

This project has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement No 643924



D4.8

Demonstrator 3, Configuration App



Copyright © 2018 The EoT Consortium

The opinions of the authors expressed in this document do not necessarily reflect the official opinion of EOT partners or of the European Commission.

1. DOCUMENT INFORMATION

Authors	J.M. Rico (UCLM) J.L. Espinosa-Aranda (UCLM) N. Vallez (UCLM) J. Parra (UCLM) O. Deniz (UCLM) M. Herbst (Evercam) V. Quinn (Evercam) J. Farooq (Evercam) A. Pagani (DFKI)
Responsible Author	Alain Pagani (DFKI) e-mail: alain.pagani@dfki.de
Keywords	Demonstrator 3 – Flexible Mobile Camera
WP/Task	WP4
Nature	Other
Dissemination Level	PU

2. DOCUMENT HISTORY

Person	Date	Comment	Version
Alain Pagani	06.06.2018	Delivered version	1.0

3. ABSTRACT

This document describes the Flexible Mobile Camera demonstrator configuration application. The Flexible Mobile Camera demonstrator describes a portable smart camera powered by the EoT device.

The objective of the document is to describe the configuration application, its purpose and use, the components and services that are used by it and how the development has been carried out, paying attention to the requirements and use cases defined in the specifications document.

4. TABLE OF CONTENTS

- 1. Document Information..... 2
- 2. Document History 3
- 3. Abstract..... 4
- 4. Table of Contents..... 5
- 5. Introduction 6
- 6. Short description of the demonstrator 7
- 7. Configuration App Software Description 8
 - 1. Requirements 8
 - 2. Modules 9
 - 3. Software architecture 12
- 8. Configuration App Software Documentation..... 14
 - 1. Third parties libraries and permissions. 14
 - 2. Use cases..... 16
 - 3. Firebase DB..... 33
- 9. Conclusions 39

5. INTRODUCTION

This document describes the Flexible Mobile Camera demonstrator configuration application. In this demonstrator, the EoT device is connected to an Android device and two cloud services, Firebase Cloud Messaging and Google Cloud Storage.

To carry out this communication, the user must make an initial configuration of the EoT Device. Therefore, the configuration application is used, in the first instance, to communicate the EoT Device with an Android system and with the Google and Firebase cloud services.

This application has been developed for Android devices. Using the configuration application the user will be able to add and set up an EoT device, making the user able to interact with it.

The application allows the user to define the operation of the EoT device, allowing him to choose between different capture modes (normal and smart) and different options ('key events', store, blur, etc.).

In the 'normal' mode, the camera captures a snapshot at regular intervals (configurable by the user). Snapshots are stored on the SD card.

In the 'smart' mode, the camera performs an analysis of the image, trying to detect 'key events' that are interesting for the user. Regions of interest can be also drawn to detect motion. Whenever one or more of the key events are detected, the image is stored in the SD card, along with text information (tags) indicating the events recognized.

Moreover, the configuration application allows the user to activate or deactivate a privacy control through the blur and store options.

The configuration applications allow the user to upload the stored images in the SD card to the Google Cloud Storage service. This way, the user will be able to interact with them in the WebApp.

Finally, the configuration application allows the user to configure the push notifications. They can be enabled and disabled at any time, and the event that launches the push notification can be chosen, just like in smart mode.

The configuration application can be found in

https://gitlab.com/espiaran/EoT/tree/UCLM_FFBoard_mdk_17.04.5/WorkPackage_4/Lifelogging_Camera/android/Wearable.

The webapp can be accessed on Github at:

https://github.com/EyesOfThings/Mobile_Camera_Demonstrator/

The reviewers will be able to access the private parts of the code on request.

6. SHORT DESCRIPTION OF THE DEMONSTRATOR

This demonstrator describes a body worn camera powered by EoT. The device is similar in design to devices such as the Narrative Clip (Figure 1) and benefits from the advanced capabilities of the EoT platform.

There is also a cloud element to the demonstrator that provides the user interface to the device, the captured media and enhanced apps built thereon.



Figure 1: Narrative Clip

Other examples of secondary features include:

- Retrieval / Browsing ... by grouping images by, for example, similarity, one can offer an improved image navigation experience.
- Cloud Applications such as a “Daily Collage” curating a single image with all the key events of the day.

The EoT device differs in its ability to do image processing locally, enabling:

- Smart Storage (Only store what’s needed)
- Annotation (Compile meta data for queries such as “show me all faces”)
- Smart Alerts (Alert (e.g. to Smartwatch))

For each stored photo, the cameras stores also meta data by running each of the EoT device’s capabilities on the image to either trigger alerts, enable queries or schedule deletion.

7. CONFIGURATION APP SOFTWARE DESCRIPTION

1. Requirements

1. Functional requirements

Below are listed each of the requirements with a brief description of the goal. These requirements are a revision of the original requirements. The original requirements can be consulted in the 'Annex 2: Demonstrator Requirements Documents'. The REQ008 described in that document, GPS Awareness, has been removed because the EoT Device has no GPS.

ID	Name	Priority	Difficulty	Description
REQ001	Store Images	High	Low	Store images to SD card
REQ002	Classify Images	High	High	Determine the contents of the image
REQ003	Synchronise to Cloud	High	High	Copy fresh images to cloud & free up local storage
REQ004	Firmware Updates	High	High	Retrieve firmware updates remotely
REQ005	Settings Sync	Medium	High	Retrieve new settings
REQ006	Image Content Meta Data	High	High	Transmit Meta Data about Image Content (Faces, Objects etc.)
REQ007	Connect to bridge	High	Low	Connect to mobile phone using bridge concept
REQ008	Apps Sync	High	High	Retrieve new apps
REQ009	Alerts	High	Low	MQTT style messaging

The above table shows the requirements of the complete demonstrator. It includes the EoT Device application and the configuration application requirements.

The configuration application includes the REQ003 functional requirements. There are functional requirements shared between the EoT Device application and the configuration application. These requirements are the REQ005 and REQ007.

The WebApp includes the REQ008 functional requirement.

The REQ004 must be done using other application. This application is the Pulga Control App in the PC part and the bootloader in the EoT Device part.

2. Non-functional requirements

ID	Name	Type	Priority	Difficulty	Description
NFR001	Battery duration	Performance	High	High	Battery of the device used in the system should last for more than one hour at least
NFR002	Ruggedness	Performance	High	Low	The device must be able survive splashes and dropping
NFR003	Storage of images	Performance	High	Low	Phone must store a reasonable # of images
NFR004	Storage of 3 rd party data	Performance	Med	Med	For social network apps

Most of the non-functional requirements are achieved due to the characteristics of the EoT device, as the NFR001 and the NFR003 non-functional requirements. The connectivity with 3rd party data (NFR004 non-functional requirements) is achieved thanks to the WebApp.

In the original non-functional specification there was five non-functional requirements. In this update, the NFR005 (Sync via mesh of other phones) requirement has been finally discarded because it was not necessary for the functionality of the demonstrator.

2. Modules

This section describes the composition of the configuration application. It consists of seven modules. Most of them include parts of various functional requirements. Others complete some requirement.

1. Sign In module.

1. Description

This module provides a sign in method with the Google account.

2. Achieved Requirements

This module does not achieve a requirement, but it helps to get the REQ003, REQ005 and REQ007 functional requirements.

2. Communication between the Android application and the embedded WebApp module.

1. Description

The native application (Android) will have an 'embedded' WebApp (as a web view). The correct integration between the WebApp and the native application is the purpose of this module.

3. Configuration section module.

1. Description

This module includes the configuration screen of the EoT Device and the configuration Application. This configuration includes:

- Setting alarms. The user is able to specify events that should generate an alarm. In addition, the user can customize the alarm notifications.
- Setting capture mode. The user is able to change the capture mode of the EoT Device.
- Setting the network parameters. The user is able to update the SSID and the password of the bridge (i.e. whatever device that acts as a router and the EoT device connects to).

2. Achieved Requirements

This module does not achieve a requirement but it helps to get the REQ005 functional requirement.

4. MQTT client module.

1. Description

The Android application includes a MQTT client. This client is in charge of:

- Connecting with the MQTT broker.
- Sending configuration parameters.
- Receiving images and their metadata.
- Sending application updates.

2. Achieved Requirements

This module does achieve the REQ005 functional requirement.

5. Synchronization in the cloud module.

1. Description

In this module, interaction between the Android application and the Firebase server is managed. This module is divided in two submodules:

- Uploading images: The Android application sends the images to the Firebase server. Each image has a document in the database. This document contains the image metadata.
- Updating configuration: The user uses the Android application to change the EoT Device configuration. This configuration is stored in the shared database.

2. Achieved Requirements

This module does achieve the REQ003 functional requirement.

6. Connectivity module.

1. Description

This module includes the connection between the EoT Device and the configuration application.

2. Achieved Requirements

This module does achieve the REQ007 functional requirement.

3. Software architecture

The Figure 2 shows the demonstrator structure. It includes the EoT device, the configuration application, the WebApp and the communications and connections.

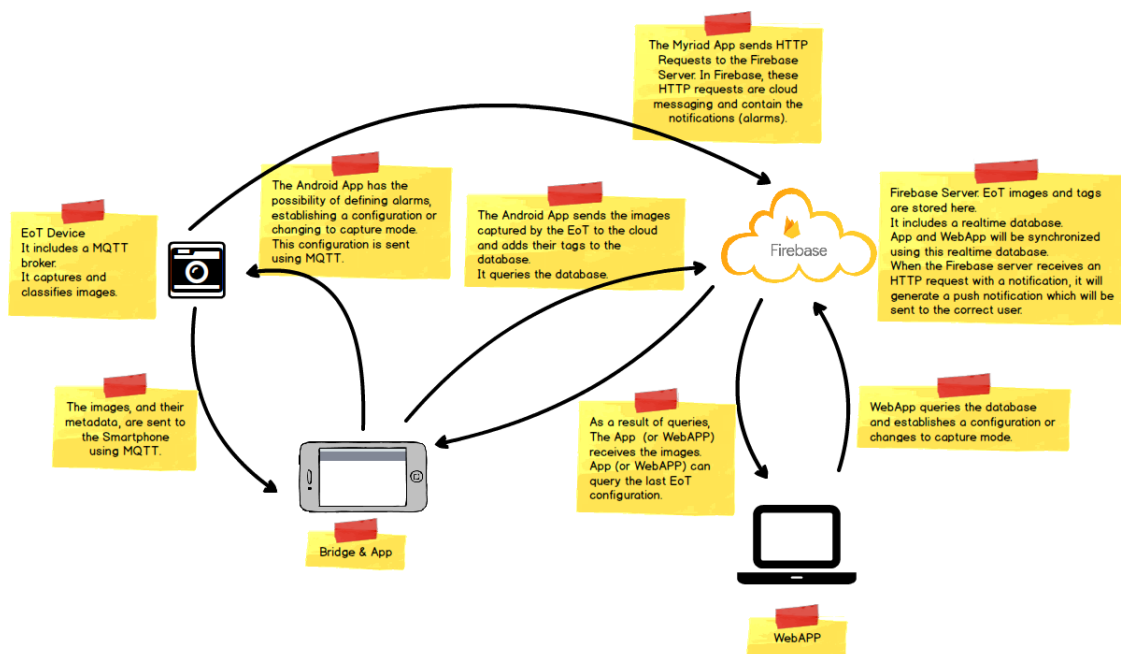


Figure 2: Demonstrator structure

This document is focused on the configuration application. Therefore, the specific scheme is shown by the Figure 3.

