

# File Manager 2.0

Pour lancer l'application, il faut mettre en place la base de données et utiliser Docker qui va vous permettre d'utiliser SFTP.

Dans cette application j'ai utilisé l'authentification avec Okta et l'authentification normale avec login et mot de passe et au final j'ai gardé que la normale.

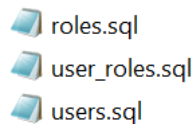
## 1. SQL

Créer une base de données :

```
CREATE DATABASE test_auth;
```

```
spring.datasource.url=jdbc:mysql://localhost:3306/test_auth?useUnicode=true&useJDBCCompliantTimezoneShift=true&useLegacyDatetimeCode=false&serverTimezone=UTC
spring.datasource.username=root
spring.datasource.password=
```

J'ai utilisé PhpMyAdmin pour lancer les requêtes SQL. Vous trouverez ci-joint les fichiers SQL que vous pouvez lancer pour créer les tables de la base de données.

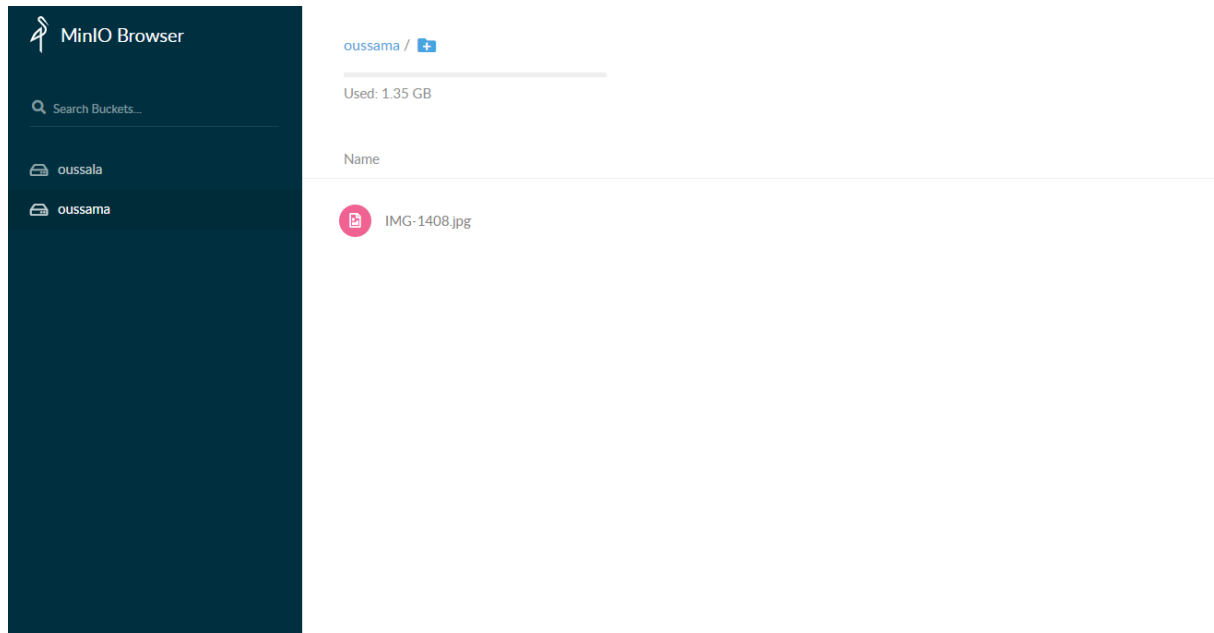


## 2. Minio

Vous trouverez au-dessous l'access key et le secret key pour tester le téléchargement des fichiers.

<https://play.min.io/>

```
# Minio Host
spring.minio.url=https://play.min.io
# Minio access key (login)
spring.minio.access-key=Q3AM3UQ867SPQQA43P2F
# Minio secret key (password)
spring.minio.secret-key=zuf+tfteSlsRu7BJ86wekitnifILbZam1KYY3TG
```



### 3. Docker

Pour lancer docker :

```
docker run -p 22:22 -d atm0z/sftp oussama:oussama:::upload
```

