



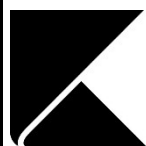
Karelia-ammattikorkeakoulu
Tietojenkäsittely, Tradenomi (AMK)

Firebasen palveluilla toteutettu sijoitusportfolio web-sovellus

Janne Peiponen

Opinnäytetyö, tammikuu 2024

www.karelia.fi



Karelia
AMMATTIKORKEAKOULU

OPINNÄYTETYÖ
Tammikuu 2024
Tietojenkäsittelyn koulutusohjelma

Tikkarinne 9
80200 JOENSUU
+358 13 260 600 (vaihde)

Tekijä(t)
Janne Peiponen

Nimeke
Firebasen palveluilla toteutettu sijoitusportfolio web-sovellus

Toimeksiantaja
-

Tiivistelmä

Opinnäytetyön tavoitteena oli tehdä sijoittajille suunnattu verkkosivusto, jonka avulla sijoittaja voi pitää kirjanpitoa sijoituskohteistaan. Sijoittajalla voi olla useita erilaisia sijoituskohteita, jolloin niistä pidettävä kirjanpito vaikeutuu. Sijoitusportfolio mahdollista sijoitusten tallentamisen koostetusti yhdelle verkkosivustolle.

Sijoitusportfolion tekemisessä käytettiin JavaScript-ohjelmointikieltä, NodeJS-kehitysalustaa, React-käyttöliittymää ja Firebasen palveluita. Firebasen palveluista sijoitusportfoliossa käytettiin dokumenttipohjaista Firestore-tietokantaa, käyttäjien varmentamiseen Authentication-palvelua ja verkkosivusto julkaistiin Hosting-palvelulla. Teoriaosuudessa käydään läpi yleisimpiä tietokantoja, kuten relaatiotietokantaa ja NoSQL-tietokantoja. Teoriaosiossa käsitellään myös Firebase-palveluita, kuten autentikointia ja sen dokumenttipohjaisia tietokantoja.

Opinnäytetyssä saatiin valmiiksi toimiva verkkosivusto, joka täyttää tavoitteet. Sijoitusportfolioon tulee rekisteröidä käyttäjätili, jonka avulla voi tallentaa palvelimelle sijoituskohteita. Sivustoa on tarkoitus tulevaisuudessa jatkokehittää.

Kieli
suomi

Sivuja 41
Liitteet 0
Liitesivumäärä 0

Asiasanat
Firebase, firestore, autentikointi



THESIS
January 2024
Degree Programme in Business Information Technology

Tikkarinne 9
80200 JOENSUU
FINLAND
+ 358 13 260 600

Author (s)
Janne Peiponen

Title
Portfolio Web Application For Investors With Firebase Services

Commissioned by
-

Abstract

The aim of this thesis was to create a portfolio website which allows investors to keep record of investments. Investors can have multiple investments spread around many websites, which makes keeping an account on all of them difficult. This project's main goal is to provide portfolio web application to store all investment data, using Firebase services.

Methods used to create this portfolio web application were JavaScript programming language, NodeJS, React and Firebase services. From Firebase's services, Firestore document-oriented database was used to store investments, Authentication service was used to verify users and the website was published via Firebase Hosting. The theoretical part of the thesis covers most common databases: relational databases and NoSQL databases. The theoretical part also covers Firebase services, like authentication and its document-oriented databases.

The project completed its goals to provide investors a working portfolio website. Investors are verified when logging in and investments can be tracked and saved in the portfolio. The portfolio website is planned to be further developed in the future to provide real time price updates.

Language
Finnish

Pages 41
Appendices 0
Pages of Appendices 0

Keywords
Firebase, firestore, authentication

Sisältö

1	Johdanto	5
2	Yleisimmät tietokantamallit	5
2.1	Suosituimmat tietokannat	5
2.2	Relaatiotietokannat	6
2.3	NoSQL	7
2.4	Dokumenttitietokannat	8
2.5	Avain-arvo tietokannat	9
2.6	SQL- ja NoSQL-tietokantamallien erot.....	9
2.7	Milloin dokumenttipohjaiset tietokannat ovat parempi vaihtoehto kuin muut tietokantamallit.....	11
3	Firebase	12
3.1	Firebase projekti	12
3.2	Firestore	13
3.3	Realtime database	14
3.4	Autentikointi	14
3.5	Hosting	15
4	React	15
5	Tulokset	17
5.1	Toiminnallisen opinnäytetyön tavoitteet	17
5.2	Verkkosivustossa käytetyt ohjelmointikielet ja palvelut	18
5.3	Firebasen alustaminen	18
5.4	Firestore-tietokannan suunnittelu ja toteutus	21
5.5	Käyttöoikeuksien rajoitus Firestore-tietokantaan	24
5.6	Sijoituksen tallentaminen Firestore-tietokantaan sovelluksessa	25
5.7	Sijotuskohteiden tietojen lukeminen Firestore-tietokannasta	26
5.8	Sijotusten poistaminen Firestore-tietokannasta	28
5.9	Sijotuskohteiden päivittäminen Firestore-tietokannassa	29
5.10	Lomakkeet	31
5.11	Autentikointi	33
5.12	Käyttöliittymän toteuttaminen.....	36
5.13	Pohdinta	38
	Lähteet.....	39

1 Johdanto

Opinnäytetyö on toiminnallinen työ, jossa toteutaan sijoittamiseen liittyvän web-sovellus. Sovelluksessa käytetään Firebase-kehitysalustaa ja React-nimistä JavaScript-kirjastoa käyttöliittymän tekemiseen. Teoriaosuudessa käsittelen yleisimpiä tietokantamalleja ja niiden eroja, sekä milloin mitäkin kannattaa käyttää. Sen lisäksi käsittelen Firebasen palveluita, joissa käydään läpi Firestore-tietokantaa, käyttäjien autentikointia ja web-sovelluksen julkaisemista sen Hosting-palvelun avulla. Toiminnallisessa osuudessa julkaisen web-sovelluksen, jonka avulla käyttäjät voivat tallentaa sijoituskohteita portfolioon. Portfoliosta voi nähdä tietoa yksittäisistä sijoituskohteista sekä tiivistelmiä portfoliosta. Käyttäjillä on käyttöoikeus vain omiin tietoihin, joita voi lukea, kirjoittaa, muokata ja poistaa. Web-sovellus on responsiivinen eli sen sisältö mukautuu erikokoisille ruuduille, kuten puhelimen, iPadin ja tietokoneen ruudulle.

2 Yleisimmät tietokantamallit

2.1 Suosituimmat tietokannat

Yleisimmillä tietokantamalleilla tarkoitetaan tässä opinnäytetyössä NoSQL- ja SQL-tietokantoja. NoSQL on kokoelma tietokantoja, joihin tieto tallennetaan eri tavalla kuin SQL:n relaatiotietokantojen tauluihin.

Suosituimpia tietokantoja on lueteltuina taulukossa 1, jonka tulokset perustuvat DB-engines-sivuston laskelmiin. Taulukossa on tietokannan nimi ja mille tietokantamallille sen toiminta perustuu. Top 10 tietokannoilla on enemmistö verrattuna NoSQL-tietokantoihin. (DB-engines 2023a.) Kaikista suosituin dokumenttitietokanta DB-Engine Rankingissa on MongoDB, joka on samalla

myös kaikista suosituin NoSQL-tietokanta. Tunnettuja yrityksiä, jotka käyttävät sitä ovat Vodafone, Bosch ja eBay. (Carvalho, Sa ja Berdardino 2023.)
 Googlen Firestore-tietokanta on sijalla 47 (Db-engines 2023a
).

Taulukko 1. 10 suosituinta tietokantaa ja Firestore (DB-engines 2023a).

Sija	Tietokanta	Tietokantamalli
1	Oracle	Relaatio
2	MySQL	Relaatio
3	Microsoft SQL Server	Relaatio
4	PostgreSQL	Relaatio
5	MongoDB	Dokumentti
6	Redis	Avain-arvo
7	IBM Db2	Relaatio
8	Elasticsearch	Search engine
9	Microsoft Access	Relaatio
10	SQLite	Relaatio
47	Google Cloud Firestore	Dokumentti

DB-Engine sivustolla mainintaan tarkemmin, mitkä osa-alueet vaikuttavat suosion arvioimisessa. Tiivistettynä suosiota mitataan hakukoneiden kyselyillä, Google Trendsillä, teknisillä kyselyillä alan keskeisillä sivustoilla, työpaikkailmoituksissa mainintoina, Twitterissä ja LinkedIn mainittuina. (Db-engines 2023b.)

