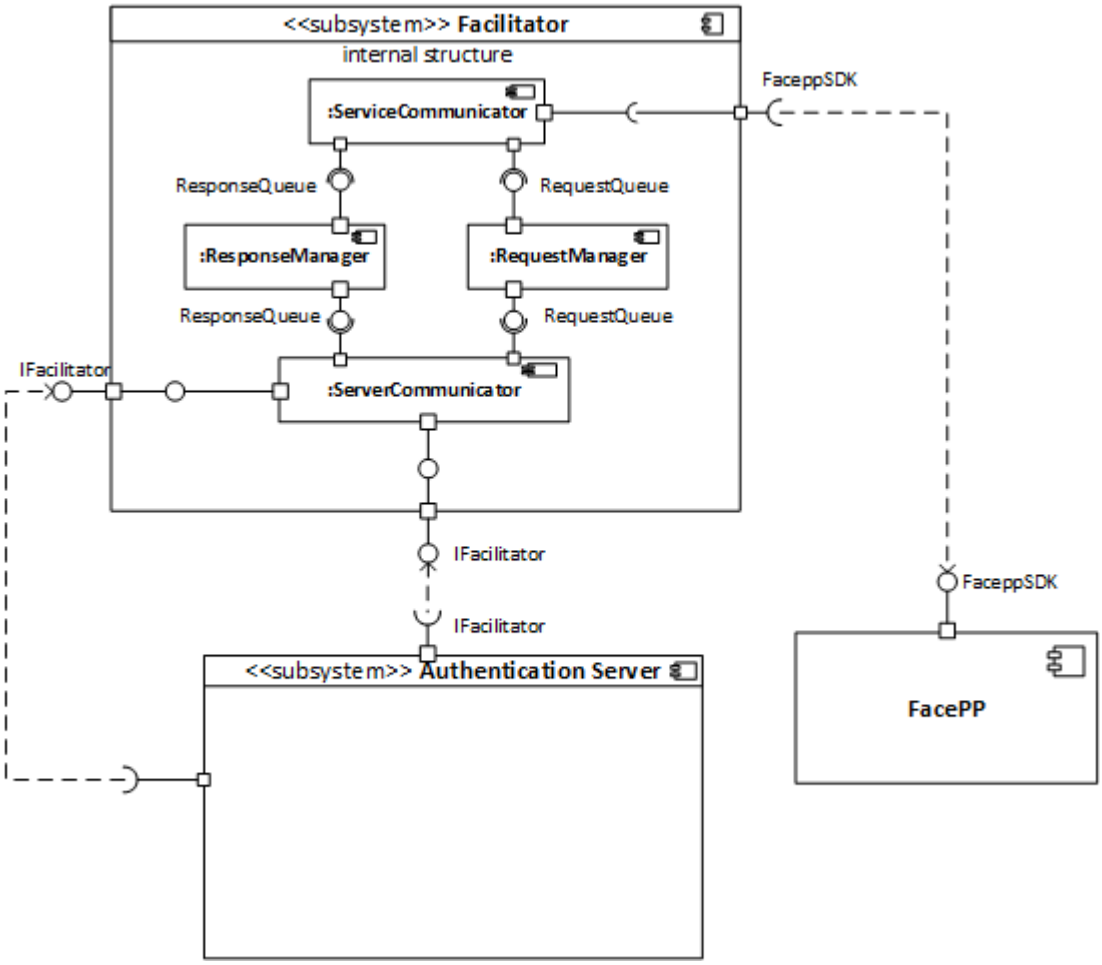
# Unified Document – Facilitator

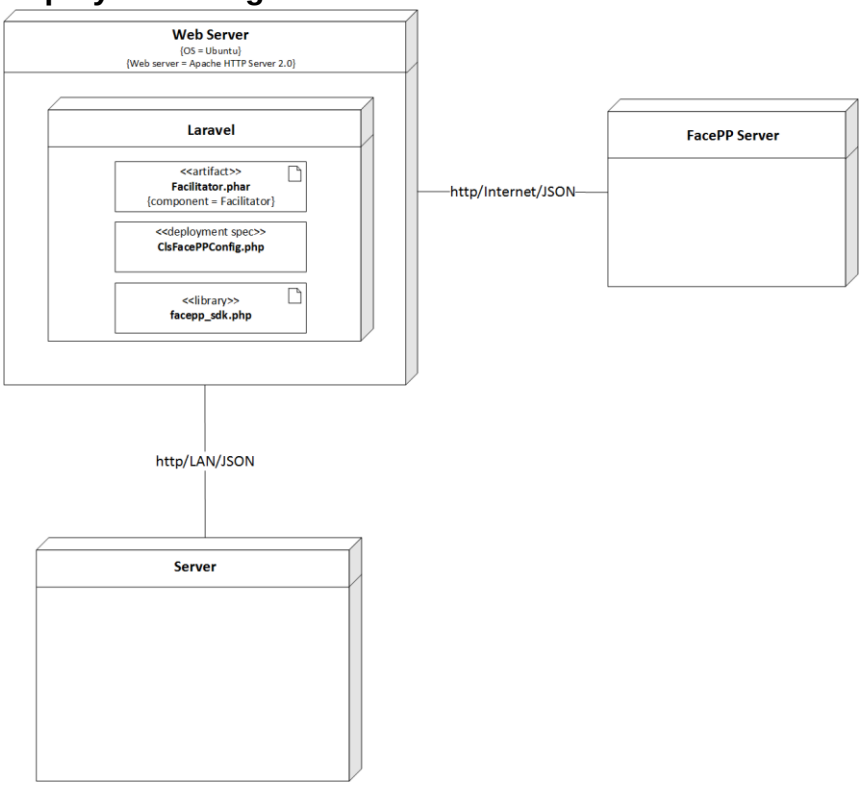
## 1. Software Design

### 1.1 Software Architectures

#### 1.1.1. Component Diagram



### 1.1.2. Deployment Diagram.




## 1.2. Module interface specifications

### I. Introduction

The Facilitator allows to perform the communication between the server and the recognition server, so that everybody users can be register and the server can be trained with the user's information and the photos that the server recognition required. Also the Facilitator lets you know if any user is registered in the system and validate this information.

### II. **Interface Overview**

#### a. Services Provided

| **Service** | **Provided by** | **Tested By** |
|---|---|---|
| 1. Facilitator communicate with the authentication server | IFacilitator | |
| 2. Manage responses to requests from training or authentication server | ResponseQueue | |

| | | |
|---|---|---|
| 3. Manage training or authentication requests from the server | RequestQueue | |
| 4. Communicate the facilitator Face Plus Plus | faceppSDK | |

**b. Access Methods**

| **Access Method** | **Parameter name** | **Parameter type** | **Description** | **Exceptions** | **Map to services** |
|---|---|---|---|---|---|
| IFacilitator | 1. Identifier: IN<br><br>2. photo: IN | 1. String<br><br>2. Url o img | 1.Identifier is the the string that represent to the user in the system.<br><br>2. Photo is the picture for train the server. | **1**<br>**2** | **1** |
| ResponseQueue | 1. ExternalId: IN<br><br>2. photo: IN | 1. String<br><br>2. Url o img | 1.ExternalId is the string that return the recognition server next to send the request.<br><br>2. Photo is the picture for train the server. | 2 | **2** |
| RequestQueue | 1. Identifier: IN<br><br>2. photo: IN | 1. String<br><br>2. Url o img | 1.Identifier is the the string that represent to the user in the system.<br><br>2. Photo is the picture for train the server. | | **3** |
| faceppSDK | 1. Identifier: IN<br><br>2. photo: IN | 1. String<br><br>2. Url o img | 1.Identifier is the the string that represent to the user in the system.<br><br>2. Photo is the | | |

| | | | | | |
|---|---|---|---|---|---|
| | | | picture for train the server. | | |

