LAPIN AMK
Lapland University of Applied Sciences

GOOGLE CLOUD FULL STACK ALUSTANA

Kallas Anssi

Opinnäytetyö

Tieto- ja viestintätekniikka
Insinööri (AMK)

2021

LAPIN AMK
Lapland University of Applied Sciences

Opinnäytetyön tiivistelmä

Tieto- ja viestintätekniikka
Insinööri (AMK)

Tekijä	Anssi Kallas	Vuosi	2021
Ohjaaja	Aku Kesti		
Toimeksiantaja	Paja Design Agency Tmi		
Työn nimi	Google Cloud Full Stack alustana		
Sivumäärä	44		

Opinnäytetyön tavoitteena oli tutustua Google Cloudin tarjoamiin web-palveluihin, Googlen tarjoamiin työkaluihin, luoda Full Stack -esimerkkisovellus ja julkaista se Google Cloud -alustalla. Sovellus luotiin käyttäen moderneja Vue.js ja Express.js JavaScript-sovelluskehyksiä. Sovelluksen visuaalinen tyyli toteutettiin helppokäyttöisen Bootstrap CSS-sovelluskehyksen ja ilmaisten Pexels-kuvapankkikuvien avulla (Pexels 2021). Kehitystyö tehtiin Visual Studio Codella ja Node.js JavaScript- suoritusympäristöllä.

Työssä esitellään Googlen tarjoamia web-kehitykseen soveltuvia palveluja, pääasiassa Google Firebasea ja Google Cloud App Engineä. Tiedot on kerätty käytettyjen palveluiden ja työkalujen dokumentaatioista. Esimerkkisovelluksen tietokantaratkaisut on toteutettu Googlen Firebase-palveluilla ja back end -sovellusta suoritetaan Google Cloud App Engine -sovelluksen palveluna.

Työ on toiminnallinen opinnäytetyö, joka on toteutettu projektina, projektisuunnitelman avulla. Projektissa on hyödynnetty aiempaa osaamista, työharjoittelujaksoilla kertynyttä tietoa Googlen palveluista sekä web-ohjelmoinnin kursseilla opittuja moderneja JavaScript-tekniikoita.

Projektin tuloksena valmistui yksinkertainen Full Stack web-sovellus, joka on julkaistu Googlen pilvipalvelussa. Sovellus toimii pääosin kaikilla moderneilla selaimilla ja myös mobiilissa. Kehityskohteita havaittiin olevan useita, varsinkin mobiilikäyttöön ja latausnopeuksien parantamiseen koodin pilkkomisen avulla.

Avainsanat Cloud, Express.js, Google, JavaScript, Vue.js

LAPIN AMK
Lapland University of Applied Sciences

Abstract of Thesis

Degree Programme in Information
and Communication Technology
Bachelor of Engineering

Author Anssi Kallas Year 2021
Supervisor Aku Kesti
Commissioned by Paja Design Agency Tmi
Subject of thesis Google Cloud as a Full Stack App Platform
Number of pages 44

The aim of this thesis was to examine the suitability of Google Cloud services for Full Stack web development. Also, the aim was to develop a simple modern full stack application using Google Cloud services and to publish the application in Google Cloud.

This project was made as a functional thesis, using a generated project plan as a guide. Google Cloud was previously used with a similar project in practical training, so the acquired knowledge was put to good use. Modern JavaScript techniques were learned in three web development courses. The presented information was gathered from the documentations of the used tools and ser-vices. The project focused on Google Firebase and Google Cloud App Engine. The example app uses Google Firebase as a database engine and the back end is run as a service in Google Cloud App Engine. The application was developed with modern Vue.js and Express.js JavaScript frameworks. The visual style for the application was handled using easy to use Bootstrap CSS frame-work and free stock photos from Pexels (Pexels 2021). The development work was made with Visual Studio Code and Node.js JavaScript runtime.

The study project was implemented using the development and publishing tools provided by Google. The study consisted of creating a simple Full Stack web application example and publishing it on Google Cloud. The result of the project is a simple Full Stack web application, hosted on Google Cloud. The application can be used with most modern web browsers, including mobile versions. It was found out that there were many places for improvements, especially for the mobile version and code splitting for faster download.

Key words Cloud, Express.js, Google, JavaScript, Vue.js

SISÄLLYS

KÄYTETYT LYHENTEET JA TERMIT

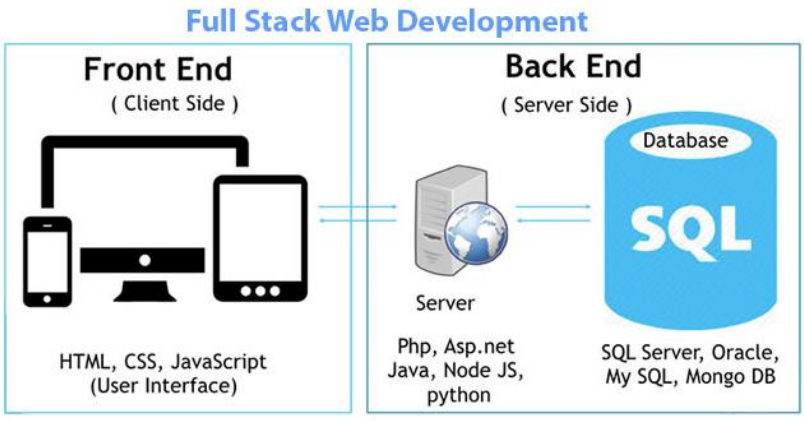
Asynchronous	asynkroninen tiedonsiirto, ei-reaaliaikainen, toisistaan riippumaton tiedonsiirtotapa. Tapahtumasilmukka ei pysähdy (Mozilla 2021a)
CDN	Content Delivery Network, sisällönjakeluverkko, sisältö hajautettu maantieteellisesti. Tiedostot ladataan pienemmällä viiveellä oman alueen palvelimelta (Google Cloud 2021c)
Serverless	pilvipalvelumuoto, laitekapasiteetti tulee automaattisesti käyttöön ja skaalautuu tarpeen mukaan (Google Cloud 2021e)
NPM	Node Package Manager, maailman suurin avoimen lähdekoodin ohjelmistorekisteri (Npmjs 2021)
Front End	sovelluksen käyttöliittymä tai asiakassovellus
Back End	palvelinsovellus, hoitaa yhteydet tietokantaan
Framework	sovelluskehys, sisältää valmiita toimintoja nopeuttamaan kehitystyötä
Hosting (web)	julkaisupalvelin, sisältää web-sovelluksen tiedostot. Jakaa tiedostoja asiakassovelluksille HTTP-protokollan avulla
Webpack	staattinen moduulien paketoija moderneille JavaScript sovelluksille (Webpackjs 2021b)
Full Stack	sovellus, sisältää käyttöliittymän tai asiakassovelluksen ja erillisen palvelin sovelluksen
Grid layout	resoluution mukaan muuntuva HTML-näkymä, jakautuu 12 sarakkeeseen ja kuuteen valmiiseen vaihtopisteeseen (Bootstrap 2021b)
Media query	CSS-ominaisuus grid layoutille. Tutkii selaimen resoluutiota ja vaihtaa automaattisesti elementtien ominaisuuksia
NoSQL	Not Only SQL, ei relaatiomallinen tietokanta

1 JOHDANTO

Web-kehitys on tuttua jo vuosien takaa, joten oli selvää että opinnäytetyönä tehdään jotain web-kehitykseen liittyvää. Työharjoittelujaksoilla perehdyin Google Cloudiin ja Vue.js JavaScript front end -frameworkkiin eli sovelluskehykseen. Yritys halusi siirtyä käyttämään Google Cloudia joten tämä sopi hyvin aihevalinnaksi. Opinnäytetyön tarkoituksena oli tutkia Google Cloud -palveluiden sopivuutta Full Stack web-sovelluksen alustaksi, luoda yksinkertainen Full Stack -sovellus ja julkaista se Google Cloud -alustalle. Projektin runkoa käytetään erään startup-hankkeen pohjana Google Cloud -ympäristössä, joten opinnäytetyö mahdollisti hieman tarkemman tutustumisen kohteeseen.

Projekti keskittyy Google Firebaseen ja Google App Engineen, jotka tarjoavat kaiken tarvittavan pienelle Full Stack -sovellukselle. Työssä esitellään tarvittavien kehitys- ja julkaisutyökalujen asentaminen ja käyttö. Kaikki kehitystyö tehdään Windows 10 -ympäristössä. Google Firebase eroaa hieman perinteisistä ratkaisuista sillä se toimii samaan aikaan hosting-palvelimena, back end -palvelimena, tietokantana ja tallennustilana. Ominaisuudet ovat sisään rakennettuina niin sanottuna serverless-ratkaisuna. (Google Cloud 2021.)

Full Stack -sovelluksesta puhuttaessa tarkoitetaan yksinkertaistettuna sovellusta jossa on front end -asiakassovellus, back end -palvelinsovellus ja database eli tietokanta (Kuvio 1). Kun puhutaan Full Stack -kehittäjistä, niin tarkoitetaan henkilöitä jotka osaavat kehittää molempia päitä, eli front endiä ja back endiä. (W3schools 2021.)



Kuvio 1. Full Stack Web-sovellus (Tech Altum 2021)

Suosittuja Full Stack -kokonaisuuksia ovat esimerkiksi:

- LAMP: JavaScript - Linux - Apache - MySQL - PHP
- LEMP: JavaScript - Linux - Nginx - MySQL - PHP
- MEAN: JavaScript - MongoDB - Express - AngularJS - Node.js
- Django: JavaScript - Python - Django - MySQL
- Ruby on Rails: JavaScript - Ruby - SQLite – Rails (W3schools 2021).

Projektissa kehitettävä sovellus tehdään nykyaikaisia Vue.js ja Express.js JavaScript -sovelluskehyksiä ja Bootstrap CSS-sovelluskehystä hyödyntäen. Käyttöliittymää ei yhdistetä perinteisesti erilliseen palvelinsovellukseen, vaan suoraan moderniin Google Firebase Serverless -pilvipalveluun. Tiedonsiirto Firebasen ja front endin välillä tapahtuu asynkronisesti. Tietokanta on esimerkeistä poiketen nykyaikaisempi NoSQL.

Front end luodaan Vue.js JavaScript -sovelluskehyksellä ja Bootstrap CSS-sovelluskehyksellä. Admin back end toteutetaan Express.js-sovelluskehyksellä. Admin sovellus mahdollistaa front end -sovelluksen käyttäjien ja tietokannan hallinnan omalta admin-käyttöliittymältä.

2 GOOGLE CLOUDIN ESITTELY

Google Cloud tarjoaa laajan valikoiman palveluja web-sovellusten kehittämiseen, ylläpitoon ja analysointiin. Jos et halua ylläpitää omaa laitteistoa, tai ostaa palveluja monesta eri paikasta, niin Google Cloudin serverless ratkaisut ovat hyvä vaihtoehto. Google saatetaan helposti mieltää enemmänkin isojen yritysten palveluntarjoajaksi, mutta esimerkiksi Firebase sopii oikein hyvin harrastelijoiden, pienempien projektien ja yritystenkin alustaksi.

Projektin voi aloittaa ilmaisella Spark Planilla (Kuviossa 2 vasemmalla) jossa käytetään kuukausittaisia käyttörajoja ja kun liikennemäärät kasvavat, niin voit siirtyä Blaze Planin käyttöön maksamaan palveluiden käyttömäärien perusteella. Firebasen ilmaisten queryjen raja on 1 500 000 kappaletta kuukaudessa, joka riittää jo melko pitkälle. Firebasen dokumentaatiosta löytyy kätevä Blaze Plan -laskuri, jolla voit hahmotella projektin hintaluokkaa käyttömäärien perusteella (Google Firebase 2021a.)

Cloud Firestore		
Stored data	1 GiB total	$0.18/GiB
Network egress	10GiB/month	Google Cloud pricing
Document writes	20K/day	$0.18/100K
Document reads	50K/day	$0.06/100K
Document deletes	20K/day	$0.02/100K

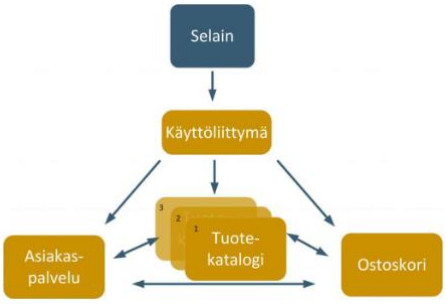
Kuvio 2. Esimerkki Google Firebasen hinnoittelusta (Google Firebase 2021a)

Raportissa keskitytään pääasiassa App Engineen ja Firebaseen, koska tässä vaiheessa projektissa ei tarvita vielä muita palveluja. App Enginen Node.js ympäristössä suoritetaan admin back end palvelin sovellusta ja Firestore huolehtii front end clientin hostauksesta, käyttäjien autentikaatiosta ja tietokannoista.

2.1 Google App Engine

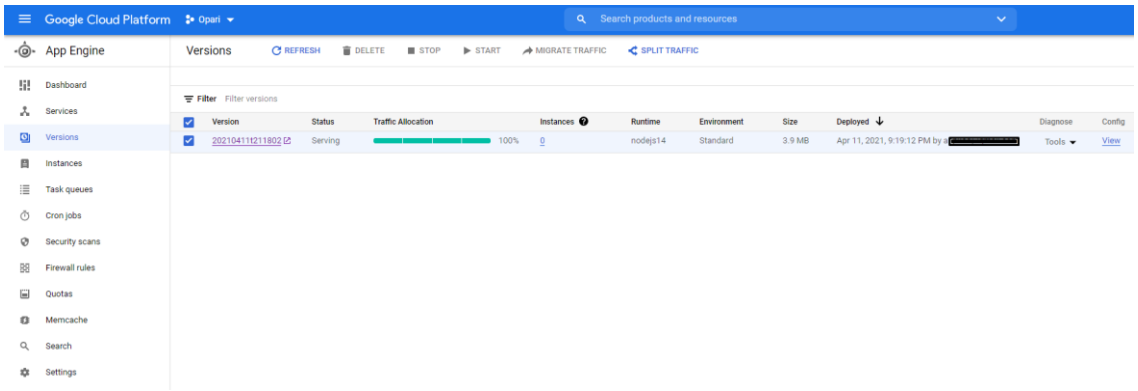
App Engine sovellus koostuu yhdestä tai useammasta servicestä eli palvelusta. Jokaiselle palvelulle voidaan määrittää oma suoritusympäristö ja suorituskykyasetukset. Palveluista julkaistaan versioita, joita voidaan ajaa rajoitettu määrä

omissa instansseissaan, konfiguroidusta liikennemäärästä riippuen. (Google Cloud 2021b.) Tässä noudatetaan mikropalveluja vastaavaa rakennetta, jolloin voidaan skaalata ainoastaan tarvittavat osa-alueet (Kuvio 3). (Wallenius Consulting 2021.)



