Go&Match V1.0 backend specifications

Introduction

Il s'agit ici de spécifier les APIs du backend pour le projet Go&Match.

Rappel de la problématique du projet Go&Match

Le projet Go&Match est un projet d'application Mobile Android/iPhone qui permet à des utilisateurs de matcher leur téléphone entre eux et de récupérer des messages publiés par ces téléphones via une plateforme.

Un message peut être une annonce commerciale, une annonce de particulier, un évènement culturel, et autre… Un message est défini par une catégorie (Art/Culture/Annonce commerciale…), un émetteur, une date de publication, une date d'expiration, un objet, une photo…

Le matching de téléphone s'effectuera par BlueTooth ou WIFI. Un utilisateur est dans la métro. Il lance l'application Go&Match. L'application scanne par Bluetooth et découvre les éventuels autres téléphones à proximité. L'application utilisera alors les adresses Mac des téléphones pour tenter de récupérer les messages correspondants à ce téléphone (et du coup à son utilisateur).

Le principe est le même en WIFI. Je me promène dans la rue. Je passe à côté d'un magasin qui possède sa borne Wifi privée. L'application scanne et découvre l'adresse mac de cette borne wifi et tente de récupérer les messages postés par ce magasin. Si le magasin a posté sur la plateforme des messages de type promotion, vente flash, alors j'en serai informé sur mon téléphone.

En tant qu'utilisateur, je peux récupérer les messages d'un téléphone matché même si celui-ci n'a pas lancer l'application Go&Match. C'est particulièrement vrai pour les bornes Wifi.

Plateforme :

La plateforme (ou backend) est un serveur qui permettra de stocker les données de chaque utilisateur. Les applications Mobile (Android/iPhone) utiliseront une API REST pour se connecter à cette plateforme et en extraire des données.

Modèle de données:

La plateforme devra être capable de gérer les données suivantes :

User:

Un « user » représente un utilisateur de l'application Android/Iphone. On y trouvera les informations suivantes :

Field	**type**	**Description**
Id	String	Unique user id
name	String	User name
token	String	Current token of the user
password	String	User pasword (encrypted)
displayName	String	User display name (by default equals to name)
dateCreation	Date	Date of the user creation
currentDeviceId	String	Id of the device currently used by the user
refreshToken	String	**For future use.** Will be used to refresh the token when it expires

UserListMatched:

Chaque « user » pourra être « matché », c'est à dire lié à un autre ou plusieurs user : La liste des user « matchés » sera défini dans une table comme suit :

Field	type	Description
Id	String	Unique id
UserId	String	User unique Id
UserIdMatched	String	User matched unique Id
dateCreation	Date	Date of the user creation

Device:

Un « device» représente un dispositif (généralement un téléphone, une box wifi..). Un device est forcément lié à un utilisateur et une seul à un moment donné. Un utilisateur peut avoir plusieurs dispositifs mais il en utilisera un seul à un instant donné. Un device peut être partagé par plusieurs utilisateurs

Field	type	Description
Id	String	Unique device id
name	String	device name
displayName	String	Device display name (by default equals to name)
dateCreation	Date	Date of the user creation
address	String	Address of the device (usually MAC address)
type	String	Type of the device (Android, Iphone…)
key	String	Identified the device (firmware version…)
status	int	Status of the device 0 pending, 1 accepted, 2 deleting, 3 deleted,…

Message:

Un message peut être une annonce commerciale, une annonce de particulier, un évènement culturel, et autre… Un Message est défini par une catégorie (Art/Culture/Annonce commerciale…), un émetteur, une date de publication, une date d'expiration, un objet, une photo…

Field	**type**	**Description**
Id	String	Unique message id
from	String	The creator of the message
fromId	String	User id of the creator of the message
dateCreation	Date	Date of the message creation
dateExpiration	Date	Expiration date of the message.
category	String	Category of the message
subject	String	Subject of the message
text	String	Texte of the message
Data1	String	Id of data1 of the message (typically the id of a photo).
deviceId	String	Device ID from where the message has been created
type	String	Type of the message
status	int	Status of the message : 0 pending, 1 deleting, 2 3 deleted, posted…

APIs:

La plateforme devra être capable de fournir des API accessibles par requêtes http. On utilisera les recommandations REST, notamment au niveau des entêtes http.