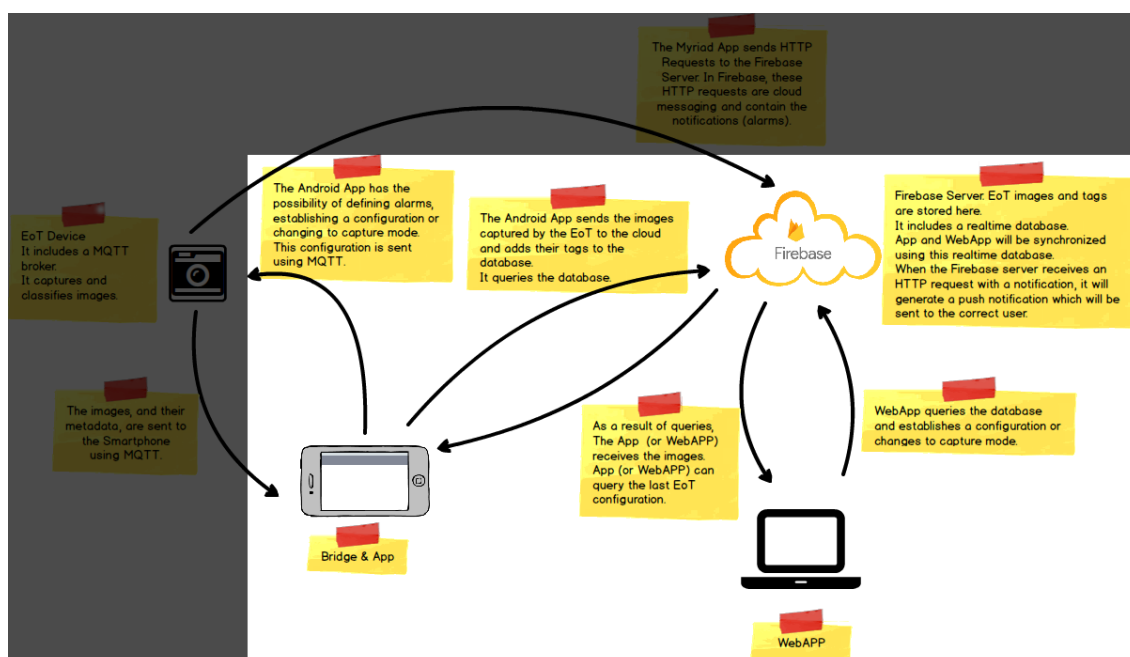


Figure 3: Configuration application structure

Figure 3 shows that:

- The configuration application interacts with the EoT Device and with the Firebase cloud services. The interaction with the EoT Device is done using the MQTT protocol. The interaction with the Firebase cloud services is done using the Firebase library.
- The WebApp, part of the configuration application, interacts with the Firebase cloud services.
- As mentioned above, the configuration application can establish the operation mode and the configuration of the push notifications of the EoT Device.
- The configuration application can request the captured images by the EoT Device using MQTT and send it to Firebase cloud service using the Firebase. In addition to the images, the labels are also sent. The label files contain the information about the images (key events detected, options enabled, etc.).
- The images are uploaded to Google Cloud Storage and the labels are stored in a real time database provided by Firebase.
- The WebApp is able to visualize the images stored in Firebase. The WebApp can filter the images stored in Firebase server by the date or by 'key events'.

8. CONFIGURATION APP SOFTWARE DOCUMENTATION

1. Third parties libraries and permissions.

1. The Android App

The configuration application is an Android application. The minimum API version to use the configuration application is API 16. This API version corresponds with Android 4.1 (JELLY_BEAN)

To communicate the configuration application with the Firebase cloud service, the following libraries have been added as dependencies:

- com.google.firebase:firebase-core:11.8.0: This dependence allows to use the Firebase core.
- firebase-database:11.8.0: This dependence allows to use the real-time database of Firebase.
- firebase-auth:11.8.0 This dependence allows to sign in with a google account. In addition, the library manages the auth 2.0
- firebase-storage:11.8.0 This dependence allows to stored images in the Firebase storage.
- firebase:firebase-messaging:11.8.0: This dependence allows to use the Firebase Cloud Messaging service.

The previous libraries needs the android.gms:play-services-auth:11.8.0 and the com.google.gms:google-services:3.2.0 libraries.

The Client MQTT has been implemented using the paho.clieent.mqttv3:1.0.2 library.

The configuration application needs the following permissions to work correctly:

- android.permission.CHANGE_WIFI_STATE
- android.permission.WRITE_EXTERNAL_STORAGE
- android.permission.WAKE_LOCK
- android.permission.INTERNET
- android.permission.ACCESS_NETWORK_STATE
- android.permission.READ_PHONE_STATE
- android.permission.ACCESS_WIFI_STATE

The permissions related to WIFI_STATE and NETWORK_STATE are necessary to change the connected network in the 'add new EoT device' process.

The WRITE_EXTERNAL_STORAGE permission is necessary to store configuration information.

The WAKE_LOCK and READ_PHONE_STATE permissions are necessary to show the push notifications.

The INTERNET permission is necessary to have connection with Internet.

To embed the WebApp into the configuration application has been used the Chrome Custom Tabs library (com.android.support:customtabs:25.3.0) Finally, to crop the image and define a ROI, the configuration application uses the com.theartofdev.edmodo.cropper package

2. The WebApp

This is a classic Javascript & Ruby v2.4.2 & Firebase implementation hosted on Heroku using multiple 3rd party APIs & SDKs

Firebase services used

- DB for all image & device metadata incl. configs
- Storage for images and animations

Firebase APIs used

- Firebase Database REST API (Server side)
- Firebase User authentication with Google Account
- Firebase SDK (client side) V5.0.4

Other Google APIs used

- Google Service Account Authentication (Ruby)
- Google Cloud Storage JSON API

Other Databases

- PostgreSQL (9.6)

Other 3rd Party APIs used

- Dropbox. If the user has an account this API provides him with the possibility of storing the captured images on this platform.
- Evercam, for connecting the developed application with other services implemented by Evercam.
- Zapier v3 for sending emails
- Emojis h
- Mailgun API <https://afeld.github.io/emoji-css>
- IFTTT
- FFMPeg for Animation creation and storing all meta information e.g FPS, Size, Resolution
- Bitlinks (<https://bitly.com/>) API for tiny URLs on sharing. (3.0) We also have Bitlinks dashboard to see, when someone clicked on shared URL and from which country.
- Rake Schedulers on Heroku for Wizards
- Semantic UI 2.3
- Rails V5.0.2
- MomentJS for timestamps V2.22.2
- SeaweedFS V0.76
- AWS Route53 for Domain hosting (Heroku (<https://quiet-beach-70370.herokuapp.com>) to AWS <http://eot.evercam.io>)
- FFMPeg Buildpack (<https://github.com/jonathanong/heroku-buildpack-ffmpeg-latest.git>) for Heroku V4.0
- Github to Heroku Automated Deployments

2. Use cases

3. Sign In

To use the Google and Firebase cloud services, the user should have a Google account. When the user signs in, the configuration application will be able to communicate with Google APIs, and the configuration application will share the access to Google APIs using the session tokens. The session tokens will be sent from the configuration application to the EoT Device and the EoT Device will save them in the SD card. This occurs when a new EoT Device is added.

Due to the EoT Device sends images to Google Cloud Storage, the application will request to have offline access.

1. In the configuration application

When the configuration application is opened, the user has two possibilities: to Sign In or to go to menu. If the user does not sign in previously, the user must do it and allow requested permissions. When these steps have been completed, the configuration application shows the menu activity.

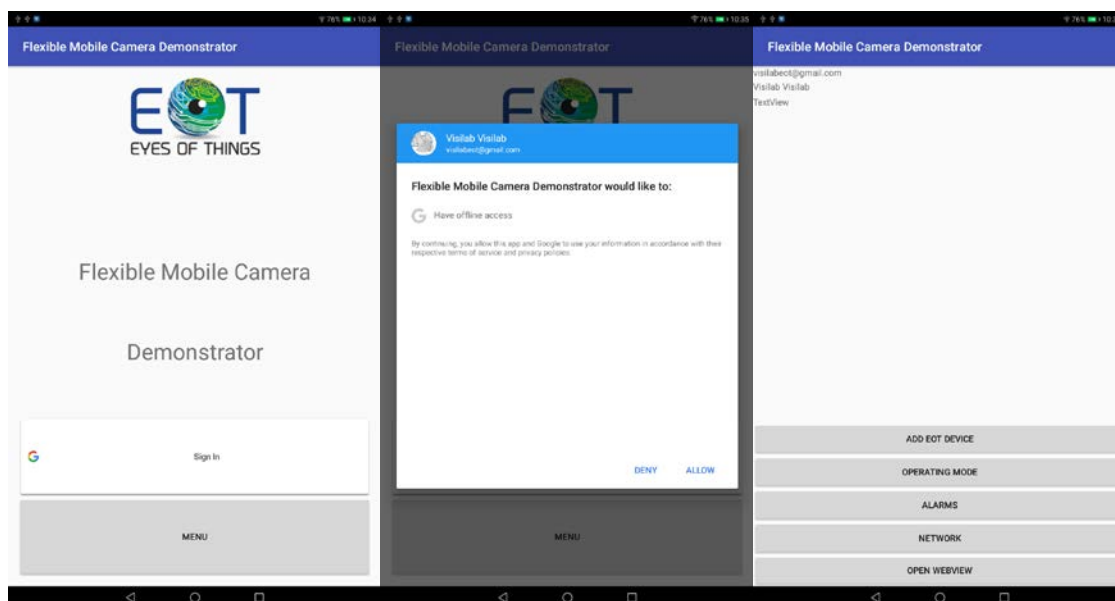


Figure 4: Sign In process in the configuration app

4. Add a new EoT Device

To work with an EoT Device from the configuration application, the first step is to add it to the list of configured devices. Once added, the user will be able to connect to the EoT Device, obtain the stored images and edit the EoT Device capture and push notification configuration.

In this step, the EoT device will be configured. This way, the EoT Device will be able to communicate with the Google and Firebase cloud services. Therefore, the stored images can be sent from the EoT Device to Google Cloud Storage and a push notification will be generated. In addition, the configuration application will set up a network profile for the EoT Device.

1. In the configuration application

To add a new EoT Device, the user must use the 'Add EoT device' button.

