja ja teknologioita, joista jokaisella on omat hyvät ja huonot puolensa. Eri tavat ja palvelut sopivat monenlaisiin käyttötarkoituksiin. Suosittuja vaihtoehtoja ovat muun muassa Firebase, AWS ja Google Play Services.

Insinöörityön ensimmäisenä tavoitteena on suunnitella ja valita verkkopohjaiseen pistetaulukon tekoon käytettävä pilvipalvelu. Eri vaihtojen vertailussa on otettu huomioon muun muassa hinta, luotettavuus, käyttöönoton helppous sekä palvelun skaalautuvuus.

2.5.1 AWS (Amazon Web Services)

AWS on Amazonin tarjoama pilvipalvelu, johon sisältyy kattava valikoima työkaluja ja palveluita. Alustan palveluvalikoimassa on paljon mobiilisovellusten kehittämiseen ja hallintaan soveltuvia ratkaisuja, kuten DynamoDB- ja RDS-tietokannat, S3-tallennustila ja Cognito-käyttäjienhallintatyökalu [3]. Verkkopohjaisen pistetaulukon luomiseen sopivat erityisesti DynamoDB ja RDS eli Relational Database Service. Nopea ja skaalautuva DynamoDB sopii hyvin mobiilipelin tietojen hallintaan.

AWS tarjoaa monipuolisia tietokantaratkaisuja, jotka skaalautuvat joustavasti käyttötarpeiden ja käyttäjämäärien mukaan. AWS:n maksuton Free Tier -tilaus sisältää useimmat palvelut, mutta esimerkiksi datan ja tietokantahakujen määrälle on asetettu rajoituksia [3].

Haasteena on se, että AWS-palveluiden käyttöönotto ja hallinta on jonkin verran monimutkaisempaa verrattuna muihin tässä käsiteltyihin tietokantapalveluihin. AWS soveltuu erinomaisesti suurten ja moniulotteisten sovellusten kehitykseen, mutta pelkän pistetaulukon luomiseen se ei ole välttämättä paras valinta.

2.5.2 Firebase

Firebase on Googlen kehittämä ja tarjoama pilvipalvelu, johon kuuluu useita sovellusten ja pelien kehittämiseen soveltuvia työkaluja. Firebase sisältää muun muassa Cloud Firestore ja Realtime Database -tietokannat, analytiikkatyökaluja sekä käyttäjien hallintaan soveltuvan Authentication-palvelun [4]. Verkkopohjaisen pistetaulukon luomiseen voisi käyttää Realtime Firebase tai Cloud Firestore tietokantapalvelua. Näistä Firestore olisi parempi valinta insinöörityön tarkoitukseen, sillä se on uudempi ja monipuolisempi.

Firebasen etuna on sen käyttöönoton helppous. Google on panostanut siihen, että palvelun integrointi mobiilisovelluksiin on sujuvaa. Cloud Firestore on skaalautuva ja joustava ratkaisu, joka mahdollistaa reaaliaikaisen tietojen synkronoinnin. Firebasen dokumentaatio on kattava, mistä on hyötyä mahdollisissa ongelmatilanteissa.

Ilmainen Spark Plan sisältää suurimman osa palveluista, joskin rajoitetulla käyttömäärällä. Tämä riittää varsin hyvin pienen tai keskisuuren sovelluksen tarpeisiin, ja käyttäjämäärän kasvaessa tilausta voi päivittää [5].

2.5.3 Google Play Games Services

Google Play Games Services tarjoaa valmiin pistetaulukkopalvelun, joka on helppo integroida Google Play Storessa julkaistuihin Android-sovelluksiin. Tämä palvelu tarjoaa sovelluskehittäjille yksinkertaisen tavan luoda ja lisätä peleihin pistetaulukoita, saavutuksia ja tapahtumia [6]. Palveluiden lisääminen julkaistuun Android-peliin on nopeaa ja yksinkertaista, eikä niiden käyttöönotto vaadi uutta käyttäjätiliä tai tilausta.

Play Games Services ja julkaisualusta Play Console ovat Googlen tarjoamia palveluita, jotka toimivat saumattomasti yhdessä. Verkkopohjaisen pistetaulukon luominen Play Games Services -palvelun avulla on hyvä vaihtoehto, mikäli tarvitaan nopea ja helppo ratkaisu nimenomaan Android-

laitteille. Valmis pistetaulukko sisältää perustoiminnot, kuten pisteiden hallinnan, lajittelun ja tallennuksen [7].

Vaikka Play Games Services tarjoaa verkkopohjaisen pistetaulukon ja helppokäyttöisen API:n, se ei ole välttämättä paras valinta tämän insinöörityön käyttötarkoitukseen. Tarkoituksena on kehittää pistetaulukko, joka toimii myös muilla alustoilla. Lisäksi on tärkeää, että pistetaulukon hallinta ja mukauttaminen on vapaampaa.

2.6 Valittu teknologia

Vertailun perusteella Firebase Cloud Firestore on hyvä valinta tämän insinöörityön aiheena olevan verkkopohjaisen pistetaulukon toteuttamiseen. Valintaan vaikuttavia seikkoja ovat:

- Firebase on monipuolinen palvelukokonaisuus, joka soveltuu hyvin erityisesti mobiilisovellusten kehittämiseen.
- Firestore tarjoaa reaaliaikaisen synkronoinnin, joten pelaajat voivat nähdä pistetilaston muutokset välittömästi.
- Palvelun käyttöönotto ja integrointi on helppoa Firebase Unity SDK:n avulla.
- Skaalautuva Firestore toimii kaikilla alustoilla, joten pistetaulukkoa voi käyttää myös jatkossa, mikäli sovellus julkaistaan esimerkiksi iOS-laitteille.

Insinöörityössä otetaan Firebase käyttöön, integroidaan se Unityssa kehitettävään mobiilipeliin ja luodaan tarvittava käyttöliittymä. Lisäksi peliin ohjelmoidaan verkkopohjaisen pistetaulukon toteuttamiseen tarvittavat uudet toiminnallisuudet.

2.7 Verkkopohjaisen pistetaulukon mahdolliset haasteet

Verkkopohjaisen pistetaulukon toteuttamiseen liittyy erilaisia haasteita ja uhkia, jotka on syytä ottaa huomioon jo suunnitteluvaiheessa. Insinöörityön aikana voi tulla eteen paitsi teknisiä haasteita ja palvelujen integrointiin liittyviä ongelmia, myös pistetaulukon toiminnallisuuteen ja tietoturvaan liittyviä haasteita.

Mahdolliset haasteet ja niihin varautuminen on esitelty tarkemmin seuraavassa taulukossa [Taulukko 1].

Taulukko 1. Verkkopohjaisen pistetaulukon kehityksessä mahdollisesti ilmenevät haasteet ja niiden ratkaisuehdotukset.

Haaste	**Ratkaisu**
Internet-yhteys: Pelaajalla tulee olla toimiva verkkoyhteys, jotta pistetaulukko päivittyy reaaliaikaisesti.	Rock Quiz -pelin muut palvelut vaativat verkkoyhteyttä, joten peli varoittaa jo nyt yhteysongelmista. Firebase Cloud Firestore tukee offline-käyttöä, joten tarvittaessa pisteet voidaan synkronoida verkkoyhteyden palauduttua.
Tietoturvapuutteet: verkkopohjainen pistetaulukko voi altistua hakkerointiyrityksille, joissa pyritään muuttamaan tai tuhoamaan pistetaulukon tietoja.	Firestore tarjoaa mahdollisuuden määrittää tietoturvasääntöjä. Säännöt tulee määritellä tarpeeksi tiukasti ja niin, että jokainen käyttäjä saa vain välttämättömät oikeudet lisätä ja hakea tietoa. Käyttäjien syöttämä tieto, kuten nimimerkit, tulee myös tarkistaa ennen niiden lisäämistä tietokantaan. Esimerkiksi koodin lisääminen voidaan estää poistamalla erikoismerkit ja rajoittamalla syötteen pituutta.
Epäsopivat nimimerkit: pelaajat voivat yrittää käyttää loukkaavia tai muuten sopimattomia nimimerkkejä, mikä voi paitsi loukata muita pelaajia, myös rikkoa Google Play Storen sääntöjä.	Lisätään peliin regex-pohjainen nimimerkkien suodatusjärjestelmä, joka tunnistaa epäsopivat syötteet.

Verkkopohjaisen pistetaulukon merkittävimmät haasteet liittyvät tietoturvaan, Internet-yhteyden toimintaan ja pelaajien syöttämiin tietoihin. Sovelluksen luotettavan toiminnan ja käyttäjäkokemuksen kannalta on olennaista, että haasteet ratkaistaan onnistuneesti. Taulukossa 1 esitellyt ratkaisuvaihtoehdot otetaan huomioon jo insinöörityön suunnitteluvaiheessa, jotta voidaan varmistaa pistetaulukon turvallisuus, toimivuus ja käytettävyys kaikissa olosuhteissa.

3 Pistetaulukon toteutuksen suunnittelu ja valmistelu

Insinöörityön tavoitteena olevan verkkopohjaisen pistetaulukon tekeminen sisältää toiminnallisuuden suunnittelun, toteutuksen ja testaamisen. Pistetaulukko lisätään The Ultimate Rock & Metal Quiz -peliin.

Insinöörityön suunnitteluvaiheessa otetaan huomioon käyttäjäkokemus (UI/UX) ja varmistetaan, että pistetaulukko on helppokäyttöinen. Myös pisteiden laskutapa suunnitellaan selkeästi ja johdonmukaisesti, jotta lopullisen pistetaulukon toiminta on pelaajien näkökulmasta reilu. Peliin lisättävien grafiikoiden ja visuaalisten elementtien on sovittava pelin tämänhetkiseen teemaan.

3.1 Valitut teknologiat ja palvelut

Insinöörityön toteuttamisessa käytetään useita teknologioita ja palveluita, jotka tukevat projektin ja kehitystyön eri osa-alueita. Verkkopohjainen pistetaulukko luodaan seuraavilla työkaluilla:

- **Unity 3D -pelimoottori**, LTS versio 2021.3.0f1. Pääasiallinen kehitystyö tehdään Unity-pelimoottorilla, jolla Rock Quiz -peli on toteutettu. Unity tarjoaa monipuolisia työkaluja toiminnallisuuden ja käyttöliittymän luomiseen.
- **C#-ohjelmointikieli**. Kaikki ohjelmointi tehdään C#-kielellä, joka toimii saumattomasti yhdessä Unityn kanssa ja on siten ensisijainen valinta.
- **Visual Studio Code**. Kehitystyössä käytetään Microsoftin Visual Studio Code -koodieditoria, joka tarjoaa hyviä ominaisuuksia ja laajennuksia C#-ohjelmointiin ja Unity-kehitykseen.
- **Firebase Cloud Firestore**. Tietokantapalveluna käytetään Firestorea, joka mahdollistaa pistetaulukon reaaliaikaisen hallinnan ja päivittymisen.
- **GitHub**. Versionhallintaan käytetään GitHubia, jonka avulla hallitaan ja tallennetaan koodimuutoksia kehitystyön aikana.
- **Google Play Console**. Kun verkkopohjainen pistetaulukko on valmis julkaistavaksi, sovelluksen päivitetty versio jaellaan käyttäjille Google Play Consolen kautta.

