Documentation

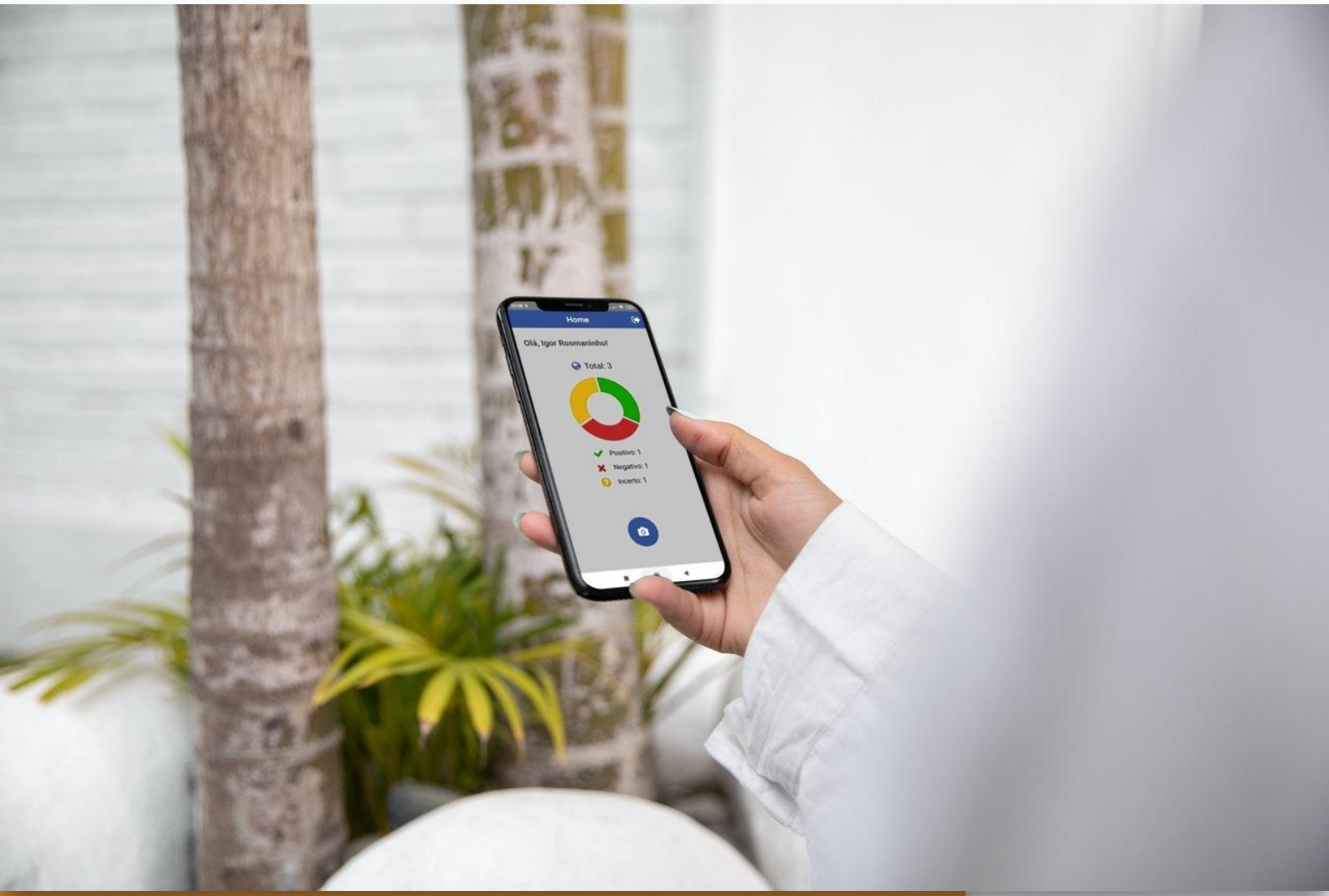


Image Dataset Recorder

Configuration, Installation and Use Guide

IGOR ROSMANINHO |GABRIEL RODRIGUES | EDUARDO ZANCUL

Introduction

This mobile application allows the creation of a cloud dataset of images organized into predefined classes. The system allows the user to capture images with the smartphone camera, classify them according to predefined classes, and upload them to a linked database to create a dataset with the classified images.

In order to use this application, we need to perform a few steps, most of them related to the Firebase configuration. Firebase is a Backend-as-a-Service (BaaS), providing developers with various tools and services to help them develop apps, such as authentication, database, and hosting. So, with the aim of only you and your team having access to your database, with full ownership and confidentiality of the data stored there, it is necessary to create, configure and connect your database, which will be explained in the following steps.

Configuration

First, you must copy the Git repository project into a new local directory.

