# Task 6.2: AAA provisioning at the Abstraction Layer

# AAA as a Service - Currently available API for authentication services

| | |
|---|---|
| Author(s) | Paulo Silva (UC), Paulo Simões (UC), Edmundo Monteiro (UC) |
| Status | **Draft**/Review/Approval/Final |
| Version | |
| Date | 31/03/2017 |

Dissemination Level

| | |
|---|---|
| X | PU: Public |
| | PP: Restricted to other programme participants (including the Commission) |
| | RE: Restricted to a group specified by the consortium (including the Commission) |
| | CO: Confidential, only for members of the consortium (including the Commission) |

## Table Of Contents

## Table of Figures

# 1 Introduction

This document presents the API methods available for authentication as a service (Applications AAAaaS). It presents as well sequence diagrams for the respective methods.

Regarding the Applications AAAaaS, there are two options:

1 - User access to AAA server user interface

By using the AAA server interface, the user interaction with AAA related functions, are directly in the AAA server user interface. There are web pages with forms for a user to introduce its login credentials and to register (currently).

In this option, considering the login action as an example, when the user presses login button in the application, it is redirected to AAA authentication server and it is presented with a login page where he can introduce its credentials.

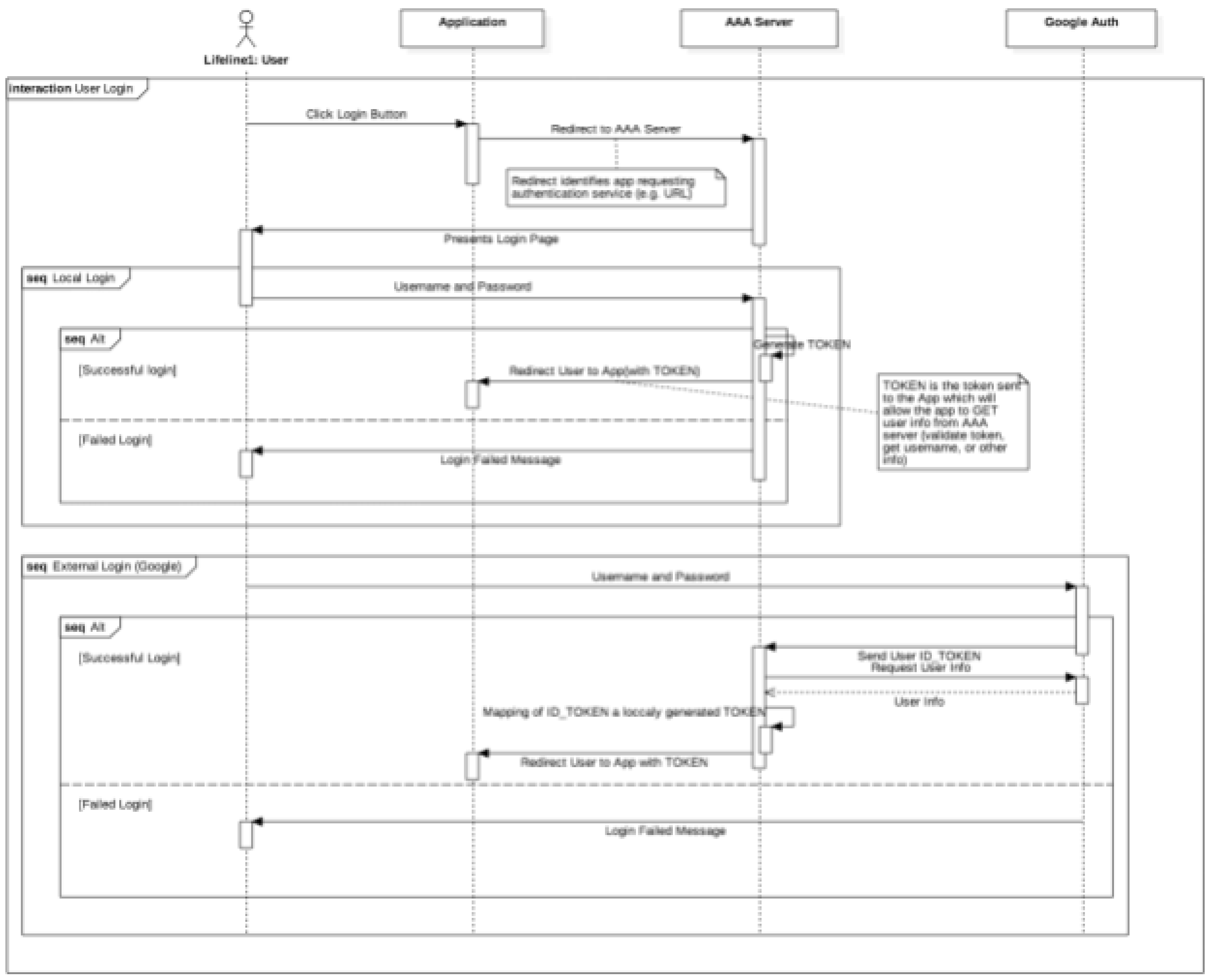
2 - User access to application’s user interface