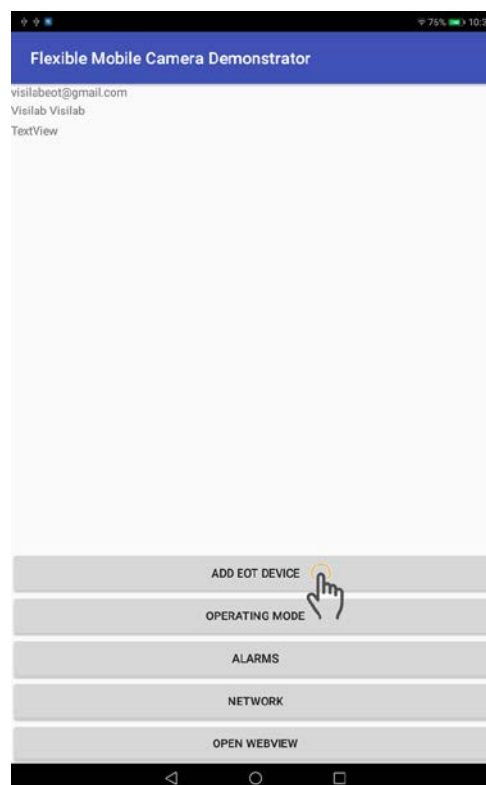


Figure 5: Configuration app main menu. Add EoT Device

The configuration process has three steps:

1) Device name and communication:

In this step, the EoT Device generates an AP with a MQTT Server. The configuration application connects to the AP and exchanges information as authentication tokens. In addition, the user can choose a name for the EoT Device.

2) New network profile:

In this step, a network profile is added to the EoT Device Wi-Fi flash memory. This way, when the EoT Device application starts, the EoT Device will search in its network profiles and will try to connect with one of them.

3) Finish configuration:

The configuration finishes. The EoT device restarts and changes its network configuration. The Android Device disconnects from the EoT Device and connects to a network point.

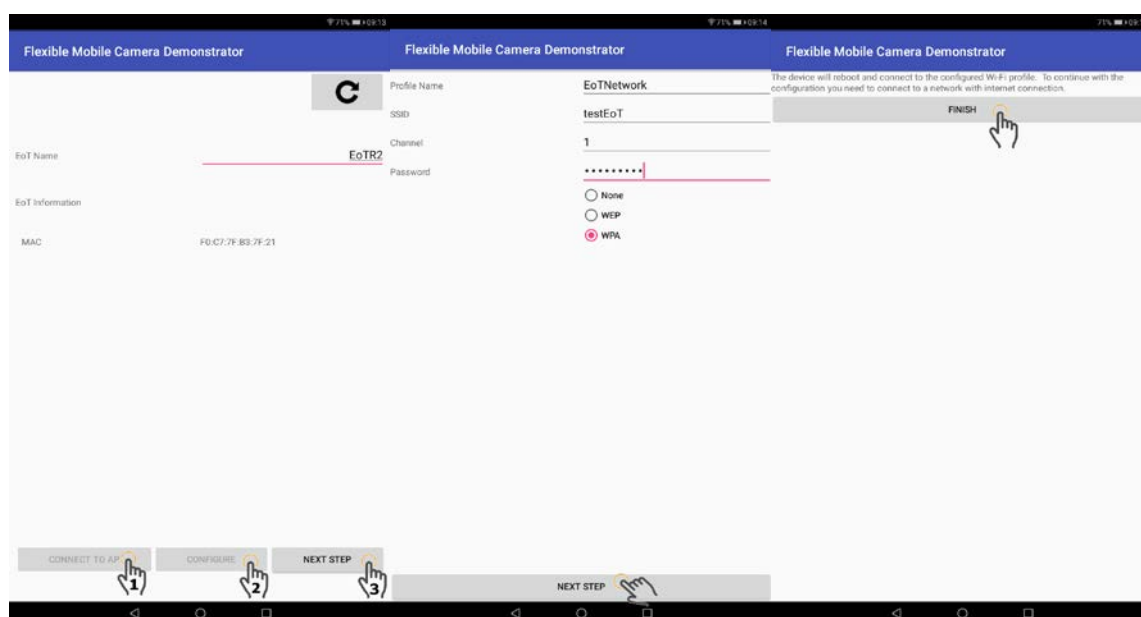


Figure 6: Adding a new EoT Device

5. Connect with an EoT Device

To interact with an EoT device, the configuration application must connect to it. When an EoT device is configured and the EoT Device application starts, the EoT Device connects to a network profile and generates a MQTT server.

1. In the configuration application

To connect to an EoT Device, the user can use the 'connect' button of the EoT profile. This button opens the connection menu to an EoT Device. This menu requires an IP direction and a port. These parameters are sent from the EoT Device to the configuration application using the push notification system.

If the user presses the 'Connect' button, the configuration application will connect to the MQTT server of the EoT Device.

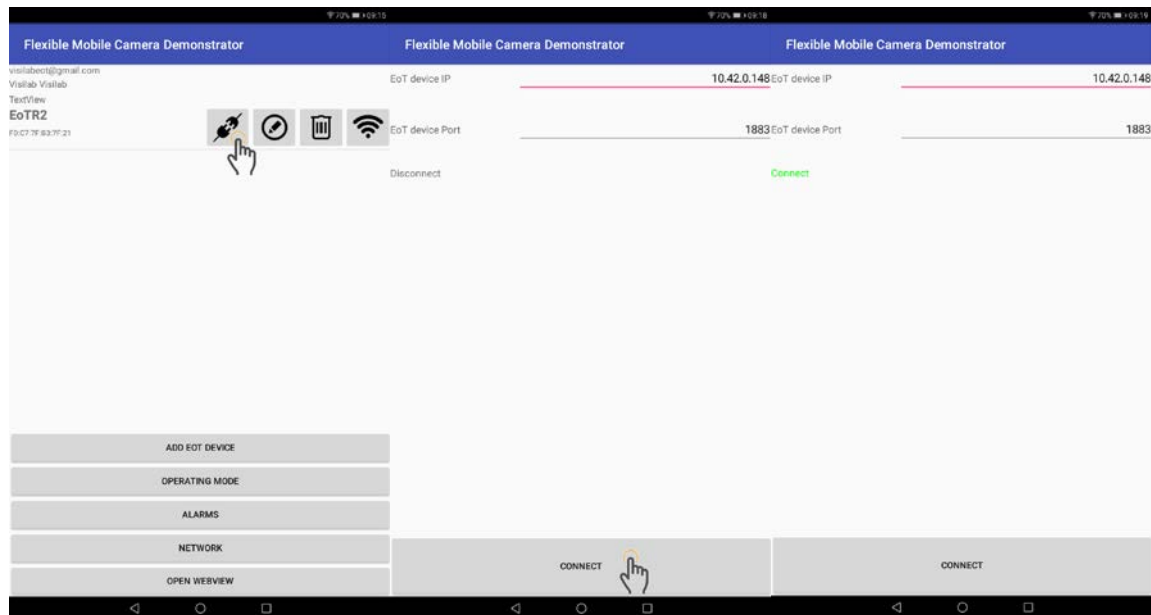


Figure 7: Connecting with an added EoT Device

6. Operational Mode

To activate, modify or deactivate the image capture of the EoT Device, the configuration application has the 'operating mode' menu. Through an MQTT server running in the EoT Device, the configuration application is able to activate, modify and deactivate the image capture.

1. In the configuration application

When the user presses the 'operating mode' button, the configuration application opens the operating mode activity. In this activity, the user can activate or deactivate the capture, change the capture mode, enable or disable 'key events' and change the privacy policy.

When the user changes an option, this change is sent to the real time database in Firebase cloud service and to the EoT Device.

In the operating mode menu, the user can:

- Enable/Disable the capture
- Change between normal mode and smart mode. In the normal mode, one image is captured every X seconds. In the smart mode, each image is examined and in case it contains a 'key event, the image will be stored.
- In the normal mode, the user can change the capture frequency.
- In the smart mode, the user can enable/disable the 'key events'
 - FaceDetected. There is a face in the image.
 - LargeFaceDetected. There is a near face in the image. A near face may be indicative of interaction with that person.
 - Anger. There is an angered face in the image.
 - Disgust. There is a disgusted face in the image.

- Fear. There is a scared face in the image
 - Happiness. There is a happy face in the image.
 - Neutral. There is a neutral face in the image.
 - Sadness. There is a sad face in the image.
 - Surprise. There is a surprised face in the image.
 - MotionDetected. A movement has been detected.
- Define an region of interest (ROI)
- Change the privacy policy
 - Blur: If the field is activated, a detected face will be blurred.
 - Store: This field indicates if the images are stored in the SD card of the EoT device.
- Sync up the images. When the user presses the 'sync now' button, the EoT Device starts sending the stored images in the SD card, along with them labels. When the configuration application receives an image and its labels, it sends the image to Google Cloud Storage and adds a new entry in the database. This entry contains the date, labels and path of the image.

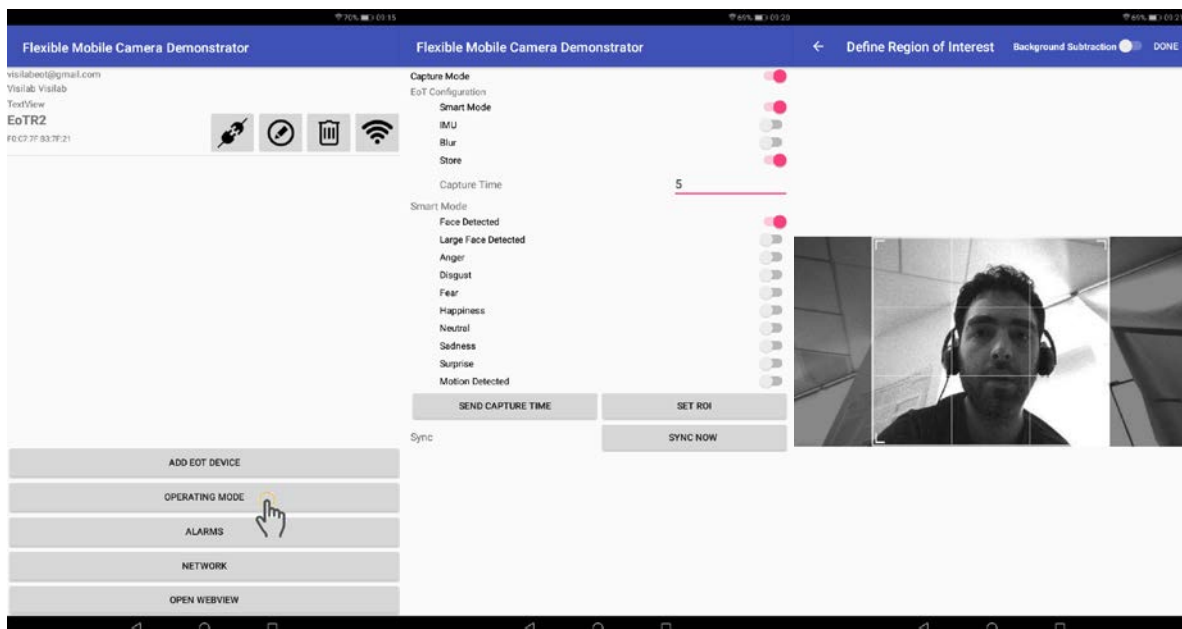


Figure 8: Operating mode. Setting a ROI

7. Alarms

To enable or disable the different alarms, the configuration application has the 'alarm' menu. The alarms are sent using the Firebase cloud messaging (FCM).

