Eduardo Silva Vieira

VideoAnnotator: Tool for Annotation Video

São Luís - MA

2021

Eduardo Silva Vieira

VideoAnnotator: Tool for Annotation Video

Trabalho apresentado ao coordenador do programa de pós-graduação em informática da PUC-Rio como requisito para obtenção de nota na disciplina INF2102-Projeto Final de Programação

Orientador: Prof. Dr. Sérgio Colcher

São Luís - MA
2021

Lista de ilustrações

Lista de tabelas

Sumário

1 Student, Advisor and Lab

Eduardo Silva Vieira is a MSc. candidate at Pontifical Catholic University of Rio de Janeiro (PUC-Rio). He received a B.S. (2019) in Computer Science from the Federal University of Maranhão (UFMA) in Brazil. He is a research assistant at Telemidia Lab – PUC-Rio. His research interests include multimedia systems and artificial intelligence, working mainly on video classification, natural language processing and image processing. Currently, he is working on the Merchant Reconciliation System project, one partnership between BTG Pactual and TeleMídia Lab, which aims to classify macro and micro categories within millions bank's transations. Eduardo is under the guidance of Prof. Sergio Colcher.

Sérgio Colcher received B.S. (1991) in Computer Engineer, M.S. (1993) in Computer Science and Ph.D. (1999) in Informatics, all by PUC-Rio, in addition to the postdoctoral (2003) at ISIMA (Institute Supérieur d'Informatique et de Modelisation des Applications - Université Blaise Pascal, Clermont Ferrand, France). Currently, he teaches in Informatics Department at Pontifical Catholic University of Rio de Janeiro (PUC-Rio). His research interests include computer networks, analysis of performance of computer systems, multimedia/hypermedia systems, Digital TV and Machine Learning. Sérgio is the curretly head of the TeleMídia Lab.

TeleMídia/PUC-Rio is formed by a team of researchers, undergraduate and postgraduate students. It is the lead architect of the Ginga and NCL TV standards in Brazil by Forum SBTVD(Brazilian Digital TV System) and internationally by ITU-T[1]. Ginga and NCL are currently adopted by over 13 countries in Latin America and Africa. The lab has experience in multimedia networking and systems, with a long list of publications and closely related topics [2]. In addition, it has an extensive list of partners that includes both Brazilian industry and international academic institutions [3]. Recent research from the lab has focused on adaptive streaming and immersive multimedia (*e.g.* 3D video and multimodal interactions), then current research focuses on using Deep Learning techniques for pattern recognition in multimedia systems. In particular, the laboratory works with industry partners in the use of these techniques. For instance, it develops research for Petrobras[4] on automatic seismic image analysis. In addition, the lab also acts by disseminating its research within the academic community, in particular through participation and publications in the RNP CT-Video.

[1] <http://handle.itu.int/11.1002/1000/12237>

[2] <http://www.telemidia.puc-rio.br/publication/publications.html>

[3] <http://www.telemidia.puc-rio.br/partners.html>

[4] <http://www.petrobras.com.br/en/>

2 Problem Definition

The wide use of video capture and services for its storage and transmission allowed the production of a large volume of video data. For example, in 2019, over 500 hours of video are uploaded to YouTube every minute footnote url https://kinsta.com/blog/youtube-stats/. The mass consumption of this information by end users brings new challenges related to browsing and searching activities on this video content.

Unstructured databases are a problem, especially in Brazil, we also highlight the video services of the RNP (National Research Network) footnote url http://rnp.br, namely, video @ RNP footnote url http : //www.video.rnp.br and videoaula @ RNP footnote url http://www.videoaula.rnp.br. Thousands of videos with little or no metadata. Therefore, it is extremely complicated to recommend videos by content or speaker.

3 Objectives

In this project, we propose the use of deep learning technologies to develop a facial recognition system for faces in video context. Aiming to allow on-demand analysis of groups of videos and organizing them by speaker.