➢ *git clone https://gitlab.com/igor.rosmaninho/image-dataset-recorder.git*

After that, you need to install all dependencies, running the following command:

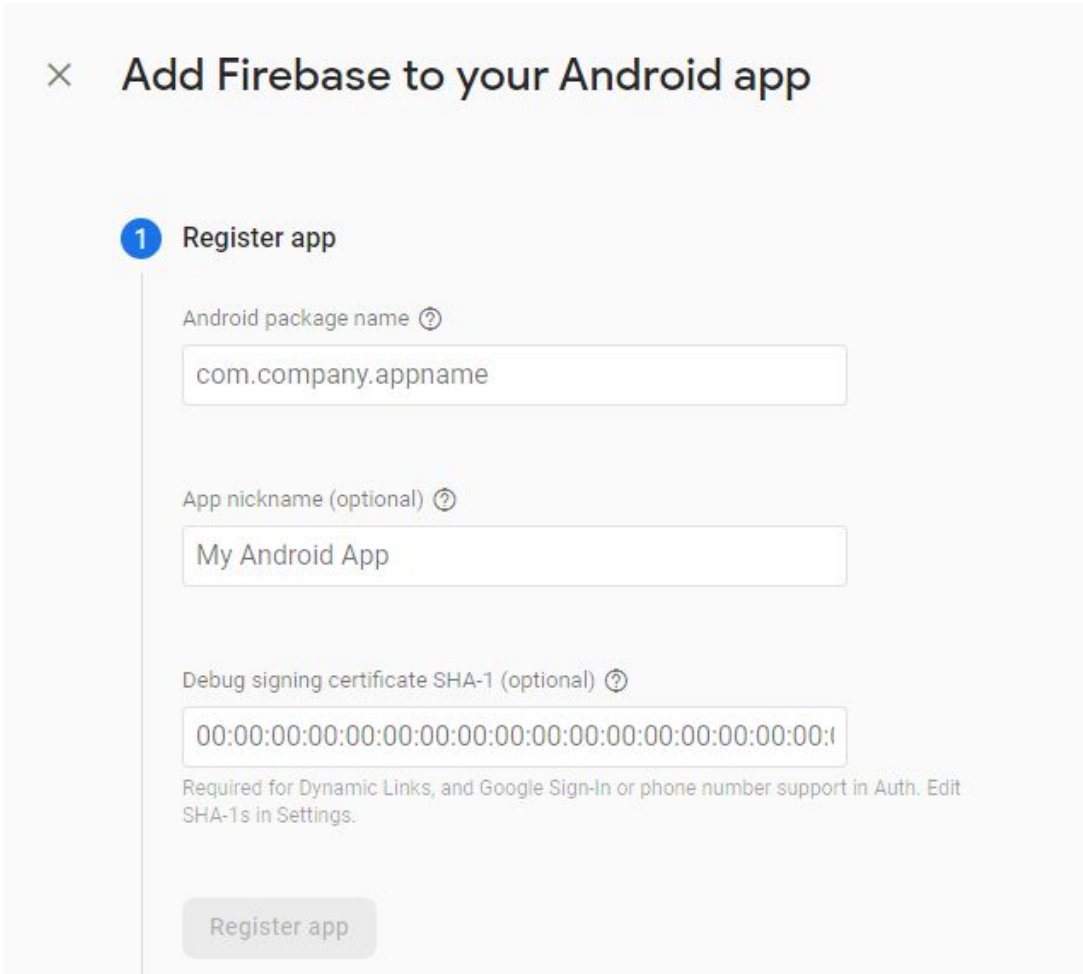
➢ *npm install*

As mentioned before, in order to have your own database and store all images that you will capture, you need to create an Firebase account. You can create your account at this link: https://firebase.google.com/

Configuration

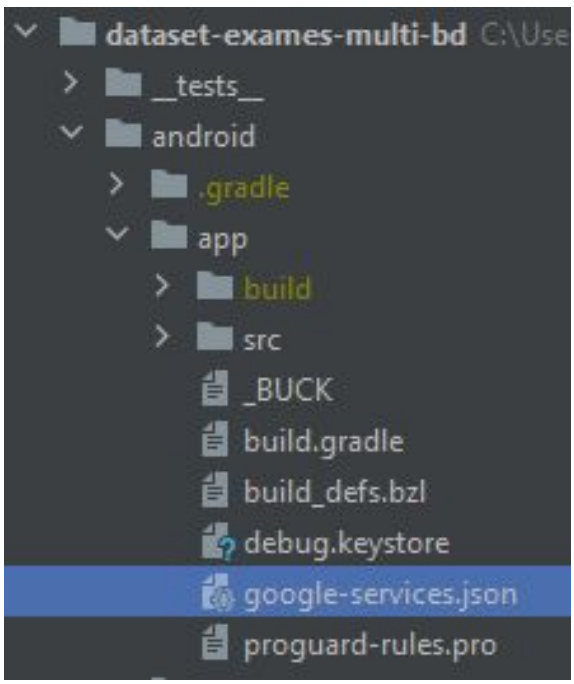
After creating the Firebase account, you need to create a Firebase project.

With the project created, the next step is to add the Firebase to the Android application. To do that, you need to register the app, inserting the Android package name where it is required. Your package name is generally the applicationId in your app-level build.gradle file.



Configuration

After registering the app you need to download the config file (“google-services.json”) and move it into your Android app module root directory.



To allow Firebase on Android to use the credentials, the Google-services plugin must be enabled on the project. You need to verify if two files in the Android directory have the following required code lines.

