Drilllight API Linux :

Cette api, est une copie de l’ancienne API, basée sur Firebase, sous la DataBase PostGreSql, les end points sont les mêmes en terminologie, des accès Url ont été classifiés, selon les groupes distincts des différentes requêtes

l’url du back en Linux est : https://devserver.drilllight.com/

/account, /econes, /exercise, /firmware, /result, /token

il s'agit donc de passer comme chemin d'accès :

Ex: <https://devserver.drilllight.com/account/createAccount>

Lorsque vous souhaitez vous connecter être authentifié à l’API, vous devrez avant tout envoyer une requête à :

/getToken

le retour de token correspond à ce type de valeur:

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJoYXNoIjoiQmFzaWMgWTI5dGNIUmxaR1Z0YjBCbmJXRnBiQzVqYjIwNlYyVnNZMjl0WlRBeElRPT0iLCJpZCI6IjczNGRiOWFkLTE4NzYtNDVlNS04MDRiLWZlMTE0MjU1MzdiNiIsImRhdGUiOiIyMDI0LTA2LTEzVDA4OjExOjU1LjQxOVoiLCJrZXkiOiI5YjM2YzMyNC01ZDk3LTRjZDktOGU4NC1mY2I5MDFlYzQ4NTAiLCJpYXQiOjE3MTgyNjYzMTksImV4cCI6MTcxODQzOTExOX0.bNlz-e06Oi0BIOxBYzGTDMnptiiggZwWkzeBbZMQIfk 734db9ad-1876-45e5-804b-fe11425537b6

ou

/validateToken

**/!\ Le token est valide 30 Jours**

Si votre token est expiré vous recevrez le return suivant:

**Return:**

return res.status(200).json({

response: {

result:'expiredTokenError',

message:'Votre token a expiré'

}

});

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

—---------------------------------------------------------------------------------------------------------------------

1. account :

ce chemin d'accès contient les end points suivants :

###########################################################################

/createAccount

Method: POST

Header: token // voir exemple de token en première page

—---------------------------------------------------------------------------------------------------------------------

Body: JSON :

data: {

id: "ab472b3a-f5ce-4ce9-aabe-b551be5ca12d",

owner:"00f072a8-a25c-49e1-b147-07406a7ad4c3", // Id du owner

role:(admin,owner,staff,user),

email: [pierrekiroul@gmail.com](mailto:pierrekiroul@gmail.com), // Obligatoire

passwordHash:"",

firstName: Pierre, // Obligatoire

familyName: Kiroul, // Obligatoire

fullName: (concaténation de firstName + ‘ ’ + familyName),(auto)

avatarURL: avatarURL, // (base 64 256x256 jpg)

personalInfo: {

birthdate:"2004-06-11T13:46:30.686Z",

simpleBirthdate: **"11/06/2004"**,

address1: "9 place John Rewald",

address2: "appt 11",

zip: "13100",

city: "Aix en provence",

region: "France",

phone: "0778988877",

comment: ""

},

privileges: {

rights: []

},

users:[],

privateExercisesChangeCount:0,

staff: [{id: },..],// tableau d’objets, id figure de repère

econes: [], // tableau d’objets, id figure de repère

trainings: ["c1d48602-da5a-46c8-9ea1-11628b267001",...],// id exos

videos: ["d45dd654-yt5b-46c8-7sa1-83654b262257",...], // id vidéos

licensed: 10, // number

warning:false, // boolean

date:DateString,

dateIso:"2024-06-13T13:46:30.686Z",

update:"13/06/2024",

privateFirmwareId:"", // id sans firmware privé si vide

updateIso:"2024-06-13T14:46:30.686Z",

privateOnly:true // si uniquement des exercices privés

}

les champs obligatoires sont email, firstName, familyName

**Returns:**

—---------------------------------------------------------------------------------------------------------------------

