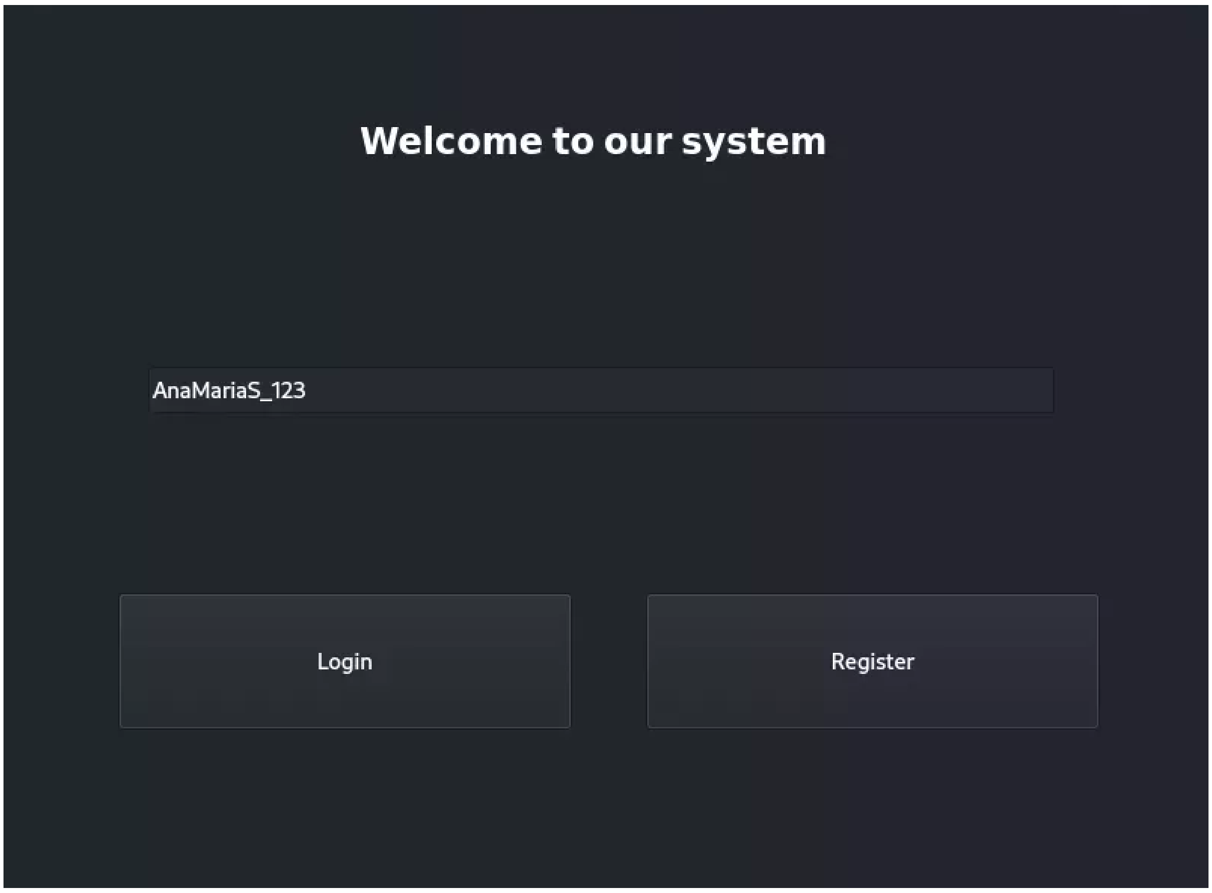
universidade de aveiro
deti departamento de eletrónica, telecomunicações e informática

BioGuardian



Maria Rocha, 93320
Pedro Souto, 93106

Group 2, Biometrics
Cybersecurity Masters, 2021/2022

Contents

1. Introduction

This project consists in the development of a biometric system to either identify or authenticate users by combining at least two biometric measures. The system should include a graphical user interface, simpler but adequate to the system purpose and context of use.

BioGuardian is a multimodal authentication system that combines two factors of its users: the hand geometry and the voice. Its interface was developed with PyQt5, a module of Python.

The authentication system discussed here may be applied in several environments. It could be useful for controlling the entrance of individuals in a smart house. It may be used by enterprises to limit access of certain employees to specific rooms. It could even be placed in laboratories or research offices (e.g. IT or IETA from the University of Aveiro) to replace the card system of researchers who have access to some rooms.

