Documentation API

M. Brunelle, H. Defrance, G. Derivery,
G. Feraud, V. Meilhac, A. Pasquini,
S. Rose, H. Xie

Tableau des révisions

Date	Auteur	Section	Commentaire
06/08/2013	Hugo Defrance	Toutes	Ajout des titres et présentation générale.
07/08/2013	Hugo Defrance	Toutes	Détail des requêtes.
07/11/2013	Hugo Defrance	Toutes	Remise en page, correction et mise à jour de toutes les parties.
11/11/2013	Hugo Defrance	Listes de lecture	Ajout de la section listes de lecture
12/11/2013	Hugo Defrance	Musique	Changement du type de la variable time d'une musique.
14/11/2013	Hugo Defrance	Toutes	Ajout de méthode de suppression en masse en POST pour compatibilité avec iOS.
24/11/2013	Hugo Defrance	Listes d'amis	Ajout du username de l'ami dans le GET + détail du type de retour de la requête.
28/11/2013	Hugo Defrance	Liste de lecture	Possibilité d'ajouter une nouvelle musique à une liste de lecture (création auto de la musique puis ajout à la liste)
05/12/2013	Hugo Defrance	Listes d'amis	Modification de la méthode de suppression d'un ami pour pouvoir en supprimer plusieurs à la fois + compatibilité iOS.
12/12/2013	Hugo Defrance	Utilisateur – Messagerie instantanée	Le username doit maintenant être unique + complément de la documentation de la messagerie.
03/01/2014	Hugo Defrance	Utilisateur – tracks – messagerie instantanée	Ajout d'une méthode pour réinitialiser un mot de passe oublié. + suppression des attributs created_at, updated_at, deleted_at inutiles + complétion de la documentation de la messagerie instantanée.

Table des matières

Présentation de l'API

Tous les clients du projet (Android, IPhone, application PC/MAC et site web) communiqueront avec le serveur principal grâce à une API de type REST. Celle-ci permettra notamment d'accéder à la base de données principale du projet, sera chargée de l'authentification des utilisateurs et des fonctionnalités sociales.
Cette API utilisera le protocole HTTPS afin de garantir un maximum de flexibilité et de sécurité dans son utilisation. En effet, la majorité des langages de programmation moderne (tel que Java ou Ruby ou Python par exemple) implémentent ce protocole nativement, ce qui permet de faire des requêtes de manière rapide et sécurisé.

Requêtes disponibles

Toutes les requêtes doivent être encodées en UTF-8 et doivent contenir un « Content- Type » de type « application/json ». Les paramètres des requêtes doivent être formatés dans un format JSON valide.

Afin d'authentifier les utilisateurs, toutes les requêtes (sauf la requêtes de création de compte, de connexion et de récupération de mot de passe) devront contenir le paramètre GET « token » contenant le token. Les requêtes auront donc le format :
« /<request>?token=<token> ».

Le serveur répondra à toutes les requêtes en utilisant les codes de réponse HTTP standard et contiendront éventuellement des données encodées en JSON.

1. Utilisateur

1. S'enregistrer – création d'un compte utilisateur et d'un token

Cette requête sert à enregistrer un nouvel utilisateur. Une fois son compte créé, l'utilisateur est connecté automatiquement.

Requête

POST /api/users

Tous les paramètres sauf first_name et last_name sont obligatoires. Le username ne peut contenir sera validé grâce à la regex /\A[a-zA-Z0-9]+\Z/ et l'adresse email sera validée conformément à la RFC 5322.

```
{
 "user":
        {
          "email" : "<user_email>",
          "password" : "<user_password>",
          "username" : "<user_username >",
          "first_name" : "<user_first_name >",
           "last_name" : "<user_last_name >",
         }
}
```

Réponse

Si la requête a été exécutée avec succès, une réponse de type « 201 Created » sera retournée contenant le token. Voici un exemple de réponse :

```
{
  "success" : true,
   "token" : "<token>",
   "id" : "<user id>",

}
```

Erreurs

En cas d'erreurs, une réponse de type « 403 Forbidden » sera retournée. Le champ error contient uniquement la première erreur détectée. Si plusieurs erreurs sont présentes, elles seront affichées une fois la première corrigée. Voici un exemple de réponse d'erreur :

```
{
  "success": false,
  "error": <first error string>
}
```

2. Connexion – création de token

Cette requête permet de générer un token pour un compte déjà existant.