Par défaut, toutes les requêtes fourniront une réponse au format Json.

Authentification :

Dans cette section, nous aborderons les notions de session d'un utilisateur. Un utilisateur pourra ouvrir une session, la fermer et obtenir les informations d'une session. Chaque utilisateur devra obtenir un token pour pouvoir utiliser les autres APIs du service au cours d'une session. Ce token sera spécifié dans toutes les requêtes via un entête http spécifique. Ce token aura une durée de vie limité et devra le cas échéant être rafraîchit (cela sera implémenté plus tard).

Login (open a session) :

Description
Creates/Opens a session for the user in the system and returns a user access token to the client.
HTTP Method
POST
URI Structure
`/oauth2/sessions`
Authentication
This call requires no authentication.
Request Headers

HeaderName	type	Description
`Accept`	String	`application/json` (default)
`Content-Type`	String	Specifies the content type of the request body. It must be set to `application/x-www-form-urlencoded`

`X-Application-Identifier`	String (Optional)	Identification of the application

Request Parameters

Parameter Name	type	Description
`name`	String	Specifies the username of the user's account.
`password`	String	Specifies the password for the user's account.
refreshToken	*String*	***TBD later.*** *Will be used to refresh the current token*

Response Codes

Successful responses return the following status codes:
`200`
Failed operations return the following status codes:
`400 401 406 500 502 503 504`

Response Document

User entity

Field	type	Description
Id	String	Unique user id
name	String	User name
password	String	User pasword (encrypted)
displayName	String	User display name (by default equals to name)
dateCreation	Date	Date of the user creation
currentDeviceId	String	Id of the device currently used by the user
currentDeviceAdr	String	Address of the device currently used by the user

token	String	The token to be used for the other futur request
refreshToken	String	The refreshToken used to refresh the user's token when it has expired

Sample Request and Response

Example 1. Request

```
POST http://api.example.com/oauth2/sessions
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Accept: application/json
X-Application-Identifier: GoMatch 1.0
Content-Length: 80

name%3Dnicolas.sabin%40libertysurf.fr%26password%3Dtoto%0A
```

Example 2. Response

```
200 OK
Content-Type: application/json
Content-Length: 471
Date: Wed, 29 Aug 2012 11:20:20 GMT

{

     "token":"AEjkHMfHn7WAVJwmSeDu7woCQVQyjkRuC0QllxIdgKuxF8xT
     LZmjDWX6TD8IMt6T5W7DKLV71_iMVTani5L7y_74MQagMU62n1QNGAc9i
     lGyT9jHEMx8_53ddVV4pG-
     raNWaF3cuf1rUovBtK_PraxgLP2qqiyvbdYROqPBGNcgTdZ-qh1nYiS-
     W1uoIqRjBzbdK7GGSocvtvGRZXTmggPjeNDbi0t8FmCt-k9rwOWag"
     ,"id":"fa721efa36a74ba68b86e769ae19d52a"
    ,"accountType":"0"
    ,"name":"nicolas.sabin@libertysurf.fr"
     ,"displayName":"nicolas.sabin@libertysurf.fr"
     ,"currentDeviceId":"2"
     ,"currentDeviceAdr":"00x00:00:58"
     ,"dateCreation":"017-06-08T13:54:06.630Z"
     "refreshToken":"AEjkHMfHn7WAVJwmSeDu7woCQVQyjkRuC0QllxIdg
     KuxF8xTLZmjDWX6TD8IMt6T5W7DKLV71_iMVTani5L7y_74MQagMU62n1
     QNGAc9ilGyT9jHEMx8_53ddVV4pG"
```

}

Session info:

Description
Returns active session details for a given token and a given user.
HTTP Method
GET
URI Structure
`/oauth2/sessions/{userId}`
Authentication
This call requires no authentication.
Request Headers

HeaderName	type	Description
`Accept`	String	`application/json` (default)

Request Parameters

Parameter Name	type	Description
token	String	The user's access token

Response Codes
Successful responses return the following status codes:
`200`
Failed operations return the following status codes:
`400 401 406 500 502 503 504`