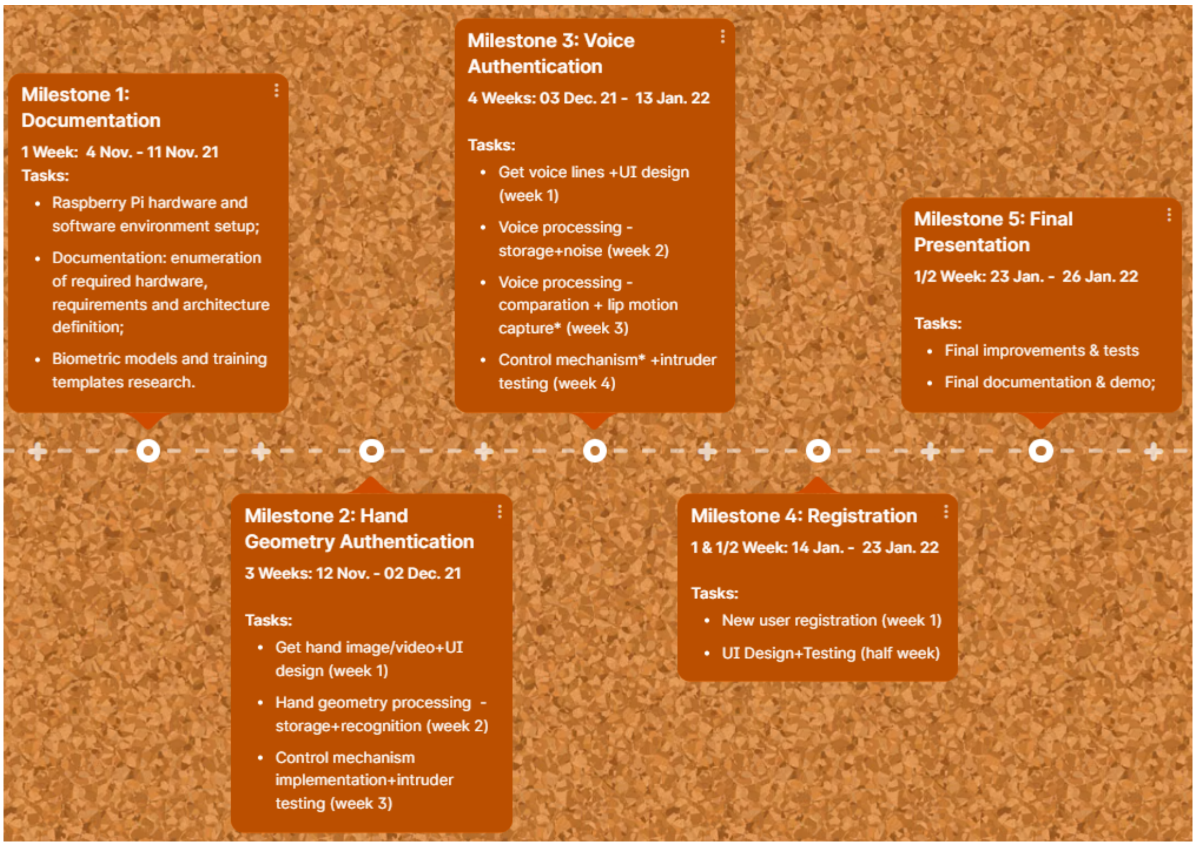
The present document aims to detail the decisions made and steps taken during the development of the project as well as the difficulties found and the improvements to be done as future work.

2. System Development

➔ Timeline

After deciding the purposes of the system and the biometric measures to be applied, it was necessary to define a timeline for the project. The project was divided into five milestones, each of them with realistic goals, considering the length of the development team and the time available until the deadline - 27th of January 2022.

Below is the referred timeline, with each task duration and respective details:



Milestone 1: Documentation
1 Week: 4 Nov. - 11 Nov. 21
Tasks:
- Raspberry Pi hardware and software environment setup;
- Documentation: enumeration of required hardware, requirements and architecture definition;
- Biometric models and training templates research.

Milestone 2: Hand Geometry Authentication
3 Weeks: 12 Nov. - 02 Dec. 21

Tasks:
- Get hand image/video+UI design (week 1)
- Hand geometry processing - storage+recognition (week 2)
- Control mechanism implementation+intruder testing (week 3)

Milestone 3: Voice Authentication
4 Weeks: 03 Dec. 21 - 13 Jan. 22

Tasks:
- Get voice lines +UI design (week 1)
- Voice processing - storage+noise (week 2)
- Voice processing - comparation + lip motion capture* (week 3)
- Control mechanism* +intruder testing (week 4)

Milestone 4: Registration
1 & 1/2 Week: 14 Jan. - 23 Jan. 22

Tasks:
- New user registration (week 1)
- UI Design+Testing (half week)

Milestone 5: Final Presentation
1/2 Week: 23 Jan. - 26 Jan. 22

Tasks:
- Final improvements & tests
- Final documentation & demo;

➔ Hardware

In the early stages of the project it was made a list of hardware devices needed to deploy the solution, considering its context of use.

The first idea was to use a Raspberry Pi 4 and two sensors: a Pi Camera, for hand recognition and a microphone connected with USB cable, for voice recognition. The Raspberry is a smaller device, cheaper and easier to set up in several environments, allowing our system to be placed almost everywhere. However, after gathering the components, configuring them and trying to run a simple script for processing some images obtained by the Pi Camera, it was concluded that the Raspberry wasn't a good solution. Its processing time was too slow for the purpose of the system in development.