Requête

POST /api/users/login

```
{
  "email" : "<user_email>",
  "password" : "<user_password>"
}
```

Réponse

Si la requête a été exécutée avec succès, une réponse de type « 201 Created » sera retournée contenant le token. Voici à quoi ressemble la réponse :

```
{
  "success" : true,
   "token" : "<token>"
}
```

Erreurs

En cas d'erreurs, une réponse de type « 403 Forbidden » sera retournée. Voici le format de la réponse :

```
{
  "success" : false,
  "error" : "Invalid email or password."
}
```

3. Déconnexion – destruction de token

Cette requête permet de détruire un token.

Requête

DELETE /api/users/logout

Réponse

Si le token est valide, une réponse de type « 200 OK » sera retournée :

```
{
  "success" : true,
  "message" : "Token deleted"
}
```

Erreurs

Si le token est invalide, une réponse de type « 404 Not Found » sera retournée :

```
 {
   "success" : false,
   "message" : "Invalid token."
}
```

4. Mise à jour des informations d'un compte utilisateur

Cette requête permet de mettre à jour les informations relatives à un compte utilisateur, comme le mot de passe par exemple. Il est possible mettre à jour tous les champs de l'objet User (voir création de compte pour une liste exhaustive) mis à part le username. Dans l'exemple ci-dessous, le mot de passe est mis à jour.

Requête

PUT /api/users

```
{
   "user" :
            {
               "password" : "<nouveau mot de passe>"
            }
}
```

Réponse

Si les paramètres sont valides, une réponse de type « 201 Created » sera retournée :

```
{
  "success" : true,
  "message" : "User has been updated"
}
```

Erreurs

Si le token est invalide, une réponse de type « 403 Forbidden » sera retournée :

```
{
  "success" : false,
  "message" : "Token is invalid"
}
```

Si le username est mis à jour, une réponse de type « 400 Bad Request » sera retournée :

```
{
  "success" : false,
  "message" : " Updating username is forbidden"
}
```

5. Récupération de mot de passe oublié

Cette requête permet de réinitialiser un mot de passe oublié par un envoi de mail de confirmation.

Requête

POST /api/users/reset-password-link

Cette requête prend soit une adresse email, soit un username en paramètre et ne nécessite pas de token.

{
"email":<email de l'utilisateur>
}

OU

{
"username":<username de l'utilisateur>
}

Réponse

Si l'adresse email ou le username correspondent à un utilisateur valide, un email sera envoyé à l'utilisateur avec la suite de la procédure de récupération. Une réponse de type « 200 OK » sera également retournée par l'API :

{
success: true
message: "You will receive an email with the password reset procedure."
}

Erreurs

Si l'adresse email ou le username ne correspondent à aucun utilisateur, une réponse de type « 404 Not Found » sera retournée :

{
success: false
error: "User not found"
}

6. Obtenir des informations sur un utilisateur

Cette requête permet d'obtenir les informations d'un utilisateur.

Requête

GET /api/users/<user_id>
GET /api/users/me

La seconde requête est un raccourci permettant d'accéder directement au profil de l'utilisateur connecté. Le format de la réponse du serveur est identique pour les deux requêtes.