By using the application’s user interface, the user interactions with authentication related options (e.g. login, register) are made within the application. Upon those interactions, the application makes requests to the AAA authentication server API. The requests contain the user relation information for validation (e.g. username, password) or registration.

In this option, considering the login action as an example, when the user presses login button in the application, it is presented with a login page within the application where he can introduce its credentials. Then, the application makes a request to the AAA authentication server API in order to validate user’s credentials. After the validation of the request, if there is a positive validation, the AAA authentication server, replies back to the application with a token associated to the user credentials that were being verified - confirming that the user exists and authentication was successfull.

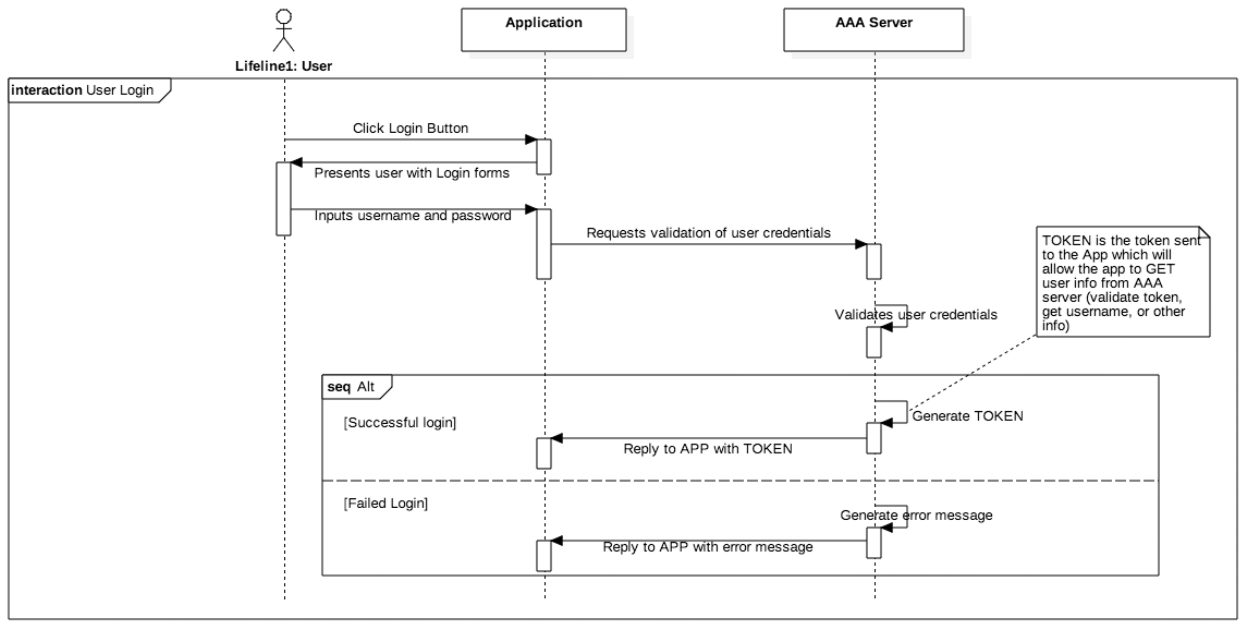
## 2 Sequence Diagrams

### 2.1 User access to AAA server login webpage



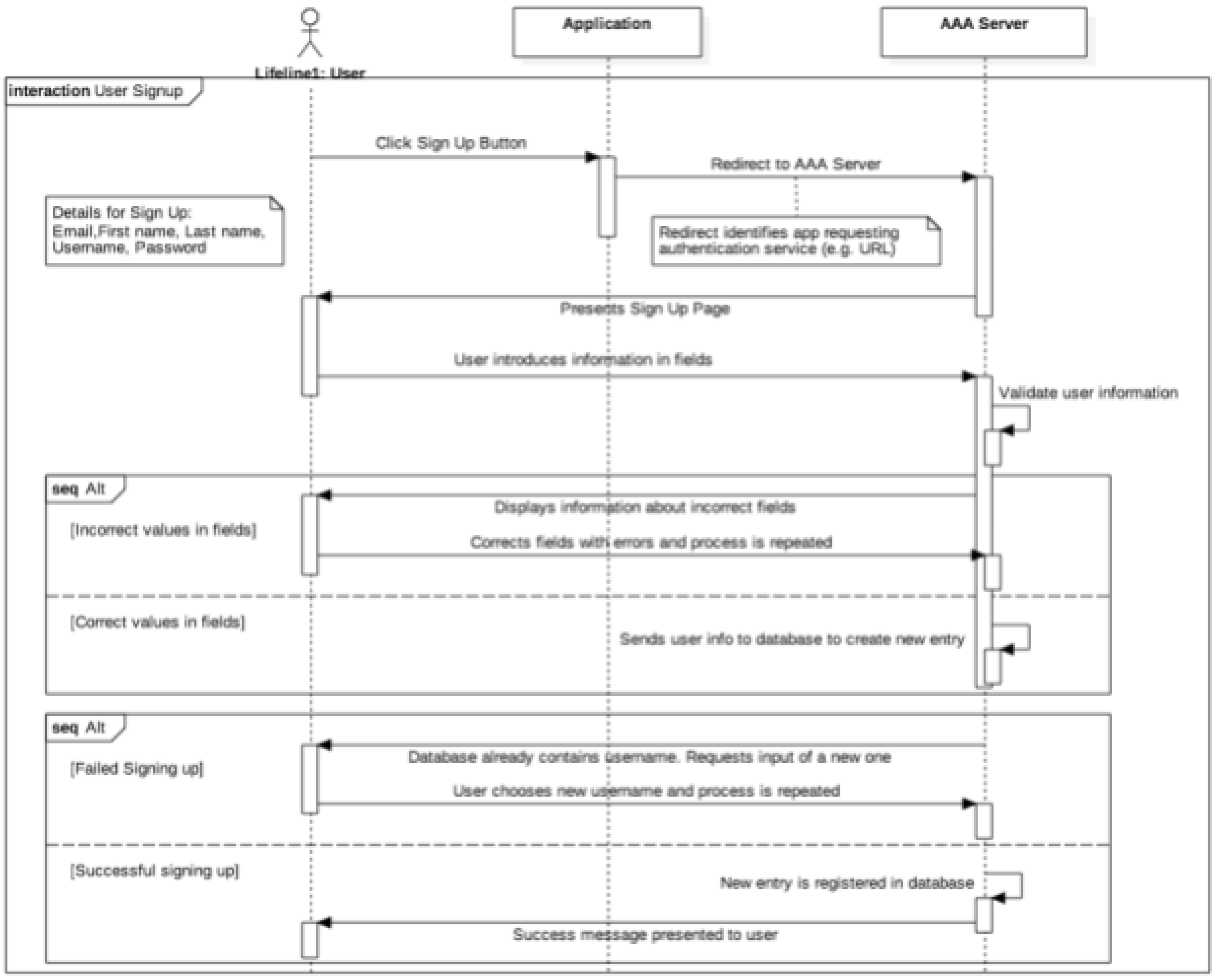
This sequence diagram presents the login flow for a user's login using the AAA server authentication page. There are two possibilities: local authentication, external authentication. Currently, local authentication is being functional. External authentication will be made available later (not for demo).

## 2.2 User access to application's login webpage


Lifeline1: User
Application
AAA Server
interaction User Login
Click Login Button
Presents user with Login forms
Inputs username and password
Requests validation of user credentials
Validates user credentials
TOKEN is the token sent to the App which will allow the app to GET user info from AAA server (validate token, get username, or other info)
seq Alt
[Successful login]
Generate TOKEN
Reply to APP with TOKEN
[Failed Login]
Generate error message
Reply to APP with error message

This sequence diagram presents the login flow for a user's login using the application user interface. Application makes a request with user's credentials for validation.

