WebIO - Home Security

Ana Margarida Silva
up201505505@fe.up.pt

Bruno Manuel Piedade
up201505668@fe.up.pt

Danny Almeida Soares
up201505509@fe.up.pt

May 30, 2019

Abstract

Nowadays, almost every device is connected to the internet and has some mechanism that allows it to be controlled remotely. This concept is called Internet of Things (IoT). IoT is an area in constant growth, with more uses and applications being made every single day, as the world is becoming more technological as time goes by.

In this article we document the research and development of a project developed in the area on IoT, called WebIO - Home Security. This project consists of a simple security system based on the connection of common day-to-day devices to the internet. There are lots of traditional security systems for houses, that simply record camera feed, store it in a hard drive and ring a bell when an intruder is detected. For example, Prosegur or Securitas, which are world renown for their security systems, used to implement these traditional systems for years.

As a (positive) consequence of the technological evolution, traditional security systems became insufficient to satisfy all the customers requirements, and companies had to step it up and start applying IoT concepts to the systems. So, taking into account the Prosegur example, recently they started selling smart security systems, based on IoT, for example, their package "Prosegur Smart Home" [1], which allows the user to monitor and control lighting, temperature, humidity, power outlets, while also doing smoke detection, gas detection and, obviously, checking the security of the house with cameras and sensors. The main advantage of these modern systems, besides allowing the user to control a lot of devices, is that all that control and monitorization can be done with a smart phone, or even by voice.

The system we developed and are documenting in this article fits on the modern systems category, having the basic features of a modern security system.

1 Motivation

As a project for the Markup Languages and Document Processing course, we chose to make and application related to the *IoT* area. This application should implement an API which will gather, process and return data. We decided to focus this application in the subject of Home Security, as it is an area of interest to us as well as an important and useful project.

Taking into account the evolution in security systems already described, it made sense for us to research and invest time in this area, because *IoT* is becoming more influential in technology, and smart houses are more popular each day that goes by.

So, we decided to try to develop a basic modern security system, *IoT* based, which should allow the users to control their house alarm systems with a smart phone, receiving notifications and camera images when the systems detect intruders, and also allow them to have access to the cameras' live feed, meaning that they can at any time see what is happening at home. All this was done keeping in mind that the system should be extensible, because at any time new modules can be added, to control other devices at home, so we developed everything as modular as we could.

2 Objectives

Our goal was to develop a simple modern security system, with a Raspberry Pi, a buzzer and a camera. The idea was that when the house owners activated the alarm, the camera became active and was triggered by any movement it detected. This detection was made with the use of YOLO object detection system. After detecting a person in the frame, an alarm would

begin to sound through speakers and the user would be alerted.

To complete the system and allow the user to monitor the system from anywhere, we developed a mobile app that allowed the user to log in and set or unset their alarm, as well as to have live feed from the cameras and access to a history of the alarm.

So, resuming, the complete system would have a Raspberry Pi with a buzzer and a camera connected to it, running a REST API. The system would allow the user to activate or deactivate an alarm, which would turn the camera on and analyze its feed, to check for intruders. On detection of an intruder, by use of an object detection model, the system activates the buzzer and sends an alert (notification) to the user's smart phone and a camera image with the intruder.

3 State of the Art

There are already systems similar to what we developed, because home security systems are very common and a necessity for today's society. With the reduced cost of Raspberry Pi's and the accessibility of programming resources, there are variants of the system we implemented, but not one exactly like ours. As mentioned above, Prosegur has modern security systems, based on *IoT*, as well as ADT [2], or FrontPoint [3]. However, the systems mentioned are professional systems, very expensive, that are not available to everyone. With our system, what we wanted was a simple and affordable system, yet still functional and properly working.

In this case, we believe that having a system built on a device so small and compact, that can be easily expanded is great. It is also very important to have instant access to camera feed, allowing the users to alert authorities in case of an emergency or to turn off the alarm in case of a false alarm. Furthermore, this system can later be integrated into a bigger system of an intelligent house, being home security one of it's modules.

4 Data Sources

The data sources of the system are the USB camera, which is used to capture video from its surroundings, to be analysed and live streamed; the RSS news feed [4] from the *Polícia de Segurança Pública* website [5] is also a data source, which is received and processed by the mobile application.

