Heikki Raussi

Firebase-tietokannan turvallisuus

Opinnäytetyö

Tieto- ja viestitekniikka

2021

Kaakkois-Suomen ammattikorkeakoulu

Tutkintonimike	Insinööri (AMK)
Tekijä/Tekijät	Heikki Raussi
Työn nimi	Firebase-tietokannan turvallisuus
Toimeksiantaja	Xamk Game Studios
Vuosi	Toukokuu 2021
Sivut	55 sivua, liitteitä 1 sivua
Työn ohjaaja(t)	Pekka Vilpponen

TIIVISTELMÄ

Opinnäytetyön tavoitteena on tutkia Google Firebase-tietokannan turvallisuutta sitä käyttävää puhelinsovellusta varten. Työssä tutustutaan tietokantojen ja niiden turvallisuuden yleiseen teoriaan sekä Unity-pelimoottorin ja Firebase-kehitysalustan yhteiseen toimintaan.

Työn toimeksiantaja on Xamk Game Studios, joka on Kaakkois-Suomen ammattikorkeakoulun kanssa toimiva järjestö. Xamk Game Studios keskittyy hankkeisiin, jotka kehittävät paikallista pelinkehitystoimintaa ja tarjoavat opiskelijoille harjoitustöitä. Toimeksiantona on tuottaa toimeksiantajalle tietoa tietokantojen turvallisuudesta ja auttaa kehittämään toimeksiantajan puhelinsovelluksen turvallisuutta.

Työssä tutkitaan tietokantojen turvallisuuteen liittyviä konsepteja, teknologioita ja menetelmiä. Työssä vertaillaan erilaisia tietokantatyyppejä ja esitellään niiden pääpirteitä sekä avaineroja. Tietokantojen ohella avataan tarpeen tullen monia niihin epäsuorasti tai suorasti liittyviä tietoteknisiä konsepteja. Tietokantojen turvallisuuden lisäksi kiinnitetään huomiota myös tiedon turvallisuuteen, kuten sen eheyteen, validiteettiin ja saavutettavuuteen.

Kehitystyössä käydään läpi Firebase-kehitysalustan ja Unity-kehitysalustan kehitysalustojen konfiguroinnit sekä selitetään niiden merkitys vaiheittain turvallisuuden näkökulmasta. Siinä myös analysoidaan puhelinsovelluksen tietokannassa käytettävät tiedot ja esitetään turvallisuuteen pohjautuva tietokantarakenne. Lopputuloksena on kokonaisuus, joka helpottaa suunnittelemaan turvallisempia tietokantarakenteita ja valitsemaan tilanteeseen sopivia ratkaisuja Firebase-kehitysalustalla sekä sen ulkopuolella.

Työ tarjoaa paljon hyödyllistä tietoa, mutta se keskittyy vähemmän toimeksiantajan varsinaiseen projektiin kuin oli tarkoitus. Kaikkia toivottuja osa-alueita ei ole katettu, koska työssä esiintyy erittäin laaja kirjo tietoteknisiä aiheita. Vaikka työ ei täysin keskity toimeksiantajan projektiin, sitä voidaan hyödyntää sen jatkokehityksessä.

Asiasanat: tietokannat, turvallisuus, kehitysalusta, sovellus, tietotekniikka

South-Eastern Finland University of Applied Sciences

Degree	Bachelor of Engineering
Author (authors)	Heikki Raussi
Thesis title	Database security in Firebase
Commissioned by	Xamk Game Studios
Time	May 2021
Pages	55 pages, 1 pages of appendices
Supervisor	Pekka Vilpponen

ABSTRACT

The purpose of this thesis was to study the security of Google's Firebase database for a phone application. The work offers an introduction to databases and their general theory on security, as well as using Unity game engine paired with Firebase.

The work was commissioned by Xamk Game Studios, which is an organization that works with South-Eastern Finland University of Applied Sciences (Xamk). Xamk Game Studios focuses on projects that enhance the local game development scene and offer Xamk students some practice work. Commission itself was to produce the commissioner information about database security and help improve the security of their phone application.

The work studied various concepts, technologies and procedures surrounding databases. The work compared databases of a different kind and presented their main features and key differences. Along with databases, many IT concepts directly or indirectly connected with them were reviewed as well. In addition to databases, attention was also paid to data security. This included things such as: Data integrity, validity, and accessibility.

Research work covered the configurations of Firebase and Unity development platforms and explained their significance from a security perspective. It also analyzed the data used in the phone application and presented a database structure based on security. The final product was something that can be used to plan safer database structures and choose more fitting solutions in Firebase and outside of it.

The work was educational and offered much useful information, but it focused less on the commissioned work than intended. Not every field was covered to a desirable degree, because of the large scope of the IT related subjects. Even though the work did not manage to focus entirely on the commissioner's project, it can be utilized in its further development.

Keywords: databases, security, development platform, application, IT

SISÄLLYS

KÄSITTEET

Horisontaalinen skaalautuvuus	Koneellisen kapasiteetin lisääminen uusilla laitteilla
Instanssi	Luokan yksittäinen esiintymä ohjelmoinnissa
MVP	Urheiluottelun tärkein pelaaja
Ohjelmointikieli	Tietokoneiden ymmärtämä koneellinen kieli
Pelimoottori	Ohjelmisto, jolla videopelejä luodaan ja suoritetaan
Renderöinti	2D tai 3D mallin kuvittaminen ruudulle
SSL	Salausprotokolla, jolla suojellaan tietoliikennettä
Validi	Noudattaa oikeanlaista muotoa
Verkkotunnus	Nimi, jolla viitataan verkossa olevaan osoitteeseen
Vertikaalinen skaalautuvuus	Koneellisen kapasiteetin lisääminen päivittämällä laitteita

1 JOHDANTO

Xamk Game Studios on Kaakkois-Suomen Ammattikorkeakoulun Xamkin kanssa toimiva järjestö. Se tarjoaa opiskelijoille harjoitteluprojekteja, työpaikkasuosituksia, tapahtumia ja kehittää alueen paikallista peliteollissuutta.

Game Game on Xamk Game Studiosin ja Xamkin oppilaiden ylläpitämä harjoitteluprojekti, jonka tarkoituksena on tarjota Android- ja iOS-laitteilla toimiva puhelinsovellus urheiluseurojen ja niiden otteluiden tulosten seurantaa varten. Sovelluksesta näkee seurojen uutisia ja siinä on erilaisia minipelejä sekä arvonta, josta urheiluseurojen fanit voivat voittaa seuran tarjoamia palkintoja.

Opinnäytetyössä selitetään mikä tietokanta on, ja siinä keskitytään eniten Game Game -projektissa käytössä olevaan Googlen Firebase-tietokannan turvallisuuteen.

2 KEHITTÄMISTYÖ

Opinnäytteen kehittämistyöhön kuuluu tutkinnallista sekä soveltavaa työtä. Kehittämistyöllä tuotetaan hyötyä ja tietoa toimeksiantoa koskevaa projektia varten, sekä perustetaan pohja tulevaa jatkokehitystä varten.

Teoriaosuuksissa tutkitaan tietokantoja ja niiden turvallisuutta, keskittyen eniten toimeksiantoa koskevalle projektille olennaisiin asioihin. Teoreettista tietoa hyödynnetään kehittämisvaiheessa, jossa perehdytään opinnäytetyön projektin tietokannan konfigurointiin ja tietokannan tekniseen käyttöön sovelluksen puolella.

Kehittävässä osuudessa tutustutaan tietokannan projektin ympäristöön ja kehitystyökaluihin. Kehitystyössä käydään läpi tietokannan rakenteellisia ominaisuuksia ja turvallisuuskonfigurointeja, sekä projektin kehittämisessä käytettäviä menetelmiä.

2.1 Opinnäytetyön tavoitteet ja tarkoitus

Opinnäytetyön tavoitteena on tutkia tietokantojen turvallisuutta ja siihen liittyviä konsepteja, teknologioita ja menetelmiä. Pääsijaisena tutkimuskohteena

on toimeksiantoa koskevan Game Game -projektin käytössä oleva Googlen Firebase-tietokanta. Tarkoituksena on soveltaa tutkimustyössä kerättyä tietoa ja parantaa Game Game -projektin tietokannan turvallisuutta tulevaisuuden jatkokehitystä varten.

Opinnäyte soveltuu myös uusien projektien käyttöönoton avuksi, koska siinä ei keskitytä ainoastaan Game Game -projektiin. Siinä käsitellään asiat yleispiirteisyyttä painottaen ja käsitellään aiheet Game Game -projektin ympärillä. Eniten siitä on hyötyä vastaaville projekteille, jotka käyttävät perustavasti Unity-kehitysalustaa ja Firebase-kehitysalustaa.

Opinnäytetyön on myös tarkoitus olla opettavainen ja tuoda kattava näkemys tietokannoista ja niiden turvallisuudesta lukijoille, joille tietokannat eivät ole entuudestaan tuttuja. Lukijan on kuitenkin hyvä omaa jonkinlainen käsite nykyaikaisesta IT-teknologiasta, koska kaikkia aihepiirejä ei pystytä täysin kattavasti avaamaan.

2.2 Opinnäytetyön rakenne

Opinnäytetyössä on 7 lukua. Johdannossa esitellään opinnäytetyön olennaiset sidosryhmät sekä siinä käsiteltävänä olevat aiheet ja kehittämisen kohteena oleva projekti. Toisessa pääluvussa kerrotaan opinnäytetyön tavoitteet ja tarkoitus, sekä sen rakenteesta ja siinä käytetyistä menetelmistä. Kolmannessa pääluvussa selitetään tietokannoista yleisesti. Siinä kerrotaan mitä SQL- ja NoSQL-tietokannat ovat ja miten ne eroavat toisistaan, sekä todetaan projektissa olevan käytössä NoSQL-tietokanta.

Neljännessä pääluvussa kerrotaan ensin Googlen Firebase-kehitysalustasta ja sitten käsitellään projektille relevantteja ominaisuuksia yksityiskohtaisemmin. Viidennessä pääluvussa esitellään Unity, joka on toimeksiantoa koskevan Game Game -projektin ensisijainen kehittämistyökalu sekä väylä tietokantaan. Siinä myös kerrotaan Unity-kehitysalustan ja Firebase-kehitysalustan yhteistoiminnasta, sekä näytetään miten ne ovat asennettu toimimaan yhdessä.

Kuudennessa pääluvussa keskitytään toimeksiantoa koskevan Game Game -projektin tietokannan, tietovaraston ja sovelluspuolen turvallisuuteen ja esitellään kaikki siihen liittyvät komponentit ja tärkeät konfiguraatiot. Siinä käytetään paljon esimerkkejä ja konsepteja, koska varsinaisen tietokannan turvallisuusyksityisyyskohtia ei haluta paljastaa. Seitsemännessä pääluvussa tarkastellaan opinnäytetyön tutkinnallisia ja soveltavia tuloksia, ja tehdään päätelmiä mahdollista jatkokehitystä varten.

2.3 Tutkimuskysymys ja tutkimusmenetelmät

Opinnäytetyön tutkimuskysymyksenä on vastata siihen, mitä Firebase-tietokannan turvallisuus tarkoittaa. Tähän kuuluu tietokantojen turvallisuuteen liittyvät yleiset konseptit, teknologiat ja menetelmät. Tutkimuskysymykseen vastataan opinnäytetyön toimeksiannon näkökulmasta, ja löydökset tulevat esille kehittämistyössä tehdyssä työssä.

Pääsijaisena tutkimusmenetelmänä käytetään lähteiden hakua ja niihin viittaamista. Lähteiden tiedon johdonmukaisuutta ja toimivuutta testataan loogisilla konsepteilla ja johtopäätöksillä, joita käytetään toimeksiantoa koskevan Game Game -projektin tietokannan kehittämiseksi.

Opinnäytetyössä omaksutaan ajattelutapa, jossa ei nojata liikaa vanhaan tietoon vaan tutkitaan asioita perustavasti tekemättä mitään olettamuksia. Opinnäytetyössä avataan käsitellyt aiheet kerroksittain, jotta voidaan hyvin ymmärtää käsiteltävinä olevat aiheet.

3 TIETOKANNAT

Tietokanta on järjestelmällinen kokoelma tietoa, jota säilötään tyypillisesti elektronisesti jollain tietokonejärjestelmällä. Tietokannan toimintaa ohjataan useimmiten tietokannan hallintajärjestelmän (DBMS) kautta, johon löytyy useita kaupallisia sekä avoimen lähdekoodin ratkaisuja. Tietokannan täytyy tarvittaessa säilöä suuria määriä tietoa, ja mahdollistaa turvattu ja valvottu pääsy usealle tietoon valtuutetulle käyttäjälle. (Oracle 2020.)

Tietokannassa säilytetty tieto on teknisesti dataa, mutta suomen kielessä “tieto” voi asiayhteydestä riippuen tarkoittaa kumpaa tahansa asiaa. Data

koostuu raaoista faktoista, joita ei ole vielä prosessoitu paljastaakseen niiden merkitystä. Data voi siis esiintyä erilaisissa muodoissa, ja varsinaista tietoa syntyy sen prosessoinnista. (Coronel ym. 2020, 6–7.)

Uutta tietokantaa laatiessa kannattaa olla pätevä ja kattava suunnitelma, jossa määritellään tarkasti tietokannan käyttötarkoitukset ja vaatimukset. Monet hankalasti jäljitettävät ja korjattavat ongelmat tietokannoissa johtuvat niiden huonosta suunnittelusta. Huonosti suunnitellun tietokannan käyttö voi muun muassa johtaa huonojen päätösten tekoon puutteellisen tai virheellisen tiedon pohjalta. (Coronel ym. 2020, 13.)

3.1 Tietokantajärjestelmä

Tietokantajärjestelmällä tarkoitetaan kaikkia komponentteja, jotka ohjaavat tiedon keräämistä, varastointia ja hallintaa tietokannan ympäristössä. Tietokantajärjestelmä koostuu tyypillisesti viidestä suuresta osasta: Laitteistosta, ohjelmistosta, ihmisistä, menettelytavoista ja tiedosta (Coronel ym. 2020, 22.)

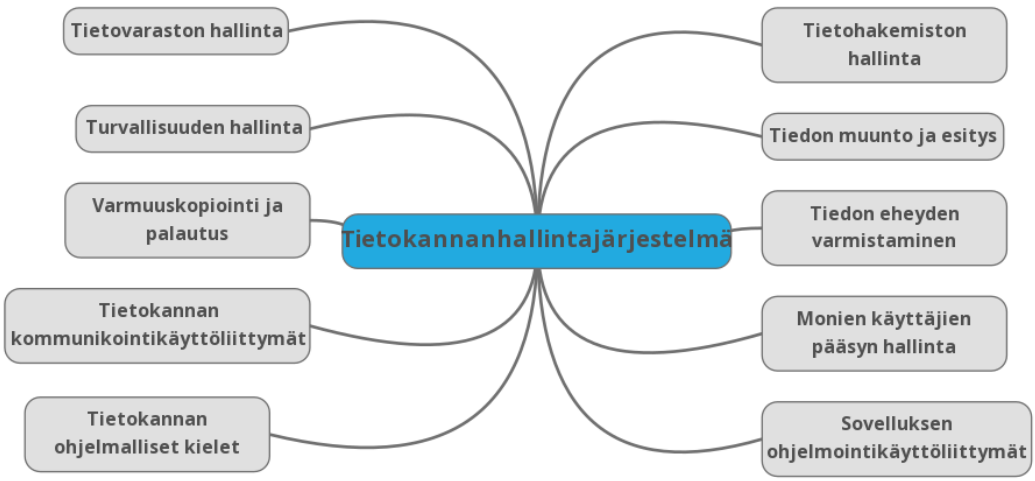
Tietokantajärjestelmä tuo organisaation hallintaan uuden ulottuvuuden. Sen monimutkaisuus vaihtelee riippuen organisaation koosta, toiminnasta ja yrityskulttuurista. Tietokantajärjestelmän tulee olla kustannuksellisesti, strategiesti ja taktisesti tehokas vastaamaan organisaation tarpeita. Käytössä oleva tietokantateknologia usein vaikuttaa valintoihin tietokantajärjestelmän suhteen (Coronel ym. 2020, 24–25.)

Tietokantajärjestelmää ei tule aina tietokantojen ohella huomioitua tarkemmin, mutta periaatteessa kaikilla tietokannan käyttäjille on sellainen jossakin muodossa. Tietokantoja käyttääkseen ei tarvitse aina luoda mitään monimutkaista järjestelmää, mutta on käytännöllistä luoda edes jonkinlainen pohja tietokantojen käytön hallinnalle. Tätä pohjaa voidaan myöhemmin uuden ymmärryksen mukana laajentaa ja parantaa.

3.2 Tietokannanhallintajärjestelmä

Tietokannanhallintajärjestelmä on ohjelmistopaketti, jonka tarkoitus on määritellä, manipuloida, hakea ja hallita tietoa tietokannassa. Tietokannanhallintajärjestelmä tyypillisesti manipuloi tietoa itsessään, tiedon tyyppiä, kenttien nimiä ja tiedostorakennetta. Se myös määrittelee säännöt tiedon validointiin ja manipulointiin. (Technopedia s.a.)

Tietokannanhallintajärjestelmä (DBMS) on tietokantajärjestelmän olennainen komponentti, ja se suorittaa monia tärkeitä toimintoja varmistaakseen tiedon eheyden sekä johdonmukaisuuden tietokannassa. Useimmat näistä toiminnoista eivät näy mitenkään loppukäyttäjälle ja niille ei löydy vastinetta tietokannanhallintajärjestelmän ulkopuolelta. (Coronel ym. 2020, 25–30.)



