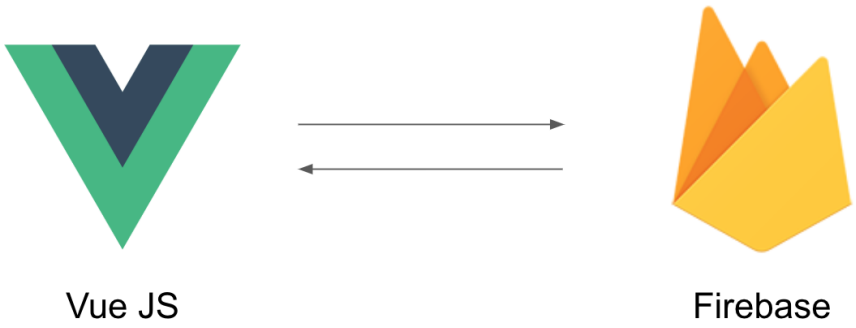
Lab 06 - Cloud

Introduction video part 1 [https://youtu.be/cR9hGEjznRw] part 2 [https://youtu.be/o2rNemKTItw]

1. Introduction

In this lab we are going to learn how to quickly build and host a full-stack web app by leveraging the power of the cloud. In particular, we are going to build a Vue JS app and connect it to a cloud hosted Firebase [https://firebase.google.com/] back-end.

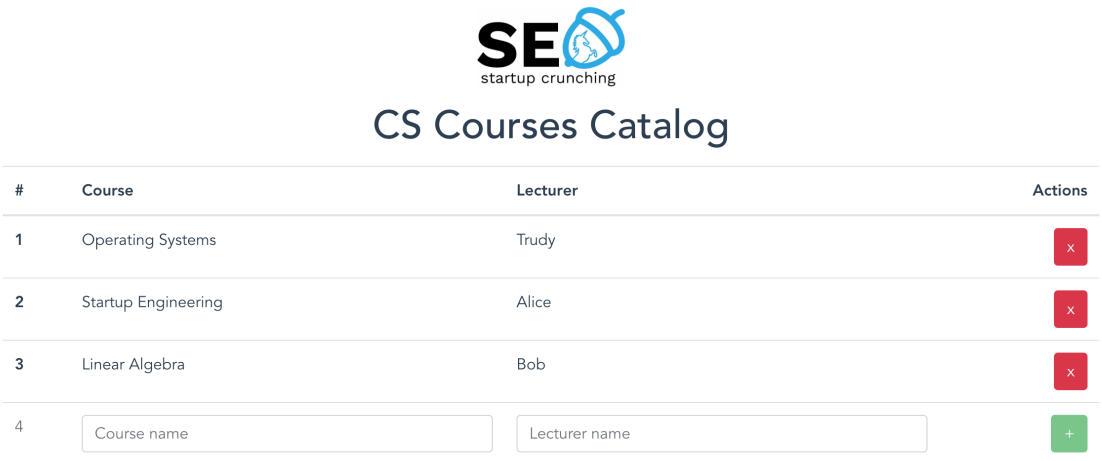


Firebase is a powerful PaaS (Platform as a Service) offering from Google - have a look at this video for a basic overview [https://www.youtube.com/watch?list=PLl-K7zZEsYLmOF_07IayrTntevxtbUxDL&time_continue=2&v=iosNuIdQoy8]. It allows us to save a lot of time implementing all the back-end infrastructure and services that would be required to power the app. Have a look through the Firebase Products page [https://firebase.google.com/products/] to see all the available functionality.

We are going to be using two services out of those:

- Cloud Firestore [https://firebase.google.com/products/firestore/] - to provide our app with a real-time NoSQL back-end for data storage and sync across devices
- Hosting [https://firebase.google.com/products/hosting/] - to serve our static files containing the Vue JS app we will build

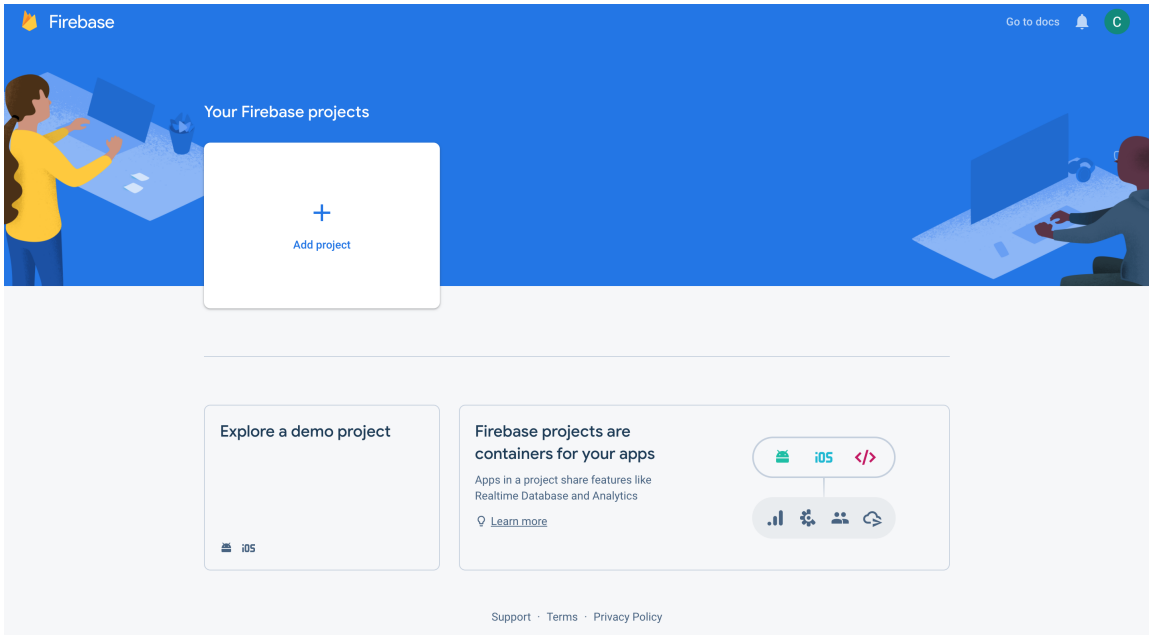
Our goal will be to implement a simple course catalog that displays a list of all the courses in the Computer Science department. We also want this list to be synced in real-time an all devices so that we can instantly see when someone updates the course catalog. This is how the app will look like:



Also, have a look at this finished version of the app [https://courses-fire-app-6bb3e.firebaseapp.com/#/] that has already been deployed. Try opening the app on multiple tabs or multiple computers and notice how the data syncs instantly across devices when modifying the course catalog.

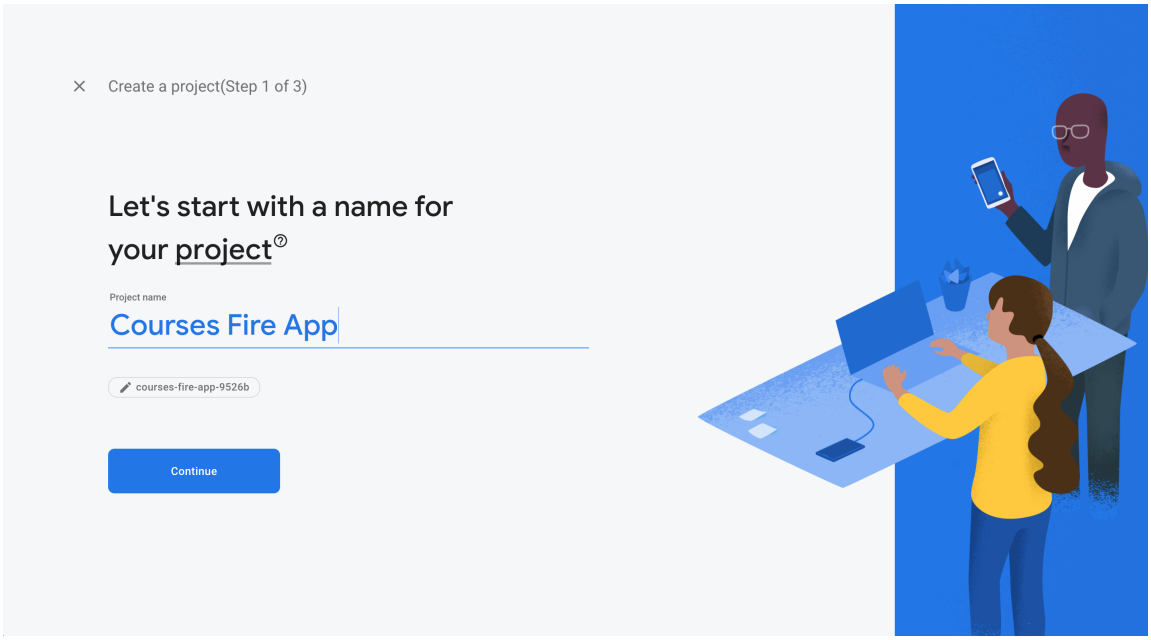
2. Setting up a Firebase project

If you already have a Google account, sign in to the Firebase Console here [https://console.firebase.google.com/]. If you don't have an account, simply create one using the `create account` button on the Login screen. Once logged in, you will see a screen similar to the one below:

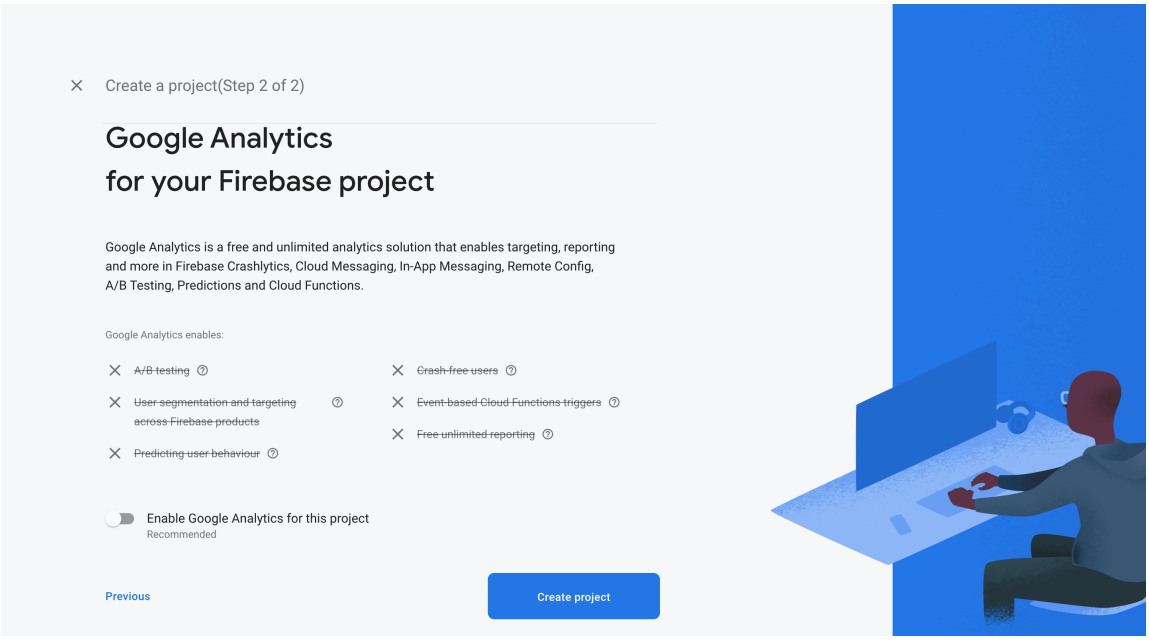


We will be using the free Spark pricing tier, which is more than enough for a small app like ours. If you are curious about how pricing works as your app grows, have a look at the pricing options here [https://firebase.google.com/pricing/].