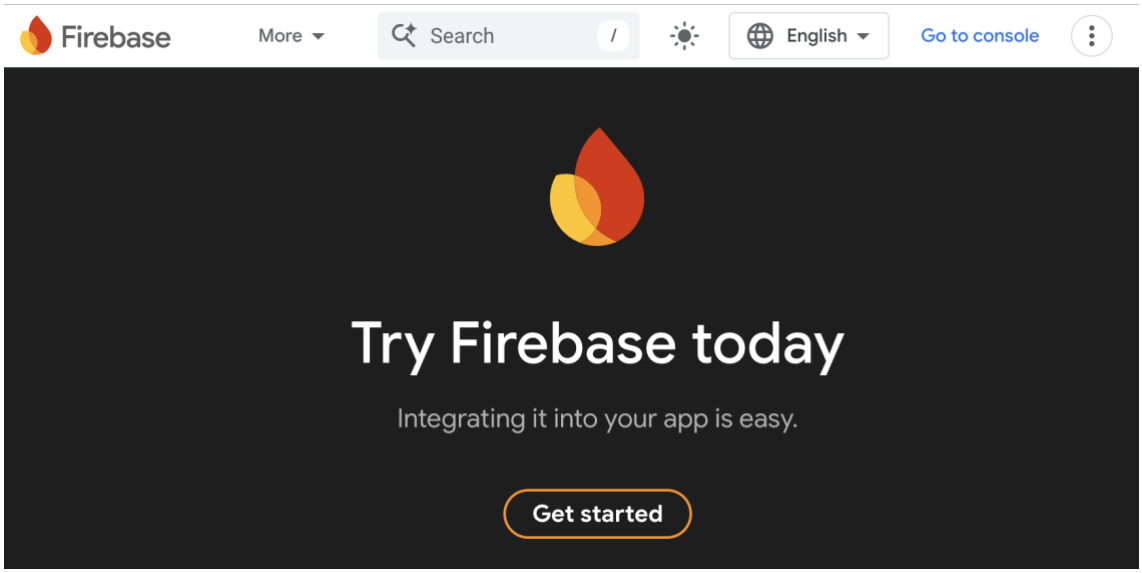
Tämä teknologioiden ja työkalujen kokonaisuus kattaa kaikki insinöörityön vaiheet aina suunnittelusta julkaisuun. Yhdessä ne mahdollistavat tehokkaan kehitystyön ja auttavat luomaan luotettavan pistetaulukon.

4 Pistetaulukon toteuttaminen mobiilipeliin

4.1 Firebasen käyttöönotto ja integrointi

Maksuton Google-tili riittää Firebasen peruspalveluiden käyttöönottoa varten. Verkkopohjaisen pistetaulukon tekemiseen käytettävän Firebase Cloud Firestoren perusversio sisältyy ilmaisiin palveluihin, joten tätä insinöörityötä varten ei tarvita maksullista tilausta. Maksuttoman Google-tilin voi tarvittaessa avata palvelun kirjautumissivun kautta [8].

Google-tilille kirjautunut käyttäjä voi käyttää palvelua ilman erillistä rekisteröitymistä. Firebasen käytön aloittaminen onnistuu helpoiten suoraan palvelun etusivulta [9] painamalla työkalupalkin ”Go to console” -linkkiä tai alempana sivulla olevaa ”Get started” -painiketta [kuva 4].

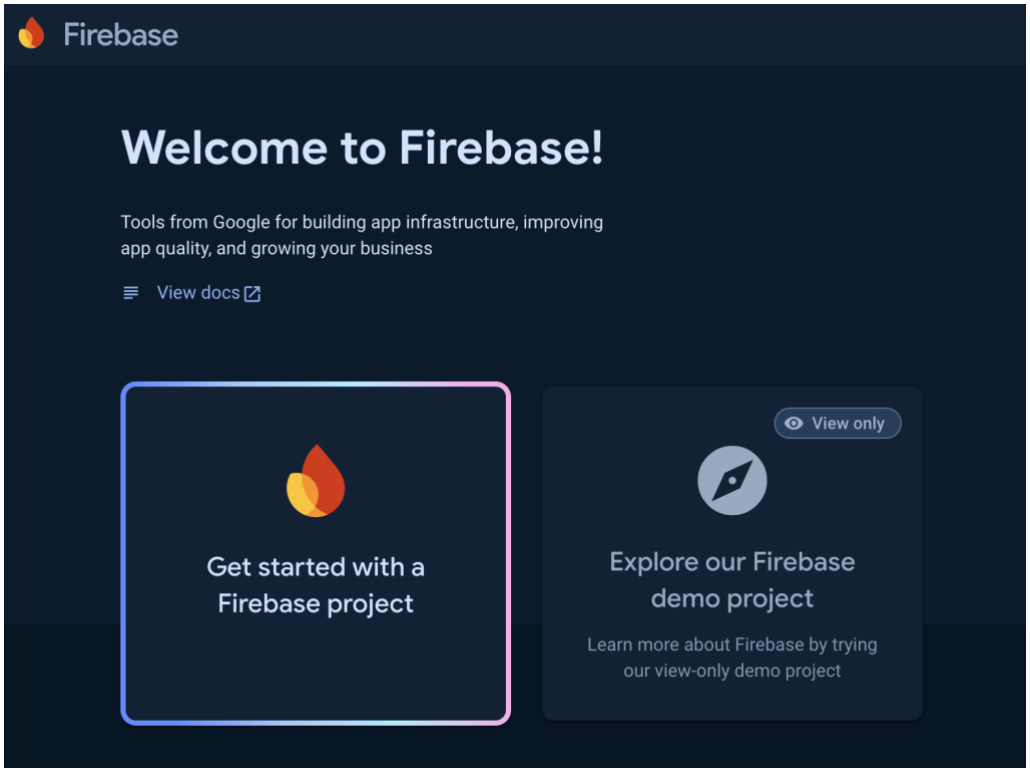


Kuva 4. Firebasen etusivulla on "Get started" -painike ja "Go to console" -linkki, joita painamalla voi aloittaa Firebasen käytön.

Sekä linkki että painike avaavat Firebase consolen, jota käytetään projektien luomiseen ja hallintaan.

Firebase console

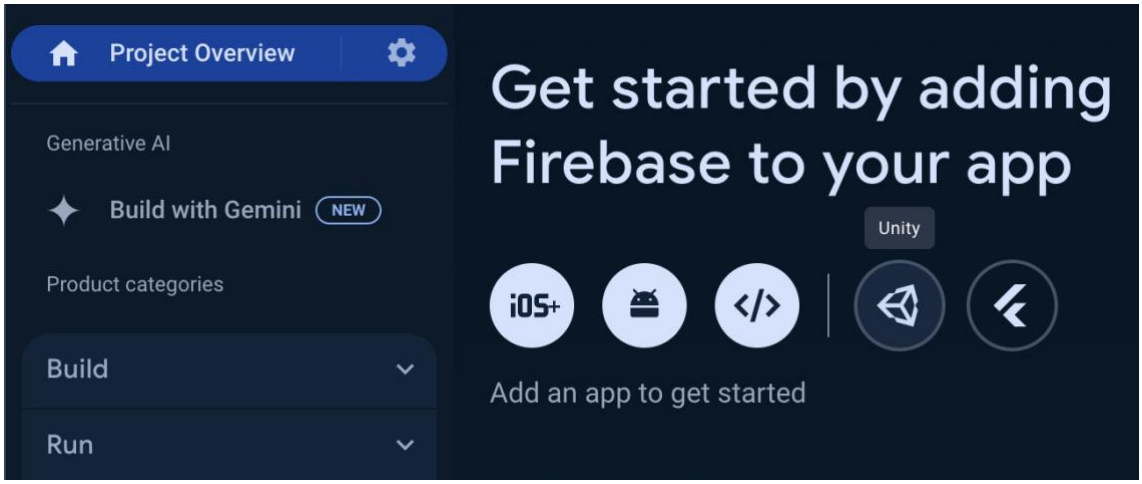
Jotta palveluita voi käyttää, Firebase consolessa täytyy luoda uusi projekti. Mikäli projektia ei ole vielä luotuna, consolen aloitusnäkymässä on painike uuden projektin luomiseen [kuva 5]. Firebasen varsinainen käyttöönotto alkaa siis projektin luomisella ja sen tietojen ja asetusten valinnalla. Uudelle projektille valitaan nimi, minkä lisäksi käyttäjän tulee hyväksyä palvelun käyttöehdot.



Kuva 5. Uusi projekti luodaan Firebase consolen aloitusnäkymän painikkeesta "Get started with a Firebase project".

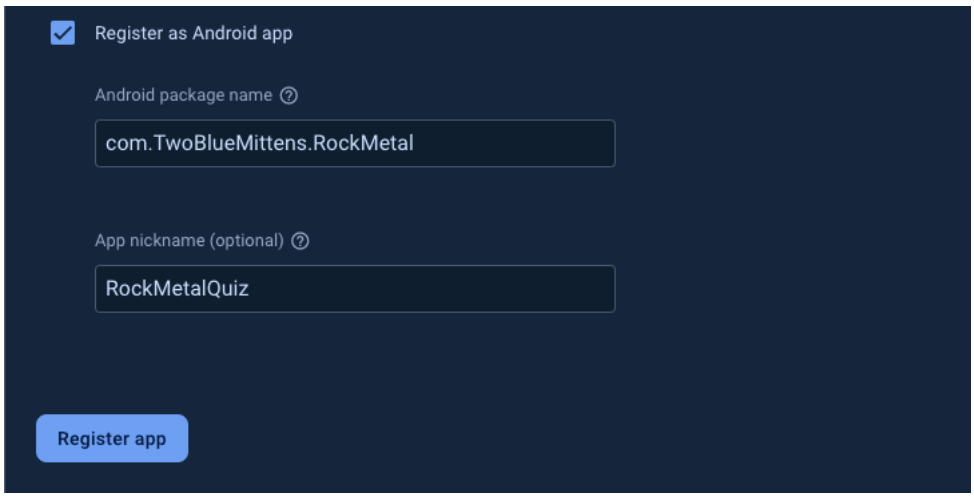
Kun Firebase-projektille on asetettu tarvittavat tiedot, siihen on mahdollista yhdistää sovellus. Firebase on helppo integroida eri teknologioilla tehtyihin sovelluksiin – se tukee suoraan ainakin iOS, Android- ja websovelluksia, sekä Unitylla ja Flutterilla kehitettyjä sovelluksia.

Koska insinöörityön aiheena oleva mobiilipeli on toteutettu Unity-pelimoottorilla, valitaan Unity-sovelluksen yhdistäminen painamalla Unityn logoa [kuva 6].



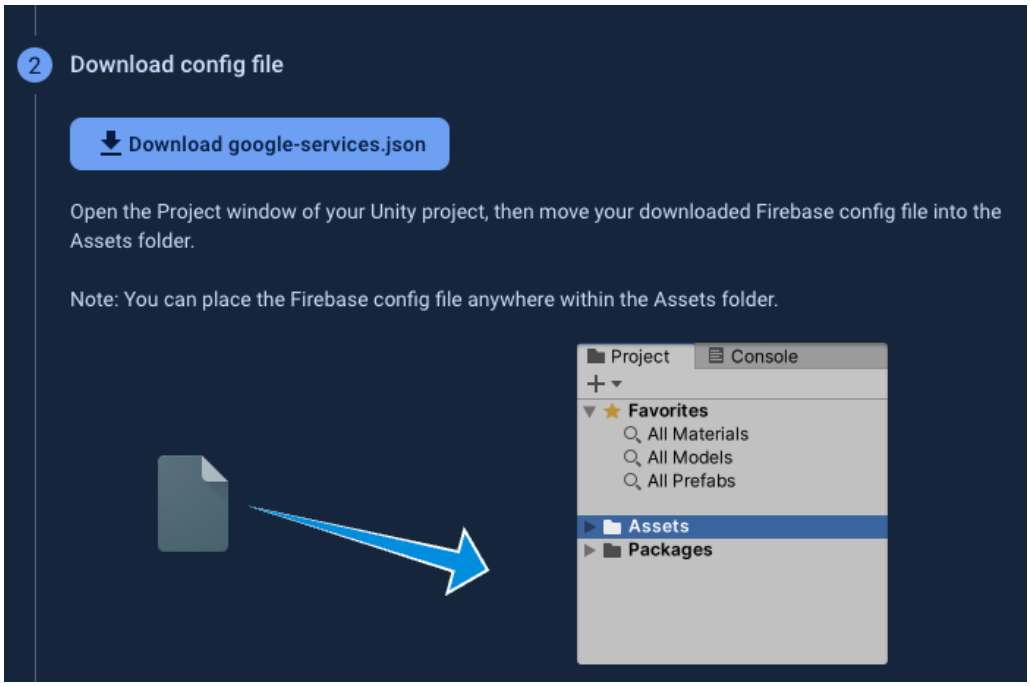
Kuva 6. Firebasen voi yhdistää Unity-pelimoottorilla kehitettävään sovellukseen. Yhdistäminen aloitetaan painamalla Unityn logoa.