**c. Access Method Effects**

| Access Method | Description |
|---|---|
| IFacilitator | It takes the information received from the server as parameters, and send it inside the facilitator for keep it in the server communicator. |
| ResponseQueue | accumulates the request in the queue for later be send to recognition server |
| RequestQueue | It receives the answer of the recognition server by know the state of the profile |
| faceppSDK | It takes the information received of the facilitator and use its own languages by send the information to the recognition server. |

**III.Local Types**

| Type | Value Space |
|---|---|
| ExternalId | Is an string that identifies the user in the server |

**IV. Terms**

**V. Uses**

| Type | Value Space |
|---|---|
| | |
| | |

**VI. Exception Dictionary**

**a. TC1: authenticating user**

| Step | Description | Input Type/Value | Expected Result | Service | Preamble |
|---|---|---|---|---|---|
| 1 | receive authentication | picture email | the user authentication | 1 | 1 |

| | | | | | |
|---|---|---|---|---|---|
| | request | | status from the facilitator | | |
| 2 | receive authentication result | | the information of the confirmation is sended to the client | 1 | 1 |

**b. TC2: authenticating user and email doesn't exist**

| Step | Description | Input Type/Value | Expected Result | Service | Preamble |
|---|---|---|---|---|---|
| 1 | receive authentication request | picture email | the identifier doesnt exist | 1 | 1 |
| 2 | sending the error to the client | error message | the error is successfully sent to the client | 1 | 1 |

**c. TC3: authenticating user and the picture is invalid**

| Step | Description | Input Type/Value | Expected Result | Service | Preamble |
|---|---|---|---|---|---|
| 1 | receive authentication request | picture and email | the facilitator returns a message telling that the picture is invalid | 1 | 1 |
| 2 | sending the error to the client | error message | the error is successfully sent to the client | 1 | 1 |

**d. TC4: registering user**

| Step | Description | Input Type/Value | Expected Result | Service | Preamble |
|---|---|---|---|---|---|
| 1 | receive registering request | pictures<br>email<br>gender<br>name | the identifier doesnt exist in the database and the facilitator returns a message validating the training process with the | 2 | |
| 2 | storing the information into the database | pictures<br>email<br>gender<br>name<br>external id | the information is successfully stored | 2 | 1 |
| 3 | sending a validation message to the | string | the message is successfully sent | 2 | 2 |

**e. TC5: registering user and the email already exist**

| Step | Description | Input Type/Value | Expected Result | Service | Preamble |
|---|---|---|---|---|---|
| 1 | receive registering request | pictures email<br>gender<br>name | the email already exist in the database | 2 | |
| 3 | sending the error message | string | the error message is successfully | 2 | 1 |

| | | | | | |
|---|---|---|---|---|---|
| | | | sent | | |

**F. TC5: registering user and the some pictures are invalid**

| Step | Description | Input Type/Value | Expected Result | Service | Preamble |
|---|---|---|---|---|---|
| 1 | receive registering request | pictures email gender name | the identifier doesnt exist in the database and the facilitator returns a message with the pictures that weren't accepted | 2 | |
| 2 | the array with the pictures that weren't accepted are sent to the client | array of pictures | the array is successfully sent | 2 | 1 |
| 3 | sending the error message | string | the error message is successfully sent with the array | 2 | 2 |

## VII. Test Cases

To be determined.

## VIII. Design Issues

## IX. Review Questions

***Requirement Validity***

1. For each service provided by the module, is the service valid for all expected uses of this module? If not, give an example of a use where the service is not valid.

2. For each service provided by the module, is the service valid for all expected configurations and versions of this module? If not, give an example of a needed configuration or version where the service is not valid.

3. For each service needed described in this specification, is a module (or set of modules) identified that this module is allowed to use to satisfy the need?

4. Are there cases where the interface specification could not be satisfied or was incomplete?
If so, how should it be changed?

***Requirements Sufficiency***

1. Does the set of services provided specify all of the services that will be needed by users of this module? Are there any services defined that are not identified in the requirements?

2. Does the set of services needed specify all of the services that this module will need from other modules in order to operate correctly? What services are needed that are not identified in the requirements?