Kuva 1. Tietokannanhallintajärjestelmän toiminnot (Coronel ym. 2020, 25–30)

Tietokannanhallintajärjestelmän tärkeimpiä toimintoja esitellään kuvassa 1 (ks. Coronel ym. 2020, 25–30). Tietokannanhallintajärjestelmään ei välttämättä tarvitse kiinnittää paljoa huomiota merkitykseltään pienten tietokantojen kanssa, mutta kuvan 1 kaltaisesta tietokannanhallintajärjestelmästä kannattaa kuitenkin omaksua aina joitain periaatteita.

3.3 SQL ja NoSQL

Relationaaliset tietokannat, joihin pääsee käsiksi SQL-kielellä, kehitettiin 1970-luvulla vähentääkseen samojen toistuvien tietojen määrää talletuskokojen ollessa kalliimpia kuin kehittäjien aika. SQL-tietokannoilla on tyypillisesti kankea, monimutkainen ja taulukkomainen rakenne ja ne vaativat kalliimpia vertikaalisia skaalauksia. NoSQL-tietokannat kehitettiin myöhäisellä 2000 luvulla keskittyen skaalautumiseen, nopeisiin kyselyihin, nopeisiin sovellusten vaihtoihin ja ohjelmoinnin helpottamiseen kehittäjille. (Schaefer s.a.)

Mikään ei kuitenkaan pakota valitsemaan yhden tai toisen väliltä. Jollain taholla voi olla useampia SQL- ja NoSQL-tietokantoja käytössä. Ne palvelevat eri tarkoituksia, joten niiden tuomia hyötyjä ja haittoja kannattaa pohtia suunnitellessa tietokantaratkaisuja. Game Game -projektissa on käytössä NoSQL-tietokanta, koska se sopii paremmin projektin tarkoituksiin ja tukee paremmin projektissa käytettyjä teknologioita ja tietotyyppejä.

Mikäli Game Game projektiin liittyisi rahatapahtumia, kirjanpitoa tai muuta vastaavaa, myös vankkarakenteisen SQL-tietokannan tuominen olisi suositeltavaa. SQL-tietokannan käyttö on rajoittuneempaa, mutta se taas voi toimia etuna esimerkiksi tiedon validoinnissa.

3.3.1 SQL-tietokannat

SQL-tietokannat käyttävät hyvin laajalti käytössä olevaa SQL-ohjelmointikieltä (Stuctured Query Language) ja ne ovat relationaalisia. Relationaalinen tietokanta järjestelee tiedon tauluihin, joita linkitetään toisiinsa käyttäen tauluista löytyvää yhtenäistä (relationaalista) tietoa. (IBM Cloud Education 2019.)

Relaatiomalli mahdollistaa SQL-tietokannoille johdonmukaisen ja yhtenevän tietorakenteen, mutta muutosten teko siihen sen käyttöönoton jälkeen on haasteellista. Relaatiotietokannat vaativat enemmän suunnittelua etukäteen, sekä tarkemman tietämyksen tietokannan tulevasta sisällöstä. (Anderson & Nicholson 2020.)

Relaatiotietokanta käyttää taulurakennetta, jossa kaksiulotteisissa tauluissa säilytetään tietoa rivissä ja sarakkeissa. Tauluissa on käyttäjän näkökulmasta

toisiinsa liittyviä kokonaisuuksia. Relaatiomalli vastaa konseptiltaan arkistointia loogisessa muodossa, tämä on tehnyt sen ymmärtämisestä helpompaa verrattuna edeltäviin tietokantamalleihin. (Coronel ym. 2020, 72.)

Relaatiotietokannan taulussa on tyypillisesti pääavain ja viiteavain, joiden avulla määritellään miten relaatiotietokannan taulut liittyvät toisiinsa. Pääavain on relaatiotietokannan taulussa sarake tai joukko sarakkeita, jota käytetään yksilöivästi tunnistamaan tietty rivi taulussa. Viiteavain on vastaavasti sarake tai joukko sarakkeita, joilla on sama tieto kuin toisesta taulusta löytyvällä pääavaimella. (IBM s.a.)

3.3.2 NoSQL-tietokannat

NoSQL-tietokannat eivät noudata relaatiomallia, tai käytä mitään tiettyä ohjelmointikieltä. NoSQL-tietokantamallien tarkoituksena on täten tarjota ratkaisuja ongelmiin, joihin SQL-tietokannat eivät voi helposti vastata. Näistä suurimpia ovat tietokantojen rakenteellinen joustavuus ja horisontaalinen skaalautuvuus. (Anderson & Nicholson 2020.)

NoSQL-tietokannat soveltuvat hyvin tapauksiin, joissa tarvitaan joustavaa tiedon mallinnusta tai mahdollisuuksia rakenteellisiin muutoksiin. Ennen kuin ottaa käyttöön NoSQL-tietokantaa, kannattaa kuitenkin tarkastella käsiteltävän tiedon soveltuvuutta relaatiotietokantaan. (Anderson & Nicholson 2020.)

Organisaatioilla on nykypäivänä tarve kerätä suuria määriä tietoa, joka voi esiintyä rakenteellisessa tai ei-rakenteellisessa muodossa. Tämän myötä organisaatiot kohtaavat uusia haasteita suorituskyvyn ja skaalautuvuuden takaamiseksi. Tiedon volyymin kasvun, suorituskyvyn, skaalautuvuuden ja kustannusten hallitsemista kutsutaan massadataksi. Massadatan yleisenä tarkoituksena on ollut löytää tehokkaita tapoja kerätä Web-pohjaista dataa ja muodostaa siitä käyttökelpoista tietoa. NoSQL-tietomalli tarjoaa ratkaisuja massadatan hallintaan liittyviin ongelmiin joihin relaatiomalli ei hyvin sovellu vastaamaan. (Coronel ym. 2020, 51.)

Konseptuaalisesti yksinkertaisin ja tämän opinnäytetyön kannalta olennaisin NoSQL-tietomalli on avain-arvo-paritietomalli. Avain-arvo-parissa avain toimii

parin yksilöivänä tunnisteena ja arvo voi olla tietotyypiltään mitä tahansa, kuten kuva, teksti tai tekstidokumentti. Avain-arvo-parit järjestellään tyypillisesti ämpäreihin, joka on avain-arvo-pari-tietokannan vastine relaatiomallin tauluille. (Coronel ym. 2020, 841–842.)

3.4 Tietokannan turvallisuuden perusteita

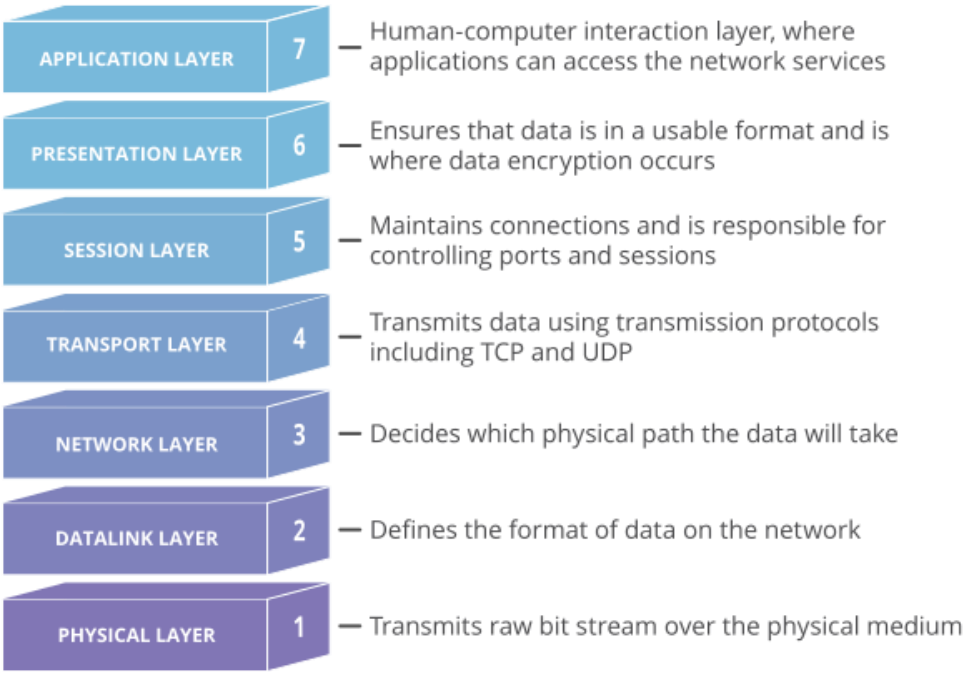
Tietokannoissa säilytetään usein luottamuksellista tietoa, tyypillisesti liittyen sitä ylläpitävään tahoon tai sen sidosryhmiin kuten työntekijöihin, asiakkaisiin ja kumppaneihin. Monien yritysten ja organisaatioiden täytyy kuitenkin tarjota turvattua pääsyä tietoihin muun muassa edellä mainituille sidosryhmille. (Stallings & Brown 2018, 170.)

Tietokannanhallintajärjestelmien nykyaikainen monimutkaistuminen tarjoaa paljon haasteita tietokannan turvallisuuden kannalta. Koska tietokannanhallintajärjestelmät tarjoavat paljon vaihtoehtoja tietokannan määrittelyä varten, niissä on tietokannan ylläpitäjän haasteeksi myös paljon turvallisuusuhkia ja haavoittuvaisuuksia. (Stallings & Brown 2018, 171.)

Useimmissa organisaatiossa ei ole täysiaikaisia työntekijöitä huolehtimassa tietokannan turvallisuudesta, vaan nämä asiat jäävät tietokannan ylläpitäjän vastuulle. Tietokannan ylläpitäjät keskittyvät yleensä tietokannan rakenteeseen ja toiminnallisuuteen, joten he eivät aina ehdi perehtyä tarpeeksi hyvin tietokannan turvallisuuteen. Tietokantojen turvallisuus ei oikein ole pysynyt tietokantojen käyttöasteen perässä, varsinkin kun organisaatiot ovat nykyään riippuvaisempia pilvipalveluista. (Stallings & Brown 2018, 171.)

3.4.1 OSI-malli

Kun keskustellaan Internetin välityksellä liikkuvan tiedon turvallisuudesta, on tärkeää tuntea OSI-malli ja omaksua pääpiirteisesti siinä esiteltävät kerrokset.



Kuva 2. OSI-malli (Cloudflare s.a.)

OSI-malli (Open Systems Interconnection) on konseptipohjainen malli, joka mahdollistaa erilaisten järjestelmien kommunikoimisen keskenään käyttäen standardoituja protokollia. OSI-mallia voidaan pitää tietoverkkojen universaalina kommunikointijärjestelmänä. (Cloudflare s.a.)

OSI-malli jakautuu seitsemään kuvassa 2 kuvattuun kerrokseen ja ne esitellään seuraavaksi ylhäältä alas:

7: Sovelluskerros - Application layer

Sovelluskerros on ainoa kerros, jolla on vuorovaikutusta käyttäjän tietojen kanssa. Sovellusohjelmat, kuten web-selaimet ja sähköpostiohjelmat tarvitsevat sovelluskerrosta kommunikoinnin aloittamiseen. Asiakaslaitteiden sovellukset eivät itsessään kuulu sovelluskerrokseen, vaan ne käyttävät sovelluskerroksen protokollia ja tiedon manipulointia esittääkseen käyttäjälle käyttökelpoista tietoa. (Cloudflare s.a.)

6: Esitystapakerros - Presentation layer

Esitystapakerroksen ensisijainen tehtävä on käsitellä liikennöitävä tieto sovelluskerroksen käytettäväksi. Esitystapakerroksen tiedon käsittelyyn kuuluu sen tulkitseminen, salaus ja pakkaaminen.

Tulkitseminen: Kommunikoivat laitteet voivat käyttää erilaisia koodausmenetelmiä tiedon kanssa, jolloin sitä täytyy tulkita esitystapakerroksessa.

Salaaminen: Jos laitteet kommunikoivat salatun yhteyden välillä, esitystapakerros lisää salauksen lähettäjän päähän ja purkaa salauksen vastaanottajan päässä.

Pakkaaminen: Esitystapakerros pakkaa tiedon kompaktimpaan muotoon viidennettä kerrosta varten. Tämä tekee kommunikoinnista tehokkaampaa ja nopeampaa. (Cloudflare s.a.)

5: Istuntokerros - Session layer

Istuntokerros avaa ja sulkee istuntoja kommunikoivien laitteiden välillä. Sen tehtävänä on varmistaa, että istunto pysyy päällä tietoliikenteen aikana ja sulkeutuu heti kun sitä ei enää tarvita. Istuntokerros myös seuraa tiedon siirron edistystä, jotta se osaa katkojen tapahtuessa jatkaa siitä missä tiedonsiirto keskeytyi. (Cloudflare s.a.)

4: Kuljetuskerros - Transport layer

Kuljetuskerros vastaa laitteiden päästä päähän tapahtuvasta kommunikoinnista kahden laitteen välillä. Tähän sisältyy tiedon ottaminen istuntokerroksesta ja sen pilkkomisesta segmentteihin kolmatta kerrosta varten lähettäjän päässä, sekä segmenttien kokoamisesta käyttökelpoiseksi tiedoksi viidennellä kerroksella vastaanottajan päässä. (Cloudflare s.a.)

Kuljetuskerros on myös vastuussa tietovirran ja virheiden hallinnasta. Tietovirran hallinta määrittelee sopivan nopeuden tiedon lähettämiselle, jotta lähettä-

jän tiedonsiirtonopeus ei ylitä vastaanottajan kapasiteettia. Kuljetuskerros varmistaa vastaanottavassa päässä, että kaikki tieto on tullut perille ja pyytää muutoin tiedon uudelleenlähetystä. (Cloudflare s.a.)

3: Verkkokerros - Network layer

Verkkokerros on vastuussa tiedonsiirrosta eri verkkojen välillä. Jos kaksi laitetta kommunikoi samassa verkossa, ne eivät tarvitse verkkokerrosta. Verkkokerros purkaa kuljetuskerroksesta tulleet segmentit paketeiksi lähettäjän päässä ja kasaa ne vastaanottajan päässä. Verkkokerros myös löytää parhaan fyysisen reitin tiedon ja sen päämäärän välillä, tätä kutsutaan reitittämiseksi. (Cloudflare s.a.)

2: Siirtokerros - Datalink layer

Siirtokerros on samankaltainen verkkokerroksen kanssa, paitsi että se mahdollistaa tiedonsiirron kahden samassa verkossa olevan laitteen välillä. Siirtokerros ottaa paketit verkkokerroksesta ja purkaa ne kehyksiin. Siirtokerros vastaa sisäverkon tietovirran ja virheiden hallinnasta. (Cloudflare s.a.)

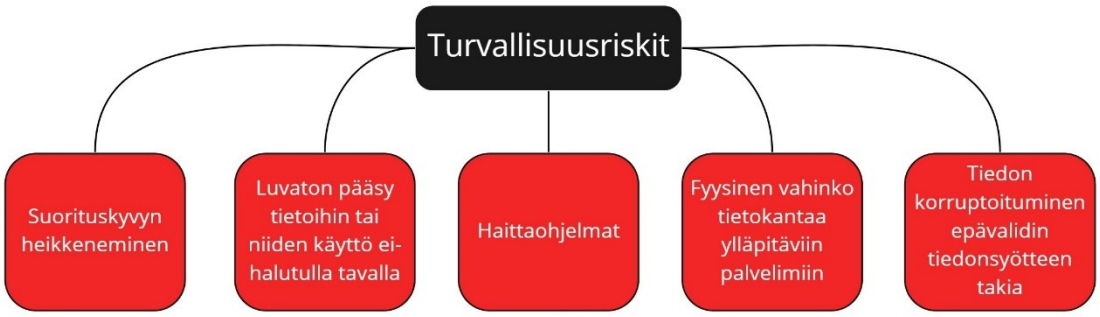
1: Fyysinen kerros - Physical layer

Fyysiseen kerrokseen kuuluvat laitteet ja liitännät, joiden kautta tieto kulkee tietokoneiden ymmärtämässä binäärisessä muodossa. Laitteiden fyysisten kerrosten täytyy noudattaa samaa signalointitapaa, jotta ne voivat tulkita tietoa samalla tavalla. (Cloudflare s.a.)

Ihmisten välittäessä tietoa verkon välillä laitteelta toiselle tiedon on kuljettava OSI-mallia kerroksittain alas lähettävässä päässä ja kuljettava samat kerrokset ylöspäin vastaanottavassa päässä (Cloudflare s.a).

3.4.2 Turvallisuusriskit

Koska tietokannat säilyttävät luottamuksellista tietoa, ne ovat suuremmassa vaarassa joutua esimerkiksi hakkereiden kohteeksi. Riskejä on monenlaisia, mutta kuvassa 3 näkyy päällimmäisiä tietokantoihin kohdistuvia riskejä.



Kuva 3. Tietokantojen päällimmäisiä riskejä (Tanna & Singh 2018, 169)

Näiden turvallisuusriskien välttämiseksi on olemassa erilaisia protokollia ja standardeja. (Tanna & Singh 2018, 169.)

Edellisen kappaleen lähteeseen ja kuvaan 3 viitaten erilaisiin turvallisuusriskien välttämiseen liittyviä protokollia ja standardeja esitellään seuraavaksi.