4 Schedule

Activities	Month				
	1	2	3	4	5
Project design	x				
Implementation and testing		x	x	x	
Writing documentation					x

Tabela 1 – Schedule

5 System Requirements Specification

5.1 Functional Requirements

RF-01 The system should allow the annotator to submit a video for face detection;

RF-02 The system should allow the annotator to visualize the set of detected faces separated by person;

RF-03 The system should allow the annotator to select one or more faces in the set of faces;

RF-04 The system should allow the annotator to delete one or more faces in the set of faces;

RF-05 The system should allow the annotator to move one or more faces to another set of faces;

RF-06 The system should allow the annotator to view a set of videos separated by people;

RF-07 The system should allow the annotator to export a set of generated metadata from videos separated by people;

5.2 Non-Functional Requirements

Non-functional requirements (RNF) are listed below:

Usability

RNF-01 A FAQ (Frequently Asked Questions) and tutorial should be built to help annotators operate the system after a certain amount of training;

Maintenance

RNF-02 Developers must maintain standards for writing and versioning code in order to facilitate the evolution and maintenance of the system;

Reliability

RNF-03 The system shall have high availability with an acceptable delay of 30 minutes;

Performance

RNF-04 The system should save all videos and artefacts generated from it;

Portability

RNF-05 The system must be web and run in any recent browser;

Reusability

RNF-06 The web system should be separated from the API to facilitate the reuse of modules in the future;

Security

RNF-07 The system must only have a single user;

6 Design and Architecture

The system was separated in two modules: the interface module and the API module, as we can see in Figure 1

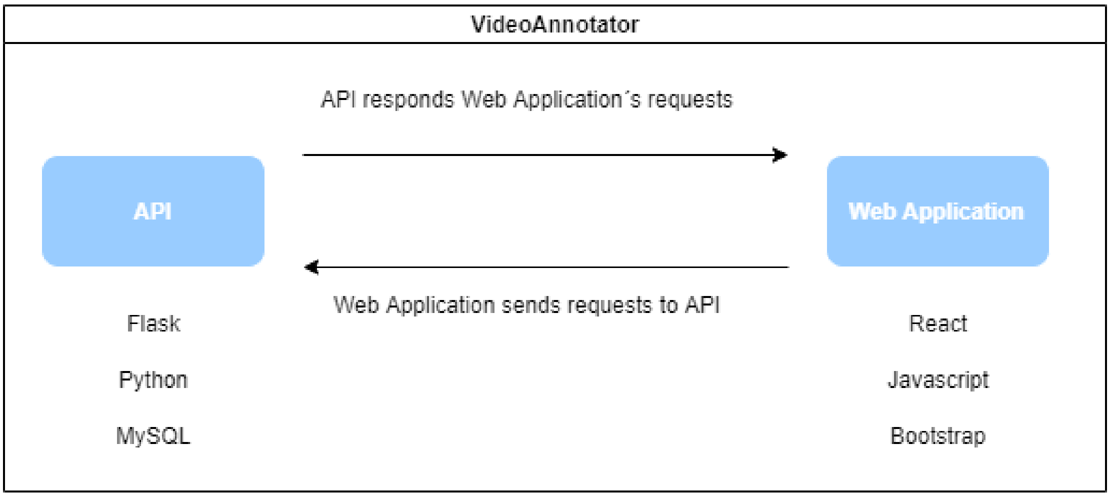

VideoAnnotator
API responds Web Application´s requests
API
Web Application
Web Application sends requests to API
Flask
Python
MySQL
React
Javascript
Bootstrap

Figura 1 – Modules

The interface module and the API module are modelled according to the REST architecture, an architectural model created by Roy Fielding, one of the creators of the HTTP protocol (*Hypertext Transfer Protocol*) as his doctoral thesis.

The *Representational State Transfer* (REST), in Portuguese Representational State Transfer, is a style of architecture that defines a set of restrictions and properties based on HTTP. Thus, REST presents itself as a set of principles and techniques on how to model Web applications.

Using a universal intermediate language, such as XML or JSON, and standard technologies such as HTTP methods, applications modelled using REST can be programmed in different programming languages and technologies, increasing interoperability between systems and communication between different applications. Thus, it is possible that new applications can interact with those that already exist and that systems developed on different platforms are compatible with each other (**??**).

Therefore, by building the modules following the REST architectural style, we allow them to work in a complementary but independent way. REST-compatible web services allow tools to access and manipulate each other's resources through standard HTTP methods such as GET, POST, PUT, and DELETE.

6.1 Diagrams

This section presents the Entity Relationship diagrams of the of API module used, indicating the entities including their fields and their types.

