Jeremy

Music Sheet Writer

Music Sheet Writer

Bilan d’architecture d’aout

Objectifs du document

Glossaire

***– A –***

API (*Application Programming Interface*) : une API est une interface de programmation. C’est une porte d’accès à des fonctionnalités en faisant abstraction de leurs fonctionnements.

***– H –***

HTTP (*Hypertext Transfer Protocol*) : HTTP est un protocole de communication client-serveur. Il peut fonctionner sur n'importe quelle connexion fiable. Les clients HTTP les plus connus sont les navigateurs Web permettant à un utilisateur d'accéder à un serveur contenant les données.

***– R –***

REST (*Representation State Tranfer*) : REST est un style d’architecture logiciel qui, la plupart du temps, est utilisé pour implémenter un Web Service. La communication se fait donc entre un client et serveur et utilise le protocole HTTP.

***– W –***

Web Service : un Web Service est un programme informatique de la famille des technologies web permettant la communication et l'échange de données entre applications. Il s'agit donc d'un ensemble de fonctionnalités exposées sur internet ou sur un intranet.

Description du document

|  |  |
| --- | --- |
| Titre | Music Sheet Writer : Bilan d’architecture d’aout |
| Date | 09/08/2015 |
| Auteur | Jeremy Harrault |
| Responsable | Jonathan Racaud |
| E-mail | musicsheetwriter\_2017@labeip.epitech.eu |
| Sujet | Bilan d’architecture d’aout |
| Version du modèle | 1.0 |

Tableau des révisions

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Auteur** | **Section(s)** | **Commentaire** |
| 18/07/2015 | Jeremy HARRAULT | Toutes | Création du document |
| 19/07/2015 | Jeremy HARRAULT | 8.1.1.2 Vue en ressources  9. Vue données | Définition des ressources de l’API REST, définition de la base de données. |
| 26/07/2015 | Jeremy HARRAULT | 9. Vue données | Modification du schéma de base de données. |
| 09/08/2015 | Jeremy HARRAULT | 8.1.1 API REST  9. Vue données | Définition des entrées/sortie de l’API REST, Modification du schéma de base de données. |
| 15/08/2015 | Jeremy HARRAULT | 6. Vue Processus  8.1.1 API REST | Définition du séquencement de certains processus, Modification des entrées/sorties de l’API REST |

Table des matières

[3. Architecture, buts et contraintes 1](#_Toc426897431)

[3.1. Objectifs spécifiques ayant un impact sur l’architecture 1](#_Toc426897432)

[6. Vue processus 2](#_Toc426897433)

[6.1. Enregistrement 2](#_Toc426897434)

[6.1.1. Description 2](#_Toc426897435)

[6.1.2. Vue dynamique 2](#_Toc426897436)

[6.2. Connexion 2](#_Toc426897437)

[6.2.1. Description 2](#_Toc426897438)

[6.2.2. Vue dynamique 2](#_Toc426897439)

[6.3. Changement d’adresse email 2](#_Toc426897440)

[6.3.1. Description 2](#_Toc426897441)

[6.3.2. Vue dynamique 2](#_Toc426897442)

[6.4. Réinitialisation de mot de passe 2](#_Toc426897443)

[6.4.1. Description 2](#_Toc426897444)

[6.4.2. Vue dynamique 2](#_Toc426897445)

[8. Implémentation 3](#_Toc426897446)

[8.1. Vue globale 3](#_Toc426897447)

[8.1.1. API REST pour le site internet et les applications mobiles 3](#_Toc426897448)

[9. Vue données 11](#_Toc426897449)