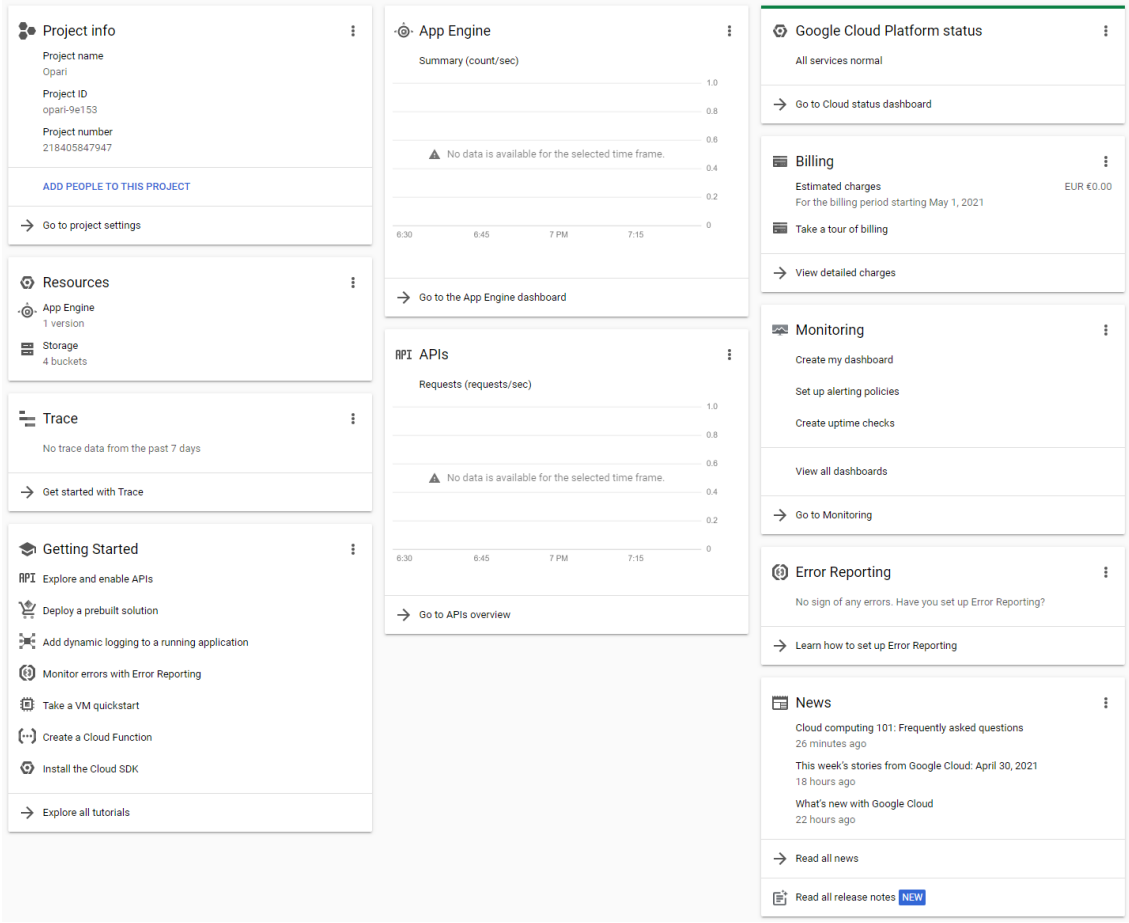
Kuvio 3. Tarvittavan mikropalvelun skaalaaminen (Wallenius Consulting 2021)

Admin back endiä ei välttämättä tarvita, mutta se nopeuttaa huomattavasti järjestelmän ylläpitoa. Taskit voi suorittaa myös Firebasen konsolista, mutta se on hitaampaa ja työläämpää. Tässä olisi yksi mahdollisuus käyttää Googlen tarjoamia pilvifunktioita, joilla voisi järjestellä uutta tietoa valmiiksi admin-paneelia varten josta admin sitten hyväksyy tai poistaa tiedot. Kuviossa 4 esitellään admin sovelluksen palvelu jota suoritetaan App Enginellä (Kuvio 4). Palvelusta on suoritettavana ainoastaan yksi versio, koska käyttö on vielä vähäistä.



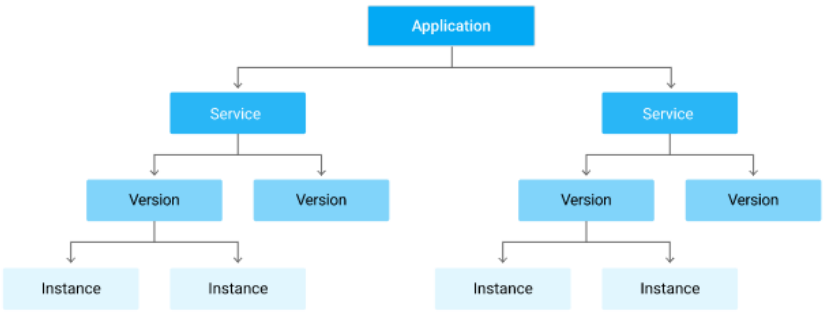
Kuvio 4. Google Cloud App Enginen admin-palvelu

Kuviossa 5 näkyy pilvialustan konsolin päänäkymä josta projektia on helppo hallita (Kuvio 5). Konsoli näyttää kaikki tärkeimmät tiedot ja käyttäjä voi vapaasti muokata konsolin ulkonäköä ja näytettäviä ominaisuuksia. Kortit toimivat myös linkkeinä omiin osioihinsa. Kuvion 5 näkymä on Googlen oletusnäkymä.



Kuvio 5. Google Cloud konsolin päänäkymä

App Engine -sovelluksen kaikki resurssit luodaan projektia luodessa valittuun samaan maantieteelliseen sijaintiin. (Google Cloud 2021b.) Projektissa käytetty admin-sovellus käyttää Node.js-suoritusympäristöä joka on esitelty tarkemmin luvussa 4.1. Eri instansseissa suoritettavalla admin-sovelluksella voidaan esimerkiksi erottaa käyttäjien ja tietokannan hallinta omiin kokonaisuuksiinsa. Kuten kuviosta 6 käy ilmi, niin ilmaisellakin sovelluksella voidaan rakentaa jopa 15 instanssin järjestelmä (Kuvio 6).



Kuvio 6. Google Cloud App Enginen palveluiden hierarkia (Google Cloud 2021b)

Taulukossa 1 on esitelty palveluiden ja niiden versioiden rajoitukset (Taulukko 1). Jos ilmaiset rajat eivät riitä niin sovellusta voidaan pilkkoa entistä pienempiin osiin, jolloin myös skaalaaminen on toteutettavissa paljon tarkemmin. Kun käytössä on useita versioita, niin päivitykset ja takaisin siirtymiset onnistuvat jouheasti ja liikennettä voidaan ohjata haluttuihin versioihin. (Google Cloud 2021b.)

Taulukko 1 Google Cloud App Enginen palveluiden rajoitukset (Google Cloud 2021b)

Limits

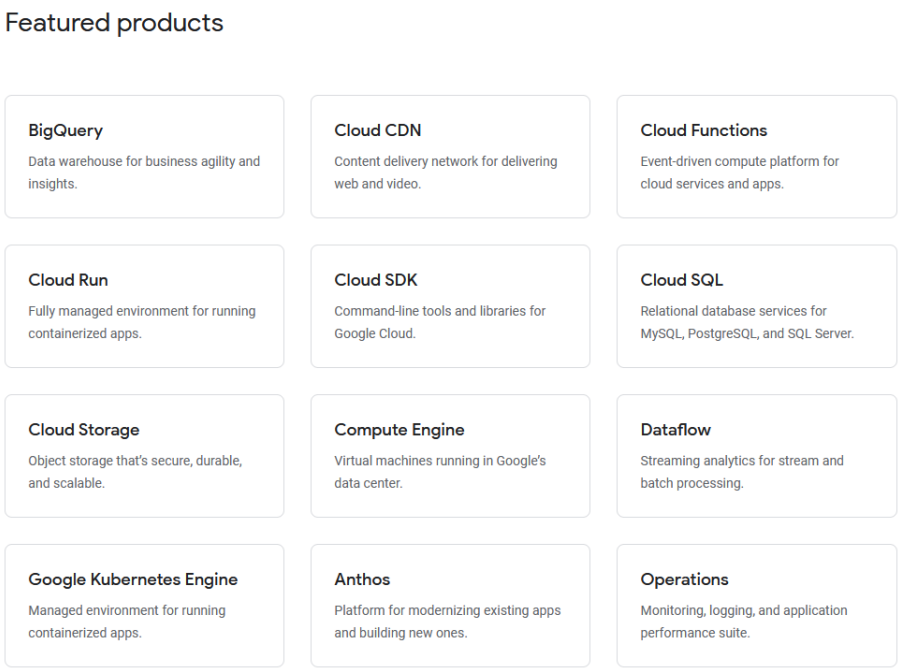
The maximum number of services and versions that you can deploy depends on your app's pricing:

Limit	Free app	Paid app
Maximum services per app	5	105
Maximum versions per app	15	210

There is also a limit to the number of instances for each service with basic or manual scaling:

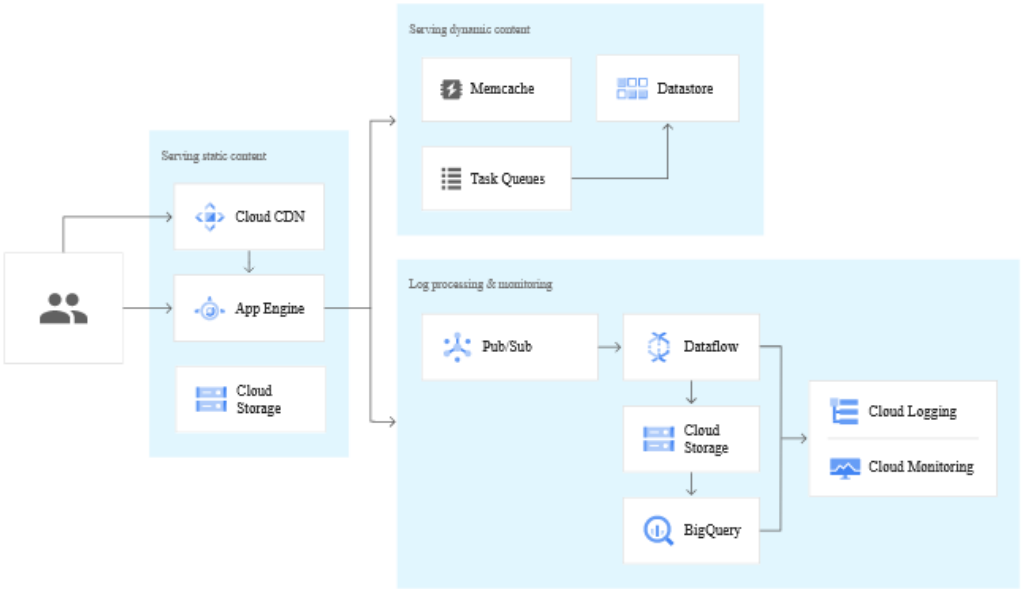
Maximum instances per manual/basic scaling version		
Free app	Paid app US	Paid app EU
20	25 (200 for `us-central`)	25

Kuviossa 8 on esitelty pilvialustan suosittuja palveluja, jotka ovat kyllä suunnattu enemmän isojen yritysten tarpeisiin (Kuvio 8). Koko yrityksen infrastruktuuri on mahdollista rakentaa Googlen pilvipalveluiden avulla, mutta tällöin yrityksen tuloksen täytyy olla kunnossa. Valikoimasta löytyy lähes kaikki mahdolliset palvelut valmiina ja jos halutaan jotain spesifisempää, niin omia toteutuksia voi suorittaa virtuaalikoneilla.



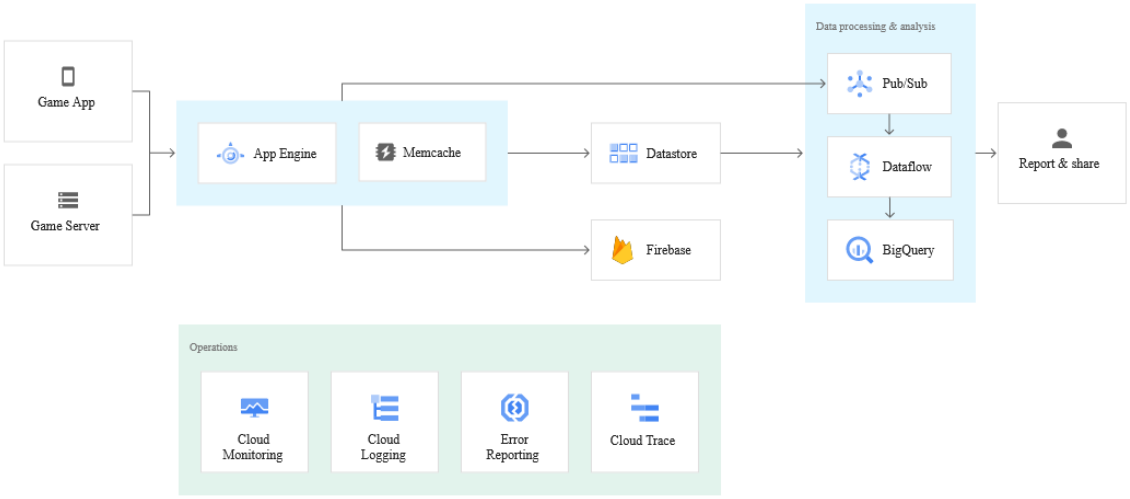
Kuvio 7. Google Cloudin suosittuja palveluja (Google Cloud 2021f)

Kuviossa 9 on Googlen luoma esimerkkiarkkitehtuuri modernista web-sovelluksesta ja sen hyödyntämistä palveluista (Kuvio 9). App Engine tukee myös automaattista skaalausta, yllättävien liikennepiikkien ilmaantuessa. Käyttäjän ei tarvitse itse konfiguroida palvelimia. (Google Cloud 2021a.)



Kuvio 8. Googlen esimerkki modernista web-sovelluksesta (Google Cloud 2021a)

Kuviossa 10 on Googlen luoma esimerkki skaalautuvasta mobiili back endistä ja sen hyödyntämistä komponenteista (Kuvio 10). Firebase toimii erinomaisesti myös Android-sovelluksen back endinä, johon esimerkiksi käyttäjien autentikoinnin lisäys onnistuu aloittelijaltakin jopa minuuteissa. Enemmän ongelmia tuottaa varmasti Android Studion kankea käytettävyys.



Kuvio 9. Googlen esimerkki skaalautuvasta mobiili back endistä (Google Cloud 2021a)

2.2 Google Firebase

Google Firebasea voi käyttää serverless-ratkaisuna ilman erillistä back end -palvelinta. Queryt tehdään suoraan firebaseen jossa on omat tietokannat sovelluksen datalle ja käyttäjille. Firebasesta löytyy myös tallennusosio tiedostoille, sekä hosting palvelut. Firebase sopii erittäin hyvin nopeiden prototyyppien luontiin. Sovelluksen ja back endin saa tarvittaessa pystyyn jopa minuuteissa. Firebase liitetään JavaScript-sovellukseen Firebasen määrittämien asetusten avulla (Kuvio 11). *InitializeApp* komento aktivoi Firebasen sovelluksessa.

```
// For Firebase JS SDK v7.20.0 and later, measurementId is optional
const firebaseConfig = {
  apiKey: "[redacted]",
  authDomain: "opariv1-f8230.firebaseapp.com",
  projectId: "opariv1-f8230",
  storageBucket: "[redacted]",
  messagingSenderId: "[redacted]",
  appId: "[redacted]"
};

// Initialize Firebase
firebase.default.initializeApp(firebaseConfig)
```

Kuvio 10. Firebasen asetukset Vue.js sovelluksessa

Kuviossa 12 näkyy Firebasen liittäminen Vue.js-sovelluksen komponenttiin. *Import* ja *require* komennot tarvitaan jokaiseen komponenttiin, jossa Firebasea halutaan käyttää. Tämä on tietysti vain yksi tapa liittää Firebase front end -sovellukseen.

```
<script>
import firebase from 'firebase/app'

// Required for side-effects
require("firebase/firestore")
```

Kuvio 11. Firebasen liittäminen Vue.js-komponenttiin

Firebase-muuttujaa voidaan tämän jälkeen käyttää funktioissa. Esimerkkinä on funktio, joka hakee Firebasesta valitun kahvilan tietoja (Kuvio 13). Google on juuri päivittänyt Firebase funktiot käyttämään aitoa *async await* -toiminnallisuutta.

```
methods: {
    async getCafeData() {
        const db = firebase.firestore()
        const citiesRef = db.collection('cafes');
        const snapshot = await citiesRef.where('name', '==', this.cafe).get();
        if (snapshot.empty) {
        console.log('No matching documents.');
        return;
        }
```