An entity relationship diagram is a systematic way of describing and defining a business process. Entities represent processes that are linked to each other by relationships that express the dependencies and requirements between them.

The API module is composed of 5 (five) entities that are related, as shown in Figure **??**.

In Figure **??**, it can be seen that, in addition to enjoying the benefit of annotation of faces, users can contribute to the evolution of the tool by indicating people to be identified for subsequent analyses.

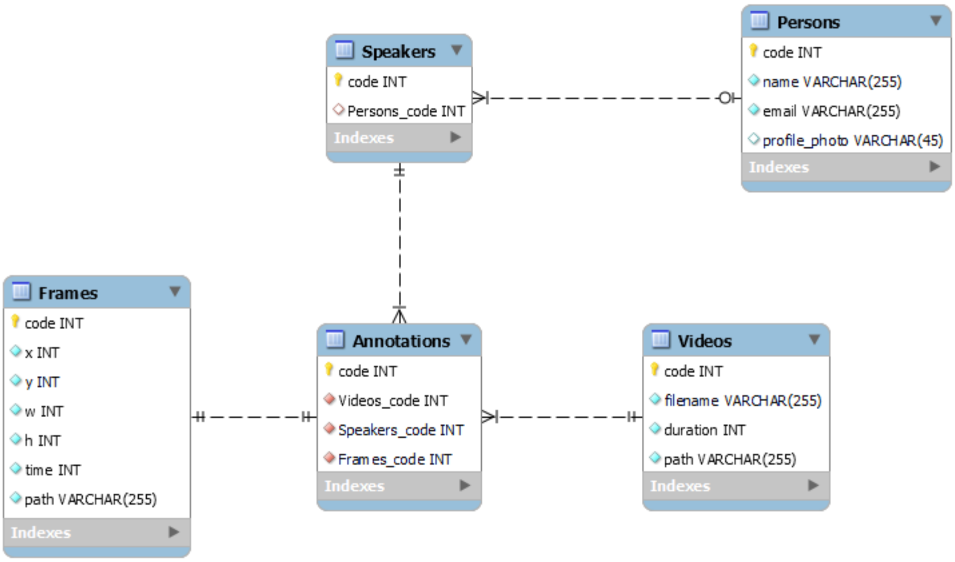


Figura 2 – Entity Relationship Model

6.2 Service Interface

This section presents the service interface of the API module used during the experiment, indicating the HTTP access method, URI and description of the available resource.

For example, using the API services interface, you can query all annotations parsed by a given video via the /api/videos/<video_code>/annotations/ route using the HTTP GET method and passing in the video ID. All available API services are listed in Table **??**.

Tabela 2 – Services available on API

Method	URI	Description
POST	/api/actors/	Create a new actor
GET	/api/actors/<code>/	Query actor by code
POST	/api/annotations/	Generate face annotations of a video
GET	/api/annotations/	Query all annotations
GET	/api/videos/<code>/annotations/	Query all video annotations
DELETE	/api/images/<code>	Deletes an annotation
PUT	/api/images/	Move an annotation to someone else
POST	/api/videos/	Analyze a video
GET	/api/videos/	Browse all videos
POST	/api/persons/	Create a new person
GET	/api/persons/<name>/	Query all people by name
GET	/api/persons/<code>/annotations/	Query all annotations by person
GET	/api/persons/	Query all people
POST	/quiz/users/<user_id>/tests/	Query all tests by user
GET	/api/reports/csv	Export annotations
POST	/api/imports/	Import .pkl file

6.3 Tools and Technologies

To implement the system, a study was initially made of the technologies that best served the purposes of the project and how they relate to the REST architecture chosen to model the application.

6.3.1 Programming Languages

API Module

The language chosen was Python [1], in its version 3.7.0. This choice is based on the fact that Python is a high-level programming language with a small learning curve compared to other languages and easy integration with the other technologies used in the project.

Interface Module

The language chosen was Javascript[2], in its version 5.1.0. JavaScript is a structured, high-level scripting interpreted programming language with weak dynamic typing and multiparadigm. Along with HTML and CSS, JavaScript is one of the top three technologies on the World Wide Web.