An alarm contains an image of the event that triggers it and a text with the key events found. When the Firebase server receives the alarm from the EoT Device, the FCM will send a push notification to the configuration application.

1. In the configuration application

When the user presses the 'Alarms' button, the configuration application opens the alarms configuration menu. In this menu, the user can enable or disable the key events for the alarms. The key events are the same that in the 'Operating mode'.

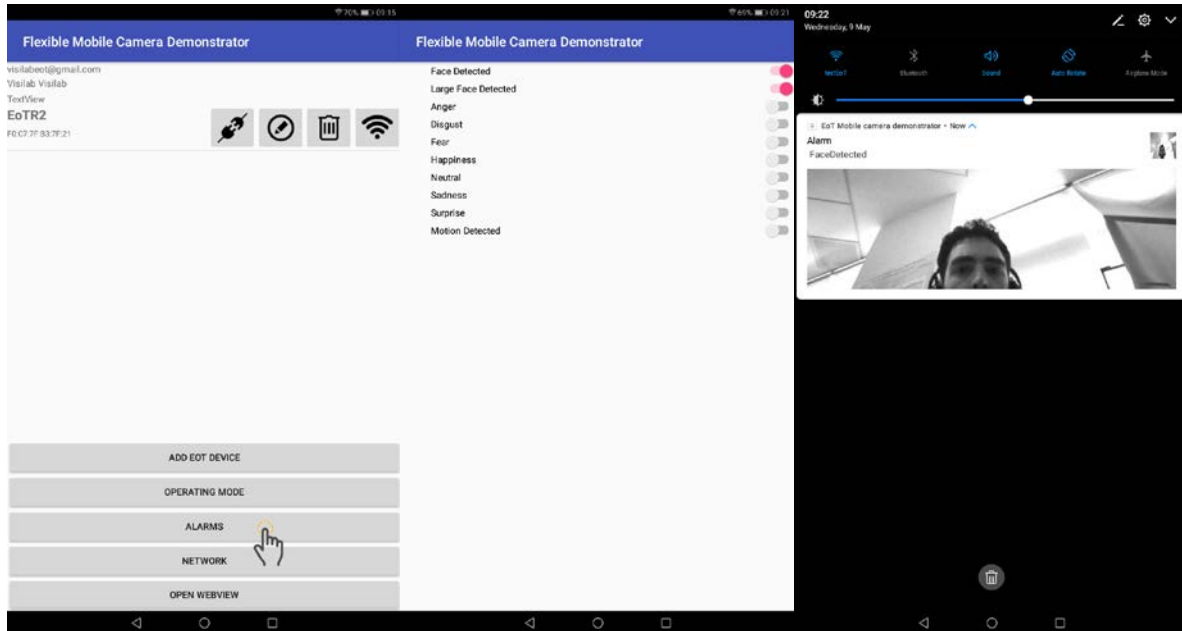


Figure 9: Alarm menu. Push Notifications working

8. WebApp

Using the WebApp, the user can obtain the images uploaded to Google Cloud Storage service. These images can be filter by date and KeyEvents. In addition, the WebApp has several functionalities, as 'make animation', or integration with different services. All of them will be explained on the sections below.

1. In the configuration application

Logging In

Using Google Single Sign On, which conveniently leverages the whole Google infrastructure for password management (including 2-factor authentication) and authentication for Firebase assets (DB & Image management).

Later, if one wanted to allow users to pay for their own usage of APIs like Google Maps and Google Vision APIs then this can all be done via this single userID.

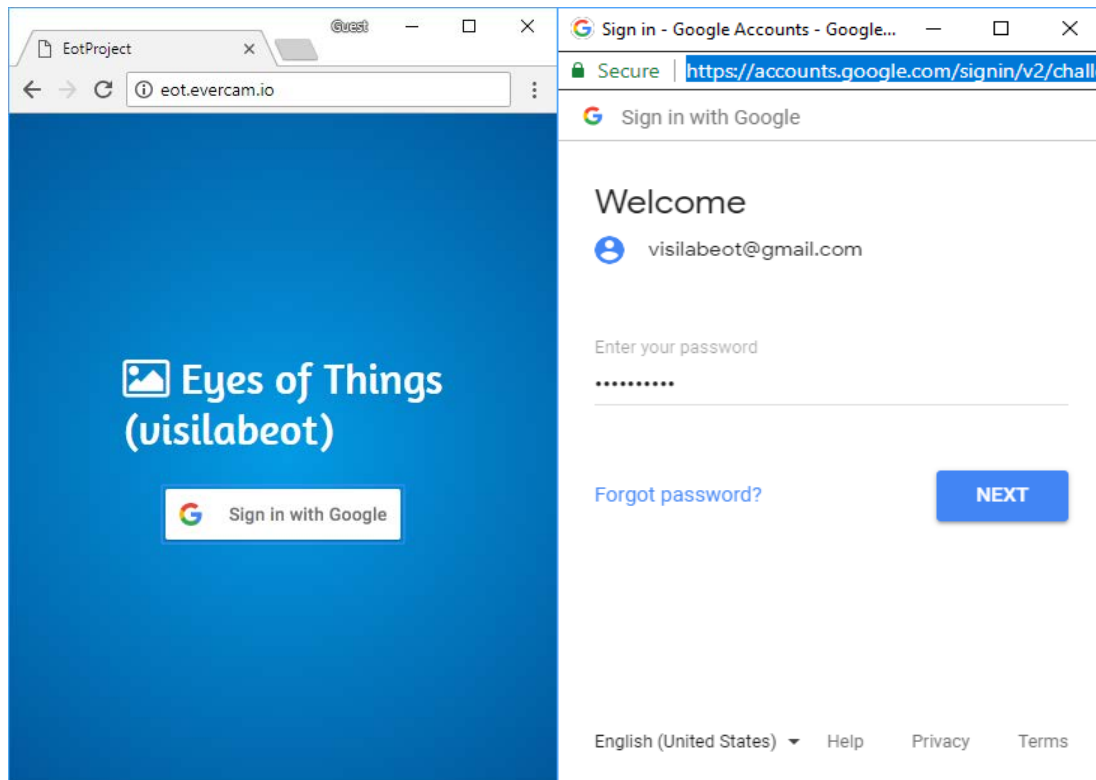



Figure 10: Login Screen

Figure 11: Google Password Screen

Home Screen

This lists all available devices. In the Figure 12, only one is present. New devices are added via the mobile app, hence there is no “Add Device” option on this screen.

Also available from this screen is the app navigation  from the symbol in the top right.

To view or manipulate the images, the user needs to click on the device image.

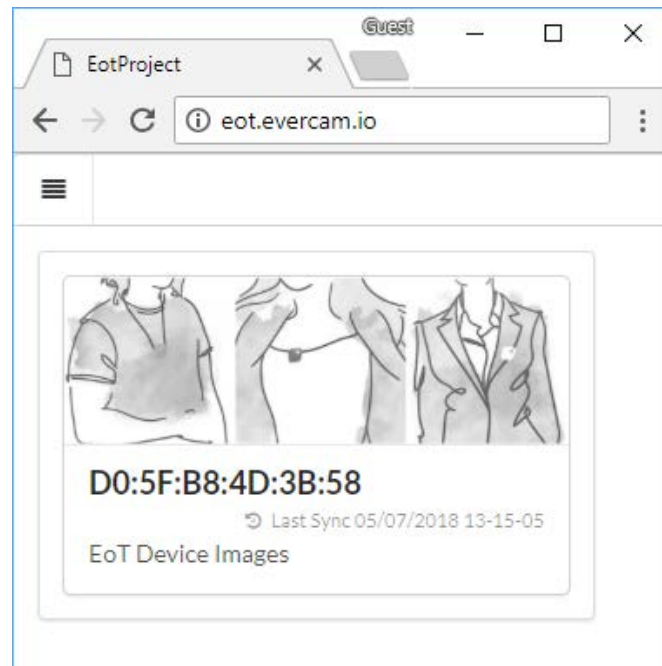


Figure 12: Home Screen with the device image

For each listed device, the user has the following functionalities:

1- Image Listing

This screen shows all the images associated with this device. In the figure 13, it is possible to see an example of this functionality in which several hundred images that have been previously captured by the device can be viewed using the scroll.

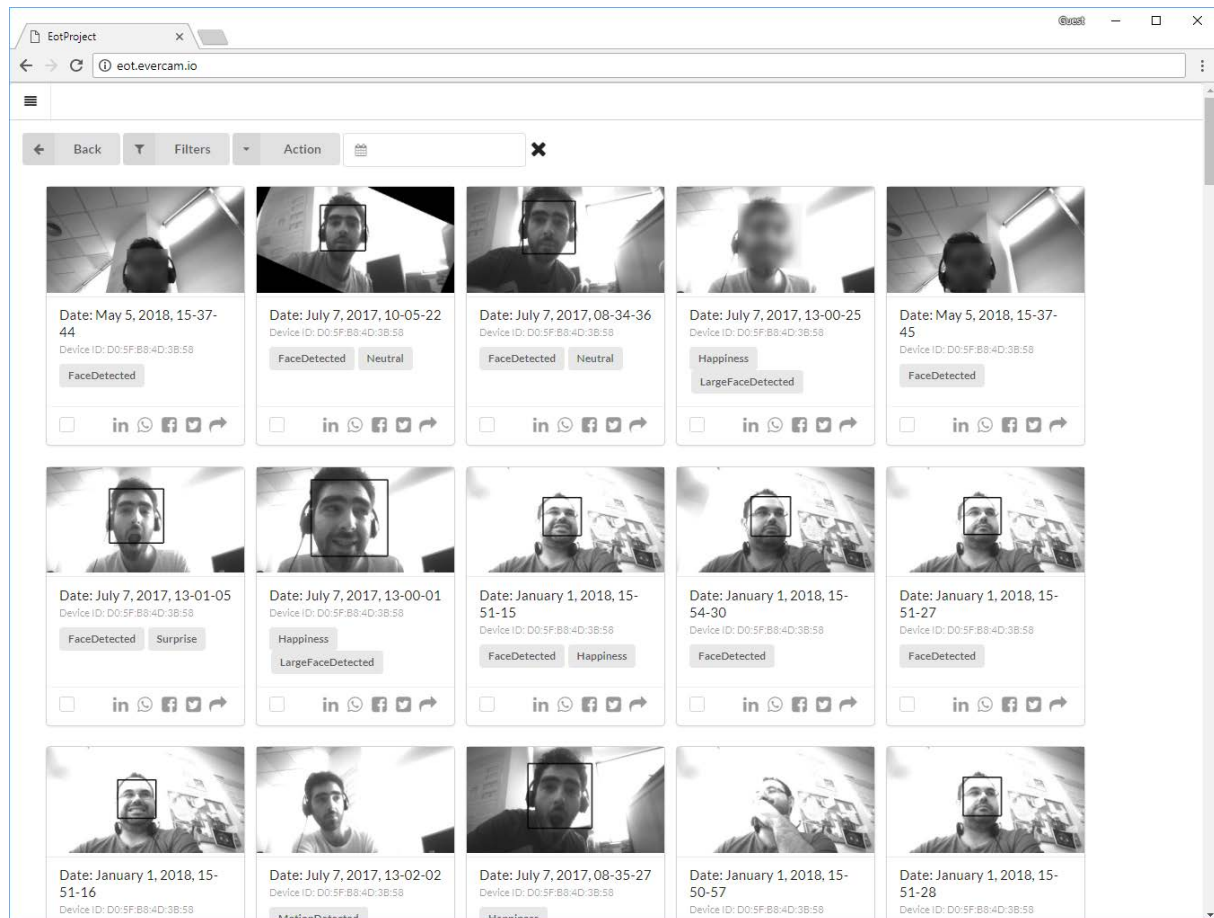


Figure 13: Full Image Listing

Images can be selected individually, or from the main menu "Select All".

