Documentation API

Groupes

User : Antoine / Antonin-Lesson : Agnès / Alexis - Teacher : Manon / Lilian

Student : Etienne / Valentin - Publication : Nico / Steph - Comment : Sylvain / Tom

**User**

POST /login : se connecter en utilisant un email et un mot de passe

***paramètres :***

***body = { email: ”***[***email@test.com***](mailto:email@test.com)***”, password: “fdsfsdfd’ }***

***Réponses :***

***200 (succès) =  {id:1, email :”***[***email@test.com***](mailto:email@test.com)***”, password : “fdsfsdfd’ , StudentId : null or id , TeacherId : null or id, token:”g54trh46yt41j8u4y”}***

***resp = { user : {},***

***student : {},***

***token }***

***400 (La syntaxe de la requête est erronée )***

***401 (mot de passe incorrect )***

***404 (email ne correspond pas )***

POST /register  : créer un nouveau compte avec un email et un mot de passe

***paramètres :***

***body  = { email: ”***[***email@test.com***](mailto:email@test.com)***”, password: “fdsfsdfd’, type: 1 }***

***Réponses :***

***200 (succès) =  {id:1, email :”***[***email@test.com***](mailto:email@test.com)***”, password : “fdsfsdfd’ , StudentId : null or id , TeacherId : null or id, token:”g54trh46yt41j8u4y”}***

***400 (syntaxe de la requête est erronée )***

***409 (email déjà utilisé )***

/////// OPTION

PUT me/modify-email/ : modifier l’email

***paramètres :***

***body  = { email :”***[***email@test.com***](mailto:email@test.com)***” }***

***headers[x-access-token] = token;***

***explication : Le token est la garantie que l’utilisateur est authentifié. On peut également extraire l’id du User à partir du token pour l’identifier.***

***Réponses :***

***200 (succés) =  {email :”***[***email@test.com***](mailto:email@test.com)***”}***

***401 (utilisateur non authentifié )***

***//Option***

PUT me/modify-password/ : modifier le mot de passe

***paramètres :***

***body  = { old\_password :”dsdsd” , new\_password:”dsqdqsdqs”}***

***headers[x-access-token] = token;***

***explication : Le token est la garantie que l’utilisateur est authentifié. On peut également extraire l’id du User à partir du token pour l’identifier.***

***Réponses :***

***200 (succès) =  {“Votre password a bien été modifié}***

***401 (utilisateur non authentifié )***

DELETE me/delete/ suppression d’un utilisateur

***paramètres :***

***headers[x-access-token] = token;***

***explication : Le token est la garantie que l’utilisateur est authentifié. On peut également extraire l’id du User à partir du token pour l’identifier.***

***Réponses :***

***200 (succès) =  {“Votre profil a été supprimé”}***

***401 (utilisateur non authentifié )***

GET /me , afficher les information de l’utilisateur authentifié

***paramètres :***

***headers[x-access-token] = token;***

***Réponses :***

***200 (succès) =  {student : {}  }***

***ou***

***200 (succès) =  {teacher : {}  }***

***401 (utilisateur non authentifié )***

**POST /me          créer un student / teacher à partir du body de ma requête**

* paramètres :  
  req.headers[‘x-access-token’] : token , req.body :{...student};
* réponse :

***200 (succès)***

body = {...student}

***401 (utilisateur non authentifié )***

**PUT /me        modifier le student / teacher dont l’id est celui passé en paramètre**

* paramètres :  
  req.headers[‘x-access-token’] : token , req.body :{...student};
* réponse :

***200 (succès)***

body = {...Student}

***401 (utilisateur non authentifié )***

**Publication**

POST /publications/ :un user, connecté, enregistre 1 publi, ajoute 1 cours, écrit 1 comment.

• req.body = { creation\_date:”2021-01-08 12:00”, title:”titre”, body\_text:”fdsfsdf”, type:1 }

• req.body = { creation\_date:”2021-01-08 12:00”, title:”titre”, body\_text:”fdsfsdf”, type:1,lesson\_id:id }

•  req.header = { token:string }

• res.status(200)

   res.json({id:id, creation\_date:”2021-01-08 12:00”, title:”titre”, body\_text:”fdsfsdf”, type:1,userId:id })

 • res.status(500) erreur serveur

 • res.status(401) identifiez-vous

PUT /publi/id:un utilisateur, connecté, modifie une de ses publications et le cours associé.