Réponse

Si le token est valide et l'id de l'utilisateur existe, une réponse de type « 200 OK » sera retournée. Voici un exemple de réponse :

```
{
  "success" : true,
  "user": {
            "email" : <user_email>,
            "email_verified": <boolean>
            "id" : <user_id>,
            "first_name" : "<user_first_name>",
            "last_name" : "<user_last_name>",
            "username" : "<user_username>"
          }
}
```

Erreurs

Si le token est manquant ou invalide, une réponse de type « 403 Forbidden » sera retournée (ci-dessous, le token est manquant) :

```
{
  success : false,
  error : "token field is missing"
}
```

Si l'utilisateur n'existe pas, une réponse de type « 404 Not Found » sera retournée :

```
{
  "success" : false,
  "error" : "User does not exist"
}
```

7. Obtenir la liste des utilisateurs

Cette requête permet d'obtenir une liste d'utilisateurs avec leurs informations publiques.

Requête

GET /api/users

Réponse

Si le token est valide, une réponse de type « 200 OK » sera retournée :

```
{
  "success" : true,
  "list":
          [
            {<user 1>}, {...}, {<user n>}
          ]
}
```

Erreurs

Si le token est manquant ou invalide, une réponse de type « 403 Forbidden » sera retournée (ci-dessous, le token est manquant) :

```
{
 success : false,
 error : "token field is missing"
}
```

1. Liste d'amis

1. Ajouter un ami

Cette requête permet d'ajouter un ami à la liste d'amis de l'utilisateur.

Requête

POST /api/friends

```
{
 "friend_email":<friend_email (string)>
}
```

Réponse

Si les paramètres sont valides, une réponse de type « 201 Created » sera retournée. Si les deux utilisateurs sont déjà amis, le type de réponse sera « 200 Created » :

```
{
 "success" : true,
 "friend" : <Utilisateur>
}
```

Erreurs

Si l'id de l'ami est invalide, une réponse de type « 404 Not Found » sera retournée :

```
{
  "success" : false,
  "error" : "Friend does not exists"
}
```

Si le token est manquant ou invalide, une réponse de type « 403 Forbidden » sera retournée (ci-dessous, le token est manquant) :

```
{
  success : false,
  error : "token field is missing"
}
```

Si le champ "friend_email" est manquant, une réponse de type "400 Bad Request" sera retournée :

{
  success: false
  error: "friend_email field is missing"
}

2. Supprimer des amis

Cette requête permet de supprimer des amis de la liste d'amis de l'utilisateur.

Requête

DELETE /api/friends ?friends_email[]=<email 1>&friends_email[]=<email n>
POST /api/friends/delete

Si la méthode POST est utilisée, une liste d'emails à supprimer doit être passée en paramètre pour effectuer la suppression.

{
  "friends_email":[<email 1>, <email 2>, <email n>]
}

Réponse

Si les paramètres sont valides, une réponse de type « 200 Ok » sera retournée :

{
  "success" : true,
  "message" : "<nb_deleted> friend have been deleted from your friendlist"
  "nb_friends": <nombre d'amis restant dans la liste d'amis (int)>
   "nb_deleted": <nombre d'amis supprimés de la liste d'amis (int)>
}

Erreurs

Si aucun email ne correspond à un ami de l'utilisateur, une réponse de type « 404 Not Found » sera retournée :

{
 "success" : false,
 "message" : "0 friend were deleted from your friendlist",
 "nb_friends":<nombre d'amis de l'utilisateur>,
 "nb_deleted": 0
}

Si le champ "friends_email" est manquant, une réponse de type "400 Bad Request" sera retournée :

{
  success: false
  error: "friends_email field is missing"
}

3. Obtenir la liste d'amis

Cette requête permet d'obtenir une liste de ses amis avec leurs informations.