## 2.3 User access to AAA server sign up webpage


Lifeline1: User
Application
AAA Server
interaction User Signup
Click Sign Up Button
Redirect to AAA Server
Details for Sign Up: Email,First name, Last name, Username, Password
Redirect identifies app requesting authentication service (e.g. URL)
Presents Sign Up Page
User introduces information in fields
Validate user information
seq Alt
[Incorrect values in fields]
Displays information about incorrect fields
Corrects fields with errors and process is repeated
[Correct values in fields]
Sends user info to database to create new entry
seq Alt
[Failed Signing up]
Database already contains username. Requests input of a new one
User chooses new username and process is repeated
[Successful signing up]
New entry is registered in database
Success message presented to user

# 3 Interaction with AAA as a Service API

## 3.1 REST API endpoints

The following table describes the REST calls that can be made to the AAA authentication server API.

| Address | Action |
|---|---|
| engine/api/checkin_data | POST method, verifies credentials and provides user data |
| engine/api/verify_token | POST method, verifies token and replies with user name |
| engine/api/signup_data | POST method, creates user entry in database |
| engine/api/checkout_data | POST method, invalidates token from user |
| /engine/api/update_user | POST method, updates user information |
| /engine/api/delete_user | POST method, deletes user account |

The currently available methods allow the following:

- signing up of a new user;

- signing in of an existing user;

- verification of a token associated to a user;

- signing out of a user;

- update/change user information;

- delete user account.

In the next section, there are examples of application of the described methods. The examples contain the calling of the method (data and endpoint) as well as the respective responses in positive or negative case (if applicable).

## 3.2 Examples with cURL commands

Following are presented examples of requests and responses for each API method.

### 3.2.1 Checkin_data

https://eubrabigsea.dei.uc.pt/engine/api/checkin_data

Request:
curl --data "user=testuser&pwd=123456" https://eubrabigsea.dei.uc.pt/engine/api/checkin_data

Response in positive case:
{"error": "", "success": true, "cancelled": false, "user_info": {"user_token": "d95e965b4e818ab8150e1a25a4890fd31c2c662e1b8a7c86b6664a7bd3f4e336d763822e25acd762f27d787cbe8fdb55cdde655d72b4d76748541bb6d059decc", "user": {"lname": "silva", "username": "testuser", "fname": "paulo"}}}

Response in negative case:
{"error": "Invalid username or password.", "success": false, "cancelled": false, "user_info": null}

### 3.2.2 Verify_token

https://eubrabigsea.dei.uc.pt/engine/api/verify_token

Request (with token gathered from previous example):
curl --data "token=67c45c889b9172ca7e4f9ea4e1de256d1486af5b911bae50b367605e1dfc9bac319adb1654c31e43e8657d9b72825cc5e421764bf5e58e971dab2ee66e213902" https://eubrabigsea.dei.uc.pt/engine/api/verify_token

Response in positive case (the username associated with the token):
{"response": "testuser"}

Response in negative case (if token does not match any user):
{"response": "invalid token"}

### 3.2.3 Sign up

https://eubrabigsea.dei.uc.pt/engine/api/signup_data

Request:

```
curl --data
"user=testuser&pwd=123456&fname=testname&lname=testsurname&email=testexample@example.com"
https://eubrabigsea.dei.uc.pt/engine/api/signup_data
```

Response in positive case:

```
{"success": "User signed up with success!"}
```

Response in negative case:

```
{"error": "Username already exists. Please choose a different one."}
```

### 3.2.4 Sign out

https://eubrabigsea.dei.uc.pt/engine/api/checkout_data

Request (with token of the user signing out):

```
curl --data
"token=84e1a2f2ef6ee78240ad98af4cf055837f167feac554cb5bc6e6c69a1b76227429be37fc05060a8c8285
2fd8aef03dc3769e1c244603cc82a5267b4f6282576a"
https://eubrabigsea.dei.uc.pt/engine/api/checkout_data
```

Response (If token exists, or not, the response is always empty):

