Technical Project Report - iOS Module

Movie Emotions

Course:	44139- Computação Móvel
Date:	Aveiro, December 22th, 2017
Authors:	71883: João Melo 73046: Tiago Henriques
Project abstract:	Movie Emotions is a mobile application developed in swift language for the iOS System. This app main objective is to trace the psychophysiological profile of sessions through a collection of emotions/feelings felt by the users. This is acquired by a facial recognition mechanism while the user watches several movie trailers. This is used to create personalized movie suggestions taking into account people's personalities preferences by a certain movie type/genre.

Table of contents:

1 Introduction

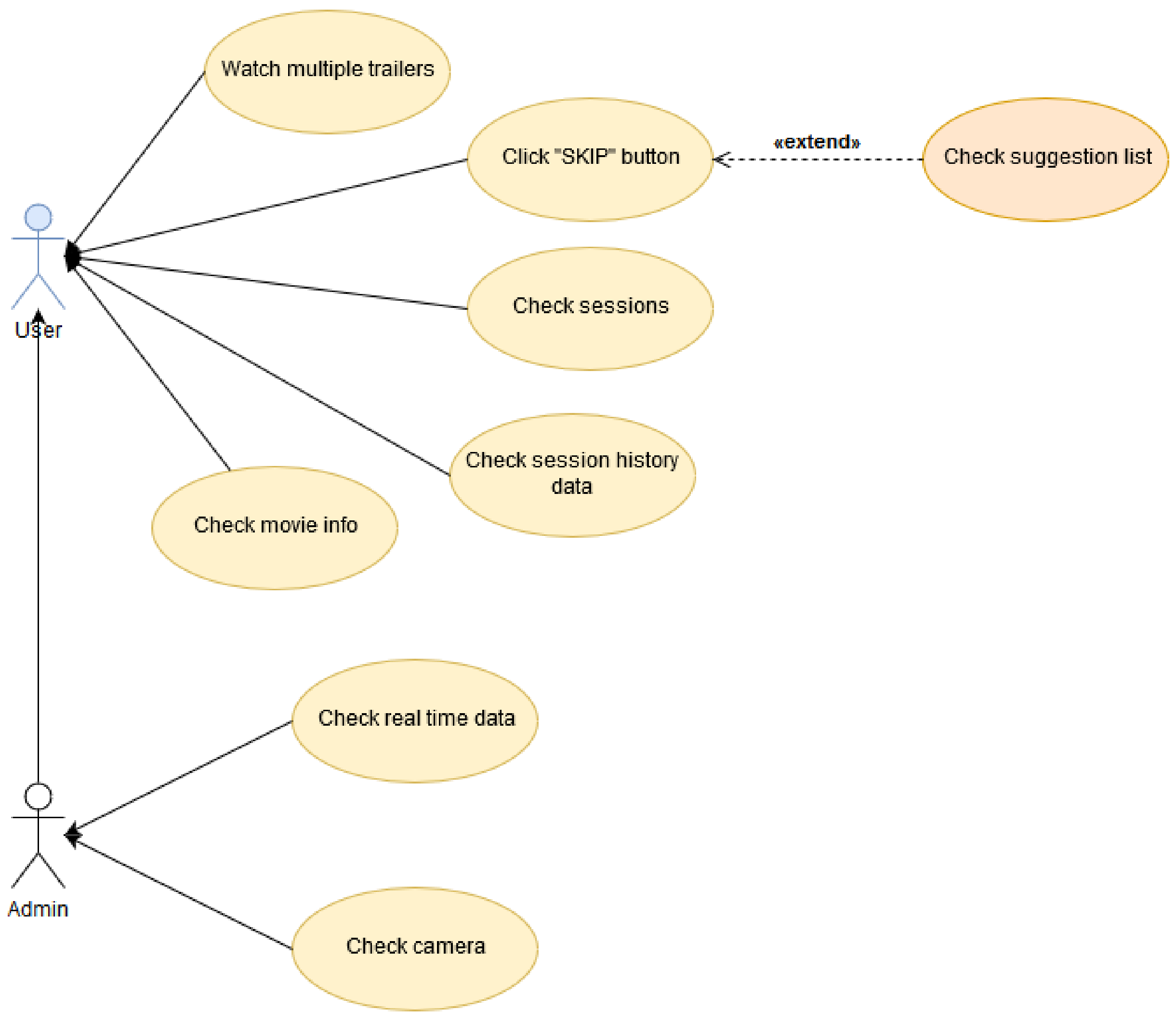
This project is related to the iOS Module of the CM course and its main objective is to improve our skills on the iOS programming. This project was developed on swift 4.0 and makes use of the external library *AffdexSDK-iOS*, for emotional facial tracking via Affectiva sdk and the library *Charts*, for real-time and historical emotional data visual displaying.