# Architecture, buts et contraintes

## Objectifs spécifiques ayant un impact sur l’architecture

# 

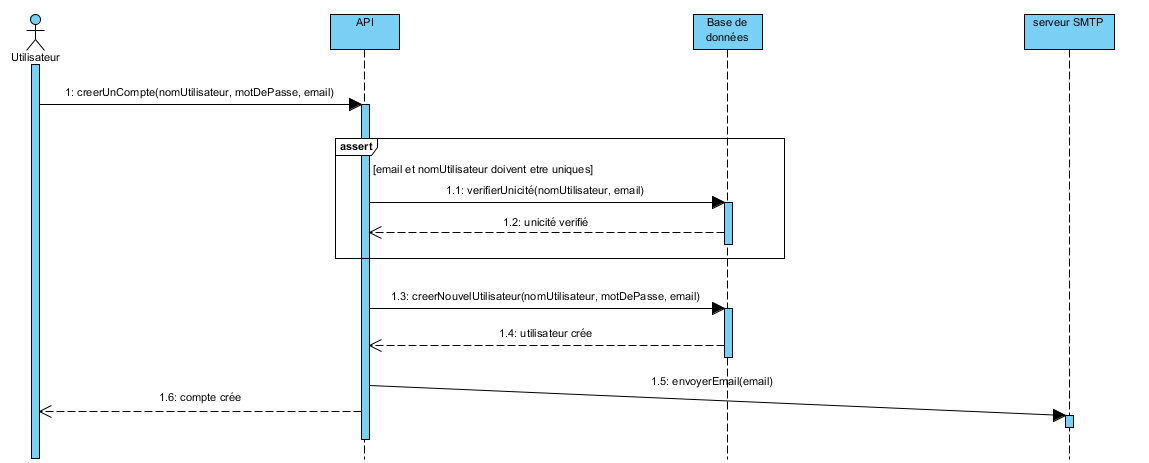
# Vue processus

## Création de compte

### Description

La création d’un compte est soumise à plusieurs contraintes d’unicité. En effet, le nom d’utilisateur ainsi que l’adresse email ne doivent appartenir à aucun autre utilisateur. Si tel est le cas, un nouvel utilisateur est créé et un email de confirmation d’adresse mail est envoyé.

### Vue dynamique

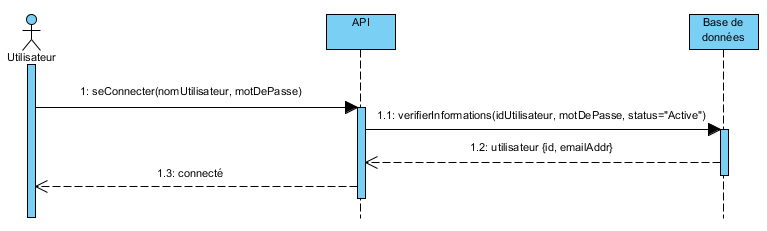


## Connexion

### Description

La connexion d’utilisateur se fait grâce au nom d’utilisateur et au mot de passe. Le compte doit être actif – c’est-à-dire qu’il ne doit pas avoir été fermé précédemment – pour pouvoir se connecter avec.

### Vue dynamique

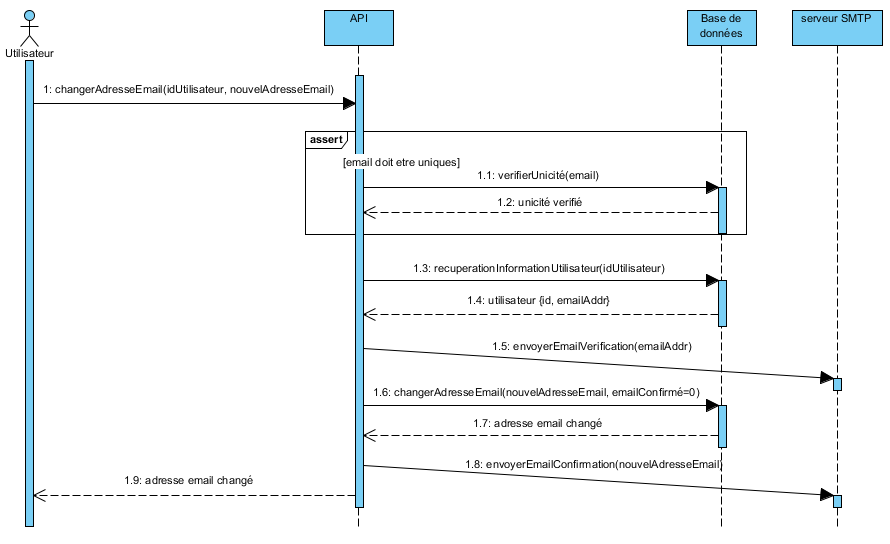


## Changement d’adresse email

### Description

Lors d’un changement d’adresse email, la nouvelle adresse ne doit pas déjà être attribué à un compte existant. SI elle est unique, alors un mail de vérification est envoyé à l’ancienne adresse email et un mail de confirmation à la nouvelle. L’utilisateur devra donc validé sa nouvelle adresse email.

