|  |  |  |
| --- | --- | --- |
| **Documentation API**  **Brunel\_m, Feraud\_g, Xie\_h, Strzel\_a, Meilhva\_v, Rose\_s, Pasqui\_c, Defran\_h**   |  |  | | --- | --- | |  |  | |

Tableau des révisions :

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Auteur | Section | Commentaire |
| 06/08/13 | Hugo Defrance | Toutes | Ajout des titres et présentation générale |
| 07/08/13 | Hugo Defrance | Toutes | Détail des requêtes |

Table des matières

1 Présentation de l’API 4

2 Requêtes disponibles 4

2.1 Utilisateur 5

2.1.1 S’enregistrer – création d’un compte utilisateur et d’un token 5

2.1.2 Connexion – création de token 7

2.1.3 Déconnexion – destruction de token 8

2.1.4 Mise à jour des informations d’un compte utilisateur 9

2.1.5 Obtenir des informations sur un utilisateur 10

2.1.6 Obtenir la liste des utilisateurs 11

2.2 Liste d’amis 12

2.2.1 Ajouter un ami 12

2.3 Supprimer un ami 13

2.4 Obtenir la liste d’amis 14

2.5 Musique 15

2.5.1 Obtenir la liste de musiques de l’utilisateur 15

2.5.2 Ajouter les informations d’une musique 15

2.5.3 Obtenir les informations sur une musique 15

2.5.4 Mettre à jour les informations sur une musique 15

2.5.5 Supprime une musique de la liste de musique 16

2.6 Image 16

2.6.1 Ajouter une image 16

2.6.2 Obtenir une image 16

2.6.3 Supprimer une image 16

2.7 Message 17

2.7.1 Obtenir la liste des conversations d’un utilisateur 17

2.7.2 Obtenir une conversation entre deux utilisateurs 17

2.7.3 Modifier un message 17

2.7.4 Supprimer un message 17

2.7.5 Supprimer une conversation 18

2.7.6 Obtenir tous les messages envoyés 18

2.7.7 Obtenir tous les messages reçus 18

2.7.8 Envoyer un message 18

# 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 chargé 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 et 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.

## Utilisateur

### 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 /users

Format de la requête:

{  
"user" :   
 {  
 "email" : "<user\_email>",  
 "password" : "<user\_password>",  
 "password\_confirmation’ : "<user\_password>"  
 }  
}

#### 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>"  
}

#### Erreurs

En cas d’erreurs, une réponse de type « 201 Created » sera retournée. Voici un exemple de réponse d’erreur contenant toutes les erreurs possibles :

{  
 "success": false,  
 "errors":  
 {  
 "email":  
 [  
 "has already been taken",  
 "can't be blank"  
 ],  
 "password":  
 [  
 "doesn't match confirmation",   
 "is too short (minimum is 8 characters)",  
 "can't be blank"  
 ]  
 }  
}

### Connexion – création de token

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

#### Requête

POST /users/login

Format de la requête:

{  
 "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,  
 "message" : "Invalid email or password."  
}

### Déconnexion – destruction de token

Cette requête permet de détruire un token.

#### Requête

DELETE /users/logout

Format de la requête:

{  
 "token" : "<token>"  
}

#### 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."   
}

### 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.

#### Requête

PUT /users

Format de la requête :  
Cette requête doit toujours contenir un token valide comme montré dans l’exemple. En plus de ce token, il faut ajouter la version mise à jour de la donnée devant être mise à jour. Par exemple, la requête suivante met à jour le mot de passe de l’utilisateur :

{  
 "token" : "<token>",  
 "password" : "<nouveau mot de passe>"  
}

#### Réponse

Si les paramètres sont valides, une réponse de type « 201Created » 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"  
}

### Obtenir des informations sur un utilisateur

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

#### Requête

GET /users/<id>?token=<token>

#### 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": {   
 "avatar\_content\_type" : null,  
 "avatar\_file\_name" : null,  
 "avatar\_file\_size" : null,  
 "avatar\_updated\_at" : null,  
 "created\_at" : "2013-08-07T20:03:09Z",  
 "email" : "toto@toto.com",  
 "id" : 1,  
 "updated\_at" : "2013-08-10T14:43:43Z"  
 }  
}

#### 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"  
}

### Obtenir la liste des utilisateurs

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

#### Requête

GET /users/?token=<token>

Cette requête ne prend qu’un token valide en paramètre GET.