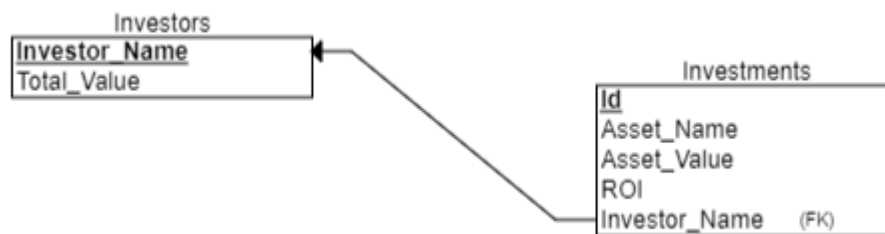
2.2 Relaatiotietokannat

Relaationtietokantojen kehitys alkoi 1970-luvulla, ja sen käyttäjäkunta levisi nopeasti maailmanlaajuisesti (Abramova, Berdardino & Furtado 2014).

Relaatiotietokannoissa tieto tallennetaan järjestettyihin tauluihin, joihin tehdään kyselyjä Structured Query Language (SQL) -kyselykielellä.

SQL on relaatiotietokantaan kehitetty kieli, jonka kehitti IBM. SQL:llä voidaan tehdä kyselyjä relaatiotietokantaan sekä muuttaa ja päivittää sitä (Dilling 2020).

Relaatiotietokannasta tehty malli esimerkki on näkyvillä kuviossa 1.



Kuvio 1. Relaatiomalli

Relaatiotietokannoissa tietoa tallennetaan tauluihin, joita voi olla useita.

Jokaisella tiedolla, joka varastoidaan taulukkaan riveittäin, sisältää oman ainutlaatuisen yksilöivän avaimen. Senpä vuoksi, samanlaista tietoa ei voida tallentaa vahingossa kahdesti samaan taulukkaan. Relatiotietokantoihin voidaan määritellä käyttäjille käyttöoikeuksia, kuten rajoittaa oikeuksia lukea tai kirjoittaa. (Dilling 2020.)

2.3 NoSQL

NoSQL nimi tulee englanninkielisistä sanoista "Not Only SQL", ja se onkin vaihtoehto relaatiotietokannoille (Sethi, Mishra ja Parnaik 2014). NoSQL

voidaan määritellä, että NoSQL-tietokantoihin ei säilötä tietoa

relaatiotietokantojen tapaan. NoSQL:n neljä keskeisintä tietokantatyyppeä ovat dokumentti (document), avain-arvo (key-value), Sarakevarasto (wide-column) ja verkko (graph). (MongoDB 2023). NoSQL-tietokannoissa on olemassa siis erilaisia tietokantatyyppejä. NoSQL-tietokannat eroavat toisistaan esimerkiksi

siinä, kuinka niihin tallennetaan tietoa ja kuinka tieto voidaan hakea niistä.
(Abramova, Berdardino & Furtado 2014.)

2.4 Dokumenttitietokannat

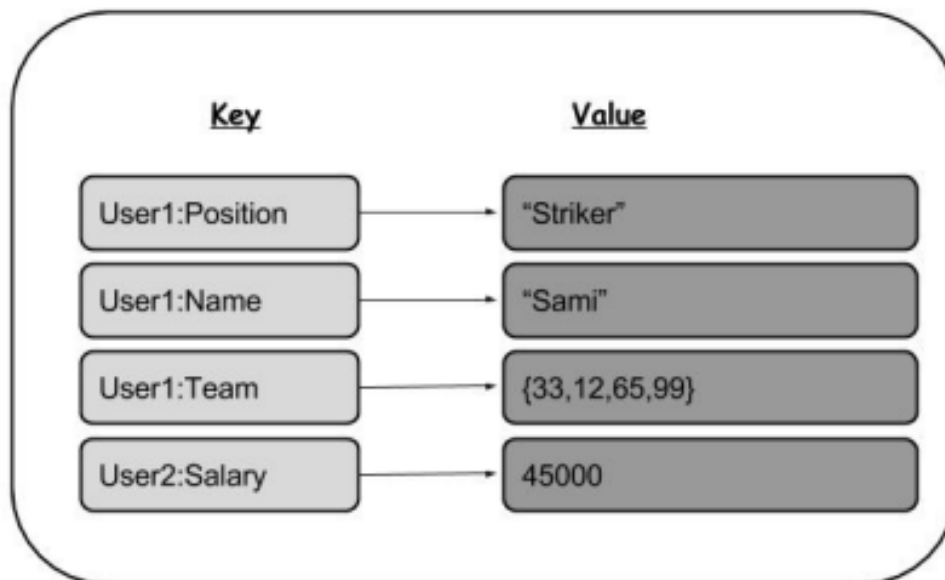
Nimensä mukaisesti dokumenttitietokannat on tehty dokumenttien hallintaan. Tietoa tallennetaan dokumenttiin, jossakin standardoidussa tiedonmuodossa, kuten JSON(JavaScript Option Notations) ja BSON (binaarinen JSON). (Eshtay, Sleit & Aldwairi 2019.) Näiden ohella myös XML (Gianina 2019). Dokumenteissa tietoa tallennetaan nimi-arvo-pareina. Malli tällaisesta tietokannasta on kuviossa 2. Dokumenteissa skeema on joustava ja ne soveltuvat suurien tietomäärien tallentamisen. Dokumentista voidaan tehdä kyselyjä hyvällä suorituskyvyllä. (Eshtay, Sleit & Aldwairi 2019.) Kyselyjä voidaan tehdä avaimen sekä arvon perusteella (Gianina 2019).



Kuvio 2. Dokumenttitietokannan malli (Eshtay 2019).

2.5 Avain-arvo tietokannat

NoSQL-tietokannoista yksinkertaisin malli on Avain-arvo-tietokanta. Se koostuu yksilöllisestä avaimesta ja siihen liittyvästä arvosta. Arvo sisältää tietoa, joka voi olla esimerkiksi asiakirja, merkkijono tai kuva. Arvo voi myös olla kooltaan ja rakenteeltaan mielivaltainen. (Ramza, Sarwar & Kazmi 2019.) Tietokannasta tiedon etsiminen onnistuu pelkästään avaimen perusteella (Gianina 2020). Tiedon etsiminen on nopeaa, koska tietomallirakenne on yksinkertainen. Malli avain-arvo-tietokannasta on esillä kuviossa 3. Tietokantaa voidaan kuvitella relaatiotaulukoksi, jossa on kaksi kenttää avaimelle ja arvolle. Esimerkiksi avain-arvo-tietokantaa käyttävät LinkedIn Voldemort ja Amazon dynamo. (Eshtay, Sleit & Aldwairi 2019.)



Kuvio 3. Avain-arvo-tietokanta malli (Eshtay 2019).

2.6 SQL- ja NoSQL-tietokantamallien erot

SQL- ja NoSQL-tietokantojen erojen vertailun voisi aloittaa, siitä kuinka NoSQL on saanut nimensä. NoSQL-nimeä alettiin käyttämään vuonna 1998 Carlos Strozzin johdosta kuvaamaan avoimen lähdekoodin tietokantaa, jossa SQL-kieltä ei käytetä. Strozzi viittasi NoSQL-tietokantaan myös nimellä Norel (no

relational). Tärkeä ero SQL- ja NoSQL-tietokantojen välillä on se, että NoSQL ei ole relaatiotietokanta. Relaatiotietokannan tietorakenne tulee myös määritellä ennakkoon ennen kuin tietoa tallennetaan siihen. (Abramova, Berdardino & Furtado 2014.) Keskeisimpiä eroavuuksia SQL-tietokannan ja NoSQL:n välillä on vertailtu taulukossa 2. (Gianina 2020).

Tietokanta	SQL	NoSQL
Tietomalli	Relaatio	dokumentti, avain-arvo, Sarakevarasto ja verkko
Transaktio	ACID	CAP theorem Compliance/BASE
Kyselykieli	SQL	OO APIs, SQL-kaltaiset
Tietorakenne	jäykkä	joustava
Skaalattavuus	matala	korkea
Hajautetun tietokannan tuki	matala	korkea
Kustannus	korkea	matala

Taulukko 2. Keskeisimpiä eroavuuksia SQL- ja NoSQL-tietokantojen välillä (Gianina 2020).