6.3.2 Framework

API Module

For project development we use Flask [3], a micro web framework written in Python

[1] https://www.python.org/

[2] https://developer.mozilla.org/pt-BR/docs/Web/JavaScript

[3] https://palletsprojects.com/p/flask/

and based on the WSGI library Werkzeug[4] and in the Jinja2[5] library. It is licensed under the terms of the BSD License[6], which allows for the creation and distribution of the tool freely. Furthermore, Flask is one of the main frameworks in Python. It has a rich library and showed good performance for the application scope, making it our choice. Among the features provided by Flask, the routing system with HTTP methods, parameters and conditions, redirection and stops, customized model rendering, error handling and debugging, in addition to the ease of configuration, stand out.

Interface Module

For project development we use React[7], React is an open source JavaScript library focused on creating user interfaces on web pages. It is maintained by Facebook, Instagram, other companies and a community of individual developers. It is used on Netflix, Imgur, Feedly, Airbnb, SeatGeek, HelloSign, Walmart and others.

6.3.3 Database Management System (DBMS)

To model the data in the system, we use MySQL [8]. MySQL is a database management system, which uses the SQL language as an interface. It is currently one of Oracle Corporation's most popular database management systems, with more than 10 million installations worldwide.

[4] https://werkzeug.palletsprojects.com

[5] https://jinja.palletsprojects.com

[6] https://opensource.org/licenses/bsd-license.php

[7] https://en-us.reactjs.org/

[8] https://www.mysql.com/

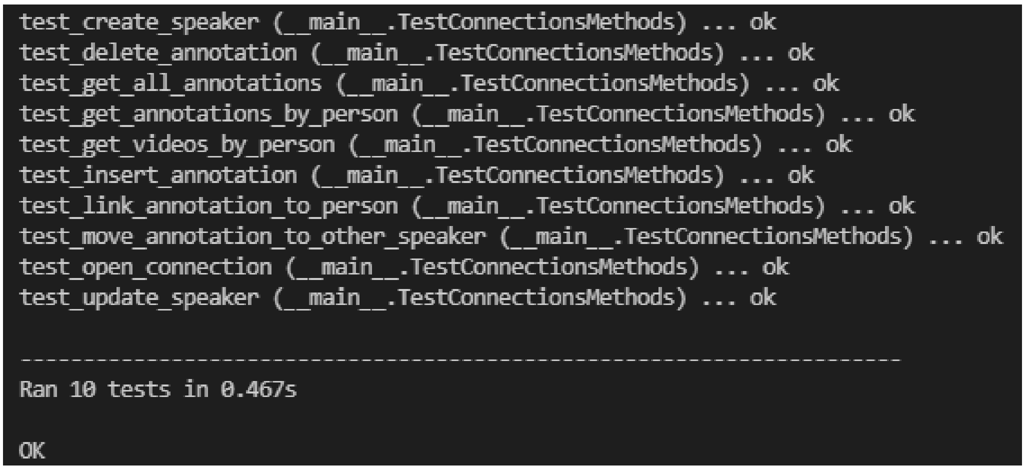
7 Tests

To validate the correct execution of our application, we perform unit tests on the functions in the API module.

Unit tests seek to verify the accuracy of the code, to its smallest fraction. In object-oriented languages, this smallest piece of code can be a method of a class. Thus, unit tests are applied to these methods, from the creation of test classes.

Using the package unittest [1], we analyzed 10 unit tests for the API module: Opening the connection with the database; Create an actor; Update information about an actor; Insert a new face annotation; Removal of a specific face annotation; Relate a face to a person; Consult notes per person; Move a wrong note to someone else; Consult all notes; Browse videos per person and Browse all videos;

In Figure 3 we can see the tests and their results.



```
test_create_speaker (__main__.TestConnectionsMethods) ... ok
test_delete_annotation (__main__.TestConnectionsMethods) ... ok
test_get_all_annotations (__main__.TestConnectionsMethods) ... ok
test_get_annotations_by_person (__main__.TestConnectionsMethods) ... ok
test_get_videos_by_person (__main__.TestConnectionsMethods) ... ok
test_insert_annotation (__main__.TestConnectionsMethods) ... ok
test_link_annotation_to_person (__main__.TestConnectionsMethods) ... ok
test_move_annotation_to_other_speaker (__main__.TestConnectionsMethods) ... ok
test_open_connection (__main__.TestConnectionsMethods) ... ok
test_update_speaker (__main__.TestConnectionsMethods) ... ok


----------------------------------------------------------------------
Ran 10 tests in 0.467s

OK
```

