Documentation AREA

Noah GIBELLI, Antoine BERNARD, Raphael PELERIN - octobre 2024

1. Enregistrement d'un utilisateur

URL: http://localhost:4000/userRoutes/user-connection/register

Méthode : POST

Corps de la requête (JSON) :

```
{
  "name": "Jean",
  "surname": "Edouard",
  "email": "noah@gmail.com",
  "password": "password123"
}
```

2. Connexion d'un utilisateur

URL: http://localhost:4000/userRoutes/user-connection/login

Méthode : POST

Corps de la requête (JSON) :

```
{
  "email": "noah@gmail.com",
  "password": "password123"
}
```

Réponse en cas de réussite :

```
{
  "success": true,
  "token": "<token>"
}
```

Réponse en cas d’échec :

{

"success": false,

"token": "<token>"

}

3. Déconnexion d’un utilisateur

URL: http://localhost:4000/userRoutes/user-connection/logout

Méthode : DELETE

Header :

{

"Authorization": "Bearer <token>"

}

Réponse en cas de succès :

{

"message": "Déconnexion réussie"

}

Réponse en cas d’érreur :

{

"message": "Erreur de déconnexion"

}

4. **Récupérer les informations de l'utilisateur connecté**

URL: http://localhost:4000/userRoutes/user-informations/user-logged

Méthode : GET

Header :

{

"Authorization": "Bearer <token>"

}

Réponse en cas de succès :

{
  "success": true,
  "name": "noah",
  "surname": "GIBELLI",
  "email": "noah2@gmail.com"
}

Réponse en cas d'erreur :

{
  "success": false,
  "name": "",
  "surname": "",
  "email": ""
}

5. Authentification Google :

URL : http://localhost:4000/userRoutes/user-connection/google-login

Méthode : GET

La réponse contiendra le token dans l'URL : http://localhost:3000/?token=<jwtToken>

6. Gestion des abonnements

Créer un abonnement :

URL: http://localhost:4000/subscriptionRoutes/subscription-management/create-subscription/:serviceName

Méthode : POST

Header : "Authorization": "Bearer <token>"

Réponse en cas de succès :

{

"success": true,

"message": "Subscription créé avec succès"

}

Supprimer un abonnement :

URL: http://localhost:4000/subscriptionRoutes/subscription-management/delete-subscription/:serviceName

Méthode : DELETE

Header : "Authorization": "Bearer <token>"

Réponse en cas de succès :

{

"success": true,

"message": "Subscription supprimé avec succès"

}

7. Création d’AREA

Création et suppression d'une AREA

- **Créer une AREA :**
- **URL :** http://localhost:4000/areaRoutes/createArea
- **Méthode :** POST
- **Header :** "Authorization": "Bearer <token>"
- **Corps de la requête (JSON) :**

```
{
  "name": "Area1",
  "ActionService": "Outlook",
  "Action": "Envoyer un mail",
  "Reactions": [
    {
      "ReactionService": "Github",
      "Reaction": "Créer un répertoire",
      "params": {
        "name_repo": "TestFinal",
        "description_repo": "Nouvel essai",
        "privacy": "true"
```

```
    }
  }
 ]
}
```

Réponse en cas de succès :

```
{
  "success": true,
  "message": "Area créée avec succès"
}
```

8. Suppression d'AREA :

URL: http://localhost:4000/areaRoutes/createArea/deleteArea/:AreaName

Méthode : DELETE

Header : "Authorization": "Bearer <token>"

Réponse en cas de succès :

```
{
  "success": true,
  "message": "Area supprimé avec succès"
}
```

9. Lister les services

- **Lister les services disponibles :**
- **URL:** http://localhost:4000/serviceRoutes/services-informations/services-list
- **Méthode :** GET
- **Réponse :** JSON contenant la liste des services.
- **Obtenir les informations d'un service :**

URL: http://localhost:4000/serviceRoutes/services-informations/service/:id

Méthode : GET

10. Gestion des abonnements utilisateurs

Lister les abonnements d'un utilisateur :

URL: http://localhost:4000/userRoutes/user-informations/user-subscriptions

Méthode : GET

Header : "Authorization": "Bearer <token>"

11. Mise à jour du profil

URL: http://localhost:4000/userRoutes/update-profile

Méthode : PUT

Header : "Authorization": "Bearer <token>"

Corps de la requête (JSON) :

```
{
  "firstName": "NouveauPrénom",
  "lastName": "NouveauNom",
  "currentPassword": "AncienMotDePasse",
  "newPassword": "NouveauMotDePasse"
```

}

Réponse en cas de succès :

{

"success": true,

"message": "Profile updated successfully"

}

12. Mise à jour d'une AREA

URL : http://localhost:4000/updateArea/:areaName

Méthode : PUT

Header : "Authorization": "Bearer <token>"

Corps de la requête (JSON):

```
{
  "newName": "NouveauNomDeLArea",
  "ActionService": "NomDuServiceAction",
  "Action": {
    "name": "NomDeLAction",
    "params": {
      "param1": "valeur1"
    }
  },
  "Reactions": [
    {
      "ReactionService": "NomDuServiceReaction",
      "Reaction": "NomDeLaRéaction",
      "params": {
        "param1": "valeur1"
      }
    }
  ]
}
```

Réponse en cas de succés :

```
{
```

```
  "success": true,
  "message": "L'Area 'NouveauNomDeLArea' a été modifiée avec succès"
}
```