Requête

GET /api/friends

Réponse

Si le token est valide, une réponse de type 200 OK sera retournée :

```
{
  success: true,
  friends: [
              {
                "id": <friendship id (int)>,
                "friend_id": <friend id (int)>,
                "user_id": <user id (int)>,
                "created_at": <datetime (string)>
                "first_name": <friend first_name (string)>,
                "last_name": <friend last_name (string)>,
                "username": <friend username (string)>,
                "email": <friend email (string)>
              },
               <utilisateur 2>, ..., <utilisateur n> ],
  nb_friends : <nombre d'amis dans la liste (int)>
}
```

Erreurs

Si le token est manquant ou invalide, une réponse de type « 403 Forbidden » sera retournée (ci-dessous, le token est manquant) :

```
{
  success : false,
  error : "token field is missing"
}
```

2. Musique

1. Afficher les détails d'une musique

Cette requête permet d'obtenir le contenu d'une musique de l'utilisateur.

Requête

GET /api/tracks/<track_id>

Réponse

Si le token est valide et que la musique appartient à l'utilisateur, une réponse de type 200 OK sera retournée :

```
{
   "success":true,
   "track":
                {
                  "album":"<track album >",
                  "artist":"<track artist>",
                  "genre":<track genre>,
                  "id":<track id>,
                  "numberTrack":<track number>,
                  "song_updated_at":<datetime field>,
                  "time":<time in seconds>,
                  "title":<track title>,
                  "user_id":<user id>
                }
}
```

Erreurs

Si le token est manquant ou invalide ou si l'utilisateur ne possède pas la musique, une réponse de type « 403 Forbidden » sera retournée (ci-dessous, le token est manquant) :

```
{
  success : false,
  error : "invalid token"
}
```

2. Obtenir la liste de musiques de l'utilisateur

Cette requête permet d'obtenir la liste de musiques de l'utilisateur.

Requête

GET /api/tracks

Réponse

Si le token est valide, une réponse de type 200 OK sera retournée :

```
{
  "success":true,
  "list":[ <Musique>, <>...]
}
```

Erreurs

Si le token est manquant ou invalide, une réponse de type « 403 Forbidden » sera retournée (ci-dessous, le token est manquant) :

```
{
  success : false,
  error : "invalid token"
}
```

3. Ajouter d'une musique

Cette requête permet d'ajouter une musique ainsi que des informations en relation avec celle-ci.

Requête

POST /api/tracks/

La requête prend les informations relatives à la musique en paramètre :

```
{
  "tracks":
            [
             {
               "album":<track album (string)>,
               "artist":<track artist (string)>,
               "genre":<track genre (string)>,
               "numberTrack":<track number (int)>,
               "time":<track duration in seconds (int)>,
               "title":<track title (string REQUIRED)>
             }, <autre track>, ...
           ]
}
```

Réponse

Si le token est valide et au moins une musique est ajoutée, une réponse de type « 201 Created » sera retournée :

```
{
 "success":true,
 "message":"Tracks have been created"
}
```

Erreurs

Si le token est manquant ou invalide, une réponse de type « 403 Forbidden » sera retournée (ci-dessous, le token est manquant) :

```
{
  success : false,
   error : "invalid token"
}
```

4. Mettre à jour les informations sur une musique

Cette requête permet de mettre à jour des informations concernant une musique.

Requête

PUT /api/tracks/<track_id>

La requête prend les champs qui sont amenés à être mis à jour (une liste exhaustive des attributs est disponible dans la requête d'une musique) en paramètre:

```
{
    "title":"New Title"
}
```

Réponse

Si le token et l'id sont correctes, et si l'utilisateur possède la musique, les informations de la musique seront mises à jour, le message retourné sera :

```
{
  "success":true,
  "message":"Track information have been updated."
}
```

Erreurs

Si le token est manquant, invalide ou si l'utilisateur ne possède pas la musique, une réponse de type « 403 Forbidden » sera retournée (ci-dessous, le token est manquant) :

```
{
  success : false,
  error : "invalid token"
}
```

Si l'id passé en paramètre ne correspond à aucune musique, une réponse de type « 404 Not Found » sera retournée :

```
{
  "success":false,
  "error":"The track cannot be found"
}
```

5. Supprime une musique de la liste de musique

Cette requête permet de supprimer une musique de la bibliothèque.