NoSQL-tietokannoissa tietoa merkitään siis eri lailla kuin relaatiotietokannoissa. Tämän lisäksi toisin kuin relaatiotietokannoissa, se on myös hajautettua. Siispä NoSQL-tietokannoissa voi olla useita palvelimia ja erilaisia ympäristöjä, jolloin tietoa voidaan käyttää joustavasti ja tehokkaasti. Eroavuuksia näiden kahden tietokantamallin välillä huomataan siis skaalattavuudessa. Relaatiotietokannat kykenevät skaalautumaan pääasiallisesti vertikaalisesti eli lisäämällä palvelimien kantokykyä. NoSQL-tietokannat skaalautuvat puolestaan vaakasuuntaisesti eli tietokantaa voidaan jakaa monelle palvelimelle ja uusia palvelimia voidaan lisätä aina tarpeen mukaan. (Gianina 2020.)

Gianinia käsittelee artikkelissaan NoSQL:n ja relaatiotietokantojen eroja (Gianina 2020). Hänen mukaansa NoSQL-tietokannoissa voi olla osa tai kaikki seuraavista ominaisuuksia, jotka poikkeavat relaatiotietokannoista:

1. NoSQL ei sovellu ACID-transaktioihin (atomisuus, eheys, eristyisyys ja pysyvyys)
2. NoSQL-tietokannat ovat skeemattomia tai niitä koskee vähäiset skeemarajoitukset. Siksi tiedon tallentamisessa on joustavuutta, ja tallennuksen voi tehdä erilaisilla rakenteilla.
3. NoSQL ei sovellu monimutkaisiin kyselyihin, joka selittyy sillä, ettei niissä ole relaatiotietokannan tapaisia suhteita, vierasavaimia ja JOIN-lauseita.
4. NoSQL:ssä ei käytetä relaatiotietokantojen SQL:ää tietokantaan tehtäviin kyselyihin.

2.7 Milloin dokumenttipohjaiset tietokannat ovat parempi vaihtoehto kuin muut tietokantamallit

Dokumenttitietokanta on NoSQL-tietokanta. NoSQL-tietokanta on tarkoitettu erityisesti suurille hajautetuille verkkosovelluksille. Ne ovatkin hyödyllisiä erityisesti, jos tiedonmäärä on valtava ja käyttäjälle halutaan taata jatkuva saatavuus tietokantaan edullisesti. (Gianina 2020.) Dokumentti-pohjainen tietokanta onkin hyvä vaihtoehto, jos käsitellään suuria määriä tietoa. Jos halutaan hyvin horisontaalisesti skaalautuvaa tietokanta, niin silloin dokumenttipohjainen soveltuu siihen hyvin. (Carvalho, Sa ja Berdardino 2023.)

Dokumenttipohjaisessa skeema on myös rakenteeltaan joustava, jota pidetään sen pääasiallisena hyötynä. Skeeman joustavuuden ansiosta dokumenttiin tehtävien muutosten päivittäminen helpottuu. Relaatiotietokannoissa sen sijaan on ennalta määriteltä rakenne. (Carvalho, Sa ja Bernardino 2023.). Tästä on erityisesti hyötyä, jos tietokannan rakennetta ei tiedetä ennakkoon.

Tämän opinnäytetyön toiminnallisessa osuudessa käytän Firebasen Firestore-dokumenttitietokantaa. Valitsin Firebasen MongoDB:n sijaan, koska Firebase on muutakin, kuin pelkkä Firestore-dokumenttitietokanta. Firebasen palveluilla pystyy myös todentamaan (autentikointi) käyttäjät sisäänkirjautumisessa, jolloin tietoturva paranee. Pystyn lisäksi myös julkaisemaan web-sovelluksen Firebasen Hosting-palvelun avulla. Firebase on siis mielestäni monipuolisempi kuin MongoDB, joka onkin pääsyy sen valitsemiseen.

3 Firebase

3.1 Firebase projekti

Firebase tarjoaa käyttäjille palveluita, kuten Firebase hosting (Google 2023c). Muita suosittuja palveluita ovat Authentication, Realtime Database, Cloud Firestore, Cloud Storage ja Cloud function (Google 2023d.). Firebase on Googlen sovellusten kehitysalusta, jonka palveluita käyttävät seuraavat yritykset: Trivago, lyft, Duolingo, Alibaba.com ja venmo (Google 2023e). Firebase tarjoaa kaksi erilaista hinnoitteluluokkaa eli Spark ja Blaze. Näistä kahdesta Spark on ilmainen, mutta sillä voi tehdä rajoitetummin. Blaze on maksullinen, mutta sillä voi tehdä laaja-alaisemmin. Blaze:n hinnoittelu menee käytönmukaisesti, ja sillä saa pääsylvän käyttää esimerkiksi Google Cloudin maksullisia palveluita ja tuotteita. (Google 2023f).

Näiden palvelujen käyttöönottamisessa pitää aluksi luoda projekti Firebaseeen, johon voidaan rekisteröidä yksi tai useampi Firebase-sovellus. Tehty projekti toimii eräänlaisena säiliönä, joka sisältää sovellusten lisäksi myös palvelut ja resurssit. Samaan projektiin rekisteröidyillä sovelluksilla on puolestaan pääsy kaikkiin samoihin palveluihin ja resursseihin. (Google 2023d).

Firebase tarjoaa myös hostingpalveluita websovellukselle (Google 2023c). Firebasen autentikoinnilla voidaan identifoida käyttäjät, jota voidaan myös käyttää tiedon tallentamisessa turvallisesti (Google 2023b).

3.2 Firestore

Firebasen Firestore on dokumenttipohjainen NoSQL-pilvitietokanta, joka skaalautuu käytön mukaisesti (Google 2023a). Tietoa säilötään dokumenteissa avain-arvo-pareissa. Dokumentit puolestaan ovat aina osa kokoelmaa. Malli dokumenteista, jotka säilötään kokoelmassa on kuviossa 4. Siinä users on kokoelma, johon on säilötty dokumentteina erilaisia käyttäjiä. Tässä tapauksessa avelace ja aturing ovat omia dokumentteja, joihin on tallennettuna tietoa avain-arvo-pareissa. (Google 2023g.)



Kuvio 4. Kokoelma ja siinä olevat dokumentit Firestoressa

Firestoren dokumentteja kuvataan Googlen omissa asiakirjoissa, että ne muistuttavat rakenteeltaan JSON-tiedostoa. Toisaalta JSON-tiedostosta ne

eroavat esimerkiksi sillä, että ne tukevat ylimää räisiä tietotyypp ejä ja kokoa on rajoitettu 1 megatavuun. (Google 2023g).

3.3 Realtime database

Realtime database on Firebasen alkuperäinen NoSQL-tietokanta, jossa tietoa säilötään yhdessä JSON-puu:ssa. (Google 2023h). Tietokanta on siis Googlen Firebasessa oleva pilvipalvelu. Siihen tietoa tallennetaan JSON-objekteina (Google 2023i). Tietokannassa olevaa tietoa voidaan siirtää eri palvelimille, ja jos pääsy yhteen estyy, tieto siirtyy toiselle palvelimelle automaattisesti. Tiedot päivitetään kaikkialle reaaliajassa ja sen voi huomata käyttäjien päivittäessä tietoa. Tieto välittyy myös muille samaa Firebase-tietokantaa käyttävien sovellusten käyttäjille.

. (Ohyver, Moniaga, Sungkawa, Subagyo ja Chandra 2019.)

3.4 Autentikointi

Firebase Authentication huolehtii käyttäjien todentamisen ja valtuuksien myöntämisen sisäänkirjautumisen yhteydessä. Todentamiseen on useita vaihtoehtoja: tunnistautua voi sähköpostilla ja salasanalla tai jonkin muun yleisesti käytössä olevan palvelun tunnuksella. Esimerkiksi Googlen, Applen, Facebookin, X ja GitHubin tunnuksilla. (Google 2023b). Tunnistautumisen yhteydessä voi pyytää kaksivaiheisen vahvistuksen, esimerkiksi puhelimeen lähetettävällä viestillä. Käyttäjille voidaan tehdä myös väliaikaisen anonyymin tilin, joilla pääsee käyttämään sovellusta ilman sisäänkirjautumista. Jos käyttäjä haluaa, hän voi tarvittaessa päivittää anonyymin tilin tavalliseksi. (Google 2023b.)

