Authentication API usage

Contents

Introduction

The SSOLO authentication server allow other servers to use the AUTH API for customer authentication.

The authentication procedure use a SCA (Strong customer authentication) and are compliant with GDPR (General Data Protection Regulation) .

To accept connection from external sites the SSOLO AUTH server need to know the SSL public key of any systems that need to ask the login authorization.

The pubkey permit to AUTH server to are sure of the origin of the request.

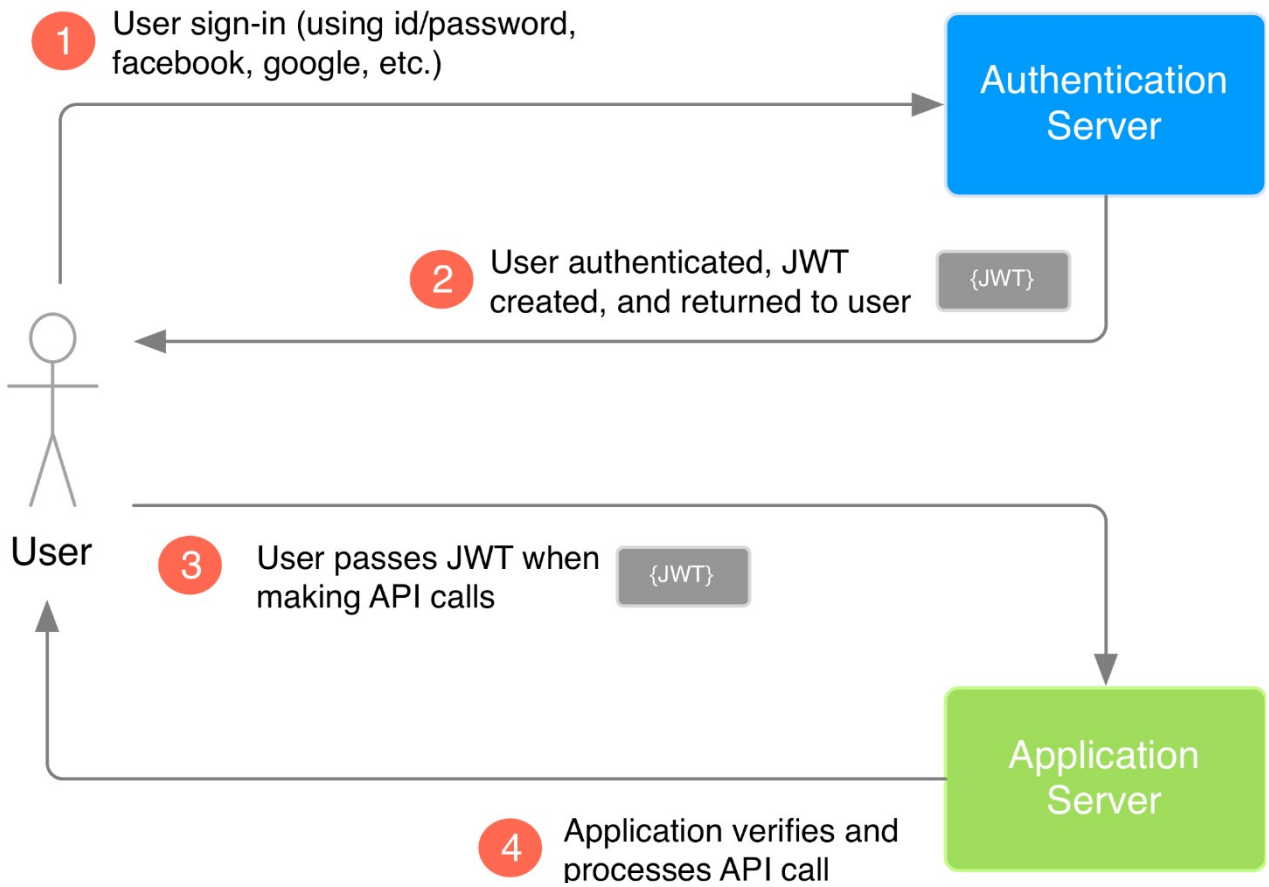
The authentication request may be done in two ways:

- Exposed credential (login and password passed with POST on https)
- Unexposed credential (login and password incapsulated in a JWT token)

All two type of authentication reply with a token and a JSON message that explain if the user is authorized or not for the login.

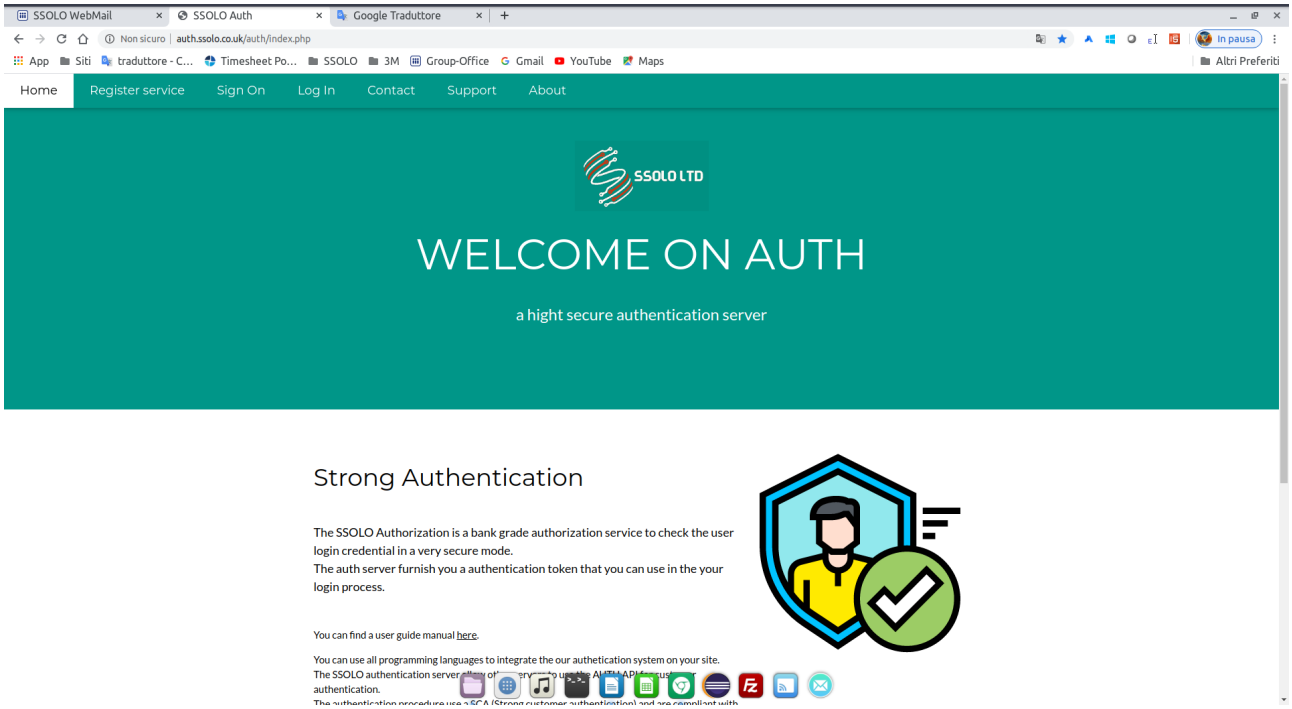
The token received must be inserted in all next API calls to have the grant at use the API.