Si l’Id envoyé ne correspond à aucun utilisateur

return res.status(200).json({

response: {

result:'noUserError',

message:''

}

});

Si les champs dans le corps de la requête ne sont pas aux bons formats

return res.status(200).json({

response: {

result:'errorBodyRequest',

message:''

}

});

**Si l’utilisateur à bien été créé 👍la réponse retourne l’objet du nouvel utilisateur**

il peut être role : admin,owner, staff ou user !

res.status(200).json({

response: {

result: 'success',

message: ''

},

account:(compte créé identique à data☝️ voir au dessus)

});

###########################################################################

/updateAccount

Method: POST

Header:

token // voir exemple de token en première page, id:”ab472b3a-f5ce-4ce9-aabe-b551be5ca12d”

—---------------------------------------------------------------------------------------------------------------------

Body: JSON :

data: {

owner:"00f072a8-a25c-49e1-b147-07406a7ad4c3", // Id du owner

role:(admin,owner,staff,user),

email: [pierrekiroul@gmail.com](mailto:pierrekiroul@gmail.com),

passwordHash:"",

firstName: Pierre,

familyName: Kiroul,

fullName: (concaténation de firstName + ‘ ’ + familyName),(auto)

avatarURL: avatarURL, // (base 64 256x256 jpg)

users:[],

privateExercisesChangeCount:0,

staff: [{id: },..],// tableau d’objets, id figure de repère

econes: [], // tableau d’objets, id figure de repère

trainings: ["c1d48602-da5a-46c8-9ea1-11628b267001",...],// id exos

videos: ["d45dd654-yt5b-46c8-7sa1-83654b262257",...], // id vidéos

},

id:"ab472b3a-f5ce-4ce9-aabe-b551be5ca12d" // Obligatoire

**Returns:**

—---------------------------------------------------------------------------------------------------------------------

Si l’Id envoyé ne correspond à aucun utilisateur

return res.status(200).json({

response: {

result:'noUserError',

message:''

}

});

Si les champs dans le corps de la requête ne sont pas aux bons formats

return res.status(200).json({

response: {

result:'errorBodyRequest',

message:''

}

});

**Si la requête est effectuée avec succès**

return res.status(200).json({

response: {

result:'success',

message:''

},

account: l’objet account entier voir au dessus ☝️

});

###########################################################################

/getAccountDetails

Method: GET

Header:

token // voir exemple de token en première page, id:”ab472b3a-f5ce-4ce9-aabe-b551be5ca12d”

—---------------------------------------------------------------------------------------------------------------------

**Returns:**

—---------------------------------------------------------------------------------------------------------------------

Cette requête renvoie tous les détails du compte sélectionné grâce à son ID, mise à part son passwordHash qui est supprimé lors chacune des réponses

Chaque staff ou user lié à ce compte, est ajouté au tableau associé

staff.push({

fullName:"Cedric Paslotre",

familyName:"Paslotre",

firstName:"Cedric",

id:"fb96841b-f55d-4d48-8f23-8603307f1",

email:"cedricpalotre@gmail.com",

role:"owner" or "admin",

validate: true or false // si le mot de passe est créé

})

users.push({...})

**Si la requête est effectuée avec succès 🥳**

return res.status(200).json({

response: {

result:'success',

message:''

},

account: objet enrichi account est envoyé🫣(voir createAccount)

});

Si aucun compte ne correspond à l’id envoyé le retour suivant apparaît

return res.status(200).json({

response: {

result:'noAccountError',

message:''

}

});

###########################################################################

/getAccountsList

Method: GET

Header:

token // voir exemple de token en première page, id:”ab472b3a-f5ce-4ce9-aabe-b551be5ca12d”

—---------------------------------------------------------------------------------------------------------------------

**Returns:**

—---------------------------------------------------------------------------------------------------------------------