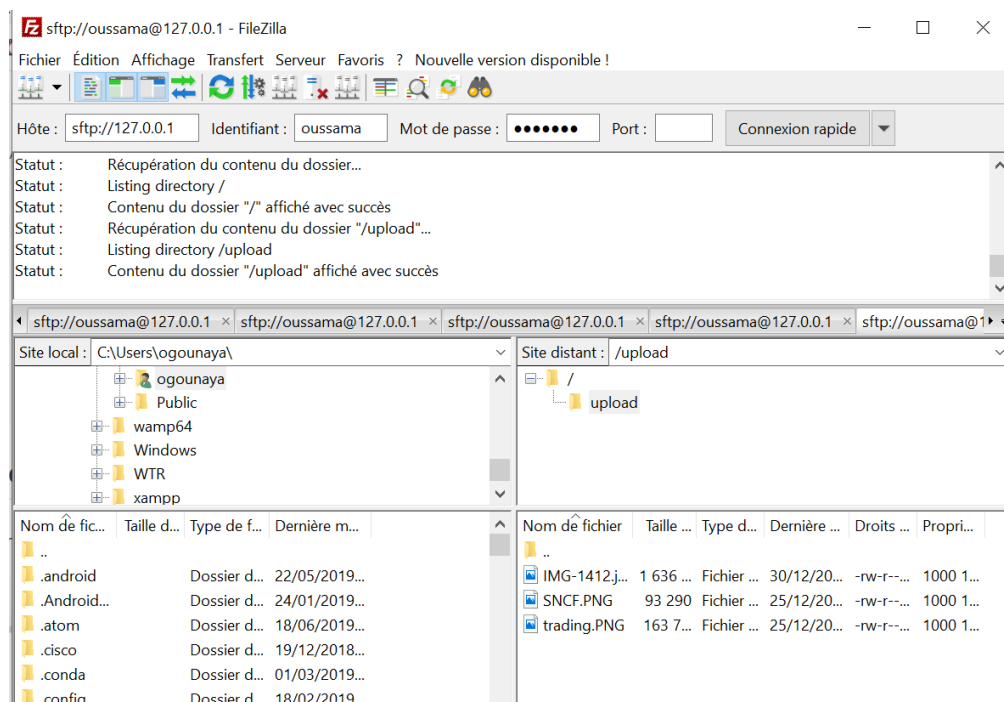
Et grâce à Docker on peut se connecter à un serveur local :