Response Document

User entity

Field	type	Description
Id	String	Unique user id

name	String	User name
password	String	User pasword (encrypted)
displayName	String	User display name (by default equals to name)
dateCreation	Date	Date of the user creation
currentDeviceId	String	Id of the device currently used by the user
currentDeviceAdr	String	Address of the device currently used by the user
token	String	The token to be used for the other futur request
refreshToken	String	The refreshToken used to refresh the user's token when it has expired

Sample Request and Response

```
GET
http://api.example.com/oauth2/sessions/2e9c8f2af42e46239bd3f3c
03ee36d2f?token=AJ_Res7qNaJh3wAbV5soMDXAPF7QwnOexO06NOqUS9MSyM
6LmL
Connection: keep-alive
Accept: application/json
```

Example 4. Response

```
200 OK
Server: Apache-Coyote/1.1
WWW-Authenticate: WRAP
Content-Type: application/json
Content-Length: 363
Date: Mon, 27 Aug 2012 13:52:13 GMT

{

    "token":"AEjkHMfHn7WAVJwmSeDu7woCQVQyjkRuC0QllxIdgKuxF8xT
    LZmjDWX6TD8IMt6T5W7DKLV71_iMVTani5L7y_74MQagMU62n1QNGAc9i
    lGyT9jHEMx8_53ddVV4pG-
    raNWaF3cuf1rUovBtK_PraxgLP2qqiyvbdYROqPBGNcgTdZ-qh1nYiS-
    W1uoIqRjBzbdK7GGSocvtvGRZXTmggPjeNDbi0t8FmCt-k9rwOWag"
    ,"id":"fa721efa36a74ba68b86e769ae19d52a"
   ,"accountType":"0"
    ,"name":"nicolas.sabin@libertysurf.fr"
    ,"displayName":"nicolas.sabin@libertysurf.fr"
```

```
      ,"currentDeviceId":"2"
      ,"currentDeviceAdr":"00x00:00:58"
      ,"dateCreation":"017-06-08T13:54:06.630Z"
      "refreshToken":"AEjkHMfHn7WAVJwmSeDu7woCQVQyjkRuC0QllxIdg
      KuxF8xTLZmjDWX6TD8IMt6T5W7DKLV71_iMVTani5L7y_74MQagMU62n1
      QNGAc9ilGyT9jHEMx8_53ddVV4pG"

}
```


Logout (delete Session):

Description
Deletes a user session.
HTTP Method
DELETE
URI Structure
`/oauth2/sessions/{userId}`
Authentication
This call requires no authentication.
Request Headers

HeaderName	type	Description
`Accept`	String	`application/json` (default)

Request Parameters

Parameter Name	type	Description
token	String	The user's access token

Response Codes
Successful responses return the following status codes:
`200`
Failed operations return the following status codes:
`400 401 406 500 502 503 504`

Sample Request and Response

```
DELETE
http://api.example.com/oauth2/sessions/3dc671c0104e4c2794a6e06
057d836fd?token=ADtlQ5HV5xGLkrP-
6itUgA6jSKSwV0mLD7bx8JI76pxOGiSElbConnection: keep-alive
Accept: application/json
```

Example 4. Response

```
204 No Content
```

Device:

Dans cette section, nous aborderons la notion de device. Un device représente un dispositif physique utilisé par un utilisateur à un instant donné. Un device peut donc être un téléphone, une borne wifi…

Un utilisateur connecté au serveur peut créer un device et récupérer les infos de son device pour éventuellement le mettre à jour.

Pas de notion de suppression pour l'instant.

Create/Update:

Description
Creates or update a device (if it already exists in the system) for the user in the system and returns a device id. This device will become the current device for the user.
HTTP Method
POST
URI Structure
`/{userId}/device`
Authentication
This call requires authentication.

Request Headers

HeaderName	type	Description
`Accept`	String	`application/json` (default)

`Content-Type`	String	Specifies the content type of the request body. It must be set to `application/x-www-form-urlencoded`
`Authorization`	String	User token
`X-Application-Identifier`	String (Optional)	Identification of the application

Request Body Format

Calls to this API must include an object of type Device in the request body.