Peli on julkaistu Android-laitteille, joten valitaan ”Register as Android app”. Sovellus yhdistyy Firebaseen sen yksilöivän ”Android package name” -tunnuksen kautta [kuva 7]. Yksilöintitunnuksen voi tarkistaa Unityn asetuksista, ja julkaistujen Android-sovellusten kohdalla sen näkee myös Google Play Consolesta ja Play Storesta.



Kuva 7. Firebase yhdistetään sovellukseen lisäämällä sovelluksen yksilöivä Android package name Firebase-projektin tietoihin.

Jotta Firebase ja Unity-projekti yhdistyvät oikein, projektien liittäminen viimeistellään lataamalla Firebasen konfiguraatiotiedosto nimeltä google-services.json [kuva 8] ja lisäämällä se Unityyn. JSON-muodossa oleva tiedosto sisältää muun muassa Firebase-projektin yksilöintitiedot ja API-avaimen.

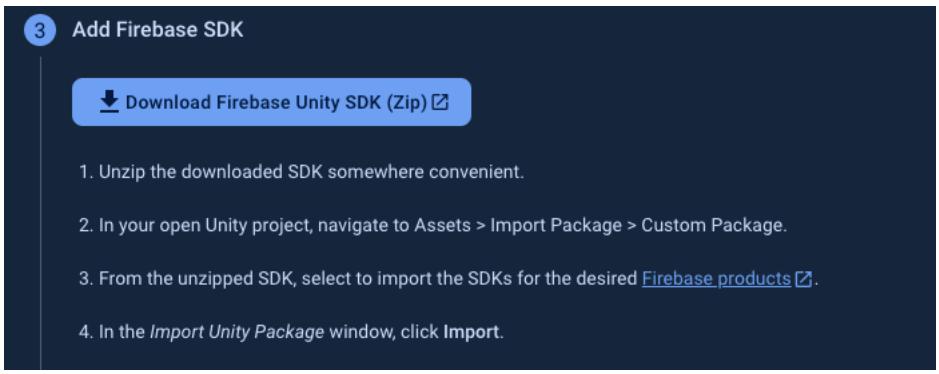


Kuva 8. Konfiguraatiotiedosto ladataan painamalla sinisellä pohjalla olevaa "Download google-services.json" -painiketta.

Ladattu tiedosto lisätään Unity-projektin Assets-kansioon. Unity-sovellus saa konfiguraatiotiedostosta tarvittavat tiedot Firebase-yhteyttä varten.

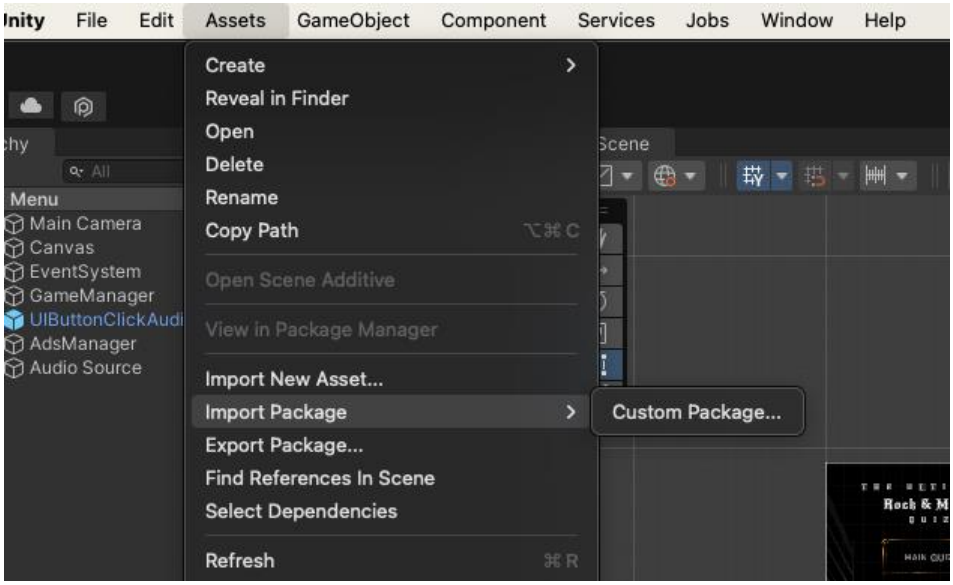
Firebasen lisääminen Unity-projektiin

Firebase asennetaan Unity-projektiin käyttämällä Firebase consolen kautta ladattavaa Firebase Unity SDK -pakettia. Firebase ehdottaa paketin lataamista sovelluksen lisäämisen yhteydessä [kuva 9], mutta tarvittaessa sen voi ladata myös erikseen.



Kuva 9. Firebase Unity SDK:n voi ladata painamalla sinisellä pohjalla olevaa "Download Firebase Unity SDK" -painiketta sovelluksen lisäämisen yhteydessä.

Ladattu SDK sisältää kaikki Unity Firebase -tuotteet ja palvelut, mutta asennusvaiheessa on mahdollista valita ne, joita projektissa tarvitaan. Firebase asennetaan Unity Editorissa Assets-valikon alla olevan Import Package -toiminnon avulla [kuva 10].



Kuva 10. Firebase Unity SDK lisätään projektiin Asset-valikon kautta. Import Package -alavalikon Custom Package -toiminto mahdollistaa valinnaisten palvelujen asentamisen.

Alavalikon Custom Package -valinta avaa tiedostonhallinnan, jonka kautta voi valita projektiin lisättävät palvelut. Asennetaan Firebase Firestore valitsemalla FirebaseFirestore.unitypackage. Paketin lisääminen viimeistellään avautuvan

Import Unity Package -ikkunan kautta [kuva 11]. Valitaan kaikki lisäosan tiedostot ja painetaan Import-painiketta.

Kuva 11. Asennus viimeistellään valitsemalla kaikki tiedostot Import Unity Package -näkymässä.

Import-toiminto lisää ja asentaa paketin sisällön Unity-projektiin. Sen ajamisen jälkeen projektissa on kaikki tarvittava Firebase-yhteyden luomista varten.

Yhteyden luominen Unityn ja Firebasen välille

Ennen varsinaisen kehitystyön aloittamista on tärkeää yhdistää Unity-projekti Firebase consoleen ja varmistaa yhteyden toimivuus. Unity-projektissa tulee olla ajantasainen version Google Play services -palveluista, jotta Firebase Unity SDK:ta voi käyttää Android-laitteilla.

Firebase Fundamentals -dokumentaation suosituksen mukaan projektin yhteensopivuus tulee varmistaa sovelluksen käynnistämisen yhteydessä [10]. Tämä toteutetaan lisäämällä suositeltu tarkistuskoodi [Esimerkkikoodi 1] pelin alkuun. Koodilla varmistetaan, että projekti sisältää kaikki Firebasen toiminnalle

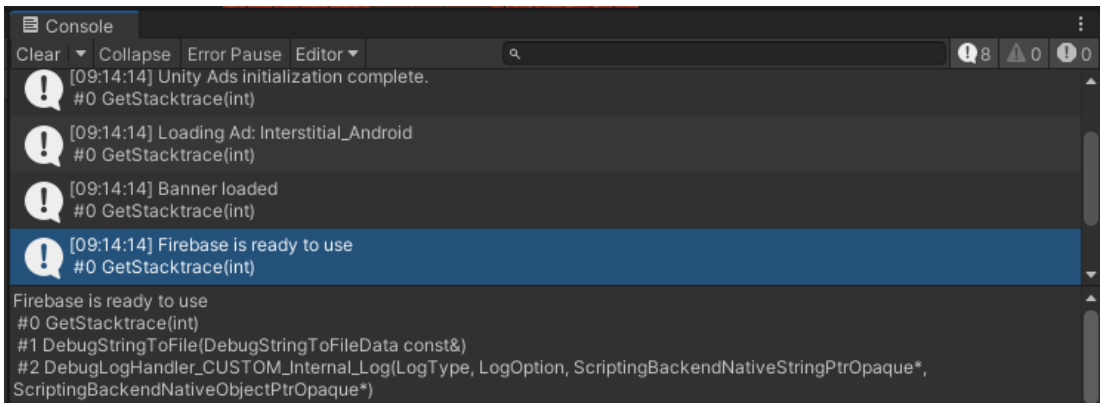
välttämättömät palvelut ja komponentit. Tarkastuksen mennessä läpi Firebase on valmis käyttöön, muussa tapauksessa Unity Editorin consoleen tulostuu virheviesti eikä Firebasea voi käyttää luotettavasti.

```
using Firebase.Extensions;

public class FirebaseManager : MonoBehaviour
{
     Firebase.FirebaseApp app;

     void Start()
     {

     Firebase.FirebaseApp.CheckAndFixDependenciesAsync().ContinueWithOn
MainThread(task =>
     {
          var dependencyStatus = task.Result;
          if (dependencyStatus == Firebase.DependencyStatus.Available)
          {
               // Kaikki kunnossa, Firebase Unity SDK toimii
               app = Firebase.FirebaseApp.DefaultInstance;
               Debug.Log("Firebase is ready to use");
          }
          else
          {
               // Virhe tarkistuksessa. Firebase Unity SDK ei toimi
               UnityEngine.Debug.LogError(System.String.Format("Could
not resolve all Firebase dependencies: {0}", dependencyStatus));
          }
     });
}
```

Esimerkkikoodi 1. FirebaseManager.cs skriptin alkuun lisätty tarkastus.

Tarkastuskoodia varten insinöörityön Unity-projektiin luotiin uusi FirebaseManager.cs skripti. Firebasen tarkastuskoodi on MonoBehaviour-luokasta perittävässä Start-metodissa, jota kutsutaan kerran skriptin lataamisen jälkeen [11]. Lisätään skripti pelin ensimmäiseen sceneen, jotta tarkastus tapahtuu heti pelin alussa.

Toimivuus voidaan nyt varmistaa käynnistämällä peli ja seuraamalla Unity editorin konsolia. FirebaseManager.cs tulostaa konsoliin tekstin ”Firebase is ready to use”, eli Firebase on asentunut ja yhdistetty oikein [kuva 12].



Kuva 12. Unity Editorin konsoliin tulostuu FirebaseManager.cs skriptiin lisätty logiteksti.

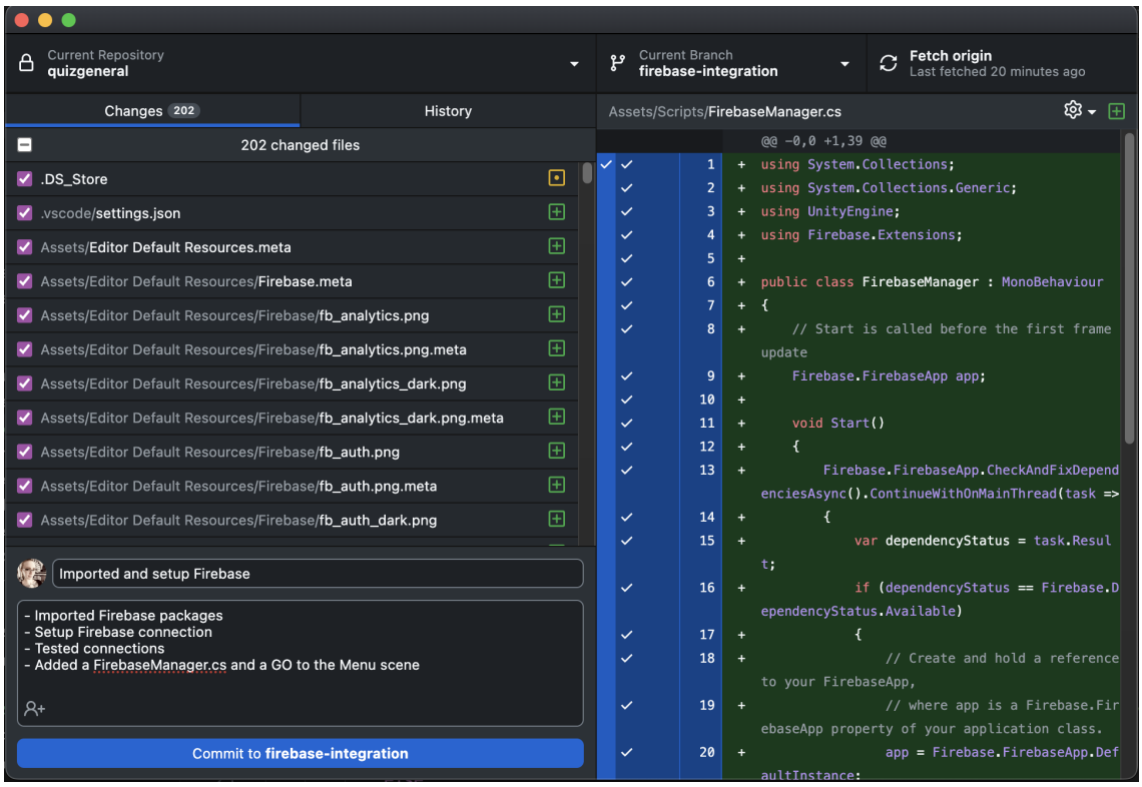
Palvelu on käyttövalmis ja Unity-projekti yhteensopiva, eli Firebasen käyttöönotto on onnistunut.