This project is based on the android project developed in the first half of the semester, therefore, it aims to apply the same concepts and get the same results of the Android project, having been slightly adapted to allow different modes of operation and allow the possibility of visualizing real-time emotional data that is being acquired by the facial tracking mechanism. The app follows the same patterns of the Android app, it recommends movies, depending of the users emotion. To allow this feature, we show several movie trailers to the end user, random trailers each time, and from different genres/types. While watching the trailer, several emotions of the user are being detected by a facial recognition system with the support of the front camera. We are detecting emotions like Joy, Fear, Attention, etc. When the user decides to stop watching trailers, based on the values collected, our app is going to suggest a list of movies. The user can then check movie informations like title, synopsis, release date, rating average... Also, the user can check the psycho emotional profile of a specific session.

The scope of this app changes a little bit from his android counterpart, because it derivates to a more professional/user type of app where the sessions can be verified on the end of them and there is a much more wide focus on the emotions that on the movie suggestion.

2 Application concept

In the next diagram it's possible to check several of the most important use cases that a user can perform in our app.



The application constantly monitors users' emotions/feelings using Affectiva facial recognition SDK, a company co-founded by Rana el Kaliouby and Rosalind Picard who specialize in software to handle and sense human emotions.

The users of the app are the smartphone users themselves, being users divided into regular users or admin users. The only requirement asked to the user is that his face is aligned accurately with the front camera, so that it is possible to use the front camera for detecting emotions/feelings while the user is watching a random movie trailer. The application has two modes of operation: the first is designed for the regular user allowing the user to watch several movie trailers and in the end check suggested movies and related movie information. The second is designed for the admin user which means that all functionalities will be available, meaning the user will have access to the facial tracking process being able to see his face on the screen while he watches movie trailers. He is also capable of accessing and visualizing the real-time data that is going to be displayed on the the real-time track graph. This graph is showing real-time emotional data, more specifically, the values of joy, surprise and attention being captured by the users face. The main purpose of this application is to trace psychophysiological session profiles through the collection of user's emotions/feelings.

3 User experience design process

One of the things that it was decided to do was to show the minimal amount of information of the recognition pattern to the end user, because this would be boring and meaningless to him. The most curious ones can see the emotion values on a file stored on the system's internal storage (it will be explained below) and the session history graph. The system was designed to be very simple and intuitive to use. On the main screen, if the user clicks on the *Start* button, he can start watching movie trailers (Image 1). When he has watched a movie trailer or if he wants to check the movies suggestions list, he then can click the *Skip* button (Image 3 and 4). A list of suggested movies (Image 5) will appear and by clicking on a movie on the list, the user gets the relevant information for that movie (Image 6).