Once the images are selected, an Action can be chosen from the Action menu. Currently the only available action is to create an animated video collage of the selected videos. Actions to be deployed include "Publish to Public Feed", "Send to Dropbox".

2- Sharing Images

Sharing is a key feature of the Life Logging camera, as such, each image can be shared individually, and we have built in integrations to the most popular social media platforms: Twitter, Facebook, Whatsapp and LinkedIn.

The process is simply a matter of clicking on the appropriate icon from the selection below:



Figure 14: Social Media Icons

We have also built in sharing to our own Public Feed which is available from the navigation menu or at the URL <http://eot.evercam.io/feed>

The sharing icon is shown in the Figure 15.



Figure 15: Sharing icon

3- Filtering by Emotions

Selecting the filter option shows a navigation bar (Figure 16) that displays all the available Emotion Tags that can be used to filter.

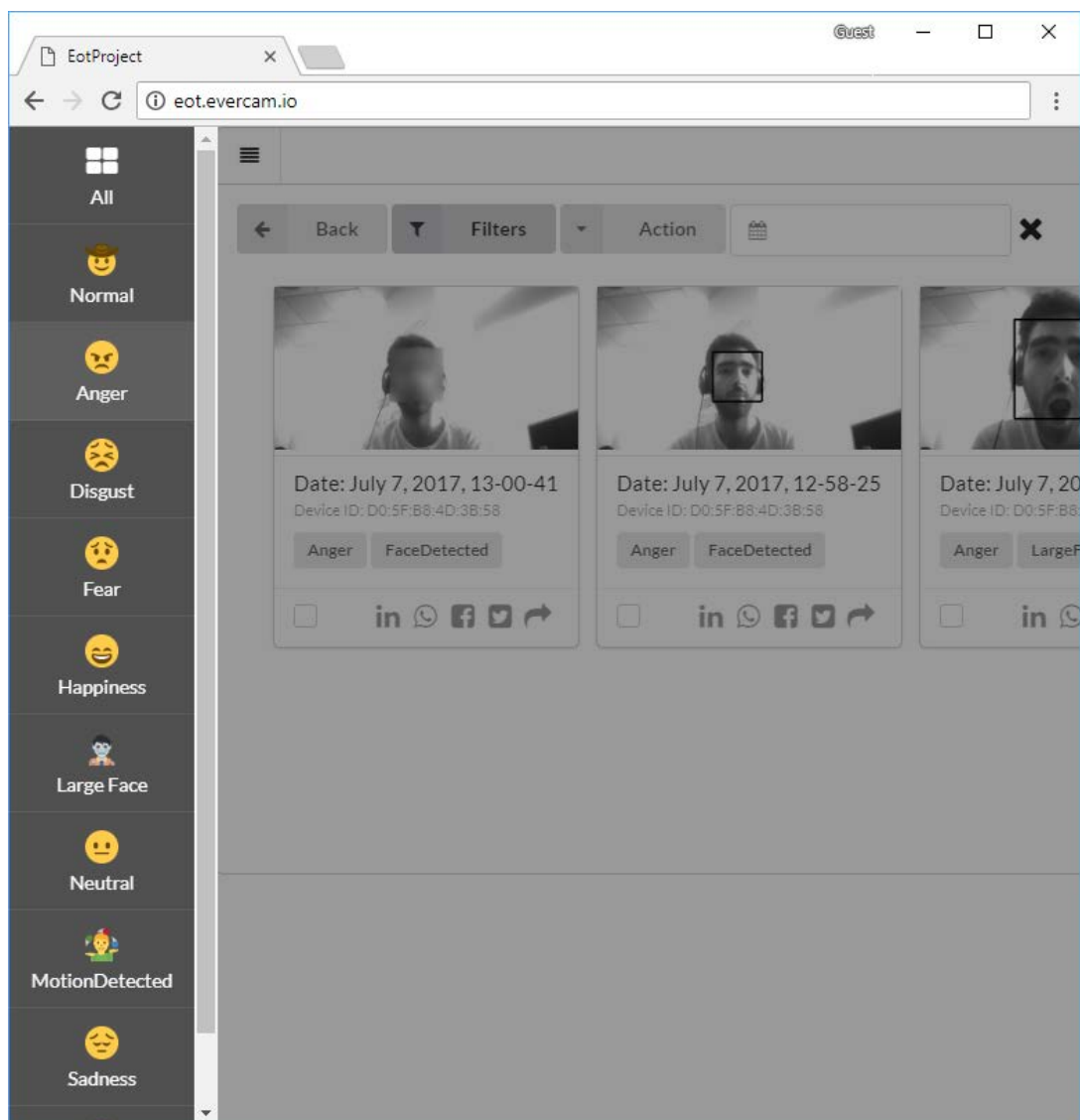


Figure 16: Emotion Filter Menu

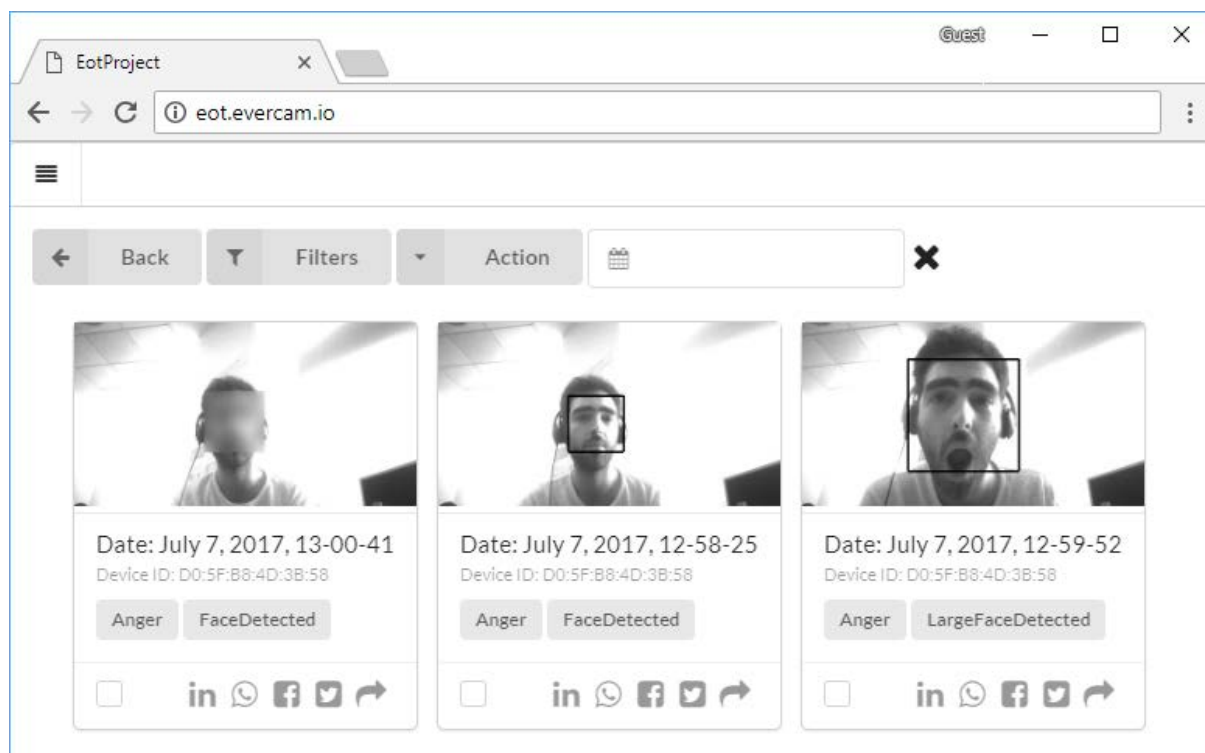


Figure 17: Filtered Images (by 'Angry Face')

4- Filtering by Date

Filtering is also possible via the Date / Calendar icon at the top of the page.

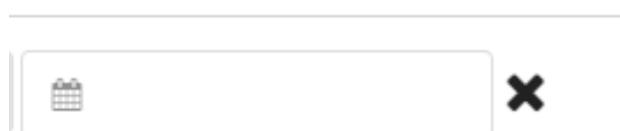


Figure 18: Date Filter

5- Action. Create Animation

Once selected, the "Create Animation" Action becomes available. This functionality combines images to create a short video. This is useful, for example, to make a single video summary of a day's activity.

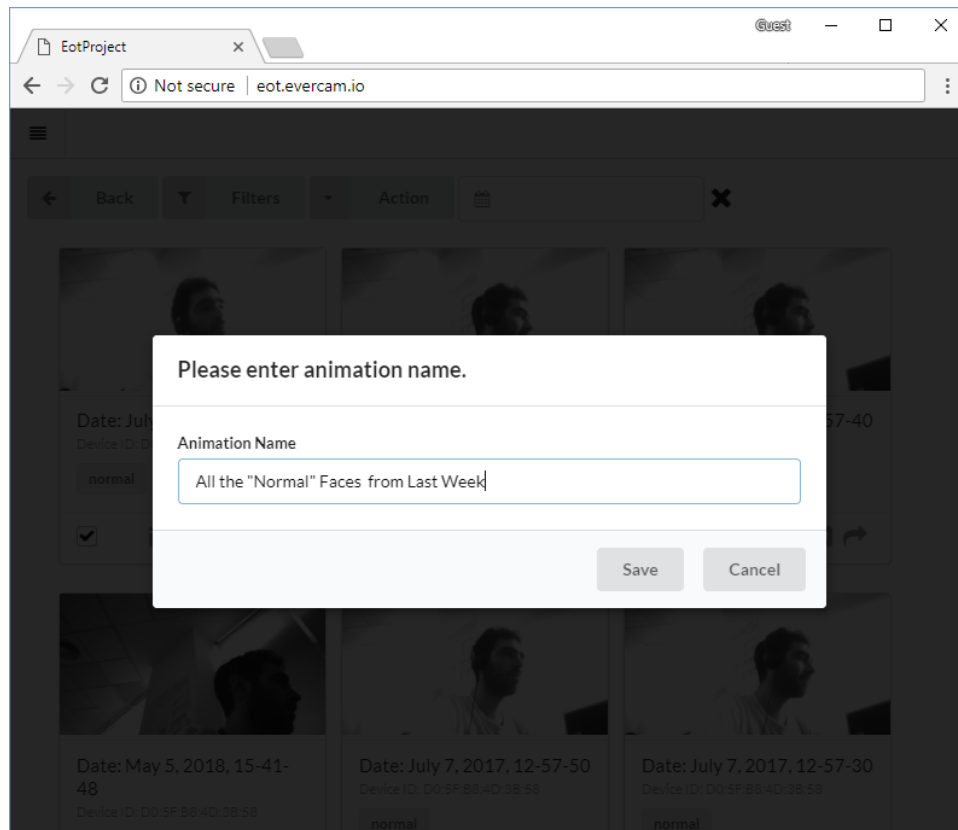


Figure 19: Adding a name to the animation

Once created, animations are available under the “Animations” tab from the main navigation.