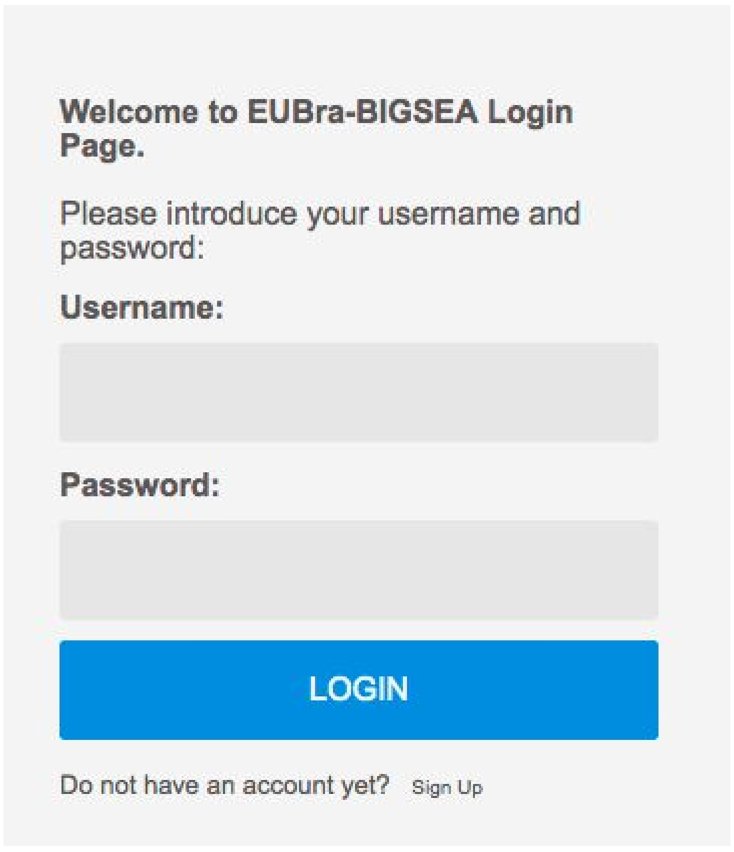
```
{}
```

# 4 Redirection Addresses and page layouts

Applications that need to use the authentication server login and register page can do so by redirecting the users to the respective pages.

## 4.1 Login

Users are redirected to "[https://eubrabigsea.dei.uc.pt/](https://eubrabigsea.dei.uc.pt/)" . On the page, users can introduce their credentials (username and password). Once the login is successful, users are redirected back to the application which made the initial redirect.



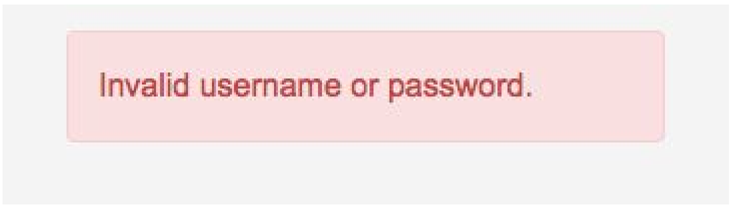
Welcome to EUBra-BIGSEA Login Page.

Please introduce your username and password:

Username:

Password:

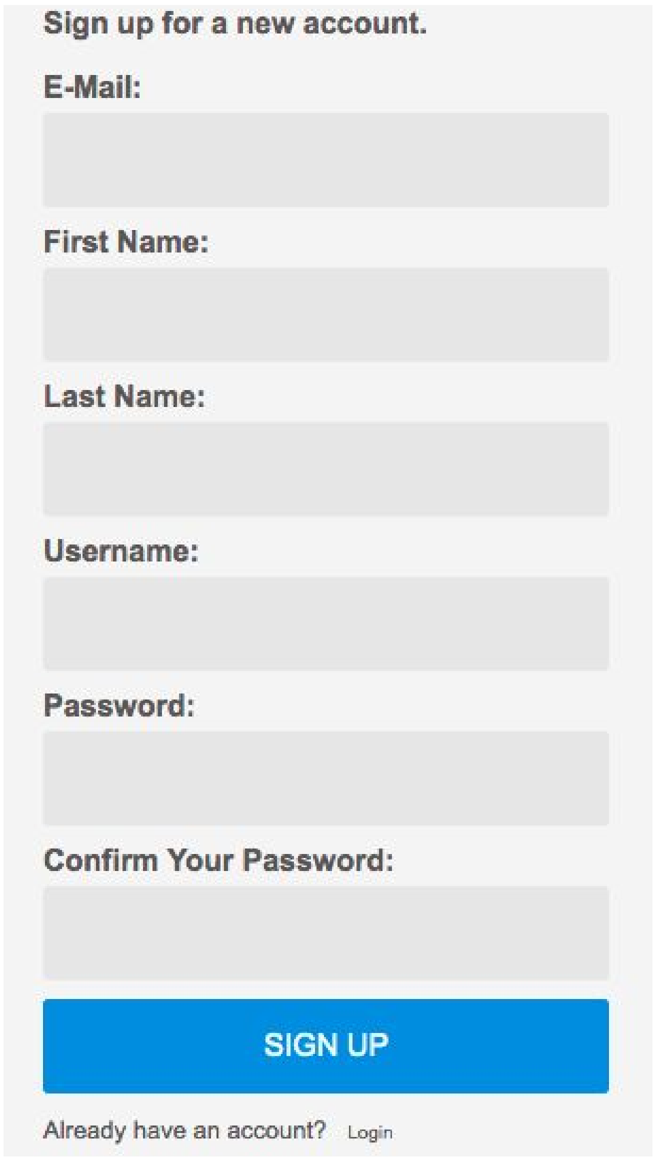
If the user's credentials are incorrect or do not match any record in the user's database, an error message is presented.



