Comment utiliser l'API:

Pour utiliser l'API, il faut passer par la fonction **fetch**. Voici comment l'utiliser :

```
let response = await fetch(`{serverUrl}/${endpoint}`, {
        method: 'POST', //GET, POST, DELETE
        headers: {
            'Content-Type': 'application/json',
            // 'Authorization': `Bearer ${token}`,
        }
        body: JSON.stringify(data),  // lire la suite pour comprendre qu'est ce que data
});

let res = response.json();  // Il faut convertir la réponse en json pour pouvoir l'exploiter
```

Où **endpoint** prend l'une des valeurs suivantes :

- Pour la création d'un utilisateur :

    auth/signup

- Pour connecter un utilisateur :

    auth/login

- Pour supprimer un utilisateur :

    auth/delete-account

- Pour envoyer un mail de réinitialisation du mot de passe à un utilisateur :

    auth/ reset-password

- Pour réinitialiser le mot de passe d'un utilisateur:

    auth/reset-password- confirmation

- Pour récupérer le profile d'un utilisateur:

    user/profile/${login}

- Pour changer l'email d'un utilisateur:

    user/update/email/

- Pour changer le login d'un utilisateur:

    user/update/login/

- Pour changer le mot de passe d'un utilisateur:

    user/update/password/

- Pour changer l'age d'un utilisateur:

    user/update/age/

- Pour changer la bio d'un utilisateur:

    user/update/bio/

- Pour changer le score d'un utilisateur:

    user/update/score/

- Pour changer le nombre de coin (argent) d'un utilisateur:

    user/update/coin

- Pour changer la liste d'émoji d'un utilisateur:

    user/update/list-emoji/

- Pour changer la photo de profile d'un utilisateur:

    user/update/profile-picture/

- Pour récupérer le tableau de score des 10 meilleurs joueurs du jeu:

    user/score- board

- Pour récupérer le tableau de score des amis d'un utilisateurs:

    user/friend/score-board/${login}

- Pour ajouter un ami à un utilisateur:

    user/friend/add/

- Pour supprimer un ami d'un utilisateur:

    user/friend/delete/

- Pour insérer une partie fini dans la base de donnée:

    game/new

- Pour récupérer l'historique de partie d'un utilisateur:

game/history/${login}

Comment passer les paramètres ?

Les requêtes de type HTTP POST ou DELETE doivent spécifier le champ **data** dans le body (les requêtes HTTP GET non pas besoin du body). Voici les fonctionnalitées de l'api nécessitant une requete HTTP POST ou DELETE et leurs **data** associés:

- auth/signup :

data = {

email: EMAIL // optionnel,

login: LOGIN ,
password: PASSWORD,

}

- auth/login :

data = {

login: LOGIN,

password: PASSWORD,

}

- auth/reset-password :

data = {

email: EMAIL,

}

- auth/reset-password-confirmation :

data = {

email: EMAIL,

password: PASSWORD,

code: CODE,

}

-auth/delete-account :

data = {

login: LOGIN

password: PASSWORD

}

- user/update/email/ :

data = {

login: LOGIN,

newEmail: EMAIL,

}

- user/update/login/ :

data = {

login: LOGIN,

newLogin: NEW_LOGIN,

}

- user/update/password/ :

data = {

login: LOGIN,

newPassword: PASSWORD,

}

- user/update/age/ :

data = {

login: LOGIN,

newAge: AGE,

}

- user/update/bio/ :

data = {

login: LOGIN,

newBio: BIO,

}

- user/update/score/ :

data = {

login: LOGIN,

newScore: SCORE,

}

- user/update/coin :

data = {

login: LOGIN,

newCoin: COIN,

}

- user/update/list-emoji/ :

data = {

login: LOGIN,

newListEmoji: LISTEMOJI,

}

- user/update/profile-picture/ :

data = {

login: LOGIN,

newProfilePicture: PROFILE_PICTURE,

}

- user/friend/add/ :

data = {

login: LOGIN,

friendLogin: FRIEND_LOGIN,

}

- user/friend/delete/ :

data = {

login: LOGIN,

friendLogin: FRIEND_LOGIN,

}

- game/new :

data = {

playersStats: PLAYERS_STATS,

selectedMode: SELECTED_MODE,

accessMode: ACCESS_MODE,

createAt: CREATE_AT,

finishedAt: FINISHED_AT

}

Les fonctionnalités suivantes ont besoins de l'option 'Authorization' dans le headers pour fonctionner :

- auth/delete-account

– user/profile/LOGIN

– user/update/email/

- user/update/login/

- user/update/password/

- user/update/age/
- user/update/bio/
- user/update/score/

- user/update/coin
- user/update/list-emoji/
- user/update/profile-picture/

- user/friend/add/
- user/friend/delete/

Le **token** est récupérer via la fonctionnalité de connection (auth/login).

