API Liberize me

Toute les requetes sous susceptible de renvoyer des erreurs sous la forme : statut 500 :

{

Error : true,

errorInfos : <ERREUR>

}

* Get : /users/connection
  + Descr : connecte l’utilisateur, set un cookie encrypté
  + Param : email, password
  + Return :

{

<INFORMATION UTILISATEUR>

}

* DELETE : /users/connection
  + Descr : déconnecte l’utilisateur, reset le cookie encrypté
  + Param : none
  + Return : none
* Get : /users[ ?offset=value]
  + Descr : Récupère la liste des utilisateurs, par paquets de 20
  + Param : [offset]
  + Return :

statut 200, retourne la liste d’utilisateur. Sous la forme :

{

error : false,

users : [

{

id : <USER\_ID>,

pseudo : <USER\_PSEUDO>

},…

]

}

* Post : /users
  + Descr : Ajoute un utilisateur a la bdd, envoie un mail de confirmation a l’utilisateur
  + Param : email, verifyEmail, password, verifyPassword, pseudo
  + Return :

Utilisateur créé et email envoyé : statut 201, retourne :

{

error : false,

user : <USER>

}

Utilisateur crée MAIS mail non envoyé : statut 500, utilisateur supprimé, retourne :

{

error : true,

errorInfos : <ERREUR>

}

Informations fournies incohérentes : statut 401, retourne :

{

error : true,

errorInfos : « INVALID INFORMATION »

}

* GET : / users/ :id/confirm/ :token
  + Descr : marque l’email de l’utilisateur comme validé
  + Param : none
  + Return :

Utilisateur trouvé ET token valide : statut 200, retourne :

{  
 error: false  
}

Utilisateur trouvé MAIS token invalide : statut 409, retourne :

{  
 error: true,

errorinfos : « NOT AUTHORISED »  
}

Utilisateur non trouvé : statut 404, retourne :

{  
 error: true,

errorinfos : « NOT FOUND »  
}

* GET : /users/ :id
  + Descr : Renvoies les informations d’un utilisateur spécifique
  + Param : none
  + Return :

Utilisateur trouvé : statut 200, retourne :

{

error : false,

user : {

id :<USER\_ID>,

mail : <USER\_MAIL>,

…

}

}

Pas d’utilisateur correspondant : statut 404, reponse sous la forme :

{

error : true,

errorInfos : « not found »

}

* Put : /users/ :id
  + Descr : Modifie les informations d’un utilisateur spécifique
  + Param : email, verifyEmail, password, verifyPassword, pseudo
  + Return :

utilisateur trouvé ET toutes les infos valides: statut 200, retourne :

{

error : false

}

utilisateur non trouvé : statut 404, retourne :

{

error : true,

errorInfos : « NOT FOUND»

}

Informations fournies incohérentes : statut 401, retourne :

{

error : true,

errorInfos : « INVALID INFORMATION »

}

* Get : /users/search[ ?keywords=<VALUE>]
  + Descr : recherche des utilisateurs selon les keywords fournis (séparé par des virgules)
  + Param : none
  + Return :

Requete traité : statut 200, retourne :

{  
 error: false,  
 users: [

{

Id :<USER\_ID>,

…  
}

]

}

* Get : /users/ :id/friends [?offset=value]
  + Descr : renvoie la liste d’amis de l’utilisateur par paquet de 20
  + Param : none
  + Return :

Utilisateur trouvé : statut 200, retourne :

{

error : false,

users : [

{

id : <USER\_ID>

},…

]

}

Utilisateur non trouvé : statut 404, retourne :

{

Error : true,

ErrorInfos : « not found »

}

* Post : /users/ :id/friends/ :id\_friend
  + Descr : Fait une demande d’amis à un utilisateur, si demande déjà effectuée et que la demande est faite dans l’autre sens alors, valide la demande
  + Param : id utilisateur
  + Return :

Utilisateurs trouvés, requête traitée  : statut 200, retourne :

{

error : false

}

Utilisateur trouvé MAIS demande déjà effectuée dans ce sens OU déjà amis : statut 409, retourne :

{

error : true,

errorInfos : « ALREADY EXIST  »

}

Utilisateur(s) non trouvé(s) : statut 404, retourne :

{

error : true,

errorInfos : « NOT FOUND  »

}

* DELETE : /users/ :id/friends/ :id\_friend
  + Descr : supprime un utilisateur de la liste de l’utilisateur OU refuse une demande d’amis
  + Param : none
  + Return :