Therefore, a different approach was considered. Instead of using a Raspberry Pi, a regular computer was used, with its integrated camera and microphone.

It was also decided that futurely the system would connect to a USB camera, instead of using its integrated one. It will also be useful to have some kind of box, maybe one similar to the figure below, to facilitate the control of the distance from the camera to the hand, its position and the fingers position too.

➔ System Requirements

In the first milestone it was also done some research on the internet and among the other groups of work to gather a list of requirements to the system.

Nextly, its a list of requirements expected to the solution:

- User interface must be simple and intuitive;
- The user may be able to easily understand which step it should take when interacting with the system;
- The user shall be able to understand how to proceed when authentication or regist fails;
- The user may be able to understand when the login was successful;
- New users' registration must be possible and as efficient as possible;
- Authentication of a non-intruder user must fail few times;
- The templates used to train the system may be diverse and relevant to its context of use
- Audios saved in the server should be encrypted while in rest in the server
- The system must detect record audios against live voices;
- The system mustn't be tricked with images instead of live hands.

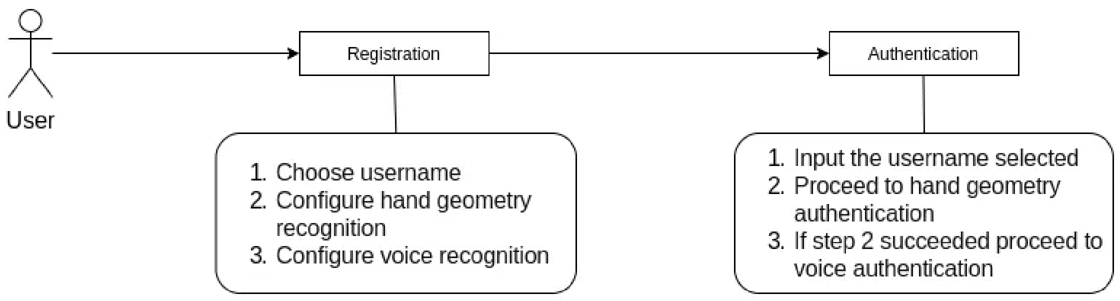
The last two are related to intruder detection mechanisms that weren't implemented for this first version of the system.

➔ System workflow

Along with the gathering of the requirements, it was stated in this stage, the workflow of the application.

As seen in the following diagram, the user needs to register first, by choosing a unique username. Then he should interact with the system exposing its traits, so that a template for him is created and saved to be used later for authentication.

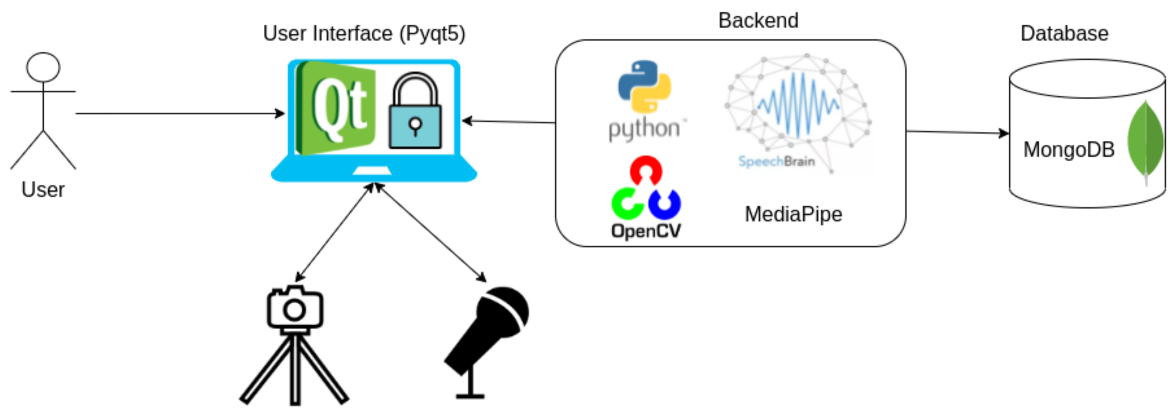
Later on, to authenticate, the same user only needs to use the username chosen in registration and use his personal traits. It was decided that the user needs to be authenticated by his hand geometry and only after that he is able to input his voice for authentication. If both measures allow the system to recognize the user that is trying to authenticate, then he is authenticated correctly. Otherwise, some error messages will be shown and the user may need to repeat the process.


User
Registration
Authentication
1. Choose username
2. Configure hand geometry recognition
3. Configure voice recognition
1. Input the username selected
2. Proceed to hand geometry authentication
3. If step 2 succeeded proceed to voice authentication

➔ Architecture Definition

In the beginning of the milestone 2, an architecture diagram was designed to help the development team visualize the modules being used as well as its connections. It suffered some changes during the development of the project, because some libraries weren't selected by the time it was created.