•  req.params.id = { id publication }

• req.body = { creation\_date:”2021-01-08 12:00”, title:”titre”, body\_text:”fdsfsdf”, type:1 }

• req.body = { creation\_date:”2021-01-08 12:00”, title:”titre”, body\_text:”fdsfsdf”, type:1,lesson\_id:id }

•  req.header = { token:string }

• res.status(200)

   res.json({id:id, creation\_date:”2021-01-08 12:00”, title:”titre”, body\_text:”fdsfsdf”, type:1,userId:id })

 • res.status(500) erreur serveur

 • res.status(401) identifiez-vous

DELETE /publications/id :un utilisateur, connecté, supprime une de ses publications.

•  req.params.id = { id publication }

•  req.header = { token:string }

• res.status(200) succès

• res.status(500) erreur serveur

• res.status(401) identifiez-vous

GET /publi/id :un utilisateur consulte une publication.

•  req.header = { token:string }

response :

• res.status(200) succés {publication, comments =[{...comment},{...comment}]

• res.status(500) erreur serveur

 • res.status(401) identifiez-vous

GET /publications : un utilisateur authentifié récupère toutes les publications

•  req.header = { token:string }

• res.status(200) succés

• res.status(500) erreur serveur

 • res.status(401) identifiez-vous

**Student**

**GET /students             récupérer tous les students existants**

* paramètres :

req.headers[‘x-access-token’] : token

req.params.id : id du student

* réponse :

***200 (succès)***

res.body  = [{...student},{...student},{...student} ]

***400 (aucun student )***

***401 (utilisateur non authentifié )***

**GET /students/id         récupérer le student dont l’id est passé en paramètre de mon url**

* paramètres :  
   req.headers[‘x-access-token’] : token

req.params.id : id du student

* réponse :

***200 (succès)*** res.body  = {...Student}

***400 (no student found)***

***401 (utilisateur non authentifié )***

**//Option**

**POST /friendship/id        si l’utilisateur connecté est student , ajouter le student avec l’id comme ami.**

* paramètres :  
   req.headers[‘x-access-token’] : token

req.params.id : id du student

* réponse :

***200 (succès)*** res.body  = {...Student}

***400 (no student found)***

***401 (utilisateur non authentifié )***

**//Option**

**DELETE /friendship/id       si l’utilisateur connecté est student , supprimer le “student” avec l’id de la liste des amis**

* paramètres :  
   req.headers[‘x-access-token’] : token

req.params.id : id du student

* réponse :

***200 (succès)*** res.body  = {...Student}

***400 (no student found)***

***401 (utilisateur non authentifié )***

**//Option**

**POST /enroll/id       si l’utilisateur connecté est student , s’inscrire dans “lesson” avec l’id**

* paramètres :  
   req.headers[‘x-access-token’] : token

req.params.id : id du student

* réponse :

***200 (succès)*** res.body  = {...Student}

***400 (no student found)***

***401 (utilisateur non authentifié )***

**Comment**

//option

**GET** /comments/:id récupérer un commentaire précis. **(selon son id)**

• **Paramètre**:

* req.headers[‘x-access-token’]

body { **id** : 1(req.params.id)}

• **Réponses**:

***200*** *(succés) =  {****id****:1,* ***creation\_date*** *: “*2021.01.13”, **body\_text**: “texte du commentaire*}*

***401*** *( Veuillez vous connectez afin de pouvoir avoir accès à vos commentaire)*

***404****(Le commentaire demandé n’existe pas (id introuvable))*

***500*** *(Un problème est survenu lors de la collecte de commentaire )*

**POST** /comments/publication/:id  rédiger un commentaire sous une publication.

• **Paramètre**:

* req.headers[‘x-access-token’] : token

req.params.id : id de la publication

body { **creation\_date** :  “2021.01.13”, **body\_text** : “texte du commentaire”}

• **Réponses**:

***200*** *(succés) =  {****id****: 1,* **creation\_date** :  “2021.01.13”, **body\_text** : “texte du commentaire”*}*

***400****(erreur: paramètre manquant ou non approprié)*

***500*** *(Un problème est survenu lors de la création de commentaire )*

*401 utilisateur non authentifié*

//option

**PUT** /comments/:id mettre à jour un commentaire précis **(selon son id)**

• **Paramètre**:

body { **creation\_date** :  “2021.12.05”, **body\_text** : “texte du commentaire”}

• **Réponses**:

***200*** *(succès) =  La modification à bien été prise en compte.*

***400****(erreur: paramètre manquant ou non approprié)*

***401*** *(Veuillez vous connectez afin de poster un nouveau commentaire)*

***500*** *(Un problème est survenu lors de la mise à jour de la  création de commentaire )*

**DELETE** /comments/:id supprimer un commentaire existant **(selon son id)** le rédacteur du commentaire est le seul à pouvoir le supprimer

• **Paramètre**:

* req.headers[‘x-access-token’] : token

body { **id** : 1(req.params.id)}

• **Réponses**:

***200*** *(succès) =  La suppression a bien été prise en compte.*

***401*** *(Veuillez vous connectez afin de pouvoir supprimer un commentaire)*

***404****(Le commentaire demandé n’existe pas (id introuvable))*

***500*** *(Un problème est survenu lors de la suppression de commentaire)*

// on suppose que nos Lessons sont accessibles par tous.

**Lesson :**

**#GET**        /Lessons        : Obtenir la liste des cours

-Paramètres :

Aucun

- Réponses :

**200** : OK  = Liste de Lessons

**400** : Bad request ->

**404** : Not Found -> cours non trouvé

**#POST**      /Lessons        : Créer un nouveau cours

-Paramètres :

body : {

title: “Petit cours de SVT oklm”,

hours: 10,

description: “Explication des relations entre chimpanzés”,

file\_name: “chinpanzésSmiling.png”,

starting\_date: 15/01/2021 ,

ending\_date: 15/01/2021

}

* req.headers[‘x-access-token’] : token

-Réponses :

200 : OK  =  {

id:33,

title: “Petit cours de SVT oklm”,

hours: 10,

description: “Explication des relations entre chimpanzés”,

file\_name: “chinpanzésSmiling.png”,

starting\_date: 15/01/2021 ,

ending\_date: 15/01/2021

}

400 : Bad request ->

401 : utilisateur non authentifié

**#PUT**         /Lessons/:id        : Modifier un cours

-Paramètres :

req.params.id : {id: 12}

body : {

title: “Petit cours de SVT MODIFIED”,

hours: 10,

description: “Explication des relations entre chimpanzés MODIFIED”,

file\_name: “chinpanzésSmiling.png”,

starting\_date: 15/01/2021 ,

ending\_date: 15/01/2021

}

* req.headers[‘x-access-token’] : token

-Réponses :

**200** : OK  =  {

id:33,

title: “Petit cours de SVT oklm”,

hours: 10,

description: “Explication des relations entre chimpanzés”,

file\_name: “chinpanzésSmiling.png”,

starting\_date: 15/01/2021 ,

ending\_date: 15/01/2021

}

**400** : Bad request ->

**404** : Not Found -> cours non trouvé {error : “ce cours n’existe pas”}

**401** : utilisateur non authentifié

**#DELETE**  /Lessons/:id        : Supprimer un cours par son rédacteur

-Paramètres :

req.params.id : {id: 12}

* req.headers[‘x-access-token’] : token

-Réponses :

**200** : OK  = {success : “Le cours a été supprimé ! ”}

**400** : Bad request ->

**404** : Not Found -> cours non trouvé {error : “ce cours n’existe pas”

**Teacher :**

**GET /teachers             récupérer tous les teachers existants**

* paramètres :
  + req.headers[‘x-access-token’] : token ,
* réponse :

***200 (succès)***

res.body  = [{...teacher},{...teacher},{...teacher} ]

***400 (aucun teacher )***

***401 (utilisateur non authentifié )***

**GET /teachers/id         récupérer le teacher dont l’id est passé en paramètre de mon url**

* paramètres :  
  req.headers[‘x-access-token’] : token , req.params.id : id du teacher
* réponse :

***200 (succès)*** res.body  = {...teacher}

***400 (no student found)***

***401 (utilisateur non authentifié )***