5 Data Models

The system needs a database to store information about the users of the system, as well as an history of alerts. We used a simple mongoDB database.

```
db.users.save({
    _id: ObjectId(123124123123)
    email: "example@gmail.com",
    firebase_token: "aNS12...aDf2df"
    name: "John Doe"
    password: "iliketrains"
})
```

```
db.history.save({
    _id: ObjectId(12329073)
    date: "2012-10-15T21:26:17Z",
    type: "Alert"
    imagePath: "images/12329073.png"
})
```

To start making request to the API regarding the alarm, the user needs to authenticate first. This is done with a POST request, and here's an example of an authentication request:

```
{
    "email": "example@gmail.com",
    "firebaseToken" : "as92ADF...1gF",
    "password": "bongocat"
}
```

The API responds with the access token, as well as the user info, except the password:

```
{
    "_id": "12...20d",
    "email": "example@gmail.com",
    "name": "example",
    "firebase_token": "as92DF...1gF",
    "token" : "a323IF...2Kp",
    "admin": false
}
```

To communicate with the users, the API in Flask receives and returns JSON requests and responses. The user is capable of getting the state of the system, as well as making some changes to it, such as turning it on and off. To turn on and off the alarm, the user must send a POST request to endpoints */alarm* and */alarm/stop*, respectively. These endpoints return the status and the history entry created by that action, which are similar to this:

```
[
    {
        "id": 1231924124,
        "date": "2018-08-19 18:51:06",
        "type": "Turn On"
    }
]
```

A possible response to a GET request to know the state of the alarm, which is endpoint */alarm/status*, could be true or false, which translates to alarm on or off, respectively.

```
{
    "status": true
}
```

The user can also get the history of the alerts and actions of the system. Below, we can see an example of the answer of the system to a GET request of endpoint */history*:

```
[
    {
        "id": 1231924124,
        "date": "2018-08-19 18:51:06",
        "type": "Turn On"
    },
    {
        "id": 1231293,
        "date": "2018-07-19 18:51:06"
        "type": "Alert",
        "image": "<URL>/1231293.png"
    },
    ...
]
```

When accessing the live feed from the camera, the user must make a POST request to */livestream/start* so that the API can start sending frames to a specific endpoint. The same action must be made to stop the livestream, but instead the request must be made to */livestream/stop*.

The livestream can be accessed by accessing with a video player (for example VLC) or the *img* tag in a browser.

Regarding the RSS feed [4] from *Polícia de Segurança Pública*'s website [5] the data received is in XML format. It contains information about the RSS channel such as title, description, copyright, etc and most importantly details about each news, in particular title, author, publishing date, and description which are parsed and stored in the app to be displayed. The description, in HTML, is parsed extracting all the relevant text and if present associated links and image.

The following is a snippet of a news article:

```
...
<item>
    <title>SANTARÉM</title>
    <link>http://www.psp.pt/Lists/Not
    cias/DispForm.aspx?ID=1757</link>
    <description>
        <![CDATA[<div><b>Data Publica
        ção:</b> 16-05-2019</div><div>
        <b>Corpo:</b><div "corte" e ...
        href="http://www.psp.pt/Imagens
        %20Noticias/_w/2019-05-15%jpg.
        jpg"></a></div>
    ]]>
    </description>
    <author>José Joaquim Lima
    Meira</author>
    <pubDate>Thu, 16 May 2019 07:10:3
    0 GMT</pubDate>
    <guid isPermaLink="true">http://
    www.psp.pt/Lists/Notcias/DispForm
    .aspx?ID=1757</guid>
</item>
...
```

6 User requirements

The user requirements for the system are defined in the following user stories:

- As a **Visitor**, I want to log in through the app so that I can control the system.
- As an **User**, I want to turn the system off through the app so that I can control its activity.
- As an **User**, I want to turn the system on through the app so that I can start monitoring the area.
- As an **User**, I want to access the camera feed in the app so that I can visualize the secured area anywhere and anytime.
- As an **User**, I want to be informed with a push notification when the system senses someone so that I know that my house is insecure.
- As an **User**, I want to view the history of past activities including previous alarms and actions so that I can keep track of past events.
- As an **User**, I want to log out through the app so that no else has access to my account.
- As an **User**, I want to see the news from the *Polícia de Segurança Pública* website [5], so that I can be informed of recent crimes and dangerous situations.
- As an **User**, I want to see each of the pieces of news from the *Polícia de Segurança Pública* website [5] in more detail, so that I can know what happened.

7 System architecture

The figure 1 is the system architecture of the home security project. The Raspberry Pi is running a Flask API with a MongoDB database, which interacts with a USB camera and a buzzer. A mobile application makes requests to the API. The Firebase Cloud Messaging application receives events from the API and sends a push notification to the mobile application. The mobile app also access the RSS feed [4] from the *Polícia de Segurança Pública* website [5], which is processed and displayed accordingly.