Now create a new project and name it Courses Fire App. Make sure that the project ID is unique as this will be part of the URL (firebase will suggest a unique ID).

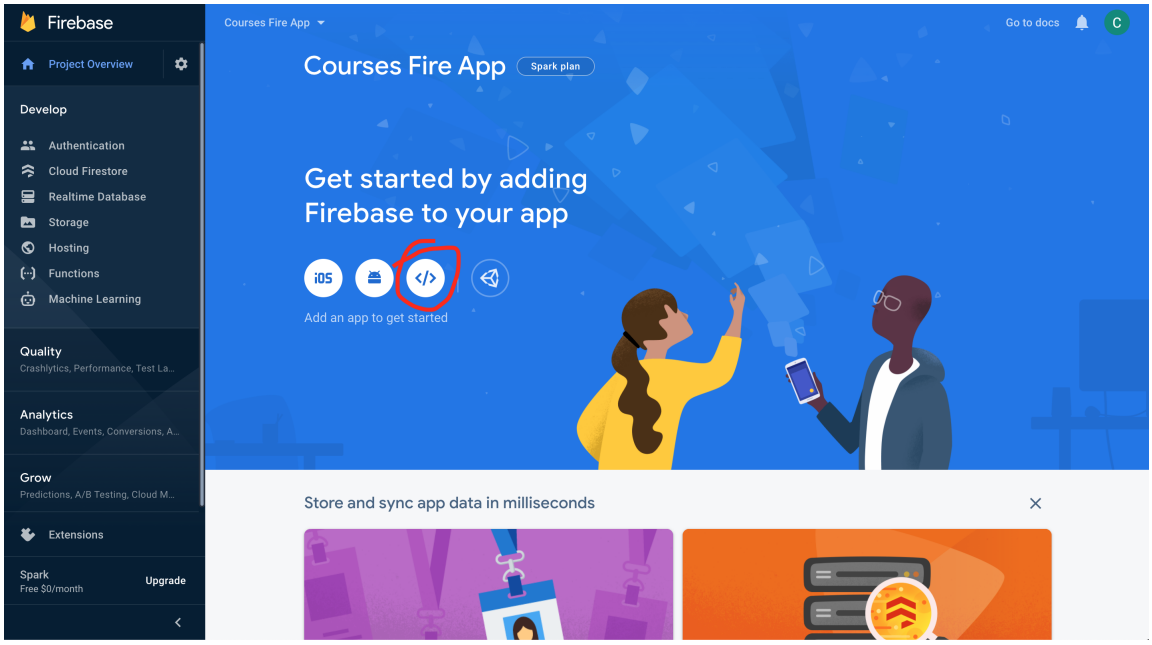


On the second screen disable Google Analytics as we don't need it for this lab.

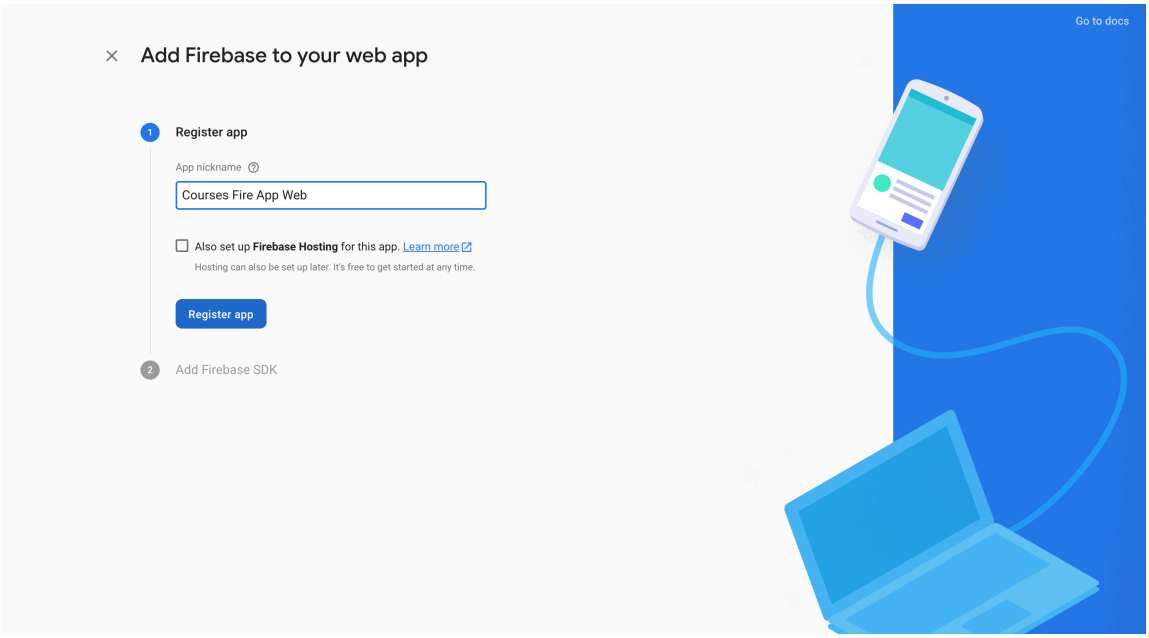


Then hit “create project”.