### Vue dynamique

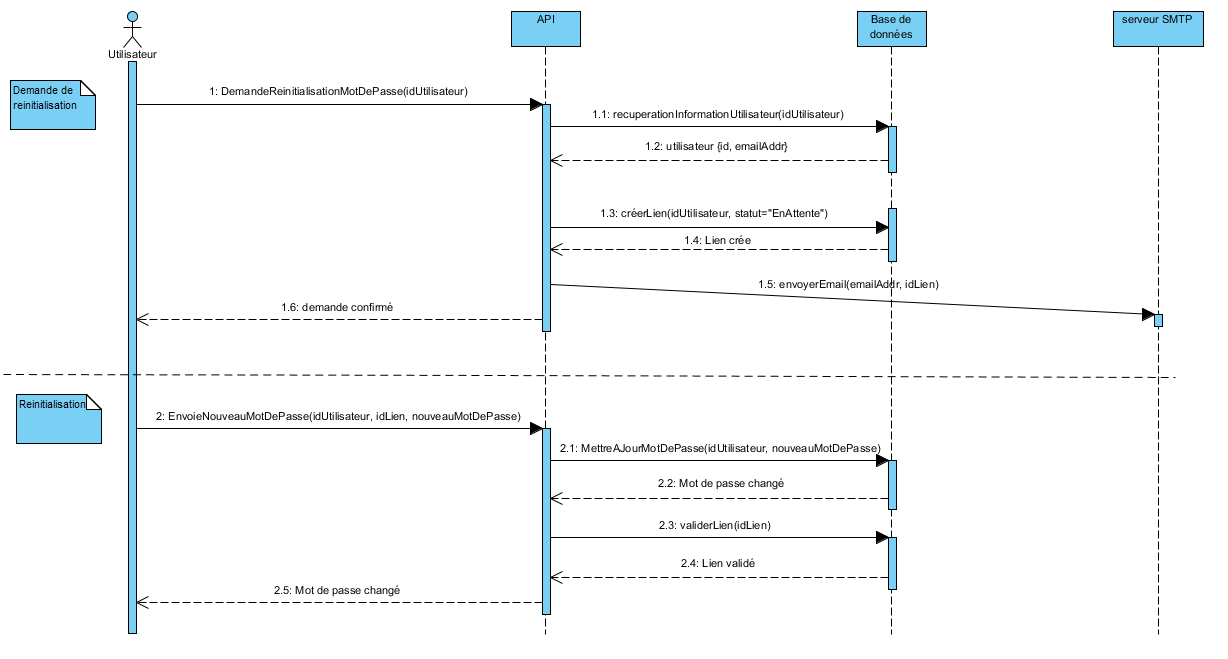


## Réinitialisation de mot de passe

### Description

La réinitialisation du mot de passe est un processus appliqué lorsqu’un utilisateur ne se souvient plus de son mot de passe. Ce processus se fait en deux temps. Tout d’abord, l’utilisateur fait une demande de réinitialisation. Un lien lui permettant de réinitialiser son mot de passe lui sera alors envoyé par mail. Une fois arrivé sur le lien, l’utilisateur pourra soumettre un nouveau mot de passe.

### Vue dynamique



# Implémentation

## Vue globale

### API REST pour le site internet et les applications mobiles

#### Description

L’API REST est un Web Service. C’est l’interface entre la vue et la base de données. Elle est appelée en utilisant des requêtes HTTPS.