Autentikoidulla käyttäjille on oletuksena, että käyttäjät pystyvät lukemaan ja kirjoittamaan tietoa Firebasen Realtime Database ja Cloud Storage tietokantoihin. Käyttäjien pääsyoikeuksia voidaan rajoittaa tai antaa

lisäoikeuksia tarpeiden mukaan. Firebase Authentication palvelulla on kaksi hintaluokittelua. Ilmaisessa Spark-nimisessä palvelu sallii 3000 aktiiviistä käyttäjää vuorokaudessa. Maksullisessa Blaze-nimisessä palvelussa veloitetaan käytön mukaan, mutta vasta kun päivittäisiä aktiivisia käyttäjiä on yli 50000. (Google 2023b.)

3.5 Hosting

Firebase Hosting palvelulla voi julkaista verkkosivustoja. Ennen verkkosivuston julkaisua, sillä voi myös testailla ja katsoa muutoksia paikallisesti isännöidyllä URL-osoitteella. Hosting palveluun kuuluu verkkosivustoille suojattu yhteys SSL-suojauksella. Oletusasetuksena verkkosivustoilla on ilmainen aliverkkotunnus (subdomain), jossa on nimeen sisällytettynä joko web.app tai firebaseapp.com. Verkkosivustolle voi myös tehdä yksilöllisen verkkotunnuksen. (Google 2023c.)

Firebase Hosting hinnoitteluun vaikuttavat käytetty tallennustila ja tiedonsiirto. Tallennustila on tosin maksuton aina 10 gigabittiin asti. Maksullisessa Blaze-suunnitelmassa tallennustila on ilmainen 10 gigabittiin asti ja jokainen sen ylittävä gigabitti maksaa 0,026\$. Ilmaisessa Spark-suunnitelmassa 10 gigabitin täyttyessä, joutuu poistamaan aiempaa sisältöä tallennustilasta, jotta voisi julkaista uutta. Tiedonsiirto on rajattu ilmaiseksi kuukausikohtaiseksi 10 gigabittiin kuukaudessa. (Google 2023j.)

4 React

React on avoimen lähdekoodin JavaScript-kirjasto, jota käytetään käyttöliittymien tekemiseen. Reactin luoja on Jordan Walke, mutta sitä kehitettiin ja ylläpidettiin Facebookin johdosta. React kehitettiin alun perin jo vuonna 2011, mutta julkaisi sen julkisesti vasta maaliskuussa 2013. Sillä toteutetut verkkosovellukset ovat single page sovelluksia. Käytännössä React

mahdollistaa suurien ja monimutkaisten verkkosivujen tekemiseen. Sovellusten teossa tehdään komponentteja, jotka sisältävät tarvittavan toiminnallisuuden. Näitä komponentteja voidaan uudelleen käyttää sovelluksessa, joka helpottaa sovelluksen ohjelmointia. React:ssa käytetään virtuaalista DOM (Document Object Model), joka nopeuttaa sovelluksen suoritusta. (Srivasta, Laxmi, Pratima & Kirti 2022.)

5 Tulokset

5.1 Toiminnallisen opinnäytetyön tavoitteet

Toiminnallisessa osuudessa tein sijoittajille tarkoitetun sijoitusportfolion. Portfolioon voi tallentaa tietoa sijoituskohteista, jotka puolestaan tallennetaan Firestoren dokumenttipohjaiseen tietokantaan. Käyttäjät voivat kirjoittaa, lukea, poistaa ja muokata omia sijoitustietoja. Käyttäjien oikeuksia on rajoitettu sellaiseksi, että vain omia dokumentteihin on pääsyoikeus. Sisäänkirjautumista varten käyttäjät autentikoidaan Firebasen Authentication palvelun avulla, ja sitä varten tulee rekisteröidä tili. Käyttäjät tunnistetaan sähköpostilla ja salasanalla. Käyttäjät voivat myös halutessaan poistaa kaiken tietonsa rekisteristä. Käyttäjää varten toteutetaan responsiivinen verkkosivusto, josta voi nähdä tiivistelmän portfolioista sekä yksittäisistä sijoituskohteista. Verkkosivusto julkaistaan Firebasen Hosting palvelun avulla. Tarkemmat toiminnalliset vaatimukset verkkosivuston ominaisuuksista on lueteltu alapuolella.

Autentikointi

1. Käyttäjä voi rekisteröityä ja sisäänkirjautua sivustolle
2. Sisäänkirjautuminen ainoastaan sähköpostin ja salasanan avulla
3. Käyttäjät autentikoidaan Firebasen Authentication-palvelun avulla
4. Käyttäjä voi uloskirjautua sivustolta

Tietokanta

5. Toteuttaa Firestore-tietokanta, johon käyttäjät voivat tallentaa tietoa
6. Käyttäjä voi tallentaa, muokata, poistaa ja nähdä omia sijoituskohteita

Julkaistu

7. Verkkosivusto on julkaistu Firebasen Hosting-palvelun kautta

5.2 Verkkosivustossa käytetyt ohjelmointikielet ja palvelut

Verkkosivuston käyttöliittymässä käytin React-JavaScript-kirjastoa. Ohjelmointikielenä toimi pääasiallisesti JavaScript. Verkkosivuston tyylittelyssä käytin CSS-kieltä (Cascading Style Sheet), jonka avulla sivustosta saadaan myös responsiivinen. Verkkosivuston ohjelmoinnissa käytin myös on HTML-kuvauskieltä (Hypertext Markup Language). Kuvassa 1 on näkyvillä kooste käytetyistä ohjelmointikielistä suhteessa toisiinsa.

Languages



Kuva 1. Käytetyt ohjelmointikielet ohjelmistossa.

Ohjelmistossa palvelimena toimivat Firebasen palvelut. Verkkosivusto julkaistiin Firebasen Hosting-palvelun kautta. Käyttäjien rekisteröitymisessä ja sisäänkirjautumisessa varmennus toteutettiin Firebasen Authentication-palvelulla. Tietokantana toimii dokumenttipohjainen Firestore-tietokanta. Ohjelmistossa käytetään myös lähdekoodin editoimiseen Visual Studio Code:a ja NodeJS:ää kehitysalustana.

5.3 Firebasen alustaminen

Firebasen palveluiden käyttöönottamista varten tuli luoda aluksi projekti. Luettelo Firebasen käyttöönoton keskeisimmistä yksityiskohdista on lueteltu alapuolella.

1. Kirjautuminen Google-tunnuksella Firebase-verkkosivustolle, joka on edellytys Firebase palvelujen käyttämiselle
2. Firebasen alustaminen tehtiin verkkosivustolla <https://firebase.google.com/>
3. Lisäsin uuden projektin Firebase console:lla
4. Annoin projektille nimen investors-database
5. Valitsin projektille Google Analytics -toiminnon. Käytännössä projektissa ei ollut tarvetta tälle palvelulle, eikä sitä siksi käytetty.
6. Projekti on nyt alustettu
7. Käytössä on ilmainen spark, joka on oletuksensa projekteissa.
8. Sovelluksen rekisteröimistä varten on olemassa 5 vaihtoehtoa: iOs+, android, web, unity ja Flutter

Valitsin näistä vaihtoehtoista web, koska tässä opinnäytetyössä tehdään verkkosivusto.
9. Seuraavaksi nimetään sovellukselle kutsumanimi, jonka ohella voi myös ottaa Firebase Hosting -palvelun. Kutsumanimi on investorApp ja rekisteröin sille Hosting-palvelun. Hosting-palvelun kautta julkaistaan myös verkkosivusto.
10. Käytän Node.js ohjelmistossa, joten asensin Firebasen seuraavalla komennolla: firebase npm install firebase.