Below, the diagram shows that the frontend is developed with PyQt5 and the backend is in Python. The libraries enumerated: opencv - for videos and images processing -, mediapipe - for hand recognition - and speechbrain - for voice recognition - are used to improve the system accuracy and facilitate its development. Also, the users registered are saved to a MongoDB database. This is a NoSQL database, which is able to handle large volumes of data at high speed.

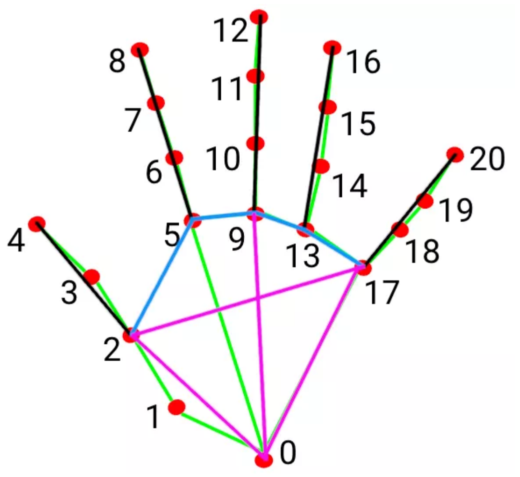


➔ Hand Recognition Module

Implementation:

The library MediaPipe is used to, from the hand, extract several key points (landmarks presented below), which are then used to calculate the distances presented in the figure below (distance between fingers, size of the fingers, distance from wrist to pinky, wrist to thumb and wrist to middle finger). 1000 frames are obtained, and for each one a value is calculated based on said distances. Then, these values are averaged resulting in a metric which is saved to that specific user entry in the database. When calculating the metric again, the value is compared with the previous one with an error of ± 0.003.

Hand Distances Diagram



Trait Future in Biometric Systems:

Though this is an interesting biometric measure it has some problems that are quite hard to overcome and compromise the future of this trait regarding biometric authentication systems.
The hand geometry is a unique measure, but it may change a bit with age, at least while the bones are still growing. Also, this part of the human body is constantly exposed to injuries that are impossible to predict and train the system about.

Control Mechanisms:

Regarding the implementation of control mechanisms to detect intruders there are some measures to be applied to improve the system robustness:

- To consider the skin tone of the hand in evaluation rather than just its geometry may help in some situations;
- To define that the user should do a certain gesture or simply move his hand during the authentication process may be enough to detect whether there's somebody trying to trick the system with a photo of another person's hand.

➔ Voice Recognition Module

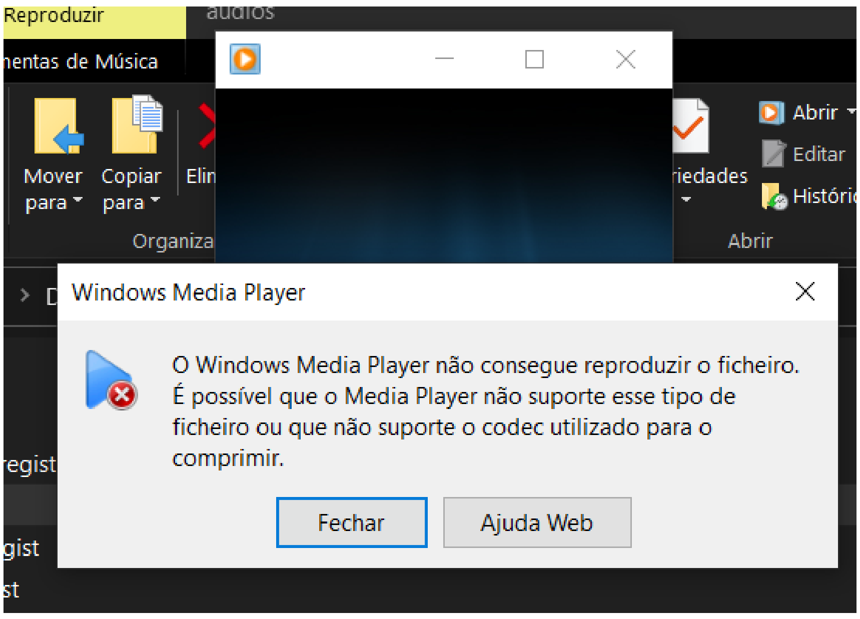
Implementation:

First we implemented a voice recorder to save the users voice files. Then we used the library SpeechBrain to perform speaker verification with a pre-trained neural network. Through the user recorded voice files, we get the speech waveform which is then used by the pre-trained neural network to calculate a score and a prediction (the score is associated with the cosine distance and the prediction is 1 or 0 depending if the voice files are from the same speaker or not).

Initially, instead of the SpeechBrain library, the implementation was based on a fast fourier transform and a convolutional neural network but the accuracy was very low, with a high number of false positives resulting in the abandonment of such methods.

