myFields

Kansas State Extension Platform

myFields Documentation for Developers

Kansas State University | Spring 2019

Table of Contents

Introduction

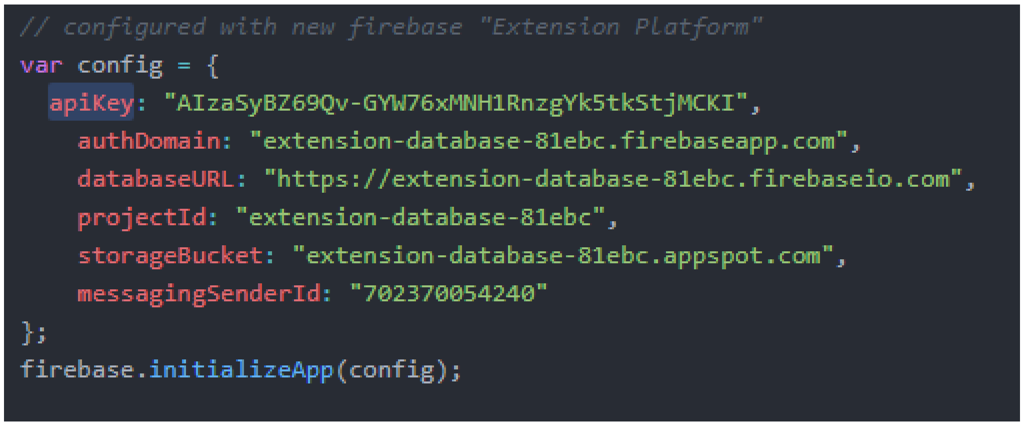
The myFields application was developed by students in Kansas State's Computer Science department for the Kansas State Entomology Department, with the intent of helping farmers in the community more effectively understand and diagnose illnesses in their crops. This documentation has been made in order to help students maintain the current software as well as potentially provide a guide for a team developing another application into the Extension Platform Suite of Applications which currently consists of myFields and the Sorghum App. Working with the myFields app requires work with Firebase, Linux, and Java Script. We will focus on focusing on Firebase and required Linux commands in this document.

Working with Firebase

Firebase is used with the application to provide authentication, data warehousing, image storage, and hosting. If you need access to the Firebase, contact the old extension platform team.

CONNECTING TO FIREBASE

Connecting to Firebase is simple. All you must do is create an instance of the connection and initialize through Firebase. In myFields, this is done in the index.js file. A screenshot of this being done is shown below.

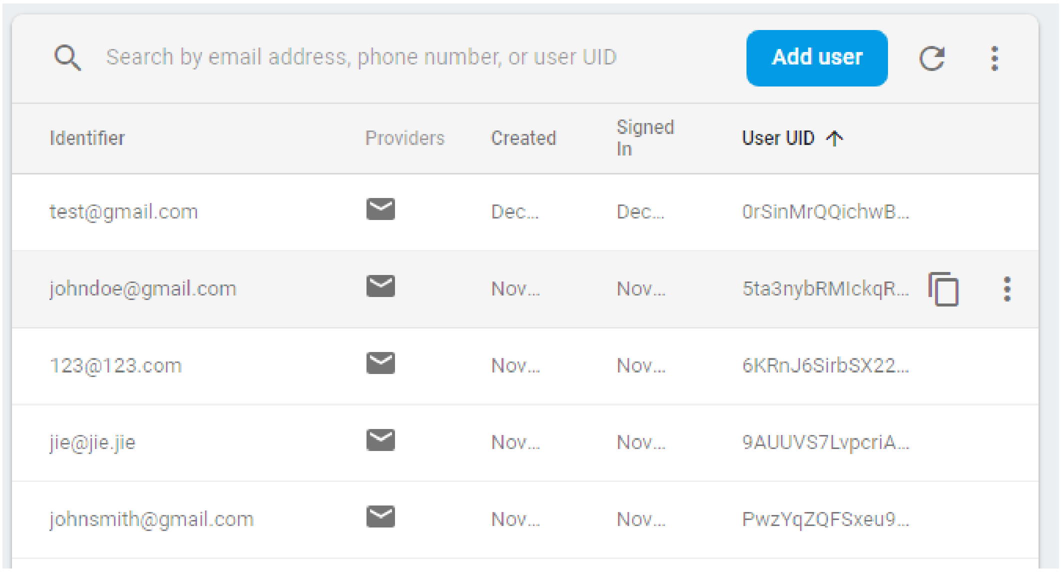


```
// configured with new firebase "Extension Platform"
var config = {
  apiKey: "AIzaSyBZ69Qv-GYW76xMNH1RnzgYk5tkStjMCKI",
    authDomain: "extension-database-81ebc.firebaseapp.com",
    databaseURL: "https://extension-database-81ebc.firebaseio.com",
    projectId: "extension-database-81ebc",
    storageBucket: "extension-database-81ebc.appspot.com",
    messagingSenderId: "702370054240"
};
firebase.initializeApp(config);
```