Tämän lisäksi web-sovellukseen tulee lisätä annettu firebaseConfig-niminen tiedosto. Se sisältää projektille yksilöivää tietoa, kuten api-avaimen (Kuva 2).

```
src > JS firebaseConfig.js > ...
1  // Import the functions you need from the SDKs you need
2  import { initializeApp } from "firebase/app";
3  import { getAnalytics } from "firebase/analytics";
4  import { getAuth } from "firebase/auth";
5  import { getFirestore } from "firebase/firestore";
6  // TODO: Add SDKs for Firebase products that you want to use
7  // https://firebase.google.com/docs/web/setup#available-libraries
8
9  // Your web app's Firebase configuration
10 // For Firebase JS SDK v7.20.0 and later, measurementId is optional
11 const firebaseConfig = {
12   apiKey: `${process.env.REACT_APP_APIKEY}`,
13   authDomain: `${process.env.REACT_APP_AUTHDOMAIN}`,
14   projectId: `${process.env.REACT_APP_PROJECT_ID}`,
15   storageBucket: `${process.env.REACT_APP_STORAGE_BUCKET}`,
16   messagingSenderId: `${process.env.REACT_APP_MESSAGING_SENDER_ID}`,
17   appId: `${process.env.REACT_APP_APPID}`,
18   measurementId: `${process.env.REACT_APP_MEASUREMENT_ID}`,
19 };
20
21 // Initialize Firebase
22 export const app = initializeApp(firebaseConfig);
23 const analytics = getAnalytics(app);
24
25 //Authentication export
26 export const auth = getAuth(app);
27
28 //Firestore db export
29 export const db = getFirestore(app);
30
```

Kuva 2. firebaseConfig.js tiedosto sovelluksessa.

firebaseConfig-tiedosto sisältää alun perin salassa pidettäviä Firebase avaimia, joiden salaamista varten tein env-tiedoston. Sovellus on julkisesti näkyvillä GitHub-sivullani, joten estin env-tiedoston avulla firebaseConfig-tiedostossa olevien yksilöllisten avaimien näkymisen muille. Env-tiedostossa olevat avaimet käsitellään ohjelmistossa process.env.REACT_APP kautta.

11. Asensin ohjelmistoon myös Firebase CLI, jota tarvitaan Firebase Hosting-palvelussa.

Asennus tehtiin komennolla: `npm install -g firebase-tools`

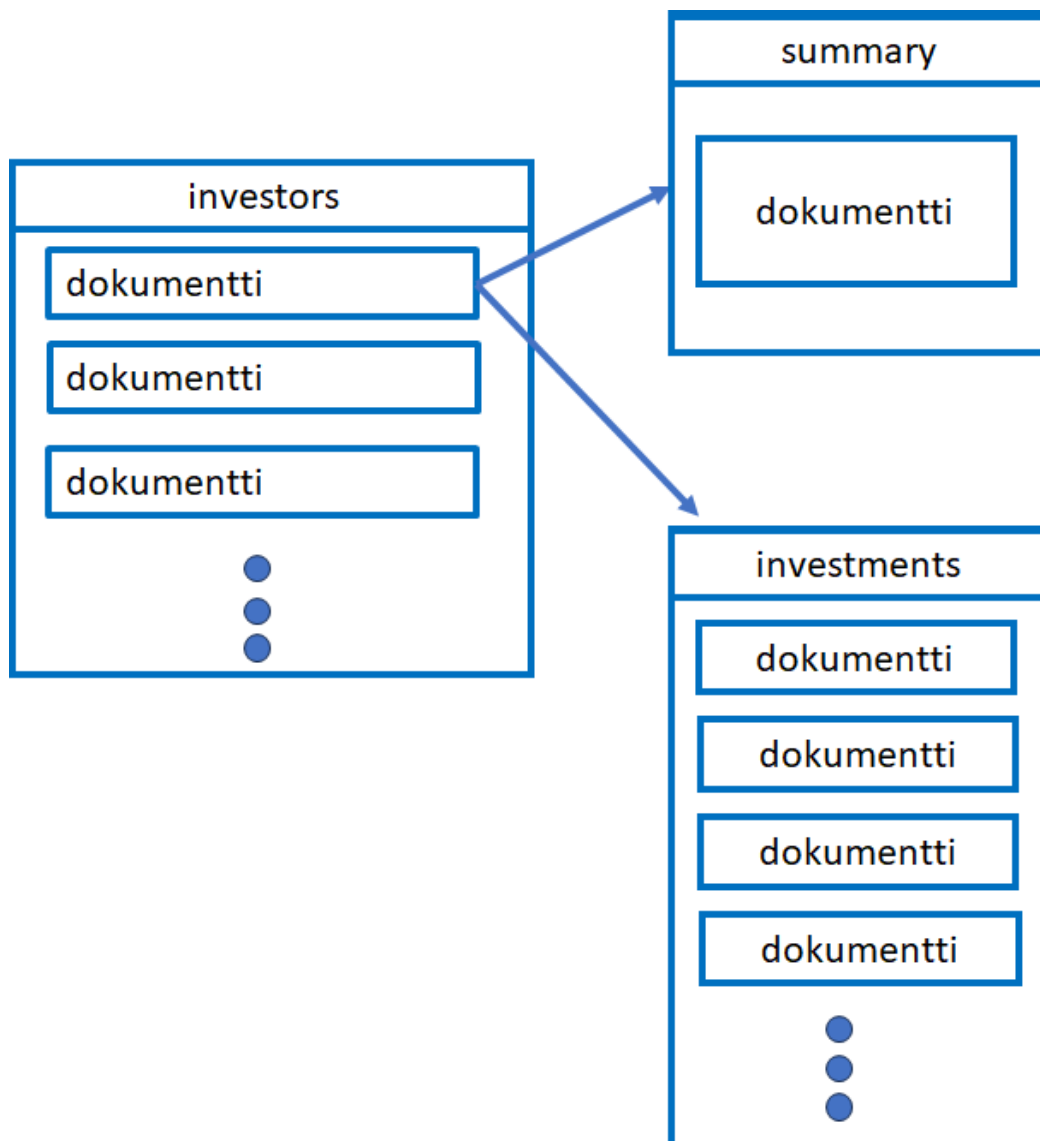
12. Firebase Hosting -palvelua käytetään verkkosivuston julkaisemissa, jota varten tuli sisäänkirjautua myös Google-tililleni.

Sisäänkirjautuminen tehdään komennolla `firebase login` terminaalissa.

Sisäänkirjautuminen jälkeen projekti myös alustettiin komennolla `firebase init`.

5.4 Firestore-tietokannan suunnittelu ja toteutus

Firestore toimii tietokantana sijoituskohteiden tallentamisessa palvelimelle. Tietokanta suunniteltiin sellaiseksi, että jokaisella käyttäjällä on yksilöivä id-tunnus. Käyttäjien id-tunnus saadaan autentikoinnin yhteydessä, ja sitä samaa käytetään myös Firestore-tietokannassa. Jokaisella käyttäjällä on 3 erilaista kokoelmaa eli `investors`, `investments` ja `summary`. Jokaiselle rekisteröidylle käyttäjälle luodaan oma dokumentti kokoelmaan `investors`. `Investors`-kokoeman dokumenteissa jokaisella käyttäjällä on oma uniikki id-tunnus. Käyttäjän henkilökohtaiset sijoitustiedot säilötään kokoelmissa `investments` ja `summary`. Näistä `investments`-kokoelmassa on tiedot yksittäisistä sijoituskohteista, joista jokaisella on oma dokumentti. Sijoituksien tiivistelmä on puolestaan `summary`-kokoelmassa. Kokoelmat ja dokumentit ovat esillä kuvioissa 5.



Kuvio 5. Firestore tietokannassa olevat kokoelmat ja niiden dokumentit.

Investors-kokelmassa voi olla useita rekisteröitynyttä käyttäjää, joilla jokaisella on omat kokoelmat. Näihin kokoelmiin käyttäjä voi tallentaa tietoa omista sijoituksistaan. Sijoittajien investors-kokelmassa on sijoittajilla oma dokumentti, jossa on pelkästään käyttäjän henkilökohtainen uid eli id-tunniste. Investments-kokelmassa voi olla useita sijoittajan dokumenttia, ja jokainen näistä dokumenteista sisältää yhden sijoituskohteen tiedot. Investments-kokoelman dokumenteilla on ainutlaatuinen id, jonka avulla dokumentit voidaan erottaa toisistaan. Sijoituskohteista tehty myös tiivistelmä dokumentti, joka on summary-kokelmassa, jolla on vain yksi dokumentti nimeltään summaryInvestments.

Sijoituskohteista tallennetaan investments-kokoelman dokumenttiin. Dokumentin kenttiin kerätään seuraavanlaiset tiedot Amount, boughtDate, cashInvested, cost, name, price ja value, joiden ohella firestore myös luo uniikin id:n dokumentille. Kuviossa 6. On malliesimerkki täytetyistä tiedoista.

investments (kokoelma)

id (dokumentti)

```
amount: 42
boughtDate: "22.11.2023"
cashInvested: 100
cost: 10
name: "Othello"
price: 20
value: 2000
```

Kuvio 6. Dokumentti sijoituksesta investments-kokoelmassa.