The token have a 12 hours of life, if you request API access in the lifetime you not need to made a new login.



The authentication server

The authentication server is reacheable at URL https://auth.ssolo.co.uk.



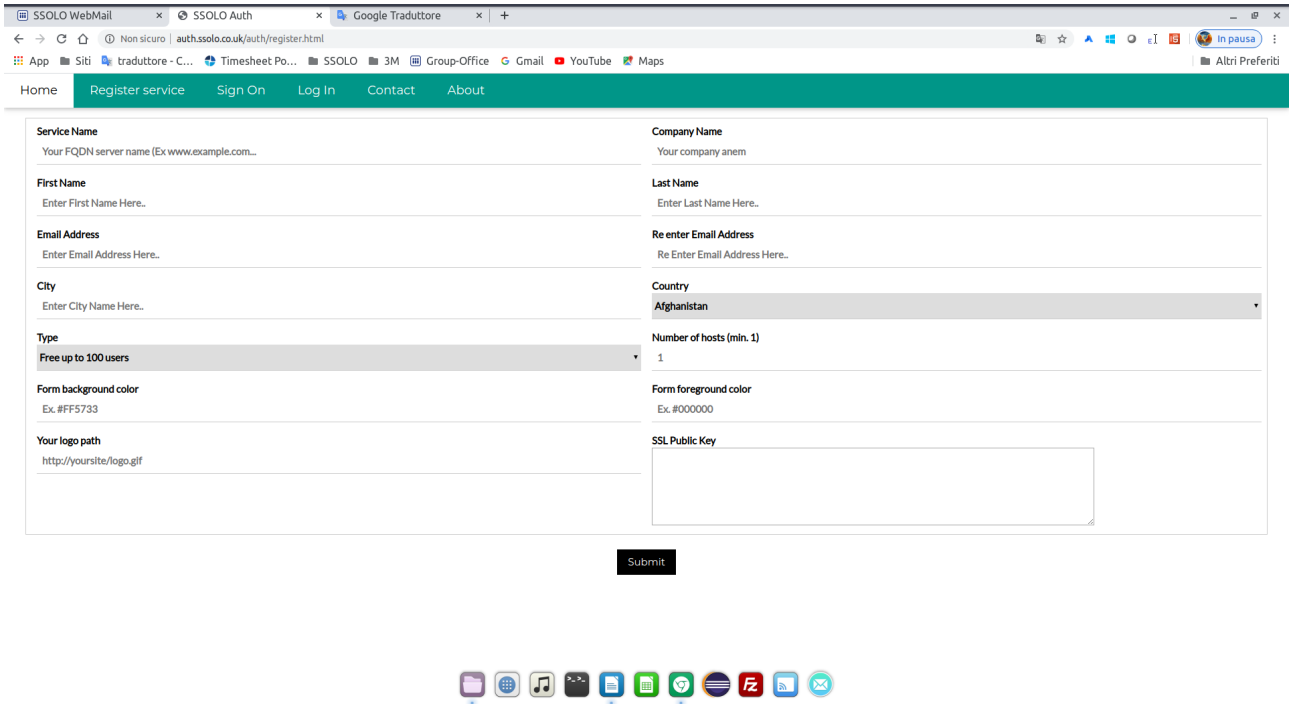
From this page, you can register a service, register a user not connected to a particular service or manage your site and your users.

The authentication server offers services both to companies that need to be sure of their users, and to individuals who want to create a verified login profile that allows them to access the services registered on AUTH without needing to register from time to time.

You will no longer have to send your documents to dozens of different sites, just register on AUTH and the internet services that ask you to register will be able to access them directly and securely.

Finally, AUTH offers anti-fraud control on all users who register.

Register a services



Registering a service involves providing a whole series of information regarding the activity that you want to activate on AUTH and the data of the service manager.

In addition to the data concerning the company and its manager, the public SSL key of the physical server hosting the service must be entered.

What is a public key?