Les requêtes seront de la forme [https://[adresse\_ip]/musicsheetwriter\_api/[URI]](https://[adresse_ip]/musicsheetwriter_api/%5bURI%5d) avec l’URI étant le chemin de la ressource. Pour le mobile, l’API sera accessible en utilisant l’URL [https://[adresse\_ip]/m/musicsheetwriter\_api/[URI]](https://[adresse_ip]/m/musicsheetwriter_api/%5bURI%5d).

Les entrées/sortie de l’API seront en anglais.

Toutes les erreurs seront de la forme :

*<error>*

*<short\_code>[short\_code]</short\_code>*

*<message>[message]</message>*

*</error>*

#### Vue en ressource

| Ressource | Méthode HTTP | | Description |
| --- | --- | --- | --- |
| /login | POST | | Connecte un utilisateur à partir de ses identifiants de connexion. |
| DELETE | | Déconnecte un utilisateur |
| /users | GET | | Récupère la liste des utilisateurs. Des filtres peuvent être ajoutés. |
| POST | | Enregistre un nouvel utilisateur |
| /users/[id] | GET | | Récupère toutes les informations d’un utilisateur |
| DELETE | | Ferme un compte |
| /users/[id]/personal\_data | GET | | Récupère les informations personnelles d’un utilisateur |
| PUT | | Change les informations personnelles d’un utilisateur |
| /users/[id]/photograph | GET | | Récupère la photo de profil d’un utilisateur |
| PUT | | Change la photo de profil d’un utilisateur |
| /users/[id]/credentials/username | GET | | Récupère le pseudonyme d’un utilisateur |
| /users/[id]/credentials/email | GET | | Récupère l’adresse email d’un utilisateur |
| PUT | | Change l’adresse email d’un utilisateur |
| /users/[id]/credentials/password | POST | | Change le mot de passe d’un utilisateur |
| PUT | | Change le mot de passe d’un utilisateur en utilisant son ancien mot de passe |
| /users/[id]/credentials/password\_link | POST | | Crée un nouveau de lien de réinitialisation de mot de passe |
| PUT | | Valide un lien de réinitialisation de mot de passe (le mot de passe a été changé) |
| DELETE | | Invalide un lien de réinitialisation de mot de passe (le lien a expiré) |
| /users/[id]/subscriptions | GET | | Récupère la liste des abonnements d’un utilisateur. Des filtres peuvent être ajoutés |
| POST | | Ajoute un nouvel abonnement à un utilisateur |
| /users/[id]/subscriptions/[id] | DELETE | | Retire un abonnement d’un utilisateur |
| /users/[id]/subscribers | GET | | Récupère la liste des abonnés d’un utilisateur. Des filtres peuvent être ajoutés. |
| /users/[id]/scores/own | GET | | Récupère la liste des partitions d’un utilisateur |
| POST | | Ajoute une nouvelle partition à un utilisateur |
| /users/[id]/scores/own/[id] | DELETE | | Retire une partition d’un utilisateur. |
| /users/[id]/scores/favourites | GET | | Récupère la liste des partitions favorites d’un utilisateur |
| POST | | Ajoute une nouvelle partition favorite à un utilisateur |
| /users/[id]/scores/favourites/[id] | DELETE | | Retire une partition favorite d’un utilisateur |
| /users/[id]/purchases | GET | | Récupère la liste des achats d’un utilisateur |
| /scores | GET | | Récupère la liste des partitions. Des filtres peuvent être ajoutés |
| /scores/[id] | GET | | Récupère une partition |
| DELETE | | Supprime une partition |
| /purchases | GET | | Récupère la liste des achats. Des filtres peuvent être ajoutés. |
| POST | | Crée un nouvel achat |
| /purchases/[id] | GET | | Récupère les informations d’un achat. |
| PUT | | Met à jour les informations d’un achat |
| DELETE | | Désactive un achat. |
| /contact | POST | Envoie un email à l’équipe MSW. | |