#### 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"  
}

## Liste d’amis

### Ajouter un ami

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

#### Requête

POST /friends

Format de la requête :

{  
 "token" : <token>,   
 "friend\_id":<friend\_id>  
}

#### Réponse

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

{  
 "success" : true,  
 "friend" : {<friend\_content>}  
}

#### 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"  
}

## Supprimer un ami

Cette requête permet de supprimer un ami de la liste d’amis de l’utilisateur.

#### Requête

DELETE /friends/<friend\_id>

Format de la requête :

{  
 "token" : <token>  
}

#### Réponse

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

{  
 "success" : true,  
 "message" : "Friendship no longer exists"  
}

#### Erreurs

Si l’id de l’ami est ne figure pas dans la liste d’amis de l’utilisateur, une réponse de type « 404 Not Found » sera retournée :

{  
 "success" : false,  
 "error" : " User <friend\_id> is not your friend"  
}

## Obtenir la liste d’amis

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

#### Requête

GET /friends ?token=<token>

Cette requête ne prend qu’un token valide en paramètre GET.

#### Réponse

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

{  
 success: true,  
 friends: [<friend 1>, <friend 2>, …, <friend 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"  
}

## Musique

### Obtenir la liste de musiques de l’utilisateur

#### Requête

#### Réponse

#### Erreurs

### Ajouter les informations d’une musique

#### Requête

#### Réponse

#### Erreurs

### Obtenir les informations sur une musique

#### Requête

#### Réponse

#### Erreurs

### Mettre à jour les informations sur une musique

#### Requête

#### Réponse

#### Erreurs

### Supprime une musique de la liste de musique

#### Requête

#### Réponse

#### Erreurs

## Image

### Ajouter une image

#### Requête

#### Réponse

#### Erreurs

### Obtenir une image

#### Requête

#### Réponse

#### Erreurs

### Supprimer une image

#### Requête

#### Réponse

#### Erreurs

## Message

### Obtenir la liste des conversations d’un utilisateur

TO DO

#### Requête

#### Réponse

#### Erreurs

### Obtenir une conversation entre deux utilisateurs

#### Requête

#### Réponse

#### Erreurs

### Modifier un message

#### Requête

#### Réponse

#### Erreurs

### Supprimer un message

#### Requête

#### Réponse

#### Erreurs

### Supprimer une conversation

#### Requête

#### Réponse

#### Erreurs

### Obtenir tous les messages envoyés

#### Requête

#### Réponse

#### Erreurs

### Obtenir tous les messages reçus

#### Requête

#### Réponse

#### Erreurs

### Envoyer un message

#### Requête

#### Réponse

#### Erreurs

Musique:

|  |  |  |  |
| --- | --- | --- | --- |
| URL | Méthode HTTP | Paramètres | Fonction |
| /songs/ | GET | Aucun | Retourne la liste des musiques |
| /songs/ | POST | band : String,  title : String,  track : String | Envoyer les informations sur une musique |
| /songs/id/ | GET | id:Integer | Obtenir les informations sur une musique |
| /songs/id/ | PATCH | band : String and/or  title : String and/or  track : String | Mettre à jour une musique partiellement |
| /songs/id/ | DELETE | Aucun | Supprime une musique de la liste de musique d’un utilisateur |

Image:

|  |  |  |  |
| --- | --- | --- | --- |
| URL | Méthode HTTP | Paramètres | Fonction |
| /images/ | POST | Fichier | Envoi d’une image |
| /images/id/ | GET | id : Integer | Récupérer les informations d’une image |
| /images/id/ | DELETE | id : Integer | Supprimer une image |

Message:

|  |  |  |  |
| --- | --- | --- | --- |
| URL | Méthode HTTP | Paramètres | Fonction |
| /messages/ | GET | Aucun | Liste tous les messages qu’un utilisateur a reçus ou envoyé |
| /messages/id/ | GET | Aucun | Retourne un message |
| /messages/id/ | PUT | Content : String | Met à jour un message |
| /messages/id/ | DELETE | Aucun | Supprime un message |
| /messages/sent/ | GET | Aucun | Liste tous les messages envoyé |
| /messages/received/ | GET | Aucun | Liste tous les messages reçu |
| /messages/ | POST | User : ID,  content:String | Envoie un message à un autre utilisateur |
| /messages/<user> | GET | Id de l’utilisateur à ajouter dans l’url | Liste tous les messages envoyé et reçu à l’utilisateur passé en paramètre |

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

Toutes les requêtes répondront en utilisant les codes de réponse HTTP standard et contiendront éventuellement des données encodées en JSON3.

Voici les codes d’erreurs utilisés :

200 => Ok

201 => Created

204 => No Content

400 => Bad Request

401 => Unauthorized

403 => Forbidden

500 => Internal Server Error