***Consistency Between Services Provided and Access Programs***

1. For each Services Provided described in this specification, which access program(s) can be used to satisfy the service?

2. For each access program specified in sections 1.2.2 which Service Provided is satisfied by the access programs?

***Access Program Adequacy***

1. Is the set of access programs sufficient to satisfy the needs of modules that are allowed to use this module?

2. Are there access programs that should be combined into one access program?

3. Are there single access programs that should be refactored into several different access programs?

4. Are the performance requirements adequate for the uses that will be made of this module?

## 2. Handoff

**(Deliverable 2)**

**HANDOFF 1**

1. What has been done, during the last period
   In the last period, we do any activities like:

- Identify the problems and the new specific objectives.
- Recording answers.
- Consistency check.
- Identified all rules of business.
- Make the class diagram.
- Make the component diagram.

2. How to work should be continued

   - We should make next to research how to make the other diagrams.
   - Take more important information for make architecture diagrams.
   - Begin the other diagrams.

3. There is any obstacle blocking the team

- Communication between the teams
- Agree on a work schedule where everybody could attend the activities.
- Coordinate the activities the right form.
- time investment in other activities.

4. What until test have been covered during this shift
- Identify the problems and the new specific objectives.
- Recording answers.
- Consistency check.
- Identified all rules of business.
- Make the class diagram.
- Make the component diagram.

**(Deliverable 2)**

**HANDOFF 2**

1. What has been done, during the last period
   In the last period, we do any activities like:

- Identify the problems and the new specific objectives.
- Recording answers.

- Consistency check.
- Identified all rules of business.
- Make the class diagram.
- Make the component diagram.
- Make the deployment diagram.
- Make the handoff.
- Make the case uses and the explanatory table.
- Propose an architecture to model the solution.
- Make the Kanban
- Make the alpha states advance.
- Make negotiation with the stakeholders the main priorities

2. How to work should be continued

   The started tasks must be finished, especially the work products like model interface specification , and continue to make the others work products like the user cases and it tables.

   Is very important to validate the models with the teacher, and get the respective feedback with its corrections.

   review the all diagrams for begin the codification from it.

3. There is any obstacle blocking the team

   - Communication between the teams
   - Agree on a work schedule where everybody could attend the activities.
   - Coordinate the activities the right form.
   - time investment in other activities.

4. What until test have been covered during this shift

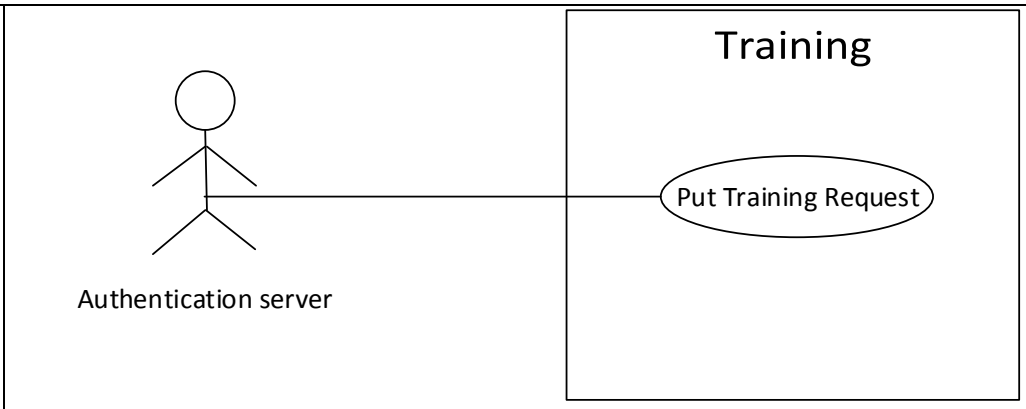
   - Make the deployment diagram.
   - Make the handoff.
   - Make the case uses and the explanatory table.
   - Propose an architecture to model the solution.
   - Make the Kanban
   - Make the alpha states advance.
   - Make negotiation with the stakeholders the main priorities