#### Interface

| Requête | | | Reponse | |
| --- | --- | --- | --- | --- |
| Ressource | Méthode HTTP | Corps | Code HTTP | Corps |
| /login | POST | *<credentials>*  *<username>[username]</username>*  *<password>[password]</password>*  *</credentials>* | 200 | *<user id=[id]>*  *<username>[username]</username>*  *<last\_activity\_date>[last\_activity\_date]</last\_activity\_date>*  *</user>* |
| 401 | *<error>*  *<shortcode>LOG-BADCREDENTIALS</shortcode>*  *<message>*  *Credentials not valid.*  *</message>*  *</error>* |
| 403 | *<error>*  *<shortcode>LOG-CLOSEDACC</shortcode>*  *<message>*  *Account closed*  *</message>*  *</error>* |
| DELETE | *<credentials>*  *<username>[username]</username>*  *</credentials>* | 200 | EMPTY |
| /users \*2  Des filtres peuvent être ajoutés en tant que paramètre GET  - uname  - fname  - uname  - email | GET | EMPTY | 200 | *<users count=[count]>*  *<user index=[index] id=[id]>*  *<username>[username]</username>*  *<personal\_data>*  *<first\_name>[first\_name]</first\_name>*  *<surname>[surname]</surname]*  *<email>[email]</email>*  *<photograph>*  *<url>[url]</url>*  *</photograph>*  *<personal\_data>*  *<message>[message]</message>*  *<status>[status></status>*  *<last\_activity\_date>[last\_activity\_date]</last\_activity\_date>*  *<creation\_date>[creation\_date]</creation\_date>*  *</user>*  *…*  *</users>*  The ‘index’ is the position (beginning by 0) of the tag among the list. |
| POST | *<user>*  *<username>[username]</username>*  *<password>[password]</password*  *<personal\_data>*  *<first\_name>[first\_name]</first\_name>*  *<surname>[surname]</surname]*  *<email>[email]</email>*  *</personal\_data>*  *</user>* | 200 | *<user id=[id]>*  *<username>[username]</username>*  *<personal\_data>*  *<first\_name>[first\_name]</first\_name>*  *<surname>[surname]</surname]*  *<email>[email]</email>*  *<photograph>*  *<url>[url]</url>*  *</photograph>*  *</personal\_data>*  *<message>[message]</message>*  *<status>[status></status>*  *<last\_activity\_date>[last\_activity\_date]</last\_activity\_date>*  *<creation\_date>[creation\_date]</creation\_date>*  *</user>* |
| 400 | *<error>*  *<shortcode>USR-BADFIELD</shortcode>*  *<message>*  *[field] not correct*  *</message>*  *</error>*  This error can be returned if a ‘field’ has a bad format. |
| 409 | *<error>*  *<shortcode>REG-NAMEALRUSED</shortcode>*  *<message>*  *Username already used*  *</message>*  *</error>* |
| 409 | *<error>*  *<shortcode>REG-MAILALRUSED</shortcode>*  *<message>*  *Email address already used*  *</message>*  *</error>* |
| /users/[id] \*1 | GET | EMPTY | 200 | *<user id=[id]>*  *<username>[username]</username>*  *<personal\_data>*  *<first\_name>[first\_name]</first\_name>*  *<surname>[surname]</surname]*  *<email>[email]</email>*  <photograph>[photograph]</photograph>  *<personal\_data>*  *<message>[message]</message>*  *<status>[status></status>*  *<last\_activity\_date>[last\_activity\_date]</last\_activity\_date>*  *<creation\_date>[creation\_date]</creation\_date>*  *</user>* |
| DELETE | EMPTY | 200 | *<user id=[id]>*  *<status>[status></status>*  *<closure\_date></closure\_date>*  *</user>* |
| 403 | *<error>*  *<shortcode>CLO-USERALRCLO</shortcode>*  *<message>*  *User account already closed*  *</message>*  *</error>* |
| /users/[id]/personal\_data \*1 | GET | EMPTY | 200 | *<personal\_data>*  *<first\_name>[first\_name]</first\_name>*  *<surname>[surname]</surname]*  *<email>[email]</email>*  *<personal\_data>* |