4.2 Versionhallinta

Firebasen lisääminen peliin on iso päivitys, jonka tallentaminen versionhallintaan on tärkeää. Versionhallinta on olennainen osa ohjelmistokehitystä, sillä se mahdollistaa projektin muutosten seuraamisen, turvallisen koodin hallinnan ja mahdollisuuden aiempien versioiden palauttamiseen esimerkiksi virhetilanteissa.

The Ultimate Rock & Metal Quiz -peliprojektissa versionhallinta on toteutettu käyttämällä GitHubia, joten myös insinöörityön versionhallintatyökaluksi valikoitui GitHub. GitHubin selkeä ja tarkka muutosten seuranta helpottaa projektin etenemisen seurantaa ja dokumentointia.

Koska Firebase on kokonaan uusi toiminallisuus, sille luodaan oma branch eli haara GitHubin repositorioon. Erillisellä haaralla työskentely mahdollistaa toiminnallisuuden kehittämisen ilman, että se vaikuttaa varsinaiseen tuotantokoodiin. Tämä varmistaa projektin eheyden ja toimivuuden ja vähentää riskejä kehitystyön aikana. Kaikki tähän mennessä tallennetut muutokset lisätään versionhallintaan tekemällä commit GitHub Desktopin kautta [kuva 13].

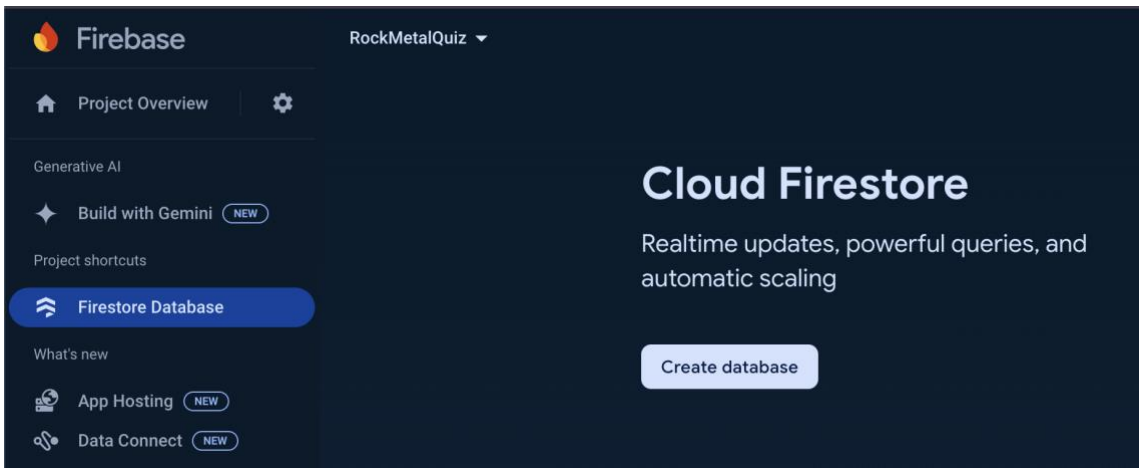


Kuva 13. Projektin muutokset tallennetaan versionhallintajärjestelmään GitHub Desktopin avulla.

Insinöörityön aikana tehdään säännöllisiä committeja versionhallintaan, jotta projektin muutokset tallentuvat turvallisesti ja jatkuvasti. Säännölliset, pienehköt ja selkeät commitit helpottavat insinöörityön edistymisen seurantaa ja dokumentointia. Säännöllinen tallentaminen varmistaa myös sen, että commit-historia pysyy helposti ymmärrettävänä ja mahdollisten virheiden löytäminen ja korjaaminen on helpompaa.

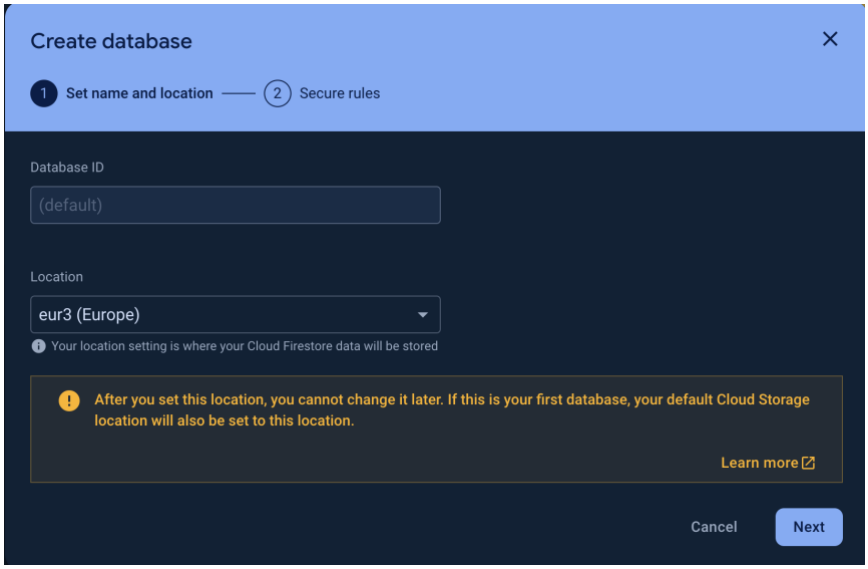
4.3 Tietokantataulu pisteiden tallentamiseen

Pelaajien tulokset on tarkoitus tallentaa tietokantatauluun, joka luodaan Firebase consolen kautta. Aloitetaan luomalla projektille uusi Cloud Firestore -tietokanta Firestore Database -valikon kautta [kuva 14].



Kuva 14. Uusi Cloud Firestore -tietokanta luodaan Firestore Consolen projektisivun kautta.

Uudelle tietokannalle tulee valita sijainti [12]. Valittu sijainti vaikuttaa siihen, mihin lisätty tieto tallennetaan. Insinöörityön tietokannan sijainniksi valikoitui Euroopassa sijaitseva eur3 sen maantieteellisen läheisyyden vuoksi [kuva 15].

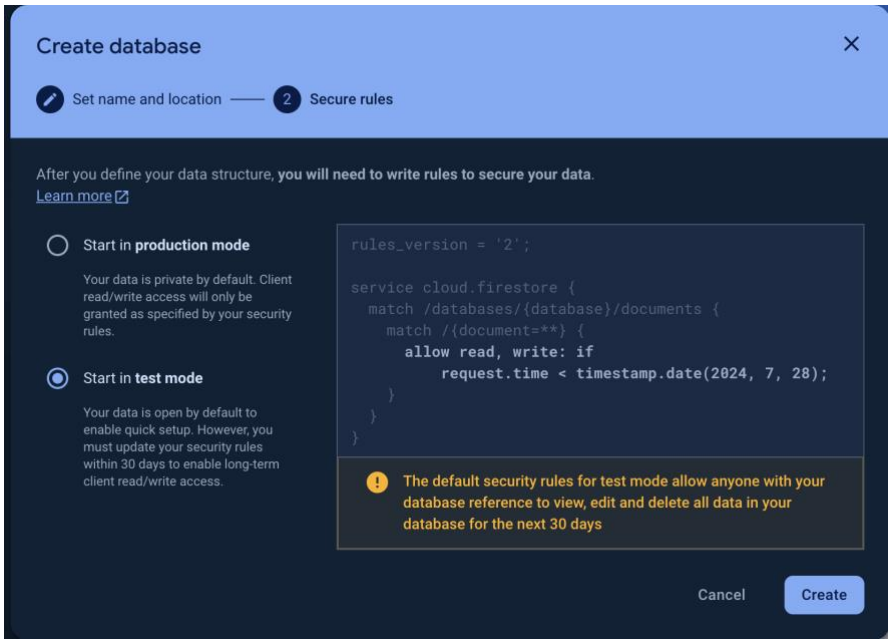


Kuva 15. Tietokannan luomisen yhteydessä sille valitaan tietojen tallentamiseen käytettävä sijainti.

Lopuksi tietokannalle valitaan tietoturvasäännöt. Tietoturvasäännöt tarjoavat kehittäjille keinon hallita ja rajoittaa tietokannan käyttöä ja tietojen näkyvyyttä [13]. Koska käytön aloittaminen onnistuu nopeammin ja helpommin

testiasetuksilla, ”test mode” on hyvä valinta alun kehitysvaiheessa. Test moden tietoturvasäännöt sallivat kaikille käyttäjille täydet luku- ja kirjoitusoikeudet, mikä helpottaa testaamista alussa.

On kuitenkin tärkeää huomata, että tuotantoversion tietoturvasääntöjen täytyy olla tiukemmat, jotta tietokanta pysyy suojattuna. Säännöt tulee siis muistaa päivittää ennen pelin julkaisemista. Hyvänä muistutuksena toimii myös se, että testisäännöt ovat voimassa ainoastaan 30 vuorokautta. [Kuva 16]



Kuva 16. Tietokannan luomisen yhteydessä sille valitaan tietoturvasäännöt. Aloitetaan testisäännöillä, jotka tekevät kehittämisestä helpompaa.

Tietokantataulun rakenteeksi valitaan yksinkertainen ja joustava nested data -rakenne. Firebasen dokumentaation [14] mukaan nested data -dokumenttimuoto sopii hyvin tilanteisiin, joissa tallennettava tieto on rakenteeltaan yksinkertaista ja ennalta määriteltyä. Vaikka nested data -rakenne ei ole yhtä joustava kuin muut vaihtoehdot, tietokantataulua on kuitenkin mahdollista laajentaa tarpeen mukaan.

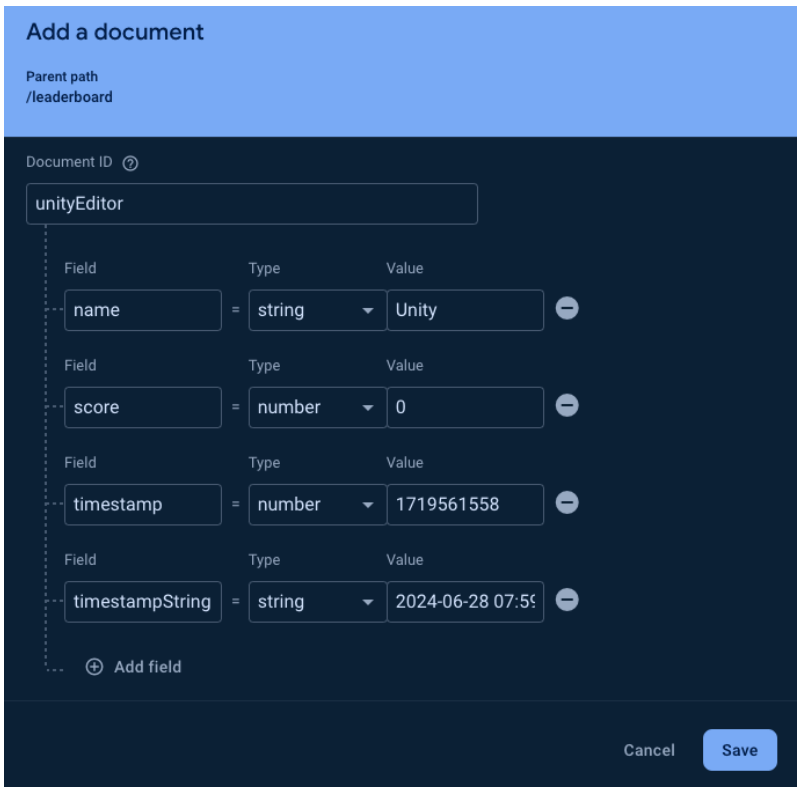
4.3.1 Dokumenttien lisääminen tietokantaan

Insinöörityön pistetaulukkoa varten tarvittava tieto on ennalta määriteltyä, ja jokaiseen lisättävään dokumenttiin tallennetaan samat tiedot. Toiminnallisuuden toteutusta varten tulee tallentaa vähintään pelaajan nimimerkki ja pisteet. Lisäksi tallennetaan tuloksen lisäämisaika tilastointia ja seurantaa varten.