## 3. Work Products

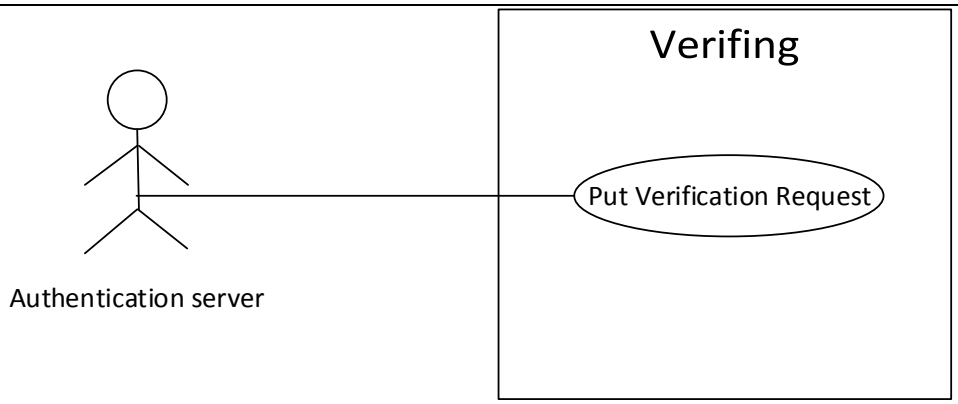
### 3.1. Use Case diagram

| **Use Case** | UC01 Put Training Request |
|---|---|

| | | | |
|---|---|---|---|
| **Version** | 1.0.0 | **Date:** | 15/03/2016 |
| **Author** | Luis Miguel Morales Alvarez | | |
| **Source** | DSD Process Work Products | | |
| **Purpose** | Receives a training request from the verification server | | |
| **Goals** | G1: Guarantee all training requests are received.<br>G2: Store properly the training request.<br>G3: Create a consistent training request status. | | |
| **Summary** | Receives and prepares a training request to be sent to the face++ server by storing its data inside the facilitator subsystem and creating its status information. | | |
| **Actors** | A0: Verification server | | |
| **Precondition** | | | |
| **Interaction Sequence** | **Authentication server** | **Facilitator** | |
| **1** | Call the operation of pass training request of the facilitator interface and gives an instance of a training request as parameter. | Puts the training request in the request manager and creates its status. | |
| **2** | Finish the operation. | Returns the method. | |
| **Alternative sequence** | **Authentication server** | **Facilitator** | |
| | There's isn't alternative sequences for this use case. | | |
| **Duration** | Time lapse of a method call | | |
| **Frequency** | When a new user request to the verification server a face training. | | |
| **Type** | Primary | | |
| **Post conditions** | | | |

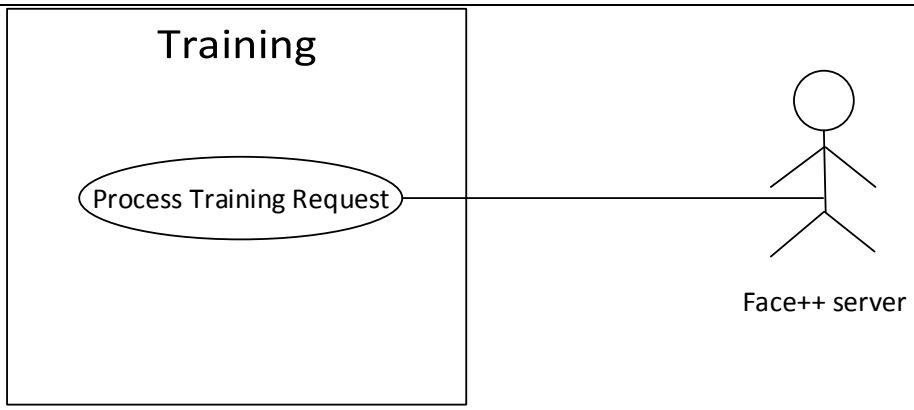
| | |
|---|---|
| **Chart** |  |
| **Interface** | |