| PUT | *<personal\_data>*  *<first\_name>[first\_name]</first\_name>*  *<surname>[surname]</surname]*  *<email>[email]</email>*  *<personal\_data>* | 200 | *<personal\_data>*  *<first\_name>[first\_name]</first\_name>*  *<surname>[surname]</surname]*  *<email>[email]</email>*  *<personal\_data>* |
| 400 | *<error>*  *<shortcode>USR-BADFIELD</shortcode>*  *<message>*  *[field] field not correct*  *</message>*  *</error>*  Cette erreur peut être retournée si un champ a un mauvais format. |
| /users/[id]/photograph \*1 | GET | EMPTY | 200 | *<photograph>*  *<url>[url]</url>*  *</photograph>* |
| PUT | *<photograph>*  *<url>[url]</url>*  *</photograph>* | 204 | *<photograph>*  *<url>[url]</url>*  *</photograph>* |
| /users/[id]/credentials/username \*1 | GET | EMPTY | 200 | *<username>*  *[username]*  *</username>* |
| /users/[id]/credentials/email \*1 | GET | EMPTY | 200 | *<email>*  *[email]*  *</email>* |
| PUT | *<email>*  *[email]*  *</email>* | 200 | *<email>*  *[email]*  *</email>* |
| 400 | *<error>*  *<shortcode>USR-BADFIELD</shortcode>*  *<message>*  *email field not correct*  *</message>*  *</error>* |
| /users/[id]/credentials/password \*1 | POST | *<password>*  *<new\_password>*  *[new\_password]*  *</new\_password>*  *</password>* | 204 | EMPTY |
| 400 | *<error>*  *<shortcode>USR-BADFIELD</shortcode>*  *<message>*  *password field not correct*  *</message>*  *</error>* |
| PUT | *<password>*  *<new\_password>*  *[new\_password]*  *</new\_password>*  *<current\_password>*  *[current\_password]*  *</current\_password>*  *</password>* | 204 | EMPTY |
| 400 | *<error>*  *<shortcode>USR-BADFIELD</shortcode>*  *<message>*  *password field not correct*  *</message>*  *</error>* |
| 403 | *<error>*  *<shortcode>USR-WRONGPASS</shortcode>*  *<message>*  *Current password not correct*  *</message>*  *</error>* |
| /users/[id]/credentials/password\_link \*1 | POST | EMPTY | 200 | *<password\_link id=[id]>*  *<status>[status]</status>*  *</password\_link>*  Le status devrait être « Pending » (en attente) |
| PUT | *<password\_link id=[id] />* | 200 | *<password\_link id=[id]>*  *<status>[status]</status>*  *</password\_link>*  Le status devrait être « OK » |
| 403 | *<error>*  *<shortcode>PAS-LINKALRUSED</shortcode>*  *<message>*  *Password link already used*  *</message>*  *</error>* |
| 404 | *<error>*  *<shortcode>PAS-BADLINK</shortcode>*  *<message>*  *Password link not correct*  *</message>*  *</error>* |
| DELETE | *<password\_link id=[id] />* | 200 | *<password\_link id=[id]>*  *<status>[status]</status>*  *</password\_link>*  Le status devrait être « NOK » |
| 403 | *<error>*  *<shortcode>PAS-LINKALRUSED</shortcode>*  *<message>*  *Password link already used*  *</message>*  *</error>* |
| 404 | *<error>*  *<shortcode>PAS-BADLINK</shortcode>*  *<message>*  *Password link not correct*  *</message>*  *</error>* |
| /users/[id]/subscriptions \*1 | GET | EMPTY | 200 | *<subscriptions count=[count]>*  *<user index=[index] id=[id]>*  *<username>[username]</username>*  *</user>*  *…*  *<subscriptions>*  The ‘index’ is the position (beginning by 0) of the tag among the list. |
| POST | *<user id=[id] />* | 200 | *<subscriptions count=[count]>*  *<user index=[index] id=[id]>*  *<username>[username]</username>*  *</user>*  *…*  *<subscriptions>* |
| 404 | *<error>*  *<shortcode>SUB-BADUSERID</shortcode>*  *<message>*  *User ID not correct*  *</message>*  *</error>* |
| /users/[id]/subscriptions/[id] \*1 | DELETE | EMPTY | 200 | *<user id=[id] />* |
| /users/[id]/subscribers \*1 | GET | EMPTY | 200 | *<subscriptions count=[count]>*  *<user index=[index] id=[id]>*  *<username>[username]</username>*  *</user>*  *…*  *<subscriptions>*  The ‘index’ is the position (beginning by 0) of the tag among the list. |