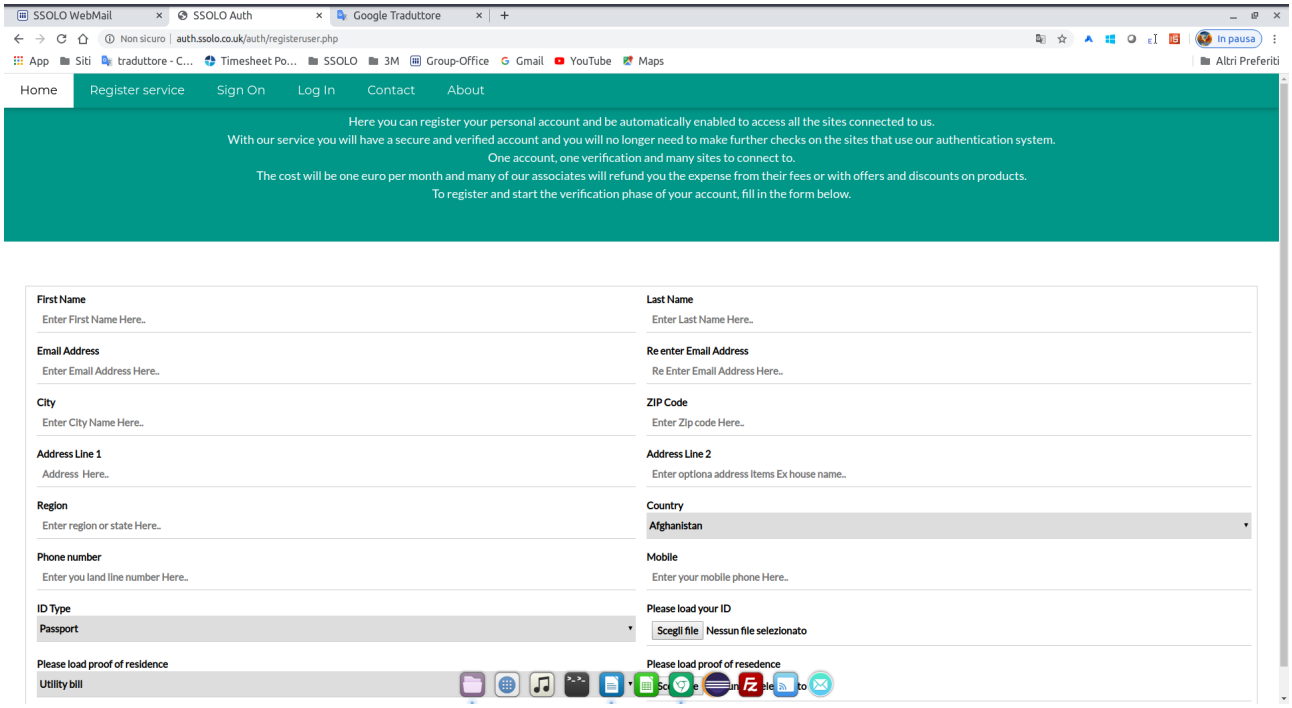
This is a rather long alphanumeric code, generated via SSL (Secure Socket layer) and which uniquely identifies the owner.

In addition to the public key SSL also generates a private key that the site manager must keep well secure and not provide to anyone.

When the service contacts the AUTH server, the request and the data it contains are encrypted using the site's private key.

Only those who have the public key will be able to decrypt this data, which allows AUTH to be sure of the origin of the data received.

User registration



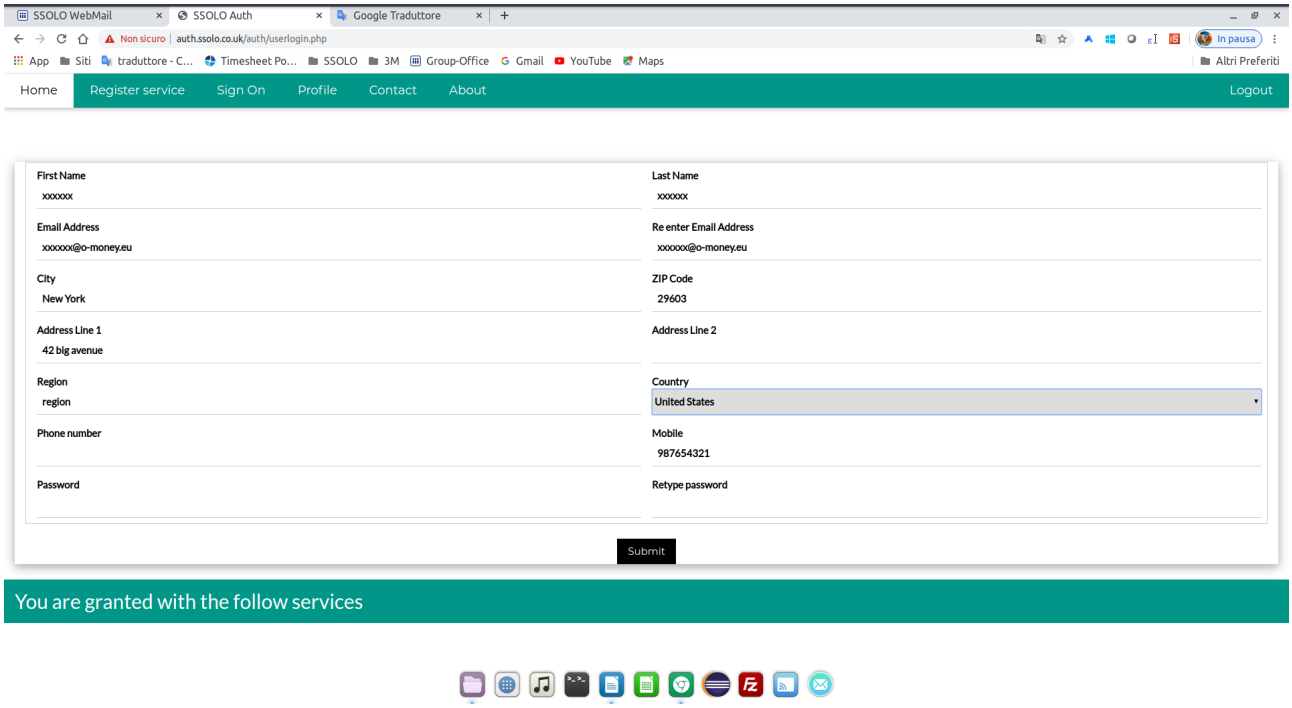
A private user can register his AUTH account and obtain login credentials for services that use AUTH, without having to register on each individual site.

During registration, the user will be asked for data such as name, surname, address and a proof of identity (ID or Passport) and a proof of residence (Utilities, bank statement, etc.).

All these data will be checked, an anti-fraud check carried out and the Secure Single Sign on account created.