| | | | |
|---|---|---|---|
| **Use Case** | UC02 Put Verification Request | | |
| **Version** | 1.0.0 | **Date:** | 15/03/2016 |
| **Author** | Luis Miguel Morales Alvarez | | |
| **Source** | DSD Process Work Products | | |
| **Purpose** | Receives a verification request from the verification server | | |
| **Goals** | G4: Guarantee all verification requests are received.<br>G5: Store properly the verification request.<br>G6: Create a consistent verification request status. | | |
| **Summary** | Receives and prepares a verification request to be sent to the face++ server by storing its data inside the facilitator subsystem and creating its status information. | | |
| **Actors** | A0: Verification server | | |
| **Precondition** | | | |
| **Interaction Sequence** | Verification server | | Facilitator |
| **1** | Call the operation of pass training request of the facilitator interface and gives an instance of a training request as parameter. | | Puts the verification request in the request manager and creates its status. |
| **2** | Finish the operation. | | Returns the method. |

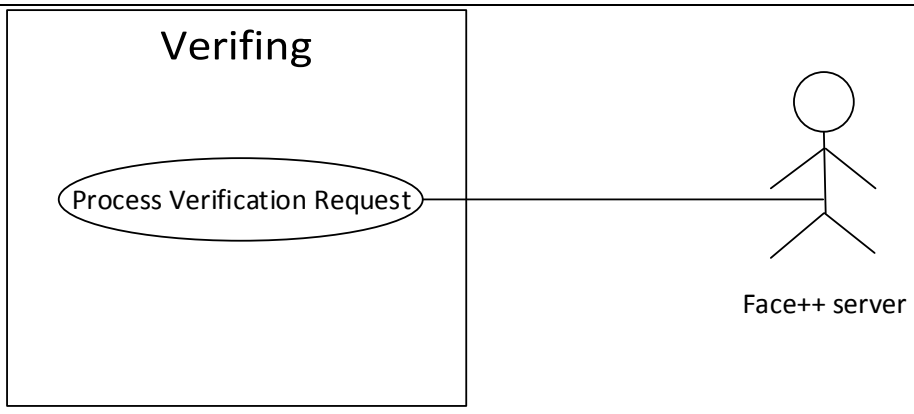
| **Alternative sequence** | Verification server | Facilitator |
|---|---|---|
| | There's isn't alternative sequences for this use case. | |
| **Duration** | Time lapse of a method call | |
| **Frequency** | When a new user request to the verification server a face training. | |
| **Type** | Primary | |
| **Post conditions** | | |
| **Chart** |  | |
| **Interface** | | |

| **Use Case** | UC03 Process Training Request | | |
|---|---|---|---|
| **Version** | 1.0.0 | **Date:** | 15/03/2016 |
| **Author** | Luis Miguel Morales Alvarez | | |
| **Source** | DSD Process Work Products | | |
| **Purpose** | Sends a training request to the Face++ server. | | |
| **Goals** | G7: Guarantee the training requests stages are responded in short time if Face++ server is available.<br>G8: Update the training request status properly.<br>G9: Complete all training request queued in the facilitator.<br>G10: Comply the training request sequence correctly. | | |
| **Summary** | Takes a training request and transforms it in a requests sequence that Face++ server can understand and sends, process the face++ responses and generates the training response. | | |

| | | |
|---|---|---|
| **Actors** | A1: Face++ server | |
| **Precondition** | The training request must be located in the request manager. | |
| **Interaction Sequence** | Facilitator | Face++ server |
| **1** | Takes a training request from the request manager. | |
| **2** | Check Face++ server availability. | Returns server status. |
| **3** | Creates http request for face detection from the training request training images and sends to Face++ server. | Returns face detection Json response. |
| **4** | Update training request face detection status in request manager. | |
| **5** | Creates an http request for person creation from training request and sends to Face++ server. | Returns person creation Json response. |
| **6** | Update training request person creation status in request manager. | |
| **7** | Creates http request for face set training and sends to Face++ server | Return session id Json response. |
| **8** | Update training request face set training status in request manager. | |
| **9** | Creates http request for querying the training request result and sends to Face++ server. | Returns training result Json response. |
| **10** | Finish training request sequence, creates the training response from finished training request, puts training response in response manager, and closes the training request. | |
| **Alternative sequence** | Facilitator | Face++ server |
| **2** | Waits for the Face++ server until it's available. | |