Figura 3 – Unit tests performed in the experiments

[1] https://docs.python.org/3/library/unittest.html

8 Documentation

API Module

Initially, the user must install Python in the latest version, after which he must download the project's source code, contained in a repository on GitHub[1]. The command to perform this operation is:

```
$ git clone https://github.com/vieiraeduardos/datasetmanager-api.git
```

After downloading the repository, you should enter the project folder:

```
cd ./datasetmanager-api/
```

Now we recommend that you make use of a virtual environment to install the necessary packages. After creating a virtual environment, install the packages using the following command:

```
$ pip install -r requirements.txt
```

This process may take a while. When finished, run the command to start the API:

```
$ python main.py
```

Interface Module

Initially, the user must install Node.Js , in the latest version, after that, he must download the project's source code, contained in a repository on GitHub[2]. The command to perform this operation is:

```
$ git clone https://github.com/vieiraeduardos/video-annotation-tool.git
```

After downloading the repository, you should enter the project folder:

```
cd ./video-annotation-tool/
```

Finally, install the necessary packages:

[1] https://github.com/vieiraeduardos/datasetmanager -api
[2] https://github.com/vieiraeduardos /video-annotation-tool

```
$ npm install
```

This process may take a while. When finished, execute the command to start the application:

```
$ npm start
```

8.1 Screenshots

In Figure 5 we can see the initial screen of the web application through which we can submit a video for analysis.

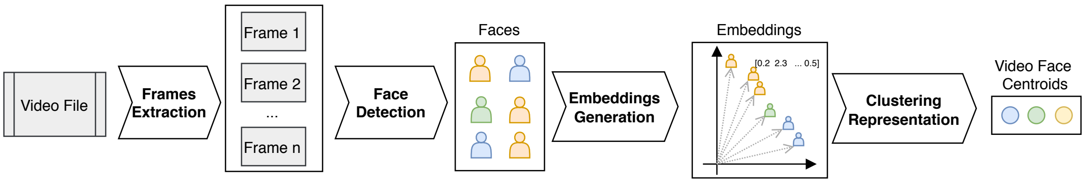
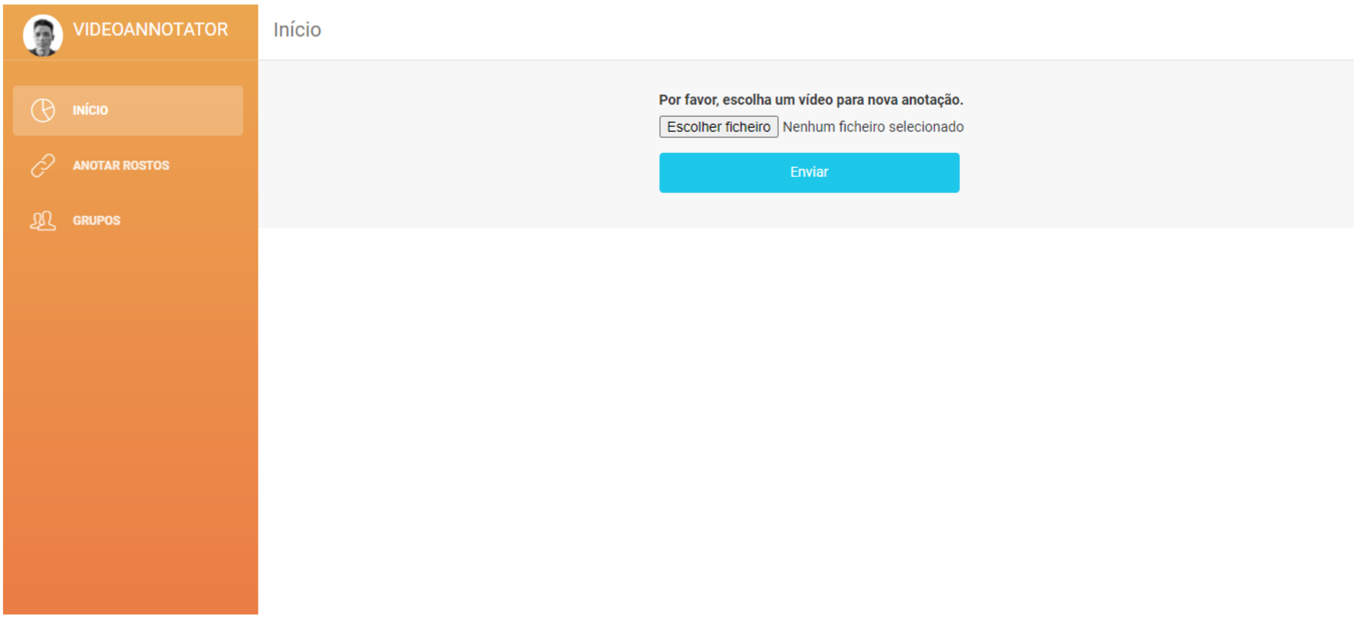