First, verify if the Google-services plugin is added as a dependency inside of your /android/build.gradle file and if the plugin version is up to date. (you can verify the latest instructions at this link.)

```
buildscript {
  dependencies {
    // ... other dependencies
    classpath 'com.google.gms:google-services:4.3.13'
    // Add me --- /\
  }
 }
```

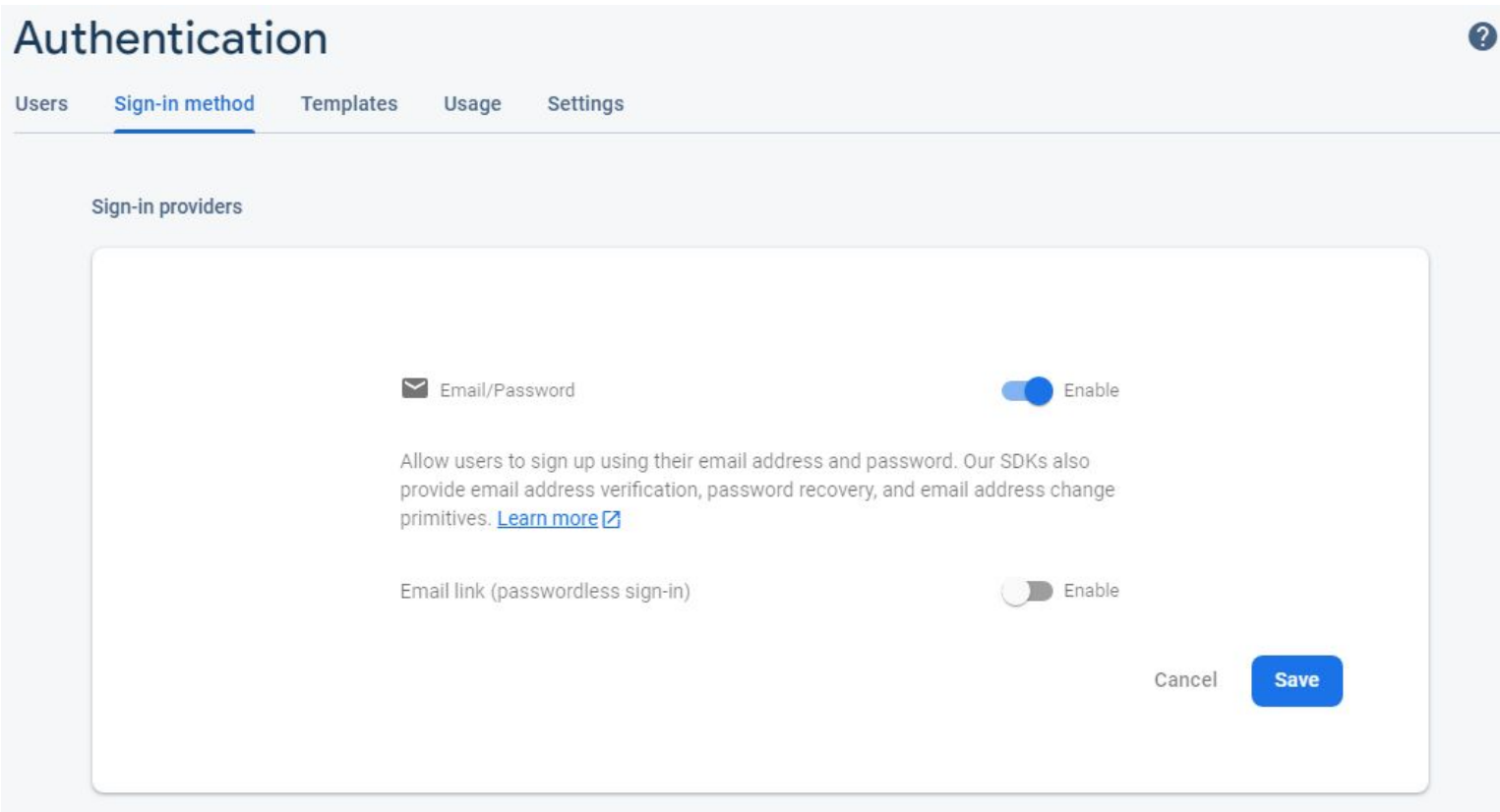
Configuration

Then, verify if the following code lines are added to your /android/app/build.gradle file:

```
apply plugin: 'com.android.application'
apply plugin: 'com.google.gms.google-services'
```

Next, you need to enable and set the security rules of the tools provided by Firebase that you are going to use.