Kuvio 12. Asynkroninen Firebase funktio kahvilan tiedoille

Jokainen käytettävä ominaisuus tulee importoida komponenttiin. Kuviossa 14 otetaan käyttöön käyttäjien autentikointi (kuvio 14). Firebasen tietokantatoiminto pitää tuoda komponentille erikseen, kuten aiemmin kerrottiin.

```
<script>
import firebase from 'firebase/app'
import 'firebase/auth'
// Required for side-effects
require("firebase/firestore")
```

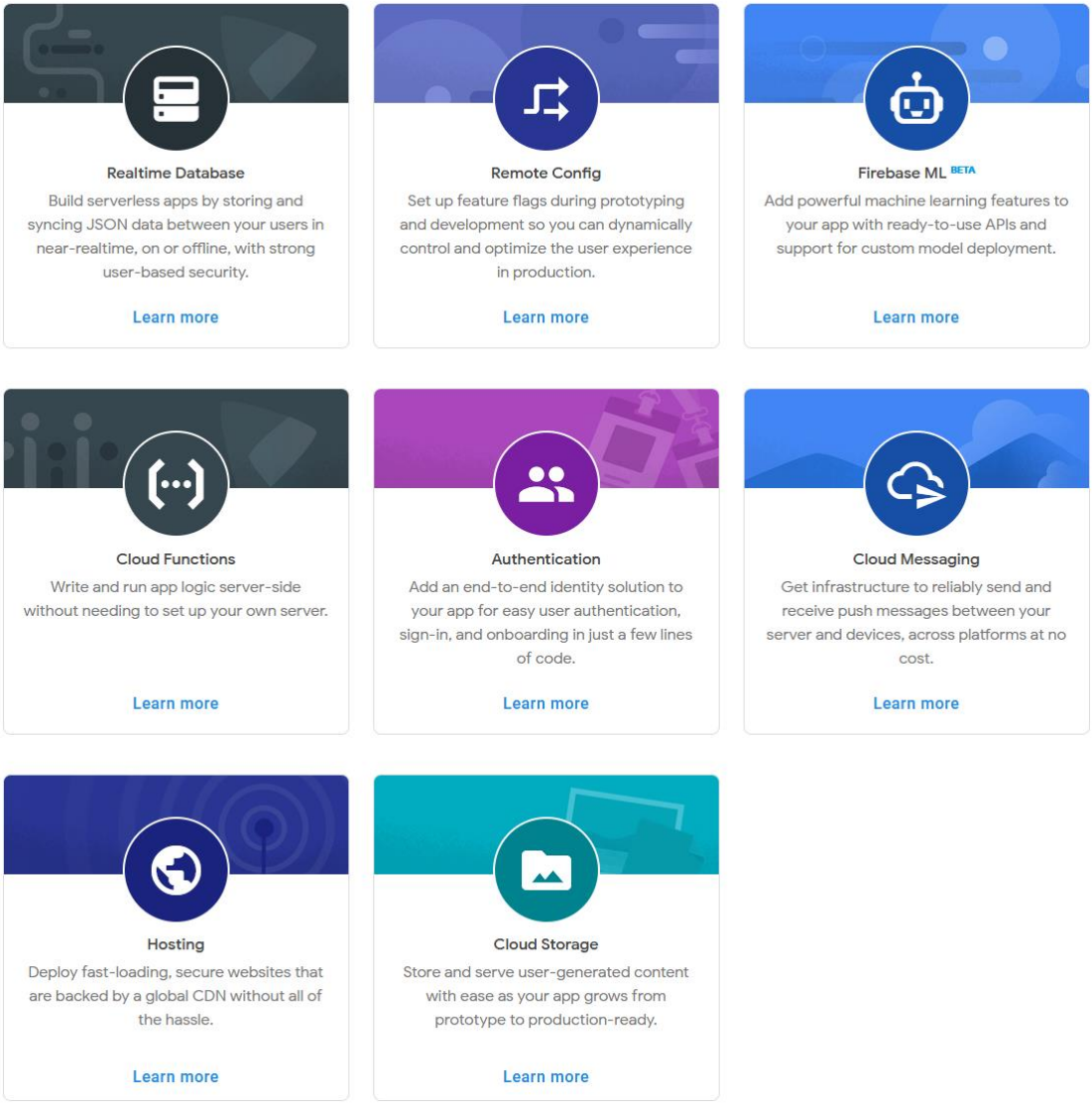
Kuvio 13. Firebasen autentikaation importointi

Importoinnin jälkeen *firebase.auth()* -toiminnot ovat käytettävissä komponenteissa (Kuvio 15). Autentikaatio tarjoaa kaikki perustoiminnot ja kehittäjä voi halutessaan käyttää myös Googlen rakentamaa autentikointikäyttöliittymää johon voidaan yhdistää kirjautumispalvelut muun muassa Googlelta, Facebookilta, Twitteriltä, Githubilta ja Microsoftilta.

```
login() {
  if (this.email != "" && this.password != "") {
    firebase.auth().signInWithEmailAndPassword(this.email, this.password)
    .then((userCredential) => {
        // Signed in
        var user = userCredential.user;
        console.log(user)
        this.$router.push('/user')
        // ...
    })
```

Kuvio 14. Firebasen autentikaation käyttäminen

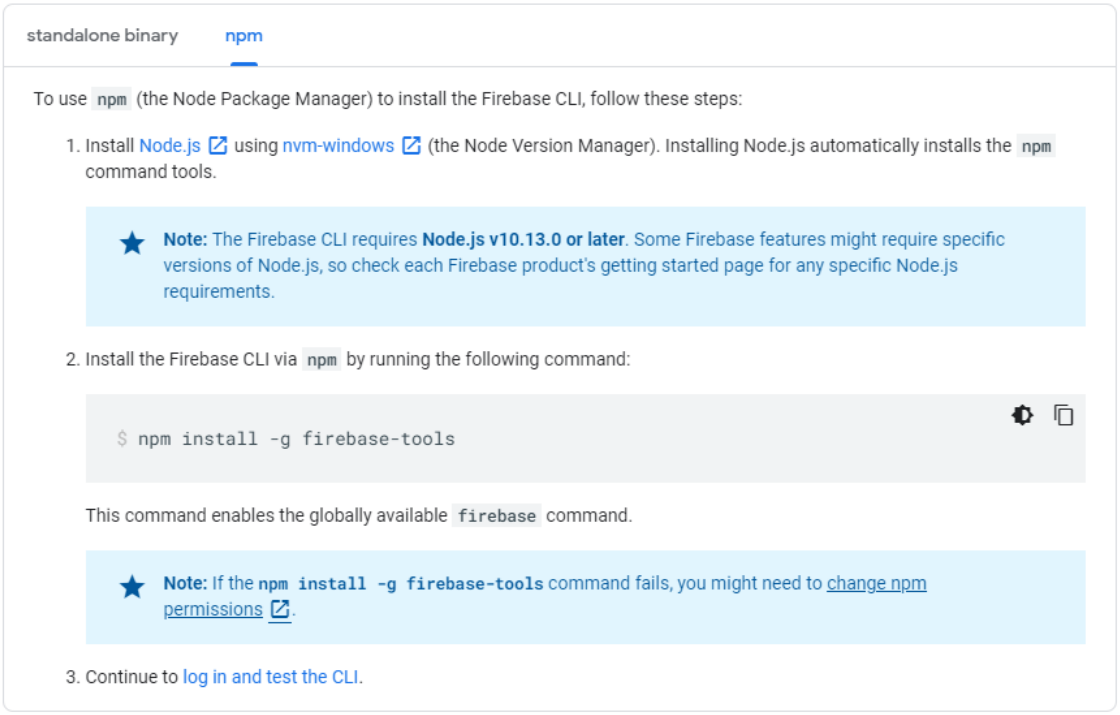
Kuviossa 16 esitellään suosittuja Firebase palveluja (Kuvio 16). Firebase tarjoaa myös koneoppimista ja push-viestien lähettämistä. Push-viestit ovat tärkeä lisä nykyaikaiseen front end -sovellukseen, oli se sitten webissä tai Androidissa. (Google Firebase 2021d.)



Kuvio 15. Suosittuja Firebase-palveluja (Google Firebase 2021d)

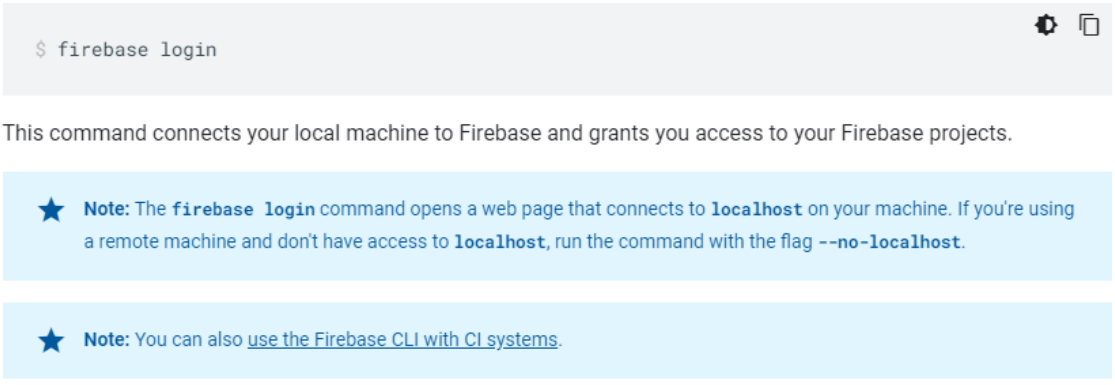
2.3 Google Firebase CLI

Firebasen oma komentoliittymä löytyy Windows-, Linux- ja macOS-versioina. Windowsille löytyy valmis asennuspaketti tai vaihtoehtoisesti voidaan käyttää noden pakettien hallintaa (Kuvio 17). NPM-asennus on suositeltua, mutta silloin täytyy muistaa koodin pilkkominen, jotta suorituskyky säilyy hyvänä. (Google Cloud 2021c.)



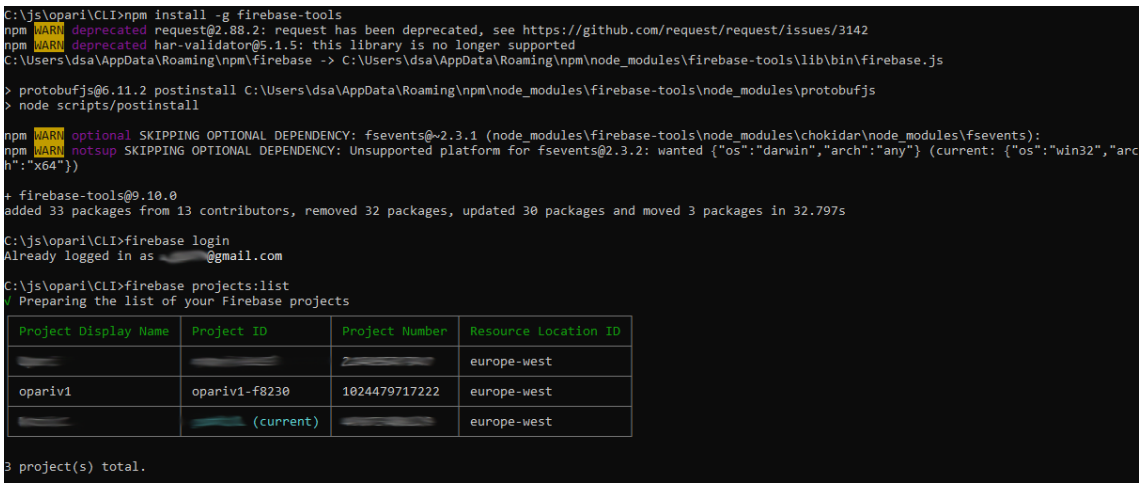
Kuvio 16. Firebase CLI:n npm-asennus (Google Firebase 2021c)

Kun CLI on asentunut, voidaan antaa komento *firebase login*. Tämä ohjaa käyttäjän selaimeen kirjautumaan Google Firebaseen (Kuvio 18). Kehityskoneessa on hyvä olla valmius internetyhteydelle jos halutaan julkaista jotain.



Kuvio 17. Firebase CLI:n autentikointi (Google Firebase 2021c)

Kun käyttäjä on kirjautuneena, niin voidaan testata järjestelmää antamalla komento *firebase projects:list*, joka listaa käyttäjän Firebase projektit (Kuvio 19). Käyttäjällä voi olla useita projekteja joiden välillä voidaan vaihdella.

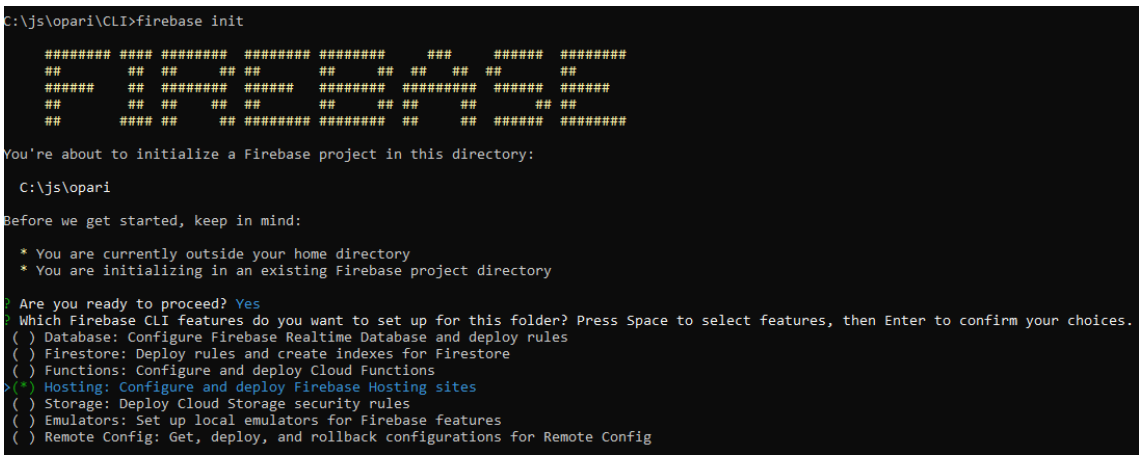


Kuvio 18. Firebase CLI:n npm asennus ja testaus

Kun projekti on siinä vaiheessa, että halutaan julkaista jotain niin CLI-kansio pitää konfiguroida ensin. Komento on *firebase init* (Kuvio 20). *Firebase init*-komento luo *firebase.json* tiedoston valittuun kansioon. Tiedosto sisältää Firebasen tarvitsemat tiedot.

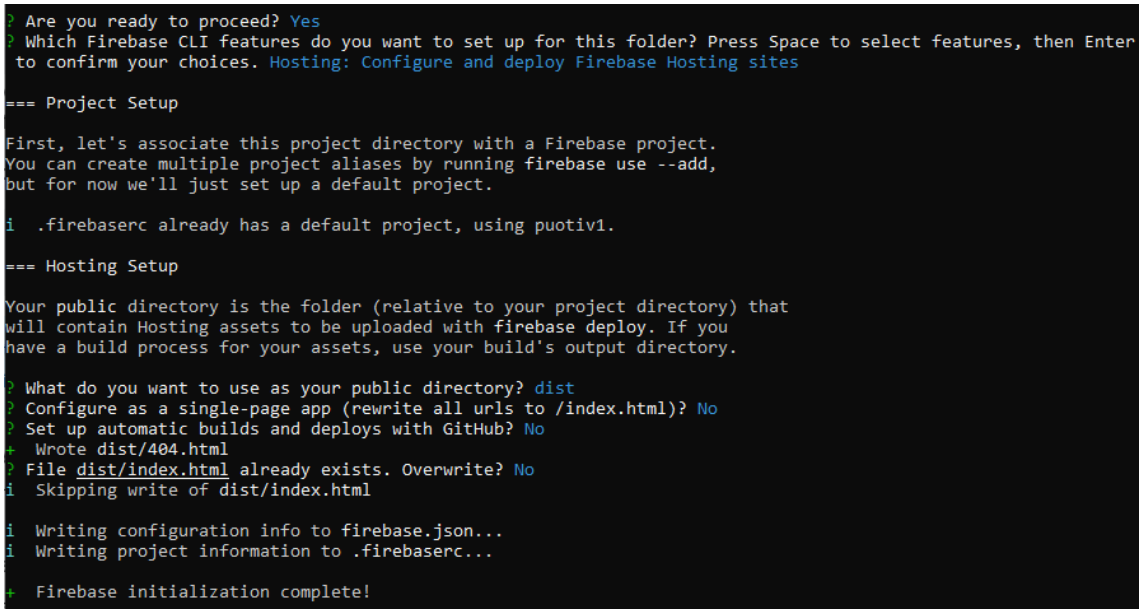
Kuvio 19. Firebase CLI:n projektin konfigurointi (Google Firebase 2021c)

Tässä vaiheessa ei tarvita muuta kuin hosting-palvelu, joten valitaan se konfiguroitavaksi (Kuvio 21). Konfiguraatioita tutkiessa on tärkeää muistaa, ettei talleta vahingossa vääriä asetuksia, jotka eivät löydy projektista. Tällä saa helposti estettyä esimerkiksi julkaisun.