Request Parameters

Parameter Name	type	Description
name	String	device name
displayName	String	Device display name (by default equals to name)
address	String	Address of the device (usually MAC address)
type	String	Type of the device (Android, Iphone…)
key	String	Identified the device (firmware version…)

Response Codes

Successful responses return the following status codes:
`200`
Failed operations return the following status codes:
`400 401 406 500 502 503 504`

Response Document

Id	String	Unique device id
name	String	device name

displayName	String	Device display name (by default equals to name)
dateCreation	Date	Date of the user creation
address	String	Address of the device (usually MAC address)
type	String	Type of the device (Android, Iphone…)
key	String	Identified the device (firmware version…)
status	int	Status of the device 0 pending, 1 accepted, 2 deleting, 3 deleted 4 active,…

Sample Request and Response

Example 1. Request

```
POST http://api.example.com/2e9c8f2af42e46239bd3f3c03ee36d2f
/device
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Accept: application/json
Authorization:
token=Res7qNaJh3wAbV5soMDXAPF7QwnOexO06NOqUS9MSyM6LmL
X-Application-Identifier: GoMatch 1.0
Content-Length: 80

name=Nexus&displayName=Nexus&address="0x00:05:07"&type=android
&key="google_nexus_android_7_0"

name%3DNexus%26displayName%3DNexus%26address%3D%220x00%3A05
%3A07%22%26type%3Dandroid%26key%3D%22google_nexus_android_7_0
%22%0A%0A
```

Example 2. Response

```
200 OK
Content-Type: application/json
Content-Length: 471
Date: Wed, 29 Aug 2012 11:20:20 GMT

{
     "id":" de727efa36a74ba68b88e769ae19d52a"
     ,"creation_date":"017-06-08T13:54:06.630Z"
    ,"name":"Nexus"
    ,"displayName":"Nexus"
```

```
      ,"address":"00x00:00:58"
       ,"type":"Android"
       ,"key":"google_nexus_android_7_0"
       ,"status":"2"

}
```


Device info:

Description
Returns active the devices for a given token and a given user.
HTTP Method
GET
URI Structure
/{userId}/device
Authentication
This call requires authentication.
Request Headers

HeaderName	type	Description
Accept	String	application/json (default)
Content-Type	String	Specifies the content type of the request body. It must be set to application/x-www-form-urlencoded
Authorization	String	User token

Request Parameters

Parameter Name	type	Description

Response Codes
Successful responses return the following status codes:
200
Failed operations return the following status codes:
400 401 406 500 502 503 504

Response Document

Id	String	Unique device id
name	String	device name
displayName	String	Device display name (by default equals to name)
dateCreation	Date	Date of the user creation
address	String	Address of the device (usually MAC address)
type	String	Type of the device (Android, Iphone…)
key	String	Identified the device (firmware version…)
status	int	Status of the device 0 pending, 1 accepted, 2 deleting, 3 deleted, 4 active…

Sample Request and Response

```
GET http://api.example.com/2e9c8f2af42e46239bd3f3c03ee36d2f
/device
Connection: keep-alive
Accept: application/json
Authorization:
token=Res7qNaJh3wAbV5soMDXAPF7QwnOexO06NOqUS9MSyM6LmL
```

Example 4. Response

```
200 OK
Server: Apache-Coyote/1.1
WWW-Authenticate: WRAP
Content-Type: application/json
Content-Length: 363
Date: Mon, 27 Aug 2012 13:52:13 GMT

Can be a list of device entities

{
     "id":" de727efa36a74ba68b88e769ae19d52a"
     ,"creation_date":"017-06-08T13:54:06.630Z"
```

```
      ,"name":"Nexus"
      ,"displayName":"Nexus"
      ,"address":"00x00:00:58"
       ,"type":"Android"
       ,"key":"google_nexus_android_7_0"
       ,"status":4"

}
```

Delete Device:

Will be done later

Description
Deletes a device.
HTTP Method
DELETE
URI Structure
/{userId}/device
Authentication
This call requires authentication.
Request Headers

HeaderName	type	Description
`Accept`	String	`application/json` (default)
`Authorization`	String	User token

Request Parameters

Parameter Name	type	Description
id	String (optionnal)	device id to delete, if not specified, delete all message

Response Codes
Successful responses return the following status codes:
`200`
Failed operations return the following status codes:

400 401 406 500 502 503 504

Sample Request and Response

```
DELETE
http://api.example.com/3dc671c0104e4c2794a6e06057d836fd/mesage
?id=ADtlQ5HV5xGLkrP-6itUgA6jSKSwV0mLD7bx8JI76pxOGiSElb
Connection: keep-alive
Accept: application/json
Authorization:
token=Res7qNaJh3wAbV5soMDXAPF7QwnOexO06NOqUS9MSyM6LmL
```

Example 4. Response

```
204 No Content
```

Messages:

Dans cette section, nous aborderons la notion de messages. On a vu qu'un utilisateur pouvait poster un message sur la plate-forme. Une liste de message créé par un utilisateur ou depuis un device pourra être récupérer. Enfin un message pourra être supprimé sur la platforme.

Create message:

Description
Creates a message for the user in the system and returns a message id.
HTTP Method
POST
URI Structure
`/{userId}/message`
Authentication
This call requires authentication.
Request Headers

HeaderName	type	Description
`Accept`	String	`application/json` (default)

`Content-Type`	String	Specifies the content type of the request body. It must be set to `application/x-www-form-urlencoded`
`Authorization`	String	User token
`X-Application-Identifier`	String (Optional)	Identification of the application

Request Body Format

Calls to this API must include an object of type Message in the request body.

Request Parameters

Field	**type**	**Description**
from	String	The creator of the message
dateExpiration	Date (optional)	Expiration date of the message.
category	String (optional)	Category of the message
subject	String	Subject of the message
text	String	Text of the message **Will be sent as body in future version**
data1	String (optional)	Id of data1 of the message (typically the id of a photo).
deviceId	String	Device ID from where the message has been created
type	String	Type of the message
status	int	Status of the message : 0 pending, 1 deleting, 2 3 deleted, posted…

Response Codes

Successful responses return the following status codes:

`200`

Failed operations return the following status codes:
400 401 406 500 502 503 504

Response Document

Field	type	Description
from	String	The creator of the message
fromId	String	User if of the creator of the message
dateCreation	Date	Date of the message creation
dateExpiration	Date	Expiration date of the message.
category	String	Category of the message
subject	String	Subject of the message
text	String	Texte of the message
Data1	String	Id of data1 of the message (typically the id of a photo).
deviceId	String	Device ID from where the message has been created
type	String	Type of the message
status	int	Status of the message : 0 pending, 1 deleting, 2 deleted, 3 posted…

Sample Request and Response

Example 1. Request

```
POST http://api.example.com/2e9c8f2af42e46239bd3f3c03ee36d2f
/message
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Accept: application/json
Authorization:
token=Res7qNaJh3wAbV5soMDXAPF7QwnOexO06NOqUS9MSyM6LmL
X-Application-Identifier: GoMatch 1.0
Content-Length: 80

from=nicolas.sabin@libertysurf.fr&subject=Hello&text="this is
```

```
the text""&dateExpiration="017-06-
08T13:54:06.630Z"&category="Events"
```

from%3Dnicolas.sabin%40libertysurf.fr%26subject%3DHello%26text%3D%22this%20is%20the%20text%22%22%26dateExpiration%3D%22017-06-08T13%3A54%3A06.630Z%22%26category%3D%22Events%22%0A

Example 2. Response

```
200 OK
Content-Type: application/json
Content-Length: 471
Date: Wed, 29 Aug 2012 11:20:20 GMT

{
     "id":" de727efa36a74ba68b88e769ae19d52a"
     ,"dateCreation":"017-06-08T13:54:06.630Z"
     ,"dateExpiration":"017-06-08T13:54:06.630Z"
     "fromId":"2e9c8f2af42e46239bd3f3c03ee36d2f "
     "from":"Nicolas.sabin@libertysurf.fr"
     "subject":"hello"
     "text":"this is the text"
     "data1":"file:http//inderdev/hook/2e9c8f2af42e46239bd3f3c0
     3ee36d2f/toto.jpg"
     "deviceId":"00x00:00:58"
     "type":"message"
     "category":"Events"
     "status":"3"

}
```

messagelist:

Description
Returns a message list for a given token and a given user or a device address
HTTP Method
GET
URI Structure
/{userId}/message
Authentication
This call requires authentication.