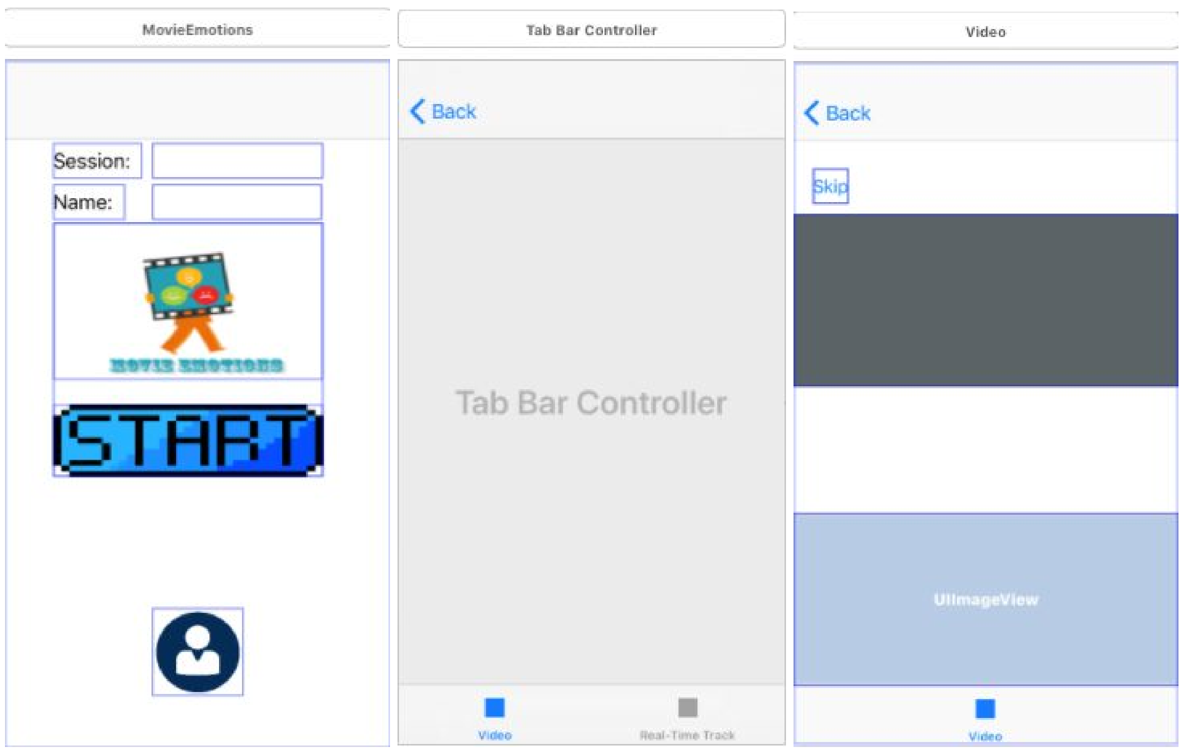
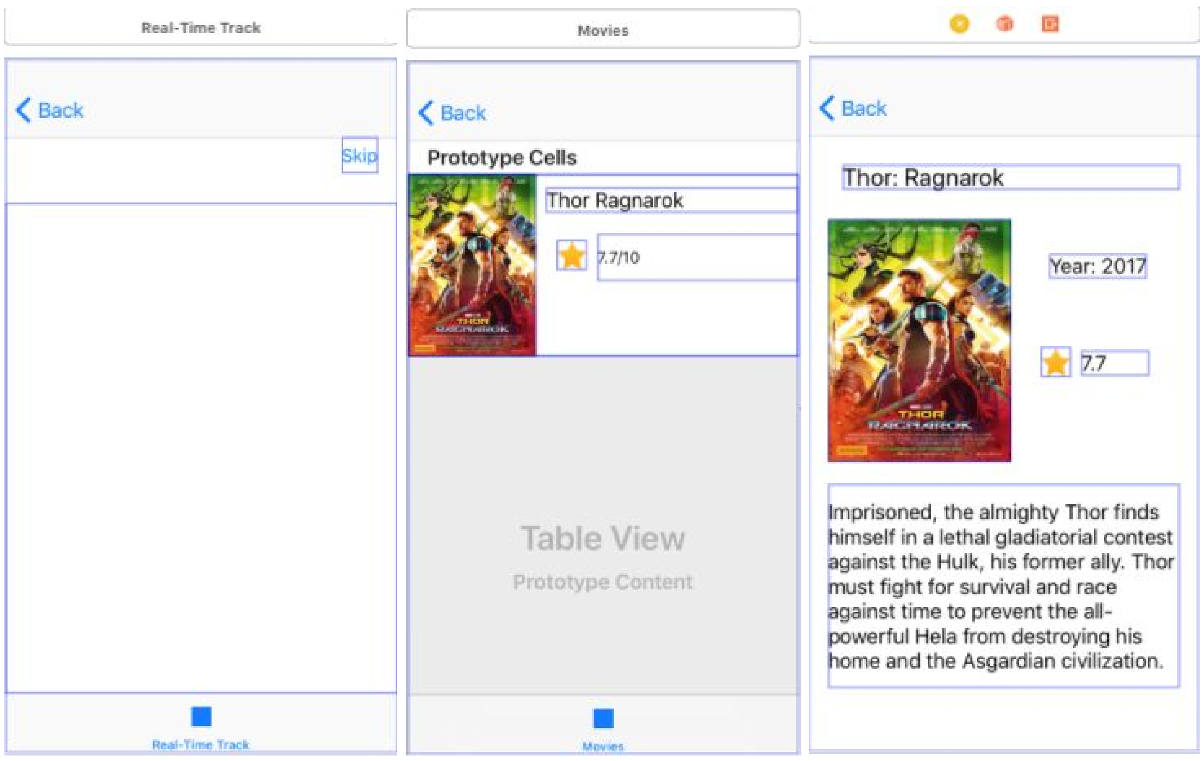


