LP1 Project - API

Oui, vous avez bien lu. Une simple API, en Java. (Version béta)

Créé par : Maxime Princelle

Lien vers le sujet

Sommaire

Comment y accéder ?

Projet encore en cours de développement...

L'API est accessible via ce lien :

https://lp1-api-jwt.princelle.org/api

Pour plus d'informations sur les routes, je vous invite à vous rendre ici.

Configuration

Une fois la base de données renseignée et l'application lancée, cette dernière, via Hibernate, va générer tout ce qui est nécessaire à son fonctionnement.

Pour la configuration, deux options s'offrent à vous.

1. Vous souhaitez tout faire sur Docker (recommandé) :

Dans ce cas, veuillez suivre la partie sur Docker.

2. Vous possédez déjà une base de données :

Dans ce cas, veuillez commencer par copier le fichier `.env.example` et le renommer en `.env`.

Une fois ouvert, vous obtiendrez le fichier suivant :

```
# Application
DATABASE_URL=mysql://${DATABASE_HOST}:${DATABASE_PORT}/${DATABASE_NAME}
DATABASE_NAME=lp1Project
DATABASE_USER=#__db_user__#
DATABASE_PASS=#__db_pass__#
DATABASE_HOST=db
```

```
DATABASE_PORT=3306

# MySQL
MYSQL_DATABASE=${DATABASE_NAME}
MYSQL_RANDOM_ROOT_PASSWORD=yes
MYSQL_USER=${DATABASE_USER}
MYSQL_PASSWORD=${DATABASE_PASS}

# S3
S3_ENDPOINT=https://s3.princelle.org
ACCESS_KEY=AKIAIOSXXXXX7XXAXPAX
SECRET_KEY=wJalrXUtnFEMI/K7MDEXXXXXxRfiCYXXAXLPXKEX

# JWT
JWT_SECRET=QhEEDgVhHpnehgVcypQNCyJYTTeTkPoncwHPBcJJFPWnJ3Pt76M49vZQioit
```

Pour changer les identifiants de connexion à la base de données, vous pouvez changer les champs suivants :

- DATABASE_USER : `#__db_user__#`
- DATABASE_PASS : `#__db_pass__#`
- DATABASE_NAME : `lp1Project` (le nom de votre base, **créée au préalable (si locale)** !)
- DATABASE_HOST : `db` (si votre base est locale, mettez `localhost`)
- DATABASE_PORT : `3306`

Pour finir la configuration, enregistrez votre fichier `.env` à la racine du projet.

Lancement

Passons maintenant au lancement de l'application.

Deux options s'offrent à vous :

- Depuis la racine du projet, lancer cette commande :

```
mvn spring-boot:run
```

- Depuis votre environnement de développement, lancer la méthode suivante :

```
Lp1ProjectApplication.main()
```

En tant que méthode lancée en "Standalone", qui va lancer le serveur intégré Tomcat sur le port 8080.

--

Après cela, il suffira de vous rendre à l'aide de votre client Web sur l'url suivante :

http://localhost:8080/api/

`/api` correspond à la route sur laquelle l'API est publiée.

Pour voir toutes les routes de l'application, cliquez-ici.

Docker

Configuration de l'environnement

Si vous souhaitez démarrer cette application, ainsi que la base de données nécessaire à son fonctionnement, en une seule fois, un fichier docker-compose est mis à disposition.

Avec cette base de données vous est également fourni un environnement phpMyAdmin afin de faciliter la gestion de cette dernière.

Pour commencer, veuillez copier le fichier `.env.example` et le renommer en `.env`.

Une fois ouvert, vous obtiendrez le fichier suivant :

```
# Application
DATABASE_URL=mysql://${DATABASE_HOST}:${DATABASE_PORT}/${DATABASE_NAME}
DATABASE_NAME=lp1Project
DATABASE_USER=#__db_user__#
DATABASE_PASS=#__db_pass__#
DATABASE_HOST=db
DATABASE_PORT=3306

# MySQL
MYSQL_DATABASE=${DATABASE_NAME}
MYSQL_RANDOM_ROOT_PASSWORD=yes
MYSQL_USER=${DATABASE_USER}
MYSQL_PASSWORD=${DATABASE_PASS}

# S3
S3_ENDPOINT=https://s3.princelle.org
ACCESS_KEY=AKIAIOSXXXXX7XXAXPAX
SECRET_KEY=wJalrXUtnFEMI/K7MDEXXXXXxRfiCYXXAXLPXKEX

# JWT
JWT_SECRET=QhEEDgVhHpnehgVcypQNCyJYTTeTkPonxxxxxcJJFPWnJ3Pt76M49vZQioit
```

Pour changer les identifiants de connexion à la base de données, vous pouvez changer les deux champs "DATABASE_USER" et "DATABASE_PASS" :

- `#__db_user__#`
- `#__db_pass__#`

Pour finir, enregistrez votre fichier `.env` à la racine du projet.