Request Headers

HeaderName	type	Description
`Accept`	String	`application/json` (default)
`Content-Type`	String	Specifies the content type of the request body. It must be set to `application/x-www-form-urlencoded`
`Authorization`	String	User token

Request Parameters

Parameter Name	type	Description
eTag	String	etag
fromId	String (optional)	List of User id to retrieve the list of the message created by this user.
`deviceId`	*String (optional)*	*List of Device id to retrieve the list of the message created from this device.* **Will be done later**

In case of the friomId and deviceId are not present, then the backend will return all the new mesages to the users mctahed with the current one.

Response Codes

Successful responses return the following status codes:
`200`
Failed operations return the following status codes:
`400 401 406 500 502 503 504`

Sample Request and Response

```
GET http://api.example.com/2e9c8f2af42e46239bd3f3c03ee36d2f
/device
Connection: keep-alive
Accept: application/json
```

```
Authorization:
token=Res7qNaJh3wAbV5soMDXAPF7QwnOexO06NOqUS9MSyM6LmL
```

Example 4. Response

```
200 OK
Server: Apache-Coyote/1.1
WWW-Authenticate: WRAP
Content-Type: application/json
Content-Length: 363
Date: Mon, 27 Aug 2012 13:52:13 GMT

List of Json messages entities

{
     "id":" de727efa36a74ba68b88e769ae19d52a"
     ,"dateCreation":"017-06-08T13:54:06.630Z"
     ,"dateExpiration":"017-06-08T13:54:06.630Z"
     "fromId":"2e9c8f2af42e46239bd3f3c03ee36d2f "
     "from":"Nicolas.sabin@libertysurf.fr"
     "subject":"hello"
     "text":"this is the text"
     "data1":"file:http//inderdev/hook/2e9c8f2af42e46239bd3f3c0
     3ee36d2f/toto.jpg"
     "deviceId":"00x00:00:58"
     "type":"message"
     "category":"Events"
     "status":"3"

}
```

Delete Message:

Description
Deletes a message.
HTTP Method
DELETE
URI Structure
/{userId}/message
Authentication
This call requires authentication.

Request Headers

HeaderName	type	Description
`Accept`	String	`application/json` (default)
`Authorization`	String	User token

Request Parameters

Parameter Name	type	Description
id	String (optionnal)	Message id to delete, if not specified, delete all message from the user

Response Codes

Successful responses return the following status codes:
`200`
Failed operations return the following status codes:
`400 401 406 500 502 503 504`

Sample Request and Response

```
DELETE
http://api.example.com/3dc671c0104e4c2794a6e06057d836fd/mesage
?id=ADtlQ5HV5xGLkrP-6itUgA6jSKSwV0mLD7bx8JI76pxOGiSElb
Connection: keep-alive
Accept: application/json
Authorization:
token=Res7qNaJh3wAbV5soMDXAPF7QwnOexO06NOqUS9MSyM6LmL
```

Example 4. Response

```
204 No Content
```

UserMatchedList:

Dans cette section, nous aborderons la notion de « user matché ». Un user pourra être matché avec un ou plusieurs autres user. Ce matching permettra de faire le lien entre les users. Par exemple obtenir les nouveaux messages

destinés à un utilisateur. Pour cela, le backend établiera la liste des users qui matchent avec cet utilisateur, et pour chacun d'entre eux les nouveaux messages postés.

Match a user:

Will be done later.

Description
Match an user with the current.
HTTP Method
POST
URI Structure
`/{userId}/usermatch`
Authentication
This call requires authentication.

Request Headers

HeaderName	type	Description
`Accept`	String	`application/json` (default)
`Content-Type`	String	Specifies the content type of the request body. It must be set to `application/x-www-form-urlencoded`
`Authorization`	String	User token
`X-Application-Identifier`	String (Optional)	Identification of the application

Request Body Format

Calls to this API must include an object of type User matching in the request body.

