API AREA

Preface

This Project is made by Benjamin Piniac, Marc-Antoine Wittling, Mathieu Desrues, Stanislas Hégron, Jean Bougarel and Mikael Gerard, for Epitech.

Introduction

The AREA is a simple dashboard with services. Each services have actions or reactions. Some of them will need OAuth2. It is build with docker. The purpose of the project is to creating AREA :

- Action : A predefined event triggering a reaction (like a new post from someone on reddit)
- REAction : Predefined event supposed to be trigger by a action (like play a song on Spotify)

The project is compose of :

- **Web client** : call the api and display informations, run on port 8081
- **API / Server:** manage database and call externe api and send information to the client, running on port 8080
- **Mobile client :** available on localhost:8081/client.apk
- **Mongo data base:** A data base by mongodb, available on the port 27017

API Routes

The following commands explain there ows use.
If nothing is specify, the content-type of the body must by in “application/x-www-form-urlencoded”
If any error occurred, the api will send the next json with a explicit message :

```
{
 success : False
 message : "An explicit message"
}
```

Connect - POST

/connectUser

Body :

username : username of the user
password : password of the user

Connect a user.
return exemple :

```
{
 success : True
 message : "User connected"
 token : "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...t2ot3Urfre4egTFGEE"
}
```

Status Success : 201

Create user - POST
/createUser

Body :

username : username of the user
password : password of the user

Connect a user.
return exemple :
```
{
 success : True
 message : "User created"
 token : "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...t2ot3Urfre4egTFGEE"
}
```
Status Success : 201

All next request need a specific header :

Authorization : 'token {token received on the connection**}'**

Get areas - GET
/GetAreas

Get all AREA of the user.
return exemple :
```
[
   {
       "area_id": "405gznvk760feck",
       "user_id": "405gznibk75yvix7",
       "area_name": "Raccoon area",
       "color": "#E3FF33",
       "action": {
           "name": "github_issue_event",
           "params": [
               {
                   "name": "repository",
                   "value": "Test"
               }
           ],
           "webhook_id": 187630834
       },
       "reaction": {
           "name": "slack_send_message",
           "params": [
               {
                   "name": "channel_id",
                   "value": "CTTUL2J4C"
               }
           ]
       }
   },
   ...
]
```
Status Success : 201

Create area - POST

/CreateArea

Create a AREA.
The content-type of the body must be in “application/json”

Body exemple :

```
{
        "area_name":"My first reaction",
        “color":"Yellow",
        "action" :{
                "name":"github_issue_event",
                "params":[
                        {
                                "name":"repository",
                                "value": "Test"
                        }
                ]
        },
        "reaction":{
                "name": "github_create_board",
                "params":[
                        {
                                "name" : "owner",
                                "value": "mikskyzo"
                        }
                ]
        }
}
```

return exemple :

```
{
 success : True
 message : "Area created successfully"
}
```

Status Success : 201

Get area’s name - GET

/GetAreas/Name

Get AREA’s name and id of the area.

return exemple :

```
   [
        {
            area_id: "405ope7bk5mpo3hi",
            action: “Github",
            reaction: "Discord",
            area_name : “Area racoon”,
            color : “Yellow”
        },
        ...
    ]
```

Status Success : 201

Get area - GET
/GetArea/{AreaId}

Get the AREA from AreaId of the user.

return exemple :

```
{
   "area_id": "405gznvk760feck",
   "user_id": "405gznibk75yvix7",
   "area_name": "Raccoon area",
   "color": "#E3FF33",
   "action": {
      "name": "github_issue_event",
      "params": [
          {
              "name": "repository",
              "value": "Test"
          }
      ],
      "webhook_id": 187630834
   },
   "reaction": {
       "name": "slack_send_message",
       "params": [
          {
              "name": "channel_id",
              "value": "CTTUL2J4C"
          }
      ]
   }
}
```

Status Success : 201

Delete area - Delete
/DeleteArea

Body :

area_id : Id of the area

Delete a AREA.

return exemple :

```
{
 success : True
 message : "Area deleted"
}
```

Status Success : 201

Get services - GET

/getServices

Get the services the user is connected to.

return exemple :

```
[
   {
       "service": "Discord",
       "active": true
   },
   {
       "service": "Github",
       "active": false
   },
  ...
]
```

Status Success :   201

Get all actions / reactions - GET

/getActions

/getReactions

Get all the actions available for the user.
The same operation exist for the reactions.

return exemple :

```
[
    {
       name : "github_new_push",
       service : "Github",
       title : "Repository push",
       description : "Trigger when someone push on a repo",
       params : [
              {
               name : "repository",
               description : "Name of the repository"
              },
              {
               name : "username",
               description : "Owner's name of the repository"
              }
         ],
       ...
     }
       ...
 ]
```

Status Success :   201

Change username - PATCH
/user/changeUsername

Body :

username : The new username of the user

Change the username of the actual user.
return exemple :
```
{
 success : True
 message : "Username changed"
}
```
Status Success : 201

Change password - PATCH
/user/changePassword

Body :

new_password: The new password of the user
password: The actual password of the user

Change the password of the actual user.
return exemple :
```
{
 success : True
 message : "Password change changed"
}
```
Status Success : 201

Auth - Redirection
/Auth/connect/:service?token={access_token}

service : The service you want to connect to
acess_token : Token get by the connection

Change the password of the actual user.
return → Redirect to the authentication system of the service

Disconnect service - Delete
/auth/delete/:service

:service : name of the service to disconnected from

Delete a service from a account.
Status Success : 200