Data Encryption:

In terms of the user data saved in our database, we don't think that a metric calculated by us or a username are sensitive data, thus no encryption was made to these values. However, in terms of the regist audios in rest in a folder, we decide to encrypt them with a symmetric encryption module named Fernet. For that, each user has a different key that is saved in the database, and used later to encrypt and decrypt the audio file when necessary. With this mechanism an attacker that is able to access the 'audios' folder in the server cannot reproduce the audios, an error similar to the one in the figure below appears.


Windows Media Player

O Windows Media Player não consegue reproduzir o ficheiro. É possível que o Media Player não suporte esse tipo de ficheiro ou que não suporte o codec utilizado para o comprimir.

Trait Future in Biometric Systems:

There are some conditions to be considered when choosing this trait for biometric systems. This trait may introduce some instability in the system due to natural conditions that are hard to control such as, colds, changes in the voice with age and ?? Further than that it's important to do some processing to suppress any noise that could compromise the accuracy of the system. When compared with other metrics, even with the problems presented, this one is considered to be a good choice, with much to explore in the future.

Control Mechanisms:

In order to make a system with this biometric resilient against impersonification attacks, it’s relevant to implement the following detection mechanisms:

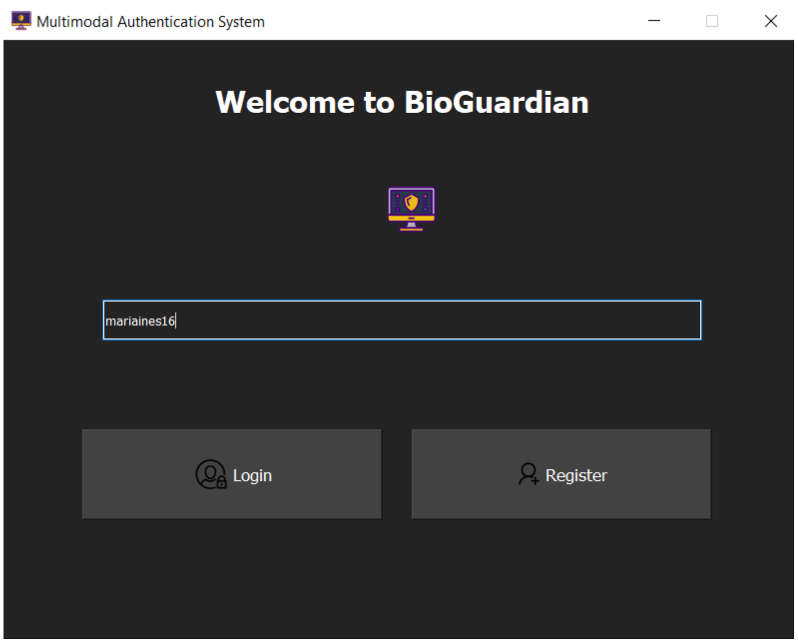
- To prevent attacks with records of the users’ voice, it may be used a camera to detect if there’s lip motion when the user is speaking or not;
- Also to prevent recorded attacks, it may be developed a mechanism to select randomly some words that the user needs to speak by the time their appear in the screen;
- In trained systems it is important to have models trained with diversified samples that cover the systems’ context of use. This may include several languages, expressions and pronunciations.

➔

➔ Graphical User Interface

The graphical interface started to be developed in milestone 4 and continued to milestone 5. It is simple and allows a user to follow the workflow explained in some sections above.

There's a main window where the user is confronted with the two scenarios: register and login. In both scenarios the user should input his username. In the register, he hadn't chosen one already, thus he's able to do it.

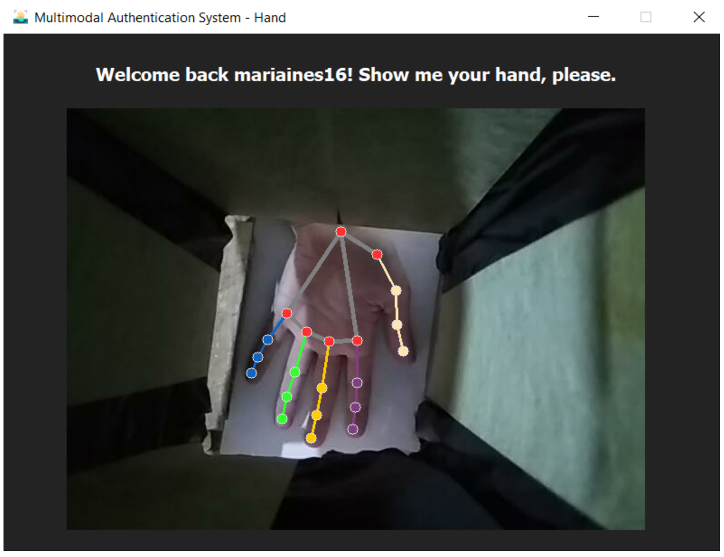


Legend: Main Window

There are two "second" windows related to the hand recognition part of the system, one for register and another for login.