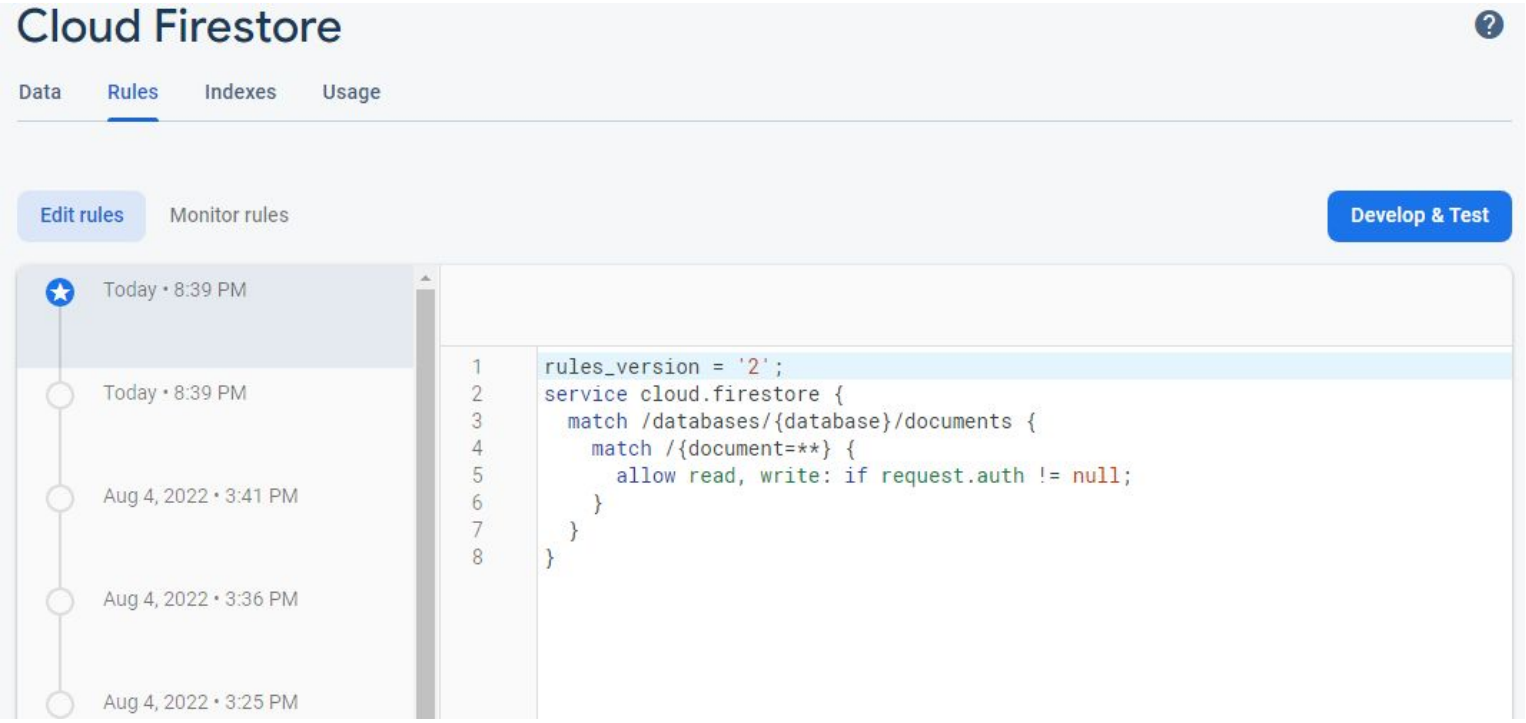
The first of them is the authentication tool. You need to enable the Email/password sign-in method.



Configuration

Another tool that you are going to use is the Firestore Database. In order to use this tool, you need to enable it. Furthermore, it is recommended that you change the security rules to the following:

```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
    match /{document=**} {
      allow read, write: if request.auth != null;
    }
  }
}
```



Configuration

Lastly, it is also recommended that you change the security rules of the Storage tool.

```
rules_version = '2';
service firebase.storage {
  match /b/{bucket}/o {
    match /{allPaths=**} {
      allow read, write: if request.auth != null;
    }
  }
}
```

Once all the above steps have been completed, the React Native Firebase library must be linked to your project, and your application needs to be rebuilt. Run the following command:

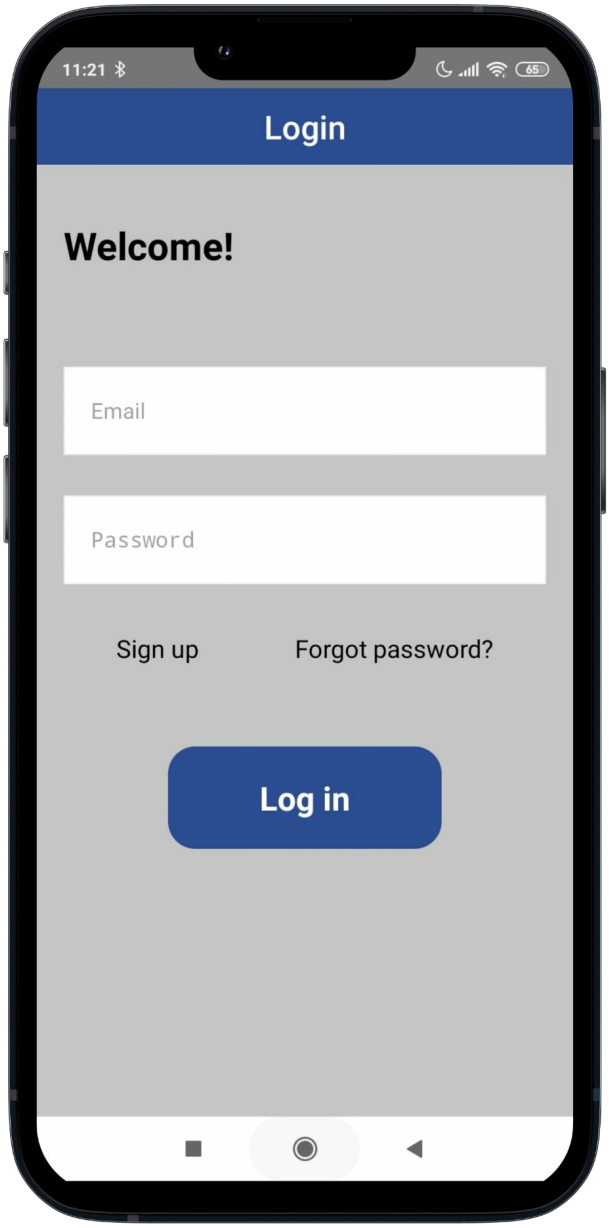
➢ *npx react-native run-android*

User Guide

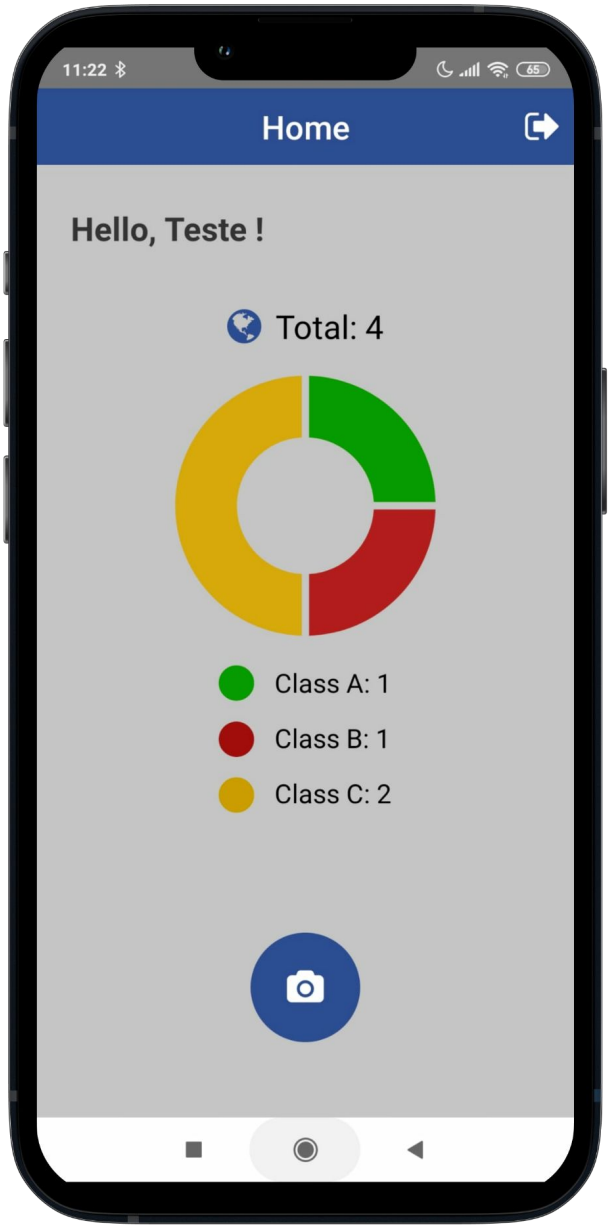
1. When you open the application, the Login screen will appear.

Select ‘Sign up’ and create an account.

This scenario of required login and different users was created so that only authorized people have access to the database and that each inserted image has the identification of the user who added it.



User Guide



2. On the start screen, the graph represents the composition of the dataset in which the images are being stored.

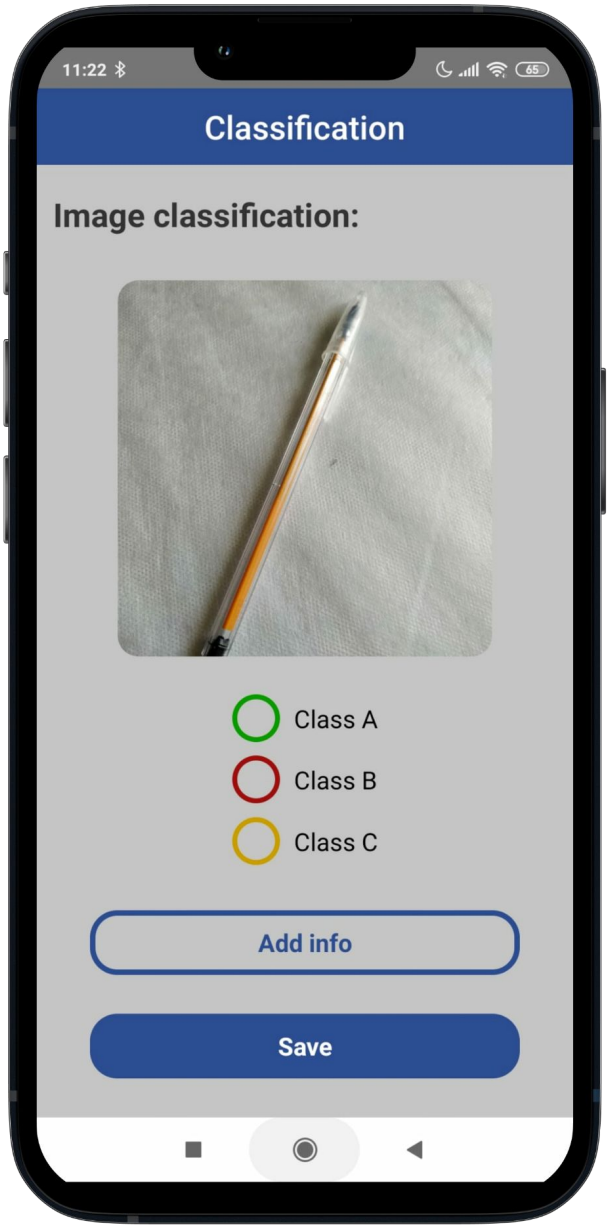
Each color represents a class and its respective quantities in the database.