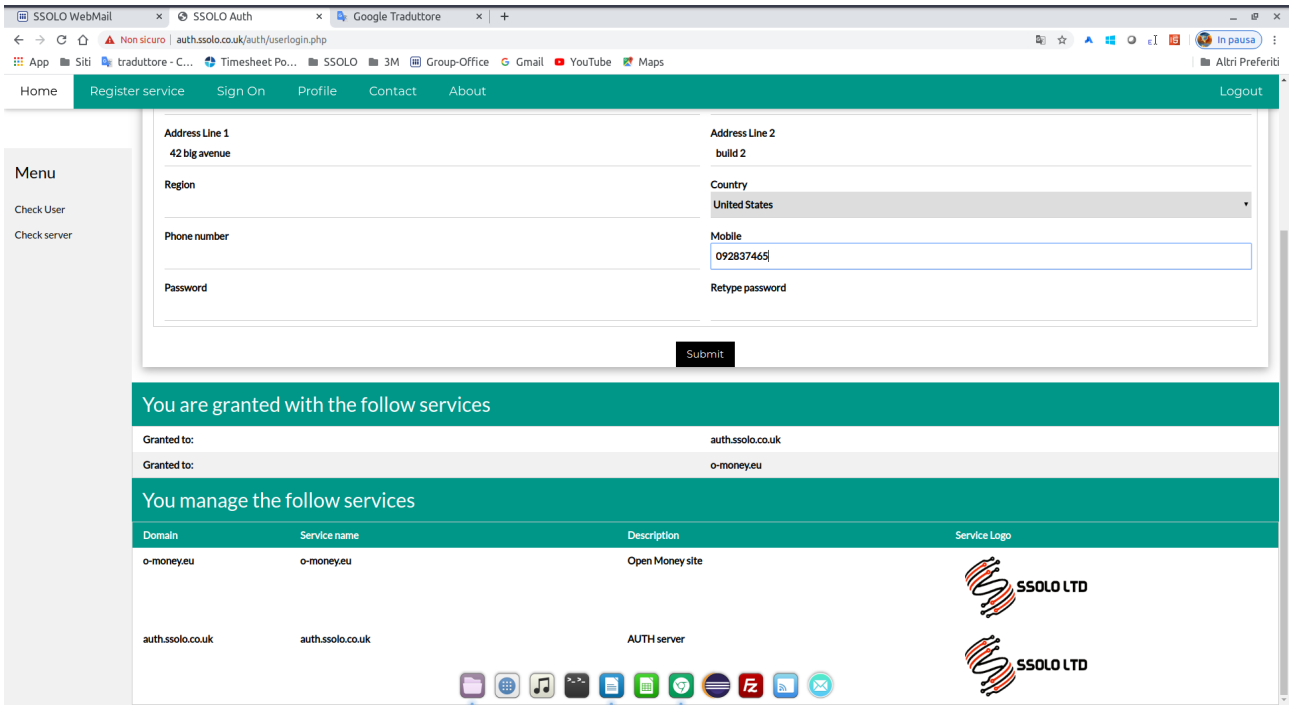
From this moment the user can register to the services operating on AUTH by simply inserting his username and password.

User management

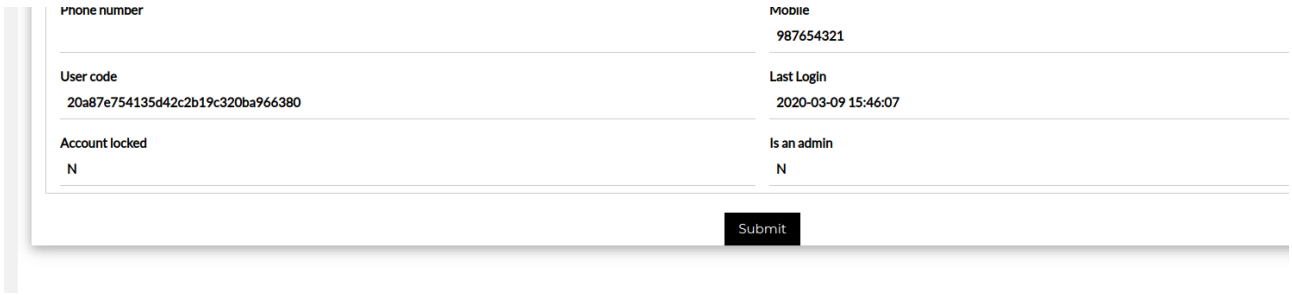


Once logged into the AUTH server, you can check and modify your personal data.

Any change that is made will allow a new check of the data entered, this to make the account always safe and verified for all the services that are accessed.



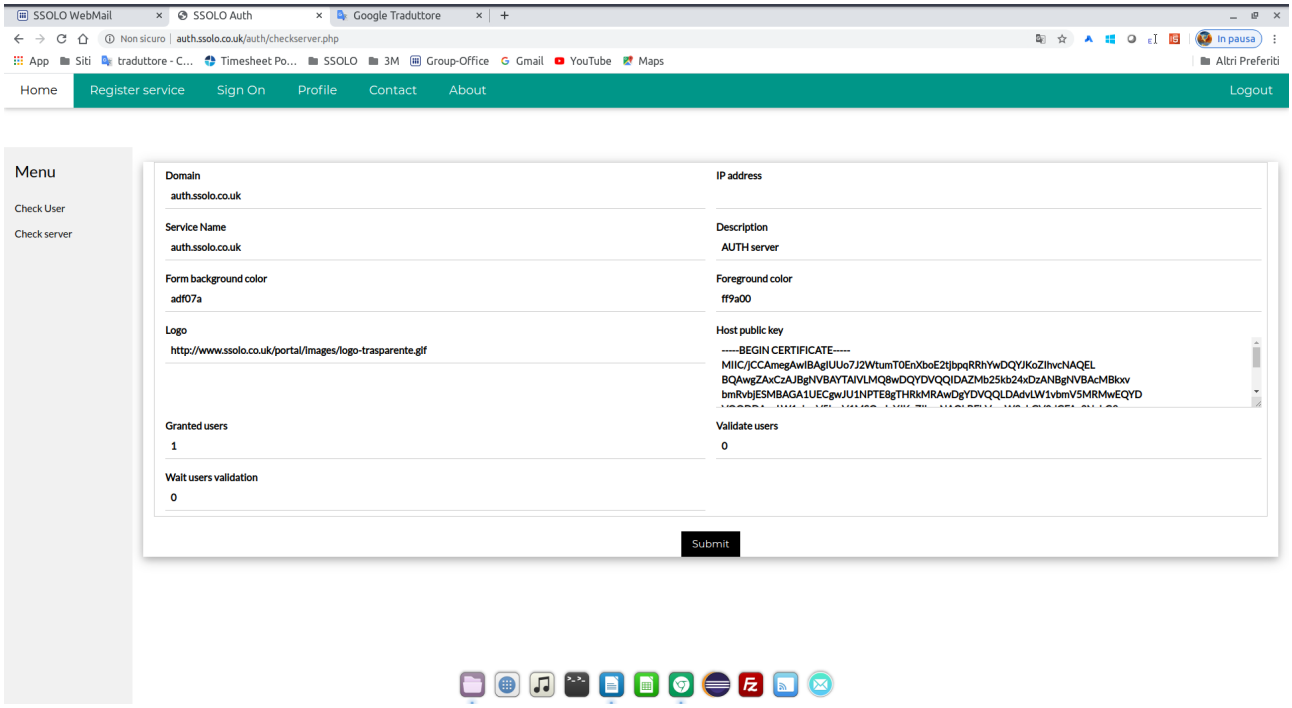
If you have registered as a service administrator (Sysadmin are all those who register a service on AUTH), after logging in, in addition to your personal data you will be able to see the servers you manage and you will have a menu on the left of the screen , with the choices to manage your users and your servers.