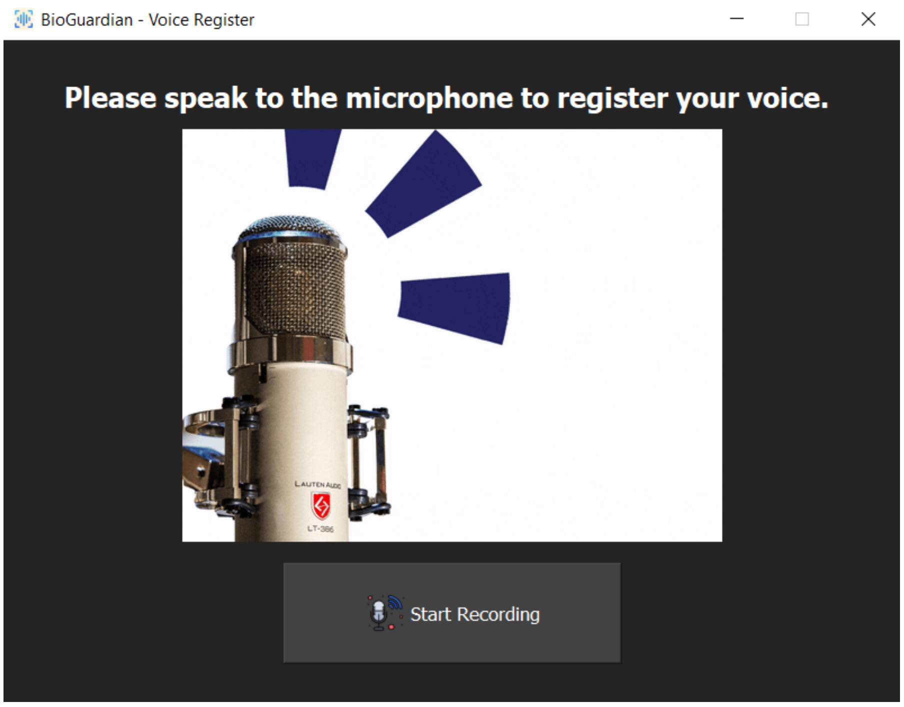
In the first scenario, the user needs to register his hand by showing it to the camera. The system will process the data acquired from the images and save it in the database along with the username of the logged user.

In the second scenario, the user also needs to show his hand so that the data acquired is compared with the one previously obtained in the register.


Multimodal Authentication System - Hand
Welcome back mariaines16! Show me your hand, please.

Legend: Hand Window

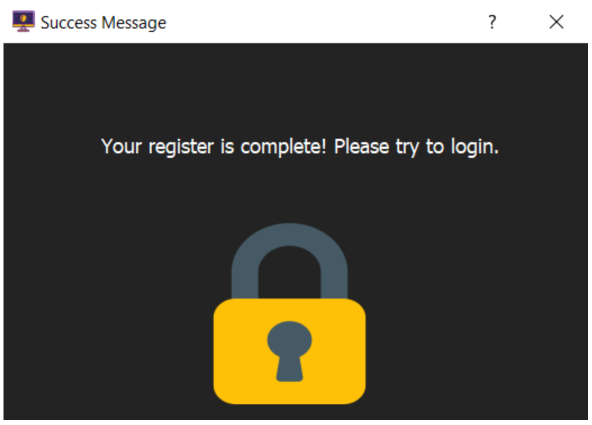
The windows related to voice recognition, shown in the figure below, only appear when the logic of the hand recognition part of the system is completed with success. Besides, here the analysis done to the voice lines gathered from the audio is done directly between the files recorded, therefore, there's no data saved in the database.


BioGuardian - Voice Register

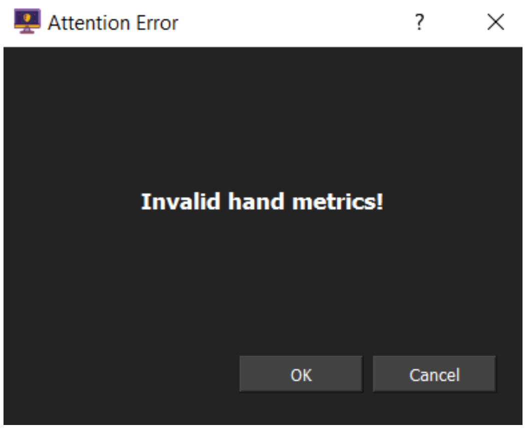
Please speak to the microphone to register your voice.

Legend: Voice Window

Finally there are also two dialog windows with different scopes of messages: error, successful registration and successful authentication - as shown below.


Success Message

Your register is complete! Please try to login.


Attention Error

Invalid hand metrics!

➔ How to use the system

Our system requires some specific hardware discussed in a topic above. In terms of software, the computer should have a windows operating system.