Pääsyn valvonta

Pääsyn valvontaan kuuluu käyttäjän tunnistautuminen sekä auktorisointi asianmukaisiin tietoihin. Tietokantajärjestelmistä löytyy tyypillisesti kirjautumisjärjestelmä, jonka kautta käyttäjät kirjautuvat sisään käyttäjätunnuksilla sekä auktorisointijärjestelmä, joka valvoo käyttäjien pääsyä tiettyihin tietoihin ja tietokantoihin. (Tanna & Singh 2018, 169.)

Tietokannan käytön tarkastelu

Tietokannan käytön tarkastelu tarkoittaa käyttäjien toiminnan seuraamista tietokannassa epäilyttävän toiminnan havaitsemiseksi. Monet tietokannat tarjoavat työkaluja tietokannan käytön tarkasteluun, jotta voidaan helposti havaita ketkä ovat poistaneet, lisänneet tai muokanneet tietoa ja milloin he ovat sen tehneet. (Tanna & Singh 2018, 169.)

Varmuuskopiointi

Varmuuskopioiden on tarkoitus säilyttää tietoa siltä varalta, että alkuperäinen tieto häviää tai korruptoituu. Varmuuskopiointiprosessit voivat olla manuaalisia

tai automaattisia, mutta on suositeltavaa ylläpitää automatisoitua ja säännöllistä varmuuskopiointiprosessia. Varmuuskopiot voivat vaatia tiedon määrästä riippuen paljon elektronista säilytystilaa, jolloin ne kannattaa pitää pakattuna pienempään muotoon. (Tanna & Singh 2018, 169.)

Tiedon eheyden varmistaminen

Tiedon eheydellä tarkoitetaan tietokantaan tallennetun tiedon johdonmukaisuutta ja tarkkuutta. Tieto täytyy eheyden varmistamiseksi myös validoida, eli tarkistaa että se on oikeanlaisessa muodossa tietokantaa varten. Monissa relaatiotietokannoissa on rajoitteita, jotka auttavat tiedon eheyden varmistamisessa, kuten pääavain ja viiteavain. NoSQL-tietokannoissa tiedon validointi tulee suorittaa tietokannan kerroksella sekä sovelluskerroksella tiedon eheyden varmistamiseksi. (Tanna & Singh 2018, 170.)

Sovelluskerroksen turvallisuus

Myös sovelluskerrokselle on kiinnitettävä huomiota, jotta asiaankuulumatonta tietoa ei päätyisi tietokantaan. Kehittäjät tyypillisesti validoivat tietoa eri kerroksilla varmistaakseen tietokantaan tallennettavan tiedon olevan validia. (Tanna & Singh 2018, 170.)

Salaus

Henkilökohtainen tieto, kuten sosiaalitunnukset ja maksutiedot tulee salata niiden väärinkäytön välttämiseksi. Asiakaslaitteen ja palvelimen välinen yhteyskin on tyypillisesti salattu SSL-salauksella (Secure Sockets Layer), verkkokerroksella tapahtuvalla tasolla. (Tanna & Singh 2018, 170.)

3.4.3 Autentikointi

Autentikointi tarkoittaa tietoteknisessä asiayhteydessä käyttäjän tai laitteen tunnistamista. Yleinen esimerkki on käyttäjänimen ja salasanan syöttö jollekin web-sivulle. Oikeiden tietojen syöttö antaa web-sivun tietää kuka on kyseessä, ja että se on juuri hän, joka on kirjautumassa sisään. (Tech Terms 2018.)

Vaikka käyttäjänimeä ja salasanaa käytetään usein tunnistukseen, on olemassa monia muitakin tunnistautumistapoja kuten 4- tai 6-numeroinen koodi puhelimen avaamiseksi. (Tech Terms 2018.)

Vaikka autentikointi auttaa pitämään henkilökohtaisia tietoja salassa, se ei ole täysin idioottivarma käytäntö. Joku voi esimerkiksi päästä käsiksi toisen sähköpostiin arvaamalla tämän salasanan. Tämän takia on tärkeää käyttää yksilöllisiä vaikeasti arvattavissa olevia salasanoja, etenkin sähköpostin kanssa. On myös suositeltavaa käyttää kaksikerroksista autentikointia, kun sellainen on saatavilla. (Tech Terms 2018.)

Kaksikerroksinen autentikointi (2FA) vaatii tyypillisesti oikeat kirjautumistiedot, ja suorittaa sen lisäksi toisenlaisen tarkastuksen. Esimerkiksi pankkiin kirjautuessa, käyttäjän täytyy syöttää kirjautumistietojen lisäksi puhelimeen tai sähköpostiin tullut väliaikainen koodi. (Tech Terms 2018.)

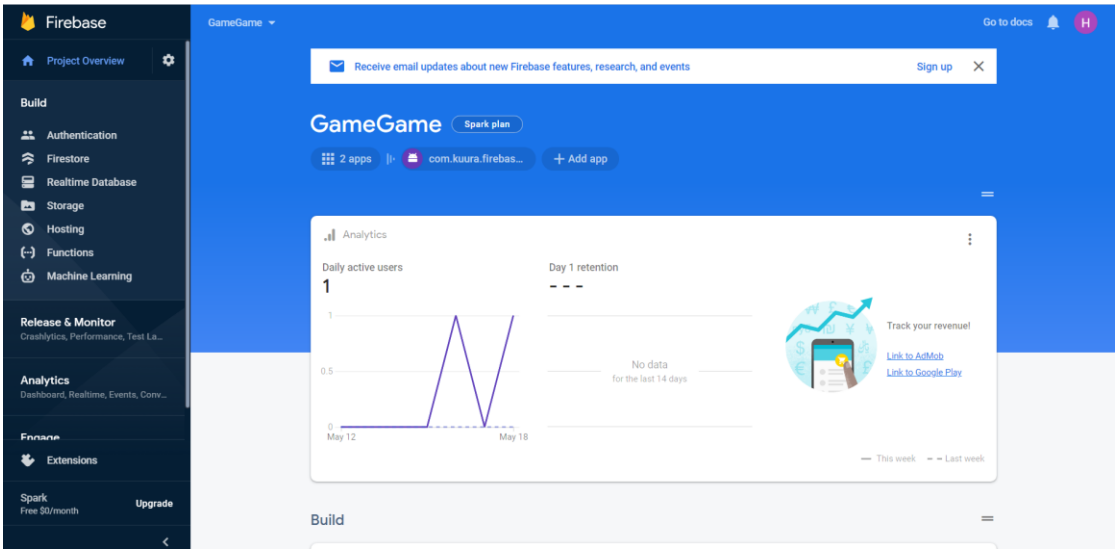
4 GOOGLE FIREBASE KEHITYSALUSTA

Firebase on Googlen tarjoama web- ja mobiilisovellusten kehityspohja, joka koostuu monista tuotteistetuista osista ja keskittyy reaaliaikaiseen tiedon käsittelyyn. Firebase tarjoaa pääsijaisesti kahta reaaliaikaista tietokantaa: Realtime Databasea sekä Cloud Firestorea. Molemmat tietokannat perustuvat NoSQL-tietomalliin ja toimivat reaaliaikaisesti tarvittaessa yhdessä tai erikseen. Firestore on ominaisuuksiltaan rikkaampi kuin Realtime Database, mutta Game Game -projektissa käytetään ainoastaan jälkimmäistä ja siihen viitataan jatkossa yksinkertaisesti reaaliaikaisena tietokantana. (Tanna & Singh 2018, 11–17.)

Firebase-kehitysalustan toimintaa ymmärtääkseen pitää tuntea ohjelmiston piiriin kuuluvat käsitteet front-end ja back-end. Front-end tarkoittaa ohjelmiston käyttäjälle näkyvää puolta, kuten sen käyttöliittymää. Back-end tarkoittaa ohjelmiston taustalla toimivaa infrastruktuuria, joka pitää sen kaikin puolin toiminnallisena. (Concepta Tech 2019.)

Firebase-kehitysalustaa voi kuvailla back-endiä palveluna tarjoavana (BaaS) kehitysalustana, joka sisältää nopeasti ja helposti käyttöönotettavia back-end

ominaisuuksia web-kehittäjille. Sen ydinominaisuuksiin kuuluu muun muassa reaaliaikaisen tiedon käsittely, sovelluksen kasvattaminen, sovelluksen analysointi, käyttäjien hankkiminen ja turvallisuuden varmistaminen. (Tanna & Singh 2018, 7–8).



Kuva 4. Firebase-kehitysalustan kehityskonsoli

Firebase-kehitysalustaa ohjataan kuvassa 4 näytetyn kehityskonsolin kautta, ja sinne voi kirjautua Google-tunnuksilla. Mikäli Google-tunnuksia ei ole, ne voi luoda ilmaiseksi.

Seuraavissa luvuissa esitellään Game Game -projektissa käytettyjä Firebase-kehitysalustan pääominaisuuksia.

4.1 Reaaliaikainen tietokanta

Firebase-kehitysalustan reaaliaikainen tietokanta on pilvessä isännöity NoSQL-tietokanta, joka synkronoi tietokannan tiedot kaikille siihen yhdistäneille asiakasohjelmille perinteisen pyyntö-vastaus mallin sijaan. Firebase säilyttää tietoa myös kovalevyllä, joten kommunikointi ei katkea heti yhteydessä ilmentyneistä katkoista ja tieto synkronoidaan yhteyden palatessa takaisin. Reaaliaikainen tietokanta tukee iOS ja Android-tekniikkaa, sekä Game Game -projektin kannalta olennaisesta Unity-pelimoottoria, joka esitellään seuraavassa pääluvussa. (Tanna & Singh 2018, 12.)

Firebase-kehitysalustan reaaliaikaiseen tietokantaan pääsee käsiksi suoraan asiakaslaitteella suoritetusta koodista, jolloin ei tarvita erillistä palvelinta toimimaan välikätenä asiakaslaitteen ja tietokannan välillä. Firebasessa on käytössä ns. turvallisuussäännöt, jotka määrittelevät kenellä on pääsy mihinkin tietoihin ja jokaista tiedonhakupyyntöä valvotaan turvallisuussääntöjen määritysten mukaisesti. (Firebase Realtime Database 2021.)

Tietoa säilytetään Firebase-kehitysalustan reaaliaikaisessa tietokannassa JSON-muodossa, jota käsitellään tarkemmin tämän luvun viidennessä aliluvussa. Firebase-kehitysalustan reaaliaikainen tietokanta käyttää hierarkkista rakennetta, joka koostuu tietoa säilyttävistä solmuista (node). Jokaisella solmulla on oma yksilöivä avain, josta noodin tunnistaa. Avaimen voi tarjota itse, tai antaa järjestelmän luoda noodille uuden avaimen. Solmuilla voi hierarkkisen rakenteen mukaisesti olla alisolmuja 32:een sisäkkäiseen alisolmuun asti. Firebase-kehitysalustan ohjeistus neuvoo kuitenkin välttämään alisolmutusta mahdollisimman paljon, koska se nostaa useasti esiintyvän saman tiedon riskiä sekä tekee tiedon käytöstä ja sen käytön valvonnasta monimutkaisempaa. (Firebase Realtime Database 2021.)

4.2 Autentikointi

Firebase tarjoaa yksinkertaisen ja turvallisen ratkaisun käyttäjien autentikointiin mobiili- ja websovelluksia varten. Siihen kuuluu useita autentikointitapoja, mainittavimpina tavanomainen lomakepohjainen sähköposti ja salasana, kirjautuminen kolmannen osapuolen kuten Twitter- tai Facebook-palveluiden kautta sekä mahdollisuus liittää se suoraan kehittäjän omaan kirjautumisjärjestelmään. (Tanna & Singh 2018, 21.)

Kolmannen osapuolen kirjautumisvaihtoehtoja soveltaessa tulee käytettyä eräänlaisia käyttöoikeustietueita (access token). Käyttöoikeustietueet tekevät rajapintapyyntöjä käyttäjän puolesta ja edustavat tietyn ohjelman pääsyä tiettyihin käyttäjän tietoihin. Käyttöoikeustietueet ovat luottamuksellista tietoa ohjelmaa, tunnistuspalvelinta sekä resurssipalvelinta varten. (OAuth s.a.)

Käyttäjä tarvitsee kirjautuessaan ensiksi autentikointitunnukset. Nämä voivat olla autentikointitapojen mukaisesti käyttäjän sähköposti ja salasana, tai käyttöoikeustietue valtuutetulta henkilöllisyyden tarkastavalta osapuolelta. Autentikointitunnukset pitää sen jälkeen toimittaa Firebase-kehitysalustan autentikointijärjestelmälle, jolloin Firebase-kehitysalustan back-end palvelut tarkastavat autentikointitunnukset ja lähettävät asianmukaisen vastauksen asiakaslaitteelle. (Firebase Authentication 2021.)

4.3 Tiedon varastointi

Firebasessa on pilvipohjainen tietovarasto, jonka avulla ohjelmat voivat ladata ja lähettää tiedostoja tietovaraston välillä. Se tukee monia erilaisia tiedostomuotoja, mukaan lukien valokuvia, äänitiedostoja sekä videoita. Tiedostojen lataamista ja lähettämistä voi hallita Firebase-kehitysalustan ohjelmistokehityspaketin (SDK) avulla, ja Firebase-kehitysalustan tietovarasto sietää heikosta yhteydestä johtuvia katkoja kuormittamatta liialti käyttäjien Internet-kaistaa tai tuhlaamatta heidän aikaansa. Firebase-kehitysalustan tietovarasto on yksi sen tärkeimmistä ominaisuuksista, ja se tukee muun muassa iOS-, Android-, Web-, C++- ja Unity-alustoja. (Kumar 2018, 138.)

Firebase-kehitysalustan tietovarasto on rakennettu skaalautuvaksi, ja tukee tarvittaessa suurta määrää tietoa eksatavuihin asti. Firebase-kehitysalustan tietovarasto säilyttää tietoa niin kutsutuissa ämpäreissä. Firebase-projektiin voi tarpeen mukaan konfiguroida useita ämpäreitä, mutta niitä on lähtökohtaisesti yksi projektia kohden. (Kumar 2018, 138–139.)

Firebase-kehitysalustan tietovaraston tiedostojärjestelmä on samankaltainen kuin tyypillinen kansiorakenne useimmissa käyttöjärjestelmissä, tosin siinä kansioiden vastineet mielletään reaaliaikaisen tietokannan tapaan solmuina ja alisolmuina. Tietovarasto hyödyntää edellisessä aliluvussa käsiteltyä autentikointijärjestelmää valvoakseen pääsyä tietovarastossa olevaan tietoon. (Kumar 2018, 139.)

4.4 Turvallisuus

Turvallisuus on erittäin tärkeää ottaa huomioon Firebase-kehitysalustan, kuten minkä tahansa tietokannan kanssa. Muuten tietoa voi päätyä vääriin käsiin tai

sitä voidaan väärinkäyttää, joka altistaa Firebase-kehitysalustaa ylläpitävän tahon taloudellisille tappioille ja lakisyytteille. Tiedon luottamuksellisuus ei ole ainoa siihen liittyvä turvallisuushuoli, vaan myös sen saatavuus ja eheys on varmistettava. (Tanna & Singh, 2018. 168.)

Firebase-kehitysalustan turvallisuuteen voi vaikuttaa sen konfiguroinneilla, turvallisuussäännöillä, autentikoinnilla, auktorisoinnilla sekä sovelluksen toimintatavoilla, joita käydään läpi kuudennessa luvussa. Jokainen Firebase-projekti on erilainen ja tarvitsee sille sopivat turvallisuussäädöt.

Kehitettävän sovelluksen osalta on tärkeää tutustua Firebase-ohjelmistokehityspaketin tarjoamiin luokkiin ja metodeihin, jotta tiedon kulku sujuu suunnitellusti. Asiakas-laitteilla ei tulisi kulkea tietoa, johon sen käyttäjällä ei kuulu olla pääsyoikeutta.

4.4.1 Turvallisuussäännöt

Turvallisuussäännöt vastaavat Firebase-kehitysalustan käytön valvonnasta ja suojaavat Firebase-kehitysalustan reaaliaikaisissa tietokannoissa sekä tietovarastossa säilytettyä tietoa, välttääkseen sitä päätymästä vääriin käsiin. Firebase-kehitysalustan turvallisuussäännöt hyödyntävät joustavaa konfigurointikieltä määrittääkseen, mihin tietoon käyttäjillä on pääsy Firebase-kehitysalustan reaaliaikaisessa tietokannassa, Firestoressa sekä Firebase-kehitysalustan tietovarastossa ja miten he pääsevät siihen käsiksi. (Firebase Security Rules 2021.)

Firebase-kehitysalustan reaaliaikainen tietokanta käyttää JSON-ohjelmointikieltä turvallisuussääntöjen määrittelyyn, kun taas Firestore ja Firebase-kehitysalustan tietovarasto käyttävät omaa ainutlaatuista ohjelmointikieltä siihen tarkoitukseen (Firebase Security Rules 2021).

Turvallisuussäännöt ovat tärkeä konfiguroida kuntoon ennen tietokannan avaamista ulkoisille käyttäjille. Ne tulisi myös testata ennen tietokannan käyttöönottoa ja päivittää aina tarvittaessa vastaamaan tietokannan rakenteellisia muutoksia.