Image 1 - 3 - From left to right - Main menu, Controller, trailer watching



Imprisoned, the almighty Thor finds himself in a lethal gladiatorial contest against the Hulk, his former ally. Thor must fight for survival and race against time to prevent the all-powerful Hela from destroying his home and the Asgardian civilization.

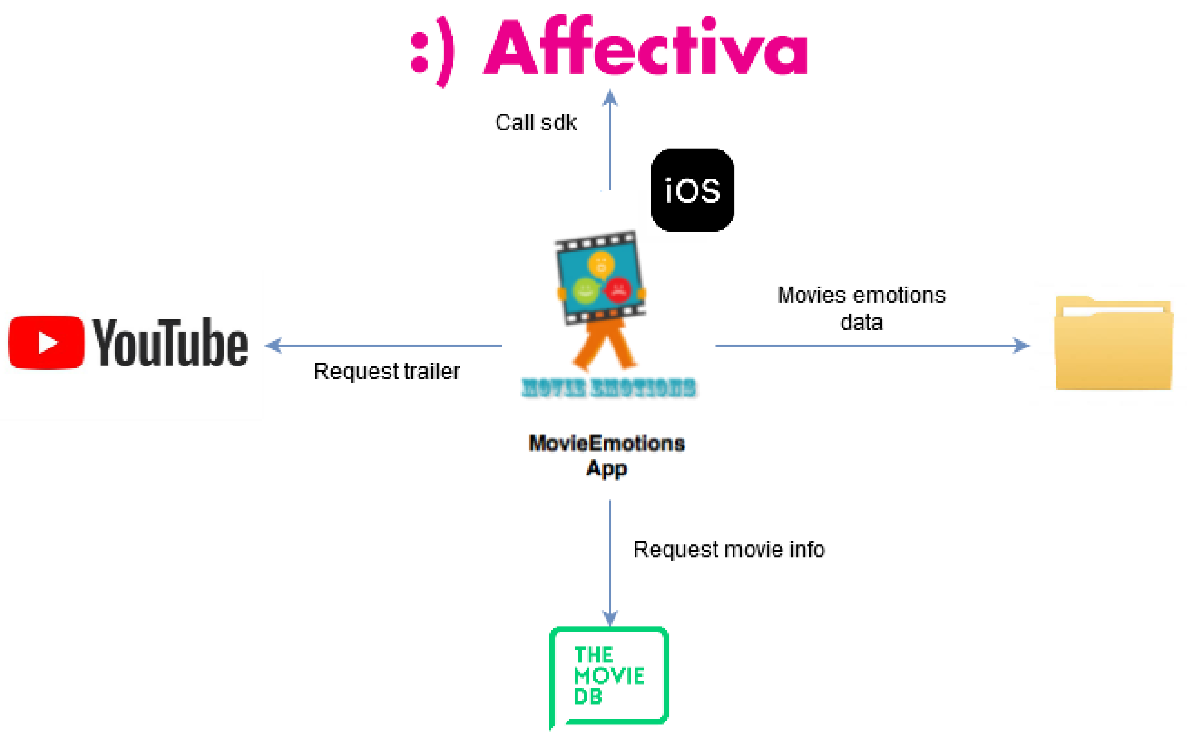
Image 4 - 6 - From left to right - Real time graph, Movie suggestion table, Movie info

On the previous images, are shown the first view controllers prototypes of the application to be developed. Our final product is slightly different but the main features remained untouched. On the images we can see the main screens that the user will get on his experience. The user will start on the first image that is the home page, where the users types his name and the session name. From here, he can start the experience by tapping the *Start* button or check all the sessions profiles after tapping the bottom screen button. After clicking the *Start* button, the user will get the second image. This view controller is a tab bar view controller which means that the app is divided into two or more distinct modes of interaction. The third image is the view controller that enables users to see some random movie trailers uploaded from *Youtube* and the fourth image is the view controller that enables users to visualize the emotional data visually on a real-time graph.