Invalid username or password.

## 4.2 Sign Up

User are redirected to "https://eubrabigsea.dei.uc.pt/web/signup". On the page, they can introduce their information: email, first name, last name, username and password. Once the fields are complete (with the user's information), the user presses the "Sign Up" button and a validation of the fields is performed.


Sign up for a new account.

E-Mail:

First Name:

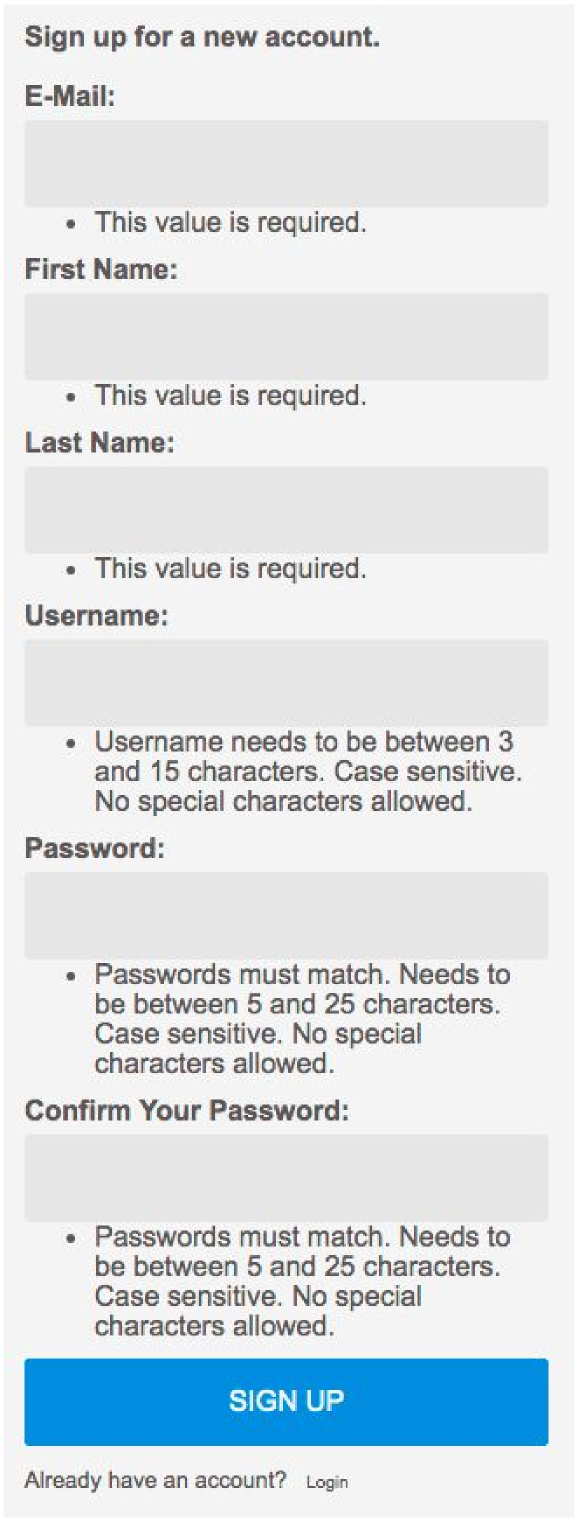
Last Name:

Username:

Password:

Confirm Your Password:

All the fields are required for the signing up process. If the validation process detects that some fields do not contain the required information, warning message(s) are presented to the user in order to correct the information.


Sign up for a new account.

E-Mail:

- This value is required.

First Name:

- This value is required.

Last Name:

- This value is required.

Username:

- Username needs to be between 3 and 15 characters. Case sensitive. No special characters allowed.