4.5 JSON

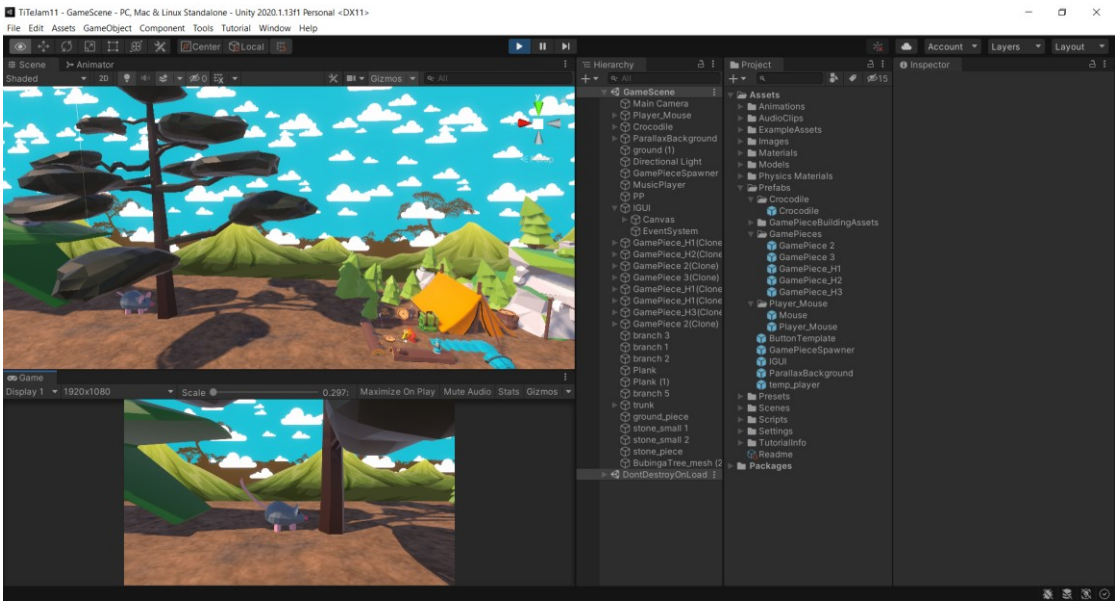
JSON (Javascript Object Notation) on JavaScript-ohjelmointikielestä periytyvä tekstipohjainen, kevyt ja ihmisille lukukelpoinen tiedonsiirtoformaatti, jonka avulla asiakaslaitteet ja palvelimet voivat siirtää tietoa keskenään. JSON ei ole riippuvainen tietystä ohjelmointikielestä, ja sitä tukevat JavaScriptin lisäksi kaikki suositut ohjelmointikielet kuten C#, PHP, Java, C++, Python ja Ruby. (Sriparasa & D'mello 2018.)

JSON on ihmisille helppo luettavaksi ja kirjoitettavaksi, sekä tietokoneille tulkittavaksi ja generoitavaksi. Se perustuu JavaScript-ohjelmointikielen alalajeihin. Sen riippumattomuus ohjelmointikielestä tekee siitä ideaalisen tiedonsiirtoformaatin. JSON perustuu avain-pareihin sekä arvolistoihin, joihin löytyy vastine miltei kaikista ohjelmointikielistä. (JSON.org s.a.)

JSON-formaattia käytetään Game Game -projektissa esimerkiksi tietokannan varmuuskopioimiseen sekä urheiluseurojen pelaajien tuomiseen. JSON-formaatin käsittely on helppoa, ja JSON-tiedostoista näkeekin nopeasti lukemalla mitä niissä on. Unity tukee JSON-formaatin käyttöä, mahdollistaen tiedon siirron asiakaslaitteiden ja tietokannan välillä.

5 UNITY JA FIREBASE

Unity on 2D- ja 3D-pelejä tukeva pelimoottori ja ohjelmointiympäristö (IDE). Unity-kehitysalustan pelimoottori tarjoaa pelien käytettäväksi tärkeitä ominaisuuksia, kuten fysiikan, 3D-renderöinnin ja -törmäysten havaitsemisen. Tämä hyödyttää pelinkehittäjiä, koska he voivat edellä mainittujen asioiden sijaan keskittyä varsinaisen pelin tekemiseen. (Sinicki 2021.)



Kuva 5. Unity-kehitysalustan pelieditori

Unity voidaan määritellä ohjelmointiympäristöksi, sillä se tarjoaa kattavan kokoelman työkaluja pelien kehittämiseen. Unity-kehitysalustalla on visuaalinen pelieditori, jonka avulla pelinkehittäjät voivat käsitellä pelin elementtejä ja navigoida projektikansioiden läpi. (Sinicki 2021.)

Unity-kehitysalustan visuaalinen pelieditori esitellään kuvassa 5. Unity käyttää pelinkehityksessä pääsääntöisesti C#-ohjelmointikieltä ja tukee sitä varten monia eri tekstieditoreita, erityisesti Microsoftin Visual Studio -ohjelmointiympäristöä. Koodilla määritellään pelin kulku ja siinä olevien asioiden käyttäytymistä. Koodi liitetään pelissä olevaan asiaan tai tasoon koodia sisältävällä tekstitiedostolla, jota kutsutaan skriptiksi (Unity s.a.)

Unity-kehitysalustan Cloud Build-palvelu mahdollistaa Unity-projektien kääntämisen monille alustoille ja käyttöjärjestelmille. Järjestelmä tukee iOS-, Android-, Windows-, Mac-, Linux-, WebGL- ja Unity Web Player -teknologioita. (Unity 2021.)

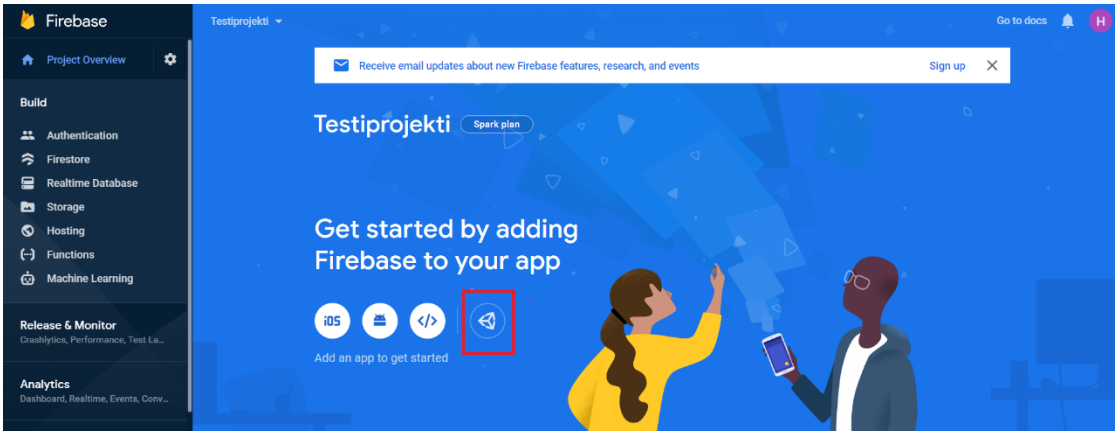
5.1 Firebase Unityssä

Firebase tukee Unity-kehitysalustaa ja tarjoaa siihen Firebase-ohjelmistokehityspaketin. Tässä luvussa ohjeistetaan, miten Androidiin keskittyvä Unity-projekti voidaan kytkeä Firebase-kehitysalustan kehitysalustaan käyttäen Fire-

base-kehitysalustan virallisia ohjeita, joihin voi tutustua liitteen 1 ensimmäisessä web-osoitteessa. Tämän avulla saa käsityksen siitä, miten Firebase-kehitysalustan ja Unity-kehitysalustan suhde toimii ja miten yhteyden saa näiden välille perustettua.

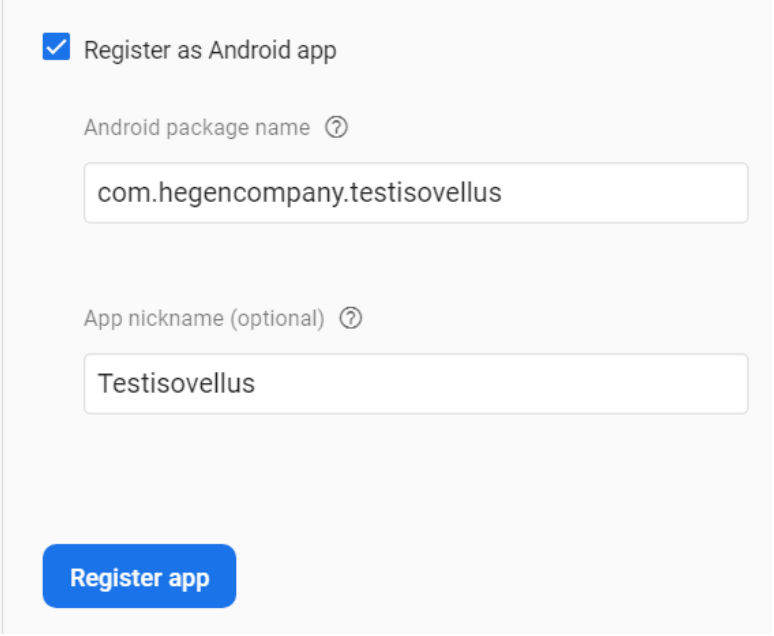
Ensiksi täytyy kirjautua Firebase-kehitysalustan kehityskonsoliin liitteen 1 toisesta web-osoitteesta. Sisään kirjautuakseen pitää löytyä Google-tunnukset, jotka voi tarvittaessa luoda samasta web-osoitteesta löytyvän linkin kautta. Tämän jälkeen pääsee projektivalikkoon, jossa on luotava uusi projekti painamalla “Add project” -painiketta. Uudelle projektille annetaan ohjatun luontiprosessin määrityksiä, kuten projektin nimi.

Kun projekti on luotu, Firebase-kehitysalustaan täytyy rekisteröidä sovellus painamalla tässä tapauksessa Unity-kehitysalustan ikonilla varustettua painiketta, joka on korostettuna kuvassa 6.



Kuva 6. Unity-painike Firebase-kehitysalustan kehityskonsolissa

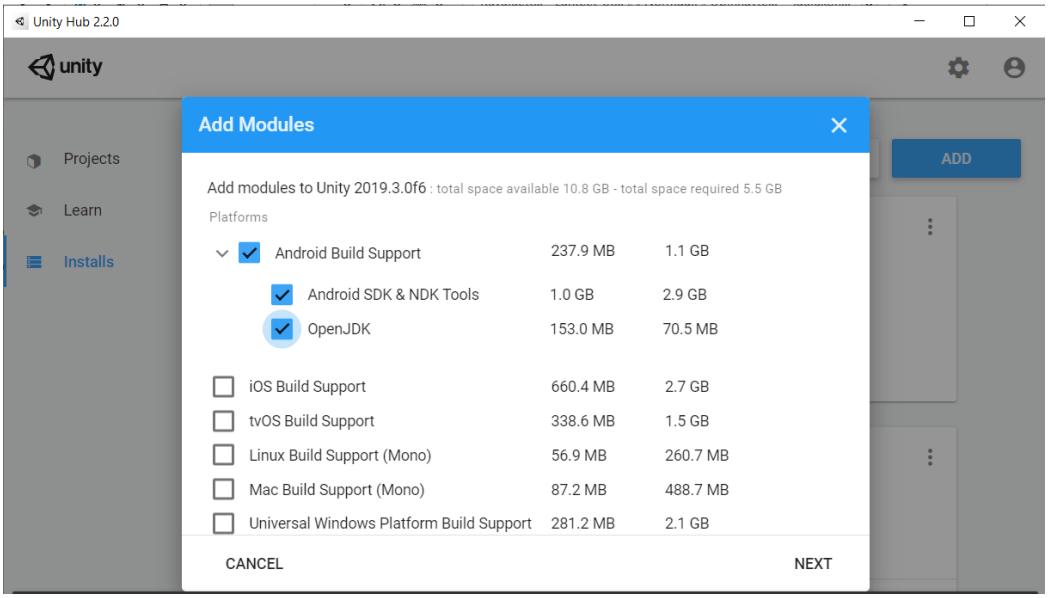
Sovelluksen voi rekisteröidä halutessaan Android- tai iOS-laitteille. Opinnäytetyön Game Game -projekti keskittyy Android-käyttöliittymän tukemiseen, joten tässä vaiheessa rekisteröidään sovellus Android-käyttöjärjstelmälle. Android-paketille täytyy antaa tiettyä muotoa noudattava nimi, kuten kuvassa 7. Kyseisellä web-sivulla kerrotaan Android-paketin nimeämisen säännöistä lisää, mikäli nimi ei vastaa oikeaa muotoa.



Kuva 7. Esimerkki sovelluksen ja sen paketin nimeämisestä

Sovelluksen ja sen paketin nimeämisen jälkeen, sovelluksen voi rekisteröidä. Tämän jälkeen Firebase tarjoaa ladattavasi Unity-kehitysalustaa varten JSON-muodossa olevan konfigurointitiedoston, joka liitetään sivulla ohjeistettuun paikkaan käyttäen Unity-kehitysalustan pelieditoria. Viimeiseksi Firebase tarjoaa ladattavaksi Firebase-kehitysalustan uusimman ohjelmistokehityspaketin Unity-kehitysalustaan ja ohjeet sen asentamiseksi.

Unity tarvitsee Android sovellusten kuten Game Game -sovelluksen kehittämiseen myös Android-ohjelmistokehityspaketin. Sen voi asentaa haluamalleen Unity-pelieditorin versiolle käyttämällä Unity-kehitysalustan tarjoamaa Unity Hub-sovellusta, joka on tarkoitettu Unity-projektien ja -pelieditorien hallitsemiseen.



Kuva 8. Android-ohjelmistokehityspaketin asentaminen

Android-ohjelmistokehityspaketin voi asentaa kuvassa 8 näytetyn Unity Hub-sovelluksen avulla painamalla haluamansa pelieditoriversion valikosta ”Add Modules” -painiketta. Kuvassa 8 on myös valittuna OpenJDK, joka on Java-ohjelmointikielen kehityspaketti. Se on kooltaan pieni verrattuna Android-käyttöjärjestelmän ohjelmistokehityspakettiin, joten sen asentaminen samassa yhteydessä on suositeltavaa.

6 TURVALLISUUDEN TOTEUTUS GAME GAME -PROJEKTISSA

Tämän luvun tarkoituksena on dokumentoida Firebase-kehitysalustan turvallisuusratkaisuja Game Game -projektin näkökulmasta. Game Game -projekti ei dokumentoinnin hetkellä ole saatavilla kuluttajakäyttäjille, eikä siihen ole implementoitu kaikkia tarkoitettuja ominaisuuksia.

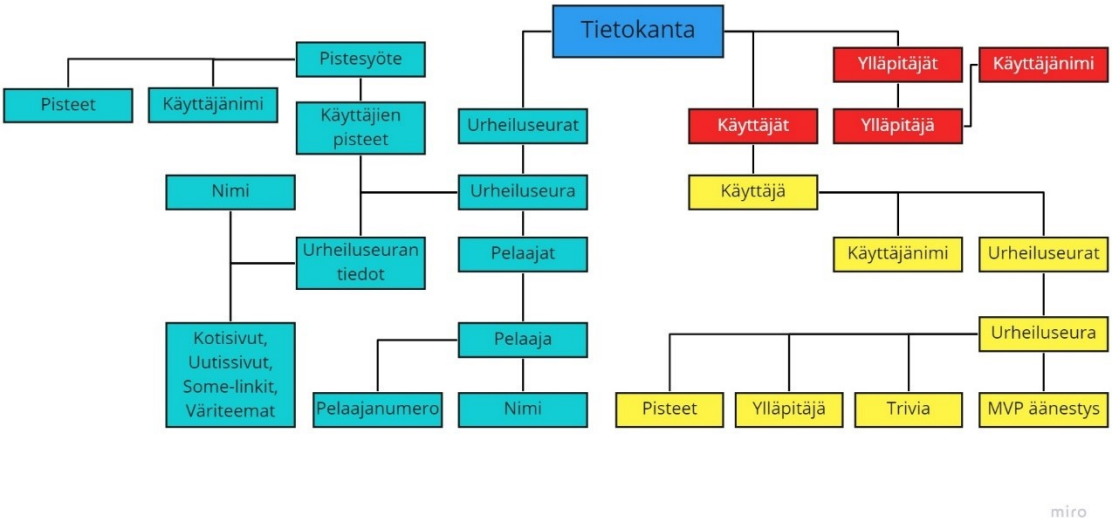
Turvallisuuden toteutus on luonteeltaan yleispiirteinen, joten sitä voi soveltaa myös muihin projekteihin kuin Game Game -projektiin. Tehdyt ratkaisut perustuvat tutkittuun tietoon ja Game Game projektin olosuhteisiin ja käytössä oleviin resursseihin. Toteutuksessa ei laajemmin huomioida esimerkiksi Firebase-kehitysalustan Firestore-tietokantaa, koska sellaiselle ei toistaiseksi ole ollut Game Game -projektissa ollut tarvetta. Monet reaaliaikaisen tietokannan turvallisuuteen liittyvät asiat ovat kuitenkin rinnastettavissa Firestore-tietokantaan.

Game Game -projektin tietokannan turvallisuudessa käydään läpi tietokannassa säilytettävän tiedon analysointi ja luokittelu, Firebase-kehitysalustan konfigurointi, turvallisuussäännöt sekä yhteyden muodostaminen ja tiedon käsittely asiakaslaitteella. Näitä käydään läpi seuraavissa alaluvuissa.