Sijoituskohteesta tallennetaan dokumenttiin tietoa sijoituskohteen lukumäärästä (amount), ostopäivämäärästä (boughtDate), kuinka paljon valuuttaa on käytetty sijoitukseen (cashInvested), sijoituksen hinta ostopäivällä (cost), sijoituksen nimi (name), sijoituksen nykyinen hinta (price), sijoituksen arvo (value). Näiden lisäksi jokaisella dokumentilla on myös oma ainutlaatuinen id.

Sijoittajan kaikista sijoituksista kootaan myös tiivistelmä, joka tallennetaan summary-kokoelmassa olevaan dokumenttiin. Dokumentissa olevat avain-arvoparit ovat näkyillä kuviossa 7.

summary (kokoelma)

summaryInvestments (dokumentti)

AssetsSum: 42
NegativeAssetsSum: 3
PositiveAssetsSum: 4
TotalCost: 21000
TotalValue: 170000

Kuvio 7. Dokumentti sijoitusten tiivistelmästä summary-kokoelmassa.

Summary-kokoelmassa olevan dokumentin nimi on summaryInvestments, joka toimii myös sen yksilöivänä id. Dokumenttiin kerätään tietoja sijoitusten kokonaislukumäärästä (AssetsSum), negatiivisten sijoitusten lukumäärästä (NegativeAssetsSum), positiivisten sijoitusten lukumäärästä (PositiveAssetsSum), sijoitukseen käytetyn valuutan kokonaismäärästä (TotalCost) ja sijoitusten kokonaisarvosta (TotalValue).

5.5 Käyttöoikeuksien rajoitus Firestore-tietokantaan

Jokaiselle rekisteröidylle käyttäjälle on tehty käyttöoikeudet Firestore-tietokantaa. Käyttäjät voivat lukea, kirjoittaa, poistaa ja päivittää vain ja

ainoastaan omia tietojaan. Käyttöoikeudet tarkistetaan vertaamalla rekisteröityneen käyttäjän omaa uniikkia id:tä (uid) investors-kokoelmassa olevaan investorsId:een. Käyttöoikeuksien rajoitukset on merkattu Firestore:ssa olevaan sääntöjen osioon, jonka sisältö on lueteltu alapuolella.

```
rules_version = '2';
```

```
service cloud.firestore {  
  match /databases/{database}/documents {  
    match /investors/{investorId}/{document=**} {  
      allow read, write, delete, update: if request.auth != null && request.auth.uid  
      == investorId;  
    }  
  }  
}
```

5.6 Sijoituksen tallentaminen Firestore-tietokantaan sovelluksessa

Uuden dokumentin kirjoittamisessa Firestore-tietokantaan käytettiin Firebasen addDoc-funktiota. Ohjelmistosta otettu kuvakaappaus sijoittajan sijoituksen lisäämisestä tietokantaan on kuvassa 3. Dokumentin haluttu sijainti kokoelmassa sekä sen sisältö luetellaan osana addDoc-funktiota.

```

14 //Firestore investors collection reference
15 const investorCollectionRef = collection(db, "investors");
16
17 const tempValue = Number(parseFloat((amount * price).toFixed(2)));
18
19 //structure of new investment data
20 const newInvestment = {
21   name: name,
22   price: price,
23   amount: amount,
24   cost: cost,
25   cashInvested: cashInvested,
26   boughtDate: boughtDate,
27   value: tempValue,
28 };
29
30 //investor own document
31
32 const investorDocument = doc(investorCollectionRef, UIDInvestor); //
33
34 //investor subcollection that contains investors own investments
35 const investorInvestmentsCollection = collection(
36   investorDocument,
37   "investments"
38 );
39 try {
40   await addDoc(investorInvestmentsCollection, newInvestment);
41 } catch (error) {
42   console.error(error);
43 }
44
45 // update portfolio Summary
46 const assetSum = 1;
47 const value = tempValue;
48 SummaryUserInvestments(UIDInvestor, assetSum, cashInvested, value);
49 };
50
51 export default AddNewInvestment;

```

Kuva 3. Sijoituksen lisääminen Firestore-tietokantaan.

Sijoituksien lisääminen luo uuden dokumentin investments-kokoelmaan. Dokumentin lisäämisen yhteydessä addDoc-funktio luo myös dokumentille oman yksilöivän id-tunnuksen.

5.7 Sijoituskohteiden tietojen lukeminen Firestore-tietokannasta

Käyttäjän sijoituskohteet luetaan Firestore-tietokannasta, josta luetaan pelkästään käyttäjän omia tietoja. Tietokannasta lukeminen tehtiin Firestoren-funktiolla `getDocs`. Sen käyttämistä varten määriteltiin luettavien dokumenttien sijainti kokoelmassa. Kaikki sijoituskohteet luettiin `investments`-kokoelmasta, jonne pääsy edellyttää myös käyttäjän `investors`-kokoelmassa olevan `id`-tunnuksen mainitsemisen. Kuvassa 4 on näkyvillä ohjelmistosta otettu kuvakaappaus tietojen lukemisesta

```
import { getDocs, collection, doc, query } from "firebase/firestore";

export const ReadUserInvestments = async (UIDInvestor, setUserdata) => {
  try {
    if (UIDInvestor === null) {
      return;
    }
    //Firestore location of investor investments based on uid
    const investorInvestmentsCollectionRef = collection(
      db,
      "investors",
      UIDInvestor,
      "investments"
    );
    //query
    const queryObject = query(investorInvestmentsCollectionRef);

    const queryInvestmentsSnapshot = await getDocs(queryObject);

    const resultInvestments = queryInvestmentsSnapshot.docs.map((doc) => ({
      ...doc.data(),
      id: doc.id,
    }));

    setUserdata(resultInvestments);
  } catch (error) {
    console.error(error);
  }
};
```

Kuva 4. Sijoitusten lukeminen tietokannasta sovelluksessa.

Firestore-tietokantaan tehdään kysely `query`-funktiolla, jonka jälkeen voidaan hakea kaikki tietystä kokoelmassa olevat dokumentit `getDocs`-funktiolla. Yksittäisten sijoituskohteiden lisäksi luetaan Firestore:sta myös tiivistelmä sijoituksista. Tiivistelmän lukemisessa käytettiin puolestaan `getDoc`-funktiota, jolla voidaan lukea vain yksi dokumentti (Kuva 5).

```

import { db } from "../../firebaseConfig";
import { getDoc, doc } from "firebase/firestore";

export const ReadUserPorfolio = async (
  UIDinvestor,
  setPortfolioUser,
  portfolioUser
) => {
  //Firestore location of investor investments based on uid
  if (UIDinvestor === null) {
    return;
  }

  const investorPorfolioRef = doc(
    db,
    "investors",
    UIDinvestor,
    "summary",
    "summaryInvestments"
  );

  try {
    const queryPortfolioSnapshot = await getDoc(investorPorfolioRef);

    // setPortfolioUser(resultPortfolio);
    const investorPortfolio = queryPortfolioSnapshot?.data();
    if (investorPortfolio !== null) {
      //without this gives typerError: cannot read properties of null
      setPortfolioUser(investorPortfolio);
    }
  } catch (error) {
    console.error(error);
  }
};

```

Kuva 5. Sijoitusten tiivistelmän lukeminen Firestore-tietokannasta.

Sijoitusten tiivistelmä on Firestore-tietokannassa summary-nimisessä kokoelmassa. Tiivistelmä on yksittäinen dokumentti, joka on nimeltään summaryInvestments (Kuva 5).

5.8 Sijoitusten poistaminen Firestore-tietokannasta

Käyttäjä voi poistaa tallentamiaan sijoituskohteita Firestore-tietokannasta. Sijoituksen dokumentti poistetaan investments-kokoelmasta, sen yksilöllisen tunnisteiden avulla (Kuva 6). Poistamisessa käytetään Firestore:n funktiota `deleteDoc`, jolla voidaan poistaa yksittäisiä dokumentteja.

```
import { db } from "../../firebaseConfig";
import { doc, deleteDoc } from "firebase/firestore";

export const DeleteInvestmentDoc = async (UIDinvestor, documentId) => {
  try {
    await deleteDoc(
      doc(db, "investors", UIDinvestor, "investments", documentId)
    );
  } catch (error) {
    console.error(error);
  }
};
```

Kuva 6. Sijoituksen poistaminen Firestore-tietokannasta.

