

Работа во облак

- вежби -



Firebase

Firebase од Google нуди два типа решенија за решавање на проблемот со чување и достапност на податоците. **База на податоци во реално време** и **Cloud Firestore** се бази на податоци базирани на облак кои поддржуваат синхронизирање на податоци во реално време. База на податоци во реално време е ефикасно решение и е наменета за мобилни апликации кои бараат синхронизација во реално време. Cloud Firestore е исто така наменета за мобилни апликации и е наследник на база на податоци во реално време со нов и поинтуитивен модел на податоци. Cloud Firestore е побогата, побрза и поскалабилна од базата на податоци во реално време.



Cloud Firestore

Cloud Firestore претставува база со податоци базирана на документи, која нуди подобро структурирање и сите податоци се сочувани во документи кои може да вклучуваат податоци од различен тип (текст, цел број, децимален број или пак објект). Сите документи се групирани во колекции.

Едноставни CRUD операции - Cloud Firestore овозможува додавање, бришење, модифицирање и читање на податоци од базата. Таа овозможува синхронизација на податоци помеѓу различни апликации и сервиси преку посебни интерфејси. За читање на податоци од базата постојат две опции и тоа читање на цела колекција и читање на документ.

Податоците се чуваат на различни локации и тоа доведува до глобална скалабилност и достапност

Синхронизација помеѓу различни апликации и сервиси во реално време (мобилни и веб апликации).

	База на податоци во реално време	Cloud Firestore
Организација на податоци	Ги складира податоците како едно големо JSON дрво што го олеснува складирањето на едноставни податоци, но затоа е потешко да се организираат сложени, хиерархиски податоци.	Ги зачувува податоците во документи и секој документ е дел од некоја колекција.
Поддршка во реално време и офлајн пристап	офлајн поддршка за мобилни клиенти на iOS и Андроид	офлајн поддршка за iOS, Андроид и веб-клиенти
Пребарување	Пребарувања со ограничена функционалност за подредување и филтрирање. Во едно пребарување, може да се подреди или филтрира, но не и двете истовремено. Пребарувањата секогаш го враќаат целото дрво.	Индексирани пребарувања со сложено подредување и филтрирање. Стандардниот тип на пребарување е индексирани при што ефикасноста на пребарувањето е пропорционална со големината на резултатот, а не со податоците.
Запишувања и трансакции	Ги олеснува основните операции за запишување и трансакција. Податоците можат да бидат напишани како индивидуална операција, додека трансакциите бараат повратен повик дека завршиле.	Ги олеснува операциите за запишување и трансакција. Операциите можат да бидат групирани и завршени веднаш, додека трансакциите ќе се повторуваат се додека не завршат.

← → ↻ 🔒 console.firebase.google.com/u/1/project/raspberrypicourse/firestore/data-2F

🔥 **Firebase**

🏠 Project Overview ⚙️

Develop

- 👤 Authentication
- 🔗 **Cloud Firestore**
- 📄 Realtime Database
- 📁 Storage
- 🌐 Hosting
- ⚡ Functions
- 🧠 Machine Learning

Quality
Crashlytics, Performance, Test La...

Analytics

- 📊 Dashboard
- 🕒 Events
- 🚩 Conversions
- 👥 Audiences
- 📈 Funnels
- 👤 User Properties
- 📢 Latest Release
- 🔄 Retention
- 📄 StreamView
- 🔌 Extensions

Spark
Free \$0/month **Upgrade**

⏪

RaspberryPICourse ▾

Go to docs 🔔 **T** ?


Cloud Firestore

Data Rules Indexes Usage

🏠

🔗 raspberrypicourse

+ Start collection



Your database is ready to go. Just add data.

Cloud Firestore location: eur3 (europe-west)

Задача 2

Креирај node.js express сервер на raspberry pi кој податоците ќе ги сочувува во Cloud Firestore база на податоци и ќе може да се справи со следниве барања:

- ☐ /students – GET метод кој ќе врати HTML документ со листа од студенти
- ☐ /students – POST метод кој ќе се користи за додавање на нови студенти (се проследува само име на студент како патаметар)
- ☐ /students – PUT метод кој ќе се користи за изменување на веќе постоечки студент (се проследува id и ново име на студент како патаметри)
- ☐ /students – DELETE метод кој ќе се користи за бришење на студент (се проследува само id на студент како патаметар)