6.1 Tietokannassa säilytettävän tiedon analysointi ja luokittelu

Game Game -projektissa hyödynnetään Firebase-kehitysalustan DataSnapshot-objekteja tuodakseen asiakaslaitteelle tietoa tietokannasta. DataSnapshot-objektilla viitataan tiettyyn tietokannan polkuun, jolloin se palauttaa siinä olevan solmun (luku 4.1) ja kaikki sen alta löytyvät alisolmut (DataSnapshot 2021).

Tietokannan rakenteeseen täytyykin siis kiinnittää huomiota, jotta asiakaslaitteet eivät säilytä DataSnapshot-objektien mukana tullutta ylimääräistä tietoa. Asiakaslaitteella olevan sovelluksen täytyy ottaa DataSnapshot-objektit tarpeeksi alhaisilta solmuilta tai varmistaa, että kaikki solmun alisolmuissa oleva tieto on turvallisuusmerkitykseltään samanarvoista.



Kuva 9. Tietokannan rakenne

Kuvassa 9 näytetään Game Game -projektin tietokannan rakennetta hierarkkisella tavalla. Tietokannan kaikille käyttäjälle tarkoitettu tieto on merkitty vaaleansinisellä, tietokannan ylläpitäjille tarkoitettu tieto on merkitty punaisella ja käyttäjälle näkyvät tiedot omista käyttäjätiedoista on merkitty keltaisella. Taulukossa 1 kuvaillaan tietokannan tietoja tarkemmin.

Taulukko 1. Kuvaukset tietokannan tiedoista

Juuri	Tieto	Kuvaus
Tietokanta	Ylläpitäjät	Sisältää tiedot tietokannan ylläpitäjistä.
Tietokanta	Käyttäjät	Sisältää tiedot tietokannan tavallisista käyttäjistä.
Tietokanta	Urheilu-seurat	Sisältää tietoa urheiluseuroista.
Ylläpitäjät	Ylläpitäjä	Yksilöllinen tunnus, joka erottaa ylläpitäjät toisistaan
Ylläpitäjä	Käyttäjä-nimi	Ylläpitäjän käyttäjänimi.
Käyttäjät	Käyttäjä	Yksilöllinen tunnus, joka erottaa käyttäjät toisistaan
Käyttäjä	Urheilu-seurat	Urheiluseurat, joiden aktiviteetteihin pelaaja on osallistunut.
Käyttäjä	Urheilu-seura	Yksilöllinen tunnus, josta tietokanta tunnistaa tietyn urheiluseuran.
Urheiluseura (Käyttäjä)	Ylläpitäjä	Käyttäjän asema urheiluseurassa. Ylläpitäjä tässä asiayhteydessä on todennäköisesti urheiluseuran jäsen tai kumppani.
Urheiluseura (Käyttäjä)	Trivia	Kertoo, onko käyttäjä suorittanut urheiluseuran viimekertaisen trivian
Urheiluseura (Käyttäjä)	MVP	Kertoo, onko käyttäjä jo äänestänyt MVP-ehdokasta
Urheiluseura (Käyttäjä)	Pisteet	Käyttäjän pisteet urheiluseuran aktiviteeteissa, kuten minipeleissä.
Urheiluseurat	Urheilu-seura	Urheiluseuran yksilöllinen tunniste, josta tietokanta tunnistaa tietyn urheiluseuran.
Urheiluseura	Pelaajat	Urheiluseuran pelaajat.
Urheiluseura	Urheilu-seuran tiedot	Urheiluseuran tiedot kuten nimi, kotisivut, Some-linkkejä ja väriteemaan liittyviä tietoja sovellukselle.
Urheiluseura	Käyttäjien pisteet	Käyttäjien pisteet urheiluseuran aktiviteeteissa kuten minipeleissä.
Pelaajat	Pelaaja	Pelaajan yksilöivä tunnus, joka tulee hänen pelaajanumerostansa.
Pelaaja	Pelaaja-numero	Pelaajan pelaajanumero.
Pelaaja	Nimi	Pelaajan nimi
Käyttäjien pisteet	Piste-syöte	Yksilöllinen tunnus, joka erottaa pistesyötteet toisistaan
Pistesyöte	Käyttäjä-nimi	Käyttäjän käyttäjänimi, joka näkyy urheiluseuran käyttäjäpistetaulukossa
Pistesyöte	Pisteet	Käyttäjän pisteet, jotka näkyvät urheiluseuran käyttäjäpistetaulukossa

Tämän tietokannan rakenteen tarkoituksena on lyöttää samoja turvallisuusluokituksia omaavia tietoja yhteen, jotta tiedonsiirto tietokannassa on helpompaa ja turvallisempaa. Myös tietokannan turvallisuuden konfigurointi helpottuu, koska turvallisuussääntöihin ei tarvitse tehdä paljon poikkeuksia. Tietokannan rakenteen visualisointi kuvan 9 kaltaisen kaavion ja taulukon 1 kaltaisen aputaulukon avulla auttaa hahmottamaan turvallisemman ja helppokäyttöisemmän tietokantarakenteen luomisessa.

6.2 Firebase-kehitysalustan konfigurointi

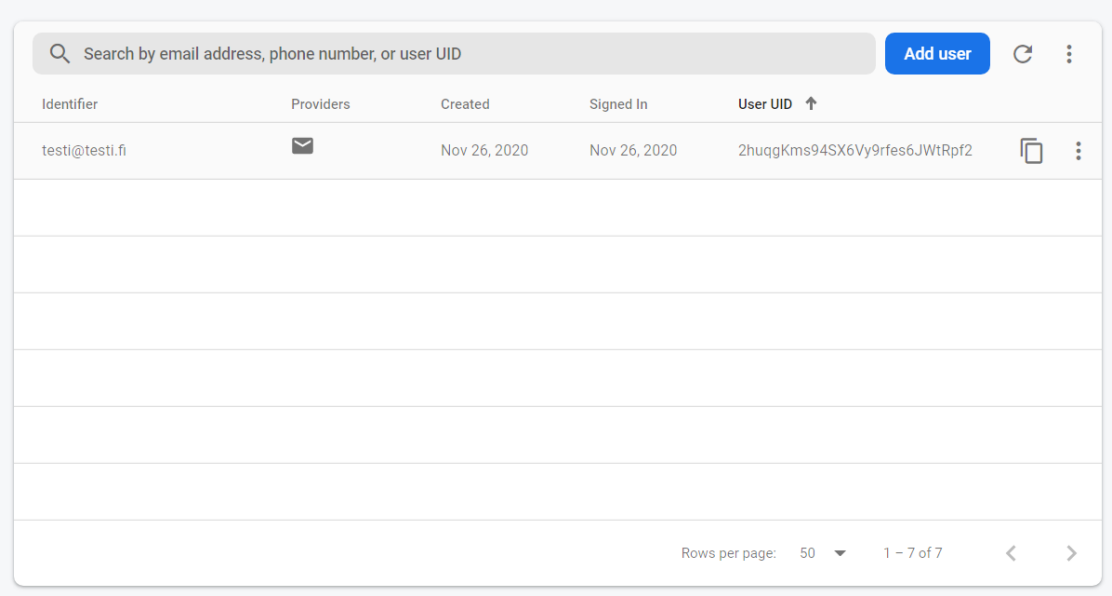
Tässä aliluvussa esitellään Firebase-kehitysalustan konfigurointia, jotka vaikuttavat Firebase-kehitysalustan turvallisuuden lisäksi sen perustoimintaan. Kaikki Firebase-kehitysalustan ominaisuudet eivät tarvitse yhtä paljon huomiota, mutta kaikki Game Game -projektiin liittyvät konfiguroinnit käydään läpi tämän luvun aliluvuissa. Konfiguroitaviin ominaisuuksiin kuuluu autentikointi, reaaliaikainen tietokanta sekä tietovarasto. Turvallisuussäännöt käydään erikseen läpi näiden konfigurointien jälkeen.

6.2.1 Autentikointi

Firebase-kehitysalustan autentikointikonfiguroinnit koskevat tietokannan käyttäjiä, sisäänkirjautumismenetelmiä ja viestintäpohjia. Autentikoinnin konfigurointivalikosta pääsee myös näkemään puhelimen välityksellä tehtyjä onnistuneita sisäänkirjautumisia.

Käyttäjät

Käyttäjien välilehdellä voi nähdä tietokannan käyttäjät, hakea tiettyä käyttäjää tai lisätä uuden käyttäjän. Käyttäjän tiedoissa näkyy kuvan 10 mukaisesti hänen tunnisteensa, luontipäivämäärä, viimeinen kirjautumispäivämäärä sekä käyttäjän tunnistamisessa ja yksilöimisessä auttava UID-tunnus. Kuvassa 10 demonstroidaan testaamiseen tarkoitettu tilapäinen käyttäjä.



Kuva 10. Käyttäjät

Käyttäjän oikealta puolelta löytyy kuvasta 10 valikko, jonka painike muistuttaa kolmea pistettä. Tästä valikosta tietyn käyttäjän salasanan voi resetoida, käyttäjän pääsyn voi estää tai käyttäjän voi poistaa kokonaan.

Salasanan resetointi lähettää käyttäjän sähköpostiin resetointilinkin, tätä voi käyttää esimerkiksi silloin kun käyttäjä on unohtanut salasanan. On kuitenkin suositeltavaa luoda salasanan resetointiin automatisoitu prosessi tavallisille käyttäjille, varsinkin jos heitä on paljon.

Käyttäjän pääsyn esto on hyödyllinen ominaisuus sisäisten sekä ulkoisten käyttäjien kanssa. Sisäisiä käyttäjiä voidaan haluta estää pääsemästä tietokantaan, mikäli tunnukset on tarkoitettu esimerkiksi asennus- ja testaustoimenpiteisiin. Tällöin heille annetaan väliaikaisesti pääsy kyseisten toimenpiteiden suorittamiseksi, jotta niihin kirjautuminen ei olisi turhaan mahdollista. Esimerkkinä tästä on kuvan 10 testikäyttäjä, joka on tarkoitettu testaustoimenpiteisiin.

Ulkoisia käyttäjiä voidaan haluta estää pääsemästä tietokantaan silloin, kun on kyse maksullisesta palvelusta ja käyttäjä ei ole maksanut viimeistä maksuerää tai hän on irtisanoutunut palvelusta poistamatta hänen käyttäjätiliään. Kun käyttäjä haluaa maksaa palvelusta taas uudelleen, hänen pääsynsä tietokantaan voidaan taas sallia.

Sisäisille sekä ulkoisille käyttäjille voi tulla tilanne, jossa heitä epäillään tietokannan tai sitä ylläpitävän organisaation sääntöjen rikkomisesta. Tästä seuraa tarvittaessa tutkinta, joka määrää käyttäjän jatkosta. Käyttäjän pääsyn esto voi myös toimia esiasteena käyttäjän poistamiselle, jos halutaan antaa käyttäjälle esimerkiksi viikko aikaa perua toiveensa tämän käyttäjätilin poistamisesta.

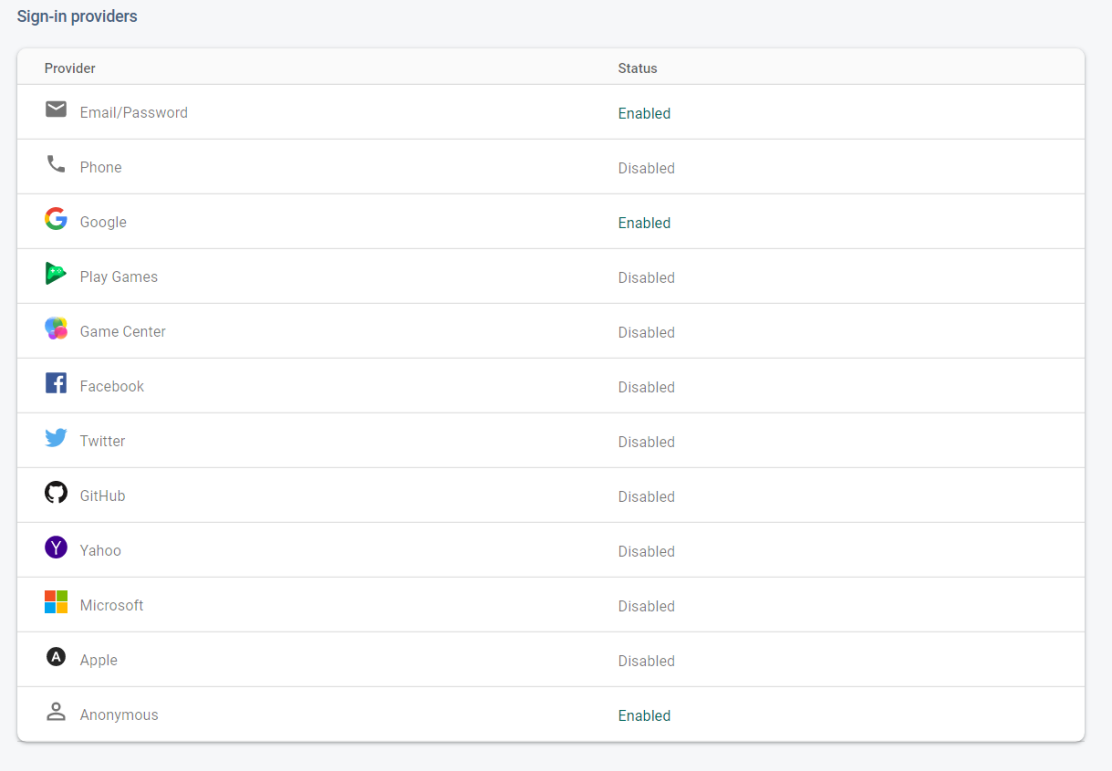
Käyttäjän poistaminen on lopullinen toimenpide, jossa käyttäjätili poistetaan. Tietokannoista ja tietovarastosta saattaa kuitenkin löytyä käyttäjästä tietoja tai tiedostoja, jotka halutaan myös poistaa. Tässä tapauksessa käyttäjän tietojen ja tiedostojen poistaminen on hyvä suorittaa ensin, koska tähän tarvittavat tunnistetiedot löytyvät vielä käyttäjätililtä, joka halutaan poistaa.

EU:n GDPR-regulaatio (General Data Protection Regulation) antaa yksityishenkilöille oikeuden pyytää organisaatioita poistamaan heitä koskevat tiedot. Organisaatioilla on lainmukainen määräys poistaa tiedot, paitsi silloin kun organisaatio tarvitse niitä sananvapauden harjoittamiseen, organisaatiolla on lainmukainen määräys säilyttää niitä tai tiedot palvelevat julkisia intressejä. (Euroopan Unioni s.a.)

Game Game -projektissa olevien käyttäjätietojen luonne ei vastaa GDPR-regulaation poikkeuksia, joten sen kuluttajakäyttäjille on tarjottava mahdollisuus poistaa heidän käyttäjätietonsa.

Sisäänkirjautuminen

Sisäänkirjautumisen välilehdellä voi määritellä käytössä olevat sisäänkirjautumispalvelut, sallitut verkkotunnukset, yhden tilin rajan sähköpostille ja väliaikaisen rajoituksen uusille sähköposteille tai hallita tilienluontia. Välilehdellä tulee ensiksi vastaan sisäänkirjautumispalvelut (Sign-in providers), jonka ikkunasta voi sallia tai estää haluamiaan sisäänkirjautumispalveluita. Niihin kuuluu sähköposti- ja salasana, muut sisäänkirjautumispalvelut sekä vieraskirjautuminen. Kaikki tuetut vaihtoehdot näkyvät kuvassa 11.



Kuva 11. Sisäänkirjautumispalvelut

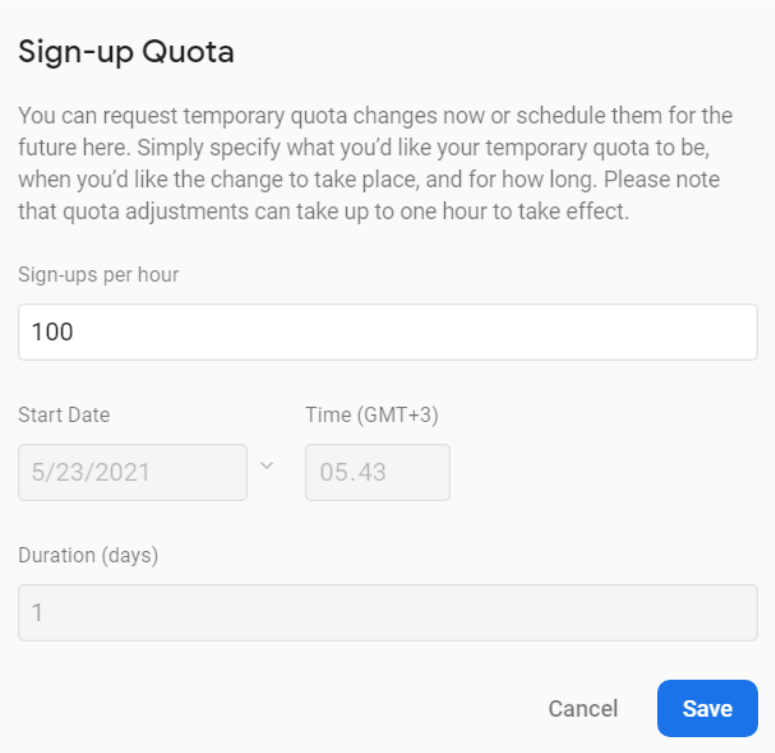
Sisäänkirjautumispalvelut ovat sähköpostia ja salasanaa lukuun ottamatta ensisijaisesti pois käytöstä. Niitä ei kannata ottaa turhaan käyttöön, jos niille ei tee varsinaiseen sovellukseen mitään kirjautumistapaa. Firebase tarjoaa ohjeita sisäänkirjautumispalveluiden käyttöönotolle liitteen 1 kolmannessa web-osoitteessa.