If he wants to check the movies suggestion list, he should click on the “skip” button. After clicking the button, it will change to the fifth image, that shows a customized list of movie suggestions.

If the user wants to check a movie relevant information, he needs to click that cell of the table view, and it will change to the sixth image that presents information like the movie title, the movie poster, the movie release date, the movie rating and the movie synopsis.

4 Architectural plan for the solution



The image immediately above represents the global architecture of our application. The mobile application developed in *iOS*, which will be installed on the user's mobile devices, saves movies' emotional information on the internal storage in a text file. It also accesses the *affectiva* and movie db api. The first one is used to get the emotion values from the facial tracking mechanism and the second to get the suggestion list and movie information. Finally, it uses the *Youtube* to display different movie trailers.

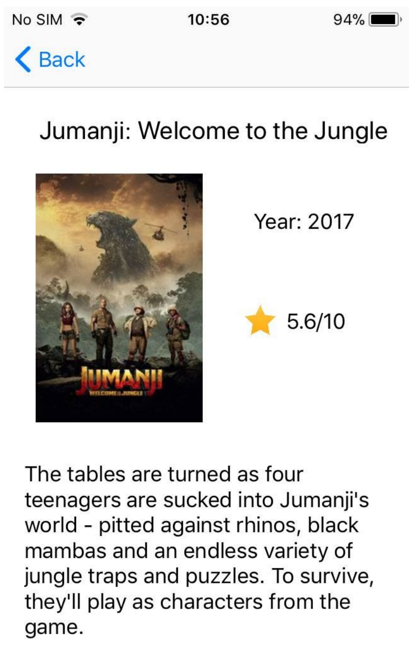
Persistence:

One of the most important features that any app should have is persistence, meaning it can store data in a more durable state so it can outlive an app relaunched or a device reboot. The data related to movies' emotions/feelings is being stored in a local storage file, in a .txt file, localized on the documents directory. The values of joy, surprise and attention are being stored on the file to later analysis and creation of an session profile.

Data Models:

For data models, we have 5 different data types. The *Data* data type is used to create and manage global variables changes. The *Movie* data type is used to create a movie to later being added to a movies array. The *MovieInfo* data type is used to represent a movie when we get it from the API. It is stored the title, poster, overview, release date and vote average. The other one is *MoviesSuggestion* where we have the list of trailers the user will watch and the trailers that we had watched in this session. The *Persistence* data type is used to write emotional data (values of joy, surprise and attention) to a local file.

For the content update, it is used the Movie Database API (https://www.themoviedb.org/), and we are getting the data in real time every time we create a suggestion list based on the emotions detected.



Jumanji: Welcome to the Jungle

Year: 2017

5.6/10

The tables are turned as four teenagers are sucked into Jumanji's world - pitted against rhinos, black mambas and an endless variety of jungle traps and puzzles. To survive, they'll play as characters from the game.

5 Implemented solution in iOS

In the iOS application that was developed were used several technologies/strategies acknowledged in the practical classes of the curricular unit. Some of them will be mentioned below:

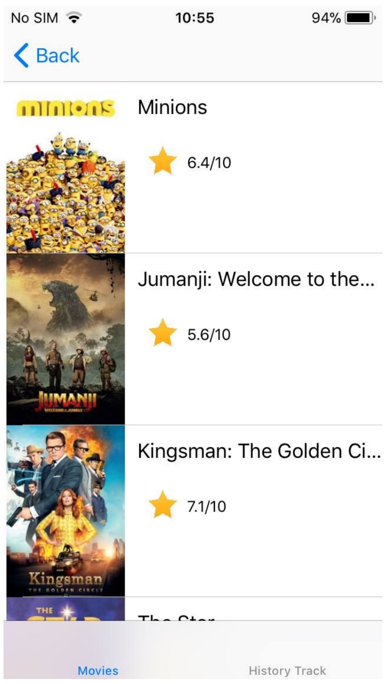
View Controllers:

Mobile iOS programming differs a lot from Android programming, and one of the most significant differences is in the use of View Controllers. In this case, there is no need to choose between an activity or fragment. Each one of our views present in the Main.storyboard is a View Controller, where view controllers serve as the communication pipeline between our views and our data models, known as *MVC* (Model-View-Controller) design pattern.