To run the program it's necessary Python 3.9.10 and the following requirements, that can be installed with Pip:

- pymongo
- opencv
- mediapipe
- speechbrain
- PyQt5
- PyAudio
- PySoundFile
- cryptography

After the requirements installation, it's important to create a database in mongo with the name "biometry" and a table, inside of it, called "user_Data".
For this is easier to install MongoDB and use it in command line, following this guide: https://docs.mongodb.com/guides/server/install/

To have a portable camera, it's suggested the installation of DroidCam in the computer and in a mobile phone (with a camera): https://www.dev47apps.com/ . In this case, in the script hand_backend.py the function of videocapture should change its number to 1, instead of 0, to select the new camera source.

Finally, to use our program, the script main_window.py must be run in a terminal (powershell) with administrator permissions. With the program running, the user should follow the guide below to test every feature.

User Guide:

➢ Register Phase:

1. The user needs to choose an username and clicks register;
2. The user shows his hand until the system changes to another window;
3. On the new window, the user clicks on 'Start Recording' and waits until the message of successful registration appears. (That will be around 15 seconds)

➢ Login Phase:

4. The user types his username and clicks login;
5. The user shows his hand until the system changes to another window;
6. On the new window, the user clicks on 'Start Recording' and waits until the message of successful login appears. (That will be around 15 seconds)

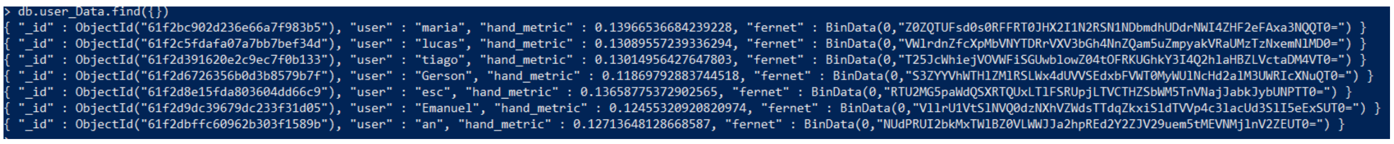
If any message error appears the user must restart the process in the main window, following the instructions.

➔ Experimental Results

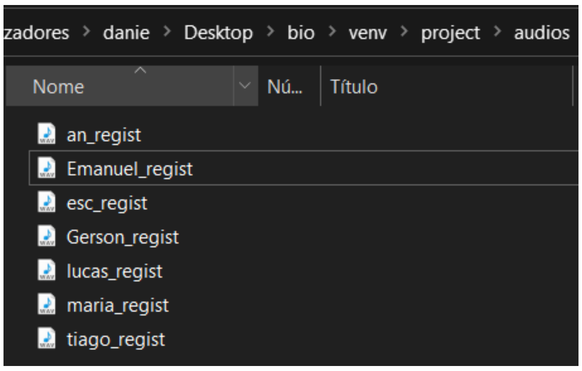
Various tests were conducted in order to ensure the accuracy of the various components that compose the system. In the first place, regarding the hand geometry module, a different number of frames were obtained to calculate the hand metric, varying between 100 and 1000. This allowed us to conclude that more frames captured meant less floating of the hand metric value, as this number of frames somewhat compensates possible movements of the user. Tests were also conducted regarding the stabilization of the image obtention, this is, ways to maintain the conditions relatively the same each time a user interacted with this module. Tests conducted include but are not limited to: the calculation of the metric with a hand drawing guide (more accurate) and without a hand drawing guide (less accurate); the calculation of the metric using a box to maintain the height between the camera and the hand (more accurate), and without the box (less accurate). The previously described tests allowed us, after correction and implementation, to have a difference of ±0.003 each time the hand metric is calculated (value corroborated by testing with 4 different users).

As for the voice module, testing compared 6 audio recordings of the same user in relation to other users and himself. First, 3 different voice files of a user were recorded, in which the things spoken by said user were completely different. Then, 3 more voice audio files were recorded of the same user saying the same things recorded in the initial 3 files. Finally, each individual voice file was compared to the remaining ones of the same user and to voice recordings of other people. This was done to attest that the system can still have a pretty good accuracy when things aren't said in the exact same way that they were previously, and that also, the system can accurately distinguish different speakers. The previously mentioned tests were a success, with the system having almost 100% accuracy.