Figure 20: Animations created

6- Main Navigation

This section describes individually the various features available in the application.

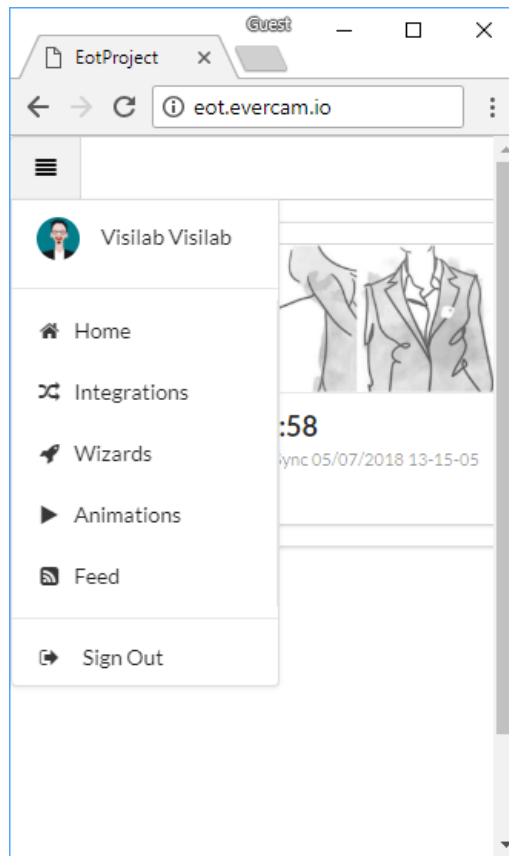


Figure 21: Navigation menu

6.1-Integrations

The purpose of these is to extend the Flexible Mobile Camera EoT functionality with these additional web services. Integrations included by default are:

- Evercam - Uploading images to your Evercam Account
- Dropbox - Uploading images to your Dropbox Account
- IFTTT - Full access to IFTTT via Webhook
- Zapier - Access to Zapier Features via Webhook

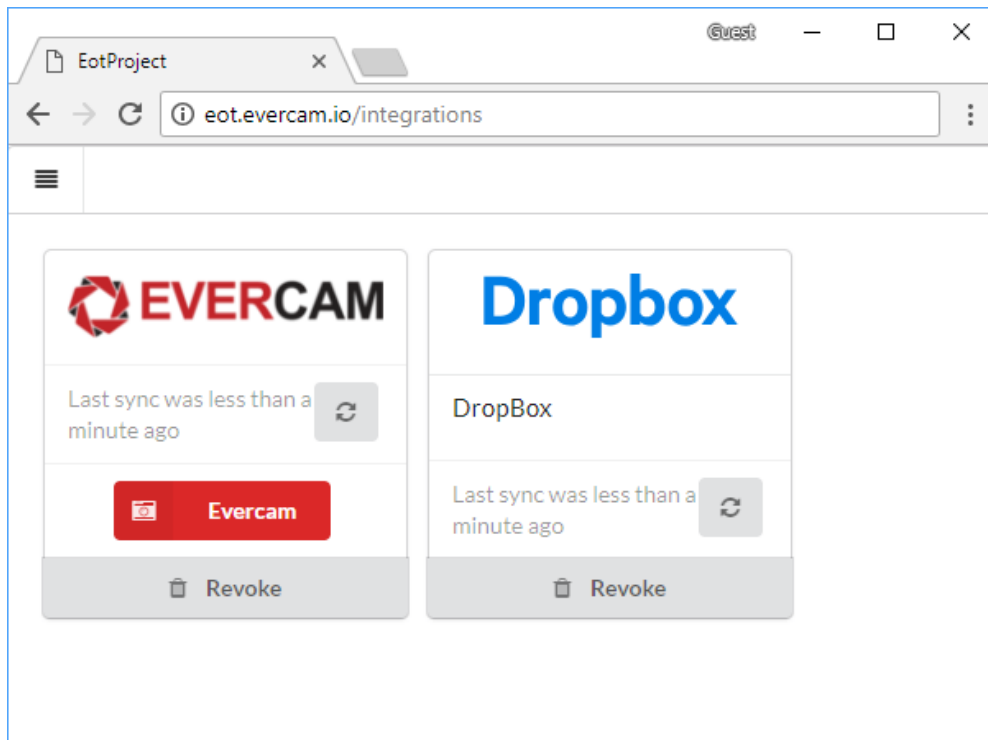


Figure 22: Evercam and Dropbox integration services in the Webapp

For example, when enabled, the Dropbox integration will send a copy of all EoT images into a Dropbox folder.

Figure 23 shows a synchronized Dropbox folder and figure 24 depicts the integration of the webapp with the Evercam API:

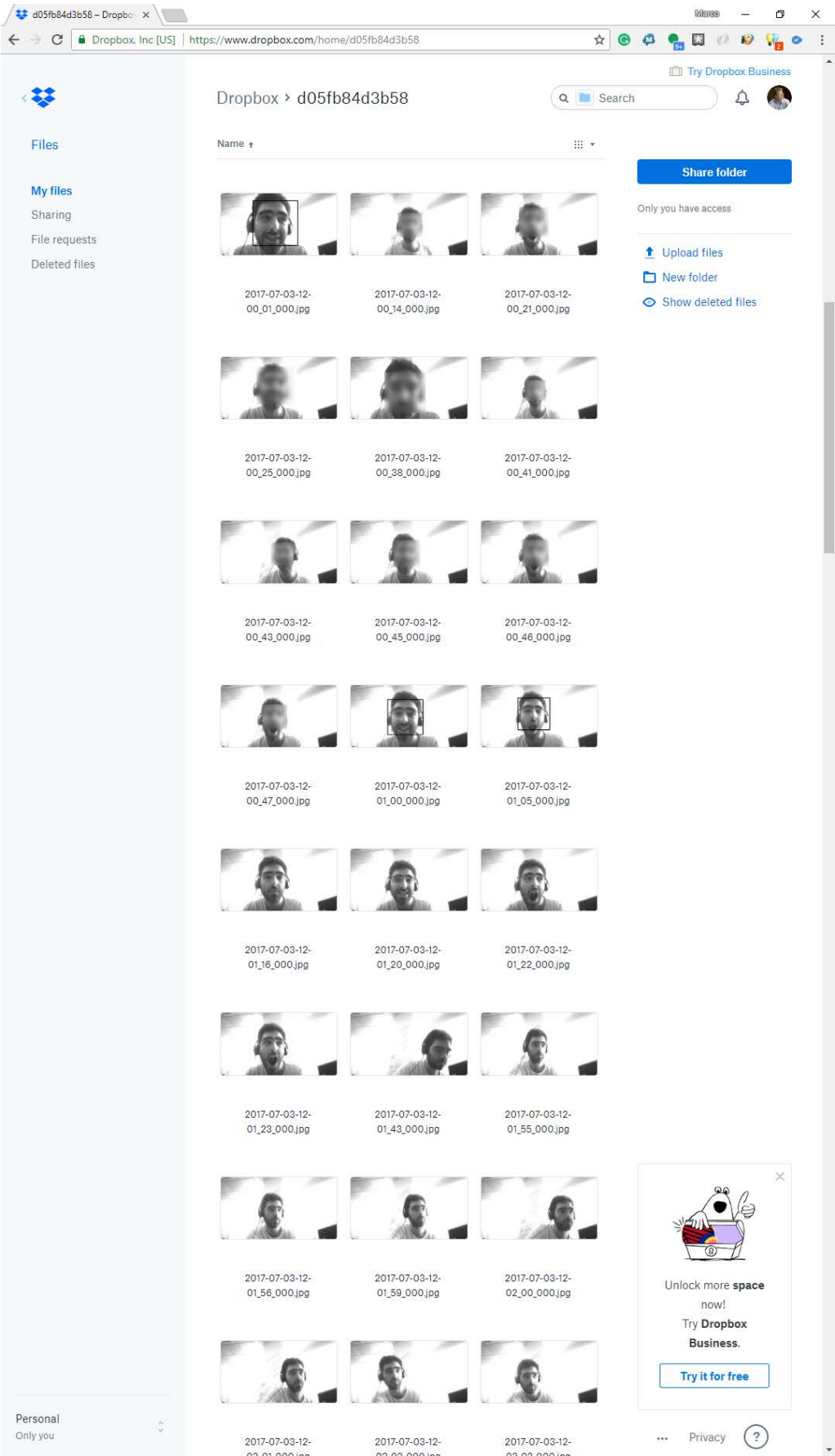


Figure 23: Integration with Dropbox

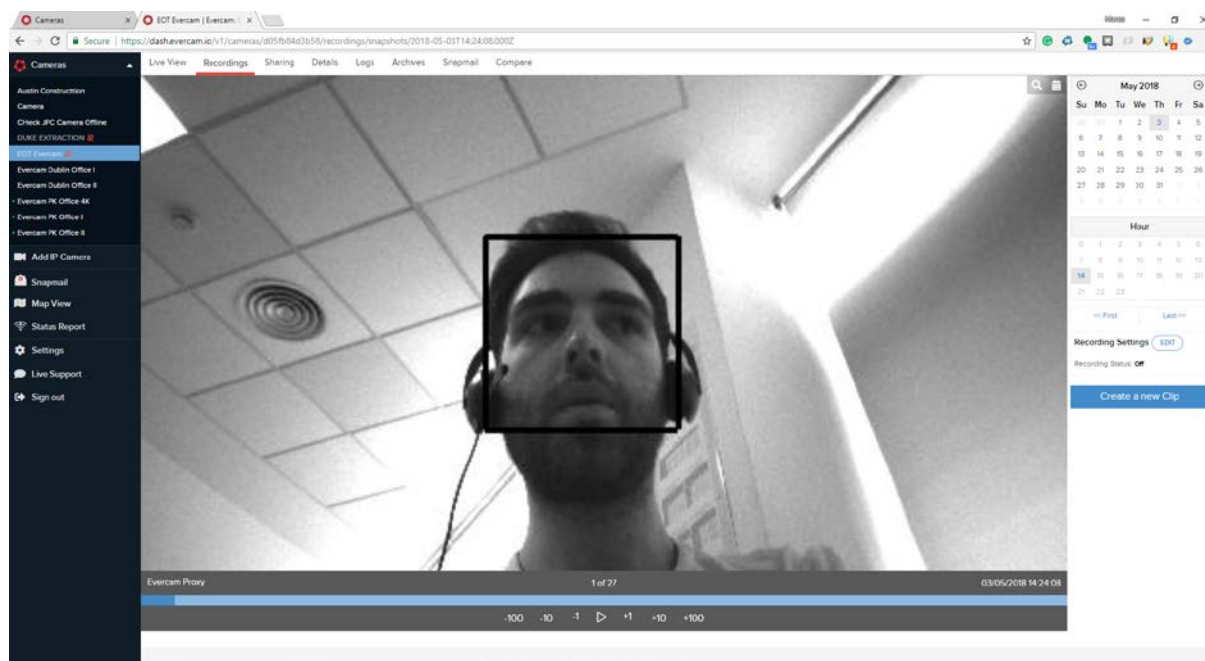


Figure 24: Integration with Evercam API

6.2-Wizards

A wizard is an integration with a service such as email to enable simple conditional functionality.