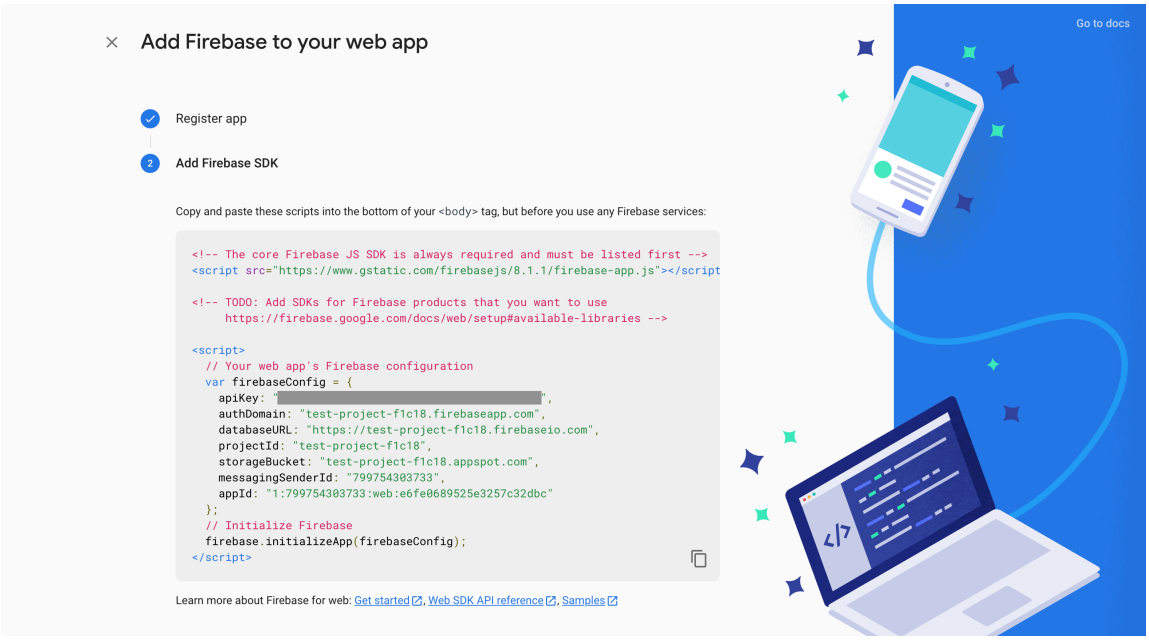
Once you see the dashboard, create a web app using the button circled in the image.



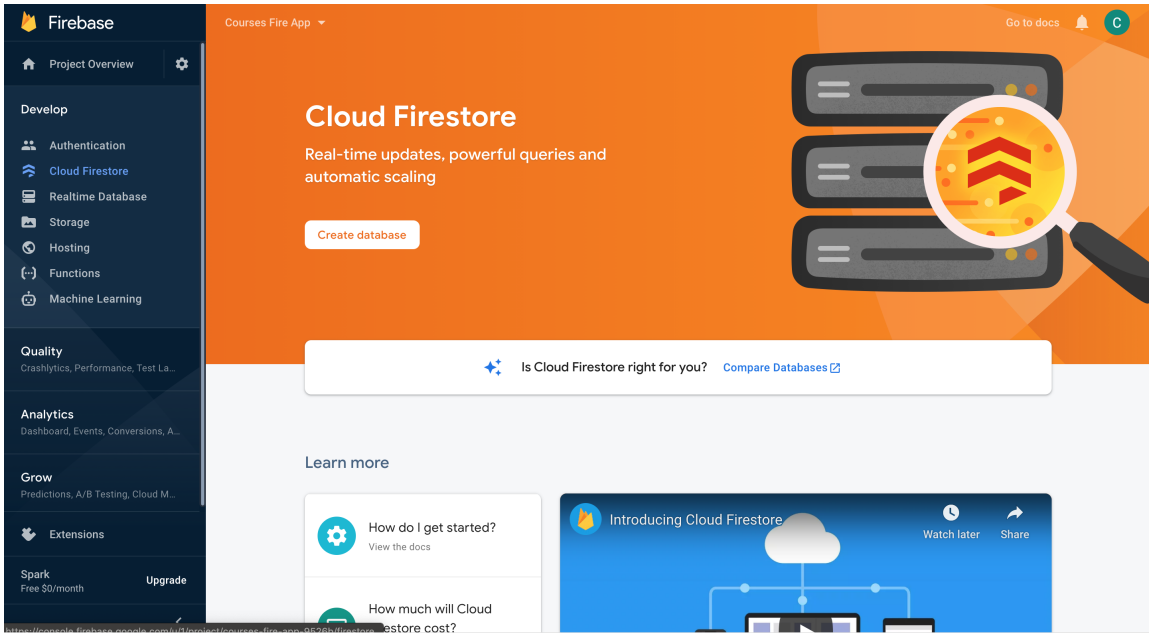
Pick a name for the web app.