- ☐ mkdir zadaca3
- ☐ cd zadaca3
- ☐ touch student.js
- ☐ npm init
- ☐ npm install express --save
- ☐ npm install --save body-parser
- ☐ npm install firebase-admin

```
tamara@tamara-ThinkPad-E15: ~
tamara@tamara-ThinkPad-E15:~$ curl -d '{"name":"Maja"}' -H "Content-Type: application/json" -X POST http://localhost:3000/students
Created new student Maja
tamara@tamara-ThinkPad-E15:~$ curl -d '{"name":"Tamara"}' -H "Content-Type: application/json" -X POST http://localhost:3000/students
Created new student Tamara
tamara@tamara-ThinkPad-E15:~$ curl -d '{"name":"Sara"}' -H "Content-Type: application/json" -X POST http://localhost:3000/students
Created new student Sara
tamara@tamara-ThinkPad-E15:~$ curl -d '{"id":"LVKKUNPF3xCNyTYfqbE"}' -H "Content-Type: application/json" -X DELETE http://localhost:3000/students
Deleted student LVKKUNPF3xCNyTYfqbE
tamara@tamara-ThinkPad-E15:~$ curl -d '{"id":"xQRACzSdOnpOPeogl5QU","nameAfter":"Tamara edited"}' -H "Content-Type: application/json" -X PUT http://localhost:3000/students
Edited student Tamara edited
tamara@tamara-ThinkPad-E15:~$
```

← → ↻ ⓘ localhost:3000/students

Students:

1. UrAEzMNi53wkIPYG7iBn - Maja
2. xQRACzSdOnpOPeogl5QU - Tamara edited

🏠 > students > UrAEzMNi53wkIPYG7iBn		
📶 raspberrypicourse	📁 students	📄 UrAEzMNi53wkIPYG7iBn
+ Start collection	+ Add document	+ Start collection
students >	UrAEzMNi53wkIPYG7iBn >	+ Add field
	xQRACzSdOnpOPeogl5QU	name: "Maja"


```
Open [F1]
~/Documents/Предаванъ

1 const express = require('express')
2 const bodyParser = require('body-parser');
3
4 const app = express()
5 const port = 3000
6
7 app.use(bodyParser.json()) // support json encoded bodies
8 app.use(bodyParser.urlencoded({ extended: true })) // support encoded bodies
9
10 const admin = require('firebase-admin')
11 const serviceAccount = require('./firebaseKey.json')
12
13 //initialize admin SDK using serviceAccountKey
14 admin.initializeApp({
15   credential: admin.credential.cert(serviceAccount)
16 });
17 const db = admin.firestore();
18
19 let students = []
20
21 app.route('/students')
22 .get(function (req, res) {
23   res.writeHead(200, { 'Content-Type': 'text/html' })
24   var data = '<html><body>'
25
26   db.collection('students').get().then(doc => {
27     if(doc) {
28       data+='\<h3>Students:</h3><ol>'
29       doc.forEach(student => data+='\<li>' + student.id + ' - ' + student.data().name + '\</li>')
30       data+='\</ol>'
31     }
32     res.write(data + '\</body></html>')
33     res.end()
34   });
35 })
```

```
36 .post(function (req, res) {
37   const name = req.body.name
38   db.collection('students').add({
39     name: name,
40   }).then(() => {
41     res.send('Created new student ' + name)
42     res.end()
43   });
44 })
45 .put(function (req, res) {
46   const id = req.body.id
47   const studentNameAfter = req.body.nameAfter
48   db.collection('students').doc(id).set({
49     name: studentNameAfter,
50   }).then(() => {
51     res.send('Edited student ' + studentNameAfter)
52     res.end()
53   });
54 })
55 .delete(function (req, res) {
56   const id = req.body.id
57   db.collection('students').doc(id).delete().then(() => {
58     res.send('Deleted student ' + id)
59     res.end()
60   });
61 })
62 |
63 app.listen(port, () => {
64   console.log('Example app listening at http://localhost:${port}')
65 })
```


Задача 2

Креирај node.js express сервер на raspberry pi кој податоците ќе ги сочувува во Cloud Firestore база на податоци и ќе може да се справи со следниве барања:

- ☐ /courses – GET метод кој ќе врати HTML документ со листа од курсеви
- ☐ /courses – POST метод кој ќе се користи за додавање на нови курсеви (име на курс, почетен и краен датум како патаметри)
- ☐ /courses/:id – PUT метод кој ќе се користи за изменување на веќе постоечки курс (се проследува име на курс, почетен и краен датум како патаметар)
- ☐ /courses/:id – DELETE метод кој ќе се користи за бришење на курс

Прашања?

**БЛАГОДАРАМ НА
ВНИМАНИЕТО! 😊**