Wizards will also make use of Integrations, such as Zapier, IFTT or any other integration that can be activated via a Webhook.

The built in Wizards (no integrations required) are triggered by email e.g. If "Angry Face" then send email.

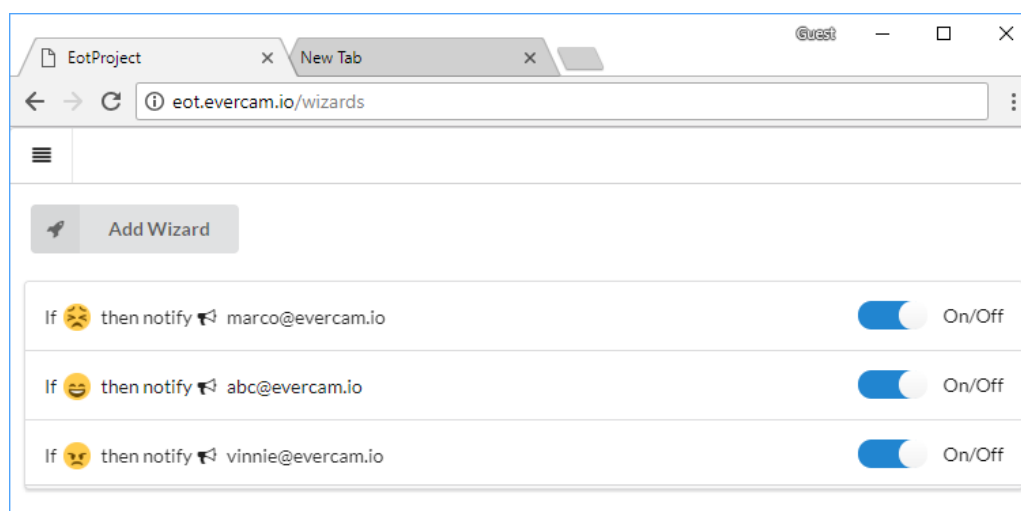


Figure 25: Wizard configuration

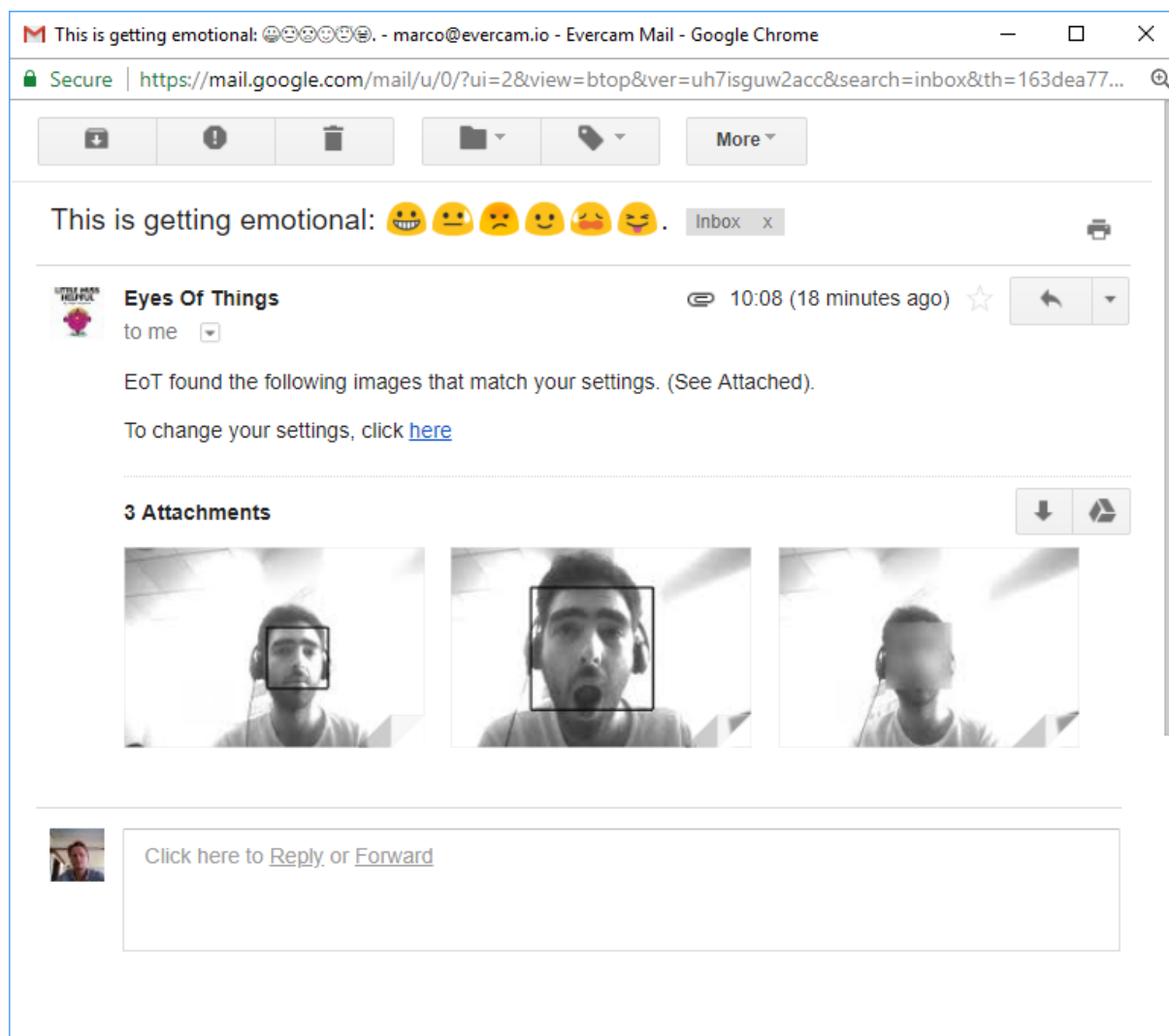


Figure 26: Sample Email Alert Wizard

6.3-Public Feed

Occasionally users may wish to share specific images publicly. This can be done via the main menu, as described, by selecting an image or images and choosing "Share to Public Feed". Animations can also be shared publicly.

This also means that a user has a unique URL which can be shared and will always show a list of images that are shared publicly.

Figure 27 is a screenshot of a sample public feed.

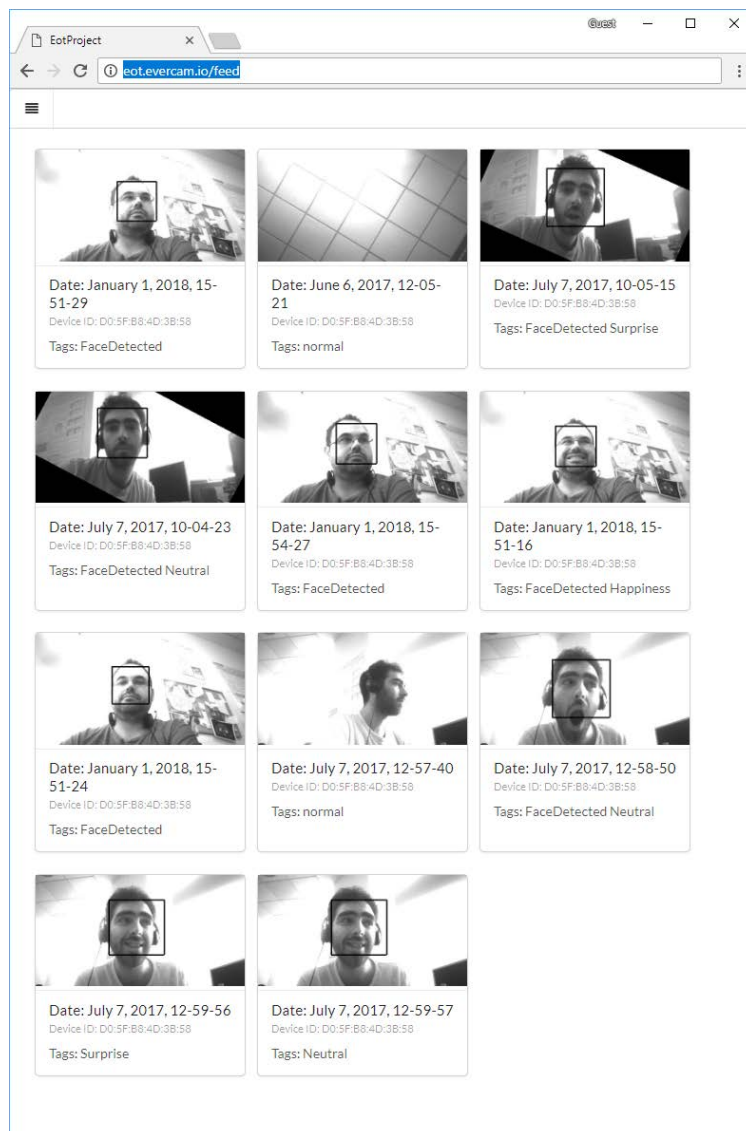


Figure 27: Public Feed

In terms of privacy, this is entirely at the user's discretion. Images can be shared publicly or not and the user can share which images they choose to share publicly.

3. Firebase DB

Each user has one NoSQL document in the database. The key of this document is the user's Google account. Each user can have several EoT devices linked to his account. For each linked EoT device, three news documents in the database will be generated. Dropbox document, Evercam document and EoTMAC document.