If you choose to manage a user account, all personal data will be read-only, only the owner of the data can modify it.

You can only block the account (you may have an incorrect user) or elevate that user to administrator for your service.

Service management

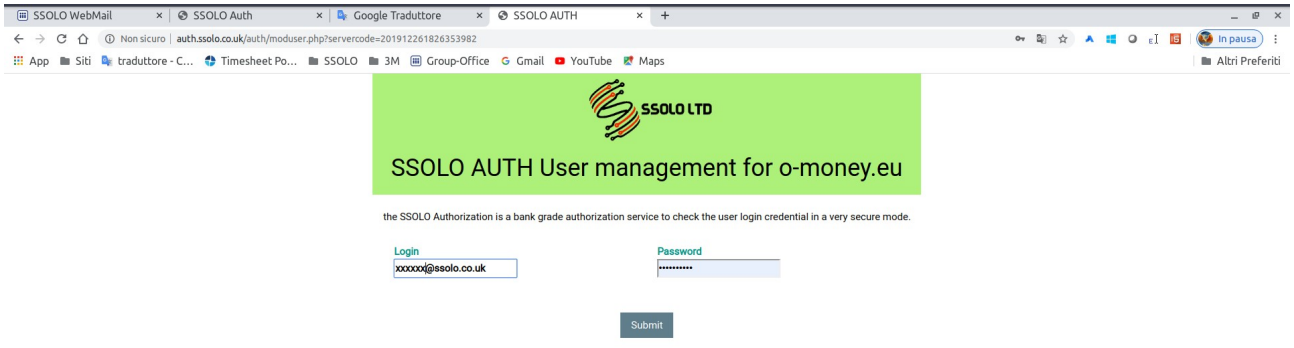


By choosing to manage a service, you will be able to change some parameters and data of your servers registered on AUTH (e.g. logo, background color of your pages on auth).

From this page you can also upload a new public key when the old one expires.

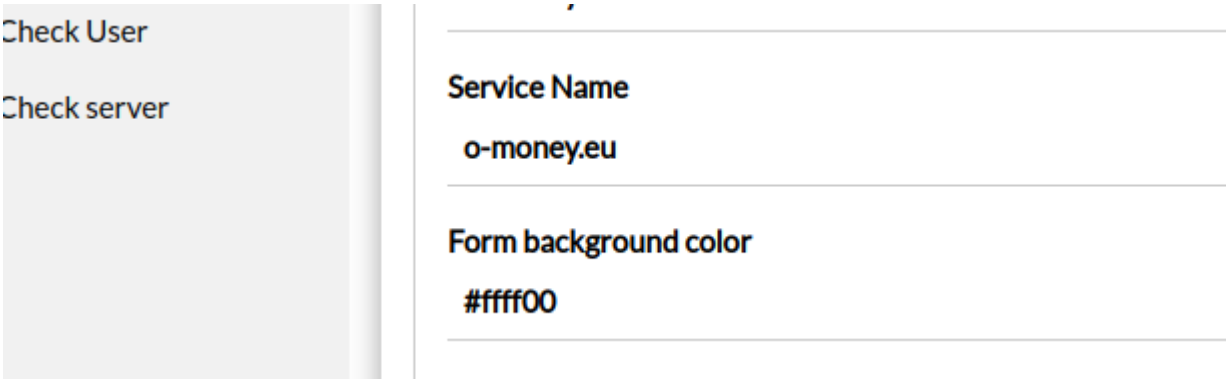
This entering us into the service data will have an impact on how the authentication server will respond to your requests made by the server of the service you have registered.

Let's take an example:

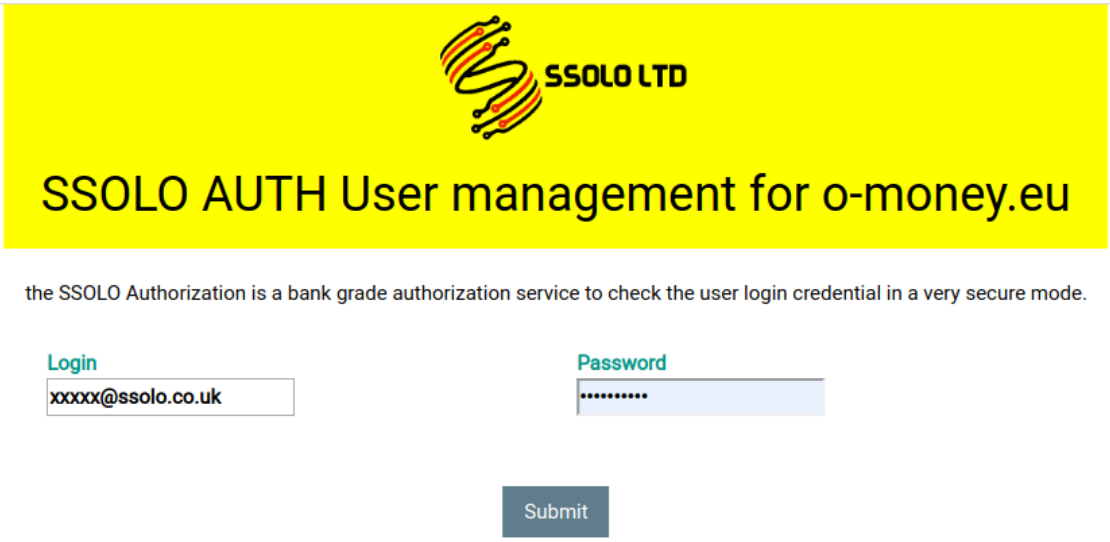


As you can see, this page presents your logo, name and colors you entered when registering the service.