Connecting another application to the Extension Platform database can be done by modelling the above. Instructions on configuring an application is JavaScript, or in any compatible language, can be found by clicking "+ Add App" on the Extension Platform Firebase homepage (For JavaScript, it will just show the code on the previous page).

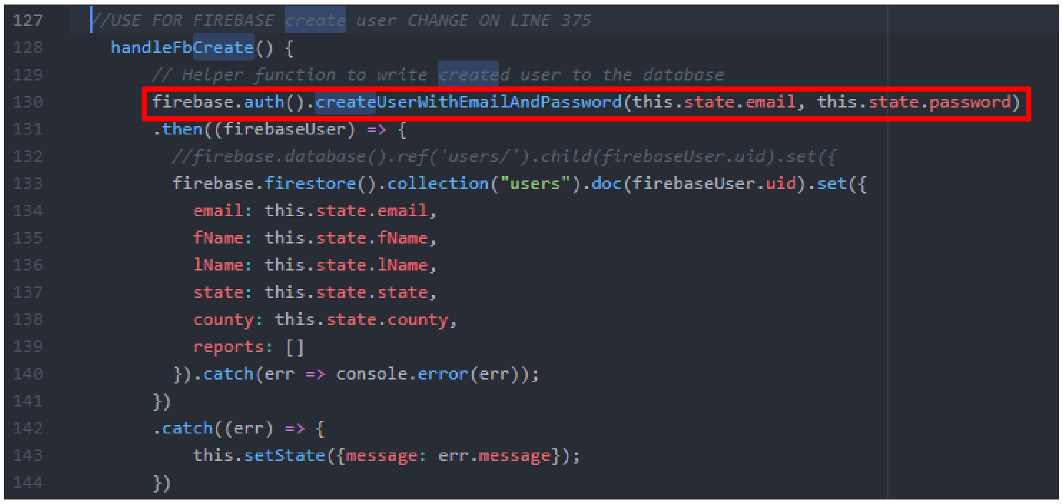
AUTHENTICATION

Firebase houses all the past users who have created accounts in one of the Extension Platform applications. When a new account is made in the app, new user information is stored here.



Identifier	Providers	Created	Signed In	User UID ↑
test@gmail.com	✉	Dec...	Dec...	0rSinMrQQichwB...
johndoe@gmail.com	✉	Nov...	Nov...	5ta3nybRMIckqR...
123@123.com	✉	Nov...	Nov...	6KRnJ6SirbSX22...
jie@jie.jie	✉	Nov...	Nov...	9AUUVS7LvpcriA...
johnsmith@gmail.com	✉	Nov...	Nov...	PwzYqZQFSxeu9...