The Firebase database has the following structure:

```
UserEmail {
  dropbox {
    accessToken
```

```
        lastSyncDate
        syncIsOn
    evercam {
        apId
        apiKey
        lastSyncDate
        syncIsOn

    EoTMAC {
        Configuration {
            CaptureMode: numeric value
            Capture Period: value,
            Blur: binary value,
            Store: binary value,
            Alarms {
                FaceDetected: binary Value,
                LargeFaceDetected: binary Value,
                Anger: binary Value,
                Disgust: binary value,
                Fear: binary value,
                Happiness: binary value,
                Neutral: binary value,
                Sadness: binary value,
                Surprise: binary value,
            MotionDetected: binary value
        },
        KeyEvents {
            FaceDetected: binary Value,
            LargeFaceDetected: binary Value,
            Anger: binary Value,
            Disgust: binary value,
            Fear: binary value,
            Happiness: binary value,
            Neutral: binary value,
            Sadness: binary value,
            Surprise: binary value,
            MotionDetected: binary value
        }
    }
    Images {
        Timestamp {
            Tags {
                FaceDetected: binary Value,
                LargeFaceDetected: binary Value,
                Anger: binary Value,
                Disgust: binary value,
                Fear: binary value,
                Happiness: binary value,
                Neutral: binary value,
                Sadness: binary value,
```

```

    Surprise: binary value
MotionDetected: binary value
    }
    Path: value
  }
  ...
}
}
EoTMAC {
  ...
}
}
UserEmail {
  ...
}
```

The dropbox document contains the accessToken and the last sync date fields.

The evercam document contains the apild, apiKey, lastSyncDate and syncIsOn fields.

The EotMAC document will have two documents, the configuration document and the image document. The configuration document has the following fields:

- CaptureMode. This field configures how to capture images. It is a numeric field. For example:
 - Normal mode. Value 0 means capturing an image every X seconds time (the X will be specified by the Capture Period value). In Normal mode no classification of the image is performed.
 - Smart mode. Value 1 could mean capturing only when a key event or feature is detected in the image. The key events are specified in the keyEvents document.
- Capture Period. This field indicates how often an image is captured.
- Blur. This field is a binary value, where 0 means disabled and 1 means activated. If the field is activated, a detected face will be blurred (this is for privacy purposes).
- Store. This field indicates if the images are stored in the SD card of the EoT device. It is a binary value, where 0 means disabled (don't store) and 1 means activated.
- Alarms. This document contains the alarms configuration. The user can activate or deactivate each alarm, and this change will be reflected in this document. Each possible alarm has a binary value, where 0 means disabled and 1 means activated.
- KeyEvents. This document contains the key events the "smart" mode can detect. Again, the fields inside this document have a binary value, where 0 means disabled and 1 means activated.

On the other hand, we have the image document. This document contains the information about the captured images. The images will be sorted by their timestamp.

- **Tags.** This document contains the tags detected in the image. As alarms and keyEvents documents, the tags have a binary value, where 0 means "it does not appear" and 1 means "it appears". For each image only one face and emotion will be recognized, since detecting more than one face and getting more than one emotion will be computationally expensive.
- **Path.** This field stores the path where the image is saved in the Firebase server. This field links the database with the storage in the Firebase server.

The Figures 28 and 29 show the backend database structure.

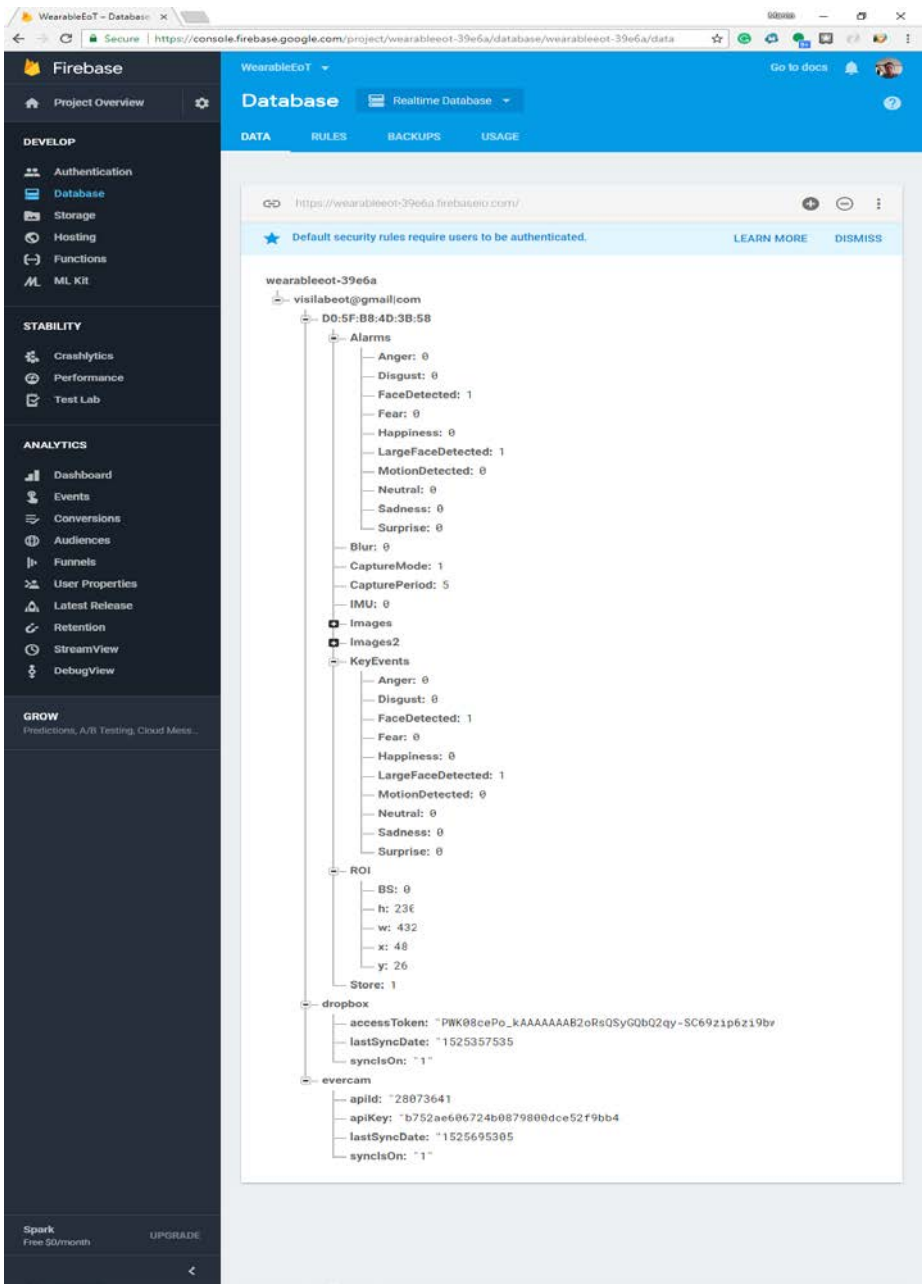


Figure 28: Database snapshot 1

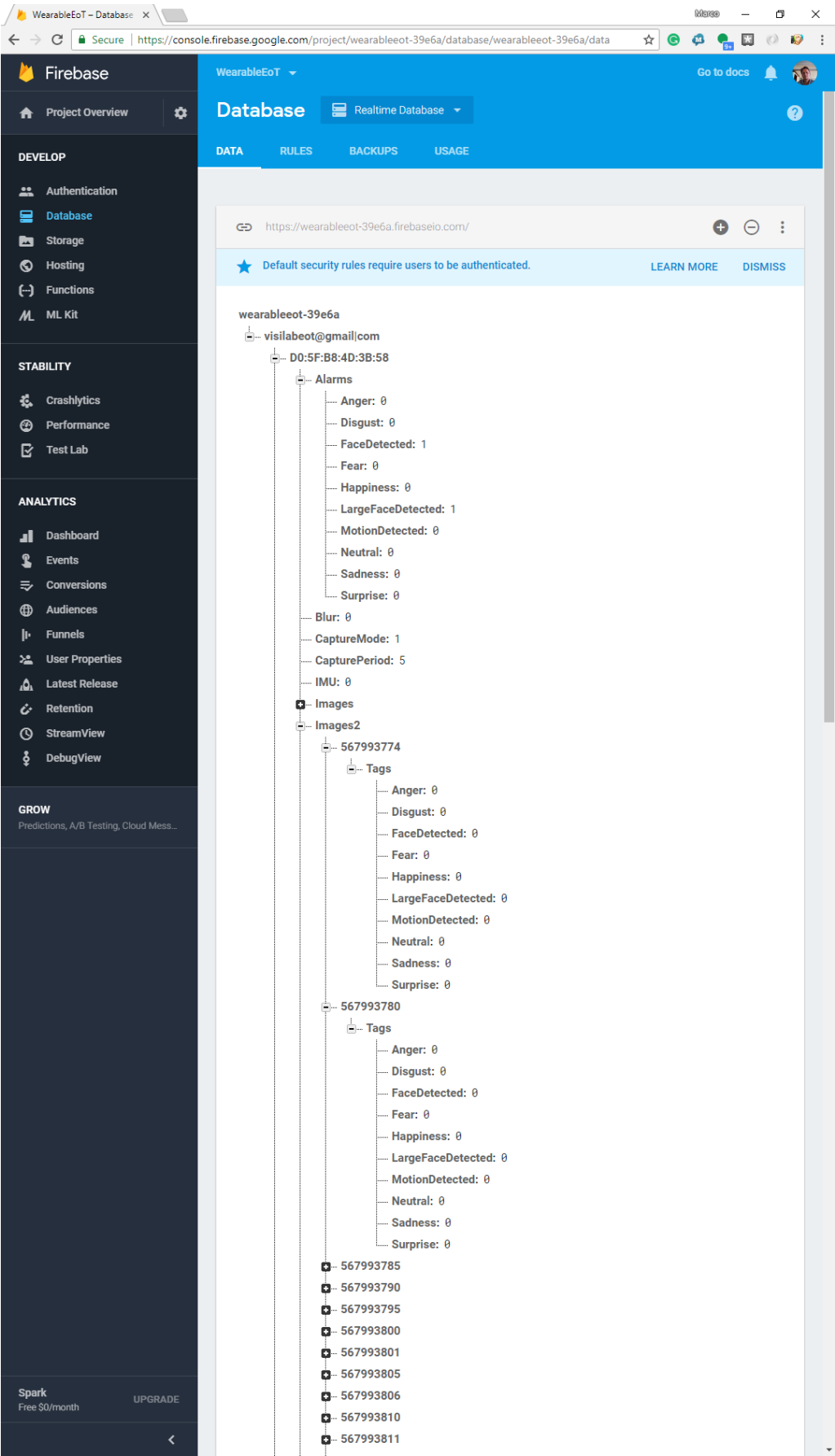


Figure 29: Database snapshot 2

9. CONCLUSIONS

In this deliverable, the configuration application of the Flexible Mobile Camera demonstrator has been described. The document collects the functional and non-functional requirements and the software architecture of the EoT Device application has been explained.

The demonstrator was divided in different modules. Each module has a main objective and achieves one or more functional requirements.

The components of the configuration application and the required permissions have been indicated.

Finally, the use cases of the configuration application have been exposed. Each use case is accompanied by a description and its operation.

- End of document -