5.9 Sijoituskohteiden päivittäminen Firestore-tietokannassa

Verkkosivustolla tallennettuja sijoituskohteita voi muokkailla lomakkeen avulla, jonka kautta muokkaukset myös päivittyvät Firestore-tietokantaan. Lomakkeesta saadut tiedot päivitetään dokumenttiin `setDoc`-funktiolla. Muokkaamista varten pitää tehdä viittaus dokumentin sijaintiin Firestore-tietokannassa, jota varten käytettiin myös `doc`-nimistä funktiota (Kuva 7).

```

src > components > Firestore > update > JS UpdateSingleInvestment.js > ...
1  import React from "react";
2  import { db } from "../../firebaseConfig";
3  import { setDoc, doc, increment } from "firebase/firestore";
4
5  export const UpdateSingleInvestment = async (
6    id,
7    UIDInvestor,
8    name,
9    amount,
10   price,
11   cost,
12   cashInvested,
13   boughtDate
14 ) => {
15   try {
16     if (UIDInvestor === null) {
17       return;
18     }
19     //reference investment location in firestore
20     const userSummaryRef = doc(db, "investors", UIDInvestor, "investments", id);
21
22     const tempValue = Number(parseFloat((amount * price).toFixed(2)));
23
24     //structure of new investment data
25     const UpdateInvestment = {
26       name: name,
27       price: price,
28       amount: amount,
29       cost: cost,
30       cashInvested: cashInvested,
31       boughtDate: boughtDate,
32       value: tempValue,
33     };
34
35     await setDoc(userSummaryRef, UpdateInvestment, { merge: true });
36   } catch (error) {
37     console.error(error);
38   }
39 };

```

Kuva 7 Yksittäisen sijoituksen päivittäminen.

Yksittäisen sijoituksen muokkaamisen yhteydessä tehdään myös päivitys sijoitusten tiivistelmään. Tiivistelmän muokkaamisessa käytettiin setDoc-funktiota, jolla muokataan jo olemassa olevaa dokumenttia. Funktion yhteydessä annetaan objekti (summaryInvestments), joka sisältää koosteen kaikkien sijoitusten yhteenlasketuista arvoista (Kuva 8).

```
const summaryInvesments = {
  AssetsSum: increment(assetSum),
  NegativeAssetsSum: increment(negativeA),
  PositiveAssetsSum: increment(positiveA),
  TotalCost: increment(cashInvested),
  TotalValue: increment(value),
};
try {
  await setDoc(userSummaryRef, summaryInvesments, { merge: true });
} catch (error) {
  console.error(error);
}
};
```

Kuva 8. Muokkauksen tekeminen tiivistelmä dokumenttiin Firestore:ssa.

5.10 Lomakkeet

Sijoitusten tietojen keräämistä varten tehtiin lomake verkkosivustolle.

Lomakkeesta annetut tiedot tallennetaan Firestore-tietokantaan. Sovellukseen tehtiin myös funktio, joka laskee lomakkeen tiedoista sijoituksen arvon.

Täytettävät kohdat lomakkeessa ovat pakollisia paitsi ostopäivämäärä.

Verkkosivuston lomakkeesta on näkyvillä kuvassa 9.



Add new investment

Name

Amount

Current price

Purchase price

Cash invested

Date of purchase

Add

Kuva 9. Lomake sijoitustietojen täyttämisestä verkkosivustolla.

Sijoituskohteiden muokkaamista varten käytetään myös lomaketta, jonka ulkomuoto mukailee kuvan 9 lomaketta. Lomakkeiden täyttämisessä muutetaan automaattisesti käyttäjän antamat lukumäärät desimaaleiksi.

5.11 Autentikointi

Käyttäjät varmennetaan Firebasen Authentication-palvelun avulla, jota rekisteröitymisessä ja sisäänkirjautumisessa. Käyttäjät varmennetaan sähköpostilla ja salasanalla. Rekisteröinnissä käytettiin Firebasen-funktiota `createUserWithEmailAndPassword`, jonka kautta saatiin myös käyttäjän yksilöivä id (Kuva 10). Edellä mainittua id:tä käytettiin myös Firestore-tietokannassa käyttäjän yksilöimisessä.

```
const auth = getAuth(app);
try {
  await createUserWithEmailAndPassword(auth, email, password).then(
    (response) => {
      const userMetaData = response?.user;
      const userUID = userMetaData?.uid;

      setUIDinvestor(userUID);
      setAccountCreated("Account created");
    }
  );
} catch (error) {
  console.error(error);
}
```

Kuva 10. Käyttäjien rekisteröiminen sovelluksessa.

Sisäänkirjautumisessa käytettiin myös Firebasen-funktiota, joista valittiin sähköpostin kautta mahdollistava kirjautuminen. Ohjelmistossa käytettiin funktiota `signInWithEmailAndPassword` eli sisäänkirjautuminen sähköpostilla ja salasanalla (Kuva 11).

```

try {
  const response = await signInWithEmailAndPassword(auth, email, password);

  const userMetaData = response?.user;
  const userUID = userMetaData?.uid;
  setUserUId(userUID);
  setUIDinvestor(userUID);

  if (userUID !== null) {
    setLogged(true);
    await ReadUserInvestments(userUID, setUserdata); //fetch from firestore user
    await ReadUserPortfolio(userUID, setPortfolioUser, portfolioUser); //fetch fr
  }
} catch (error) {
  console.error(error);
}

```

Kuva 11. Käyttäjien sisäänkirjautuminen sähköpostilla sovelluksessa.

Sisäänkirjautumisen yhteydessä luetaan myös Firestore-tietokannasta käyttäjän dokumentit eli sijoitukset ja tiivistelmä. Sisäänkirjautuminen ja rekisteröityminen tehdään try-catch-ehdolla ja asynkronisesti. Sisäänkirjautuminen odottaa siis vastausta Firebase Authentication-palvelusta, ennen kuin voidaan edetä lukemaan tietoa Firestore-tietokannasta.

Sijoitusportfolio-verkkosivustossa on lomakkeet rekisteröitymiselle ja sisäänkirjautumiseen. Sisäänkirjautumisesta otettu kuvakaappaus on näkyvillä kuvassa 12 ja rekisteröitymisestä kuvassa 13.

Welcome back!

Email

Password

[New user? Create new account](#)

Kuva 12. Sisäänkirjautuminen verkkosivustolla

Uusia käyttäjiä varten tulee klikata tekstiä "New user? Create new account", jonka jälkeen avautuu näkymä uuden tilin rekisteröitymisessä.

Register new account

Email

Password

[Log in to new account, click here](#)

Kuva 13. Rekisteröityminen sovelluksessa.

Uuden tilin rekisteröitymisen jälkeen tulee vielä sisäänkirjautua sovellukseen erikseen. Rekisteröitymisen yhteydessä ei siis automaattisesti samalla sisäänkirjauduta, koska halusin varmistaa, että käyttäjä on kirjoittanut oikean sähköpostiosoitteensa rekisteröitymisen yhteydessä. Sisäänkirjautumisen jälkeen käyttäjä voi tallentaa sijoituskohteista tietoa Firestore-tietokantaan.

5.12 Käyttöliittymän toteuttaminen

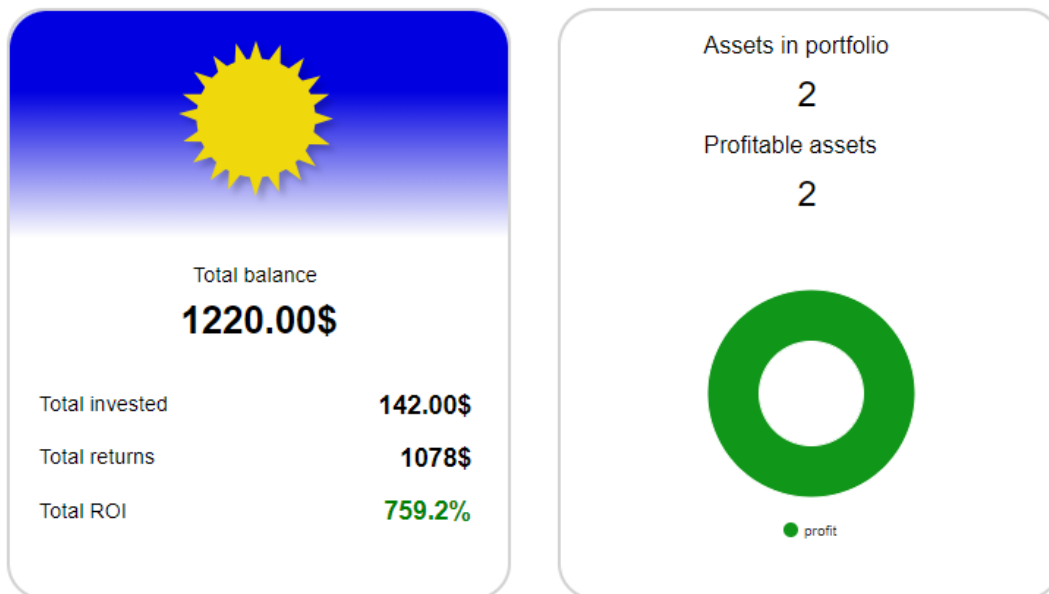
Verkkosivuston käyttöliittymän tein React JavaScript-kirjastolla ja tyylittelyssä Cascading Style Sheets (CSS). Verkkosivustolla sijoituksia varten tein taulukon, jossa on lueteltu kaikki sijoituskohteet. Verkkosivuston etusivulla on näkyvillä myös kaksi tiivistelmää sijoituksista (Kuva 14). Ensimmäisessä tiivistelmistä on lueteltu kokonaismääriä sijoituksista eli tässä tapauksessa arvoista, kokonaiskustannuksista, tuotoista ja sijoitetun pääoman tuotto prosentista (ROI). Verkkosivusto myös näyttää tuoton perusteella joko aurinko-symbolia jos ROI on positiivinen tai lumihietale-symbolia jos ROI on negatiivinen.