Sallitut verkkotunnukset (Authorized domains) määrittelevät, minkä verkkotunnuksien kautta voi autentikoida käyttäjiä. Nämä täytyy syöttää silloin, kun käyttää Googlea, puhelinta tai kolmansien osapuolien palveluita. Verkkotunnuksissa on kaksi Firebase-kehitysalustan verkkotunnusta valmiiksi.

Yhden tilin raja sähköpostille (One account per email address) tarkoittaa sitä, ettei käyttäjä voi luoda useita käyttäjätilejä käyttäen eri sisäänkirjautumispalveluita. Tämä rajoitus on lähtökohtaisesti päällä, eikä sitä kannata muuttaa ilman hyvää syytä. Rajoituksen poistamisen vaaroista kerrotaan Firebase-kehitysalustan sivuilla liitteen 1 neljännessä web-osoitteessa.

Tilien luonnin hallinta (Manage sign up quota) tarkoittaa uusien sähköpostia ja salasanaa tai vierastunnusta käyttävien käyttäjätilien luonnin rajoittamista samasta IP-osoitteesta. Rajoittaminen asetetaan tuntipohjaisesti ja tilapäisesti,

kuvassa 12 näkyvästä ikkunasta. Oletusrajoitus on 100 uutta käyttäjätiliä samasta IP-osoitteesta tunnissa. Esimerkki rajoituksen nostamisesta voisi olla tietokannan käyttöönotto jossain suuressa organisaatiossa tai tapahtumassa, jossa uusia käyttäjiä tulee ennakoidusti samasta IP-osoitteesta.

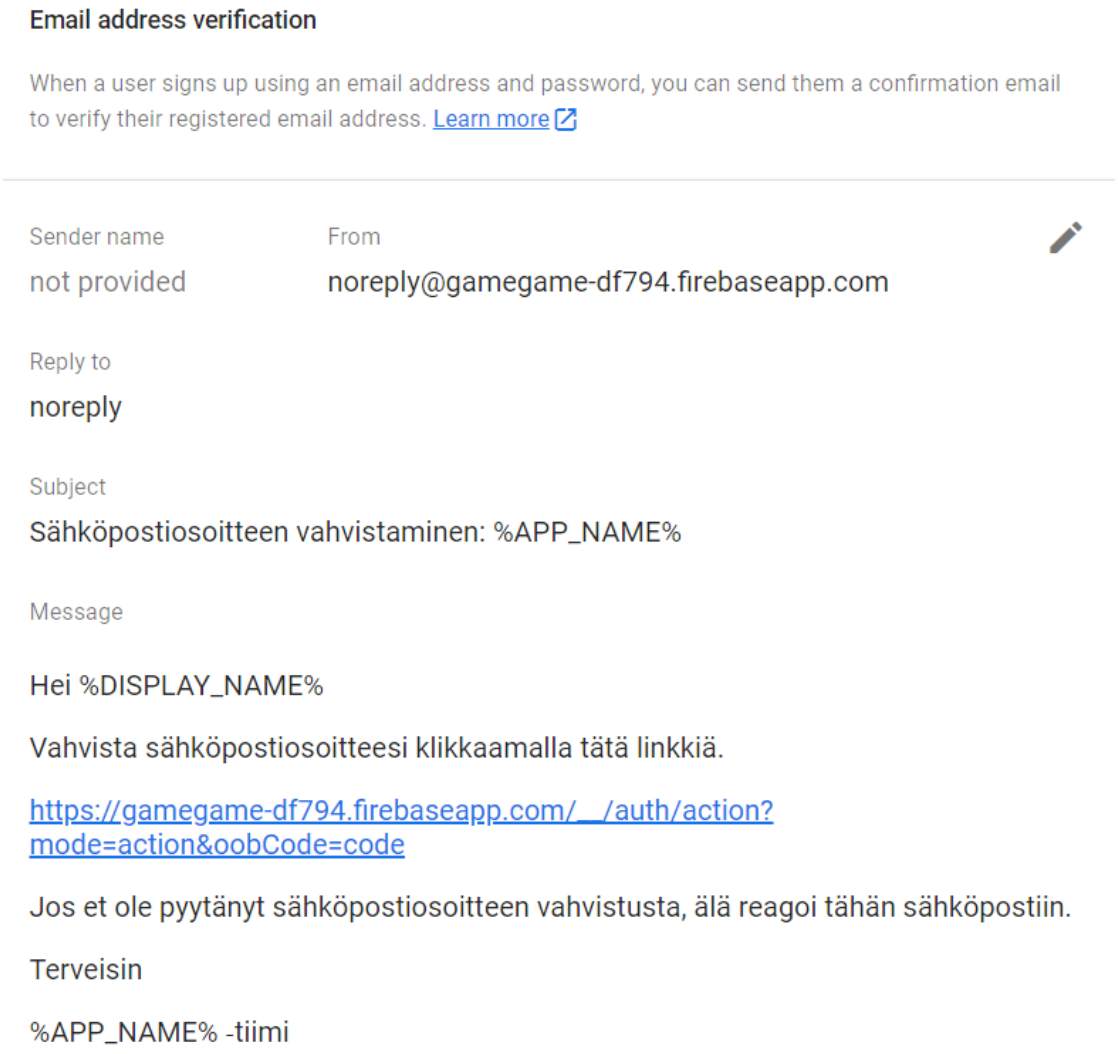


Kuva 12. Tilien luonnin hallinta

Viestintäpohjat

Viestintäpohjissa on sähköposti- ja tekstiviestipohjia, joiden kautta Firebase ottaa yhteyttä asiakaslaitteisiin. Sähköpostipohjiin kuuluu oletuksena salasanan resetointi, sähköpostin vahvistaminen ja sähköpostin vaihto. Tekstiviestipohjista löytyy pohja sisäänkirjautumiskoodin lähettämistä varten.

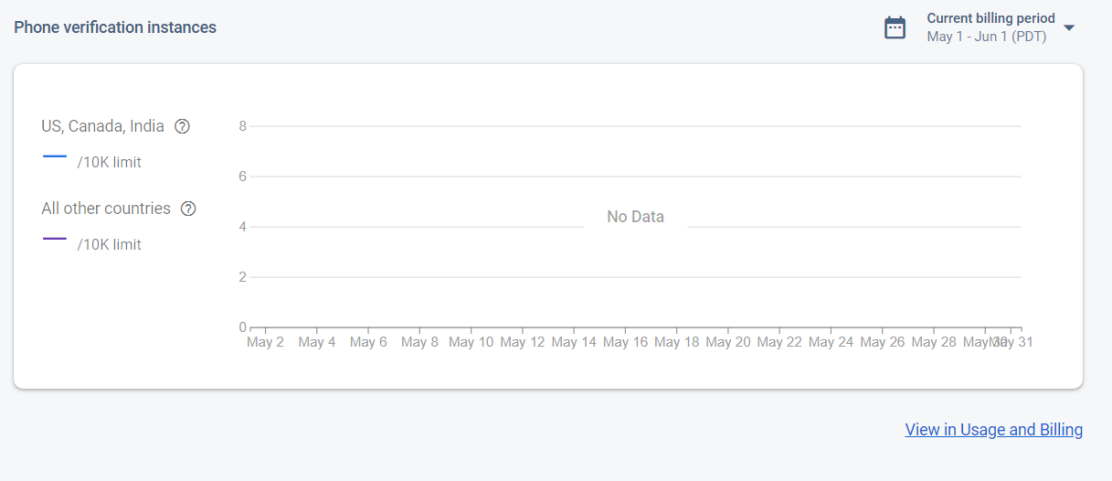
Firebase tarjoaa sisäänrakennetun sähköpostipalvelun, mutta viestintäpohjien välilehdellä voi määritellä sen tilalle oman sähköpostipalvelun. Viestintäpohjien muokkaaminen on rajattua, niistä voi vaihtaa lähettäjän ja korvata Firebase-kehitysalustan palveluihin johtavan linkin. Varsinaisen viestin pystyy muuttamaan ainoastaan salasanan resetoinnissa. Sähköpostipohjat koskevat ainoastaan Firebase-kehitysalustan omaa sähköpostia ja salasanaa, eikä muita käyttöönotettuja sisäänkirjautumispalveluita. Kuvassa 13 näkyy oletusarvoinen pohja sähköpostin varmistusta varten suomen kielellä.



Kuva 13. Sähköpostin varmistaminen suomen kielellä

Käyttö

Käytön välilehdessä ei varsinaisesti voi konfiguroida mitään, mutta sieltä voi nähdä puhelimella tehtyjä sisäänkirjautumisia eroteltuna Yhdysvalloista, Kanadasta, Intiasta tai kaikista muista maista laskutusjaksoittain. Firebase-kehitysalustan ilmainen versio tarjoaa 10 000 sisäänkirjautumista yhdessä laskutusjaksossa. Kuvassa 14 näytetään puhelimella tehtyjen sisäänkirjautumisten määrä yhden laskutusjakson aikana.



Kuva 14. Puhelimella tehdyt sisäänkirjautumiset laskutusjaksoittain

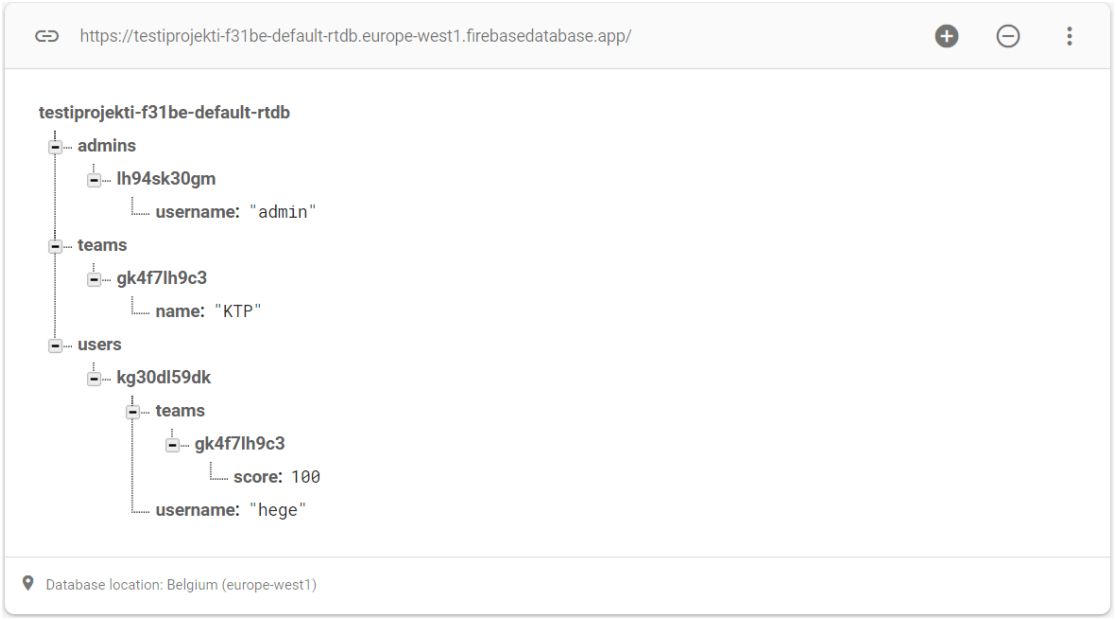
Tätä välilehteä ei tarvitse seurata, jos ei ole mahdollistanut puhelimella sisäänkirjautumista sovelluksessa. Puhelimen käytön hinnaston Firebase-kehitysalustan maksullisessa versiossa voi nähdä muiden palveluiden ohella liitteen 1 viidennessä web-osoitteessa.

6.2.2 Reaaliaikainen tietokanta

Reaaliaikaisen tietokannan konfiguroinnit koskevat tietokannan tietoja, turvallisuussääntöjä ja varmuuskopioita. Reaaliaikaisen tietokannan konfigurointivalikosta pääsee myös näkemään tietokannan käyttöastetta. Turvallisuussäännöistä kerrotaan erikseen niille omistetussa luvussa.

Tiedot

Tietojen välilehdellä pääsee näkemään reaaliaikaisen tietokannan ja muokkaamaan sitä. Muutokset tietokantaan tulevat lähinnä kuitenkin tietokannan ja asiakaslaitteiden tai web-sivujen välillä tapahtuvasta vuorovaikutuksesta, tarve manuaalisille muutoksille tämän välilehden kautta pitäisi lopulta eliminoida kyseen ollessa melkein mikä tahansa sovellus. Kuvassa 15 näytetään yksinkertaistettu demoversio Game Game -projektin tietokannasta, jossa käyttäjä ”hege” on kerännyt 100 pistettä KTP:n minipeleistä.



Kuva 15. Yksinkertaistettu demoversio Game Game -projektin tietokannasta

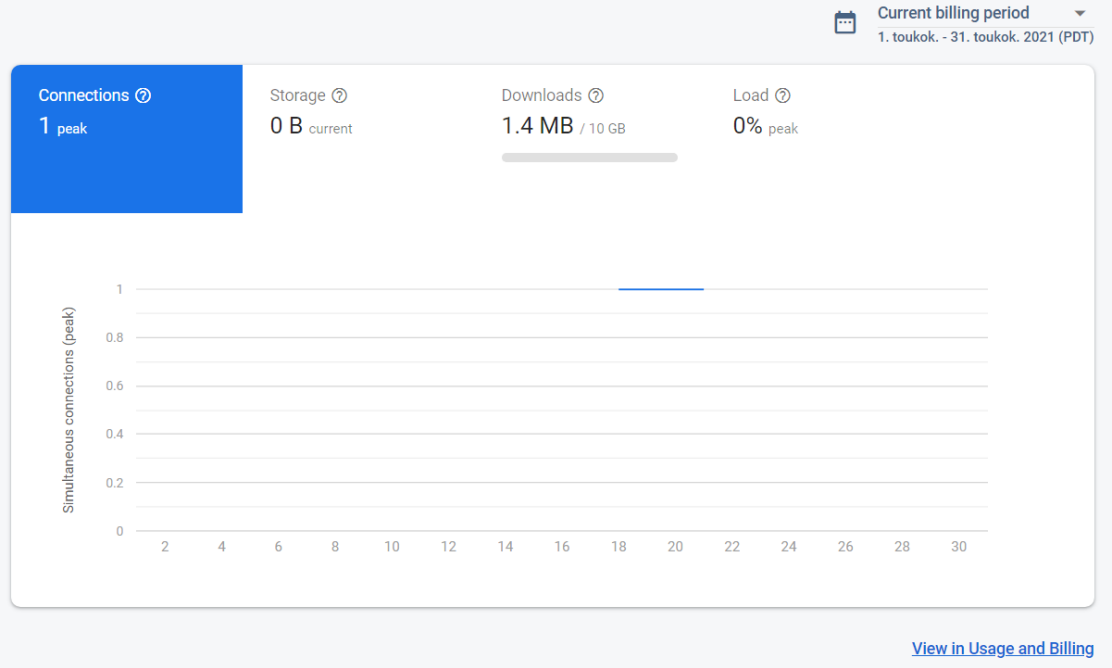
Tältä välilehdeltä löytyy oikeasta yläkulmasta kuvassa 15 näkyvä valikko, jonka painike muistuttaa kolmea pistettä. Tämän valikon kautta voi viedä tai tuoda tietokannan JSON-muodossa tai lisätä uuden tietokannan. Uuden tietokannan lisääminen vaatii Firebase-kehitysalustan maksullisen version, koska ilmainen versio sallii vain yhden tietokannan ylläpitämisen sovellusta kohti.

Varmuuskopiot

Varmuuskopioiden välilehdessä voi hallita Firebase-kehitysalustan maksullisen version tarjoamaa varmuuskopiointipalvelua. Varmuuskopiointipalvelu ottaa päivittäisiä varmuuskopioita, joihin tietokannan voi halutessaan palauttaa. Maksullinen varmuuskopiointi vain helpottaa varmuuskopiointiprosessia, mikään ei estä käyttäjää esimerkiksi ottamasta omia varmuuskopioita tietojen välilehdeltä JSON-muodossa tai luomaan omaa prosessia tätä varten. Lisätietoa maksullisesta varmuuskopiointipalvelusta löytyy liitteen 1 kuudennesta web-osoitteesta.

Käyttö

Käytön välilehdellä voi tarkastella tietokannan käyttöastetta. Se kertoo yhteyksien määrän, tiedon säilytyskoon, latausten koon sekä rasitusasteen prosentteina. Näiden tietojen esittämistapa näkyy kuvassa 16.



Kuva 16. Tietokannan käyttö

Firebase-kehitysalustan ilmainen versio takaa 100 samanaikaista yhteyttä, 1 gigatavun säilytettyä tietoa ja 10 gigatavua ladattua tietoa. Tätä välilehteä kannattaa seurata, ettei tietokannan ilmaisten tai maksullisten käyttörajoitusten ohittuminen tule yllätyksenä. Lisätietoa Firebase-kehitysalustan reaaliaikaisen tietokannan hinnoittelusta ja rajoitteista löytyy liitteen 1 viidennestä web-osoitteesta.