Lorsque c’est un **admin** qui effectue cette requête, tous les comptes “admin” et “owner”, sont retournés dans un tableau, nommé accounts, il est enregistré avec les données de chacun des comptes comme ci-dessous

accounts.push({

fullName:"Pierre Kiroul",

familyName:"Kiroul",

firstName:"Pierre",

id:"ab472b3a-f5ce-4ce9-aabe-b551be5ca12d",

email:"pierrekiroul@gmail.com",

role:"owner" or "admin",

validate: true or false // si le mot de passe est créé

})

**Les comptes visibles sont envoyés ainsi que le message succès**

return res.status(200).json({

response: {

result:'success',

message:''

},

accounts: accounts, // le tableau empli des comptes

});

Lorsque c’est un **owner**, l’Api récupère les utilisateurs, liés à son propre compte selon le même modèle du tableau accounts que pour la demande en tant qu’*admin ! 👍*

**ainsi les comptes visibles sont envoyés ainsi que le message succès**

return res.status(200).json({

response: {

result:'success',

message:''

},

accounts: accounts, // le tableau empli des comptes

});

Lorsque c’est un **staff**, l’Api récupère les utilisateurs du compte propriétaire,son **owner** lié à son propre compte,le même modèle du tableau accounts est envoyé en réponse comme pour la demande en tant qu’*admin ou owner! 👍!!*

**ainsi les comptes visibles sont envoyés ainsi que le message succès**

return res.status(200).json({

response: {

result:'success',

message:''

},

accounts: accounts, // le tableau empli des comptes

});

Si le rôle, ou l’id n’existe pas en base alors le retour suivant apparaît

return res.status(200).json({

response: {

result:'noAccountError',

message:''

}

});

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

—---------------------------------------------------------------------------------------------------------------------

2. /econes:

ce chemin d'accès contient les end points suivants :

###########################################################################

/createEcone

Method: POST

Header:

token // voir exemple de token en première page,

—---------------------------------------------------------------------------------------------------------------------

Body:

**Returns:**

###########################################################################

/getEconeDetails

Method: GET

Header:

token // voir exemple de token en première page,

id

—---------------------------------------------------------------------------------------------------------------------

**Returns:**

—---------------------------------------------------------------------------------------------------------------------

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

—---------------------------------------------------------------------------------------------------------------------

3. /exercise:

ce chemin d'accès contient les end points suivants :

###########################################################################

/createExercise

/!\ *Lors d’un create, si l’exercice est public, le script se charge de vérifier que le* ***json*** *est dans le bon format, aussi il incrémente un champ dans une table permettant une mise à jour, rapide si de nouveaux exercices ont été ajoutés !*

/!\ *Dans l’autre cas, si l’exercice est en mode privé, il incrémente un champ dans l’objet du compte de l’utilisateur pour lequel cette exercice est sélectionné*

Method: POST

Header:

token // voir exemple de token en première page,

id ”ab472b3a-f5ce-4ce9-aabe-b551be5ca12d”

—---------------------------------------------------------------------------------------------------------------------

Body: JSON :

👇 voir exemple *[Le format doit être respecté selon l’ordre, que ce soit dans la clef d’objet* ***header*** *ou dans* ***steps****]*

json : {

"header": {

"json\_version": 3,

"id": "274ef5d7-2456-444a-aa4a-3fe9ddcbgero",

"title": "Appui et Équilibre 1 E-Cone",

"description": "Un.",

"image": "",

"author": {

"id": "ab472b3a-f5ce-4ce9-aabe-b551be5ca12d",

"fullname": "gero"

},

"editor": {

"id": "ab472b3a-f5ce-4ce9-aabe-b551be5ca12d",

"fullname": "gero"

},

"owner": {

"id": "ab472b3a-f5ce-4ce9-aabe-b551be5ca12d",

"fullname": "gero"