Let's try changing the background color.



From the server management page, we modify the bgcolor field with the value # ffff00 (yellow) and save the configuration.

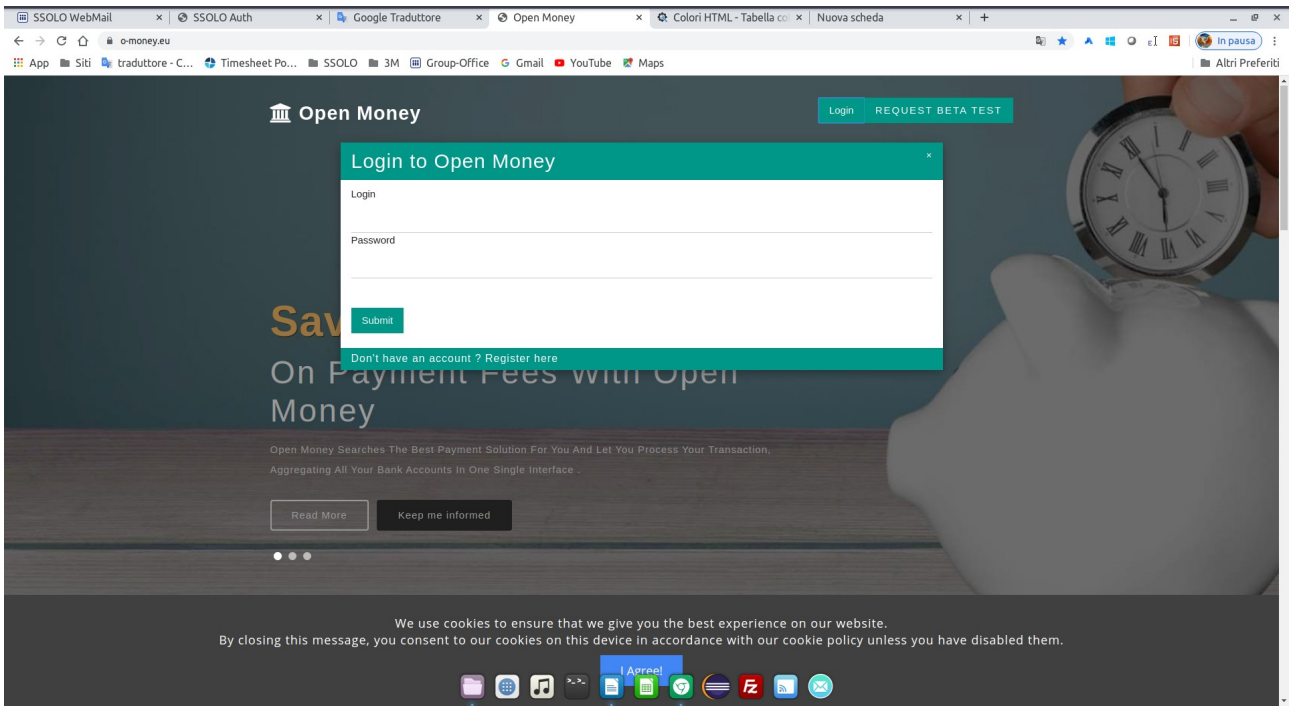


The background color of the page that will be presented to your user has been changed.

Service side web pages

If you want to have more personalized pages more similar to your website, you can create them on your server and use the AUTH development libraries to connect your service to the authentication system.

For example, if you wanted a custom login form, you can simply build it around the templates provided with the AUTH development environment.



Per ottenere un simile risultato puoi prelevare dal template login le funzioni di autenticazione e costruirgli intorno la tua interfaccia con il tuo stile.

```
<?php
/* template to get the auth token */
// ssolo library inclusion
include "ssoloauth-lib.php";
if (isset($_POST['submit'])){
    include "ssoloauth-lib.php";
    $username=$_POST['login'];
    $password=$_POST['password'];
```

```
$response = GetLogin($username,$password);
$valid=ValidateToken($response['data']['token'],$response['data']['secret']);
// if external authentication was successful
if( $valid[0]  == "invalid" ) {
   // User does not exist,  send back an error message
   echo "<div class=\"w3-row w3-padding-64\">";
   echo "<div class=\"w3-twothird w3-container\">";
   echo "<div w3-display-topleft>";
   echo "<img src=images/openmoneylogo.png width=\"400\">";
   echo "</div>";
   echo "<h1 class=\"w3-text-teal\">Login invalid</h1>";
   echo "<br><br>";
   echo "<a href=\"https://o-money.eu\">Please try a new login</a>";
   echo "</div></div>";
   exit;
} else {
         if (!empty($_SERVER['HTTP_CLIENT_IP'])) {
      $ip = $_SERVER['HTTP_CLIENT_IP'];
   } elseif (!empty($_SERVER['HTTP_X_FORWARDED_FOR'])) {
      $ip = $_SERVER['HTTP_X_FORWARDED_FOR'];
   } else {
      $ip = $_SERVER['REMOTE_ADDR'];
   }
   // authentication done and valid, now you need to set the session cookies
   $currentCookieParams = session_get_cookie_params()
   $rootDomain = '.o-money.eu';
   session_set_cookie_params(
      $currentCookieParams["lifetime"],
      $currentCookieParams["path"],
      $rootDomain,
      $currentCookieParams["secure"],
      $currentCookieParams["httpsonly"]
      );
   session_name('openmoneysession');
   session_start();
```

```
        setcookie('token', $response['data']['token'], time() + 4320000, '/', $rootDomain);
        setcookie('secret', $response['data']['secret'], time() + 4320000, '/', $rootDomain);
        setcookie('ip', $ip, time() + 4320000, '/', $rootDomain);
    }
} else {
    echo "<br><p>";
    echo "<form action=getauthlogin.php method=POST>";
    echo "<label>Login</label>";
    echo "<input type=text name=login class=\"w3-input\">";
    echo "<label>Password</label>";
    echo "<input type=password name=password class=\"w3-input\">";
    echo "<br><br>";
    echo "<button class=\"w3-button w3-teal\" w3-display-bottomleft type=submit name=submit
value=\"Submit\">";
    echo "Submit";
    echo "</button>";
    echo "<br><br></p>";
    echo "</form>";
}
?>
```

This template simply provides two boxes in which to enter your login and password

With this code you
are already ready the functionality concerning the user login.