Figura 4 – Video representation process with lecturers face centroids.

Once submitted, our algorithm identifies the faces of all the people present in the video and groups them together. All the process is shown in Figure 4 and described in paper (MENDES et al., 2020), where I am one of the authors.



Por favor, escolha um vídeo para nova anotação.

Nenhum ficheiro selecionado

Figura 5 – Home Page: Submitting videos

During this process, our application shows an animation indicating that the process is in progress, as we can see in Figure 6.

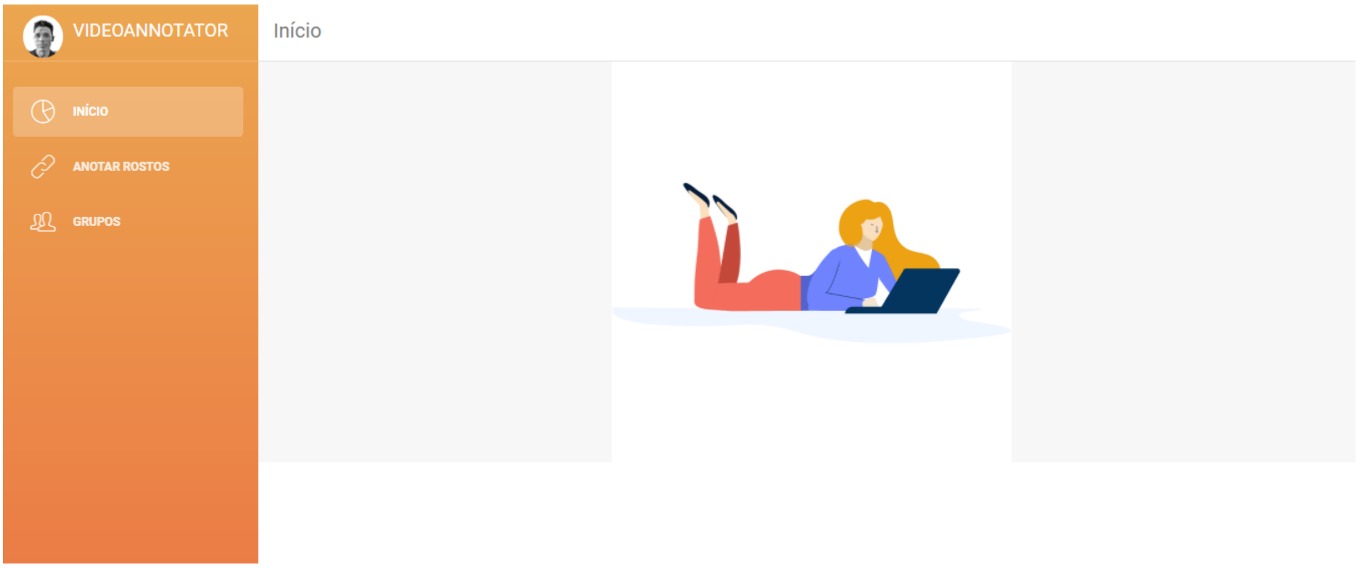
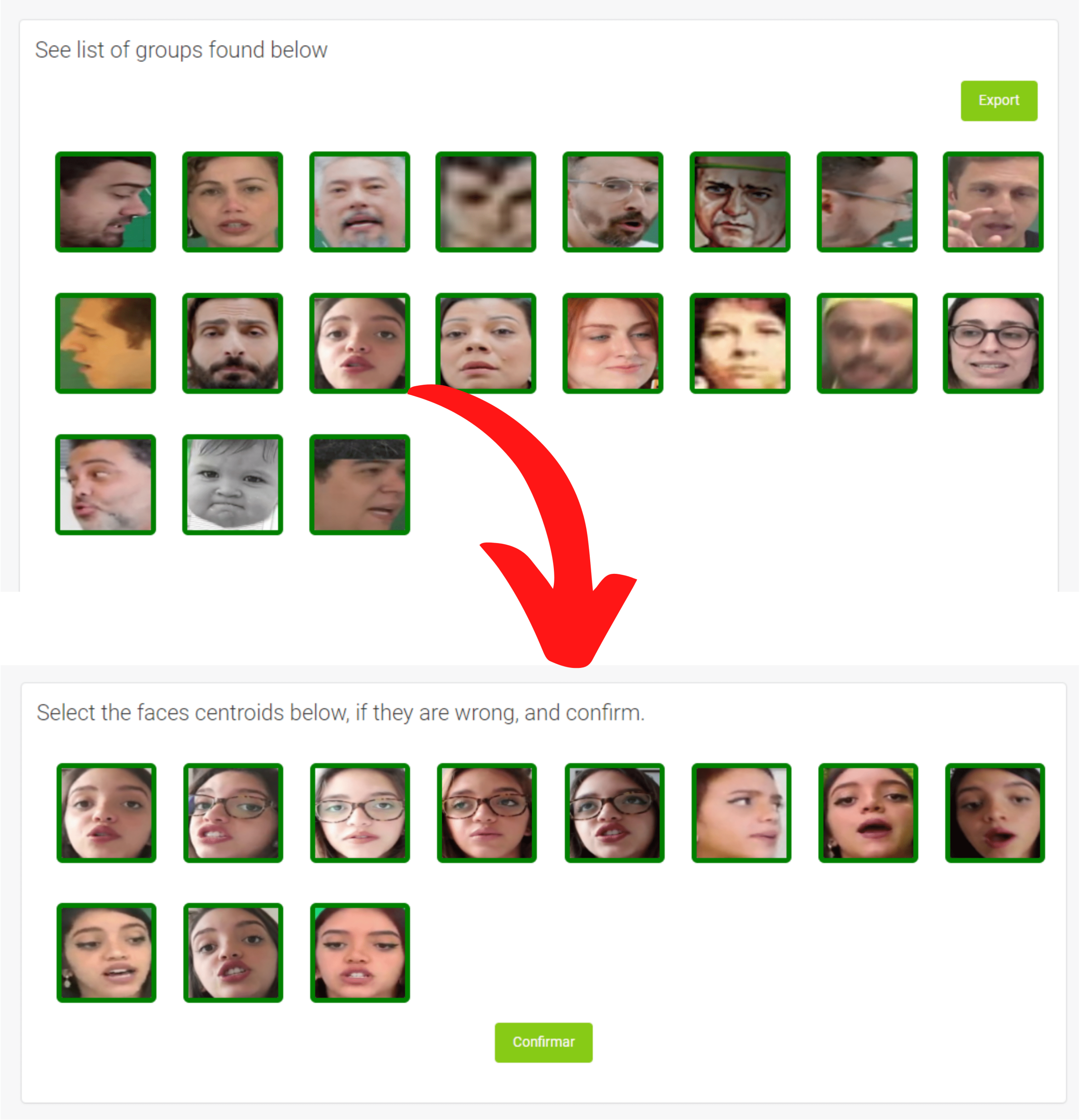


Figura 6 – Processing Page: Waiting while video is processing

After the analysis, we can visualize all the extracted and grouped faces in Figure 7. In this way, we can export this meta information to create a dataset, for example. And view in the tool all the links between videos where the speaker appears.



See list of groups found below

Select the faces centroids below, if they are wrong, and confirm.

Figura 7 – Groups Page: Visualizing groups by face

Referências

MENDES, P. R. C.; VIEIRA, E. S.; GUEDES, L. V.; BUSSON, A. J. G.; COLCHER, S. A clustering-based method for automatic educational video recommendation using deep face-features of lecturers. In: *2020 IEEE International Symposium on Multimedia (ISM)*. [S.l.: s.n.], 2020. p. 158–161. Citado na página 17.