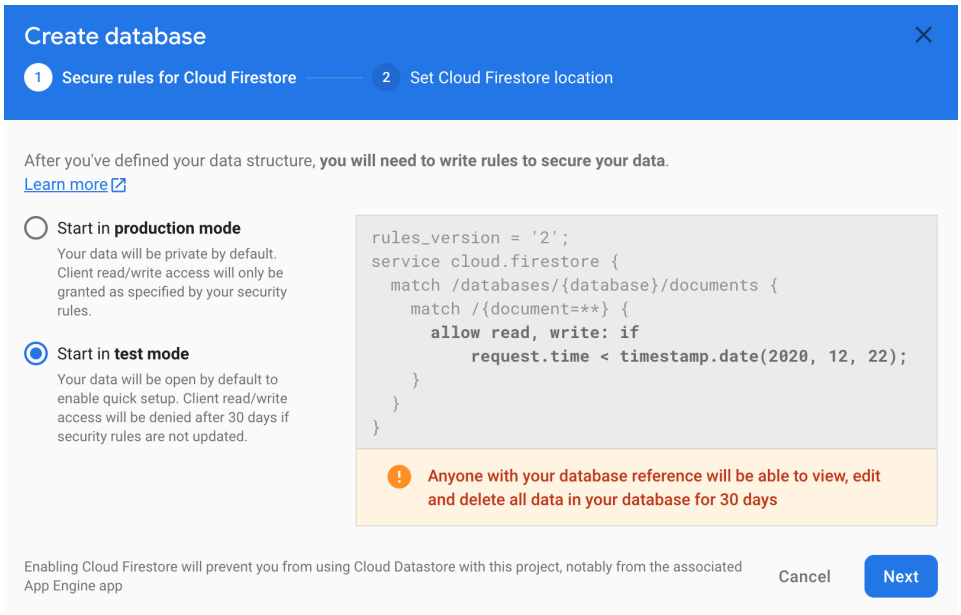
You will then see the config data needed by the Firebase javascript SDK. Take a note of this, as you will need it later whey you configure your Vue app to be able to talk to Firebase.



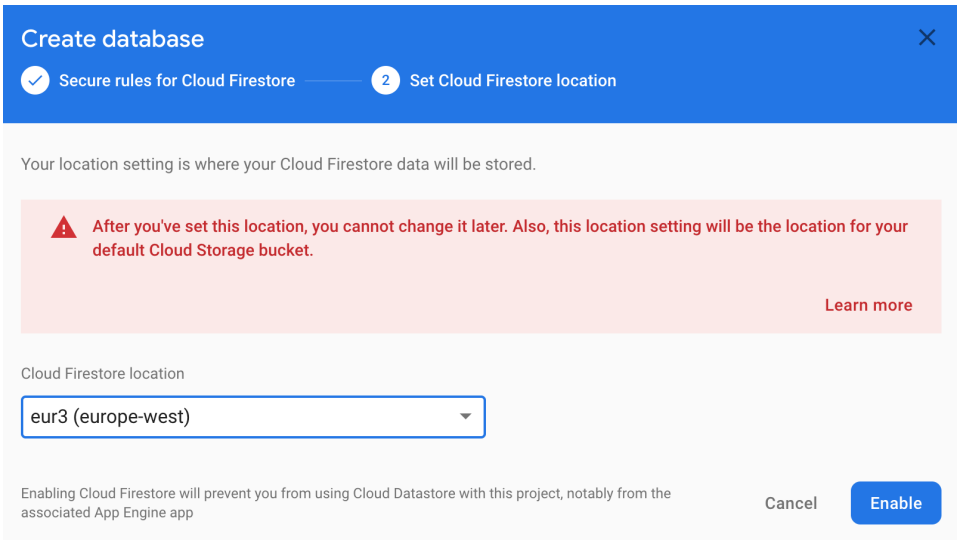
Go to the Cloud Firestore menu item on the left under Develop, and create a new Cloud Firestore database for our project.



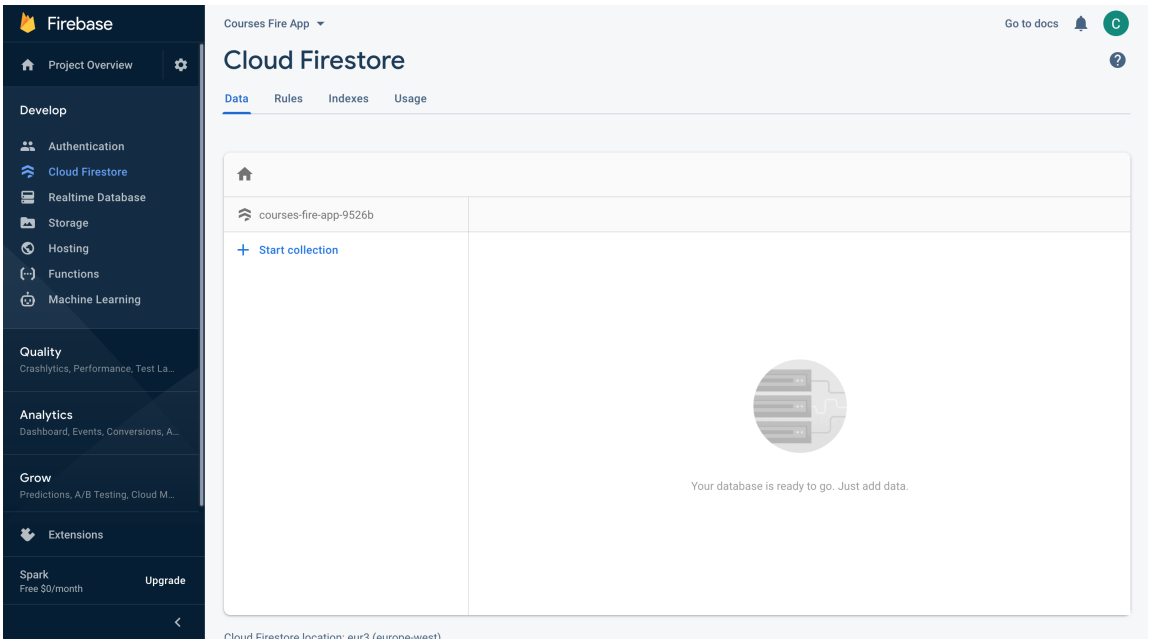
When prompted for security rules, select “Start in test mode”.