Lancement de l'environnement

Afin de lancer l'environnement, exécutez la commande suivante :

```
docker-compose up -d
```

API

Une fois fait, l'API sera déployée à l'adresse :

http://localhost:8085/api/

`/api` correspond à la route sur laquelle l'API est publiée.

Pour voir toutes les routes de l'application, cliquez-ici.

phpMyAdmin

Pour accéder à phpMyAdmin, ce dernier se trouve sur le port 8084, vous pouvez y accédez via le lien suivant :

http://localhost:8086/

Routes

Comme dit précédemment, `/api` correspond à la route sur laquelle l'API est publiée.

Voici l'arborescence de l'application :

NB : Les seules routes qui sont ouvertes sont : `/login`, `/signup` et `/`. Les autres routes nécessitent le passage d'un token (valable 10 jours après génération) via "Bearer" inséré dans "Header" avant chaque requête.

- Auth
    - POST `/login` : Connexion
      Réponse (JSON) : Renvoie le token pour une Authentification via Headers.
      Requête (JSON) : Attends les paramètres suivants :
      Ici, on a le choix, soit on utilise `emailId`, soit `pseudo`, dans tous les cas, l'un des deux ainsi que le mot de passe sont obligatoires.

      ```
      {
              "emailId": "contact@princelle.org",
              "pseudo": "ThePrince",
              "password": "test1",
      }
      ```

    - POST `/signup` : Ajoute un utilisateur.
      Réponse (JSON) : Renvoie l'utilisateur ajouté.
      Requête (JSON) : Attends les paramètres suivants (non optionnels) :

      ```
      {
              "firstName": "Maxime", //Obligatoire
              "lastName": "Princelle", //Obligatoire
              "emailId": "contact@princelle.org", //Obligatoire, et
      ```

```
unique
        "password": "test1", //Obligatoire
        "pseudo": "ThePrince" //Obligatoire, et unique.
}
```

- GET `/` : Hello World !
- Users
  - GET `/users` : Liste tous les utilisateurs et leurs informations.
  - GET `/users/{id}` : Affiche les informations d'un utilisateur en fonction de son ID.
  - GET `/users/pseudo/{pseudo}` : Affiche les informations d'un utilisateur en fonction de son Pseudo.
  - GET `/users/nocoloc` : Affiche les utilisateurs qui n'ont pas de Colocation.
  - PUT `/users/{id}` : Modifie les informations d'un utilisateur en fonction de son ID.
    Réponse (JSON) : Renvoie l'utilisateur modifié.
    Requête (JSON) : Attends un ou plusieurs des paramètres suivants :

    ```
    {
            "firstName": "Maxime", //Optionnel
            "lastName": "Princelle", //Optionnel
            "emailId": "contact-maxime@princelle.org", //Optionnel
            "password": "test2", //Optionnel
            "pseudo": "TheKing" //Optionnel
    }
    ```

  - DELETE `/users/{id}` : Supprime un utilisateur en fonction de son ID.
    Réponse (JSON) : Renvoie un boolean confirmant sa suppression.

    ```
    {
            "deleted": true
    }
    ```

- Colocations
  - GET `/colocs` : Liste toutes les colocations et leurs informations.
  - GET `/colocs/{id}` : Affiche les informations d'une colocation en fonction de son ID.
  - GET `/colocs/{id}/members` : Affiche les membres d'une colocation en fonction de son ID.
  - POST `/colocs` : Ajoute une colocation.
    Réponse (JSON) : Renvoie la colocation ajoutée.
    Requête (JSON) : Attends les paramètres suivants (non optionnels) :

```
{
        "name": "L'équipeDe@W.", //Obligatoire
}
```

- POST `/colocs/{id}/members/{userID}` : Ajoute l'utilisateur avec son ID (`userID`) (si il existe) à la Colocation déterminée elle aussi par son ID.
  Réponse (JSON) : Renvoie la liste des membres de la Colocation.

- PUT `/colocs/{id}` : Modifie les informations d'une colocation en fonction de son ID.
  Réponse (JSON) : Renvoie la colocation modifiée.
  Requête (JSON) : Attends un ou plusieurs des paramètres suivants :

```
{
        "name": "L'équipeDe@H.", //Optionnel
}
```

- DELETE `/colocs/{id}/members/{userID}` : Retire l'utilisateur de la Colocation déterminé par son ID avec l'ID de l'Utilisateur (`userID`) (si il existe et s'il est bien membre).
  Réponse (JSON) : Renvoie la liste des membres de la Colocation.

- DELETE `/colocs/{id}` : Supprime une colocation en fonction de son ID.
  Réponse (JSON) : Renvoie un boolean confirmant sa suppression.

```
{
        "deleted": true
}
```

- Tasks

  - GET `/tasks` : Liste toutes les taches et leurs informations.