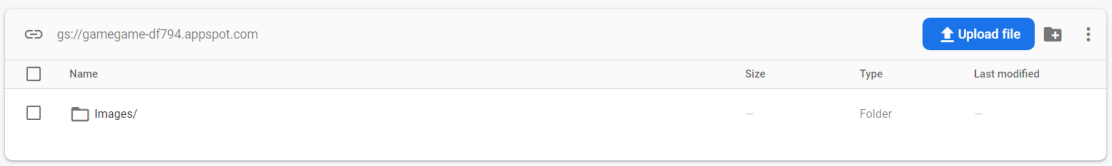
Firebase-kehitysalustan Firestore-tietokanta toimii jokseenkin samankaltaisesti, mutta se on ominaisuuksiltaan rikkaampi ja monimutkaisempi. Game Game -projektissa ei ole Firestore käytössä, joten sen konfigurointiin ei kiinnitetä sen koommin huomiota. Firestore-tietokannan konfiguroinnista löytyy kuitenkin tietoa liitteen 1 seitsemännestä web-osoitteesta.

6.2.3 Tietovarasto

Tietovaraston konfiguroinnit koskevat tietovaraston tiedostoja ja turvallisuussääntöjä. Tietovaraston konfigurointivalikosta pääsee myös tarkastelemaan tietovaraston käyttöä. Turvallisuussäännöistä kerrotaan erikseen niille omistetussa luvussa.

Tiedostot

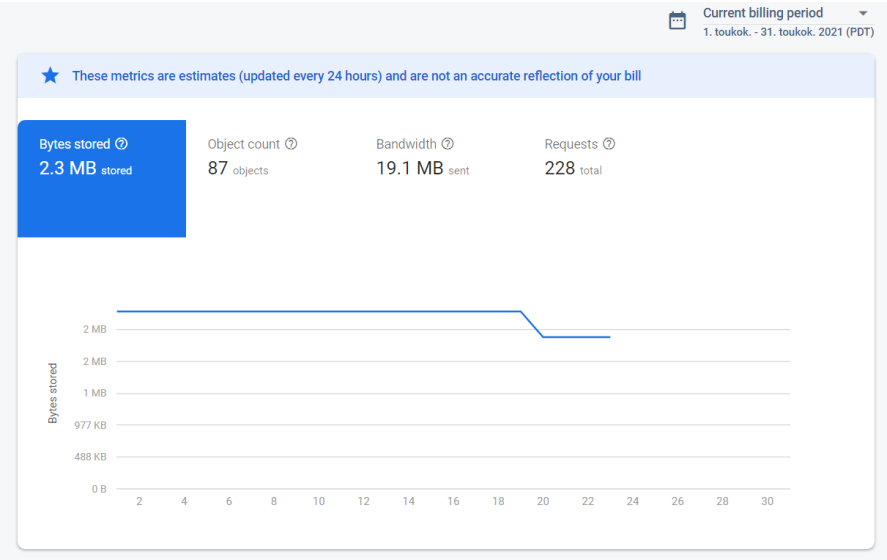
Tiedostojen välilehdeltä pääsee tarkastelemaan tiedostoja sekä lisäämään tai poistamaan niitä. Tiedostoja säilytetään niin kutsutuissa ämpäreissä ja näytetään välilehdellä ämpäreittäin. Tältä välilehdeltä löytyy oikeasta yläkulmasta kuvassa 17 näkyvä valikko, jonka painike muistuttaa kolmea pistettä. Tästä valikosta pystyy lisäämään tai poistamaan ämpäreitä Firebase-kehitysalustan maksullisella versiolla. Firebase-kehitysalustan ilmaisella versiolla sovelluksella voi vain olla yksi ämpäri, jota ei voi poistaa.



Kuva 17. Tiedostojen välilehti

Käyttö

Käytön välilehdellä pääsee tarkastelemaan tietovaraston käyttöä. Se kertoo tallennetun tiedon koon, tallennettujen objektien määrän, asiakaslaitteiden käyttämän verkkokaistan ja asiakaslaitteilta tulleiden pyyntöjen määrän kuvassa 18 näytetyllä tavalla.



Kuva 18. Tietovaraston käyttö

Firebase-kehitysalustan ilmainen versio takaa 5 gigatavua tallennustilaa, 1 gigatavun latauksia päivässä, 20 000 talletusoperaatiota päivässä ja 50 000 latausoperaatiota päivässä. Kuten muissakin tapauksissa, Tietovaraston hinnoittelusta löytyy lisätietoa liite 1 5. web-osoitteesta.

6.3 Turvallisuussäännöt

Turvallisuussäännöillä määritellään käyttäjien pääsyä ja toimintaa Firebase-kehitysalustan tietokannoissa sekä tietovarastossa. Tässä luvussa käydään läpi reaaliaikainen tietokanta sekä tietovarasto, keskittyen Game Game -projektin varsinaisten turvallisuussääntöjen koodien sijaan niiden konseptuaaliseen puoleen esimerkkien avulla tietoturvallisista syistä. Firestore noudattaa pitkälti samoja periaatteita, mutta sitä ei käydä tässä luvussa erikseen läpi koska se ei ole Game Game -projektin käytössä. Firebase-kehitysalustan turvallisuussäännöt on virallisesti dokumentoitu liitteen 1 kahdeksannessa web-osoitteessa.

6.3.1 Reaaliaikainen tietokanta

Firebase-kehitysalustan reaaliaikaisen tietokannan turvallisuussäännöt määrittelevät kenellä on luku- ja kirjoitusoikeus tietokantaan, kuinka tieto on jäsennetty ja mitä indeksejä on olemassa. Nämä turvallisuussäännöt elävät Firebase-kehitysalustan palvelimella ja niitä valvotaan automaattisesti koko ajan. Jokainen kirjoitus- ja lukupyyntö suoritetaan ainoastaan sääntöjen salliessa tämän. Oletusarvoisesti mitään pääsyä ei sallita. Tämä suojaa tietokantaa väärinkäytöltä, kunnes turvallisuussäännöt ja autentikointi on konfiguroitu. (Understand Firebase Realtime Database Rules 2021.)

Reaaliaikaisessa tietokannassa on neljä erilaista sääntötyyppiä, jotka ovat luku, kirjoittaminen, validointi ja indeksointi. Sääntötyypit ja niiden syntaksi ovat kuvattu samassa järjestyksessä kuvassa 19.

Rule Types	
.read	Describes if and when data is allowed to be read by users.
.write	Describes if and when data is allowed to be written.
.validate	Defines what a correctly formatted value will look like, whether it has child attributes, and the data type.
.indexOn	Specifies a child to index to support ordering and querying.

Kuva 19. Reaaliaikaisen tietokannan turvallisuussäännöt (Understand Firebase Realtime Database Rules 2021)

Ohessa kuvaillaan sääntötyypit viitaten kuvaan 19. Luku-sääntö määrittelee miten ja milloin käyttäjät voivat lukea tiedon. Kirjoittamis-sääntö määritellee milloin tiedon päälle voi kirjoittaa. Validointi-sääntö määrittelee miltä oikeaa muotoa noudattava tieto näyttää, onko sillä alihaarautuneita ominaisuuksia ja tiedon tyypin. Indeksointi-sääntö määrittelee mitä alihaarautunutta tietoa käytetään järjestelyssä ja kyselyissä. Kuvassa 20 näytetään esimerkki sääntöjen käytöstä, jossa käyttäjillä ovat luku- muttei kirjoitusoikeudet polkuun ”testi”

```
{
  "rules": {
    "testi": {
      ".read": true,
      ".write": false
    }
  }
}
```

Kuva 20. Turvallisuussääntöjen syntaksi

Viitaten kuvaan 9 ja taulukkoon 1, turvallisuussääntöjä voisi Game Game -projektin kannalta kuvailla seuraavanlaisesti. Luku-säännöllä voi sallia kaikkien käyttäjien lukuoikeuden urheiluseuroihin ja kirjoittamis-säännöllä voi estää niiden päälle kirjoittamisen, ellei kyse ole urheiluseuran tai tietokannan ylläpitäjästä. Samaa periaatetta käytetään käyttäjien kanssa, käyttäjän on tarkoitus pystyä lukemaan tai muuttamaan hänen omia käyttäjätietojaan. Ylläpitäjien tietoihin ei tule päästää ketään muuta kuin ylläpitäjiä.

Validointia tarvitaan pitkälti kaikissa Game Game -projektin tietokannan osa-alueissa. Esimerkiksi käyttäjän pisteet jollain urheiluseuralla eivät voi olla missään muussa kuin numeerisessa muodossa, ja urheiluseurojen tietojen on noudatettava tarkkaa formaattia. Validointia kannattaa tehdä kuitenkin jokai-

sella kerroksella, jossa tietoa liikkuu. Turvallisuussäännöissä määritelty validointi on hyvä varotoimenpide, mutta se ei esimerkiksi voi antaa asiakaslaitteelle palautetta vääränlaisen tiedon syöttämisestä front-end päässä.

Indeksointi on Game Game -projektin kannalta harvinaisempi sääntö, koska sen tiedon käsittely ei ole kovin monimutkaista eikä siinä tehtyjen kyselyiden määrä vastaa suurta volyymiä. Tämän säännön käyttöä tullaan tutkimaan Game Game -projektin myöhäisemmässä kehitysvaiheessa.

6.3.2 Tietovarasto

Tietovarasto mahdollistaa käyttöoikeuksien sääntöjen määrittelyn tiedosto- ja polkukohtaisesti. Firebase-kehitysalustan palvelimet käyttävät näitä määrityksiä valvoakseen, kenellä on pääsy sovelluksen tiedostoihin. (Understand Firebase Security Rules for Cloud Storage 2021.)

Kuvan 21 esimerkissä on Firebase-kehitysalustan oletussäännöt tietovarastolle, jotka sallivat tiedostojen lukemisen ja kirjoittamisen ainoastaan autentikoiduille käyttäjille.

```
rules_version = '2';
service firebase.storage {
  match /b/{bucket}/o {
    match /{allPaths=**} {
      allow read, write: if request.auth != null;
    }
  }
}
```

Kuva 21. Tietovaraston oletusarvoiset turvallisuussäännöt

Game Game -projektin tietovarastossa säilytetään toistaiseksi pelkästään urheiluseuroihin liittyviä kuvia, kuten logoja ja pelaajien kuvia. Luku- ja kirjoittamis-sääntöjä hyödyntäen käyttäjät voivat tarkastella kaikkia kuvia ja ainoastaan tietyn urheiluseuran tai tietokannan ylläpitäjä voi kirjoittaa niiden päälle. Oletussääntöihin poiketen, vieraskäyttäjillä on pääsy sovelluksessa käytettäviin kuviin.

Turvallisuussääntöjä voi myös käyttää tiedon validoimiseen. Turvallisuussäännöillä voi muun muassa määritellä minkä tyyppisiä ja kokoisia tiedostoja tietovarastoon sallitaan. Kuvan 22 esimerkissä näytetään, miten vain alle 5 megatavun suuruiset kuvatiedostot sallitaan.

```
service firebase.storage {
  match /b/{bucket}/o {
    match /images/{imageId} {
      // Only allow uploads of any image file that's less than 5MB
      allow write: if request.resource.size < 5 * 1024 * 1024
                   && request.resource.contentType.matches('image/.*');
    }
  }
}
```

Kuva 22. Esimerkki alle 5 megatavun kuvien sallimisesta (Understand Firebase Security Rules for Cloud Storage)

Tiedon validointi on Game Game -projektin kannalta tärkeää, koska sen sovellus käyttää tietyn kokoisia ja tyyppisiä kuvatiedostoja tietyissä paikoissa. Tämä johtuu siitä, että Unity-pelimoottori ei pysty tulkitsemaan ja täten välittämään kaikkia mahdollisia kuvatiedostoja, joita tietovarasto pystyy säilyttämään.

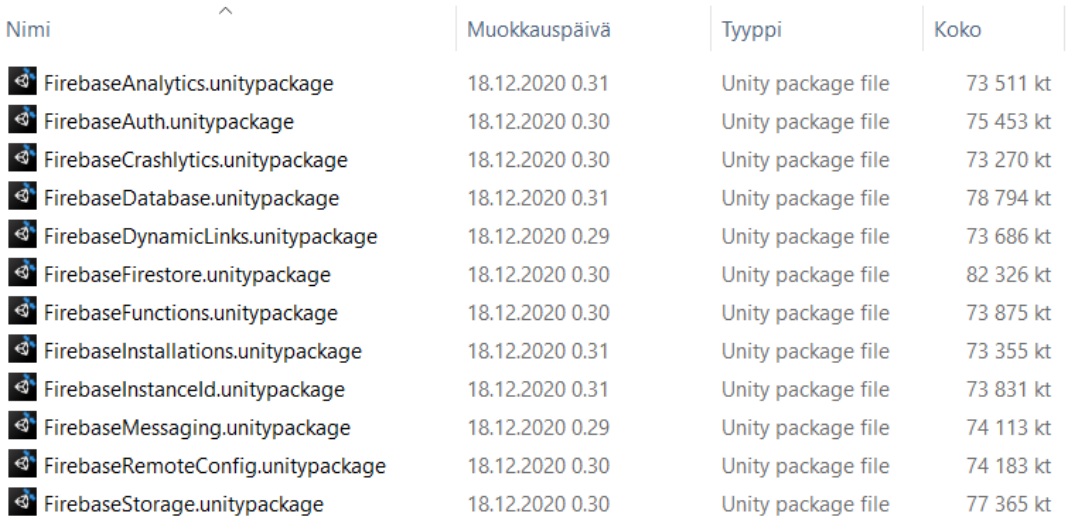
6.4 Yhteyden muodostaminen ja tietojen käsittely asiakaslaitteella

Yhteyden muodostaminen ja tietojen käsittely tarkoittaa tässä asiayhteydessä tietokannan toimivuutta Unity-kehitysalustan ja siinä kehityksessä olevan sovelluksen kanssa. Saadakseen yhteyden tietokantaan, Firebase täytyy asentaa ja konfiguroida Unity-kehitysalustaan luvun 5.1 ohjeiden mukaisesti.

Olennaisia yhteystietoja Firebasesta ja sen sovellukseen liittyvästä tietokannasta löytyy JSON-muodossa olevasta google-services-tiedostosta. Sen voi ladata konfiguroidessa Firebase-kehitysalustaa Unity-kehitysalustan käyttöä varten, tai myöhemmin menemällä Firebase-kehitysalustan kehityskonsolista projektin asetuksiin rattaan muotoisesta painikkeesta. Toimivan yhteyden kannalta on tärkeää, että tietokannan yhteystiedot ovat oikein, joten muutokset Firebasessa saattavat vaatia tämän tiedoston lataamisen uudelleen. Firebase generoi tiedoston sisällön automaattisesti Firebase-kehitysalustan konfigurointien mukaisesti.

Google-services-tiedosto laitetaan Unity-kehitysalustan peliprojektin ”Assets”-kansioon mihin tahansa haluamaan alipolkuun, esimerkiksi Unity-kehitysalustan pelieditorin avulla. Tiedostosta löytyy monenlaisia yksilöllisiä Firebase-kehitysalustan tunnuksia ja verkko-osoitteita, joten sen turvallisuusluokitus voi olla jo muutenkin luottamuksellisia pelitiedostoja korkeampi.

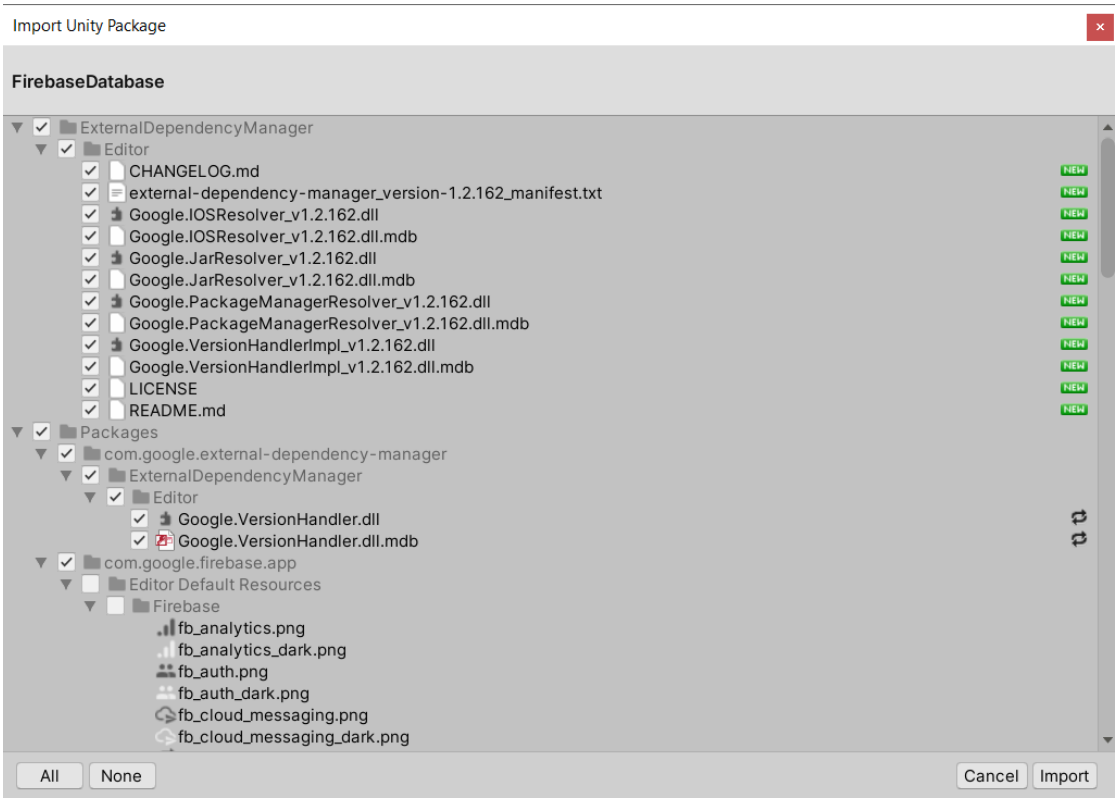
Unity-kehitysalustan ja Firebase-kehitysalustan toimintaan tarvitaan Unity-kehitysalustan puolelle vielä Firebase-kehitysalustan ohjelmistokehityspaketti. Kuvassa 23 näkyy Firebase-kehitysalustan version 7.0.2 ohjelmistokehityspaketti Unity-kehitysalustaa varten, joka on ladattu Firebase-kehitysalustan websivulta.