Requête

DELETE /api/tracks/<track_id>

Réponse

Si le token est valide et la musique a bien été supprimée, une réponse de type 200 « OK » sera retournée.

```
{
  "success":true,
  "message":"Track has been deleted."
}
```

Erreurs

Si le token est manquant ou invalide, une réponse de type « 403 Forbidden » sera retournée (ci-dessous, le token est manquant) :

```
{
   success : false,
   error : "invalid token"
}
```

Si la musique n'existe pas ou n'appartient pas à l'utilisateur, une réponse de type type « 404 Not Found » sera retournée.

6. Supprime une ou plusieurs musiques de la liste de musique

Cette requête permet de supprimer une ou plusieurs musiques.

Requête

DELETE /api/tracks ?tracks_id[]=<id 1>&tracks_id[]=<id n>
POST /api/tracks/delete

Si la méthode POST est utilisée, une liste d'id de musique doit être passée en paramètre pour effectuer la suppression.

```
{
   "tracks_id":[<id 1>, <id 2>, <id n>]
}
```

Réponse

Si le token est valide, une réponse de type 200 « OK » sera retournée, même si les ids de musiques étaient invalide.

```
{
  "success":true,
  "message":"Elements have been deleted."
}
```

Erreurs

Si le token est manquant ou invalide, une réponse de type « 403 Forbidden » sera retournée (ci-dessous, le token est manquant) :

```
{
  success : false,
  error : "invalid token"
}
```

3. Listes de lecture

1. Création d'une liste de lecture

Requête

POST /api/playlist

L'attribut « title » doit faire une longueur minimum de 5 caractères.

```
{
  "title":<nom de la piste de lecture> (string)
}
```

Réponse

Si les paramètres sont valides, une réponse de type « 201 Created » sera retournée :

```
{
 "success":true,
 "message":"Playlist has been created",
 "id":   <id de la liste de lecture>
}
```

Erreurs

Si le token est manquant ou invalide, une réponse de type « 403 Forbidden » sera retournée (ci-dessous, le token est manquant) :

```
{
  success : false,
  error : "invalid token"
}
```

Si les paramètres sont invalides, une réponse de type « 400 Bad Request » sera retournée contenant le détail de l'erreur :

```
{
   success: false,
   error: "title has already been taken"
}
```

2. Modification d'une liste de lecture

Requête

PUT /api/playlist/<id>

Cette requête prend les attributs devant être mis à jour en paramètre.

{
  "title":<nom de la piste de lecture>
}

Réponse

Si les paramètres sont valides, une réponse de type « 200 Ok » sera retournée :

{
 "success":true,
 "message":"Playlist has been created"
}

Erreurs

Si le token est manquant ou invalide, une réponse de type « 403 Forbidden » sera retournée (ci-dessous, le token est manquant) :

{
  success : false,
  error : "invalid token"
}

Si les paramètres sont invalides, une réponse de type « 400 Bad Request » sera retournée contenant le détail de l'erreur :

{
   success: false,
   error: "title has already been taken"
}

3. Suppression d'une liste de lecture

Requête

DELETE /api/playlist/<id>

Réponse

Si la liste de lecture existe et appartient à l'utilisateur, une réponse de type « 200 Ok » sera retournée :

{
 "success":true,
 "message":"Playlist has been deleted"
}

Erreurs

Si le token est manquant ou invalide, une réponse de type « 403 Forbidden » sera retournée (ci-dessous, le token est manquant) :

```
{
  success : false,
  error : "invalid token"
}
```

Si la liste de lecture n'existe pas ou n'appartient pas à l'utilisateur, une réponse de type « 404 Not Found » sera retournée :

```
{
   success: false,
   error: " Playlist does not exist"
}
```