During the project presentation some colleagues have also tested our system. Below we have two figures one with their entries in the database with the respective usernames, hand metrics and fernet keys, and another with their voice register files. Our colleagues interact with the system as regular users, without previous training to use it and the results gathered were quite good. In fact, only in one situation a user was able to authenticate as another, the resting attempts to use the system worked as expected. We have no better explanation for the one attempt that went bad, rather than the existence of a false positive, something that has always some probability to happen in these systems.



```
> db.user_Data.find({})
{ "_id" : ObjectId("61f2bc902d236e66a7f983b5"), "user" : "maria", "hand_metric" : 0.13966536684239228, "fernet" : BinData(0,"Z0ZQTUFsd0s0RFFRT0JHX2I1N2RSN1NDbmdhUDdrNWI4ZHF2eFAxa3NQQT0=") }
{ "_id" : ObjectId("61f2c5fdafa07a7bb7bef34d"), "user" : "lucas", "hand_metric" : 0.13089557239336294, "fernet" : BinData(0,"VWlrdnZfcXpMbVNYTDRrVXV3bGh4NnZQam5uZmpyakVRaUMzTzNxemN1MD0=") }
{ "_id" : ObjectId("61f2d391620e2c9ec7f0b133"), "user" : "tiago", "hand_metric" : 0.13014956427647803, "fernet" : BinData(0,"T25JcWhiejVOVWFiSGUwblowZ04tOFRKUGhkY3I4Q2hlaHBZLVctaDM4VT0=") }
{ "_id" : ObjectId("61f2d6726356b0d3b8579b7f"), "user" : "Gerson", "hand_metric" : 0.11869792883744518, "fernet" : BinData(0,"S3ZYYVhWTHlZMlRSLWx4dUVVSEdxbFVWT0MyWUlNcHd2alM3UWRIcXNuQT0=") }
{ "_id" : ObjectId("61f2d8e15fda803604dd66c9"), "user" : "esc", "hand_metric" : 0.13658775372902565, "fernet" : BinData(0,"RTU2MG5paWdQSXRTQUxLT1FSRUpjLTVCTHZSbWM5TnVNajJabkJybUNPTT0=") }
{ "_id" : ObjectId("61f2d9dc39679dc233f31d05"), "user" : "Emanuel", "hand_metric" : 0.12455320920820974, "fernet" : BinData(0,"VllrU1VtSlNVQ0dzNXhVZWdsTTdqZkxiSldTVVp4c3lacUd3SlI5eExSUT0=") }
{ "_id" : ObjectId("61f2dbffc60962b303f1589b"), "user" : "an", "hand_metric" : 0.12713648128668587, "fernet" : BinData(0,"NUdPRUI2bkMxTWlBZ0VLWWJJa2hpREd2Y2ZJV29uem5tMEVNMjlnV2ZEUT0=") }
```



zadores › danie › Desktop › bio › venv › project › audios

Nome	Nú...	Título
an_regist		
Emanuel_regist		
esc_regist		
Gerson_regist		
lucas_regist		
maria_regist		
tiago_regist		

3. Conclusion

Overall the main objectives of the project were achieved, thus the system developed is a good solution.

Nonetheless, there are some objectives initially planned that weren't accomplished due to lack of time and struggle in different phases of the development.

The main difficulties found, besides the problems with the Raspberry Pi at the beginning, were related to the libraries and the metrics chosen to do the recognition. It was hard to develop solutions with good accuracy even when several github projects and documentation were found for each biometric trait of the system. A consequence of this is that the hand geometry module, even after a lot of corrections, tweaking and environmental variables control still gives a similar value to different users, which could be associated with the biometric in itself, but also possibly corrected with the use of a better library, artificial intelligence training and a better environment. Also, the voice library requires administrator privileges, which forces the application to have to start with such privileges to work.

In terms of future work, it is necessary to build the "perfect" environment for hand image acquisition, as explained before, and it's also necessary to implement the mechanisms to detect intruders mentioned in each biometric recognition module presented above.

4. References

https://www.geeksforgeeks.org/opencv-python-tutorial/
https://towardsdatascience.com/sample-hand-geometry-biometric-identification-system-b122446e3fdb
https://google.github.io/mediapipe/solutions/hands.html
https://www.geeksforgeeks.org/python-facial-and-hand-recognition-using-mediapipe-holistic/
https://www.nec.com/en/global/techrep/journal/g18/n02/pdf/180218.pdf
https://www.tutorialspoint.com/pyqt5/index.htm
https://www.youtube.com/watch?v=rZcdhles6vQ&list=PLCC34OHNcOtpmCA8s_dpPMvQLyHbvxocY&index=1
https://realpython.com/playing-and-recording-sound-python/
https://huggingface.co/speechbrain/spkrec-ecapa-voxceleb
https://cryptography.io/en/latest/fernet/

Voice Recognition/Identification Repositories consulted during the project development:

- https://github.com/MohamadMerchant/Voice-Authentication-and-Face-Recognition
- https://github.com/emanuelegiona/BS2019
- https://github.com/NaveedShahid/Voice-Authentication-CNN
- https://speechbrain.readthedocs.io/en/latest/API/speechbrain.pretrained.interfaces.html

Hand Geometry Recognition/Identification Repositories consulted during project development:

- https://github.com/ishfulthinking/Python-Hand-Gesture-Recognition
- https://github.com/mrunlikeliest/ROI-Extraction-From-Hand-Image/tree/main/Method_That_Worked