Now you just need to apply a style to the web page compatible with your site.

Same thing goes for the other available templates.

Let's see instead the complete management of the APIs that allow you to build your application from scratch.

Server code

What it is and what we need the servercode for.

The servercode is a unique identifier that identifies a server registered on AUTH.

We must use the servercode every time we want to use a web page present on AUTH and format it with the settings that we entered during the configuration phase.

For example, the login page will have our logo and colors if we pass the servercode in the URL.

Ex: https://auth.ssolo.co.uk/auth/getlogin.php?servercode=yourservercode

Below are the URLs that accept the servercode and the functions performed.

Function	Description	URL
Lost password	Generate a new login password	https://auth.ssolo.co.uk/auth/forgotpwd.php?servercode=yourservercode
Modify user	Modify userdata	https://auth.ssolo.co.uk/auth/moduser.php?servercode=yourservercode
User registration	Register new user	https://auth.ssolo.co.uk/auth/reguser.php?servercode=yourservercode
Delete user	Remove account from AUTH	https://auth.ssolo.co.uk/auth/deleteaccount.php?servercode=yourservercode

Service connection

Each server that need to authenticate their users, need to follow some simple rules.

1. Must enter its SSL public key in the login phase.
2. Must have an fixed IP address
3. Must have a valid SSL certificate for its web server (https enabled)

On the AUTH server there are only the essential data needed to register the client and are :

Column	Type	Nullable	Default
user_id	integer	not null	nextval('users_user_id_seq'::regclass)
login	character varying(30)		The login name
name	character varying(30)	not null	Name
surname	character varying(50)	not null	Surname
email	character varying(80)		Email address of the client
phone	character varying(50)		Landline phone of the client
mobile	character varying(50)		Mobile phone of the client
address_line1	character varying(150)		The first line of the client address
address_line2	character varying(150)		The second line of the client address
zip	character varying(20)		Zip code
city	character varying(40)		City
region	character varying(20)		Region
country	character varying(30)		Country
authtype	character varying(10)		Type of two factors authentication, email or SMS
secret	character varying(40)		Secret word generated at client registration, it is used to encrypt sensitive customer data
usercode	character varying(16)		Code generated at client registration, it is used to retrive user data on external databases
password	character varying(40)		Password choosen from client and saved crypted
lastlogin	timestamp without time zone		The last login data of the client
pubkey	bytea		The SSL pubkey of the application server, it is used to permit the single sign on ov different services.
bgcolor	character varying(8)		The html color code for the external server page presentation
fgcolor	character varying(8)		The fonts color for the external server page presentation

logo	character varying(100)		Iperlink for remote service logo

If your application need more customer data you can implement an external database and use the usercode sent from auth server to identify the client on external database.

For example, you need to store the age, and the date of birth for the client.

You can create a local database as the follow:

Usercode	the usercode sent from AUTH server
Age	number of years
Birth	Date of birth

When you get the customer data from AUTH server, you can integrate its with the data on external database using as key of search the usercode.

In this way you can build your application without restrictions regarding type of data and number of informations needed.

Each time that the customer request a login, if login and password are correct, a second login steps is started and , depending from value on the authtype field, will be send an email or an SMS with a control code that must will be inserted on the login screen.

In this way the AUTH server is sure of the identity of the client.

API functions

The AUTH server make available the following APIs:

- Getuser gives you all the customer data
- Adduser user registration
- Updateuser modify client data
- Deluser delete user
- Lockuser block user access without cancel any data
- Unlockuser unlock blocked users
- Statuser user statistics (number of login, login for day, message sent to user, last login, failed login)
- Login exposed login
- Jwtlogin cryptographic login with JWT

All API work with JSON messages exchange, you can use all programming languages that exists without problems.

The JSON is a standard for communicate for internet servers and its supported from all languages and CRM on the market.

The data paylod is divided in two parts, header and pars.

In the header there is the authentication bearer, the token sent during the login process.

In the pars there are all data the we need to pass at the servers.

The server will answer you with the same mode, one JSON message.

Example of json request:

**curl -X GET https://auth.ssolo.co.uk/auth/api/v1/getuser.php/{userId} **

-H "Authorization: Bearer <your api token>"

Example of response:

{

"id": 101,

"name": "Example Person",

"email": person@example.com,

"firstName": "Example",

"lastName": "Person",

"phoneNumber": "+37111111111",

"city": "London",

“country": "United Kingdom",

"postCode": "11111"

"firstLine": "Road 123 }

All data exchange from AUTH server and the application servers are made in this way.

You send a JSON message to the API server and the API server reply you with a JSON message.

Now we can see all API calls and needed parameters.

API calls

Login function

The login function si not a real API but follow the same rule of the real API.

The only difference is that the login not require the bearer token because is it that furnish it.

The authentication mechanism use the JSON WEB TOKEN (JWT) to initiate the user section and furnish the operation token.

Server requirement

On the server that connect the AUTH server must be have installed the JWT library to exchange crypto data.

Are supported all programming languages.

On the site jwt.io you can find the ideal library for your server.

Create login request

The login procedure can be use two different calls, exposed or not.

The exposed procedure have the login and password sent to the auth server with a POST message.

Ex.

< form method=”POST” action=”https://auth.ssolo.co.uk/auth/login.php”>

Login:<input type=”text” name=”login” >

Password:<input type=”text” name=”password”>

<input type=”submit” name=”submit” value=”Login”>

</form>

The AUTH server receive the login and password, check if are the same registered on database and, if yes, grant the access.

The response of the server is a JWT message with encrypted the user data.

{"data":
{"name":"demo","surname":"test","lastlogin":null,"secret":"123456","token":"YWpMcEY0MWRo
Q3VMd1BRemJwV01MRjBQcFQyNkhzL2hTNUtHN0p0bEdqWT06OqPYAwCcLMNodY2WBltl72Q="
},"iss":"https:\/\/auth.ssolo.co.uk","sub":"2019-10-31 11:33:48"}

The JWT perform the cryptografy using the their SSL private key, the application server, with the public key of the AUTH server can be decrypt the data.

The field token contain the authorization token that is needed to insert in all API call that you need to make.

If you not receive a JSON message the login is not valid and you receive the error:

Invalid login

The unexposed procedure is similar but the login and password are in a JWT message .