13. Ajout d’un service sur le projet AREA

Étape 1 : Créer le modèle du service

1. **Créer un fichier modèle** dans le dossier `models`. Par exemple, pour un service "Twitter", créez un fichier `Twitter.js`.

```javascript
// server/models/Twitter.js
import mongoose from 'mongoose';

const TwitterSchema = new mongoose.Schema({
  userId: { type: mongoose.Schema.Types.ObjectId, ref: 'User', required: true },
  twitterId: { type: String, required: true },
  username: { type: String, required: true },
  accessToken: { type: String, required: true },
  refreshToken: { type: String, required: true },
});
```

```
const Twitter = mongoose.model('Twitter', TwitterSchema);

export default Twitter;
```

Étape 2 : Créer les routes d'authentification

1. **Créer un fichier de routes** dans le dossier `serviceRoutes/ServicesAuthentification`. Par exemple, `Twitter.js`.

```javascript
// server/serviceRoutes/ServicesAuthentification/Twitter.js

import express from 'express';

import { authenticateToken } from '../../middlewares/Authentification.js';

import axios from 'axios';

import Twitter from '../../models/Twitter.js';

import jwt from 'jsonwebtoken';


const router = express.Router();


router.get('/twitter', authenticateToken, (req, res) => {

   // Logique pour rediriger vers l'authentification Twitter

});
```

```
router.get('/twitter/callback', async (req, res) => {
  // Logique pour gérer le callback de Twitter
});

router.delete('/logout-twitter', authenticateToken, async (req, res) => {
  // Logique pour déconnecter et supprimer les données Twitter
});

export default router;
```

Étape 3 : Ajouter les actions et réactions

1. **Créer un fichier pour les actions et réactions** dans le dossier `ActionReactionSetup`. Par exemple, `TwitterActions.js`.

```javascript
// server/ActionReactionSetup/TwitterActions.js
export const postTweet = async (userId, message) => {
  // Logique pour poster un tweet
};

export const followUser = async (userId, targetUserId) => {
  // Logique pour suivre un utilisateur
```

};

```

### Étape 4 : Ajouter les routes d'actions et réactions

1. **Créer un fichier de routes** dans le dossier `ActionReactionSetup`. Par exemple, `TwitterRoutes.js`.

```javascript
// server/ActionReactionSetup/TwitterRoutes.js

import express from 'express';

import { postTweet, followUser } from './TwitterActions.js';

import { authenticateToken } from '../../middlewares/Authentification.js';

const router = express.Router();

router.post('/twitter/post-tweet', authenticateToken, async (req, res) => {
  const { userId, message } = req.body;
  try {
    await postTweet(userId, message);
    res.status(200).json({ success: true });
  } catch (error) {
    res.status(500).json({ success: false, message: error.message });
  }

```
});

router.post('/twitter/follow-user', authenticateToken, async (req, res) => {
  const { userId, targetUserId } = req.body;
  try {
    await followUser(userId, targetUserId);
    res.status(200).json({ success: true });
  } catch (error) {
    res.status(500).json({ success: false, message: error.message });
  }
});

export default router;
```

Étape 5 : Enregistrer les routes dans l'application principale

1. **Modifier le fichier principal des routes** pour inclure les nouvelles routes. Par exemple, dans `server.js` ou `app.js`.

```javascript
// server.js ou app.js
import twitterAuthRoutes from './serviceRoutes/ServicesAuthentification/Twitter.js';
import twitterActionRoutes from './ActionReactionSetup/TwitterRoutes.js';
```

app.use('/api/auth', twitterAuthRoutes);

app.use('/api/actions', twitterActionRoutes);

```

### Étape 6 : Mettre à jour les variables d'environnement

1. **Ajouter les variables nécessaires** dans votre fichier `.env`.

```

TWITTER_CLIENT_ID=your_twitter_client_id

TWITTER_SECRET=your_twitter_secret

SERVER_URL_CALLBACK=http://yourserver.com/callback

```

### Conclusion

En suivant ces étapes, vous pouvez ajouter un nouveau service à votre plateforme, ainsi que des actions et des réactions associées. Assurez-vous de tester chaque étape pour vérifier que tout fonctionne correctement.

---

Pour ajouter des actions et des réactions à votre projet, suivez ces étapes basées sur votre structure existante :
```

Étape 1 : Ajouter une nouvelle action

1. **Créer un fichier d'action** dans le dossier webhook, nommé en fonction du service et de l'action (par exemple, sendTweetAction.js pour une action Twitter).
2. **Définir la fonction d'action** dans ce fichier, avec les paramètres nécessaires. Cette fonction est celle qui déclenchera l'action du service choisi.
3. **Exporter la fonction d'action** pour pouvoir la lier à une réaction.

Étape 2 : Ajouter une réaction

1. **Aller dans le dossier action_reaction_setup** et ouvrir le fichier produceReaction.js.
2. **Créer une fonction pour la réaction** dans le fichier du service correspondant (par exemple, dans Twitter.js pour une réaction Twitter) en ajoutant la logique pour ce que la réaction doit effectuer.
3. **Ajouter la réaction dans produceReaction.js** en exportant la fonction et en l'appelant dans le switch de LinkActionToReaction. Par exemple :

```
import { ProduceTwitterReaction } from "./Twitter.js";
// Ajoutez "ProduceTwitterReaction" dans `LinkActionToReaction` pour le cas "Twitter".
```

1.

Ces étapes garantissent que le nouveau service est bien intégré avec ses actions et réactions, prêt à être utilisé dans l'application.