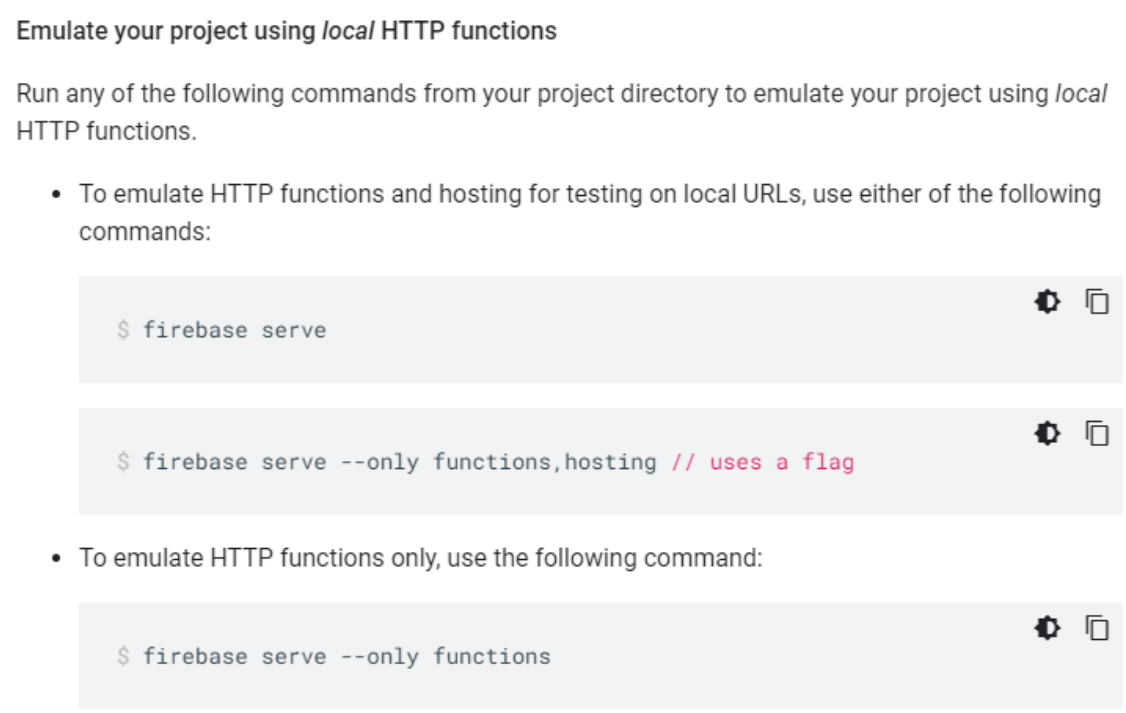
Kuvio 20. Firebase CLI:n hosting palvelun konfigurointi

Vue.js tekee valmiiksi *dist*-nimisen kansion kun projekti käännetään julkaisupaketiksi komennolla *npm run build*. Firebase CLI voidaan nyt konfiguroida lataamaan *dist*-kansion sisältö Firebasen hosting-palvelimelle (Kuvio 22).



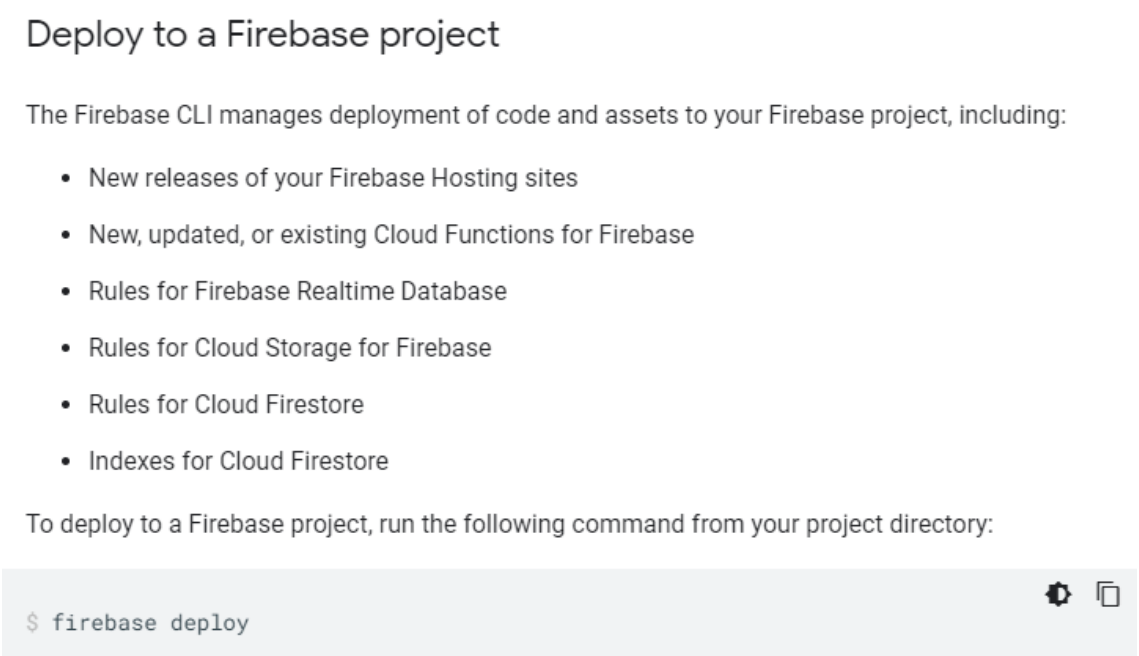
Kuvio 21. Firebase CLI:n julkaisu konfiguraatio

Firebase CLI:llä voidaan myös suorittaa kehityspalvelinta komennolla *firebase serve*. Tätä voi kokeilla jos jostain syystä ei voida käyttää Vue CLI:n kehityspalvelinta. Moottorina toimii sama Node.js-suoritusympäristö kuin Vue CLI:ssä (Kuvio 23).



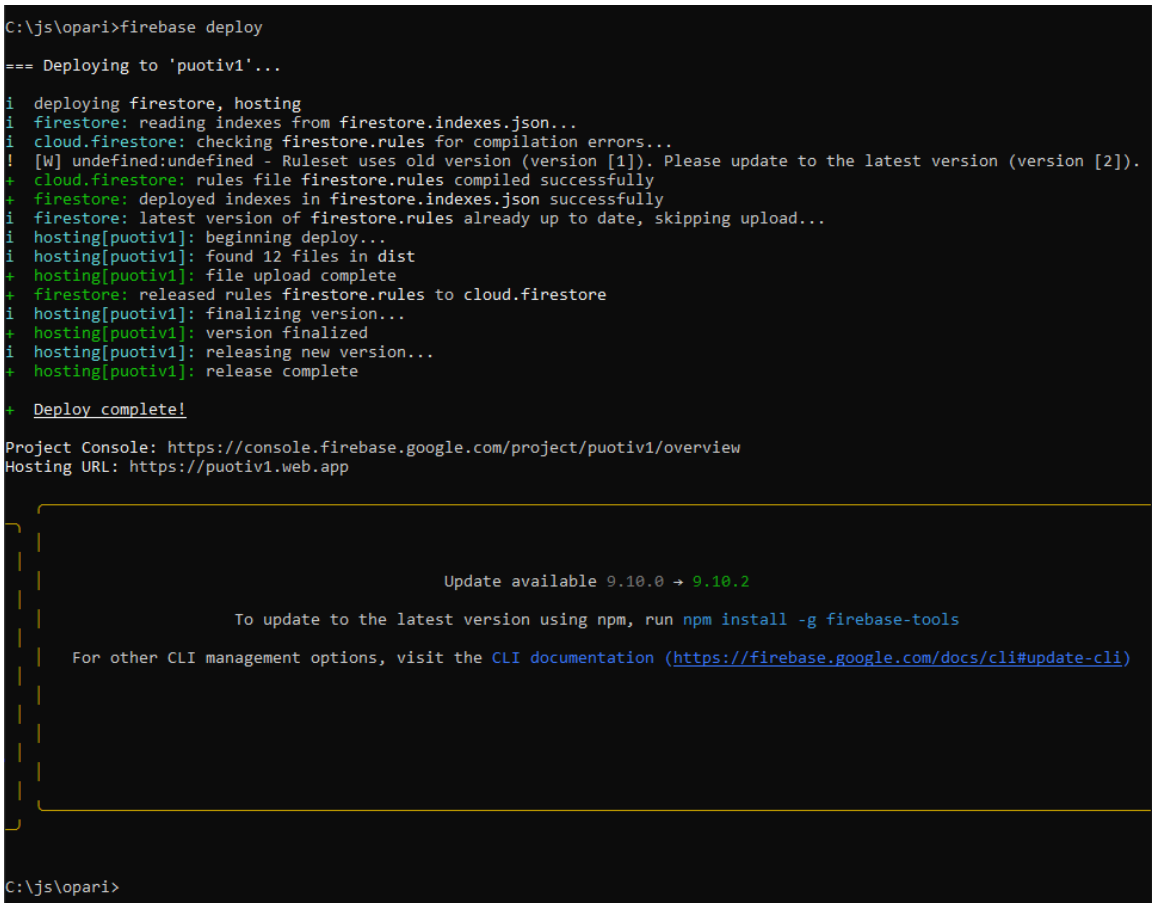
Kuvio 22. Firebase CLI:n kehityspalvelin (Google Firebase 2021c)

Projekti julkaistaan Firebaseen komennolla *firebase deploy*. Jos komennon kanssa on ongelmia, niin ensimmäisenä kannattaa tarkistaa *firebase.json*-tiedoston sisältö joka luodaan kohdassa *firebase init*. Jos tiedostossa on esimerkiksi kokeilujen takia konfiguroituna vääriä palveluja, ne saattavat estää julkaisun (Kuvio 24).



Kuvio 23. Firebase CLI:n projektin julkaisukomento (Google Firebase 2021c)

Deploy-komento lataa *dist*-kansion sisällön Firebase hosting-palvelimelle, jonka jälkeen sovellus on heti käyttövalmis ilmoitetussa osoitteessa. Kuviossa 20 näkyy Firebasen julkaisuprosessi. Tässä tapauksessa julkaisupalvelin on konfiguroitu eri projektille, mutta prosessi on identtinen (Kuvio 25).



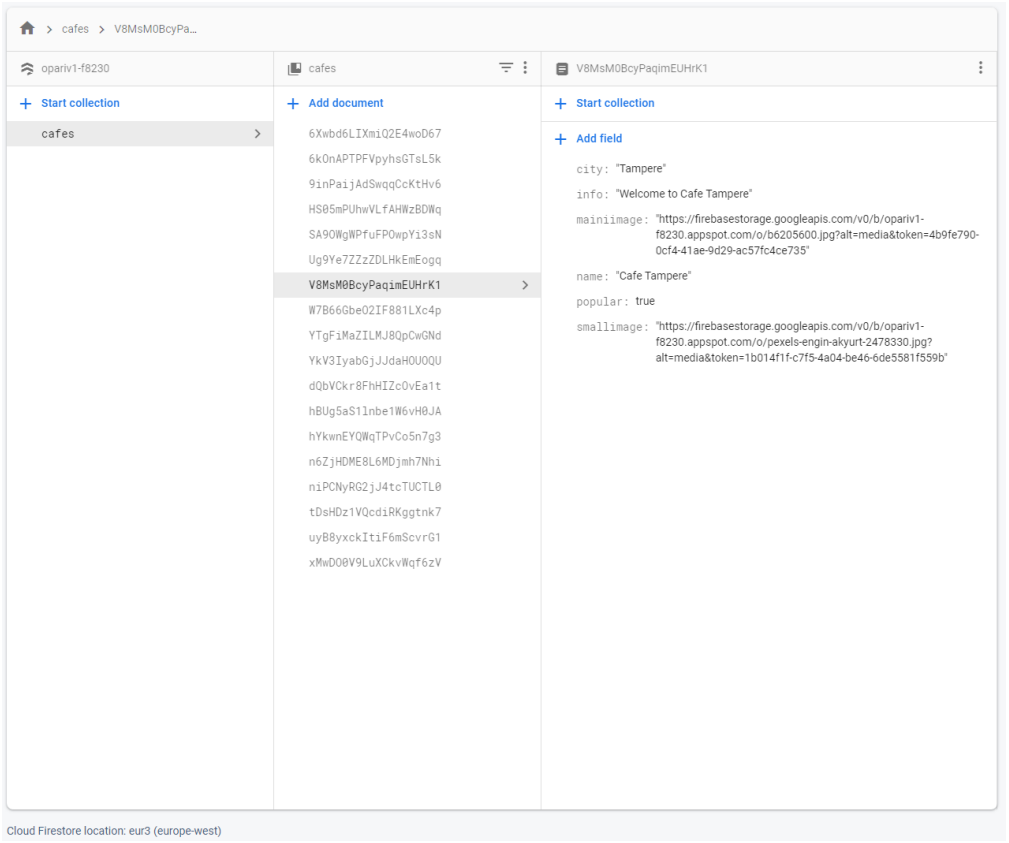
Kuvio 24. Firebase CLI:n valmis julkaisu

Kun Firebase CLI on kerran konfiguroitu, niin muutosten ja päivitysten julkaiseminen on todella helppoa ja nopeaa. Kun käytössä on Vue.js, tarvitaan ainoastaan kaksi komentoa *npm run build*, jonka jälkeen *firebase deploy* ja muutokset ovat heti nähtävillä verkko-osoitteessa.

2.4 Google Firestore Database

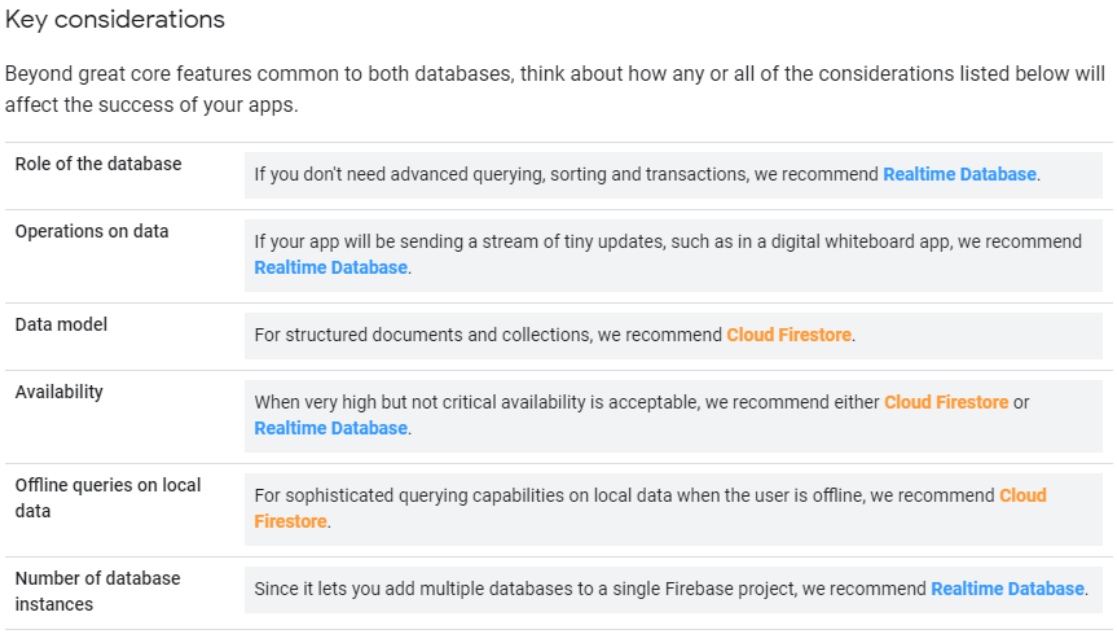
Valittavana on kaksi eri tietokantaversiota, Cloud Firestore ja Realtime Database. Tämän opinnäytetyön sovelluksessa käytetään kevyempää Firestorea, joka

riittää kevyessä käytössä oikein hyvin ja viiveet Keski-Euroopasta ovat siedettävällä tasolla. Projektin sovelluksia suoritetaan Saksassa sijaitsevassa datakeskuksessa. Kuviossa 21 näkyy projektin Firestore-tietokanta. Kokoelmien luominen on helppoa ja nopeaa Firestoren konsolista (Kuvio 26). (Google Firebase 2021b.)