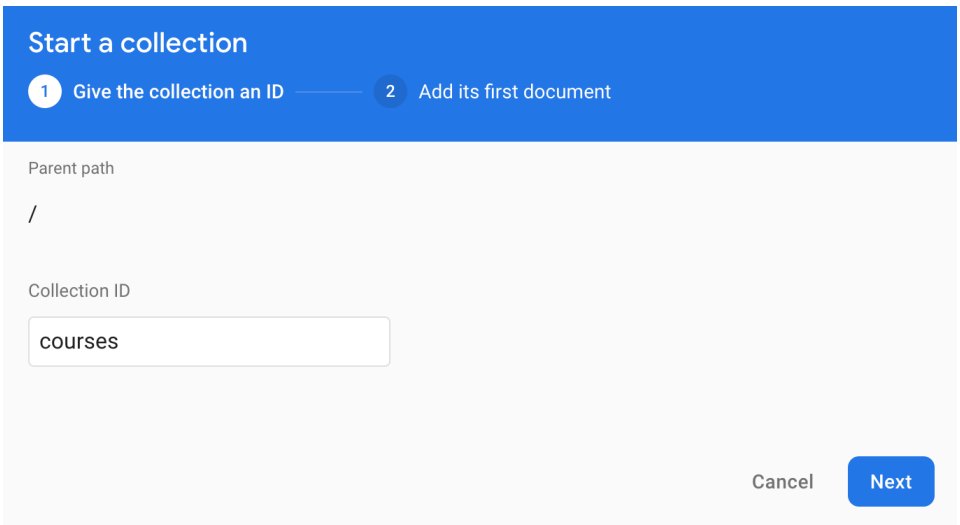
Since we are in Europe, let's pick a European datacenter location.



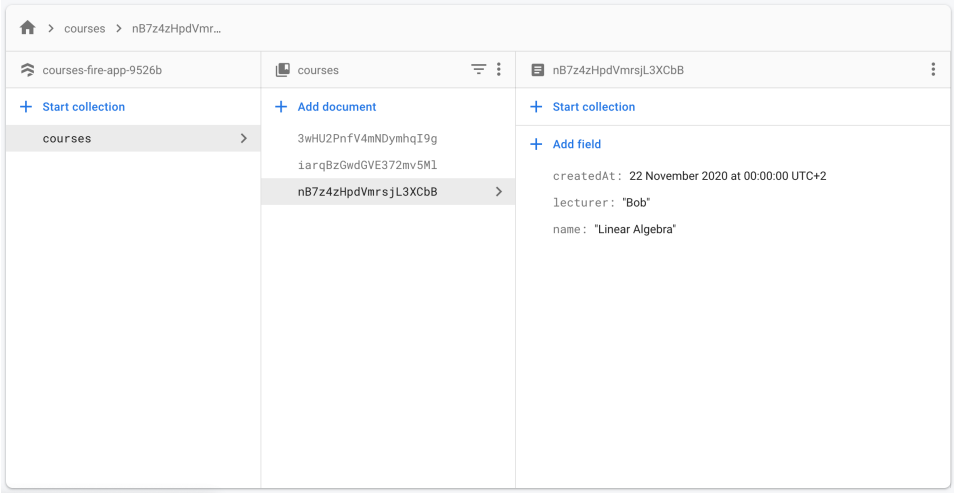
You will then see the Data inspector.



Use it to create a new collection called “courses”.

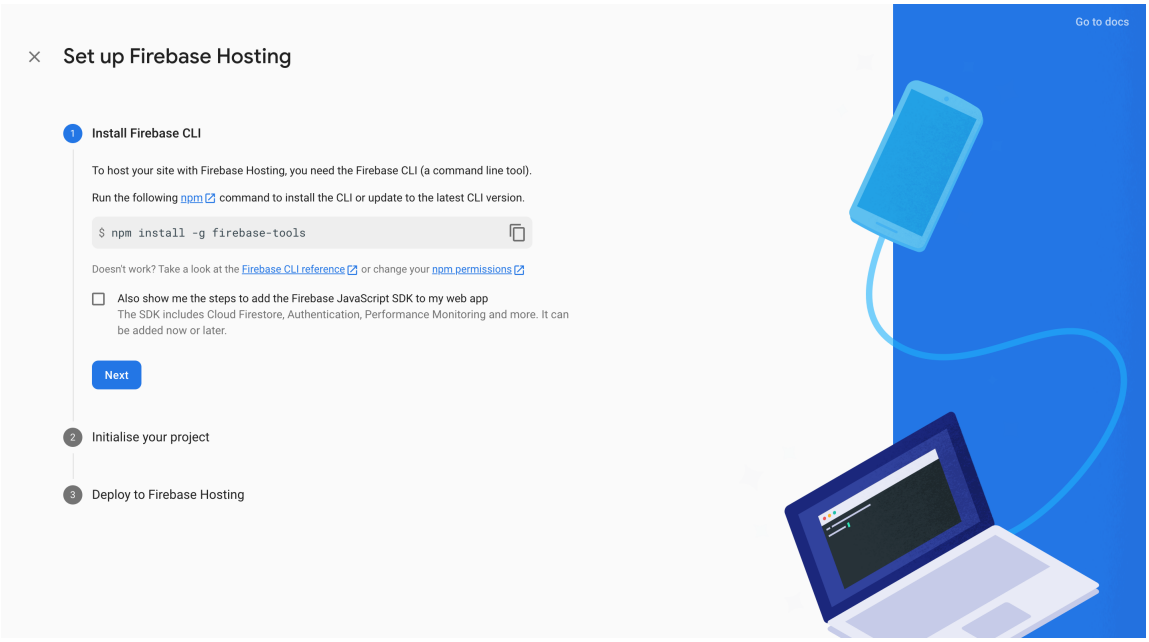


Then, add a few documents in our collection representing the courses. Make sure to include these 3 fields for all courses documents: name (string), lecturer (string), createdAt (timestamp).