| | | |
|---|---|---|
| **4** | Error in the face detection, Creates training response with face detection error information and puts in response manager, closes the Training request. | |
| **6** | Error in the person creation, Creates training response with person creation error information and puts in response manager, closes the Training request. | |
| **8** | Error in face set training, Creates training response with face set training error information and puts in response manager, closes the Training request. | |
| **10** | Error in query face set training result, Creates training response with query face set training result error information and puts in response manager, closes the Training request. | |
| **Duration** | 0.5 seconds for each training photo. | |
| **Frequency** | When a new user request to the verification server a face training. | |
| **Type** | Secondary | |
| **Post conditions** | Creates a Training response | |
| **Chart** |  | |
| **Interface** | | |

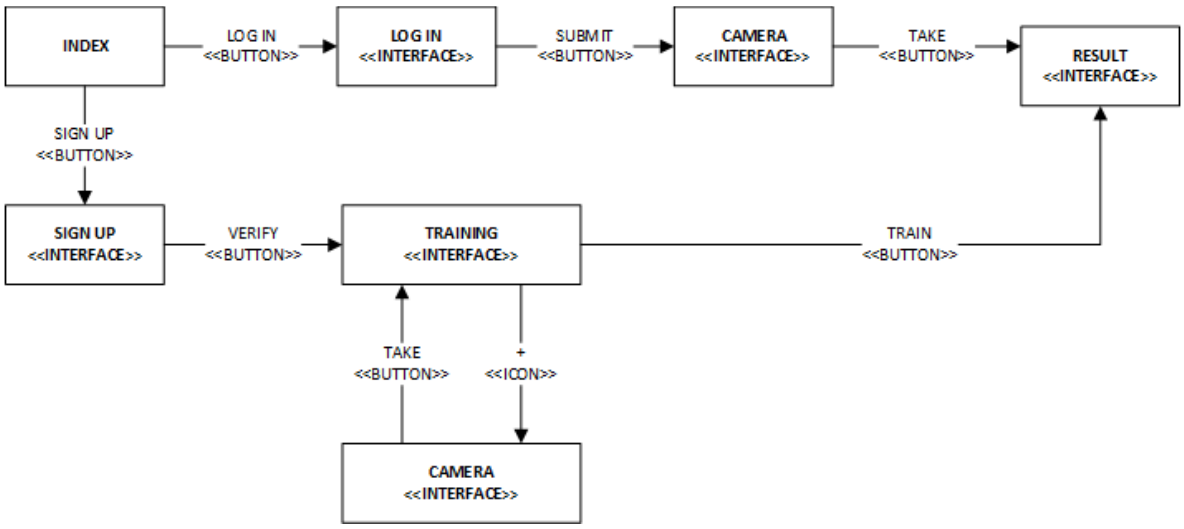
| **Use Case** | UC04 Process Verification Request | | |
|---|---|---|---|
| **Version** | 1.0.0 | **Date:** | 15/03/2016 |
| **Author** | Luis Miguel Morales Alvarez | | |
| **Source** | DSD Process Work Products | | |
| **Purpose** | Receives a verification request from the verification server | | |
| **Goals** | G11: Guarantee the verification requests stages are responded in short time if Face++ server is available.<br>G12: Update the verification request status properly.<br>G13: Complete all verification request queued in the facilitator.<br>G14: Comply the verification request sequence correctly. | | |
| **Summary** | Takes a verification request and transforms it in a requests sequence that Face++ server can understand and sends, process the face++ responses and generates the verification response. | | |
| **Actors** | A1: Face++ server | | |
| **Precondition** | The verification request must be located in the request manager. | | |
| **Interaction Sequence** | Facilitator | Face++ server | |
| **1** | Takes a verification request from the request manager. | | |
| **2** | Check Face++ server availability. | Returns server status. | |
| **3** | Creates http request for face detection from the verification request image and sends to Face++ server. | Returns face detection Json response. | |
| **4** | Update verification face detection status in request manager. | | |
| **5** | Creates http request for face and person verification and sends to Face++ sever. | Returns person verification Json response. | |
| **6** | Finish the verification request sequence, creates verification response from finished verification | | |