Edellisessä vaiheessa luotuun tietokantaan tulee tehdä vielä uusi kokoelma tietojen tallentamista varten. Avataan pistetaulukkoa varten kokoelma nimeltä leaderboard, luodaan esimerkkidokumentti ja määritellään tallennettava tieto seuraavasti:

- Document ID: Tämä on dokumentin yksilöivä tunniste. Tunniste on mahdollista luoda automaattisesti, mutta sitä voi myös muokata manuaalisesti. Insinöörityössä Document ID määritellään manuaalisesti silloin, kun pelaajan pisteet tallennetaan ensimmäisen kerran.
- Name: Pelaajan nimimerkki tallennettaan merkkijonona (string) tähän kenttään.
- Score: Pelaajan paras tulos tallennetaan kokonaislukuna (number) tähän kenttään.
- Timestamp: Tähän kenttään tallennetaan aika, jolloin pisteet lisättiin tietokantaan. Aika tallennetaan kokonaislukuna (number) Unix-aikana eli sekunteina tai millisekunteina 1.1.1970 alkaen. [15]
- TimestampString: Tähän tallennetaan merkkijonona tallennettava aikaleima. Aika on sama kuin edellisessä timestamp-kentässä, mutta se esitetään helpommin luettavassa päivämäärämuodossa.

Lisätään edellä määritellyt tiedot esimerkkidokumenttiin ja tallennetaan se tietokannan leaderboard-kokoelmaan. Ensimmäinen, tietorakennetta havainnollistava dokumentti luodaan kuvan 17 mukaisella lomakkeella Firebase consolen kautta, mutta jatkossa tiedot lisätään koodin avulla suoraan pelistä.



Kuva 17. Tietokantaan voi lisätä uuden dokumentin Firebase consolen kautta. Kuvakaappaus testikäyttäjän pisteiden lisäämisestä.

Lisätyn koekäyttäjän tiedot tallennetaan kokoelmaan, minkä jälkeen niitä on mahdollista katsella ja muokata joko Firebase consolen kautta tai ohjelmistokoodia käyttäen. Insinöörityössä keskitytään tietojen hakuun ja muokkaamiseen pelin aikana ohjelmistokoodin avulla.

Luodaan Unity-projektiin skripti, jolla voidaan testata dokumentin muokkaamista ja varmistaa yhteyden toimivuus. Pelissä on jo paikallinen pisteiden tallennus, ja GameManager.cs skriptissä on highScore-niminen muuttuja niiden tallentamista varten. Lisätään GameManageriin muut tarvittavat muuttujat ja asetetaan niihin tietokantaan lisätyn koekäyttäjän tiedot testaamista varten Esimerkkikoodi 2:n mukaisesti:

```
public static GameManager manager;
public string userName = "Unity";
public string userUID = "unityEditor";
public int highScore = 20;
```

Esimerkkikoodi 2. GameManager.cs skriptiin tietokannan testaamista varten lisätyt muuttujat.

Muokataan FirebaseManager-skriptiä ja lisätään siihen metodi, jolla pisteet tallennetaan tietokantaan. Pelaajan pistetiedoista muodostetaan Firestoren dokumenttimallin mukainen tietorakenne, joka voidaan lähettää tietokantaan. Päivitetty FirebaseManager.cs sisältää kaksi uutta metodia, AddScore() ja TriggerAddScore(). Metodit ovat seuraavanlaiset:

AddScore

Asynkroninen AddScore-metodi vastaa pelaajan pisteiden lisäämisestä tietokantaan. Esimerkkikoodi 3:n mukainen metodi tarvitsee kolme parametria: playerUID pelaajan yksilöimiseen, playerName pelaajan nimimerkin asettamiseen ja score, joka kertoo tallennettavat pisteet.

Metodissa haetaan docRef-viite tietokannan "leaderboard"-kokoelman dokumenttiin, jonka ID on sama kuin pelaajan UID. Tällä varmistetaan, että pisteet tallennetaan pelaajan omaan dokumenttiin tietokannassa. Mikäli pelaajan UID:tä vastaavaa dokumenttia ei löydy, metodi luo ja lisää uuden dokumentin automaattisesti.

Ennen tietojen lisäämistä luodaan vielä dokumenttiin lisättävä Unix-aikaleima kahdessa eri muodossa. Ensimmäinen, long-tyyppisenä 64-bittisenä kokonaislukuna tallennettava timestamp on helpompi käsitellä ohjelmallisesti, kun taas merkkijonona string-tyyppisenä tallennettava timestampString on helposti ymmärrettävässä, niin sanotusti ihmisystävällisessä muodossa.

Lopuksi Dictionary-tietorakenteella luotu dokumentti lähetetään tietokantaan asynkronisesti docRef-viitteen SetAsync()-metodin avulla. Koska tietokantayhteys on asynkroninen, ohjelman pääsäie ei jää odottelemaan sen valmistumista, vaan jatkaa toimintaansa.

```
public async Task AddScore(string playerUID, string playerName, int
score)
{
     Debug.Log("Lisätään pisteet pistetauluun: " + score + " pelaajan
nimimerkki " + playerName + " ja UID: " + playerUID);
     DocumentReference docRef =
db.Collection("leaderboard").Document(playerUID);

     long timestamp = DateTimeOffset.UtcNow.ToUnixTimeSeconds();
     string timestampString = DateTimeOffset.UtcNow.ToString("yyyy-MM-
dd HH:mm:ss");

     Dictionary<string, object> data = new Dictionary<string, object>
     {
          { "name", playerName },
          { "score", score },
          { "timestamp", timestamp },
          { "timestampString", timestampString }
     };
     await docRef.SetAsync(data);
}
```

Esimerkkikoodi 3. Asynkroninen AddScore()-metodi lisää pelaajan pisteet tietokantaan.

TriggerAddScore

Esimerkkikoodi 4 kuvaa TriggerAddScore-metodia, jota kutsutaan silloin, kun pisteet on tarkoitus lisätä. Asynkronisen operaation valmistuttua tarkistetaan, onnistuiko pisteiden lisääminen. Lopuksi tieto pisteiden lisäämisen onnistumisesta tai epäonnistumisesta tulostetaan Unityn konsoliin.

```
public void TriggerAddScore()
{
     Debug.Log("Triggering add score");
     AddScore(GameManager.manager.userUID, GameManag-
er.manager.userName, GameManag-
er.manager.highScore).ContinueWithOnMainThread(task =>
     {
          if (task.IsCompleted)
          {
               Debug.Log("Pisteiden lisääminen onnistui");
          }
          else
          {
               Debug.LogError("Virhe pisteideiden lisäämisessä");
          }
     });
}
```

Esimerkkikoodi 4. Pisteet lisätään kutsumalla TriggerAddScore()-metodia.

Välittömästi testiskriptien ajamisen jälkeen leaderboard on päivittynyt Cloud Firestoressa, eli yhteys todetaan toimivaksi.

4.4 Pelaajan tietojen ja pisteiden hallinta

Jotta pelaajien pisteiden hallinta ja lisääminen tietokantaan on mahdollista, niitä täytyy voida hallita ja ylläpitää pelin kautta ohjelmallisesti. Pelissä olevan paikallisen pistetaulukon pisteet tallennetaan jo käyttäjän laitteen muistiin. Pisteiden lisäksi tietokantaan tarvitaan pelaajan yksilöivä UID ja pistetaululla näytettävä nimimerkki.

Luodaan pelin ohjelmakoodiin uusi UserData-luokka [Esimerkkikoodi 5] käyttäjän tietojen ylläpitoa ja tallentamista varten. Luokka on merkitty [Serializable] attribuutilla. Tämä kertoo C#-kääntäjälle, että kyseisen luokan olion voi serialisoida eli muuttaa muotoon, jossa se voidaan tallentaa esimerkiksi tiedostoon tai tietokantaan [17].

```
[Serializable]
class UserData
{
    public string userUID;
    public string userName;
}
```

Esimerkkikoodi 5. UserData-luokka tallentaa pelaajan nimimerkin ja yksilöintitunnuksen.

Lisätään pelin GameManager-luokkaan käyttäjän tietojen tallentaminen ja lataaminen. Tiedot tallennetaan laitteen muistiin binäärimuodossa C#-ohjelmointikielen BinaryFormatter-työkalun avulla. Esimerkkikoodi 6 kuvaa tähän käytettävää SaveUserData-metodia.

```
public void SaveUserData()
{
     BinaryFormatter bf = new BinaryFormatter();
     FileStream file = File.Create(Application.persistentDataPath +
"/userData.dat"); // Create a save file
     UserData data = new UserData();

     // Tallennettavat tiedot
     data.userUID = userUID;
     data.userName = userName;

     bf.Serialize(file, data);
     file.Close();
}
```

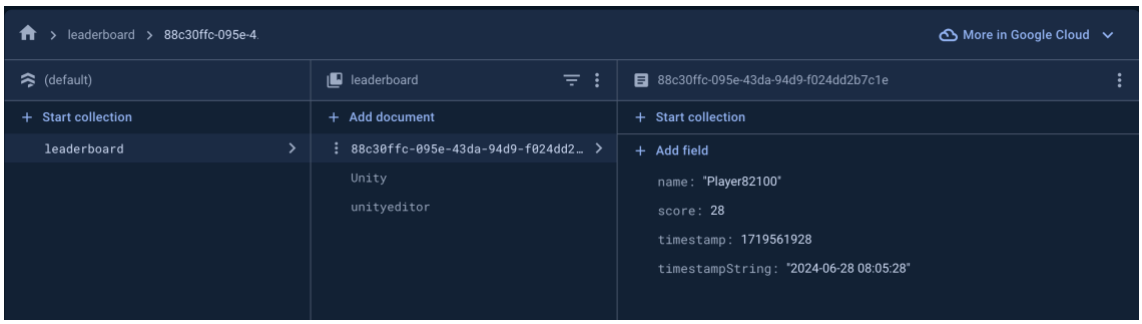
Esimerkkikoodi 6. SaveUserData()-metodi tallentaa pelaajan tiedot laitteelle.

Pelin alussa pelaajan tiedot ladataan laitteen muistista. Mikäli kyseessä on uusi pelaaja, jolla ei ole vielä UID:ta, se luodaan ja tallennetaan samalla. UID:n luomiseen käytetään System.Guid.NewGuid() -komentoa, jolla voidaan varmistua sen ainutlaatuisuudesta [18]. Lisäksi uudelle pelaajalle annetaan alussa satunnainen nimimerkki Esimerkkikoodi 7:n mukaisesti.

```
userUID = System.Guid.NewGuid().ToString();

// Nimimerkki sisältää alun ”Player” ja satunnaisen numeron
System.Random random = new System.Random();
int randomNumber = random.Next(10000, 99999);
string suffix = randomNumber.ToString();
userName = "Player" + suffix;
```

Esimerkkikoodi 7. Uudelle pelaajalle luodaan UID ja nimimerkki.

Pelin ja tietokannan toimintaa testataan skriptien päivitysten jälkeen. Tietokannan leaderboard-kokoelmaan tulee uusi dokumentti ohjelmallisesti luoduilla tiedoilla [kuva 18].



Kuva 18. Tietokannan leaderboard-kokoelmaan on tullut uusi dokumentti, joka lisättiin ohjelmallisesti suoraan pelistä.