Quelles sont les réponses renvoyés par l'API ?

Si endpoint = auth/signup

res = {

status: 200 (réussite) ou 400 (échec de la requête) ou 409 (Utilisateur déjà existant),

error: ERROR (s'il y en a)

}

Si endpoint = auth/login

res = {

status: 200 (réussite) ou 400 (échec de la requête) ou 404 (Utilisateur non existant) ou 401 (Le mot de passe neconcorde pas),

token: TOKEN,

error: ERROR (s'il y'a)

}

Si endpoint = auth/delete-account

res = {

status: 200 (réussite) ou 400 (échec de la requête) ou 404 (Utilisateur non existant) ou 401 (Le mot de passe ne concorde pas),

error: ERROR (s'il y a)

}

Si endpoint = auth/reset-password

res = {

status: 200 (réussite) ou 400 (échec de la requête) ou 404 (aucun utilisateur ne possède cet email),

error: ERROR (s'il y a)

}

Si endpoint = auth/reset-password-confirmation

res = {

status: 200 (réussite) ou 400 (échec de la requête) ou 404 (aucun utilisateur ne possède cet email) ou 401 (Le code ne concorde pas),

error: ERROR (s'il y a)

}

Si endpoint = user/profile/LOGIN

res = {

status: 200 (réussite) ou 400 (échec de la requête) ou 404 (aucun utilisateur),

error: ERROR (s'il y a en)

userProfile: USER_PROFILE

}

Si endpoint = user/update/email/

res = {

status: 200 (réussite) ou 400 (échec de la requête) ou 404 (aucun utilisateur) ou 'P2002' (lorsque l'email que l'on veut changer est déjà pris par un autre utilisateur),

error: ERROR (s'il y en a)

}

Si endpoint = user/update/login/

res = {

status: 200 (réussite) ou 400 (échec de la requête) ou 404 (aucun utilisateur) ou ‘P2002’ (lorsque le login que l’on veut changer est déjà pris par un autre utilisateur),

error: ERROR (s’il y en a)

}

Si endpoint = user/update/password/

res = {

status: 200 (réussite) ou 400 (échec de la requête) ou 404 (aucun utilisateur),

error: ERROR (s’il y en a)

}

Si endpoint = user/update/age/

res = {

status: 200 (réussite) ou 400 (échec de la requête) ou 404 (aucun utilisateur),

error: ERROR (s’il y en a)

}

Si endpoint = user/update/bio/

res = {

status: 200 (réussite) ou 400 (échec de la requête) ou 404 (aucun utilisateur),

error: ERROR (s’il y en a)

}

Si endpoint = user/update/score/

res = {

status: 200 (réussite) ou 400 (échec de la requête) ou 404 (aucun utilisateur),

error: ERROR (s’il y en a)

}

Si endpoint = user/update/coin

res = {

status: 200 (réussite) ou 400 (échec de la requête) ou 404 (aucun utilisateur),

error: ERROR (s’il y en a)

}

Si endpoint = user/update/list-emoji/

res = {

status: 200 (réussite) ou 400 (échec de la requête) ou 404 (aucun utilisateur),

error: ERROR (s’il y en a)

}

Si endpoint = user/update/profile-picture/

res = {

status: 200 (réussite) ou 400 (échec de la requête) ou 404 (aucun utilisateur),

error: ERROR (s'il y en a)

}

Si endpoint = user/score-board

res = {

status: 200 (réussite) ou 400 (échec de la requête),

error: ERROR (s'il y en a)

scoreBoard: SCORE-BOARD

}

Si endpoint = user/friend/score-board/LOGIN

res = {

status: 200 (réussite) ou 400 (échec de la requête) ou 404 (aucun utilisateur),

error: ERROR (s'il y en a)

scoreBoard: scoreBoard

}

Si endpoint = user/friend/add/

res = {

status: 200 (réussite) ou 400 (échec de la requête) ou 404 (aucun utilisateur ou ami),

error: ERROR (s'il y en a)

}

Si endpoint = user/friend/delete/

res = {

status: 200 (réussite) ou 400 (échec de la requête) ou 404 (aucun utilisateur ou ami),

error: ERROR (s'il y en a)

}

Note :

Pour avoir plus d'information sur les types des chams dans res ou dans data veuillez consulter le document **compodoc.pdf.**