},

"version": 24,

"status": "private",

"created": "2024-04-29T15:02:22+02:00",

"modified": "2024-04-29T15:02:22+02:00",

"tags": [],

"thematic": [],

"sports": [],

"category": [],

"age\_category": [],

"level": "",

"equipments": [],

"mode": "time",

"time": 60,

"countdown": true,

"random\_step": false,

"pods\_layout": [{ "name": "E-Cone 1", "x": 0, "y": -20, "z": 0}]

},

"steps": [

{

"timer": 4,

"step\_mode": "OR",

"pods": [

{

"conditions": [],

"actions": [

{

"led": "onright 4 blu"

}

]

}

]

},

{

"timer": 2,

"step\_mode": "OR",

"pods": [

{

"conditions": [],

"actions": [

{

"led": "onleft 2 red"

}

]

}

]

}

}

]

}

]

}

**Returns:**

—---------------------------------------------------------------------------------------------------------------------

**Si tout est correct ! 🤔**

return res.status(200).json({

response: {

result:'success',

message:''

},

json:json de l’exercice ,

idUser:identifiant de l’utilisateur

});

note: idUser est utilisé pour le private

dans le cas contraire ! 😡

return res.status(200).json({

response: {

result:errorJsonExport,

message:'veuillez vérifier le format de l’exercice'

}

});

###########################################################################

/updateExercise

Method: POST

Header:

token // voir exemple de token en première page,

id: ”ab472b3a-f5ce-4ce9-aabe-b551be5ca12d”

—---------------------------------------------------------------------------------------------------------------------

Body: JSON :

json : {

"header": {...}, ( se référer à createExercise )

"steps": [...]

}

**Returns:**

—---------------------------------------------------------------------------------------------------------------------

**Si tout est correct ! 🤔**

return res.status(200).json({

response: {

result:'success',

message:''

},

json:json de l’exercice ,

idUser:identifiant de l’utilisateur

});

dans le cas contraire ! 😡

return res.status(200).json({

response: {

result:errorJsonExport,

message:'veuillez vérifier le format de l’exercice'

}

});

###########################################################################

/getExercisesList

Cette requête renvoie les exercices associés à un compte, tous les exercices, rattachés à celui-ci

Le script vérifie si de nouveaux exercices sont disponibles 👏, aussi si **le tableau trainings de l’utilisateur contient des id,** si c’est le cas, l’on récupère uniquement les identifiants associés contenus dans ce tableau ! 🤗

Method: GET

Header:

token // voir exemple de token en première page,

id: ”ab472b3a-f5ce-4ce9-aabe-b551be5ca12d”

publicExercisesChangeCount: 0

privateExercisesChangeCount: 0

**Returns:**

**IMPORTANT /!\ 😲:**

Deux tableaux biens distincts sont renvoyés lors de cet appel 👍

publicExercises et privateExercises

S'ils sont vides c’est qu’aucun changement en base pour cet utilisateur n’a été effectué l’on retourne publicChanged et privateChanged vide!🤷

—---------------------------------------------------------------------------------------------------------------------

Si l’utilisateur est en mode privateOnly, mais qu’il n’a pas d’exercice en privé

return res.status(200).json({

response: {

result:'errorNoPrivateExercise',

message:'Aucun exercice privé associé'

}

});

**Si tout est correct, aussi et qu’il n’y à de changement que pour les exercices privés! 🤔🙂**

**privateChanged = true, son tableau est constitué des différents json des exercices**

return res.status(200).json({

response: {

result:'success',

message:''

},

publicExercises:[],

privateExercises:[...],

publicChanged:false,

privateChanged:true,

publicExercisesChangeCount:4,

privateExercisesChangeCount:3,

idUser:”ab472b3a-f5ce-4ce9-aabe-b551be5ca12d”

});

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

—---------------------------------------------------------------------------------------------------------------------

4. /firmware:

ce chemin d'accès contient les end points suivants :

###########################################################################

/createFirmware

Method: POST

Header:

token // voir exemple de token en première page,

—---------------------------------------------------------------------------------------------------------------------