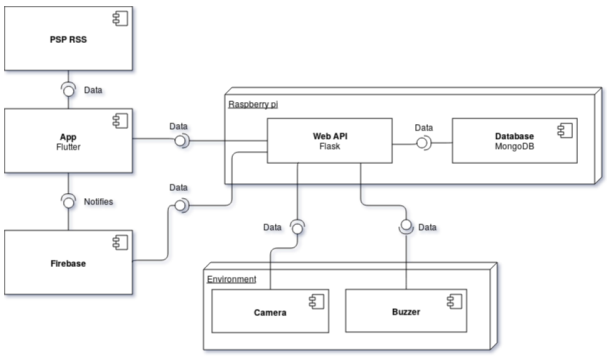

Figure 1: *System Architecture*

8 Mockups

In the figure 2 the user logs into the application with their email and password.

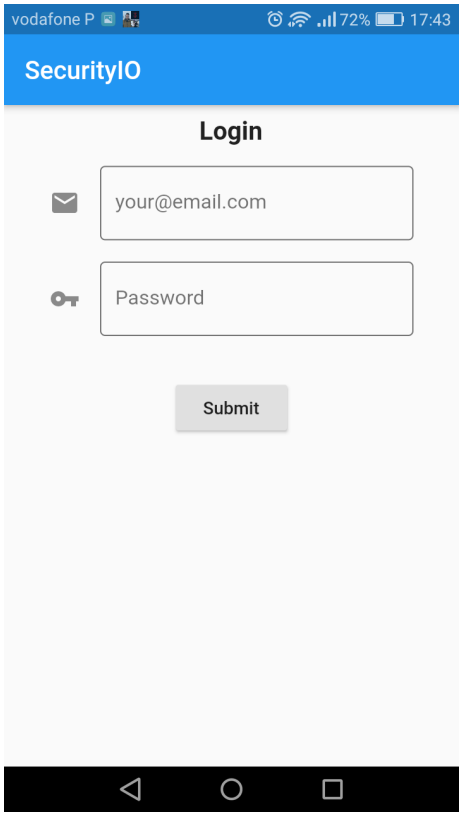

Figure 2: *Login View*

In the figure 3 the user can turn the alarm on or off and consult the history of events that happened to the system. In the alert events, it is possible to see the image of the intruder that was captured by the camera.

In the figure 4 the user can access the live feed from the system's camera.

In the figure 5 the user can access the news feed from the *Polícia de Segurança Pública* website, as well as access each individual news.

In the figure 6 the user can access the information and image of a specific news selected in the news feed from figure 5.

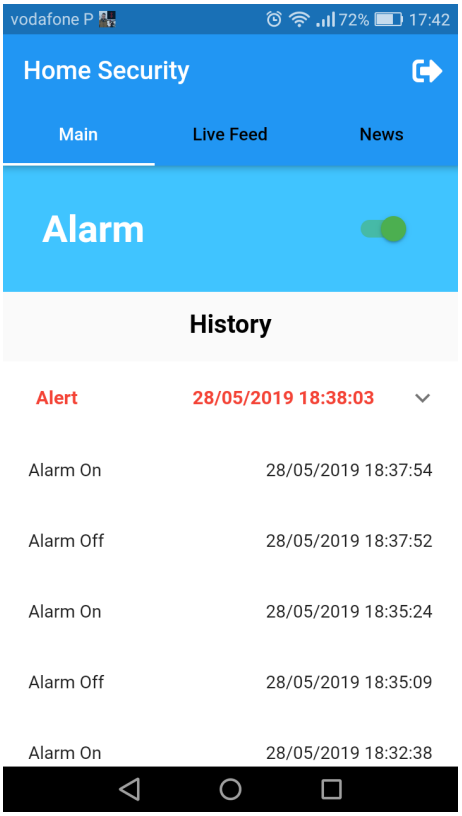

Figure 3: *Main Page View*

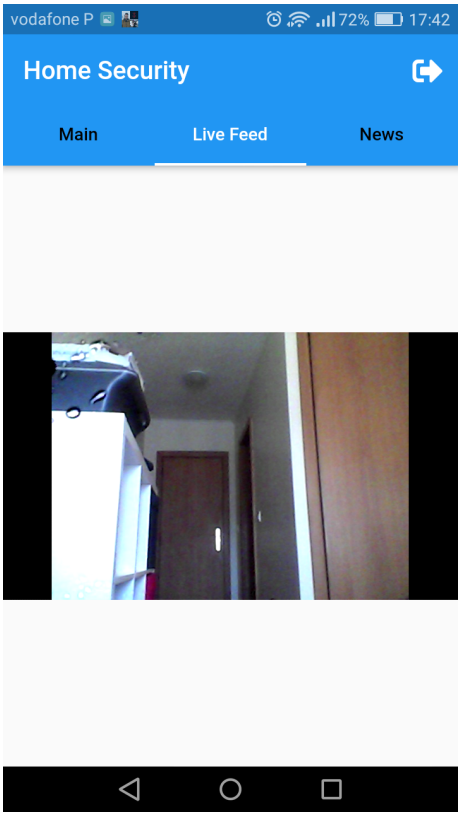

Figure 4: *Camera Live Feed View*

Figure 5: *News Feed View*

Figure 6: *News View*

9 Calendar

01/04 - 07/04	Setup of the Raspberry Pi
08/04 - 14/04	Endpoints to list history, login,state of alarm
15/04 - 21/04	Flutter application with histories and login
22/04 - 05/05	Livestream support
06/05 - 12/05	Intruder detection
13/05 - 23/05	Push Notifications
24/05 - 30/05	Report