Kuvio 25. Google Firestoren tietokanta

Kuviossa 27 esitellään Firebasen dokumentaatiosta löytyvä kätevä työkalu sopivan tietokannan valintaan. Käyttäjälle esitetään perustason kysymyksiä tietokannan käytöstä jonka perusteella näytetään käyttötapaukseen paremmin soveltuva tietokantavaihtoehto (Kuvio 27). (Google Firebase 2021b.)



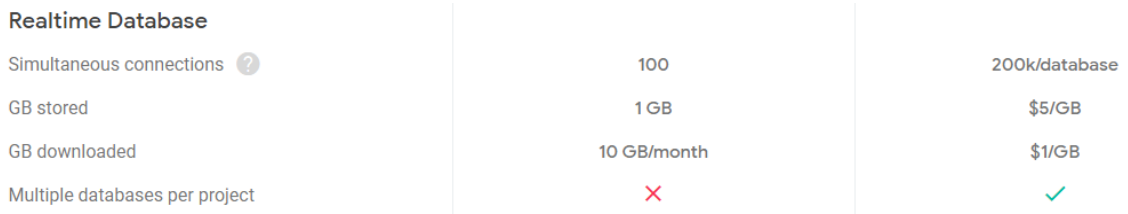
Key considerations

Beyond great core features common to both databases, think about how any or all of the considerations listed below will affect the success of your apps.

Role of the database	If you don't need advanced querying, sorting and transactions, we recommend **Realtime Database**.
Operations on data	If your app will be sending a stream of tiny updates, such as in a digital whiteboard app, we recommend **Realtime Database**.
Data model	For structured documents and collections, we recommend **Cloud Firestore**.
Availability	When very high but not critical availability is acceptable, we recommend either **Cloud Firestore** or **Realtime Database**.
Offline queries on local data	For sophisticated querying capabilities on local data when the user is offline, we recommend **Cloud Firestore**.
Number of database instances	Since it lets you add multiple databases to a single Firebase project, we recommend **Realtime Database**.

Kuvio 26. Google Firebasen tietokantojen vertailu (Google Firebase 2021b)

Myös Realtime Databasea on mahdollista käyttää ilmaisella Spark Planilla. Kuviossa 28 esitellään Google Firebase Realtime Databasen ilmaisen käytön rajat sekä maksullisen "*pay as you go*" (Google Firebase) Blaze Planin hinnat (Kuvio 28). Tässä projektissa ei ole vielä ollut tarvetta Firestorea nopeammalle tietokannalle, mutta tällekin varmasti löytyisi muutamia käyttötapauksia joissa se olisi parempi. (Google Firebase 2021a.)



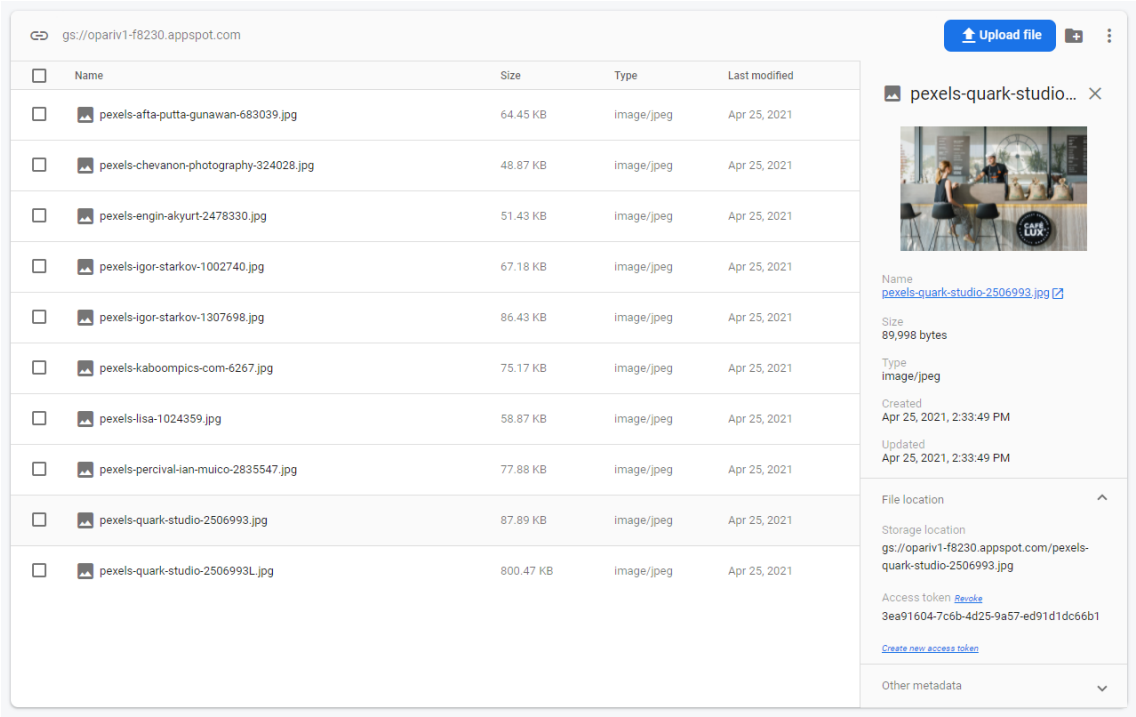
Realtime Database		
Simultaneous connections	100	200k/database
GB stored	1 GB	$5/GB
GB downloaded	10 GB/month	$1/GB
Multiple databases per project	✕	✓

Kuvio 27. Google Firebasen Realtime Database rajoitukset (Google Firebase 2021a)

2.5 Google Firebase Storage

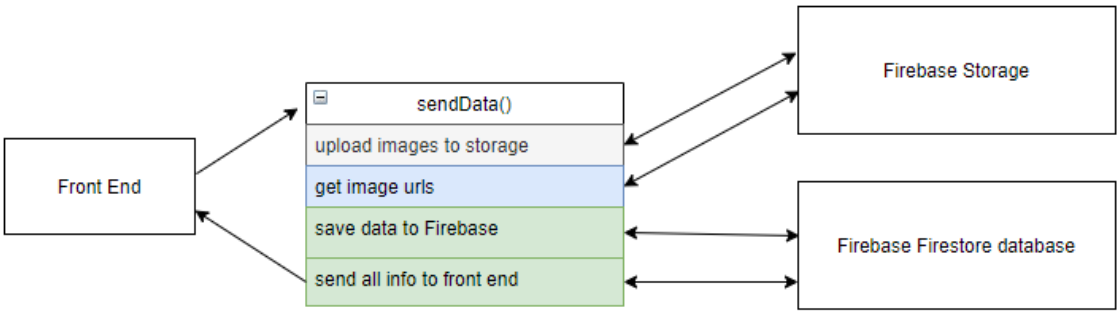
Google Firebasesta löytyy myös tallennusoptio erilaisille tiedostoille (Kuvio 29). Tässä projektissa Firebase Storage on käytössä kuvien tallennukseen. Kuvien latauslinkit on tallennuksen jälkeen lisätty tietokantaan. Sovelluksessa ei ole vielä

valmista työkalua ilmoituksen jättämiseen. Ilmoituksessa ladattaisiin ensin kuvat storageen, minkä jälkeen kuvien latauslinkit lisättäisiin tietokantaan muiden tietojen kanssa.



Kuvio 28. Google Firebase Storagen kuvien tallennus

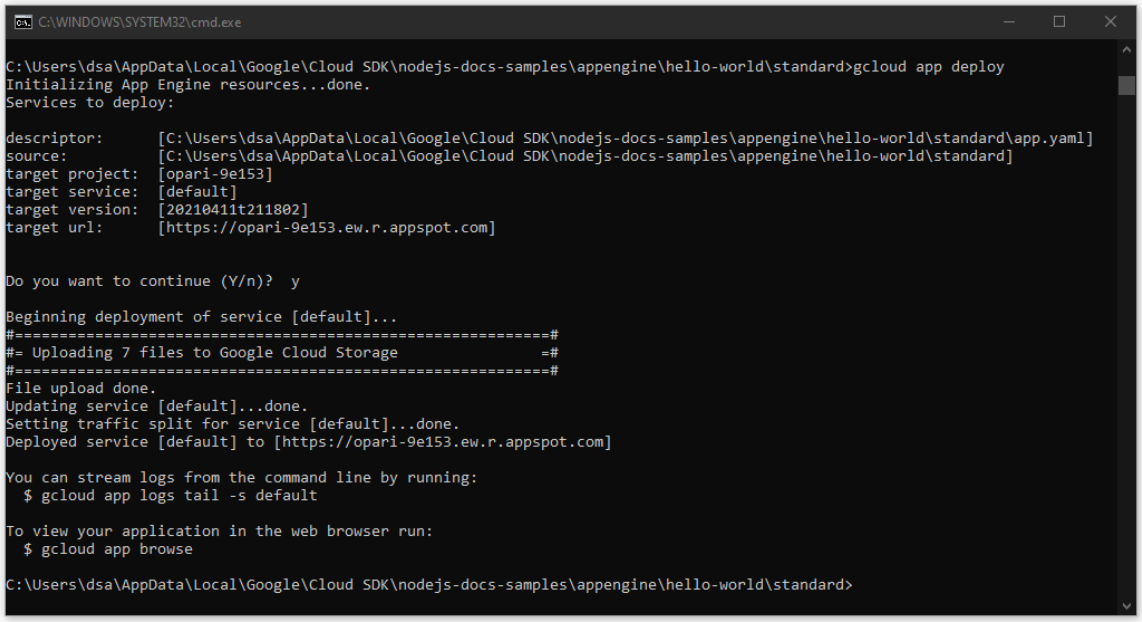
Kuviossa 30 on esimerkkikaavio funktiosta, jolla voidaan jättää ilmoitus kuvien kanssa (Kuvio 30). Toiminta on ehkä hieman kömpelöä, mutta kun käytössä on asynkroninen tiedonsiirto, niin prosessi suoritetaan taustalla jolloin käyttäjän ei tarvitse tehdä asioita erikseen.



Kuvio 29. Funktion toiminta kuvitteellisessa ilmoituksessa

2.6 Google Cloud SDK

Jotta voidaan käyttää Google Cloud App Engineä, tarvitaan myös Google Cloud SDK. Windowsille löytyy valmis asennuspaketti ja asennus on yksinkertainen, mutta vie jonkin verran aikaa. Asennuksen jälkeen voidaan kloonata Googlen esimerkit Githubista tai ruveta työstämään heti omaa sovellusta. Cloud SDK clienttiin kirjaudutaan Firebase-clientin tavoin selaimen kautta. Sovelluksen julkaiseminen on yhtä helppoa kuin Firebasen clientillä ja yksi komento riittää *gcloud app deploy* (Kuvio 31). (Google Cloud 2021d.)

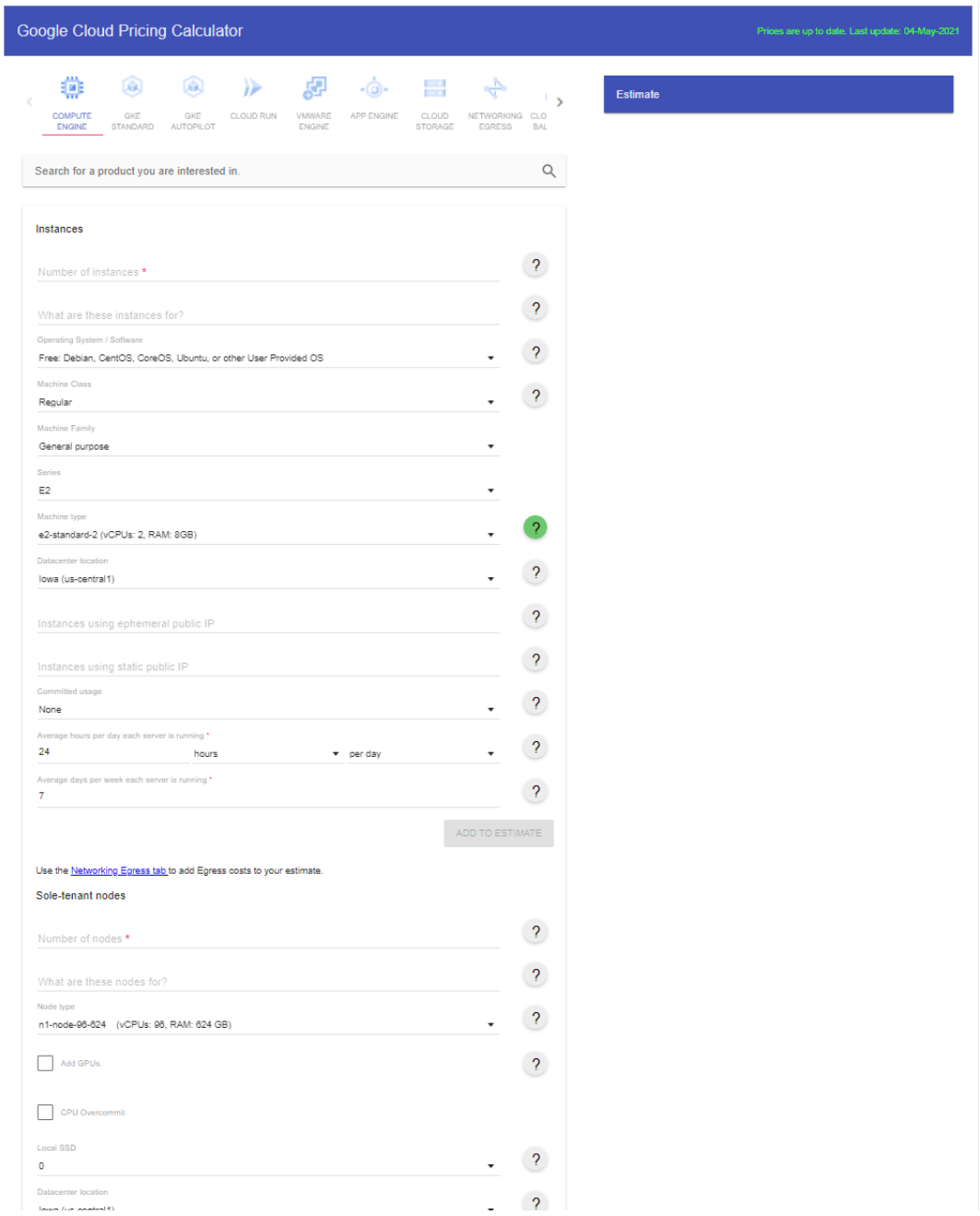


Kuvio 30. Google Cloud App Enginen palvelun julkaiseminen

Tämän jälkeen sovellus on heti käytettävissä ilmoitetussa osoitteessa. Jos käytössä on Node Express ja Nodemon-kehitystyökalu, se täytyy poistaa projektin *package.json*-tiedostosta ennen sovelluksen julkaisua. Jos tietää jo projektin alussa että käytetään vanhempia Node-paketteja, kannattaa asentaa NVM eli Node Version Manager. Tällä on helppo hallita Noden asennusversioita sopivan kompromissin löytämiseksi.