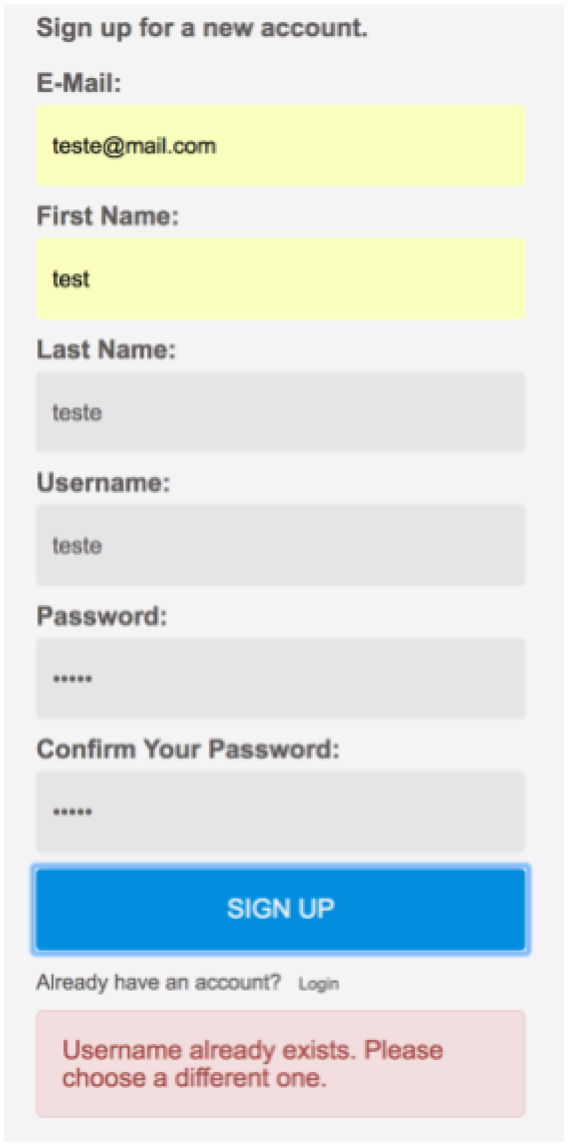
Password:

- Passwords must match. Needs to be between 5 and 25 characters. Case sensitive. No special characters allowed.

Confirm Your Password:

- Passwords must match. Needs to be between 5 and 25 characters. Case sensitive. No special characters allowed.

After all the fields contain the required information, the pressing of the "Sign Up" button requests the registration of the user in the database. If the username does not exist in the database, the registration process is concluded with success. If the username already exists, then an error message is presented and the user is asked to choose a different username.



Sign up for a new account.

E-Mail: teste@mail.com

First Name: test

Last Name: teste

Username: teste

Password: •••••

Confirm Your Password: •••••

Username already exists. Please choose a different one.

# 5 Configuration for redirects (popup's)

## 5.1 Button to create popup with "Sign In" page

This step creates a button which opens a popup window (or new tab) with EUBra-BIGSEA authentication page.

In the application create a button pointing to the "Sign In" page. Following there is an example:

```
<input type="button" value="Open window" onclick="openWin('https://eubrabigsea.dei.uc.pt'); return false;" />
```

The function "openWin" is called with the address of the "Sign In" page as a parameter (https://eubrabigsea.dei.uc.pt).

To receive the answer of the popup page, a JavaScript function is needed (as well as the openWin function to create the popup). The following code should be in the Javascript file associated with the webpage of the app. Please note that the function "HandlePopupResult" is the one handling the answer.

```
function openWin(url, w, h)
{
        console.log('openwindow');
        var left = (screen.width/2)-(w/2);
        var top = (screen.height/2)-(h/2);
        w=window.open(url, '_blank');
        w.focus();
}

var answer_data;
// answer_data is the variable that will contain the answer from the popup page
function HandlePopupResult(answer_data) {
        console.log('Received the answer!');
        console.log(answer_data); This displays in the console the received JSON answer
        //From here, take care of answer. Do your logic with the returned JSON
}
```

The format of the JSON answers are described in section 3.2.1 - Checkin_data.

## 5.2 Button to create popup with “Sign Up” page

This second step creates a button which opens a popup window (or new tab) with EUBra-BIGSEA sign up page.

In the application create a button pointing to the “Sign Up” page. Following there is an example:

```
<input type="button" value="Open window"
onclick="openWin('https://eubrabigsea.dei.uc.pt/web/signup'); return false;" />
```

The function “openWin” is called with the address of the “Sign Up” page as a parameter (https://eubrabigsea.dei.uc.pt/web/signup ).

