**Explication des Endpoints et Authentification**

**Super Admin**

Il existe un super admin qui se connecte avec les identifiants suivants:

**Username** : 780186229

**Password** : admin

**Endpoints accessibles par le rôle ADMIN**

1. **Gestion des membres**
   * **URL**: /api/membres\*\*
   * **Méthode**: GET, POST, PUT
   * **Description**: Permet de récupérer la liste de tous les membres, ajouter un nouveau membre, mettre à jour les informations d'un membre.
   * **Accessibilité**: Accessible uniquement aux utilisateurs avec le rôle **ADMIN**.
2. **Valider le paiement d'une cotisation (POST)**
   * **URL**: /api/cotisations/valider/{cotisationId}
   * **Description**: Permet de valider le paiement d'une cotisation en utilisant l'ID de la cotisation.
   * **Accessibilité**: Accessible uniquement aux utilisateurs avec le rôle **ADMIN**.
3. **Envoyer des notifications de retard (GET)**
   * **URL**: /api/cotisations/notifications/retard
   * **Description**: Permet d'envoyer des notifications de retard de paiement aux membres concernés.
   * **Accessibilité**: Accessible uniquement aux utilisateurs avec le rôle **ADMIN**.
4. **Tester la création d'un événement (POST)**

* **URL** : http://localhost:9090/api/evenements/creer
* **Accessibilité**: Accessible uniquement aux utilisateurs avec le rôle **ADMIN**.

1. **Tester la suppression d'un événement (DELETE)**

* **URL** : http://localhost:9090/api/evenements/supprimer/1
* **Accessibilité**: Accessible uniquement aux utilisateurs avec le rôle **ADMIN**.

1. **Tester la mise à jour d'un événement (PUT)**

* **URL** : http://localhost:9090/api/evenements/modifier/1
* **Accessibilité**: Accessible uniquement aux utilisateurs avec le rôle **ADMIN**.

**Endpoints de la Gestion des Membres**

1. **Récupérer tous les membres**
   * **Description**: Cet endpoint permet de récupérer la liste de tous les membres enregistrés dans la base de données.
   * **Méthode**: GET
   * **URL**: /api/membres
   * **Authentification**: Requiert authentification
   * **Body**: None
2. **Créer un nouveau membre**
   * **Description**: Cet endpoint permet d'ajouter un nouveau membre à la base de données.
   * **Méthode**: POST
   * **URL**: /api/membres
   * **Authentification**: Requiert authentification
   * **Body**:

{

"prenom": "Laity",

"nom": "FAYE",

"addresse": "Thiès",

"dateIntegration": "2022-01-01",

"telephone": "780186228",

"password": "password",

"role": "USER"

}

1. **Récupérer un membre par ID**
   * **Description**: Cet endpoint permet de récupérer les détails d'un membre spécifique en utilisant son ID.
   * **Méthode**: GET
   * **URL**: /api/membres/{id}
   * **Authentification**: Requiert authentification
   * **Body**: None
   * **Exemple de réponse**:
2. **Mettre à jour un membre**
   * **Description**: Cet endpoint permet de mettre à jour les informations d'un membre spécifique en utilisant son ID.
   * **Méthode**: PUT
   * **URL**: /api/membres/{id}
   * **Authentification**: Requiert authentification
   * **Body**:

{

"prenom": "Laity",

"nom": "FAYE",

"addresse": "Thiès",

"dateIntegration": "2022-01-01",

"telephone": "780186220",

"password": "password",

"role": "USER"

}

**Endpoints de la Gestion des Cotisations Mensuelle**

1. **Enregistrer une cotisation**
   * **Description**: Cet endpoint permet d'enregistrer une nouvelle cotisation pour un membre.
   * **Méthode**: POST
   * **URL**: /api/cotisations
   * **Authentification**: Requiert authentification
   * **Body**:

{

"membreId": 1,

"montant": 100.0,

"datePaiement": "2023-04-05",

"moisCotisation": "2023-04"