Tiedot myös tallentuvat pelaajan laitteelle oikein ja latautuvat seuraavan pelikerran alussa. Yhteys ja pelaajan hallinnan perustoiminnot todetaan toimiviksi.

4.5 Pistetaulukon hakeminen tietokannasta

Lisätään FirebaseManager-luokkaan asynkroninen GetTopScores-metodi, jolla voidaan hakea parhaat pisteet tietokannasta. Haun toteutuksessa käytetään asynkronista metodia, jotta se ei estä sovelluksen muiden osien sujuvaa toimintaa tietokantakyselyn aikana.

Kyselyn alussa luodaan Query eli tietokantakysely, jolla pyydetään tietoja leaderboard-kokoelmasta. Palautettavat tiedot järjestellään score-muuttujan eli pistemäärän mukaan laskevaan järjestykseen. Limit-parametrillä voidaan määritellä, kuinka monta dokumenttia haku palauttaa.

await query.GetSnapshotAsync() suorittaa itse kyselyn, ja sen tulokset luetaan QuerySnapshot-vastauksen Documents-listalta. Lopuksi metodi palauttaa listan parhaista tuloksista.

```
public async Task<List<Dictionary<string, object>>> GetTopScores(int
limit)
{
     Query query =
db.Collection("leaderboard").OrderByDescending("score").Limit(limit);
     QuerySnapshot querySnapshot = await query.GetSnapshotAsync();
     List<Dictionary<string, object>> topScores = new
List<Dictionary<string, object>>();

     foreach (DocumentSnapshot document in querySnapshot.Documents)
     {
          Dictionary<string, object> dataWithId =
document.ToDictionary();
          dataWithId["userUID"] = document.Id;
          topScores.Add(dataWithId);
        }

     return topScores;
}
```

Esimerkkikoodi 8. GetTopScores()-metodi hakee parhaat tulokset tietokannasta.

Esimerkkikoodi 8:n mukaista metodia voidaan käyttää pistetaulukon hakemiseen pelin alussa. Lisäksi sen avulla voidaan suorittaa tietokantahakuja pelin aikana.

4.6 Pistetaulukon käyttöliittymä

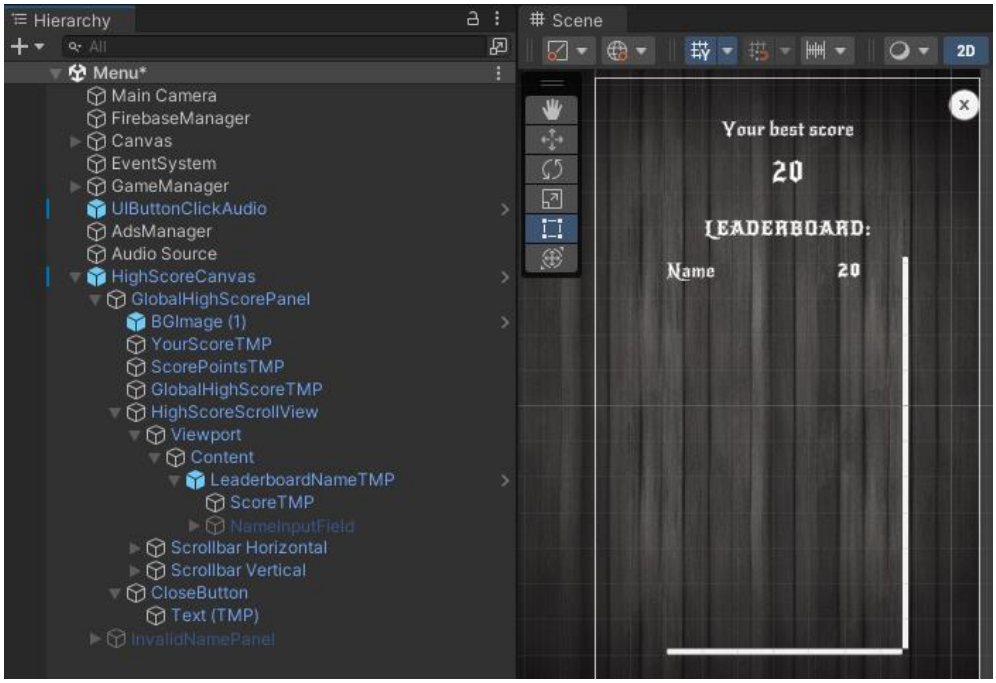
Jotta tietokannasta haetut pisteet voidaan näyttää pelaajille, peliin lisätään pistetaulukon käyttöliittymä ja sen hallintaan tarvittavat skriptit. Käyttöliittymä toteutetaan luomalla lista, jolla näytetään kunkin pelaajan sijoitus, nimimerkki ja pistemäärä. Pistetaulukko näytetään pystysuoraan vieritettävässä ikkunassa, ja näytettävien sijoitusten enimmäismäärä on 50.

Pistetaulukon visuaalisen toteutuksen perustana käytetään seuraavia Unityn UI-elementtejä:

- Canvas: käyttöliittymän pohja, jonka alaobjekteiksi lisätään muita UI-elementtejä. Canvas mahdollistaa käyttöliittymän skaalaamisen käytetyn laitteen näytön kokoon sopivaksi.
- Image: kuvaobjekti, jota käytetään muun muassa käyttöliittymän taustakuvan lisäämiseen.
- TextMeshPro: laajasti muokattava tekstikomponentti, jota käytetään kaikkien käyttöliittymän tekstien näyttämiseen.

- Scroll view: vieritysnäkymä, jota käytetään varsinaisen pistelistan pohjana. Objekti mahdollistaa listan vierittämisen.
- Input field: tekstikenttä, johon käyttäjä voi syöttää tekstiä. Tätä käytetään nimimerkin lisäämisen yhteydessä.
- Button: painike, johon voidaan ohjelmoida haluttuja toiminnallisuuksia. Button-objekteja käytetään käyttöliittymän painikkeiden luomiseen.

Käyttöliittymän rakenne luodaan pelin tyylin mukaiseksi käyttämällä samoja värejä ja fontteja. Kuva 19 havainnollistaa pistetaulukon perusrakennetta ja visuaalista ilmettä.



Kuva 19. Pistetaulukon rakenne on luotu hyödyntämällä Unityn UI-elementtejä.

Koska pisteet lisätään pistetaulukkoon vasta pelin aikana, listalla näytettävien nimimerkkien ja pistemäärien tulee olla ohjelmallisesti muokattavissa. Jotta lisättävien pisteiden määrän ja sisällön muokkaaminen olisi mahdollisimman joustavaa, ne lisätään peliin prefab-objekteina. Niin sanottuja malliobjekteja voi luoda ja lisätä listalle ohjelmallisesti.

4.6.1 Nimen lisääminen pistetaulukkoon

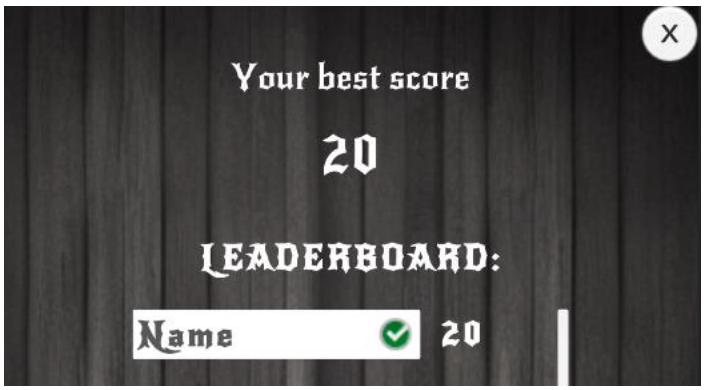
LeaderboardNameTMP-nimiseen prefab-objektiin lisätään tekstikomponentit pelaajan nimen ja pistemäärän näyttämiseksi sekä input field -tekstikenttä nimimerkin lisäämistä varten. Näiden lisäksi luodaan LeaderboardName-skripti, jonka avulla prefabin sisältöä voi muokata ohjelmallisesti.

Luokkaan lisätään viitemuuttujat prefabin muokattaville komponenteille ja julkinen metodi nimen ja pisteiden asettamista varten. Esimerkkikoodissa 9 näkyvän SetName-metodin parametreilla määritellään listalle lisättävä nimimerkki ja pisteet. Jos kyseessä on pelaajan oma pistetulos, asetetaan isMine-totuusarvoksi tosi. Tällöin pisteet esitetään korostusvärillä.

```
public void SetName(string name, string score, bool isMine=false)
{
     this.isMine = isMine;
     if (isMine)
     {
          SetupOwnName();
     }
     nameText.text = name;
     scoreText.text = score;
}
void SetupOwnName()// omat pisteet korostusvärillä
{
     nameText.color = myColor;
     scoreText.color = myColor;
}
```

Esimerkkikoodi 9. SetName()-metodi täyttää pistetaulukon nimet.

Jos pelaaja ei ole vielä asettanut nimimerkkiä, listalle avataan kuvan 20 mukainen tekstinsyöttökenttä. Tietoturvan varmistamiseksi sallittua tekstiä rajataan input field -komponentin asetuksista – nimen enimmäispituus rajataan 14 merkkiin ja tekstiin hyväksytään ainoastaan kirjaimia ja numeroita valitsemalla content type -asetukseksi alphanumeric.



Kuva 20. Pelaaja voi valita nimimerkin pistetaulukolle.

Nimimerkin valitsemisen jälkeen pelaaja voi tallentaa sen painamalla kuvassa 20 näkyvää vihreää painiketta. Painaminen kutsuu LeaderboardName-skriptin AddName-metodia [Esimerkkikoodi 10], joka tallentaa käyttäjän tiedot ja lisää pisteet Firebaseen.

```
public void AddName()
{
     addedName = nameInputField.text; // pelaajan syöttämä nimi
     GameManager.manager.userName = addedName;
     GameManager.manager.SaveUserData(); // tallenna nimimerkki
     FirebaseManager.manager.TriggerAddScore(); // lisää pisteet
     nameText.text += addedName;
     nameInputField.gameObject.SetActive(false);
}
```

Esimerkkikoodi 10. AddName()-metodi tallentaa käyttäjän tiedot laitteelle ja tietokantaan.

4.6.2 Pistetaulukon täyttäminen pelin aikana

Pistetaulukon toiminnallisuuden ja sisällön hallintaa varten peliin luodaan uusi Leaderboard-luokka. Luokan tehtävänä on täyttää pistetaulukko ja päivittää sitä tarpeen mukaan pelin aikana. Pistetaulukosta tehdään koko pelin ajan säilyvä objekti singleton-kaavaa hyödyntäen [19]. Singleton luodaan asettamalla skriptiin staattinen viittaus ja kutsumalla objektin DontDestroyOnLoad-metodia [20]. Esimerkkikoodi 11 havainnollistaa singleton-kaavan käyttöä.

```
public static Leaderboard leaderboard;

void CreateSingleton()
{
     if (leaderboard == null)
     {
          DontDestroyOnLoad(gameObject);
          leaderboard = this;
     }
     else
     {
          Destroy(gameObject);
     }
}
```

Esimerkkikoodi 11. Leaderboard-luokan CreateSingleton-metodi luo staattisen viittauksen.

Pistetaulukon näyttäminen ja nimien lisääminen listalle tapahtuu kutsumalla Esimerkkikoodi 12:n mukaista julkista FillLeaderboard-metodia. Lähtökohtaisesti pistetaulukon tulokset haetaan ainoastaan pelin alussa, jotta vältytään turhalta tiedonsiirrolta. Uusi haku suoritetaan tarvittaessa esimerkiksi silloin, kun pelaaja saa oman nimensä listalle.

Metodin alussa lista tyhjennetään mahdollisista vanhoista pisteistä. Sen jälkeen kutsutaan aiemmin kappaleessa 4.5 luodun FirebaseManager-luokan GetTopScores-metodia, joka suorittaa tietokantahaun ja palauttaa listan parhaista pisteistä.