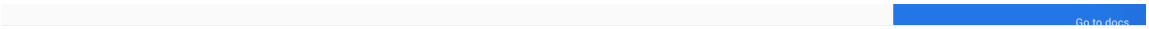
Now that we've finished setting up our real-time database, let's also configure Hosting for our Vue JS app files. Go to the Hosting menu item on the left and click “Get started”.

You will then have to install firebase tools on your machine using NPM. We will need these later.



For now it's easier if you skip steps 2 and 3. Come back to these later after you've created your Vue JS project (as described in the next section).

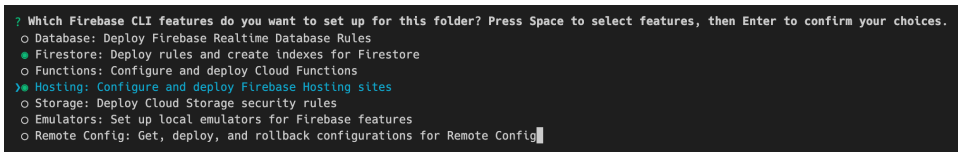
When you run “firebase init” and “firebase deploy” make sure you are within your Vue project directory.



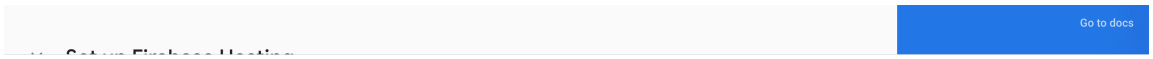
When running “firebase init” you will be prompted to select a public directory. The default for a Vue JS is dist so use that instead of the public (the default option).

```
=== Hosting Setup

Your public directory is the folder (relative to your project directory) that
will contain Hosting assets to be uploaded with firebase deploy. If you
have a build process for your assets, use your build's output directory.

? What do you want to use as your public directory? dist
```

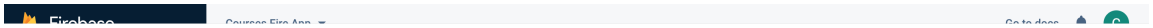
You will also be prompted to select what CLI features to enable. Choose the two we need: Firestore and Hosting.



Finally, before you deploy, make sure that your public folder (dist) contains some files before you deploy. For Vue JS to populate this directory you need to run "npm run build".



Once hosting is setup you will see this screen which includes your new app domain.



3. Setting up a Vue JS app

Let's setup a new Vue JS app for our course list. First, make sure you've got vue-cli installed globally.

```
$ npm install -g vue-cli
```

Now generate a new Vue app using Webpack as the build tool and give your app a name: "courses-fire-app". Then change into the project's directory.

```
$ vue init webpack courses-fire-app
[...]
$ cd courses-fire-app
```

Now install Bootstrap-Vue [https://bootstrap-vue.js.org/] to make our app look prettier with Twitter Bootstrap [http://getbootstrap.com/] styles and components.

```
$ npm install bootstrap-vue --save
```

Great, now open up the project with a code editor like VS Code [https://code.visualstudio.com/].

```
$ code .
```

Open `src/main.js` and add the following code to enable Bootstrap:

```
// imports at the top
import BootstrapVue from 'bootstrap-vue'
import 'bootstrap/dist/css/bootstrap.css'
import 'bootstrap-vue/dist/bootstrap-vue.css'
...
// register BootstrapVue
Vue.use(BootstrapVue)
...
```

Update the logo from the Vue JS logo to the Startup Engineering logo. Do this by opening `src/App.vue` and modifying the `<img>` tag inside the template like so:

```
<template>
  <div id="app">
    <img src="https://ocw.cs.pub.ro/courses/res/sigla_se.png" width="150">
    <router-view/>
  </div>
</template>
```

Also, let's update the default component template in `src/components/HelloWorld.vue` so that we get a simple heading under which we will list our courses later on.

```
<template>
  <div class="container">
    <h1 class="my-3">CS Courses Catalog</h1>
    <!-- TODO: courses list table -->
  </div>
</template>
```

Now try running the app locally. You should see the SE course logo and the “CS Courses Catalog” title.

```
$ npm run dev
[...]
```

After the app starts, open up a browser and point it at `http://localhost:8080` [`http://localhost:8080`].

4. Connecting the app to Cloud Firestore

In order to connect our app to the Firebase backend we need some additional NPM modules: Firebase [https://www.npmjs.com/package/firebase] and the latest Vue Fire [https://www.npmjs.com/package/vuefire] (hence the @next). Let's install these (stop the dev server first if you have it running from the last step):

```
$ npm install firebase vuefire@next --save
```

Now let's update `main.js` in order to connect our Vue app to the Firebase backend:

```
// imports at the top
import { firestorePlugin } from 'vuefire'
```

```
import firebase from 'firebase/compat/app'
import 'firebase/compat/firestore'
...

// enable Vue Fire
Vue.use(firestorePlugin)

// provide connection details to firebase (you should have these from the Firebase console at step 2 above)
firebase.initializeApp({
  apiKey: '...',
  authDomain: '...',
  databaseURL: '...',
  projectId: '...',
  storageBucket: '...',
  messagingSenderId: '...'
})

// export the firebase connection
export const db = firebase.firestore()
...
```

Now we can make use of the Firebase connection to render the courses on our page. We will use a Bootstrap styled table for this.