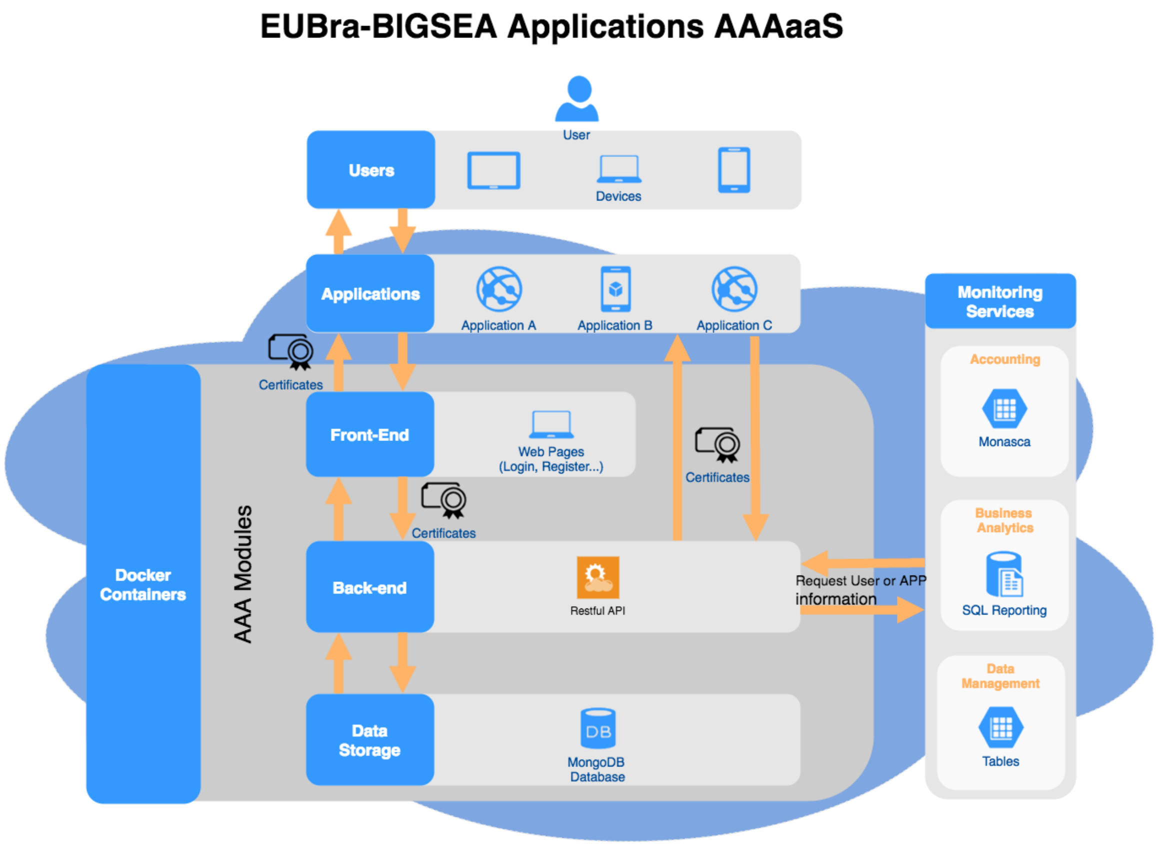
To receive the answer of the popup page, a JavaScript function is needed (as well as the openWin function to create the popup). The following code should be in the Javascript file associated with the webpage of the app. Please note that the function “HandlePopupResult” is the one handling the answer.

```
function openWin(url, w, h)
{
        console.log('openwindow');
        var left = (screen.width/2)-(w/2);
        var top = (screen.height/2)-(h/2);
        w=window.open(url, '_blank');
        w.focus();
}

var answer_data;
// answer_data is the variable that will contain the answer from the popup page
function HandlePopupResult(answer_data) {
        console.log('Received the answer!');
        console.log(answer_data); This displays in the console the received JSON answer
        //From here, take care of answer. Do your logic with the returned JSON
}
```

The format of the JSON answers are described in section 3.2.3 - Sign up.

## 6 ARCHITECTURE


EUBra-BIGSEA Applications AAAaaS
User
Users
Devices
Applications
Application A
Application B
Application C
Monitoring Services
Certificates
Front-End
Web Pages (Login, Register...)
Certificates
Certificates
Docker Containers
AAA Modules
Back-end
Restful API
Request User or APP information
Data Storage
MongoDB Database
Accounting
Monasca
Business Analytics
SQL Reporting
Data Management
Tables