2.7 Google Cloudin hinnoittelu

Google Cloudin dokumentaatiosta löytyy kattava laskuri projektin hinta-arviolle (Kuvio 32). Käyttäjä voi valita eri komponentteja ja laskea niiden konfiguraatioiden perusteella hinta-arvioita projektilleen. On suositeltavaa tutustua hinnotteluun jo etukäteen.



Kuvio 31. Google Cloudin hintalaskuri

3 FRONT ENDIN TOTEUTTAMINEN

3.1 Node.js

Node.js on avoimeen lähdekoodin perustuva, Chromen V8 JavaScript -moottorilla kehitetty, asynkroninen, tapahtumasilmukassa toimiva suoritusympäristö, JavaScript-koodille. Node.js-ympäristöä ei välttämättä tarvita, jos esimerkiksi asennetaan Vue.js käyttäen CDN-asennusta. Vuen kehittäminen ilman Node.js-ympäristöä, automaattisella uudelleenkäynnistyksellä, kuulostaa jo niin työläältä prosessilta ettei sitä voi suositella kenellekään. Myös Vuen suositut työkalut ja lisäosat käyttävät Noden pakettienhallintaa. (Nodejs 2021.)

```
const http = require('http');

const hostname = '127.0.0.1';
const port = 3000;

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World');
});

server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});
```

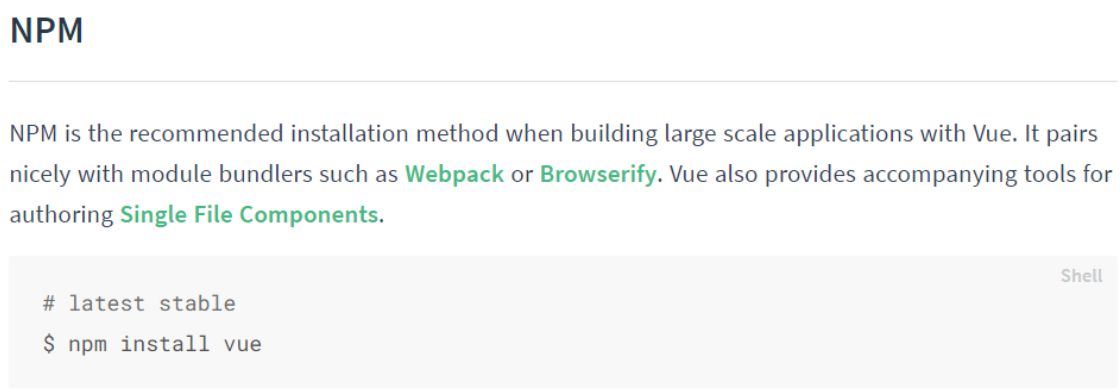
Kuvio 32. Yksinkertainen Node.js-palvelin (Nodejs 2021)

3.2 Vue.js

Vue.js on käyttöliittymien valmistukseen tehty JavaScript framework eli ohjelmistokehys. Vuen avulla web-sivu voidaan jakaa yksittäisiin komponentteihin, joilla kaikilla voi olla omat HTML-, CSS- ja JavaScript-ominaisuudet. Vue.js soveltuu hyvin dynaamisiin yhden sivun applikaatioihin. (Vuejs 2021c.)

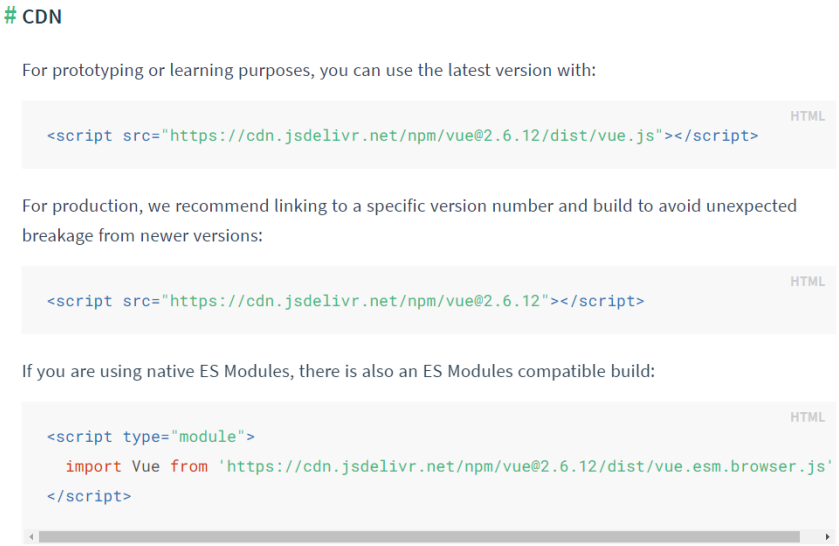
Vue.js voidaan liittää web-sivulle sisällönjakeluverkon tai noden pakettienhallinnan avulla. Kehittäjät suosittelevat noden pakettienhallinnan käyttöä suurempien applikaatioiden kehityksessä. NPM-asennuksen huono puoli on nopeasti

kasvava JavaScript-tiedostojen koko. Kuviossa 33 esitellään NPM-asennuksen ohjeet. NPM-asennus suoritetaan yksinkertaisesti komennolla *npm install vue* (Kuvio 33). (Vuejs 2021e.)



Kuvio 33. Asentaminen käyttäen Noden pakettienhallintaa (Vuejs 2021e)

CDN-asennuksen eduksi voidaan mahdollisesti lukea sivustoa ladattaessa tapahtuva erillinen Vuen lataus, joka ei kasvata sivuston hosting-palvelimelta ladattavien tiedostojen kokoa, mutta aiheuttaa omat haasteensa CDN-palvelimen yhteyden kanssa varsinkin hitaassa mobiiliverkossa. Sivuston toimivuudessa voi esiintyä ongelmia ja puutteita latauksen aikana. Tämän takia kuvien kanssa on tärkeää käyttää esimerkiksi HTML-koodia *alt="Ladataan..."*, jotta käyttäjä ymmärtää sivuston toimivan vaikka yhteys onkin hidas ja sisällön lataus on vielä kesken (Kuvio 34).

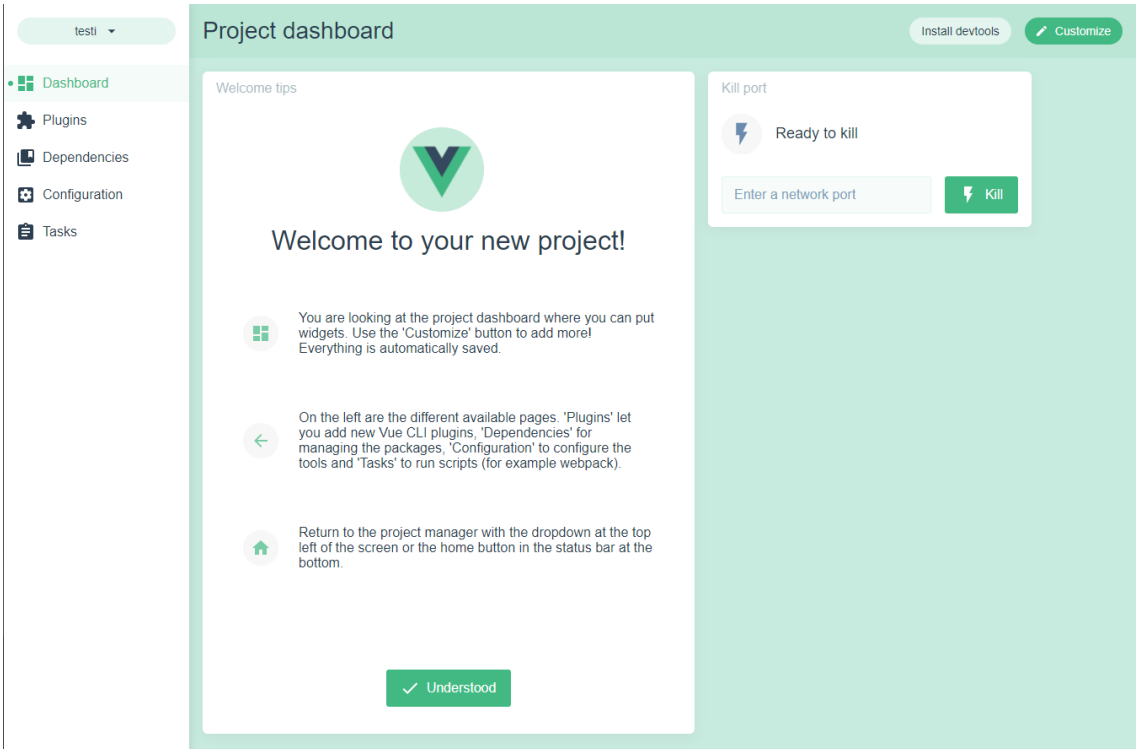


Kuvio 34. Asentaminen käyttäen CDN-menetelmää

Vuen kehityksessä lähes poikkeuksetta esiin tuleva yleinen ongelma on palvelimelta ladattavien JavaScript-tiedostojen suuri koko. Tähän ongelmaan ja sen ratkaisuihin on syytä perehtyä heti alussa, jolloin sovelluksen osat on helpompaa rakentaa loppuun samaa tyyliä noudattaen. Vue Router -dokumentaatio käsittelee reittien lazy loadingia eli laiskaa lataamista yhdessä Webpackin koodin pilkkomisen kanssa. Tästä on hyvä aloittaa jotta voi paremmin ymmärtää ongelman ja sen ratkaisut. (Webpackjs 2021a. Vuejs 2021a.)

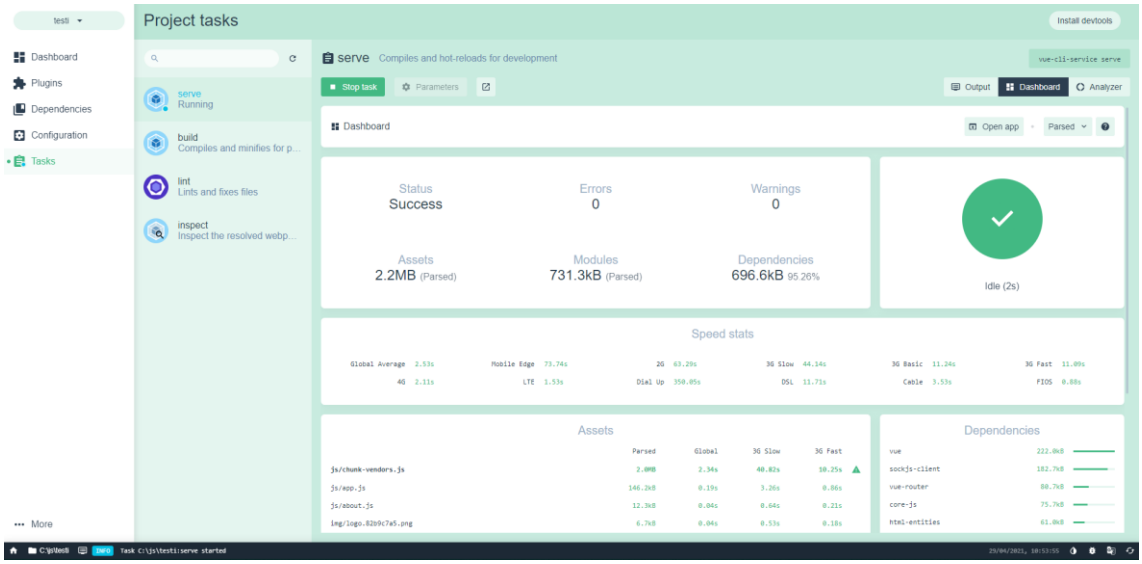
3.3 Vue CLI

Vue CLI on graafisella käyttöliittymällä varustettu järjestelmä nopeiden prototyyppien valmistamiseen. Sillä voidaan rakentaa automaattisesti projektin runko ja lisätä käytettävissä olevia lisäosia projektiin. Esimerkiksi Vue-reitittimen lisääminen onnistuu yhdellä komennolla ja CLI tekee kaikki tarvittavat kansiot ja tiedostot valmiiksi. Kuviossa 35 esitellään Vue CLI:n etusivu (Kuvio 35).



Kuvio 35. Vue CLI:n graafisen käyttöliittymän etusivu

Vue CLI:n graafinen käyttöliittymä sopii hyvin esimerkiksi vasta Vueen tutustuville käyttäjille tai sellaisille käyttäjille jotka vierastavat komentoriviltä tapahtuvaa konfigurointia ja käyttöä. Vue CLI:n graafinen käyttöliittymä sisältää tarvittavat tehtävät projektin kehittämiseen ja julkaisupaketin tekemiseen. Kuviossa 39 näkyy Vue ui:n tehtävänäkymä (Kuvio 39).



Kuvio 36. Vue CLI:n graafisen käyttöliittymän tehtävät

Tehtävistä löytyy myös lint-komento joka tarkistaa koodin virheiden varalta. Inspect-komento näyttää käytössä olevan Webpack-konfiguraation (Kuvio 39). Sovelluksen voi käynnistää, sammuttaa ja buildata käyttöliittymältä.

3.4 Vue Router

Vue Router on Vuen virallinen reititin jonka avulla on helppo rakentaa dynaamisia yhden sivun sovelluksia. Projektiin sijoitetaan *<router-view></router-view>* -osio, johon Vue renderöi reitittimelle kulloinkin navigaatiopolkuna annetun komponentin sisällön. Muutamista hyvistä ominaisuuksista voidaan mainita Vue-reitittimen lisäämä mahdollisuus käyttää parametreja navigaatioreiteille ja navigaation historiatila. Kuviossa 40 näkyy Vue-reititinnäkymän lisääminen koodiin (Kuvio 40). Reititinnäkymä voidaan sijoittaa lähes mihin tahansa HTML-koodissa, jolloin kehittäjä saa vapaasti valita missä kohti sivua reitittimen sisältö näkyy.

```
                </li>
                </ul>
              </form>
            </div>
          </div>
        </nav>

        <router-view></router-view>

      </div>
    </div>

</template>
```

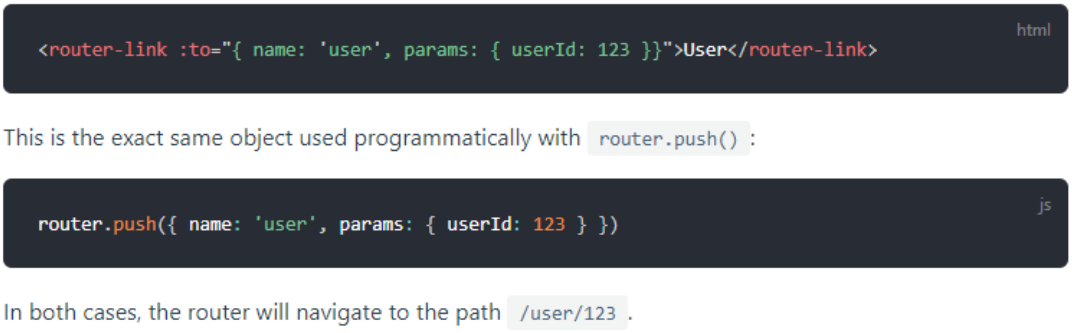
Kuvio 37. Vue-reitittimen reititinnäkymä

Kuviossa 41 näkyy esimerkki reitittimen reittikonfiguraatiosta, jossa kaupunkireitille on annettu parametriksi kaupunki (Kuvio 41). Navigaatio voidaan tällöin toteuttaa ohjelmallisesti esimerkiksi funktiosta seuraavalla tavalla *this.$router.push({{ this.selectedcity }}).* Näin voidaan toteuttaa automaattinen navigointi vaikka dropdown-menusta. Kun käyttäjä valitsee listasta kaupungin, niin *selectedcity:n* arvoksi muuttuu valittu kaupunki ja navigointi voidaan tarvittaessa tehdä aiemmin esitellyllä tavalla vaikka samasta funktiosta. (Vuejs 2021d.)

```
const routes = [
  { path: '/', component: Home },
  { path: '/city/:city', component: City },
  { path: '/user', component: User },
  { path: '/userinfo', component: Userinfo },
  { path: '/usersales', component: Usersales },

]
```

Kuvio 38. Parametrin antaminen reitille

Reitti käyttää City-komponenttia johon valittu kaupunki saadaan nyt reitistä parametrina jollei haluta käyttää esimerkiksi Vuex-tilaa (Kuvio 42). Parametria voidaan käyttää navigoidussa komponentissa seuraavasti *this.$route.params.city*. (Vuejs 2021.)