4. Suppression de plusieurs listes de lecture

Requête

DELETE /api/playlist/ ?playlist_ids[]=<id 1>&playlist_ids[]=<id n>
POST /api/playlist/delete

Si la méthode POST est utilisée, les paramètres suivants sont nécessaires :

```
{
  "playlist_ids":[<id 1>, <id 2>, <id n>]
}
```

Réponse

Si au moins une des listes de lecture existe et appartient à l'utilisateur, une réponse de type « 200 Ok » sera retournée :

```
{
 "success":true,
 "message":"Playlist has been deleted"
}
```

Erreurs

Si le token est manquant ou invalide, une réponse de type « 403 Forbidden » sera retournée (ci-dessous, le token est manquant) :

```
{
  success : false,
  error : "invalid token"
}
```

Si aucune des listes de lecture n'existe pas ou n'appartiennent à l'utilisateur, une réponse de type « 404 Not Found » sera retournée :

{
  success: false,
  error: " Playlist does not exist"
}

5. Ajout de plusieurs musiques dans une liste de lecture

Requête

Il est possible d'ajouter une musique à une liste de lecture soit en donnant son id, soit en indiquant le contenu de la musique. Dans ce dernier cas, la musique sera d'abord ajoutée à la liste de musique puis à la liste de lecture.

POST /api/playlist/<id>/tracks

{
  "tracks_id" : [<track_id 1>, <track_id 2>, <track_id n>]
  "tracks" : [<track 1>, <track 2>, <track n>]
}

Réponse

Si les paramètres sont valides, une réponse de type « 201 Created » sera retournée :

{
 "success":true,
  "message": "Tracks have been added to the playlist"
}

Erreurs

Si le token est manquant ou invalide, une réponse de type « 403 Forbidden » sera retournée (ci-dessous, le token est manquant) :

{
  success : false,
  error : "invalid token"
}

Si au moins une des musiques n'existent pas ou n'appartient pas à l'utilisateur, une réponse de type « 404 Not Found » sera retournée :

{
  success: false
  error: "Couldn't find track <track_id>"
}

Si aucune musique n'est présente dans les paramètres, une réponse de type « 400 Bad Request » sera retournée :

{
  success : false,
   error : "No track found in params"
}

Si la liste de lecture n'existe pas ou n'appartient pas à l'utilisateur, une réponse de type « 404 Not Found » sera retournée :

```
{
  success : false,
   error : "Playlist does not exist"
}
```

6. Suppression de plusieurs musiques dans une liste de lecture

Requête

DELETE /api/playlist/<id>/tracks ?tracks_id[]=<track_id 1>&
tracks_id[]=<track_id n>
POST /api/playlist/<id>/tracks/delete

Si la méthode POST est utilisée, les paramètres suivants sont nécessaires :

```
{
  "tracks_id" : [<track_id 1>, <track_id 2>, <track_id n>]
}
```

Réponse

Si les paramètres sont valides, une réponse de type « 200 Ok » sera retournée :

```
{
 "success":true,
 "tracks":  "<n> tracks have been deleted from the playlist",
 "nb_tracks":  <nouveau nombre de musique dans la liste de musique>,
 "nb_deleted":  <nombre de musiques supprimés de la liste de lecture>,
}
```

Erreurs

Si le token est manquant ou invalide, une réponse de type « 403 Forbidden » sera retournée (ci-dessous, le token est manquant) :

```
{
  success : false,
  error : "invalid token"
}
```

Si la liste de lecture n'existe pas, une réponse de type « 404 Not Found » sera retournée :

```
{
  success : false,
  error : " Playlist does not exist"
}
```

Si aucune musique n'existe dans la liste de lecture, une réponse de type « 404 Not Found » sera retournée :

```
{
  success : false,
  error : "0 track has been deleted from the playlist"
}
```