}

1. **Récupérer les cotisations par mois**
   * **Description**: Cet endpoint permet de récupérer toutes les cotisations effectuées pour un mois donné.
   * **Méthode**: GET
   * **URL**: /api/cotisations/mois/{moisCotisation}
   * **Authentification**: Requiert authentification
   * **Body**: None
2. **Récupérer les cotisations par membre**
   * **Description**: Cet endpoint permet de récupérer toutes les cotisations effectuées par un membre spécifique.
   * **Méthode**: GET
   * **URL**: /api/cotisations/membre/{membreId}
   * **Authentification**: Requiert authentification
   * **Body**: None
3. **Récupérer une cotisation par membre et mois**
   * **Description**: Cet endpoint permet de récupérer la cotisation d'un membre pour un mois spécifique.
   * **Méthode**: GET
   * **URL**: /api/cotisations/{membreId}/{moisCotisation}
   * **Authentification**: Requiert authentification
   * **Body**: None
4. **Valider le paiement d'une cotisation**
   * **Description**: Cet endpoint permet de valider le paiement d'une cotisation en utilisant l'ID de la cotisation.
   * **Méthode**: POST
   * **URL**: /api/cotisations/valider/{cotisationId}
   * **Authentification**: Requiert authentification
   * **Body**: None
5. **Envoyer des notifications de retard**
   * **Description**: Cet endpoint permet d'envoyer des notifications de retard de paiement aux membres concernés.
   * **Méthode**: GET
   * **URL**: /api/cotisations/notifications/retard
   * **Authentification**: Requiert authentification
   * **Body**: None

**Endpoints Gestion des Evénements**

**1. Tester la création d'un événement (POST)**

* **URL** : http://localhost:9090/api/evenements/creer
* **Méthode** : POST
* **Corps de la requête** : Assurez-vous d'utiliser un **JSON valide** avec les champs nécessaires, par exemple :

{

"nom": "Réunion générale",

"description": "Réunion annuelle de l'entreprise",

"date": "2025-02-15",

"heure": "10:00",

"lieu": "Salle de conférence A",

"type": "Collectif",

"statut": "Planned",

"organisateur": "Laity Faye",

}

Si l'événement est déjà existant (basé sur les critères de doublon), vous devriez recevoir une réponse d'erreur :

{

"message": "Un événement avec ce nom, date et heure existe déjà."

}

**2. Tester la récupération de tous les événements (GET)**

* **URL** : http://localhost:9090/api/evenements
* **Méthode** : GET
* **Réponse attendue** : Liste des événements, par exemple :

[

{

"id": 1,

"nom": "Réunion générale",

"description": "Réunion annuelle de l'entreprise",

"date": "2025-02-15",

"heure": "10:00",

"lieu": "Salle de conférence A",

"type": "Collectif",

"statut": "Planned",

"organisateur": "Laity Faye",

"membreId": null

}

]

**3. Tester la récupération d'un événement spécifique (GET avec ID)**

* **URL** : http://localhost:9090/api/evenements/1
* **Méthode** : GET
* **Réponse attendue** : Détails de l'événement avec l'ID spécifique.

{

"id": 1,

"nom": "Réunion générale",

"description": "Réunion annuelle de l'entreprise",

"date": "2025-02-15",

"heure": "10:00",

"lieu": "Salle de conférence A",

"type": "Collectif",

"statut": "Planned",

"organisateur": "Laity Faye",

"membreId": null

}

**4. Tester la mise à jour d'un événement (PUT)**

* **URL** : http://localhost:9090/api/evenements/modifier/1
* **Méthode** : PUT
* **Corps de la requête** :

{

"nom": "Réunion générale mise à jour",

"description": "Réunion annuelle avec des mises à jour importantes",

"date": "2025-02-16",

"heure": "11:00",

"lieu": "Salle de conférence B",

"type": "Collectif",

"statut": "Planned",

"organisateur": "Laity Faye",

}

* **Réponse attendue** : L'événement mis à jour avec les nouvelles informations.

{

"id": 1,

"nom": "Réunion générale mise à jour",

"description": "Réunion annuelle avec des mises à jour importantes",

"date": "2025-02-16",

"heure": "11:00",

"lieu": "Salle de conférence B",

"type": "Collectif",

"statut": "Planned",

"organisateur": "Laity Faye",

"membreId": null

}

**5. Tester la suppression d'un événement (DELETE)**

* **URL** : http://localhost:9090/api/evenements/supprimer/1
* **Méthode** : DELETE
* **Réponse attendue** : Aucun contenu, ou un message de confirmation que l'événement a été supprimé.

{

"message": "Événement supprimé avec succès."

}