| | | |
|---|---|---|
| | request, puts verification response in the response manager, closes verification request. | |
| **Alternative sequence** | Facilitator | Face++ server |
| **2** | Waits for the Face++ server until it's available. | |
| **4** | Error in the face detection, Creates verification response with face detection error information and puts in response manager, closes the verification request. | |
| **6** | Isn't the same person in the photo, Creates verification response with person mismatch information and puts in response manager, closes the verification request. | |
| **Duration** | 0.5 seconds for verification photo. | |
| **Frequency** | When a new user request to the verification server a face training. | |
| **Type** | Secondary | |
| **Post conditions** | Creates a verification response. | |
| **Chart** |  | |
| **Interface** | | |

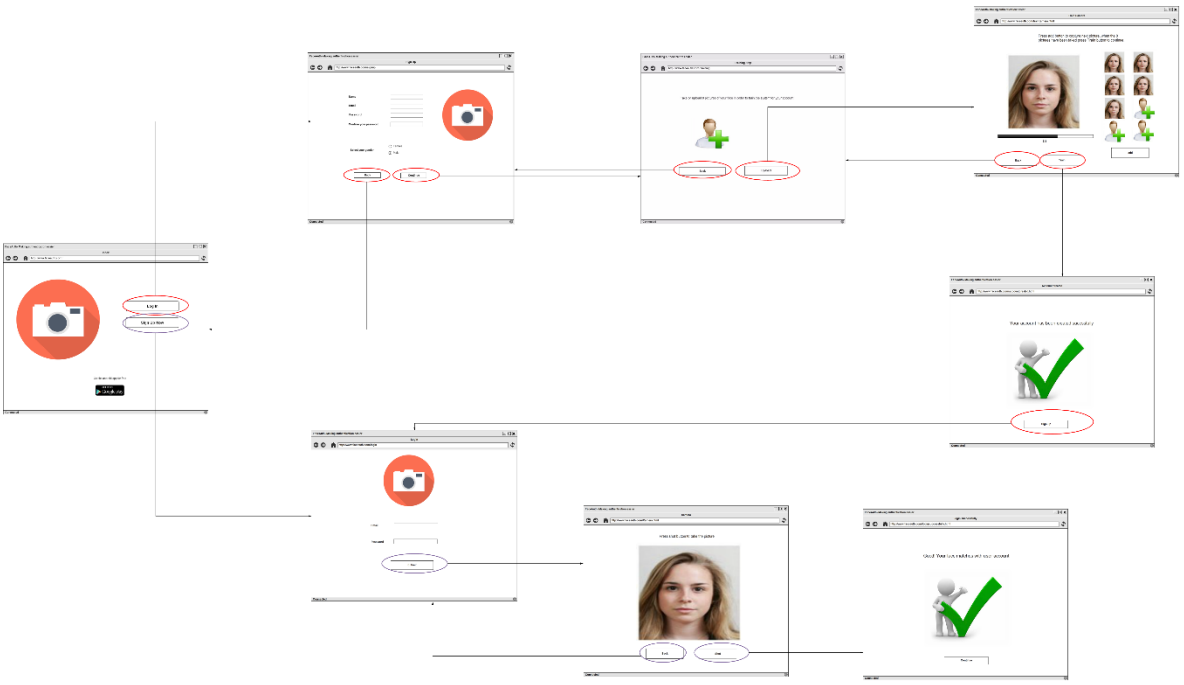
### 3.2. GUI flow diagram

**As the facilitator does not have a GUI as such, then we use the interfaces that provide both Android and PC client.**

## GUI flow diagram Android Client




## GUI flow diagram PC Client.



## 3.3. Class Diagram