7. Suppression d'une musique de la liste de lecture

Requête

DELETE /api/playlist/<playlist_id>/tracks/<track_id>

Réponse

Si les paramètres sont valides, une réponse de type « 200 Ok » sera retournée :

```
{
 "success":true,
 "tracks":  " Track <track_id> has been deleted from the playlist",
 "nb_tracks":  <nouveau nombre de musique dans la liste de musique>,
}
```

Erreurs

Si le token est manquant ou invalide, une réponse de type « 403 Forbidden » sera retournée (ci-dessous, le token est manquant) :

```
{
  success : false,
  error : "invalid token"
}
```

Si la liste de lecture n'existe pas, une réponse de type « 404 Not Found » sera retournée :

```
{
  success : false,
  error : " Playlist does not exist"
}
```

Si aucune musique n'existe dans la liste de lecture, une réponse de type « 404 Not Found » sera retournée :

```
{
  success : false,
  error : "0 track were deleted from the playlist"
}
```

8. Récupérer les musiques d'une liste de lecture

Requête

GET /api/playlist/<id>/tracks

Réponse

Si le token est valide, une réponse de type « 200 Ok » sera retournée :

```
{
 "success":true,
 "id": <id de la liste de lecture>
 "tracks":[<Musique 1>, <Musique 2>, <Musique n>]
}
```

Erreurs

Si le token est manquant ou invalide, une réponse de type « 403 Forbidden » sera retournée (ci-dessous, le token est manquant) :

```
{
  success : false,
  error : "invalid token"
}
```

9. Lister les listes de lectures

Requête

GET /api/playlist

Réponse

Si le token est valide, une réponse de type « 200 Ok » sera retournée :

```
{
 "success":true,
 "list":[<playlist 1>, <playlist 2>, <playlist n>]
}
```

Erreurs

Si le token est manquant ou invalide, une réponse de type « 403 Forbidden » sera retournée (ci-dessous, le token est manquant) :

```
{
  success : false,
  error : "invalid token"
}
```

10. Afficher les details d'une liste de lecture

Requête

GET /api/playlist/<id>

Réponse

Si le token est valide et la liste de lecture appartient à l'utilisateur, une réponse de type « 200 Ok » sera retournée :

```
{
  success: true
  playlist:
            {
              id: <id>
              nb_tracks: <nombre de musique dans la liste de lecture>
              title: <nom de la liste de lecture>
              user_id: <id de l'utilisateur possédant la liste de lecture>
            }
}
```

Erreurs

Si le token est manquant ou invalide, une réponse de type « 403 Forbidden » sera retournée (ci-dessous, le token est manquant) :

```
{
  success : false,
  error : "invalid token"
}
```

Si la liste de lecture n'existe pas ou n'appartient pas à l'utilisateur, une réponse de type « 404 Not Found » sera retournée :

```
{
  success : false,
  error : "Playlist does not exist"
}
```

4. Messagerie instantanée

Le chat entre utilisateur est fait à l'aide d'un serveur compatible avec les protocoles Jabber et XMPP. Il faut donc utiliser une librairie compatible pour l'implémenter.

L'url du serveur est https://audiowire.co et le serveur utilise les ports standard conformément aux protocoles.

Les utilisateurs d'AudioWire auront des comptes sur le serveur XMPP de manière automatique.

Les ajouts et suppression d'amis doivent être faits en utilisant les méthodes de l'API qui mettra automatiquement à jour le serveur XMPP.
Le nom d'utilisateur à utiliser pour connecter un utilisateur est <username>@audiowire.co et le mot de passe est le même que celui du compte AudioWire.
Il n'est pas possible de modifier directement le mot de passe du compte de messagerie instantanée. Le mot de passe et les autres informations des comptes utilisateurs du serveur Jabber sont synchronisés avec celles du compte AudioWire. Il faut donc utiliser l'API pour modifier ses informations.

L'interface d'administration du serveur Jabber est disponible à l'adresse http://audiowire.co:9090. Pour obtenir un accès, contactez un administrateur.