Free Portfolio Tracker



Create an account and start tracking investments

register > log in

> manage > view



My Assets

name	price	return	value	amount	
 welcome	42\$	378\$	420\$	10	Edit Delete
 investor	200\$	700\$	800\$	4	Edit Delete

Kuva 14. Verkkosivuston etusivun sijoitusten taulukko sekä tiivistelmät.

Toisessa tiivistelmistä käyttäjä voi nähdä kokonaismäärän kaikista sijoituskohteista lukumääränä. Lisäksi voitollisista ja tappiollisista sijoituskohteista on tehty donitsinmuotoinen muoto, jossa käytetään Googlen React Charts -kirjastoa. Siinä voitolliset sijoitukset näytetään vihreänä ja tappiolliset punaisena.

5.13 Pohdinta

Opinnäytetyön toiminnallisessa osuudessa saatiin tehtyä toimiva verkkosivusto, joka täyttää vaaditut vaatimukset. Sijoittaja voi rekisteröityä sivustolle ja tallentaa sijoituskohteistaan tietoa. Tietokanta toteutettiin Firestore dokumenttipohjaisella tietokannalla, jonka käyttäminen edellyttää sisäänkirjautumisen. Verkkosivustolla varmentaminen on toteutettu Firebasen Authentication-palvelulla ja se on julkaistu Firebase Hosting-palvelulla. Sijoituksia voi lisäämisen ohella myös muokkailla ja poistaa. Toiminnallisessa osuudessa sain toteutettua suunnittelua vastaavan tuotteen. Opin opinnäytetyössä NoSQL-tietokantojen teoriaa ja kuinka niitä voidaan käyttää käytännössä. Aikataulu venyi opinnäytetyötä tehdessä, ja se ei edennyt niin kuin olin suunnitellut. Tulevaisuudessa verkkosivustoa on tarkoitus ylläpitää ja mahdollistaa reaaliaikaisia hintoja sijoituksista. Kirjallisuuskatsaus edesauttoi toiminnallisen osuuden tekemistä, joista erityisesti Firebase palveluiden teoriaosuus.

Lähteet

- Abramova, V., Berdardino, J., Furtado, P. 2014. Experimental Evaluation Of NoSQL Databases. International Journal of Databases Management Systems (IJDMS). Vol.6, No.3.
https://d1wqtxts1xzle7.cloudfront.net/56327684/6314ijdms01-libre.pdf?1523852774=&response-content-disposition=inline%3B+filename%3DEXPERIMENTAL_EVALUATION_OF_NOSQL_DATABASES.pdf&Expires=1683630410&Signature=dJXvla-GQ2G5KeLwo4gvZcbXTWAlaXmJcPxXtOVuwl45qqhCE~Y9ND0~uFYV29HvL4~PddjqWONEhSP9b6ALSDuTaKzPrvKiDFiiFIEq82WuL47FEjKEx83G8147UUPuinHONYBpKp3SDqXFmf48aOCRrbEeTLcdpX-hXxpWf3ojRqXhkVmVky3sgxxHZ7RvDvLzt2708qi~nHjSfrLmiROwSZZdMIGgrdip94r3YPwYYqpyQhfgxnzoUrlPSFHIqpWMCaxuyafUPPvdBT6Cn6kpOrArH4AFPOIW9wXdF8iFvWPJAKJ67cZ7ZZ6GY9gdEGV9Kg8mElbcEcM8nz07BQ_&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA. 9.5.2023.
- Carvalho, I. Sá, F. Bernardino, J. 2023. Performance Evaluation of NoSQL Document Database: Couchbase, CouchDB, and MondoDB.
<https://www.mdpi.com/1999-4893/16/2/78>. 30.8.2023.
- DB-engines. 2023a. <https://db-engines.com/en/ranking>. 17.7.2023.
- DB-engines. 2023b. https://db-engines.com/en/ranking_definition. 17.7.2023.
- Dilling, T. Artificial Intelligence Research: The Utility and Design of a Relational Database System.
<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7718502/>. 27.7.2023.
- Eshtay, M. Sleit, A. Aldwaiti, M. 2019. Implementing Bi-temporal Properties Into Various NoSQL Database Categories. International Journal of Computing. 18(1) 2019, 45-52.
https://www.researchgate.net/publication/332188615_Implementing_bi-temporal_properties_into_various_NoSQL_database_categories. 4.8.2023.
- Gianina, M. 2020. Comparison between Relational and NoSQL Databases.
https://www.researchgate.net/publication/349427122_Comparison_between_Relational_and_NoSQL_Databases. 3.8.2023.
- Google. 2023a. Cloud Firestore. <https://firebase.google.com/docs/firestore>. 21.3.2023.
- Google. 2023b. Firebase Authentication. <https://firebase.google.com/docs/auth>. 21.3.2023.

- Google. 2023c. Firebase Hosting. <https://firebase.google.com/docs/hosting>. 21.3.2023.
- Google. 2023d. Understand Firebase projects. <https://firebase.google.com/docs/projects/learn-more>. 21.3.2023.
- Google. 2023e. Make your app the best it can be. <https://firebase.google.com/>. 15.8.2023.
- Google. 2023f. Firebase pricing plans. <https://firebase.google.com/docs/projects/billing/firebase-pricing-plans?hl=en&authuser=0>. 15.8.2023.
- Google. 2023g. Cloud Firestore Data model. <https://firebase.google.com/docs/firestore/data-model>. 18.8.2023.
- Google. 2023h. Choose a database: Cloud Firestore or Realtime Database. <https://firebase.google.com/docs/firestore/rtdb-vs-firestore>. 21.8.2023.
- Google. 2023i. Structure Your Database. <https://firebase.google.com/docs/database/web/structure-data>. 21.8.2023.
- Google. 2023j. Learn about usage levels, quotas, and pricing for Hosting. <https://firebase.google.com/docs/hosting/usage-quotas-pricing>. 28.8.2023.
- Ohhyver, M. Moniaga, J. Sungkawa, I. Subagyo, B. Chandra, I. 2019. The Comparison Firebase Realtime Database and MySQL Database Performance using Wilcoxon Signed-Rank Test. <https://www.sciencedirect.com/science/article/pii/S1877050919311500>. 21.8.2023.
- MongoDB. 2023. What is NoSQL. <https://www.mongodb.com/nosql-explained>. 8.5.2023.
- Ramza, S. Sarwar, I. Kazmi, R. 2019. Challenges in NoSQL-Based Distributed Data Storage: A Systematic Literature Review. https://www.researchgate.net/publication/332763207_Challenges_in_NoSQL-Based_Distributed_Data_Storage_A_Systematic_Literature_Review. 1.8.2023.
- Sethi, B. Mishra, S. Patnaik, P. A Study of NoSQL Database. International Journal of Engineering Research & Technology. Vol. 3 Issue 4, April 2014. <https://www.ijert.org/research/a-study-of-nosql-database-IJERTV3IS041265.pdf>. 5.9.2023.
- Srivasta, A. Laxmi, V. Singh, P. Pratima, K. Kirti, V. 2022. React JS (Open Source JavaScript Library). IJIRT. Volume 8 Issue 9.

https://ijirt.org/master/publishedpaper/IJIRT153854_PAPER.pdf.
29.8.2022.