Haun valmistuttua lista käydään läpi, ja pisteet lisätään käyttöliittymän listalle kohdassa 4.6.1. määriteltyjen prefabien avulla. Listalle luodaan uusia LeaderboardName-objekteja Instantiate-komennolla. Object-luokan Instantiate-metodi mahdollistaa prefabin mukaisten uusien objektien luomisen pelin aikana [21].

```
public void FillLeaderboard(bool needsUpdate = false)
{
     if (isFilled && !needsUpdate) // uusi haku vain tarvittaessa
     {
          return;
     }
     foreach (Transform child in leaderboardContentTransform)
     {
          Destroy(child.gameObject); // tyhjennetään lista
     }
     scrollRect.enabled = false;

     // Haetaan 50 parasta tulosta Firebasesta
     FirebaseManager.manager.GetTopScores(50).ContinueWithOnMainThread(
task =>
     {
          if (task.IsCompleted)
          {
               List<Dictionary<string, object>> topScores =
task.Result;
               for (int i = 0; i < topScores.Count; i++)
               {
                    GameObject leaderboardName =
Instantiate(leaderboardNamePrefab, leaderboardContentTransform); //
uusi leaderboardName
                    string nameString = (i + 1).ToString() + ". " +
topScores[i]["name"];
                    string scoreString =
topScores[i]["score"].ToString();
                    string userUID =
topScores[i]["userUID"].ToString();
                    bool isMine = false;
                    // tarkistetaan, onko pelaajan oma UID
                    if (userUID.Equals(GameManager.manager.userUID))
                    {
                         isMine = true;
                    }
                    // asetetaan nimi ja pisteet

     leaderboardName.GetComponent<LeaderboardName>().SetName(nameString
, scoreString, isMine);
                    }
                    isFilled = true;
                    scrollRect.enabled = true;
               }
               else
               {
                    Debug.LogError("Virhe pisteiden haussa");
               }
     });
}
```

Esimerkkikoodi 12. Leaderboard-luokan FillLeaderboard-metodi hakee parhaat tulokset ja täyttää pistetaulukon pelin aikana.

4.6.3 Pisteiden lisääminen tietokantaan pelin aikana

Pistetaulukolla näytetään korkeintaan 50 parhaan pelaajan tulokset. Jotta tietokanta pysyy kohtuullisen kokoisena, pisteet tallennetaan sinne vain, jos ne riittävät listalle pääsemiseen. Tätä varten FirebaseManager-luokkaan lisätään pistetilastojen ja listalle pääsyyn vaadittavien pisteiden tarkistus.

Pelaajan parhaiden pisteiden mukainen sijoitus saadaan kutsumalla GetPlayerRank-metodia, jonka parametriksi lisätään pistemäärä. Esimerkkikoodi 13:n mukainen metodi hakee Firebasen leaderboard-kokoelmasta 50 parasta tulosta ja vertaa niitä pelaajan pisteisiin sijoituksen määrittämiseksi.

```
public async Task<int> GetPlayerRank(int playerScore)
{
     int rank = 0;
     bool foundRank = false;
     await GetTopScores(50).ContinueWithOnMainThread(task =>
     {
          if (task.IsCompleted)
          {
               List<Dictionary<string, object>> topScores =
task.Result;
               leaderboardEntriesCount = topScores.Count;
               for (int i = 0; i < topScores.Count; i++)
               {
                    if (!foundRank && playerScore >
Convert.ToInt32(topScores[i]["score"]))
                    {
                         foundRank = true;
                         rank = i + 1;
                    }
               }
               if (!foundRank && topScores.Count < leaderboardLength)
               {
                    rank = topScores.Count + 1;
               }
          }
          else
          {
               Debug.LogError("Failed to get top scores");
          }
          });
     Debug.Log("Returning rank: " + rank);
     GameManager.manager.playerRank = rank;
     return rank;
}
```

Esimerkkikoodi 13. GetPlayerRank-metodi selvittää pelaajan sijoituksen.

Mikäli pelaaja suoriutuu hyvin ja saa paremman tuloksen kuin aiemmin, pisteet lisätään tietokantaan AddScoreIfTopRank-metodilla. Metodi tarkistaa pelaajan pisteiden mukaisen sijoituksen ja lisää tuloksen vain, jos se riittää listalle pääsyyn. Esimerkkikoodi 14 havainnollistaa pisteiden lisäämistä.

```
public async Task AddScoreIfTopRank(int playerScore)
{
     int playerRank = await GetPlayerRank(playerScore);
     Debug.Log("Player rank: " + playerRank + " with score: " +
playerScore + " and leaderboard length: " + leaderboardLength);

     if (playerRank <= leaderboardLength)
     {
          TriggerAddScore();
     }
     else
     {
          Debug.Log("Player does not qualify for the top " +
leaderboardLength + ". Rank: " + playerRank);
     }
}
```

Esimerkkikoodi 14. AddScoreIfTopRank-metodi lisää pisteet tietokantaan vain, jos ne riittävät listalle pääsyyn.

4.7 Mahdolliset haasteet ja ongelmatilanteet

Mahdollisten haasteiden ja ongelmatilanteiden huomioiminen on tärkeä osa insinöörityötä. Verkkopohjaisen pistetaulukon kehitystyön haasteet voivat liittyä esimerkiksi teknisiin ongelmiin, kuten ohjelmointivirheisiin tai virhetilanteisiin. Lisäksi käyttäjien toiminta saattaa johtaa ongelmatilanteisiin – esimerkiksi epäasiallisten nimimerkkien näkymiseen listalla.

4.7.1 Sopimattomien nimimerkkien suodattaminen

Yhtenä insinöörityön mahdollisena haasteena on se, että pelaajat saattavat lisätä pistetaulukkoon loukkaavia tai muuten sopimattomia nimimerkkejä. Käyttäjien luoma loukkaava sisältö voi aiheuttaa ongelmia esimerkiksi julkaisualusta Play Storen käyttöehtoihin liittyen, joten epäasiallisten nimimerkkien suodattaminen on tärkeää.

Vaikka tietokantaan lisätään rajallinen määrä pisteitä ja nimimerkkejä, niiden manuaalinen tarkistaminen olisi hyvin työlästä. Nimimerkit pitäisi myös käydä säännöllisesti läpi, jotta sääntöjenvastaiset nimet tulisi poistettua mahdollisimman nopeasti.

Ongelman ratkaisuksi valikoitui käyttäjän lisäämän nimimerkin tarkistaminen GitHub-käyttäjä karlseguinin laatimaan ”badwords” regex-listaan [22] pohjautuvalla suodattimella. Nimimerkkien tarkistamista varten luotiin uusi staattinen OffensiveWordFilter-luokka, joka käyttää listaa nimimerkin tarkistamiseen ennen sen hyväksymistä ja lisäämistä tietokantaan. Nimimerkki tarkistetaan Esimerkkikoodi 15:n mukaisella metodilla.

```
public static bool IsUsernameValid(string username)
{
     bool usernameIsValid = !badWordRegex.IsMatch(username);
     return usernameIsValid;
}
```

Esimerkkikoodi 15. Staattinen IsUsernameValid tarkistaa nimimerkin sopivuuden ja palauttaa tuloksen totuusarvona.

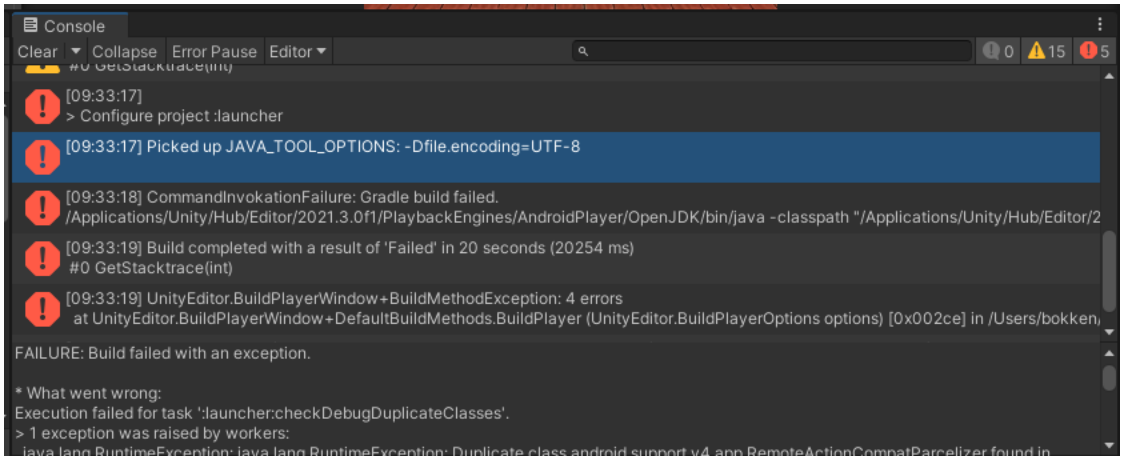
Mikäli nimimerkki todetaan loukkaavaksi, pelaajalle näytetään virheviesti ja kehotus välttää loukkaavia nimiä.

4.7.2 Tekniset ongelmat

Tekniset ongelmat, kuten yhteysongelmat ja ohjelmointivirheet, voivat vaikuttaa pistetaulukon toimivuuteen ja heikentää käyttäjäkokemusta. Ongelmien ennakointi, havaitseminen ja korjaaminen ovat tärkeä osa laadukkaan pistetaulukon kehittämistyötä. Jatkuva testaaminen on hyvä tapa havaita ongelmia sekä kehitystyön aikana että sen jälkeen.

Insinöörityön suurin tekninen haaste liittyi Firebase SDK:n integrointiin. Paketin lisäämisen jälkeen pelin asentaminen Android-laitteelle ei onnistunut, vaikka kaikki vaikutti toimivan oikein Unity editorissa. Unityn konsoliin tulostuneet virheviestit [kuva 21] kertovat päällekkäisistä luokista ja siitä, että Android-sovelluksessa tarvittavan Gradlen luominen ei onnistunut. Android-sovellukseen

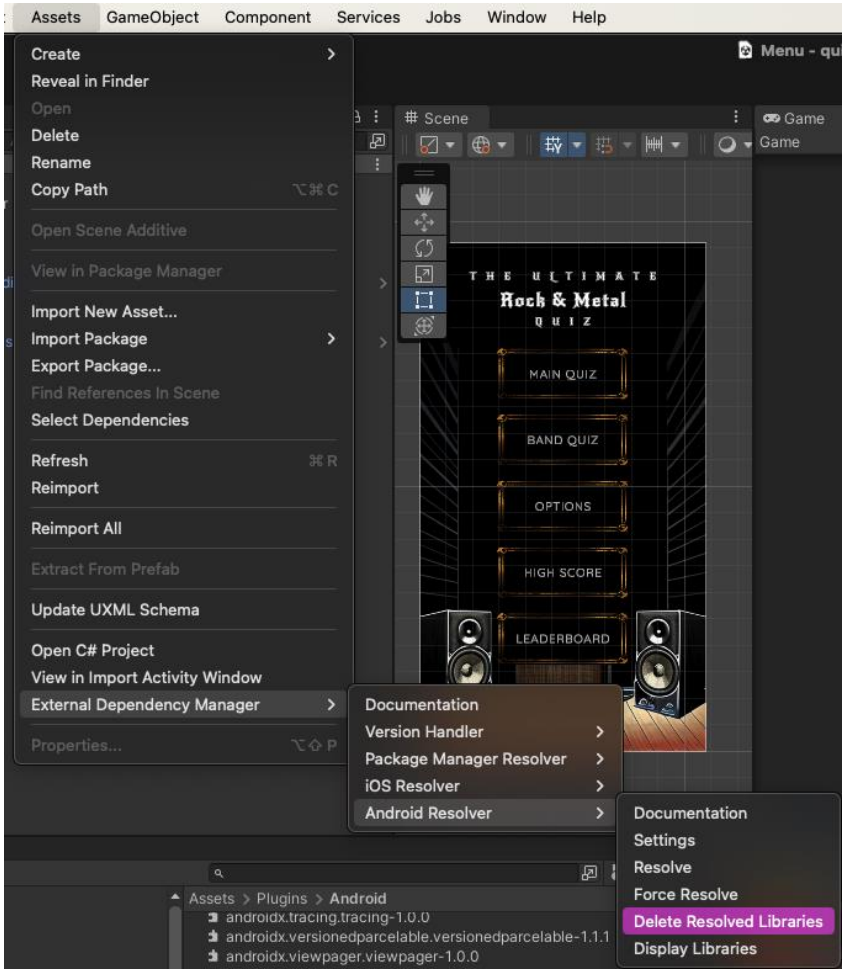
liittyvien ongelmien ratkaiseminen oli ensiarvoisen tärkeää, sillä sovellus on julkaistu nimenomaan Android-laitteille.