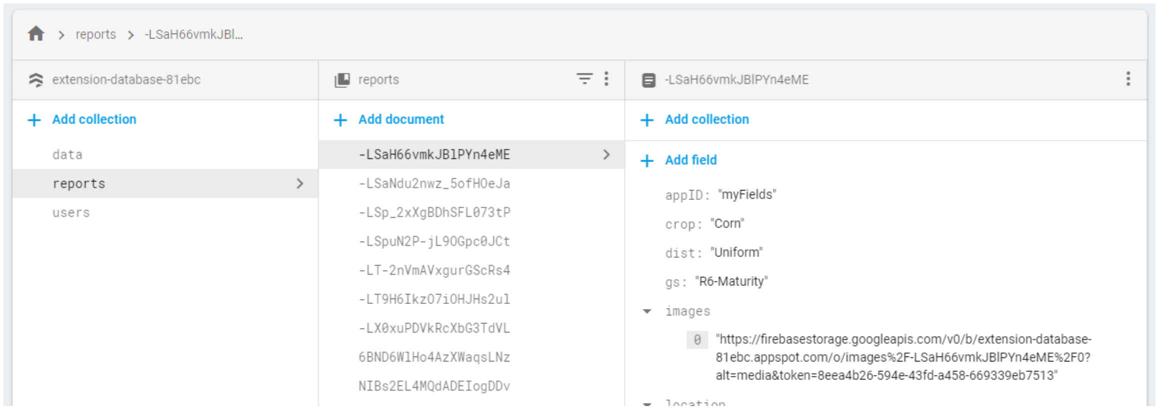
There is no need to add or remove users in this console except perhaps for testing purposes. These accounts are made in the code shown here from \src\components\login-form\login-form.js, the code shown also stores other user info in the database.



```
//USE FOR FIREBASE create user CHANGE ON LINE 375
handleFbCreate() {
    // Helper function to write created user to the database
    firebase.auth().createUserWithEmailAndPassword(this.state.email, this.state.password)
    .then((firebaseUser) => {
      //firebase.database().ref('users/').child(firebaseUser.uid).set({
      firebase.firestore().collection("users").doc(firebaseUser.uid).set({
        email: this.state.email,
        fName: this.state.fName,
        lName: this.state.lName,
        state: this.state.state,
        county: this.state.county,
        reports: []
      }).catch(err => console.error(err));
    })
    .catch((err) => {
        this.setState({message: err.message});
    })
```

DATA WAREHOUSING

Data is stored in the Cloud Firestore Database. Cloud Firestore was chosen because it allows for a more seamless connection for any user who attempts to make a report outside without cellular data. The structure of the database flows from collection to document to collection or fields. You can see an example of this bellow:

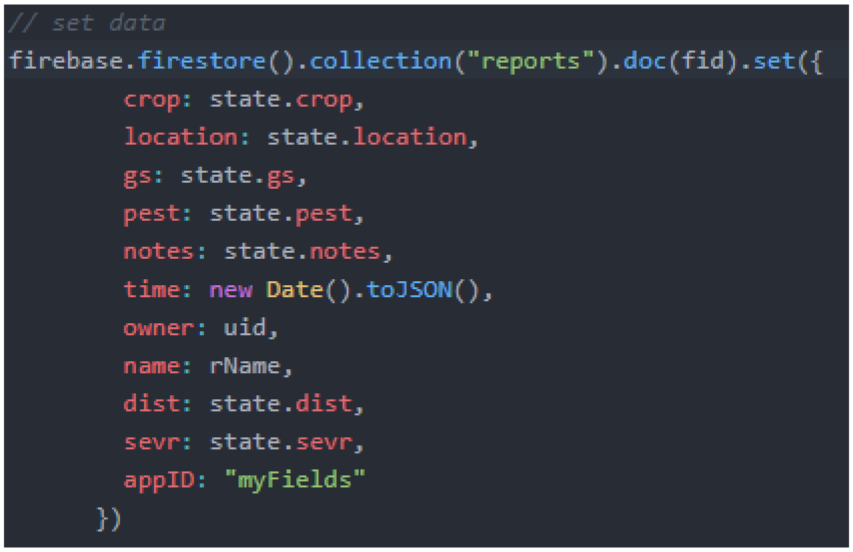


```
data
reports
users

-LSaH66vmkJB1PYn4eME
-LSaNdu2nwz_5ofH0eJa
-LSp_2xXgBDhSFL073tP
-LSpuN2P-jL90Gpc0JCt
-LT-2nVmAVxgurGScRs4
-LT9H6Ikz07i0HJHs2ul
-LX0xuPDVkRcXbG3TdVL
6BND6W1Ho4AzXWaqsLNz
NIBs2EL4MQdADEIogDDv

appID: "myFields"
crop: "Corn"
dist: "Uniform"
gs: "R6-Maturity"
images
  0 "https://firebasestorage.googleapis.com/v0/b/extension-database-81ebc.appspot.com/o/images%2F-LSaH66vmkJBlPYn4eME%2F0?alt=media&token=8eea4b26-594e-43fd-a458-669339eb7513"
location
```