  - GET `/tasks/{id}` : Affiche les informations d'une tache en fonction de son ID.

  - GET `/tasks/coloc/{colocId}` :
    Affiche les taches liées à une Colocation (déterminée par son ID).

  - GET `/tasks/not_attributed` : Affiche les taches non attribuées.

  - GET `/tasks/coloc/{colocId}/not_attributed` :
    Affiche les taches non attribuées liées à une Colocation (déterminée par `colocId`).

  - GET `/tasks/pending` : Affiche les taches en attente.

  - GET `/tasks/coloc/{colocId}/pending` :
    Affiche les taches en attente liées à une Colocation (déterminée par `colocId`).

- GET `/tasks/users/to/{userId}/pending` :
  Affiche les taches en attente, liées à un 'User' (déterminé par `userId`).

- GET `/tasks/users/from/{userId}/pending` :
  Affiche les taches en attente, déposées par un 'User' (déterminé par `userId`).

- GET `/tasks/proposed` : Affiche les taches proposées.

- GET `/tasks/coloc/{colocId}/proposed` :
  Affiche les taches proposées liées à une Colocation (déterminée par `colocId`).

- GET `/tasks/users/to/{userId}/proposed` :
  Affiche les taches proposées à un 'User' (déterminé par `userId`).

- GET `/tasks/users/from/{userId}/proposed` :
  Affiche les taches proposées par un 'User' (déterminé par `userId`).

- GET `/tasks/users/to/{userId}` :
  Affiche les taches liées à un 'User' (déterminé par `userId`).

- GET `/tasks/users/from/{userId}` :
  Affiche les taches créées par un 'User' (déterminé par `userId`).

- POST `/tasks` : Ajoute une tache.
  Réponse (JSON) : Renvoie la tache ajoutée.
  Requête (JSON) : Attends les paramètres suivants :

```
{
        "title": "Faire des pâtes", //Obligatoire
        "description": "Ne pas oublier l'eau...", //Obligatoire
        "cost": 5, //Type: Entier //Obligatoire
        "proposed": false, //Type: Booléen //Obligatoire
        "coloc": {
                "id": 1
        }, //Obligatoire
        "fromPerson": {
                "id": 1
        },
        "toPerson": {
                "id": 2
        }
}
```

- PUT `/tasks/{id}` : Modifie une tache, en fonction de son ID et de son état.
  Réponse (JSON) : Renvoie la tache modifiée.
  Requête (JSON) : Attends les paramètres suivants (optionnels) :

```
{
        "title": "Faire du riz",
```

```
        "description": "Parce que c'est la vie...",
        "cost": 5, //Type: Entier
        "proposed": false, //Type: Booléen
        "coloc": {
                "id": 1
        },
        "fromPerson": {
                "id": 1
        },
        "toPerson": {
                "id": 2
        },
        "picture": "https://tenor.com/SXcQ.gif" //Type: URL
directe vers Image
}
```

NB : Si la tache a déjà été effectué (présence d'une "finishDate"), seule l'image est modifiable.

- PUT `/tasks/{id}/valid/to` :
  Marque la tache comme validée du côté de la personne à qui elle l'a été attribuée.
  Attribue également à la tache la date et l'heure du jour.
  Réponse (JSON) : Renvoie la tache modifiée.

- PUT `/tasks/{id}/valid/from` :
  Marque la tache comme validée du côté de la personne qui l'a attribuée.
  Attribue également à l'utilisateur à qui la tache été dédié (s'il y en a un, et qu'il l'a validée avant de son côté), les points eux sont ajoutés automatiquement.
  Réponse (JSON) : Renvoie la tache modifiée.

- PUT `/tasks/{id}/valid/to/rev` :
  Annule le marquage de la tache comme validée du côté de la personne à qui elle l'a été attribuée.
  Si la tâche a été validée par la personne qui l'a créée, il ne sera plus possible de faire cette requête
  Réponse (JSON) : Renvoie la tache modifiée.

- PUT `/tasks/{id}/valid/from/rev` :
  Annule le marquage de la tache comme validée du côté de la personne qui l'a attribuée.
  Les points sont retirés automatiquement s'ils ont été attribués à un utilisateur. La date et le statut de la tache du côté de la personne a qui elle est attribuée, sont supprimés.
  Réponse (JSON) : Renvoie la tache modifiée.

- DELETE `/tasks/{id}` : Supprime la tache en fonction de son ID.
  Réponse (JSON) : Renvoie un boolean confirmant sa suppression.

```
{
        "deleted": true
}
```

NB : Les points sont retirés automatiquement du score de l'utilisateur s'ils ont été attribués auparavant.

Technologies utilisées

Ce projet Maven est basé sur :

- Apache Tomcat
- Spring
- JPA / Hibernate avec MySQL Connector
- JWT

Remonter en haut