- **Table View Controllers:**

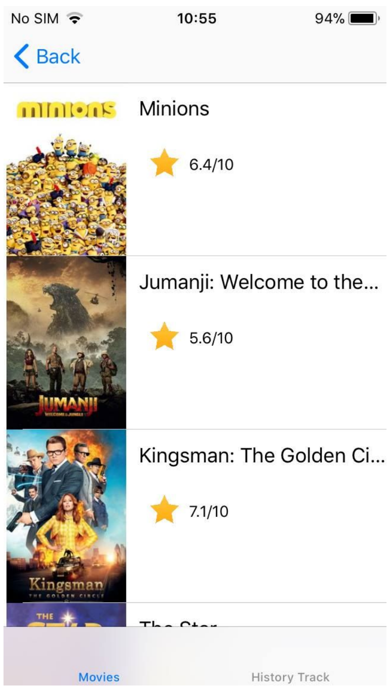
iOS doesn't offer the benefits of Android's ListViews and Adapters, fortunately, iOS comes with a built-in class, UITableView, designed specifically to display a scrolling list of items with a very simple and intuitive implementation. A table view is managed by a table view controller which is designed to handle table view-related logic. A table view controller, *MovieTableViewController*, was implemented in order to display the suggested movies displaying process.

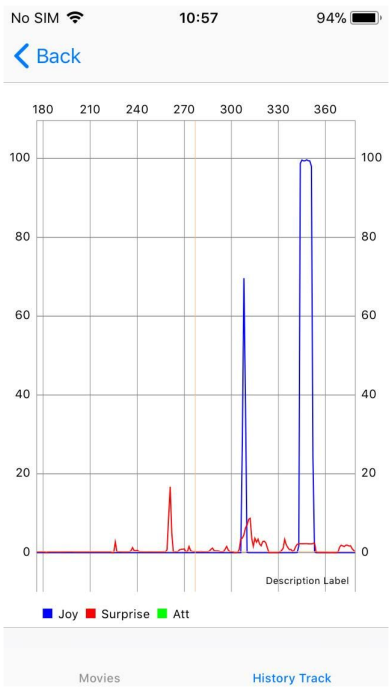
In iOS, it is so much easy to layout a single row of a table view. These individual rows are managed by table view cells, which are responsible for drawing their contents. Table view cells come with a variety of predefined behavior and default cell styles; however, in our case, we chose to define a custom cell style, *MovieTableViewCell*, in order to present the suggested movie information such as the movie poster, the movie title, the movie rating, the movie release date and the movie synopsis.



- **Tab Bar View Controllers:**

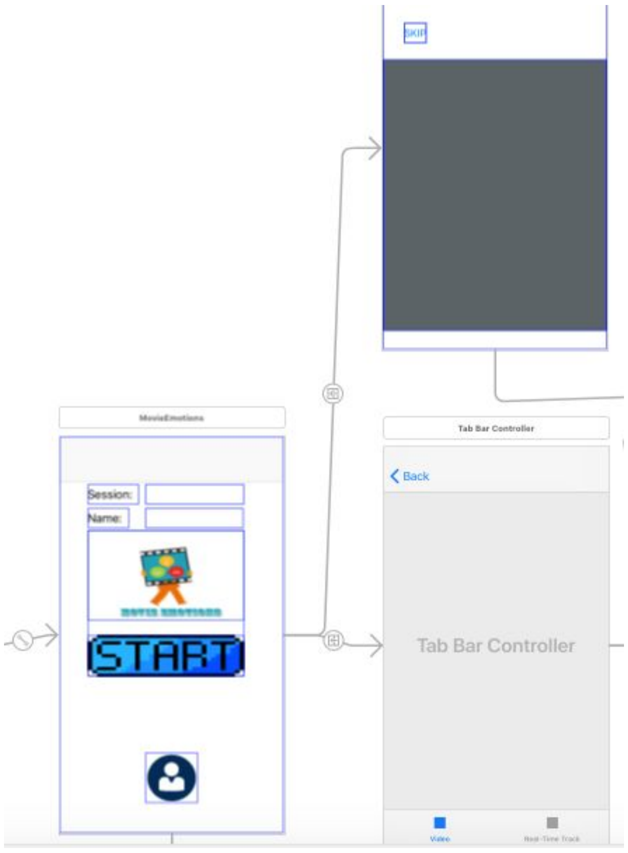
Tab bars are used to divide an app into two or more distinct modes of operation. In our case, in the admin user mode of operation, we use tab bars to alternate between the videos streaming while capturing emotions and the real-time graph displaying of emotional values. In the regular user mode of operation, we use tab bars to alternate between suggested movies and history track graph.