| /users/[id]/scores/own \*1 | GET | EMPTY | 200 | *<scores count=[count]>*  *<score index=[index] id=[id]>*  *<url>[url]</url>*  *</score>*  *…*  *</scores>* |
| POST | *<score>*  *<url>[url]</url>*  *</score>* | 200 | *<scores count=[count]>*  *<score index=[index] id=[id]>*  *<url>[url]</url>*  *<user id=[id] />*  *</score>*  *…*  *</scores>* |
| /users/[id]/scores/own/[id] \*1 | DELETE | EMPTY |  | *<score>*  *<url>[url]</url>*  *</score>* |
| /users/[id]/scores/favourites \*1 | GET | EMPTY | 200 | *<scores count=[count]>*  *<score index=[index] id=[id]>*  *<url>[url]</url>*  *<user id=[id] />*  *</score>*  *…*  *</scores>* |
| POST | *<score>*  *<url>[url]</url>*  *</score>* | 200 | *<score id=[id]>*  *<url>[url]</url>*  *<user id=[id] />*  *</score>* |
| /users/[id]/scores/favourites/[id] \*1 | DELETE | EMPTY |  | *<score>*  *<url>[url]</url>*  *</score>* |
| /users/[id]/purchases \*1 | GET | EMPTY | 200 | *<purchases count=[count]>*  *<purchase index=[index] id={id]>*  *<product\_key>[product\_key]<product\_key>*  *<date>[date]</date>*  *<user id=[id] />*  *</purchase>*  *</purchases>* |
| /scores | GET | EMPTY | 200 | *<scores count=[count]>*  *<score index=[index] id=[id]>*  *<url>[url]</url>*  *<user id=[id] />*  *</score>*  *…*  *</scores>* |
| /scores/[id] | GET | EMPTY | 200 | *<score id=[id]>*  *<url>[url]</url>*  *<user id=[id] />*  *</score>* |
| DELETE | *<score id=[id]>*  *<url>[url]</url>*  *<user id=[id] />*  *</score>* | 200 | *<score id=[id]>*  *<url>[url]</url>*  *<user id=[id] />*  *</score>* |
| 404 | *<error>*  *<shortcode>SCR-BADUSERID</shortcode>*  *<message>*  *User ID not correct*  *</message>*  *</error>* |
| /purchases | GET | EMPTY | 200 | *<purchases count=[count]>*  *<purchase index=[index] id={id]>*  *<product\_key>[product\_key]<product\_key>*  *<date>[date]</date>*  *<user id=[id] />*  *<status>[status]</status>*  *</purchase>*  *</purchases>* |
| POST | *<purchase index=[index] id={id]>*  *<product\_key>[product\_key]<product\_key>*  *<date>[date]</date>*  *<user id=[id] />*  *</purchase>* | 200 | *<purchases count=[count]>*  *<purchase index=[index] id={id]>*  *<product\_key>[product\_key]<product\_key>*  *<date>[date]</date>*  *<user id=[id] />*  *</purchase>*  *</purchases>* |
| 404 | *<error>*  *<shortcode>PUR-BADUSERID</shortcode>*  *<message>*  *User ID not correct*  *</message>*  *</error>* |
| /purchases/[id] | GET | EMPTY | 200 | *<purchase id={id]>*  *<product\_key>[product\_key]<product\_key>*  *<date>[date]</date>*  *<user id=[id] />*  *<status>[status]</status>*  *</purchase>* |
| PUT | *<purchase>*  *<product\_key>[product\_key]<product\_key>*  *<user id=[id] />*  *<status>[status]</status>*  *</purchase>* | 200 | *<purchase id={id]>*  *<product\_key>[product\_key]<product\_key>*  *<date>[date]</date>*  *<user id=[id] />*  *<status>[status]</status>*  *</purchase>* |
| 404 | *<error>*  *<shortcode>SUB-BADUSERID</shortcode>*  *<message>*  *User ID not correct*  *</message>*  *</error>* |
| DELETE | *<purchase id=[id] />* | 200 | *<purchase id={id]>*  *<product\_key>[product\_key]<product\_key>*  *<date>[date]</date>*  *<user id=[id] />*  *<status>[status]</status>*  *</purchase>* |
| /contact | POST | *<contact>*  *<user id=[id] />*  *<subject>[subject]</subject>*  *<message>[message]</message>*  *</contact>* | 200 | EMPTY |
| 404 | *<error>*  *<shortcode>CON-BADUSERID</shortcode>*  *<message>*  *User ID not correct*  *</message>*  *</error>* |