As you can see, there are three 1st level collections each with a number of documents. These documents can each contain either another set of collections, or fields. In our case, the set of collections each represent a report and the fields are information regarding the report. This information can be added to, referenced, or even changed. An example of a new report being added is seen bellow. The complete algorithm is too long to show, however how you do this insert is shown on the next page from \src\components\reports\reports.js.

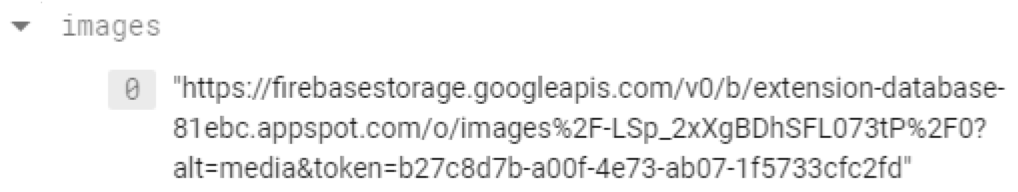


```
// set data
firebase.firestore().collection("reports").doc(fid).set({
        crop: state.crop,
        location: state.location,
        gs: state.gs,
        pest: state.pest,
        notes: state.notes,
        time: new Date().toJSON(),
        owner: uid,
        name: rName,
        dist: state.dist,
        sevr: state.sevr,
        appID: "myFields"
      })
```

As you can probably tell, the .set() function is used to add a new report. However, to pull a report you will use the .get() function, and to edit you will use the .update() function. Examples of these can be seen in \src\components\showReport\showReport.js and in myFieldReporterApp\src\components\editReport\editReport.js respectively.

IMAGE STORAGE

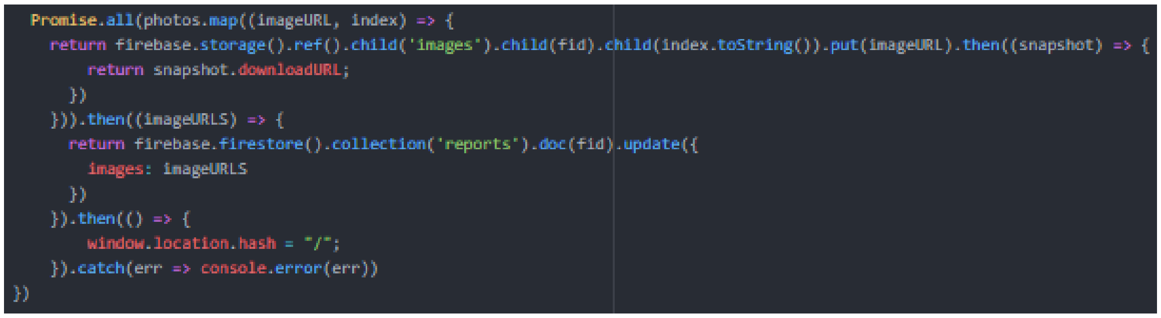
Firebase has an easy way of string the images the myFields uses in their reports. These reports are stored with a URL, that is needed in order to access an image in the future. We have stored this URL with the corresponding report. This is shown below.



```
▾ images

    0  "https://firebasestorage.googleapis.com/v0/b/extension-database-
       81ebc.appspot.com/o/images%2F-LSp_2xXgBDhSFL073tP%2F0?
       alt=media&token=b27c8d7b-a00f-4e73-ab07-1f5733cfc2fd"
```

You can view these images in the Firebase console, but this is not important for the purpose of this document. Storing these images as well as their URL is done asynchronously. This algorithm loops through the uploaded pictures and uploads them into storage, and then their URL into the database. This is done in \src\components\reports\reports.js and a code snippet is shown below.