Nimi	Muokkauspäivä	Tyyppi	Koko
FirebaseAnalytics.unitypackage	18.12.2020 0.31	Unity package file	73 511 kt
FirebaseAuth.unitypackage	18.12.2020 0.30	Unity package file	75 453 kt
FirebaseCrashlytics.unitypackage	18.12.2020 0.30	Unity package file	73 270 kt
FirebaseDatabase.unitypackage	18.12.2020 0.31	Unity package file	78 794 kt
FirebaseDynamicLinks.unitypackage	18.12.2020 0.29	Unity package file	73 686 kt
FirebaseFirestore.unitypackage	18.12.2020 0.30	Unity package file	82 326 kt
FirebaseFunctions.unitypackage	18.12.2020 0.30	Unity package file	73 875 kt
FirebaseInstallations.unitypackage	18.12.2020 0.31	Unity package file	73 355 kt
FirebaseInstanceId.unitypackage	18.12.2020 0.31	Unity package file	73 831 kt
FirebaseMessaging.unitypackage	18.12.2020 0.29	Unity package file	74 113 kt
FirebaseRemoteConfig.unitypackage	18.12.2020 0.30	Unity package file	74 183 kt
FirebaseStorage.unitypackage	18.12.2020 0.30	Unity package file	77 365 kt

Kuva 23. Firebase-kehitysalustan ohjelmistokehityspaketti Unity-kehitysalustalle

Game Game -projekti käyttää Firebase-kehitysalustan autentikointia, tietokantaa ja tietovarastoa. Halutun Unity-paketin voi asentaa Unity-kehitysalustaan avaamalla valikon ”Assets”, valitsemalla ”Import Package” ja ”Custom Package” valiten tiedostoista haluamansa Unity-paketin. Kuvassa 24 näytetään esimerkkinä Unity-kehitysalustan peliruudulle avautuva ikkuna Database-paketin asennusta varten.



Kuva 24. Firebase-kehitysalustan Database-paketin asennus Unity-kehitysalustaan

Tarvittavat asetukset, joita näkee kuvassa 24, ovat Firebase-kehitysalustan paketeissa oletuksena oikein, eikä niihin tarvitse puuttua. Kun ensimmäisen Firebase-paketin asentaa, sen mukana tulee myös Firebase-kehitysalustan olennaisimmat komponentit Unity-kehitysalustaa varten.

Tämän jälkeen kannattaa vielä tarkistaa Unity-kehitysalustan ja Firebase-kehitysalustan välillä olevien riippuvuuksien toiminta. Firebase tarjoaa ratkaisun liitteen 1 yhdeksännessä web-osoitteessa. Kuvassa 25 tehtyjen muokkausten pohjalta skripti ilmoittaa Unity-kehitysalustan ja Firebase-kehitysalustan välien riippuvuuden toiminnan.

```
using UnityEngine;


public class FirebaseDepencies : MonoBehaviour
{
    // Start is called before the first frame update
    void Start()
    {
        Firebase.FirebaseApp.CheckAndFixDependenciesAsync().ContinueWith(task => {
            var dependencyStatus = task.Result;
            if (dependencyStatus == Firebase.DependencyStatus.Available)
            {
                Debug.Log("Firebase dependencies successfully resolved");
            }
            else
            {
                UnityEngine.Debug.LogError(System.String.Format(
                  "Could not resolve all Firebase dependencies: {0}", dependencyStatus));
                // Firebase Unity SDK is not safe to use here.
            }
        });
    }
}

```

Kuva 25. Skripti, jonka avulla voi tarkastaa Firebase-kehitysalustan ja Unity-kehitysalustan riippuvuuksien toiminnan

Mikäli skripti palauttaa vastauksen ”Firebase dependencies successfully resolved” käynnistäessä sovelluksen Unity-kehitysalustan pelieditorin puolella, Unity-kehitysalustan ja Firebase-kehitysalustan riippuvuudet ovat kunnossa.

6.4.1 Tärkeimmät luokat ja metodit

Kun tietokantaan on onnistuneesti luotu yhteys, voi alkaa käyttämään Firebase-kehitysalustan tietokannan luokkia ja metodeja Unity-kehitysalustan käyttämällä C#-ohjelmointikielellä.

Kaikki C#-ohjelmointikielessä liittyy luokkiin ja objekteihin sekä niiden attribuutteihin ja metodeihin. Oikean maailman esimerkissä auto on objekti. Autolla on attribuutteja kuten paino ja väri, sekä metodeja kuten ajaminen ja jarruttaminen. Luokka on eräänlainen objektien rakentaja tai ”suunnitelma” objektien rakentamiselle. (W3Schools s.a.)

Firebase-kehitysalustalta löytyy Unity-kehitysalustalle monia tärkeitä alati päivittyviä luokkia ja metodeja, mutta tärkeimpiä Game Game -projektille ovat DatabaseReference, Datasnapshot, GetValueAsync ja SetValueAsync.

DatabaseReference

Firebase-viittaus edustaa tiettyä polkua tietokannassa ja sitä voidaan käyttää lukemaan tai kirjoittamaan tietoa kyseiseen tietokannan polkuun. Tämä luokka on suora aloituspiste kaikille tietokantaoperaatioille. Sen jälkeen, kun sen on alustanut URL-osoitteella, sitä voidaan käyttää lukemaan tietoa, kirjoittamaan tietoa ja luomaan uusia tietokantaviittauksia. (DatabaseReference 2021.)

DatabaseReference-luokkaa käytettäessä kannattaa ottaa huomioon, että viittaus hakee kaikki polun alta löytyvät tiedot. Viittaus tarpeettoman korkeaan polkuun tietokannan hierarkiassa saattaa siis altistaa alta löytyvää luottamuksellista tietoa tai viedä turhaa muistia ilman hyvää syytä.

DatabaseReference-luokan ollessa olennainen Firebase-kehitysalustan toiminnalle, sitä käytetään Game Game -projektissa hakemaan tietoa kaikilta tietokannan osa-alueilta.

DataSnapshot

DataSnapshot-instanssi on eräänlainen kopio tietokannan tiedoista, joita on haettu esimerkiksi DatabaseReference-luokan avulla. DataSnapshot-instanssia käytetään tietokannasta haetun tiedon säilyttämiseen ja käyttämiseen. Sovelluksessa kannattaa käyttää DataSnapshot-instansseja aina kun mahdollista välttääkseen turhaa verkkoliikennettä tietokannan kanssa. DataSnapshot instansseja ei kannata ottaa tiedoista, joiden täytyy olla aina reaaliaikaisia.

GetValueAsync ja SetValueAsync

GetValueAsync on DatabaseReference-luokan metodi, jota käytetään hakemaan tietoa tietokannasta. GetValueAsync-metodi hakee tietoa määritellystä tietokannan polusta, joten sitä käytettäessä kannattaa varmistaa tämän polun luku-sääntöjen olevan konfiguroitu oikein reaaliaikaisen tietokannan turvallisuussäännöissä.

SetValueAsync on DatabaseReference-luokan metodi, jota kirjoittamaan tietoa tietokantaan. SetValueAsync-metodi kirjoittaa tietoa määriteltyyn tietokannan polkuun, joten sitä käytettäessä kannattaa varmistaa tämän polun kirjoittamis-sääntöjen olevan konfiguroitu oikein reaaliaikaisen tietokannan turvallisuussäännöissä.

Unityssä on käytettävissä satoja Firebase-kehitysalustan tarjoamia luokkia sen eri ominaisuuksiin, joita tämä luku ei voi kattaa. Ottaessa Unity-sovellukseen käyttöön uusia Firebase-kehitysalustan luokkia ja niiden metodeja, kannattaa tutustua niiden dokumentaatioon liitteen 1 kymmenennessä web-osoitteessa.

7 YHTEENVETO

Minulla ei ollut entuudestaan paljoa kokemusta tietokannoista, niiden turvallisuudesta tai suunnittelusta. Tietokantojen maailma on erittäin laaja, joten sopivan tiedon löytäminen ja opiskelu oli työlästä ja aikaa vievää. Tämän takia oli myös vaikeaa rajata aihepiiri ja kiteyttää tieto mahdollisimman tehokkaaseen muotoon. Opin teoreettista osuutta tutkiessa tietokannoista paljon, ja tajusin että minulla on vielä paljon enemmän opittavaa.

Game Game -projekti on Xamkin ammattikorkeakoulussa aloitettu hanke, joka on enemmän tai vähemmän toiminut harjoitusprojektina opiskelijoille. Projektissa on tullutkin siksi vastaan paljon haasteita, koska monien ongelmien ratkaisuun ei ole löytynyt aiempaa kokemusta. Tämä projekti on kuitenkin kehittänyt kykyäni tietoturvallisten ongelmien havaitsemiseen ja välttämiseen huomattavasti. Yksi suurimmista oivalluksista projektin myötä on ollut se, että tietokantojen suunnittelu tulisi aloittaa turvallisuusnäkökulmasta. Turvallisuus ei ole asia, jonka voi lisätä rakenteellisesti huonosti suunniteltuun tietokantaan myöhemmin. Huomasin myös, että olin itse harjoittanut monia turvallisuutta vaarantavia käytäntöjä aiemmin.

En onnistunut tutkimaan tai kattamaan kaikkia osa-alueita niin hyvin kuin olisin toivonut. Tästä yksi vahva esimerkki on Firebase-kehitysalustan Firestore, joka olisi voinut olla hyödyllistä tutkia paremmin Game Game -projektia varten. Firestore on ominaisuuksiltaan rikkaampi kuin reaaliaikainen tietokanta, ja

näitä kahta voi käyttää yhdessäkin. Game Game -projektin jatkokehityksen kannalta voisi siis olla kannattavaa pohtia Firestore-tietokannan mahdollisia hyötyjä.

Game Game -projektin varsinaisen tietokannan kehitys ei tullut välttämättä tarpeeksi hyvin huomioiduksi opinnäytetyön aikana, mutta tämä opinnäytetyö antoi minulle paremmat eväät sen jatkokehitykseen. Näen, että opinnäytetyö vastasi hyvin toimeksiannon tarkoitusta.

LÄHTEET

Anderson, B. & Nicholson, B. 2020. SQL vs. NoSQL Databases: What's the Difference? Blogi. Päivitetty 18.6.2020. Saatavissa: https://www.ibm.com/cloud/blog/sql-vs-nosql [viitattu 15.4.2021].

Cloudflare s.a. What is the OSI Model? WWW-dokumentti. Saatavissa: https://www.cloudflare.com/learning/ddos/glossary/open-systems-interconnection-model-osi/ [viitattu 18.5.2021].

Concepta Tech. 2017. What Is The Difference Between Front-End And Back-End Development? WWW-dokumentti. Päivitetty 27.2.2019. Saatavissa: https://www.conceptatech.com/blog/difference-front-end-back-end-development [viitattu 24.4.2021].

Coronel, C., Morris, S., Crockett, K & Blewett, C. 2020. Database Principles: Fundamentals of Design, Implementation and Management. 3. painos. United Kingdom: Cengage Learning EMEA.

DataSnapshot s.a. Google. WWW-dokumentti. Päivitetty 11.5.2021. Saatavissa: https://firebase.google.com/docs/reference/android/com/google/firebase/database/DataSnapshot [viitattu 20.5.2021].

DatabaseReference s.a. Google. WWW-dokumentti. Päivitetty 11.5.2021. Saatavissa: https://firebase.google.com/docs/reference/android/com/google/firebase/database/DatabaseReference [viitattu 22.5.2021].

Euroopan Unioni s.a. Do we always have to delete personal data if a person asks? WWW-dokumentti. Saatavissa: https://ec.europa.eu/info/law/law-topic/data-protection/reform/rules-business-and-organisations/dealing-citizens/do-we-always-have-delete-personal-data-if-person-asks_en [viitattu 20.5.2021].

Firebase Authentication s.a. Google. WWW-dokumentti. Päivitetty 11.5.2021. Saatavissa: https://firebase.google.com/docs/auth [viitattu 17.5.2021].

Firebase Realtime Database s.a. Google. WWW-dokumentti. Päivitetty 11.5.2021. Saatavissa: https://firebase.google.com/docs/database [viitattu 26.4.2021].

Firebase Security Rules s.a. Google. WWW-dokumentti. Päivitetty 19.5.2021. Saatavissa: https://firebase.google.com/docs/rules [viitattu 18.5.2021].

Structure Your Database s.a. Google. WWW-dokumentti. Päivitetty 20.5.2021. Saatavissa: https://firebase.google.com/docs/database/web/structure-data [viitattu 10.5.2021].

IBM Cloud Education. 2019. Relational Databases. WWW-dokumentti. Saatavissa: https://www.ibm.com/cloud/learn/relational-databases [viitattu 17.4.2021].

IBM s.a. Primary and foreign keys. WWW-dokumentti. Saatavissa: https://www.ibm.com/docs/en/ida/9.1.1?topic=entities-primary-foreign-keys [viitattu 17.4.2021].

JSON.org s.a. Introducing JSON. WWW-dokumentti. Saatavissa: https://www.json.org/json-en.html [viitattu 18.5.2021].

Kumar, S. 2018. Mastering Firebase for Android Development: Build Real-Time, Scalable, and Cloud-enabled Android Apps with Firebase. E-kirja. United Kingdom: Packt Publishing, Limited. Saatavissa: https://kaakkuri.finna.fi/ [viitattu 10.5.2021].

OAuth s.a. Access Tokens. WWW-dokumentti. Saatavissa: https://www.oauth.com/oauth2-servers/access-tokens/ [viitattu 14.5.2021].

Oracle. 2020. What Is a database? WWW-dokumentti. Saatavissa: https://www.oracle.com/database/what-is-database/ [viitattu 12.4.2021].

Schaefer, L s.a. NoSQL vs SQL Databases. WWW-dokumentti. Saatavissa: https://www.mongodb.com/nosql-explained/nosql-vs-sql [Viitattu: 15.4.2021].

Sinicki, A. 2021. What is Unity? Everything you need to know. WWW-dokumentti. Saatavissa: https://www.androidauthority.com/what-is-unity-1131558/ [viitattu 18.5.2021].

Sriparasa, S. & D'mello, B. 2018. JavaScript and JSON Essentials: Build Light Weight, Scalable, and Faster Web Applications with the Power of JSON, 2nd Edition. E-kirja. United Kingdom: Packt Publishing, Limited. Saatavissa: https://kaakkuri.finna.fi/ [viitattu 10.5.2021].

Stallings, W. & Brown, L. 2018. Computer Security: Principles and Practice. 4. painos. United Kingdom: Pearson Education Limited.

Tanna, M. & Singh, H. 2018. Serverless Web Applications with React and Fire-base: Develop real-time applications for web and mobile platforms. E-kirja. United Kingdom: Packt Publishing, Limited. Saatavissa: https://kaakkuri.finna.fi/ [viitattu 19.4.2021].

Tech Terms s.a. Authentication. WWW-dokumentti. Päivitetty 13.7.2018. Saatavissa: https://techterms.com/definition/authentication [viitattu 19.5.2021].

Technopedia s.a. Database Management System (DBMS). WWW-dokumentti. Saatavissa: https://www.techopedia.com/definition/24361/database-management-systems-dbms [viitattu 24.5.2021].

Understand Firebase Realtime Database Rules s.a. Google. WWW-dokumentti. Päivitetty 11.5.2021. Saatavissa: https://firebase.google.com/docs/database/security [viitattu 20.5.2021].

Understand Firebase Security Rules for Cloud Storage s.a. Google. WWW-dokumentti. Päivitetty 11.5.2021. Saatavissa: https://firebase.google.com/docs/storage/security [viitattu 20.5.2021].

Unity. 2021. Supported platforms. WWW-dokumentti. Saatavissa: https://docs.unity3d.com/Manual/UnityCloudBuildSupportedPlatforms.html [viitattu 24.5.2021].

Unity s.a. Coding in C# in Unity for beginners. WWW-dokumentti. Saatavissa: https://unity3d.com/learning-c-sharp-in-unity-for-beginners [viitattu 19.5.2021].

W3Schools s.a. C# Classes and Objects. WWW-dokumentti. Päivitetty 11.5.2021. Saatavissa: https://www.w3schools.com/cs/cs_classes.asp [viitattu 22.5.2021].

KUVALUETTELO

Liite 1

Web-osoiteluettelo

1. https://firebase.google.com/docs/unity/setup

2. https://firebase.google.com/

3. https://firebase.google.com/docs/auth

4. https://support.google.com/firebase/answer/9134820?hl=en&ref_topic=6386702&authuser=0

5. https://firebase.google.com/pricing

6. https://firebase.google.com/docs/database/backups?authuser=0

7. https://firebase.google.com/docs/firestore/quickstart

8. https://firebase.google.com/docs/rules

9. https://firebase.google.com/docs/unity/setup

10. https://firebase.google.com/docs/reference/unity