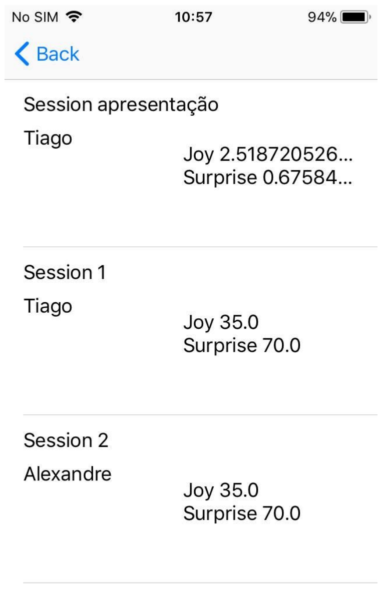
Segues:

Segues are used to define the flow of your app’s interface. A *segue* defines a transition between two view controllers in your app’s storyboard file. The starting point of a segue is the button, table row, or gesture recognizer that initiates the segue. The endpoint of a segue is the view controller you want to display. A segue always presents a new view controller, but you can also use an *unwind segue* to dismiss a view controller. We use segues to move between screens and to choose which screens and app flow type should be displayed depending on the mode of operation (regular user or admin user).



No Internet connection:

If the user does not have Internet access, the user will not be able to access all the features offered by the application, because much of the data to be obtained depends on the visualization of trailers of videos uploaded from Youtube. However, the user will be able to view the information stored in internal storage such as the emotional movie data recorded when trailers were seen when with Internet access.


No SIM 10:57 94%

Back

Session apresentação
Tiago
Joy 2.518720526...
Surprise 0.67584...

Session 1
Tiago
Joy 35.0
Surprise 70.0

Session 2
Alexandre
Joy 35.0
Surprise 70.0

On the picture above, even if the user has no internet connection, he still is able to access data stored on the device's' internal storage such as the mean emotional values for each recorded session.

6 Non-Implemented solution in iOS

From the beginning we had several problems with the Xcode platform. In the first place we had to update it to version **9.1** so the youtube api could work but with that update, various problems arrived like dependencies and version related errors.

Several times we were working in a different code and another one started giving errors on the imports, after several tries we were able to fix this.

We had several problems introducing *multipeer connectivity*. It was created a separated project to be the master and in that one everything worked just fine, but in our project we had several errors with the same exact file that we used on the other project.

The main error was - **Type '...' does not conform to protocol '...' Xcode 9.1**

We tried reading the documentation and use the Xcode fix tool to solve the issue, but nothing worked so we couldn't put it working, and chose to implement a two modes of operations app instead.

7 Conclusion

The main project goals were fulfilled because with the application developed it is possible to trace psychophysiological movie profiles through the collection of information coming from the detection of emotions/feelings by facial recognition.

The main problems, also encountered in Android, were that the youtube player doesn't let overlay components to its fragment, so we couldn't set visual elements on top to give users some kind of feedback that allows them to check if the facial tracking is working or not, as we implemented in the Android app. We would like to change the design, put it more user friendly and more material design. Another thing that we would like to improve is the suggested movie algorithm and use a neural network to this end, this way the suggestions would be even more personalized.

Making an analogy between Android and iOS programming, we came to mind that while both platforms (Android Studio and Xcode) are quite usable and that there is a programming facility in both, the iOS platform allows a much faster and more intuitive development when dealing with the graphical component of the application than on the Android platform. In addition, because iOS programming is divided into view controllers, it is much simpler to make views independent from each other and as well as pass information between them using segues.

8 References and resources

https://developer.apple.com/library/content/referencelibrary/GettingStarted/DevelopiOSAppsSwift/

https://developer.affectiva.com/

https://www.themoviedb.org/

https://www.ralfebert.de/tutorials/ios-swift-multipeer-connectivity/

https://www.raywenderlich.com/category/swift

Project resources:

Project resources for the iOS module:

- Code repository: https://github.com/henriquestiagoo/ios-module-movie-emotions