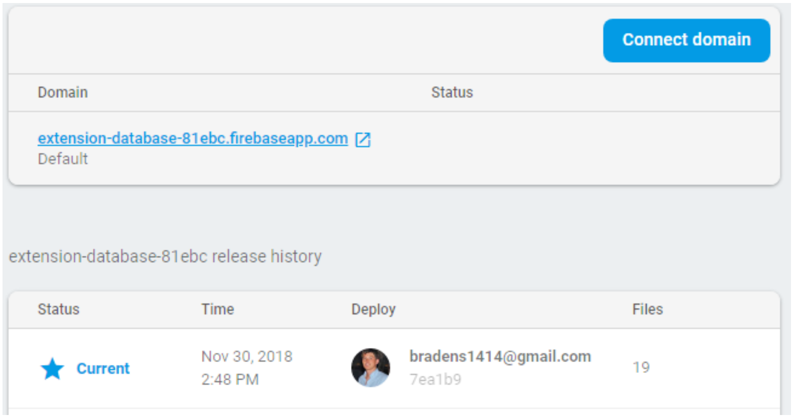


```
  Promise.all(photos.map((imageURL, index) => {
    return firebase.storage().ref().child('images').child(fid).child(index.toString()).put(imageURL).then((snapshot) => {
        return snapshot.downloadURL;
      })
    })).then((imageURLS) => {
      return firebase.firestore().collection('reports').doc(fid).update({
        images: imageURLS
      })
    }).then(() => {
        window.location.hash = "/";
    }).catch(err => console.error(err))
})
```

Showing these images in html is done by pulling the URLs the same way that you would any other information from Cloud Firestore, with the .get() function. These URLs can then be used as the source for an image in html and displayed in the myFields application.

HOSTING

Hosting can be done through the Firebase Application. The domain of the hosted Application can be found in the hosting section of the Firebase Console. Here you can also see the previous pushes you have made to the cite.



Domain	Status
extension-database-81ebc.firebaseapp.com Default	

extension-database-81ebc release history

Status	Time	Deploy	Files
★ Current	Nov 30, 2018 2:48 PM	bradens1414@gmail.com 7ea1b9	19

You can see how to host the application using Linux commands in the upcoming Linux commands section.

Working with Linux

You will be needing knowledge of a variety of Linux tools in order to work with the myFields application. You will be needing to use git, run the application in a local test environment, and host the application.

PUSHING AND PULLING WITH GIT

In order to pull using git, you will first need to navigate into where you have the project stored. You will then run the following command.

```
git pull origin master
```

When you want to push your changes, you will run the commands below. The first command is a repeat of above in order to make sure your code is up to date before pushing.

```
git pull origin master

git stage *

git commit -m "Insert commit message here"

git push origin master
```

At this point you will be prompted to sign into git. Sign in here, and the files that you have changed will be committed into your project.

RUNNING THE PROGRAM LOCALY

In order to make sure your code works before hosting, it is important that you test the program locally. You can do this by running a couple of commands in Linux. It is important to note that some features that utilize external resources may not work properly because you are not providing a secure connection. An example of this is the geolocation. These should work when hosting normally.

```
npm build

npm start
```

You will know that the program is running because you will get messages from the software. Some of these may be warnings but are okay. Now that the program is running you will be able to view it at **http://linux.cis.ksu.edu:3000/** . Feel free to play around with the application here if you have not yet.

HOSTING

When you want to host the application the first time, you will need to use the following command once.

~/node_modules/.bin/firebase init

This will ask load a prompt in the console and ask you a couple of questions. Answer these questions and move on to the next command. You will only need to run the above command once, however will need to run the below commands ever time that you want to push to the live site. It is also important to note that the part of the command in both init and deploy listed as “~/node_modules/.bin/firebase init” may change depending on where you have installed firebase locally.

npm build

~/node_modules/.bin/firebase deploy

Once this is complete, your project is hosted. You can confirm this in the hosting section of the Firebase Console.

Conclusion

This document should help you start working in the Linux environment and with the Firestore database. JavaScript is not included in this documentation because there are better ways to learn this language than from documentation like this. You can find a variety of videos about JavaScript online. Please feel free to edit this document as you work with the myFields application.