10 Discussion

In the beginning stages of the project, we started by developing an API in NodeJS which was responsible for receiving and answering requests from a future mobile application, as well as handling frames from the USB camera connected to the Raspberry Pi, where the API was running.

The first solution we found for the detection of intruders was by *subtracting* the current frame with an average frame from the empty room. This implementation was very sensible to changes in lighting as well as shadows, giving several false-positives.

During the development of the alarm, as well as the history and login, we started developing the mobile app with the Flutter framework. Later came the development of a solution for the livestream, and this feature became the biggest problem of the project.

Several streaming methods were implemented to get the feed from the camera to the flutter app. Soon we noticed that NodeJS was a problem because it was single threaded, which created difficulties in the fluidity of the feed as well as running the alarm at the same time. This problem derived from the code that detected the intruder being in Python, since the OpenCV libraries for NodeJS are very limited and did not have the necessary methods.

We reached a point where we had to change the API to Python, and we chose Flask, because it is light weight and we needed a limited number of dependencies. Taking advantage of the multi-threaded capacities of Flask, we redid

the login, history actions and livestream. We also changed the intruder detection from image processing to YOLO Object Detection, which detects if there is a person in a frame.

After succeeding with the livestream and the alarm, we implemented push notifications, to alert the user, even if the mobile app is closed. This was done with Google's Firebase Cloud Messaging, which is supported by Flask. So, whenever an alarm is triggered, the users of the household receive a push notification on the smart phone.

In the final stages of the development, we implemented the news feed from the *Polícia de Segurança Pública* website, where we had some problems with blocking, due to many requests made to their website. We also made a simple web page to sign in a new account in the / endpoint, where the sign in has to be validated by using admin credentials.

It is important to note that the alarm takes at most one minute to run the object detection model and return a result. During this analysis, the livestream and everything else runs normally. Despite optimizations and using faster detection models, it is impossible to improve, due to the limitations of the processing power of the Raspberry Pi. In spite of several problems and technology constrains, all the initial objectives of the system were accomplished and some new ones were added.

11 Conclusions

After finishing the development of the project, we believe that it went well, despite the obstacles that appeared, since we managed to overcome them all.

Looking at the objectives that we set on the beginning of the project, all objectives were completely accomplished, which means that by the end of the project we had developed a simple modern security system, on a Raspberry Pi, using a camera and a buzzer, connected to the Raspberry Pi. The Raspberry Pi runs a REST API that processes all the information and communicates with the users through a mobile app installed on their smart phones. The system detects intruders and alerts the users on the phone using push notifications and alerts inside the app. The users can activate or deactivate the alarm at any time and access the live feed from the camera on the app.

After completing the objectives we had set, we decided to explore the system a bit more and include news from the Polícia de Segurança Pública's website, which are related to security.

So, by the end of this project, we have a fully functional security system (video based) that informs the users with up-to-date pieces of news.

References

[1] Eren Golge: RaspberryPi Home Surveillance with only 150 lines of Python Code, accessed in 24-02-2019

https://hackernoon.com/raspberrypi-home-surveillance-with-only-150-lines-of-python-code-2701bd0373c9

[2] Being Engineers: How to Make Raspberry Pi Webcam Server and Stream Live Video || Motion + Webcam + Raspberry Pi, accessed in 12-03-2019

https://www.instructables.com/id/How-to-Make-Raspberry-Pi-Webcam-Server-and-Stream-/

[3] Miguel Grinberg: Flask Video Streaming Revisited, accessed in 12-05-2019

https://blog.miguelgrinberg.com/post/flask-video-streaming-revisited

[4] Sunita Nayak: Deep Learning based Object Detection using YOLOv3 with OpenCV (Python / C++), accessed in 13-05-2019

https://www.learnopencv.com/deep-learning-based-object-detection-using-yolov3-with-opencv-python-c/

[5] Polícia de Segurança Pública: RSS feed for Site PSP, accessed with the mobile app

https://bit.ly/2Wc2vFh

[6] Polícia de Segurança Pública: Website, accessed in 15-05-2019

http://www.psp.pt/Pages/defaultPSP.aspx