The button below the graphic triggers the camera.

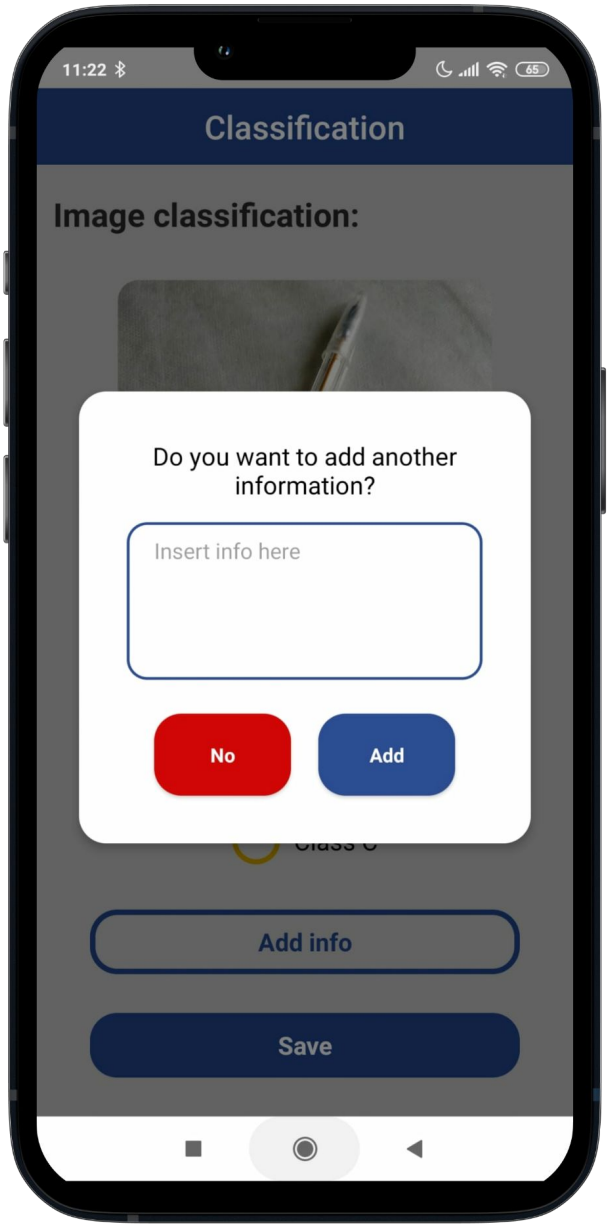
When you take the picture, you can still check it by confirming or taking it again.

User Guide

3. The Classification screen appears after you have confirmed the photo.
There you can sort between one of three options: Class A, Class B, and Class C.



User Guide

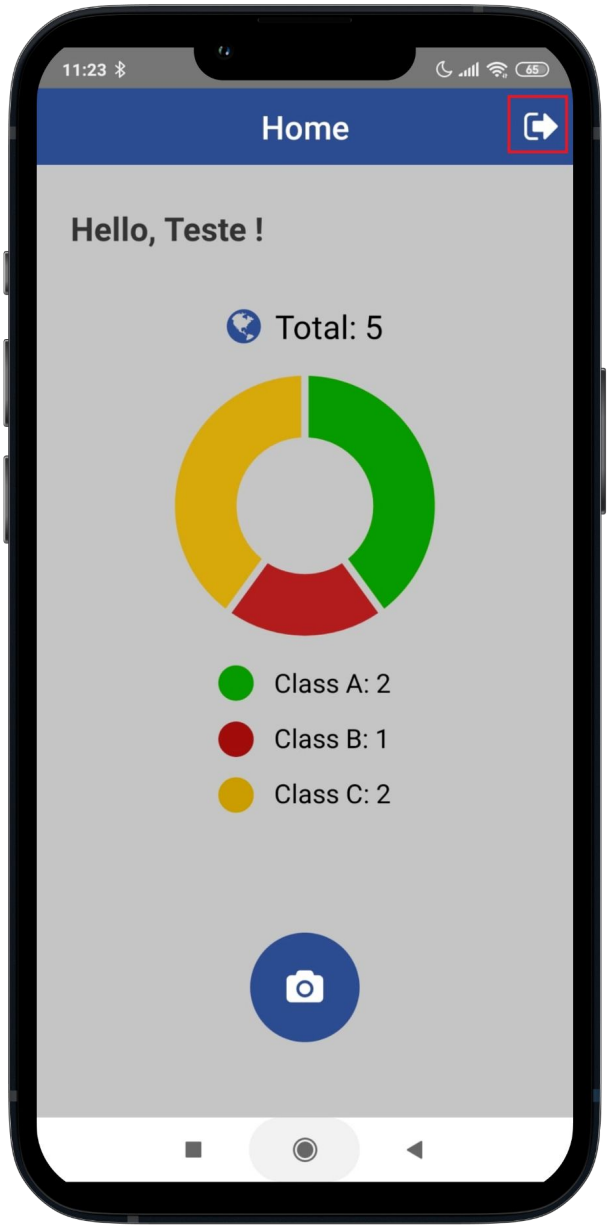


4. Optionally, you can add some comment that you think is pertinent to the record.

Once these steps are completed, save and wait for the upload.