Request Parameters

Parameter Name	type	Description
matchedDeviceId	String	Address of the device (usually MAC address) (optional)

matchedUserId	String	userId of the user to be matched

Response Codes

Successful responses return the following status codes:
200
Failed operations return the following status codes:
400 401 406 500 502 503 504

Response Document

Return the new user entity matched with the current user

Sample Request and Response

Example 1. Request

```
POST http://api.example.com/2e9c8f2af42e46239bd3f3c03ee36d2f
/device
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Accept: application/json
Authorization:
token=Res7qNaJh3wAbV5soMDXAPF7QwnOexO06NOqUS9MSyM6LmL
X-Application-Identifier: GoMatch 1.0
Content-Length: 80

name= address="0x00:05:07"
```

Example 2. Response

```
200 OK
Content-Type: application/json
Content-Length: 471
Date: Wed, 29 Aug 2012 11:20:20 GMT

Json user entity (without token/refreshToken)

{

     ,"id":"fa721efa36a74ba68b86e769ae19d52a"
    ,"accountType":"0"
    ,"name":"Michel.Dupont"
```

```
      ,"displayName":" Michel Dupont"
      ,"currentDeviceId":"00x00:00:58"
      ,"deviceId":"00x00:00:58"
      ,"dateCreation":"017-06-08T13:54:06.630Z"
}
```

userMatchedList:

Description
Creates or update a device (if it already exists in the system) for the user in the system and returns a device id. This device will become the current device for the user.
HTTP Method
GET
URI Structure
/{userId}/usermatch
Authentication
This call requires authentication.

Request Headers

HeaderName	type	Description
Accept	String	application/json (default)
Content-Type	String	Specifies the content type of the request body. It must be set to application/x-www-form-urlencoded
Authorization	String	User token
X-Application-Identifier	String (Optional)	Identification of the application

Request Body Format

Calls to this API must include an object of type Device in the request body.

Response Codes
Successful responses return the following status codes:
200
Failed operations return the following status codes:
400 401 406 500 502 503 504

Response Document

Return the list of the users entities matched with the current user

Sample Request and Response
Example 1. Request

```
POST http://api.example.com/2e9c8f2af42e46239bd3f3c03ee36d2f
/device
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Accept: application/json
Authorization:
token=Res7qNaJh3wAbV5soMDXAPF7QwnOexO06NOqUS9MSyM6LmL
X-Application-Identifier: GoMatch 1.0
Content-Length: 80

name= address="0x00:05:07"
```

Example 2. Response

```
200 OK
Content-Type: application/json
Content-Length: 471
Date: Wed, 29 Aug 2012 11:20:20 GMT

List of Json user entities

{

     ,"id":"fa721efa36a74ba68b86e769ae19d52a"
    ,"accountType":"0"
     ,"name":"Michel.Dupont"
     ,"displayName":" Michel Dupont"
     ,"currentDeviceId":"00x00:00:58"
     ,"deviceId":"00x00:00:58"
```

```
      ,"dateCreation":"017-06-08T13:54:06.630Z"
}…
```


unmacth a user:

wil be done later

Description
unmatch an user with the current.
HTTP Method
DELETE
URI Structure
/{userId}/usermatch
Authentication
This call requires authentication.

Request Headers

HeaderName	type	Description
Accept	String	application/json (default)
Content-Type	String	Specifies the content type of the request body. It must be set to application/x-www-form-urlencoded
Authorization	String	User token
X-Application-Identifier	String (Optional)	Identification of the application

Request Body Format

Calls to this API must include an object of type User matching in the request body.

Request Parameters

Parameter Name	type	Description

address	String	Address of the device (usually MAC address) (optional)
userId	String	userId of the user to be macthed

Response Codes

Successful responses return the following status codes:

200

Failed operations return the following status codes:

400 401 406 500 502 503 504

Response Document

None

Sample Request and Response

Example 1. Request

```
POST http://api.example.com/2e9c8f2af42e46239bd3f3c03ee36d2f
/device
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Accept: application/json
Authorization:
token=Res7qNaJh3wAbV5soMDXAPF7QwnOexO06NOqUS9MSyM6LmL
X-Application-Identifier: GoMatch 1.0
Content-Length: 80

name= address="0x00:05:07"
```

Example 2. Response

```
200 OK
Content-Type: application/json
Content-Length: 471
Date: Wed, 29 Aug 2012 11:20:20 GMT
```