Kuvio 39. Nimetty reitti (Vuejs 2021)

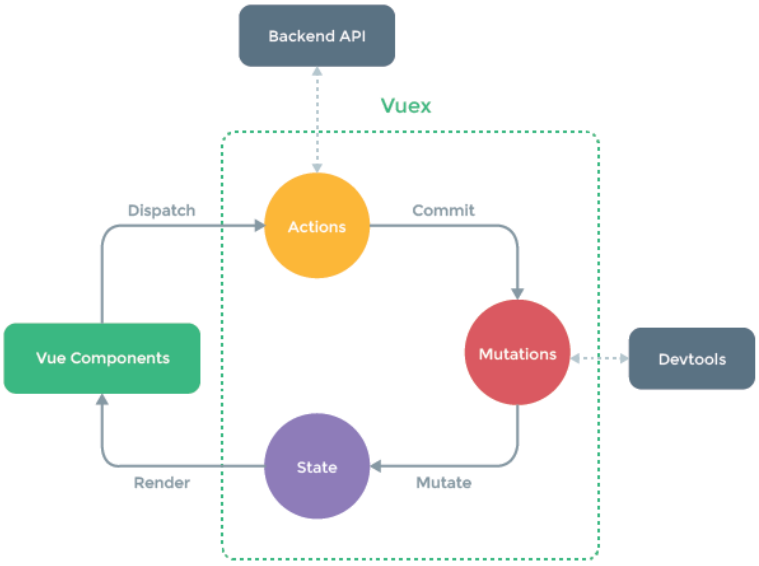
3.5 Vuex Store and State

Vuex on työkalu sovelluksen gloobaleille muuttujille, joiden avulla samaa dataa voidaan hyödyntää kaikissa komponenteissa reaaliajassa. Esimerkiksi käyttäjien hallinta vaatii tällaisen ratkaisun, kun sivusto käyttää useita komponentteja. Storesta voidaan hakea helposti ja nopeasti nykyisen käyttäjän tiedot ja kirjautumisen tilaa on helppo valvoa (Kuvio 43). (Vuejs 2021g.)

Kuvio 40. Autentikoinnin tilan valvonta

Kun käyttäjä kirjautuu sovellukseen, käyttäjän tiedot tallennetaan Vuex storeen, joka valvoo tapahtumia Firestoren avulla reaaliajassa. Jos käyttäjä poistaa ses-

sion tai tekee jotain muuta autentikaatioon vaikuttavaa, saadaan tieto automaattisesti Storesta kaikille komponenteille, ilman että jokainen komponentti tarkistaa käyttäjän tilan jokaisella päivityksellä erikseen. Kuviossa 44 on esitelty Vuexin arkkitehtuuria ja toimintaa (Kuvio 44).



Kuvio 41. Vuexin tila ja tallennus (Vuejs 2021g)

Vuex lisätään app-instanssin viitteeksi jolloin sitä voidaan käyttää globaalisti komennolla *this.$store*, ilman *import*- ja *require*-komentoja jokaiseen komponenttiin. Vuex pitää ottaa käyttöön komenolla *Vue.use()*, kuten muissakin sovelluskehyksissä (Kuvio 45).

```
import Vuex from 'vuex'
Vue.use(Vuex)

// Global state vars
const store = new Vuex.Store({
  state: {
    currentCity: '',
    user: '',
    email: '',
    popularCafes: [
```

Kuvio 42. Vuexin konfiguraatio

Kuviossa 46 esitellään Vuexin liittäminen Vue-sovellukseen (Kuvio 46). Seuraavassa osiossa esitellään Vuex-konfiguraatiota ja Vuexin käyttöä tarkemmin. Vue-reititin on lisätty sovellukseen samalla tavalla.

```
new Vue({
  el: '#app',
  router,
  store,
  render: h => h(App)
})
```

Kuvio 43. Storen lisääminen sovellukseen

3.6 Vuex Storen käyttö

Komponentissa voidaan käyttää Vuen *computed()*-ominaisuutta, joka hakee tiedon Vuex storesta. Storen tietoa hallitaan gettereillä ja mutaatioilla. Seuraavaksi esitellään *count*-tiedon noutaminen Storesta.

```
computed: {
  count () {
    return this.$store.state.count
  }
}
```

Komennolla *this.$store.commit('increment', 10)* voidaan muuttaa storessa oleva arvo arvoksi 10. Kuviossa 47 on esitelty projektin Vuex Store -testikonfiguraatio (Kuvio 47). Tietoa haetaan gettereillä eli noutajilla ja päivitetään mutaatioilla eli tiedon muokkaamisella. Kun jossain komponentissa halutaan tietoa käyttäjästä, voidaan kutsua *this.$store.getters.user*, jolloin saadaan tieto parhaillaan kirjautuneesta käyttäjästä.

```
// Global state vars
const store = new Vuex.Store({
  state: {
    currentCity: '',
    user: '',
    email: '',
    popularCafes: [
      ["Ladataan...", "~@./assets/pexels-chevanon-photography-324028.jpg", "Ladataan...", "Ladataan..."],
      ["Ladataan...", "~@./assets/pexels-chevanon-photography-324028.jpg", "Ladataan...", "Ladataan..."],
      ["Ladataan...", "~@./assets/pexels-chevanon-photography-324028.jpg", "Ladataan...", "Ladataan..."],
      ["Ladataan...", "~@./assets/pexels-chevanon-photography-324028.jpg", "Ladataan...", "Ladataan..."],
    ]
  },
  getters: {
    user: state => {
      console.log("USER getter: " + state.user)
      return state.user
    },
    email: state => {
      console.log("EMAIL getter: " + state.email)
      return state.email
    },
    city: state => {
      console.log("city getter: " + state.currentCity)
      return state.currentCity
    },
    cafes: state => {
      console.log("cafe getter: " + state.popularCafes)
      return state.popularCafes
    }
  },
  mutations: {
    updateCurrentCity (state, newCity) {
      if (newCity != "") {
        state.currentCity = newCity
      }
    },
    updateUser (state, newuser) {
      console.log("User Setter: " + newuser)
      if (newuser != null) {
        state.user = newuser
      }
      else {
        state.user = ""
      }
    },
    updateEmail (state, newuser) {
      console.log("Email Setter: " + newuser.email)
      if (newuser != null) {
        state.email = newuser.email
        console.log("storing: " + newuser.email)
      }
      else {
        state.email = ""
      }

    },
```

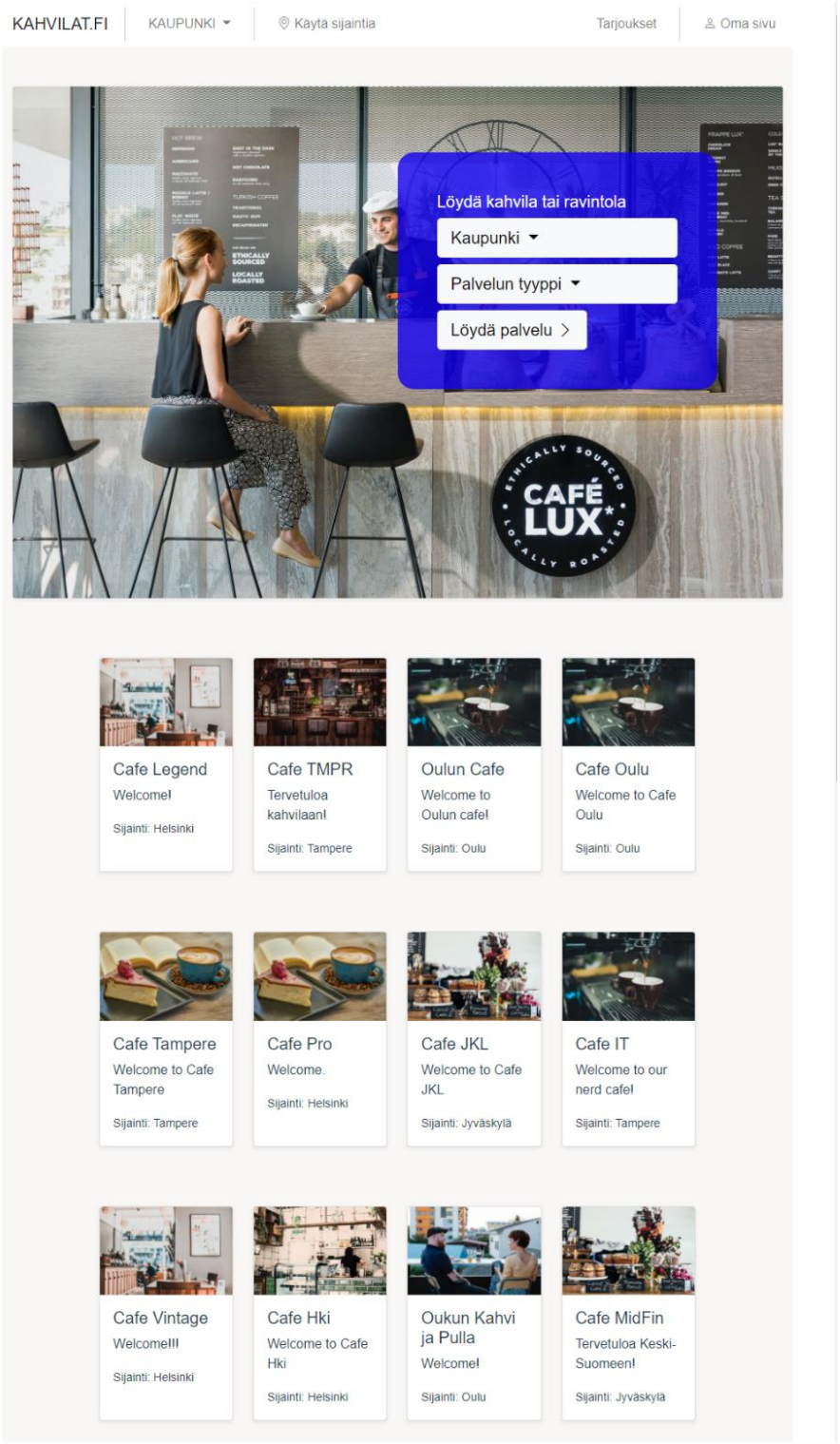
Kuvio 44. Vuex Storen konfiguraatio

Kuviossa 48 esitellään projektin funktio, joka hakee tiedot Firebasesta ja päivittää suositut kahvilat Vuex Storeen (Kuvio 48). Käyttöliittymän etusivu kutsuu *oncreated()*-toiminnossaan funktiota, joka valmistuessaan päivittää etusivulle suosituimmat kahvilat.

```
async getPopularCafes() {
  var tempArr = []

  const db = firebase.firestore()
  const cafesRef = db.collection('cafes')
  const snapshot = await cafesRef.where('popular', '==', true).get()
  if (snapshot.empty) {
    console.log('No matching documents.')
    return;
  }

  this.popular = []

  snapshot.forEach(doc => {
    //console.log(doc.id, '=>', doc.data())
    tempArr.push(doc.data().name, doc.data().info, doc.data().city, doc.data().smallimage, doc.data().mainimage)
    this.popular.push(tempArr)
    console.log("Pushed temp: " + tempArr)
    tempArr = []

  });

  console.log(this.popular)
  this.$store.commit('updatePopCafes', this.popular)

},
```

Kuvio 45. Vuex Storen tiedon päivittäminen

Font end -osion lopuksi vielä kuva Vue.js-käyttöliittymästä (Kuvio 49). Käyttäjä voi käyttää pikahakua dropdown-menuilla tai valita kaupungin navigaatio-palkista. Tämän jälkeen käyttäjä ohjataan kaupunkisivulle, joka lataa Firebasesta kyseisessä kaupungissa sijaitsevat kahvilat.



Kuvio 46. Projektin käyttöliittymä

4 ADMIN BACK ENDIN TOTEUTTAMINEN

4.1 Google Firebase Admin SDK:n esittely

Google Firebase admin SDK:n avulla voidaan suorittaa monenlaisia tehtäviä projektin ylläpidossa (Taulukko 2). Projektissa esitellään Firebase Admin SDK:n käyttöä käyttäjien hallinnan avulla. Admin SDK:lla voidaan hallita myös muita tietokantoja.

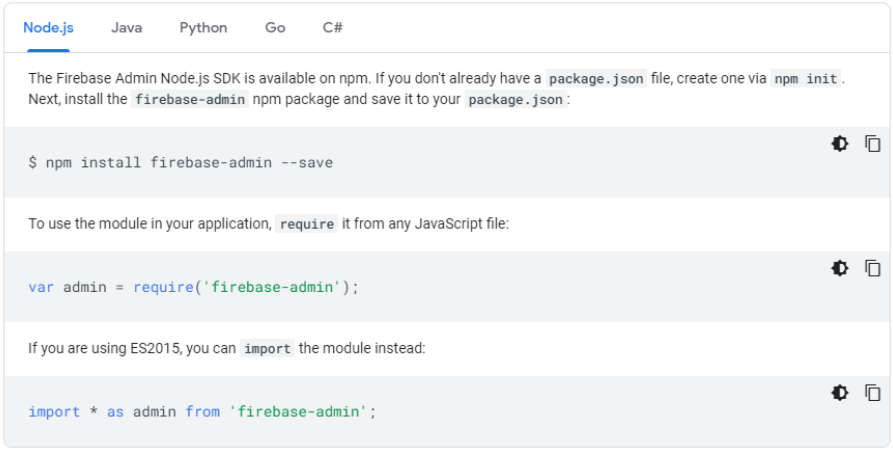
Taulukko 2. Admin SDK:n ominaisuudet (Google Firebase 2021f)

Feature	Node.js	Java	Python	Go	C#
Custom Token Minting	✓	✓	✓	✓	✓
ID Token Verification	✓	✓	✓	✓	✓
User Management	✓	✓	✓	✓	✓
Control Access With Custom Claims	✓	✓	✓	✓	✓
Refresh Token Revocation	✓	✓	✓	✓	✓
Import Users	✓	✓	✓	✓	✓
Session Cookie Management	✓	✓	✓	✓	✓
Generating Email Action Links	✓	✓	✓	✓	✓
Managing SAML/OIDC provider configurations	✓	✓	✓	✓	✓
Multi-tenancy support	✓	✓	✓	✓	✓
Realtime Database	✓	✓	✓	✓ *	
Firebase Cloud Messaging	✓	✓	✓	✓	✓
FCM Multicast	✓	✓	✓	✓	✓
Manage FCM Topic Subscriptions	✓	✓	✓	✓	✓
Cloud Storage	✓	✓	✓	✓	
Cloud Firestore	✓	✓	✓	✓	
Project Management	✓	✓	✓		
Security Rules	✓				
ML Model Management	✓		✓		
Firebase Remote Config	✓	✓			

4.2 Asennus

Asennus on helppoa ja suoritetaan Noden pakettienhallinalla, komento on *npm install firebase-admin –save* (Kuvio 51). Admin SDK tarvitsee JSON-tiedoston,

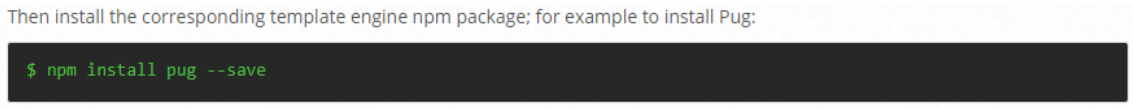
jossa on konfiguroituna salausavaimet ja projektin tiedot. Tiedoston tulee olla palvelimella Admin SDK-palvelun saatavilla. (Google Firebase 2021f)



Kuvio 47. Admin SDK:n npm-asennus (Google Firebase 2021f)

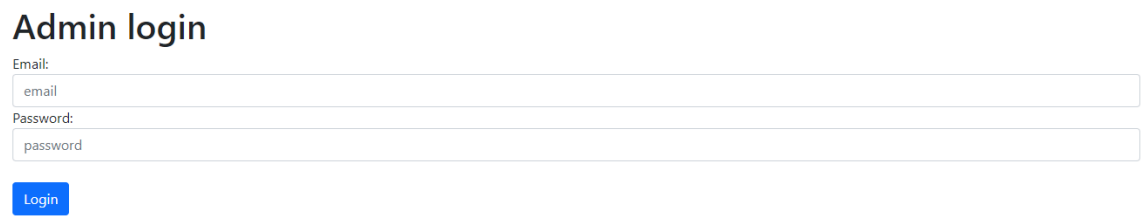
4.3 Käyttö

Admin SDK:ta suoritetaan omana palveluna App Enginen puolella, jossa sillä on oma Node Expressin avulla toteutettu sovellus ja käyttöliittymä. Node Expressin kanssa on tärkeää noudattaa selkeää rakennetta heti alusta, jolloin vian haku ja debuggaus pysyvät selkeänä sovelluksen koon kasvaessa. Projektissa käytetään Bootstrappia ja pug view engineä eli pug.js-kirjastoa käyttöliittymän näkymien HTML-generaattorina (Kuvio 52) ja Bootstrappia CSS-tyylien sovelluskehyksenä. (Expressjs 2021. Pugjs 2021.)