First, update the code/script section for our `src/components/HelloWorld.vue` component:

```
<script>
import { db } from '../main' // notice the firebase database connection import

export default {
  name: 'HelloWorld',
  data () { // here we provide the initial values for our data model: courses, name, lecturer
    return {
      courses: [], // list of course objects
      name: '', // course name form field
      lecturer: '' // lecturer name form field
    }
  },
  firestore () { // this method will get called automatically by VueFire in order to update the courses property of our data model
    return {
      courses: db.collection('courses').orderBy('createdAt')
    }
  },
  methods: {
    addCourse () { // method to add a new course to our Firebase collection
      db.collection('courses').add({
        name: this.name,
        lecturer: this.lecturer,
        createdAt: new Date()
      })
      this.name = ''
      this.lecturer = ''
    },
    deleteCourse (id) { // method that deletes a course from our Firebase collection by id
      db.collection('courses').doc(id).delete()
    }
  }
}
</script>
```

And now let's update the template section of the `HelloWorld.vue` component such that we render the courses in a table. The table also provides buttons to delete courses and a form to add a new course at the bottom.

```
<template>
  <div class="container">
    <h1 class="my-3">CS Courses Catalog</h1>
    <table class="table text-left">
      <thead>
        <tr>
          <th scope="col">#</th>
          <th scope="col">Course</th>
          <th scope="col">Lecturer</th>
          <th scope="col" class="text-right">Actions</th>
        </tr>
      </thead>
```

```
      <tbody>
        <tr v-for="(course, idx) in courses" :key="idx">
          <th scope="row">{{ idx + 1 }}</th>
          <td>{{ course.name }}</td>
          <td>{{ course.lecturer }}</td>
          <td class="text-right">
            <button title="Delete course" class="btn btn-danger" v-on:click="deleteCourse(course.id)">x</button>
          </td>
        </tr>
      </tbody>
      <tfoot>
        <tr>
          <td>
            <span class="text-secondary">{{ courses.length + 1 }}</span>
          </td>
          <td>
            <input type="text" v-model="name" placeholder="Course name" class="form-control">
          </td>
          <td>
            <input type="text" v-model="lecturer" placeholder="Lecturer name" class="form-control">
          </td>
          <td class="text-right">
            <button title="Add course" class="btn btn-success" v-on:click="addCourse()" :disabled="!name || !lecturer">+</button>
          </td>
        </tr>
      </tfoot>
    </table>
  </div>
</template>
```

Great! We are almost done. Let's now fire up the app.

```
$ npm run dev
[...]
```

If you spot any issues, use your browser's JS console and the Firebase web console to debug your app.

5. Deploying the app on Firebase Hosting

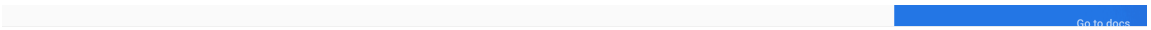
We have managed to create a functional Vue JS app that can talk to our Cloud Firestore database hosted in the cloud. But in order to get the app to our users we also have to put the Vue JS app files on a server. Fortunately firebase also provides a hosting solution for this.

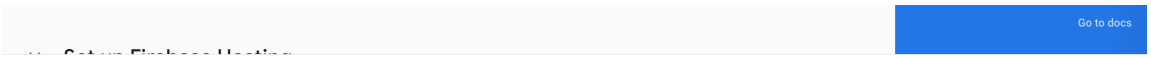
First, let's compile a production version of our app.

```
$ npm run build
[...]
```

This will build an optimised version of the app under a directory called `dist` in the root of the project.

Our task now is to get the contents of this directory hosted in the Firebase cloud for our users to access. Remember from step 2, that we have already configured Firebase Hosting in the web console and we have already installed the firebase CLI tools.





Make sure you are in your project's project's root.

Login with your firebase/google account using the firebase CLI tool:

```
$ firebase login
[...]
```

Then, run the init command and follow the instructions. This will create a `firebase.json` file and a `.firebaserc` file to store info about how your project should be hosted on Firebase. Remember to change the public directory to `dist` and to add Firestore and Hosting CLI features.

```
$ firebase init
[...]
```

Here are is a sample of a `firebase.json` file to help guide you (note the public directory is set as dist):

```
{
  "firestore": {
    "rules": "firestore.rules",
    "indexes": "firestore.indexes.json"
```

```
  },
  "hosting": {
    "public": "dist",
    "ignore": [
      "firebase.json",
      "**/.*",
      "**/node_modules/**"
    ]
  }
}
```

And here's an example of `.firebaserc` (notice this just saves your Firebase project id):

```
{
  "projects": {
    "default": "courses-fire-app-6bb3e"
  }
}
```

Now you can simply run the deploy command and your app will get deployed, hosted, and managed by the Firebase cloud:

```
$ firebase deploy
[...]
```

After it finishes, you will be able to access your app at a URL similar to: https://courses-fire-app-6bb3e.firebaseapp.com [https://courses-fire-app-6bb3e.firebaseapp.com]

6. Bonus - Edit course feature

Modify the app so that a user is able to edit a specific course. You could for instance add an edit button that when clicked causes the row to show input fields and a save button.

7. Bonus - Password protect the course catalog

Make use of Firebase's Authentication service [https://firebase.google.com/products/auth/] to restrict access to list edits: only logged in users should be able to edit the list, otherwise the list should be read-only.

Feedback

Please take a minute to fill in the **feedback form [https://forms.gle/NuXCJktudGzf4rLg6]** for this lab.

se/labs/06.txt · Last modified: 2021/11/15 21:56 by cosmin.dumitrache