Example to call JWT login:

```
1.  <?php
2.  $curlSES=curl_init();
3.  //step2
4.  $pars=array(
5.  "login" => "demo",
6.  "password" => "test"
7.  );
8.  $pars=json_encode($pars);
9.  $headers = array();
10. $headers[] = "authorization: Bearer " .
    "ZEwwRWdYd1F5NU9iTHBtN0J1WlNLRkJNbTA1TkM5UjdLTTI5Wml0YlZEYz06Ogy5ivydHnPBsSHkvc
    A/1o0=";
11. curl_setopt($curlSES,CURLOPT_URL,"https://auth.ssolo.co.uk/auth/jwtlogin.php");
12. curl_setopt($curlSES,CURLOPT_RETURNTRANSFER,true);
13. curl_setopt($curlSES,CURLOPT_HTTPHEADER, $headers);
14. curl_setopt($curlSES, CURLOPT_POST, true);
15. curl_setopt($curlSES, CURLOPT_POSTFIELDS,$pars);
16. curl_setopt($curlSES, CURLOPT_CONNECTTIMEOUT,10);
17. curl_setopt($curlSES, CURLOPT_TIMEOUT,30);
18. $result=curl_exec($curlSES);
19. //step4
20. curl_close($curlSES);
21. ?>
```

On the line 2 we initialize the curl library.

From line 4 to 8 we define the login and password to pass at the AUTH server.

The line 10 is the header of the JSON request that contain the token received with the login procedures.

(`$headers[] = "authorization: Bearer " . "ZEwwRWdYd1F5NU9iTHBtN0J1WlNLRkJNbTA1TkM5UjdLTTI5Wml0YlZEYz06Ogy5ivydHnPBsSHkvcA/1o0="`)

The line 18 execute the API call.

The response at this request can be similar to the next:

eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJkYXRhIjp7Im5hbWUiOm51bGwsInN1cm5hbWUiOm51bGwsImxhc3Rsb2dpbiI6bnVsbCwic2VjcmV0IjpudWxsLCJ0b2tlbiI6IlVqZEtVVXBhVTNkYWJXdENPWGx6YjFwWVJrUnRjM1pUWkc5elkzRkdRVFk0TDNablZYSkZkMDB5TUQwNk90UVk3eUdBc0JVM1NqOXJUMnc2Z0hjPSJ9LCJpc3MiOiJodHRwczpcL1wvYXV0aC5zc29sby5jby51ayIsInN1YiI6IjIwMTktMTAtMzEgMTE6NTg6MzMifQ.Euq4Mu6wl6p0eq27Cmw9CFHEuuDVvyExPcX0tY3JnC0UzxuFofmqT1a-tPV7wQj3OLe4RVq0gAzX8LNnwq9q7TF1EpaJ6g1UwG1tOVF3f6AeTDaCPM4Ga-YBIJh5ney7APcjZ88gBn-hVYLgwqMKJuoo--VnRh18VYdAKFfT0oU

This is a JWT message, you can read the contents decrypting it to the AUTH server SSL public key.

"data" => [

"name" => “demo”,

"surname" => “test”,

"lastlogin" => “2019-07-12”,

"secret" => “KJRERBJHHL”,

"token" =>
“ZEwwRWdYd1F5NU9iTHBtN0J1WlNLRkJNbTA1TkM5UjdLTTI5Wml0YlZEYz06Ogy5ivydHnPBsSHkvcA/1o0="

```
    ],
    "iss" => "https://auth.ssolo.co.uk",
    "sub" => "2019-07-22” );
```

If the credentials are not valid you receive and JSON error message :

```
"data" => [
        "error" => "Invalid login",
        "date" => “2019-07-28,
        "message" => "Please login first"
      ],
      "iss" => "https://auth.ssolo.co.uk",
      "sub" => "2019-07-28",
    );
```

The access token has a 12 hours of live, after this time you need to relogin.

Getuser

The getuser API furnish you the all user data in JSON format.

You need to insert into the request the authorization token and id userid that login procedure gave you.

```
1.  <?php
2.  $curlSES=curl_init();
3.  //step2
4.  $pars=array(
5.  "userid" => "2"
6.  );
7.  $pars=json_encode($pars);
8.  $headers = array();
9.  $headers[] = "authorization: Bearer " .
    "ZEwwRWdYd1F5NU9iTHBtN0J1WlNLRkJNbTA1TkM5UjdLTTI5Wml0YlZEYz06Ogy5ivydHnPBsSHkvc
    A/1o0=";
10. curl_setopt($curlSES,CURLOPT_URL,"https://auth.ssolo.co.uk/auth/api/v1/getuser.php");
11. curl_setopt($curlSES,CURLOPT_RETURNTRANSFER,true);
12. curl_setopt($curlSES,CURLOPT_HTTPHEADER, $headers);
13. curl_setopt($curlSES, CURLOPT_POST, true);
14. curl_setopt($curlSES, CURLOPT_POSTFIELDS,$pars);
15. curl_setopt($curlSES, CURLOPT_CONNECTTIMEOUT,10);
16. curl_setopt($curlSES, CURLOPT_TIMEOUT,30)
17. $result=curl_exec($curlSES);
18. //step4
19. curl_close($curlSES);
20. echo $result;
21. ?>
```

In the line 5 we define the userid value.

In the line 9 we create the header and in line 17 we execute the JSON call.

In the $result you can find all user data:

{"data":
{"login":"demo","name":"demo","surname":"test","email":"test@test.com","phone":"9899887766","mobile":"7542044284","address_line1":"via po
80","address_line2":null,"zip":"00100","city":"roma","region":"RM","country":"Italy","authtype":"jwt","secret":"123456","usercode":"1a1","lastlogin":null},"iss":"https:\/\/auth.ssolo.co.uk","sub":"2019-10-31
13:36:53"}

If the token is expired you receive a follow error message in $result:

{"data":{"error":"Token expired","date":"2019-10-31 13:33:40","message":"Please login
first"},"iss":"https:\/\/auth.ssolo.co.uk","sub":"2019-10-31 13:33:40"}

If the token is not present (user never login) you receive the following error message in $result:

{"data":{"error":"Missing token","date":"2019-10-31 13:33:40","message":"Please login
first"},"iss":"https:\/\/auth.ssolo.co.uk","sub":"2019-10-31 13:33:40"}

You can decode the JSON string, in your applications and obtain an array with all customer data.