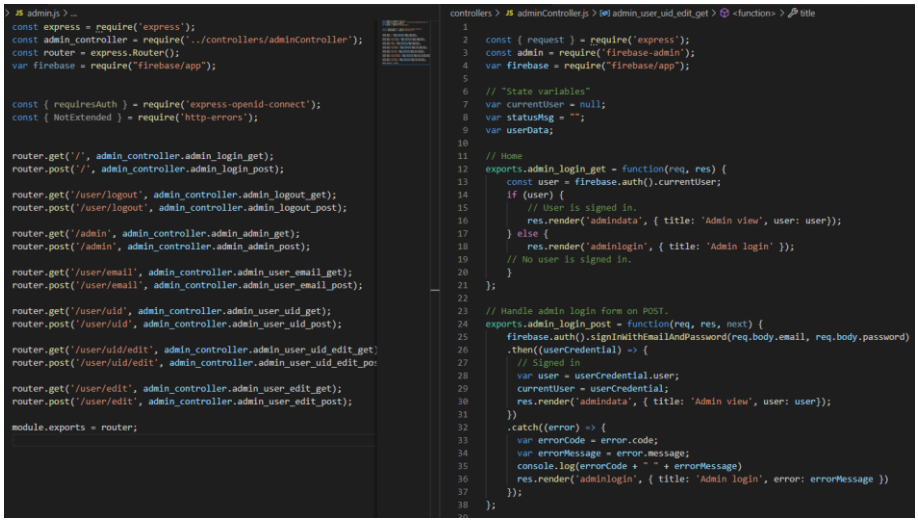
Kuvio 48. Pugin asennus Express sovelluskehykseen (Expressjs 2021)

Admin-sovellus tulee niin sanotusti omaan käyttöön, jolloin ulkonäkö noudattaa korutonta insinöörityökalun tyyliä (Kuvio 53). Admin käyttäjät autentikoidaan omaan Firebase-tietokantaan. Tällaisessa ratkaisussa on ehdottoman tärkeää käyttää vähintään kaksivaiheista todennusta.



Kuvio 49. Admin back endiin kirjautuminen

Admin-sovellukselle on tehty oma Firebase-projekti, jolloin voidaan käyttää Firebasen tuttua käyttäjien autentikointia. Admin back end noudattaa hyväksi havaittua Mozillan The Local Library website Express.js-tutoriaalin pohjaa (Kuvio 54). (Mozilla 2021.)



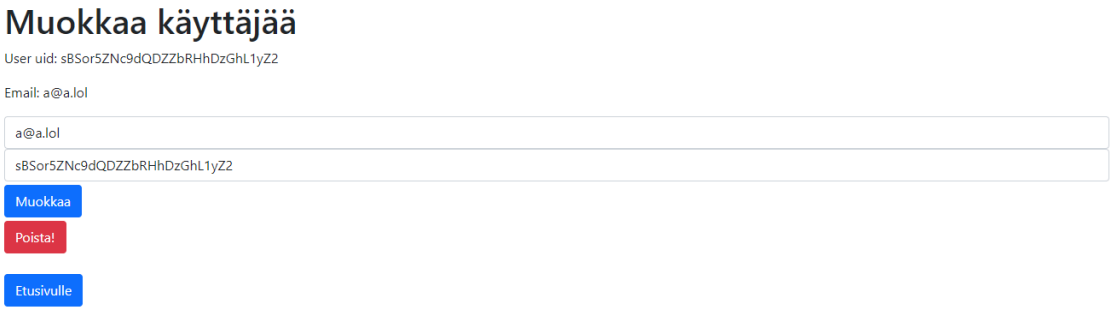
Kuvio 50. Admin back endin runko

Firebase on liitetty Express.js-projektiin samalla tavalla kuin Vue.js-projektiin. Kuviossa 55 näkyy käytetty Firebasen admin-konfiguraatio, sekä aiemmin mainittu pug view engine (Kuvio 55).

```
// Firebase App (the core Firebase SDK) is always required and
// must be listed before other Firebase SDKs
var firebase = require("firebase/app");

// Add the Firebase products that you want to use
require("firebase/auth");
require("firebase/firestore");

const firebaseConfig = {
  apiKey: "[redacted]",
  authDomain: "opariadmin.firebaseapp.com",
  projectId: "opariadmin",
  storageBucket: "[redacted]",
  messagingSenderId: "[redacted]",
  appId: "[redacted]"
};

// Initialize Firebase
firebase.initializeApp(firebaseConfig);

const adminRouter = require('./routes/admin');
const app = express();

// view engine setup
app.set('views', path.join(__dirname, 'views'));
app.set('view engine', 'pug');
```

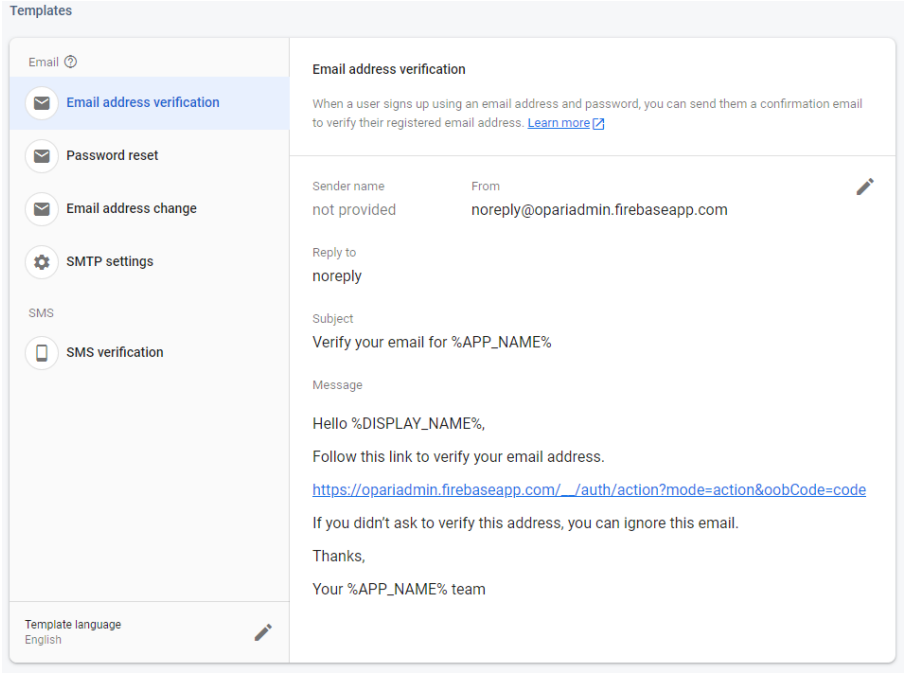
Kuvio 51. Firebasen asetukset Express.js-projektissa

Admin-sovelluksessa on mahdollista päivittää tai poistaa käyttäjä. Käyttäjiä voidaan hakea sähköpostin ja uid:in eli käyttäjätunnisteen avulla (Kuvio 56). Uid on automaattisesti luotu Firebasen toimesta, joten sitä ei voi muokata.



Kuvio 52. Käyttäjän hakeminen uid:illa

Firebase sisältää runsaasti ominaisuuksia käyttäjille, kuten avatarin, puhelinnumeron, erillisen käyttäjänimen ja muita suosittuja ominaisuuksia. Näitä tietoja on mahdollista muokata Admin SDK:n avulla. Yksi Firebasen parhaista ominaisuuksista on valmiit sähköposti mallit joiden avulla käyttäjät voivat suorittaa perustoimintoja. Kuviossa 57 esitellään Firebasen sähköpostimallit (Kuvio 57).



Kuvio 53. Firebasen sähköpostimallit

Sähköpostimallien käyttö voidaan yhdistää Firebasen funktioiden avulla käyttöliittymälle. Käyttäjä voi tällöin pyytää funktion avulla esimerkiksi salasanan vaihtoa ja saa mallin mukaisen viestin sähköpostiin. Kehittäjä voi muokata mallit omaan sovellukseen sopivaksi.

5 POHDINTA

Tuloksena syntyi yksinkertainen Full Stack web-sovellus, joka toimii valtaosalla moderneista selaimista, myös mobiiliselaimilla. JavaScript-tuki vaaditaan, koska käytössä on moderni JavaScript-sovellus. Projekti eteni suunnitelman mukaisesti, mutta muutamia päivityksistä aiheutuneita ongelmia ja muutostarpeita ilmeni, jotka viivästivät projektin valmistumista ajoissa. Suurimpana parannuskohteena ilmeni Vue.js:tä tuttu vendors.js tiedoston koko, joka tulisi pilkkoa Vue-reittimen lazy loadingin ja Webpackin koodin pilkkomisen avulla. Suurin syy JavaScript-sovellusten hitauteen on palvelimelta ladattavien tiedostojen suuri koko. Tähän on tärkeää kiinnittää huomiota heti projektin alussa.

Toisena kehityskohteena voidaan mainita sovelluksen mobiililayout. Ihmiset viettävät runsaasti aikaa mobiililaitteiden parissa jonka takia sovellusten tulee toimia myös mobiiliresoluutioilla. Grid layout-elementtien muuttaminen selaimen resoluution mukaan media queryillä on suhteellisen helppoa, mutta sen vaatiman runsaan testaamisen kanssa aikaa vievää. Tästä syystä tämä osio jäi vielä odottamaan tulevaisuuteen.

Sovelluksen runko valmistui nopeasti, koska kehitysympäristö oli jo ennestään tuttu. Työkalujen asennus on melko yksinkertaista, joten vasta-alkajakin saa sovelluksen käyntiin varsin nopeasti. Käytettyjen työkalujen dokumentaation laatu on ihan hyvällä tasolla, mutta aiempi kokemus tietysti auttaa paljon eteenpäin. Web-kehitys tuntuu nykyään pyörivän erilaisten sovelluskehysten ympärillä, koska niillä saadaan nopeasti tuloksia. Kun käytössä on esimerkiksi Bootstrap CSS-sovelluskehys, saadaan harrastekäytössäkin luotua varsin ammattimaisen näköisiä sovelluksia. Projektin runkoa jatkokehitetään erään startup-hankkeen pohjana. Kaiken kaikkiaan projekti onnistui ihan hyvin ja lopputuloksena voidaan todeta Google Cloudin soveltuvan Full Stack-sovellusalustaksi erittäin hyvin.

LÄHTEET

Bootstrap 2021a. Build fast, responsive sites with Bootstrap. Viitattu 31.3.2021 https://getbootstrap.com.

Bootstrap 2021b. Grid system. Viitattu 10.5.2021 https://getbootstrap.com/docs/5.0/layout/grid.

Expressjs 2021. Using template engines with Express. Viitattu 10.5.2021 https://expressjs.com/en/guide/using-template-engines.html.

Google Cloud 2021a. App Engine. Viitattu 11.4.2021 https://cloud.google.com/appengine#all-features.

Google Cloud 2021b. An overview of app engine. Viitattu 1.5.2021 https://cloud.google.com/appengine/docs/standard/nodejs/an-overview-of-app-engine.

Google Cloud 2021c. Cloud CDN. Viitattu 25.5.2021 https://cloud.google.com/cdn.

Google Cloud 2021d. Installing google cloud SDK. Viitattu 11.4.2021 https://cloud.google.com/SDK/docs/install#windows.

Google Cloud 2021e. Serverless Computing. Viitattu 11.4.2021 https://cloud.google.com/serverless.

Google Cloud 2021f. Google Cloud Products. Viitattu 30.5.2021 https://cloud.google.com/products.

Google Firebase 2021a. Blaze plan calculator. Viitattu 3.5.2021 https://firebase.google.com/pricing.

Google Firebase 2021b. Choose a Database: Cloud Firestore or Realtime Database. Viitattu 3.5.2021 https://firebase.google.com/docs/database/rtdb-vs-firestore.

Google Firebase 2021c. Firebase CLI reference. Viitattu 31.3.2021 https://firebase.google.com/docs/cli.

Google Firebase 2021d. Products / Build. Viitattu 1.4.2021 https://firebase.google.com/products-build.

Google Firebase 2021e. Pricing plans. Viitattu 14.4.2021 https://firebase.google.com/pricing.

Google Firebase 2021f. Add the Firebase Admin SDK to your server. Viitattu 30.5.2021 https://firebase.google.com/docs/admin/setup.

Mozilla 2021a. Async function. Viitattu 1.5.2021 https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/async_function.

Mozilla 2021b. Express Tutorial: The Local Library website. Viitattu 10.5.2021 https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/Tutorial_local_library_website.

Nodejs.org 2021. About Node.js. Viitattu 22.4.2021 https://nodejs.org/en/about.

Npmjs 2021. About npm. Viitattu 26.4.2021 https://docs.npmjs.com/about-npm.

Pexels 2021. The best free stock photos & videos shared by talented creators. Viitattu 10.5.2021 https://www.pexels.com.

Pugjs 2021. Express Integration. Viitattu 9.5.2021 https://pugjs.org/api/express.html.

Tech Altum 2021. Fullstack Web Developer. Viitattu 3.5.2021 https://www.techaltum.com/fullstack-web-development.html.

Vuejs 2021a. Vue Router, Lazy loading routes. Viitattu 9.5.2021 https://router.vuejs.org/guide/advanced/lazy-loading.html#lazy-loading-routes.

Vuejs 2021c. What is Vue.js? Viitattu 23.4.2021 https://vuejs.org/v2/guide/#What-is-Vue-js.

Vuejs 2021d. Vue Router. Viitattu 29.4.2021 https://router.vuejs.org.

Vuejs 2021e. Installation. Viitattu 30.5.2021 https://vuejs.org/v2/guide/installation.html.

Vuejs 2021g. What is Vuex? Viitattu 30.5.2021 https://vuex.vuejs.org/#what-is-a-state-management-pattern.

Wallenius Consulting 2021. Mitä mikropalvelut ovat? Viitattu 1.5.2021 https://niklaswallenius.fi/mikropalvelut.

Webpackjs 2021a. Code splitting. Viitattu 9.5.2021 https://webpack.js.org/guides/code-splitting.

Webpackjs 2021b. Concepts. Viitattu 10.5.2021 https://webpack.js.org/concepts.

W3schools 2021. What is Full Stack? Viitattu 3.5.2021 https://www.w3schools.com/whatis/whatis_fullstack.asp.