User Guide

5. That's it! Now it is possible to make another classification.

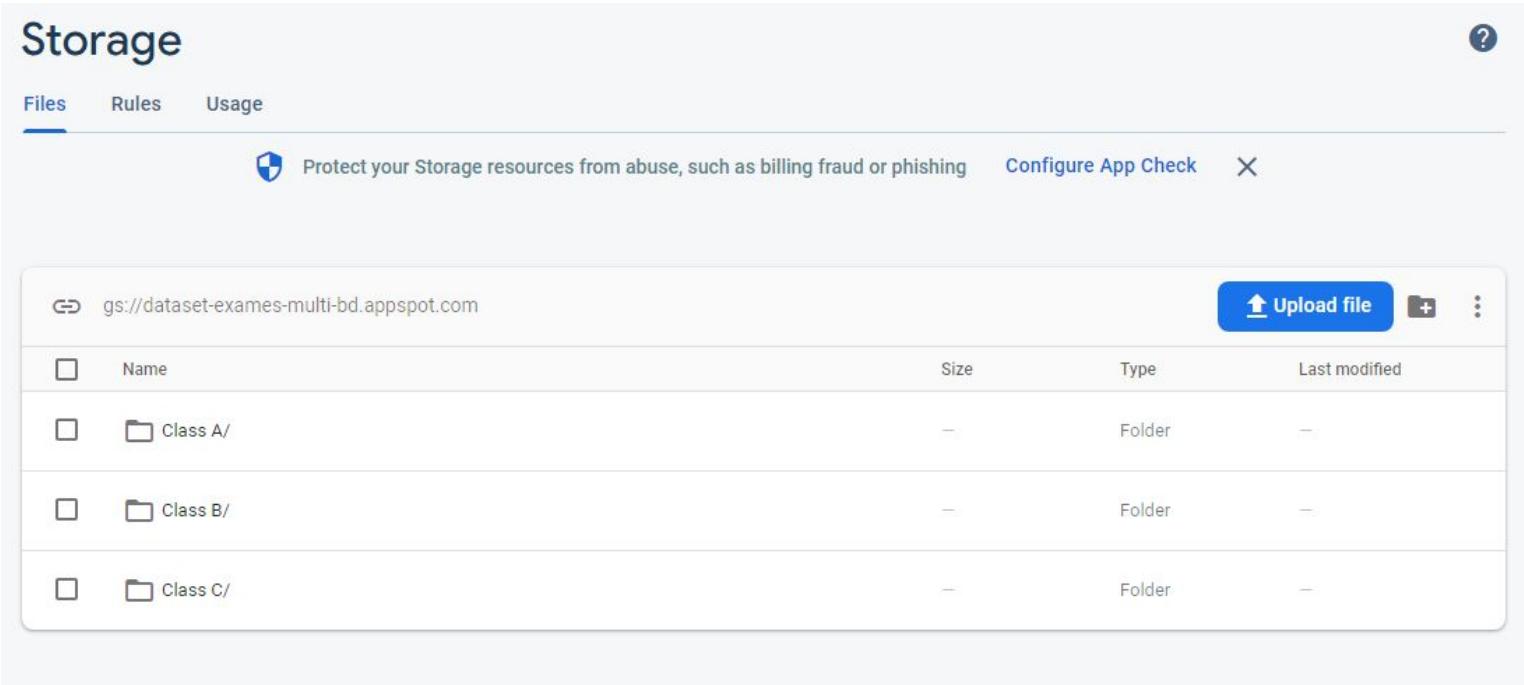


To log out, click the Logout button in the upper right corner of the Home screen.

Database

The database has been hosted on Firebase, so that when you submit a record for upload via the application, the image and its respective metadata are saved.

The record is saved in two steps. The first consists of uploading the image into directories on the Firebase Storage, with the directories separated by classification (Class A, Class B, and Class C) - some metadata is kept in the image itself, namely: class and author.