**Returns:**

—---------------------------------------------------------------------------------------------------------------------

###########################################################################

/updateGlobalFirmware

Method: POST

Header:

token // voir exemple de token en première page,

—---------------------------------------------------------------------------------------------------------------------

Body: JSON : data: {\*email: [xxx@xxx.xx](mailto:xxx@xxx.xx)}, id: abcdefghijklmnopqrstuvwxyz

**Returns:**

—---------------------------------------------------------------------------------------------------------------------

###########################################################################

/getFirmware

Method: GET

Header:

token // voir exemple de token en première page,

—---------------------------------------------------------------------------------------------------------------------

**Returns:**

—---------------------------------------------------------------------------------------------------------------------

###########################################################################

/getFirmwaresList

Method: GET

Header:

token // voir exemple de token en première page,

—---------------------------------------------------------------------------------------------------------------------

**Returns:**

—---------------------------------------------------------------------------------------------------------------------

###########################################################################

/getFirmwareDetails

Method: GET

Header:

token // voir exemple de token en première page,

—---------------------------------------------------------------------------------------------------------------------

**Returns:**

—---------------------------------------------------------------------------------------------------------------------

###########################################################################

/getGlobalFirmware

Method: GET

Header:

token // voir exemple de token en première page,

—---------------------------------------------------------------------------------------------------------------------

**Returns:**

—---------------------------------------------------------------------------------------------------------------------

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

—---------------------------------------------------------------------------------------------------------------------

5. /result:

ce chemin d'accès contient les end points suivants :

###########################################################################

/createResult

Method: POST

Header:

token // voir exemple de token en première page,

—---------------------------------------------------------------------------------------------------------------------

Body: JSON : data: {\*email: [xxx@xxx.xx](mailto:xxx@xxx.xx)}, id: abcdefghijklmnopqrstuvwxyz

**Returns:**

###########################################################################

/getResultList

Method: GET

Header:

token // voir exemple de token en première page,

—---------------------------------------------------------------------------------------------------------------------

**Returns:**

—---------------------------------------------------------------------------------------------------------------------

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

—---------------------------------------------------------------------------------------------------------------------

6. /token:

ce chemin d'accès contient les end points suivants :

###########################################################################

/getToken

Method: GET

Header: passwordhash, username

—---------------------------------------------------------------------------------------------------------------------

**Returns:**

—---------------------------------------------------------------------------------------------------------------------

si aucun compte trouvé :

res.status(200).json({

response: {

result:'errorNoAccount',

message:'aucun utilisateur'

}

});

Si le compte de l’utilisateur est suspendu:

res.status(200).json({

response: {

result:'errorBlockedAccount',

message:''

}

});

Si le mot de passe est incorrect:

res.status(200).json({

response: {

result:'invalidPasswordError',

message:'Mot de passe incorrect'

}

});

**Si la requête est aboutie en totalité:**

res.status(200).json({

response: {

result:'success',

message:''

},

token:token,

id:**ID DE L’UTILISATEUR**

});

});

—---------------------------------------------------------------------------------------------------------------------

###########################################################################

/validateToken

Method: GET

Header:

token // voir exemple de token en première page,

id: ”ab472b3a-f5ce-4ce9-aabe-b551be5ca12d”

—---------------------------------------------------------------------------------------------------------------------

**Returns:**

—---------------------------------------------------------------------------------------------------------------------

Si il n’y a pas d’utilisateur avec cet ID!

res.status(200).json({

response: {

result:'errorNoAccount',

message:'aucun utilisateur'

}

});

**Votre token est valide !**

res.status(200).json({

response: {

result:'success',

message:''

}

});

—---------------------------------------------------------------------------------------------------------------------

###########################################################################

/passwordHash

Method: GET

Header:

passwordhash,

username (email)

—---------------------------------------------------------------------------------------------------------------------

**Returns:**

—---------------------------------------------------------------------------------------------------------------------