Utilisateurs trouvés, requête traitée : statut 200, retourne :

{

error : false

}

Utilisateurs trouvés, mais informations incohérentes : statut 409, retourne :

{

error : true,

errorInfos : « INVALID INFORMATION »

}

Utilisateur(s) non trouvé(s) : statut 404, retourne :

{

error : true,

errorInfos : « NOT FOUND »

}

* Get : /users/ :id/products
  + Descr : renvoie la liste des produits de l’utilisateur
  + Param : none
  + Return :

Utilisateur trouvé : statut 200, retourne :

{

error : false,

products : [

{

id : <PRODUCT\_ID>,

EAN: <PRODUCT\_EAN>,

…

},…

]

}

Utilisateur non trouvé : statut 404, retourne :

{

Error : true,

errorInfos: « USER NOT FOUND »

}

* POST : /users/ :id/products/ :id\_product
  + Descr : ajoute un produit a la liste des produits
  + Param : none
  + Return :

Utilisateur ET produit trouvés : statut 200, retourne :

{

error : false

}

Utilisateur non trouvé : statut 404, retourne :

{

error : true,

errorInfos : « USER NOT FOUND »

}

Produit non trouvé : statut 404, retourne :

{

error : true,

errorInfos : « PRODUCT NOT FOUND »

}

* Delete : /users/ :id/products/ :id\_product
  + Descr : supprime un produit a la liste des produits
  + Param : none
  + Return :

Utilisateur trouvé et produit valide : statut 200, retourne :

{  
 error: false   
}

Utilisateur trouvé MAIS pas le produit : statut 403, retourne :

{  
 error: true,  
 errorInfos: "PRODUCT NOT FOUND"  
}

Utilisateur non trouvé  : statut 404, retourne :

{  
 error: true,  
 errorInfos: "USER NOT FOUND"  
}

* Post : /users/ :id/friends/ :id\_friend/products/ :id\_product
  + Descr : ajoute une demande d’emprunt auprés d’un utilisateur
  + Param : none
  + Return :

Utilisateurs trouvés ET produit trouvé/possédé par l’utilisateur  : statut 200, retourne :

{

error : false

}

Utilisateur(s) non trouvé(s): statut 404, retourne :

{

Error : true,

ErrorInfos : « USER NOT FOUND »

}

Produit non existant : statut 404, retourne :

{

error: true,

errorInfos: "PRODUCT NOT FOUND"

}

Produit non possédé par l’utilisateur : statut 403, retourne :

{

Error : true,

ErrorInfos : « PRODUCT NOT AVAILABLE»

}

Utilisateurs trouvés, produit valide MAIS demande déjà effectué et non terminée : statut 409, retourne :

{

error: true,

errorInfos: "ALREADY EXIST"

}

* GET : /products/seach[?EAN=<VALUE>[&keywords=<VALUE>[&type=<VALUE>[&offset=<VALUE>]]]]
  + Descr : retourne la liste des produits en fonction des filtres par paquet de 20, si la recherche n’est pas concluante fait une recherche sur amazon et retourne tous les resultats au format de la bdd
  + Param : none
  + Return :

Requete valide : statut 200, retourne :

{

error : false,

products : [

{

Id : <PRODUCT\_ID>,

…

},…

]

}

Pas de resultants concluant: statut 404, retourne:

{

error: true,

errorInfos: "NOT FOUND"

}

* POST : /products
  + Descr : ajoute un produit a la base de donné
  + Param : title, type, EAN, publisher, author, realease, director, actor, platform, artist, ASIN
  + Return :

Produit ajouté : statut 201, retourne :

{

error : false,

product :{

id : <PRODUCT\_ID>,

EAN : <PRODUCT\_EAN>,

…

}

}

Si EAN ou ASIN non fournis : statut 401, retourne :

{

Error : true,

ErrorInfos : « INVALID INFORMATION »

}

Produit déjà ajouté : statut 403, retourne :

{  
 error: true,  
 errorInfos: "ALREADY EXIST"  
}

* GET : /products/ :id
  + Descr : retourne le detail d’un produit
  + Param :
  + Return :

Produit trouvé : statut 200, retourne :

{

error : false

product :{

id :<PRODUCT\_ID>,

entitled :<PRODUCT\_ENTITLED>,

…

}

}

Produit non trouvé : statut 404, retourne :

{

Error : true,

ErrorInfos : « not found »

}