Note:

\*1 l’URL */users/[id]* et */users/[id]/\** sont équivalent à */users/[username]* et */users/[username]/\**. Ainsi, un pseudonyme ne peut pas être un nombre.

\*2 quand un filtre est appliqué, sa valeur est interprétée comme un englobeur. De plus, si plusieurs filtres sont appliqués, ils sont associatifs.

Exemples:

*/users?uname=boss* retournera tous les utilisateurs dont le pseudonyme commence par *boss* (*boss, boss31, bossy, etc.*).

*/users?uname=boss&fname=fr* retournera tous les utilisateurs dont le pseudonyme commence par *boss* et dont le prénom commence par *fr*.

#### Liste des erreurs

| Short code | Code HTTP | Message | Explication (optionnel) |
| --- | --- | --- | --- |
| *LOG-BADCREDENTIALS* | 401 | *Credentials not valid.* |  |
| *LOG-CLOSEDACC* | 403 | *Account closed* |  |
| *USR-BADFIELD* | 400 | *[field] field not correct* |  |
| *USR-WRONGPASS* | 403 | *Current password not correct* |  |
| *REG-NAMEALRUSED* | 409 | Username already used |  |
| *REG-MAILALRUSED* | 409 | Email address already used |  |
| *CLO-USERALRCLO* | 403 | User account already closed |  |
| *PAS-LINKALRUSED* | 403 | Password link already used |  |
| *PAS-BADLINK* | 404 | Password link not correct |  |
| *SUB-BADUSERID* | 404 | User ID not correct |  |
| *SCO-BADUSERID* | 404 | User ID not correct |  |
| *PUR-BADUSERID* | 404 | User ID not correct |  |
| *CON-BADUSERID* | 404 | User ID not correct |  |

#### Erreurs génériques

Certaines erreurs peuvent être retournées par toutes les requêtes et ne sont donc pas propre à une seule.

| Short code | Code HTTP | Message | Explication (optionnel) |
| --- | --- | --- | --- |
| GLO-MISSING | 400 | The following required parameter are missing: [-param1][-param2]… | Seuls les paramètres manquants et obligatoires peuvent être retournés avec cette erreur |
| GLO-UNAUTHORIZED | 403 | You must be logged to perform this action | Cette erreur est retournée lorsque qu’un utilisateur (connecté ou non) tente d’effectuer une action sur le compte d’un autre utilisateur |
| GLO-RESNFOUND | 404 | The requested resource is not found |  |
| GLO-METHNALLOW | 405 | The method used is not allowed on the resource |  |
| GLO-BADROOT | 415 | *The XML root name is not correct* |  |
| GLO-BADXML | 415 | The requested XML is not correct |  |
| GLO-DBSERERR | 502 | The database server does not respond |  |
| GLO-DBSCHERR | 500 | An SQL error occurred | Cette erreur peut être causée par une anomalie dans le schéma de la base de données. Elle ne devrait pas être retournée en production |
| GLO-INTERNERR | 500 | An unknown error occurred | Cette erreur ne devrait pas être retournée en production |

# Vue données