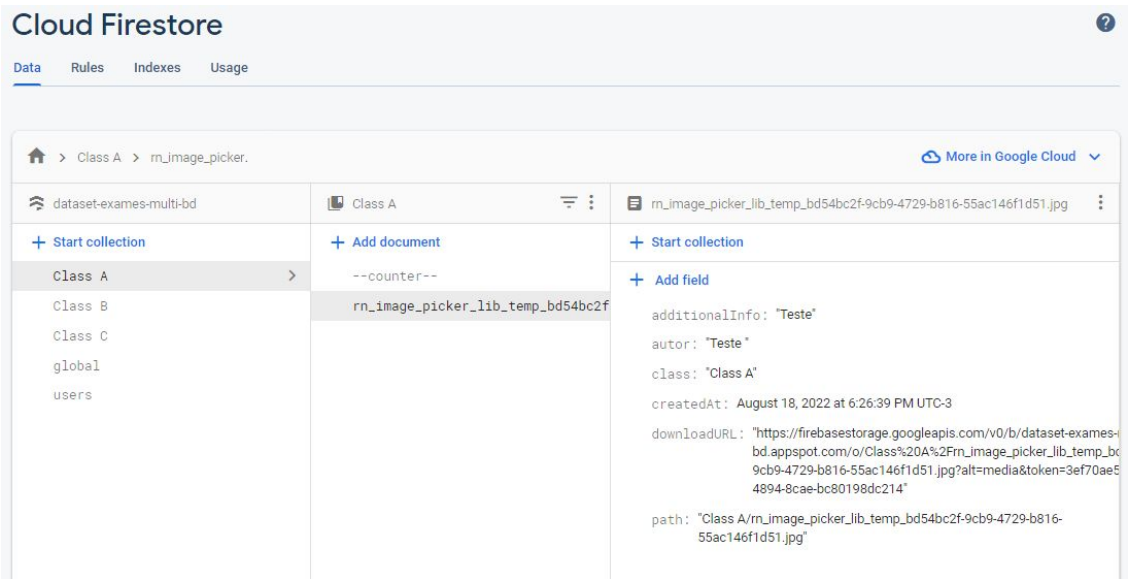


In the second step the registry metadata is saved to the Firestore Database, which is a NoSQL database in which the records are organized into documents and collections. Like the directories with the images in Storage, the collections in Firestore are separated by exam classification - i.e. Class A, Class B and Class C.

Database

For each record are saved:

- ➢ Additional information (String)
- ➢ Author's name (String)
- ➢ Class (String)
- ➢ Date and time of the record (Timestamp)
- ➢ Url of image download (String)
- ➢ Relative path of the image in the directories (String)



The users collection stores some user information, while the global collection contains a counter representing the total and the relative number of uploads already made.

Database

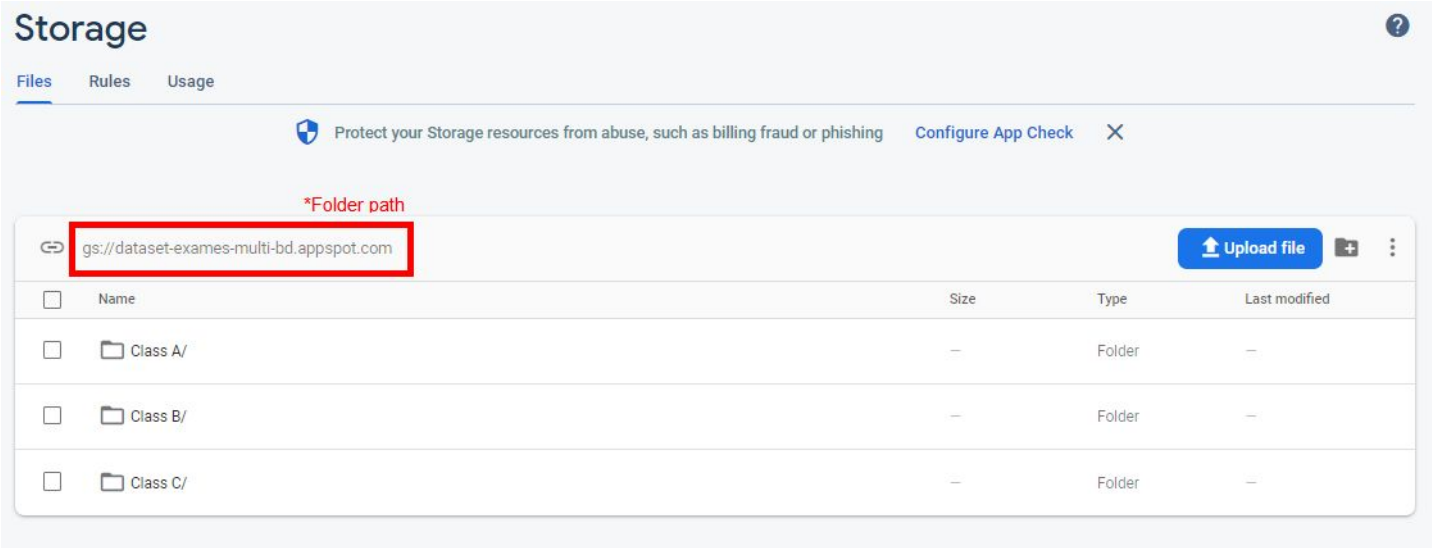
To download the directories with the images, you must first install gsutil, which is an application that allows access to Cloud Storage via command line. The installation is relatively simple and can be done by following the step-by-step instructions in this link.

With the installation complete, the terminal will open in the Google Cloud SDK installation folder, which usually has the following ending:

➢ *C:\...\Google\Cloud SDK>*

You will be requested to login and select the project.

To download the repository, you need to copy the folder path located in the Cloud Storage in the Firebase project.



Database

Then you just need to navigate in the terminal to the folder where you want the repository to be downloaded and run the following command:

➢ *gsutil cp -r **<insert folder path here>**/ .*

Important: the dot at the end of the line is necessary for the command to run correctly!

More information about the available gsutil commands can be found at documentation.