Kuva 21. Android-sovelluksen rakentamiseen liittyvät virheviestit Unityn konsolissa.

Android-sovellukseen liittyvä ongelma ratkesi Unityn External Depency Managerin Android Resolver -työkalun avulla. Googlen julkaisema External Dependency Manager [23] auttaa ratkaisemaan Android-projektin riippuvuuksista koituvia ongelmia automaattisesti. Työkalu on erityisen hyödyllinen silloin, kun käytössä on Firebase SDK:n kaltaisia lisäosia ja kirjastoja.

External Dependency Manager on mukana useissa Android-lisäosissa, mutta sen voi tarvittaessa ladata ja asentaa erikseen Googlen GitHubista [18]. Kun paketti on asennettu Unityyn, sitä voi käyttää editorin Assets-valikon kautta.



Kuva 22. External Dependency Manageria voi käyttää Unity editorin Assets-valikon kautta.

Sovelluksen asentamiseen liittyvät ongelmat ja build-prosessin epäonnistuminen johtuivat siitä, että eri SDK:t käyttivät yhteensopimattomia versioita samoista riippuvuuksista. Android Resolverin ajaminen [kuva 22] ratkaisi yhteensopimattomuusongelman ja sovelluksen asentaminen laitteelle onnistui.

5 Lopputulos ja toteutuksen arviointi

Insinöörityön tavoitteena oli kehittää ja integroida toimiva verkkopohjainen pistetaulukko Android-laitteilla toimivaan The Ultimate Rock & Metal Quiz -mobiilipeliin. Työn tavoitteisiin kuului uuden toiminnallisuuden suunnittelu ja toteutus. Kokonaisuus pitää sisällään paitsi tietokantapalvelun käyttöönoton ja

integroinnin, myös pelin käyttöliittymän ja ohjelmakoodin muokkaukset ja tarvittavat lisäykset.

Työn suunnitteluvaiheessa vertailtiin eri teknologioita ja tietokantapalveluita, joista valittiin Firebasen Firestore-tietokantapalvelu. Firestore osoittautui hyväksi valinnaksi, sillä sen integrointi Unitylla kehitettyyn sovellukseen oli helppoa laadukkaan SDK:n, selkeän API:n ja hyvän dokumentaation ansiosta.

Pelin käyttöliittymän muokkaukset toteutettiin Unity-pelimoottorin UI-elementtejä hyödyntäen. Suunnittelussa huomioitiin sekä mobiilipelialan yleiset käytännöt että päivitettävän pelin olemassa oleva käyttöliittymä – esimerkiksi värimaailma ja fontit valittiin niin, että ne sopivat pelin muihin elementteihin ja teemaan. Pistetaulukon käyttöliittymä on yksinkertainen ja selkeä, ja tavoitteessa onnistuttiin hyvin [kuva 23].

Kuva 23. The Ultimate Rock & Metal Quiz -pelin verkkopohjaisen pistetaulukon käyttöliittymä.

Myös pelin ohjelmakoodin muutokset onnistuivat hyvin ja pistetaulukko toimii suunnitellusti. Kehitystyössä otettiin huomioon myös tietoturva-asiat ja käyttäjien syöttämän tiedon aiheuttamat riskit.

5.1 Jatkokehitysmahdollisuudet

Insinöörityön tavoitteena oli kehittää verkkopohjainen pistetaulukko mobiilipeliin. Työn laajuus on kuitenkin rajallinen, ja pistetaulukon kehitystyö painottui perustoiminnallisuuksiin ja siihen, että lopputuloksesta tulee toimiva osa olemassa olevaa peliä.

Työn lopputuloksena syntynyttä pistetaulukkoa voi kuitenkin jatkokehittää useilla eri tavoilla. Pelin menestyksestä ja käyttäjämääristä riippuen pistetaulukon jatkokehitys voisi koostua esimerkiksi seuraavista parannuksista ja uusista toiminnoista:

- Käyttöliittymän parantaminen. Nykyinen käyttöliittymä on yksinkertainen ja selkeä, mutta sen ulkoasua ja käytettävyyttä voisi suunnitella paremmaksi. Käyttöliittymään voisi lisätä esimerkiksi animaatioita ja muita visuaalisesti houkuttelevia elementtejä.
- Maailmanlaajuinen ja yksityinen pistetaulukko. Jos pelin suosio ja pelaajamäärät kasvavat, yksi mahdollinen jatkokehitysidea voisi olla pistetaulukon jakaminen eri kategorioihin. Peliin voisi lisätä kaikkien pelaajien pisteet tilastoivan, maailmanlaajuisen pistetaulukon oheen esimerkiksi maakohtaisia pistetaulukoita tai ystäväryhmien omia pistetaulukoita.
- Pelillistämisen lisääminen. Yhtenä jatkokehitysmahdollisuutena on pelillistämiselementtien lisääminen pistetaulukkoon. Tämä voisi tarkoittaa vaikkapa erilaisia palkintoja tai kunniamerkkejä, joita pelaajat voivat kerätä suorittamalla haasteita.
- Sosiaalinen jakaminen. Peliin voisi lisätä mahdollisuuden jakaa omia tuloksia ja saavutuksia sosiaalisen median kanaville. Tämä voisi lisätä myös pelin näkyvyyttä ja sitä kautta kasvattaa pelaajamääriä.

Nämä pistetaulukon jatkokehitysideat voisivat parantaa pelin käyttäjäkokemusta entisestään. Pistetaulukon parantaminen voi myös lisätä pelaajien sitoutumista ja pelin houkuttelevuutta ja siten kasvattaa pelaajamääriä.

5.2 Insinöörityön merkityksellisyys

Verkkopohjaisen pistetaulukon lisääminen mobiilipeliin tuo hyötyjä sekä pelaajille että kehittäjälle. Pistetaulukko edistää kilpailuhenkisyyttä ja motivoi pelaajia jatkamaan pelaamista. Tämän myötä pelissä vietettävä aika voi kasvaa, mikä osaltaan auttaa lisäämään pelistä saatavia mainostuloja.

Insinöörityö kattaa useita ohjelmistokehityksen osa-alueita, kuten pelisuunnittelua, ohjelmointia, käyttöliittymän kehitystä ja tietokantojen hallintaa. Projektin aikana kertynyt käytännön kokemus ja lisäosaaminen ovat kehittäneet ammattitaitoani ohjelmistokehittäjänä. Lisäksi insinöörityön suunnittelu- ja kehitystyö on auttanut minua pysymään ajan tasalla mobiilipelialan trendeistä.

Verkkopohjaisen pistetaulukon toteuttaminen Firebase-alustan avulla tarjoaa lisäetuja myös työelämässä. Firebase on paljon käytetty pilvialusta varsinkin mobiili- ja webpalveluita kehittävissä yrityksissä. Sen käyttäminen insinöörityössä on lisännyt osaamistani tietokantojen hallinnasta ja integroinnista myös tulevia työelämän projekteja silmällä pitäen.

6 Yhteenveto

Tämän insinöörityön päätavoitteena oli verkkopohjaisen pistetaulukon toteuttaminen mobiilipeliin. Tämä tavoite toteutui suunnitellusti. Työssä käytetty Firebase Cloud Firestore osoittautui hyväksi valinnaksi, sillä sen integrointi peliin onnistui hyvin. Lisäksi palvelu mahdollistaa pistetaulukon reaaliaikaisen synkronoinnin ja luotettavan tiedonhallinnan.

Kehitetty pistetaulukko lisää pelin kilpailuhenkisyyttä ja parantaa pelikokemusta. Tämä voi lisätä pelissä vietettyä aikaa ja kasvattaa pelin mainostuloja. Jatkokehitysmahdollisuuksia ovat esimerkiksi käyttöliittymän parantaminen ja pelillisten elementtien lisääminen pistetaulukkoon.

Projektin aikana kertyi monipuolista osaamista pelinkehityksen ja tietokantojen hallinnan parissa sekä käyttäjäkokemuksen ja tietoturvan kehittämisessä.

Lähteet

1 Kärpuk, Kalev. 2024. How to Create a Leaderboard for Gamification. Verkkoaineisto. Adact OÜ. <https://adact.me/blog/creating-a-leaderboard/> Luettu 6.8.2024

2 Stankovic, Stanislav. 2021. Building better leaderboards. Verkkoaineisto. Medium. <https://uxdesign.cc/building-better-leaderboards-a5013d19cbd7> Luettu 6.8.2024

3 AWS Free Tier. 2024. Verkkoaineisto. Amazon Web Services, Inc. <https://aws.amazon.com/free/> Luettu 7.8.2024

4 App Development Solutions. 2024. Verkkoaineisto. Google. <https://firebase.google.com/solutions> Luettu 7.8.2024.

5 Pricing plans. 2024. Verkkoaineisto. Google. <https://firebase.google.com/pricing> Luettu 7.8.2024.

6 Google Play Games Services. 2024. Verkkoaineisto. Google. <https://developers.google.com/games/> Luettu 7.8.2024.

7 Leaderboards. Google Play Games Services. 2024. Verkkoaineisto. Google. <https://developers.google.com/games/services/common/concepts/leaderboards> Luettu 7.8.2024.

8 Sign in. 2024. Verkkoaineisto. Google. <https://accounts.google.com/signin> Luettu 7.8.2024.

9 Firebase. 2024. Verkkoaineisto. Google. <https://firebase.google.com/> Luettu 7.8.2024.

10 Firebase Fundamentals. 2024. Verkkoaineisto. Google. <https://firebase.google.com/docs/unity/setup?hl=en&authuser=0#confirm-google-play-version> Luettu 11.8.2024

11 Unity Documentation. 2024. Verkkoaineisto. Unity Technologies. < https://docs.unity3d.com/ScriptReference/MonoBehaviour.Start.html> Luettu 11.8.2024

12 Firebase Documentation. 2024. Verkkoaineisto Google. <https://firebase.google.com/docs/projects/locations> Luettu 11.8.2024

13 Firebase Documentation. 2024. Get started with Cloud Firestore Security Rules. Google. <https://firebase.google.com/docs/firestore/security/get-started> Luettu 11.8.2024

14 Firebase Documentation. 2024. Choose a data structure. Google. <https://firebase.google.com/docs/firestore/manage-data/structure-data#nested_data_in_documents> Luettu 11.8.2024

15 Linux.fi-wiki. 2021. Unix-aika. Linux.fi. <https://www.linux.fi/wiki/Unix-aika> Luettu 15.8.2024

16 Microsoft Learn. 2022. Integral numeric types (C# reference). Microsoft. <https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/builtin-types/integral-numeric-types> Luettu 15.8.2024

17 Unity Documentation. 2024. Scripting API Serializable. Unity Technologies. <https://docs.unity3d.com/ScriptReference/Serializable.html> Luettu 15.8.2024

18 Microsoft Learn. 2022. Guid.NewGuid Method. Microsoft. <https://learn.microsoft.com/en-us/dotnet/api/system.guid.newguid?view=net-8.0> Luettu 15.8.2024

19 French, John. 2024. Singletons in Unity (done right). Verkkoaineisto. <https://gamedevbeginner.com/singletons-in-unity-the-right-way/> Luettu 21.8.2024

20 Unity Documentation. 2024. Scripting API Object.DontDestroyOnLoad. Verkkoaineisto. Unity Technologies. <https://docs.unity3d.com/ScriptReference/Object.DontDestroyOnLoad.html> Luettu 21.8.2024

21 Unity Documentation. 2024. Instantiating Prefabs at run time. Verkkoaineisto. Unity Technologies. <https://docs.unity3d.com/Manual/InstantiatingPrefabs.html> Luettu 21.8.2024

22 Seguin, Karl. 2023. badwords. Verkkoaineisto. <https://github.com/mogade/badwords/blob/master/en.txt> Luettu 21.8.2024

23 Google Samples. 2024. External Dependency Manager for Unity. Verkkoaineisto. Google Inc. <https://github.com/googlesamples/unity-jar-resolver> Luettu 6.9.2024