Hôte : 127.0.0.1

Identifiant : oussama

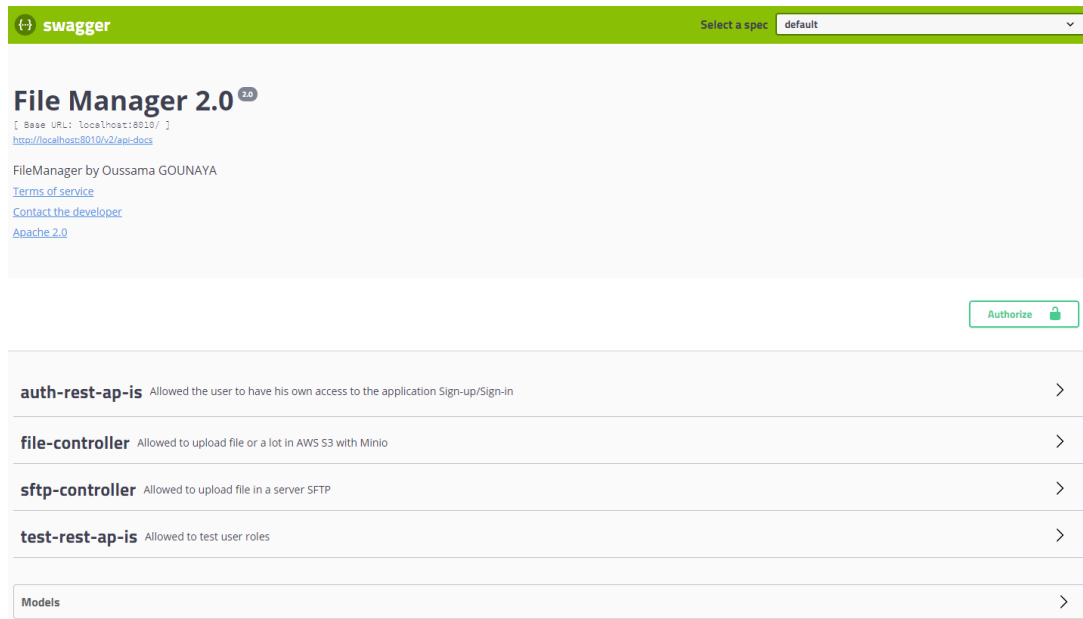
Mot de passe : oussama

Port : 22



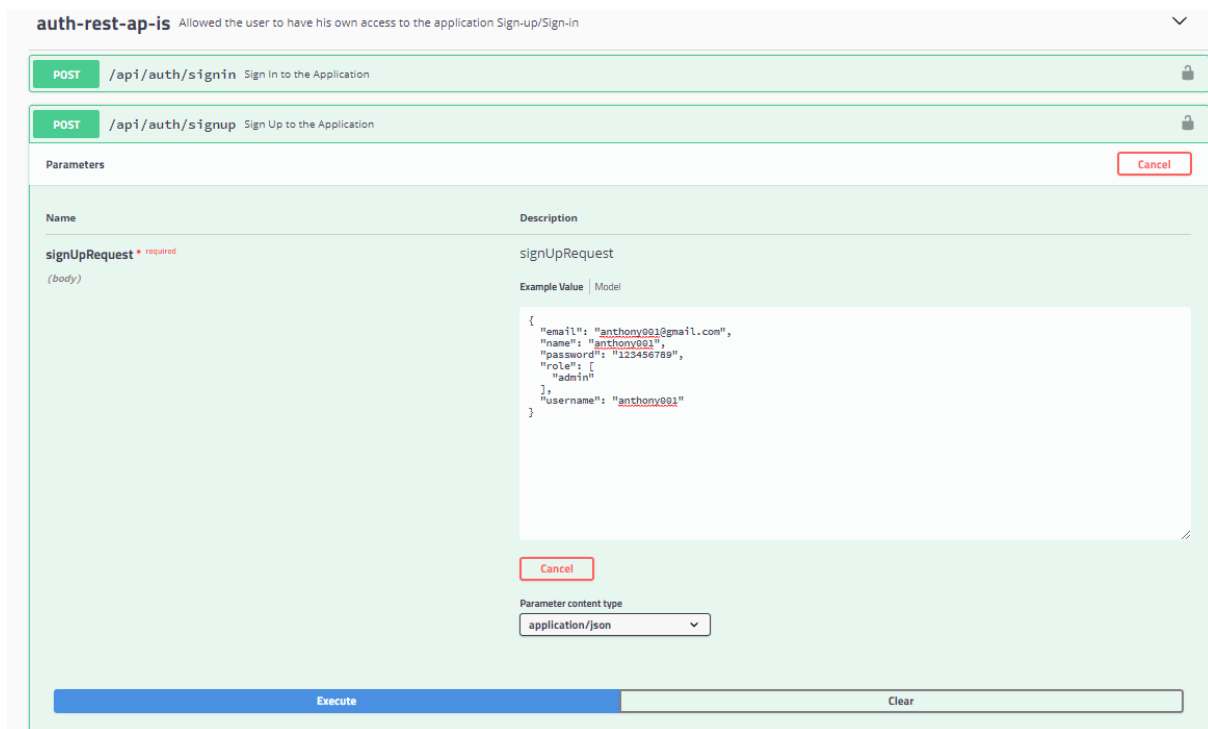
**Swagger :**

Et quand vous lancer l'application, vous pouvez tester les APIs avec Swagger :

**Création d'un compte « user » ou « admin » :**

(Admin a les mêmes droits que l'user)

Notre utilisateur Anthony001 a le rôle d'admin.



**Se connecter avec votre compte pour avoir le token qui va vous permettre d'utiliser les autres services :**

POST

/api/auth/signin Sign In to the Application

Parameters

loginRequest required

(body)

loginRequest

Example Value | Model

```
{  "password": "123456789",  "username": "anthony901"}
```

Cancel

Parameter content type  
application/json

Execute

Clear

Responses

Response content type \*/\*

Curl

```
curl -X POST "http://localhost:8010/api/auth/signin" -H "accept: */*" -H "Content-Type: application/json" -d '{"password": "123456789", "username": "anthony901"}'
```

Request URL

http://localhost:8010/api/auth/signin

Server response

Code

Details

200

Response body

```
{  "accessToken": "eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJhbnRob255MDAxIiwiaWF0IjoxNTc5MzY0NDIzLCJleHAiOiJlMjc4MjQ0RjN9.m3RGenGuXkiVvR8L0xRezTT7XYVK9Z7Szi-E049009h1voBy",  "tokenType": "Bearer"}
```

Download

### **Mettre le token dans autorisation :**

The screenshot shows the Swagger UI interface. In the foreground, a modal dialog titled "Available authorizations" is open. It lists the "JWT (apiKey)" authorization type. The details shown are: Name: Authorization, In: header, and a Value field containing "Bearer eyJhbGciOiJIUzUxMi...". At the bottom of the dialog are "Authorize" and "Close" buttons. In the background, the Swagger UI for the "auth-rest-api" is visible, showing a POST endpoint for "/api/auth/signin" with a "loginRequest" parameter.

**Pour file-controller :**

Il faut juste mettre le bucketName et ajouter le fichier.

**file-controller** Allowed to upload file or a lot in AWS S3 with Minio

**DELETE** /api/minio/delete deleteFile

**GET** /api/minio/getAll Get A List All Files - Minio

**POST** /api/minio/uploadFile Upload A File - Minio

**POST** /api/minio/uploadFile Upload A File - Minio

Parameters

Name	Description
<b>bucketName</b> * required string (query)	bucketName <input type="text" value="anthonybond007"/>
<b>file</b> * required file (FormData)	file <input type="button" value="Choose File"/> IMG-1410.jpg

Execute

Clear

Responses

Response content type \*/\*

Curl

```
curl -X POST "http://localhost:8010/api/minio/uploadFile?bucketName=anthonybond007" -H "accept: */*" -H "Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXZW4iLCJ0eXAiOiJKV1QiLCJkaXI6ImF1dG8iLCJleHRAcjE1Mzc4NjQ8NjN9.n3RGenGuKcYvR8LoxRezIT7XYV9Z7Sz1-E049009h1voBy-yEnOyGRMM01aaNuz6q8LgiYbFjFKStf6JL0QA" -H "Content-Type: multipart/form-data" -F "file=@IMG-1410.jpg;type=image/jpeg"
```

Request URL

```
http://localhost:8010/api/minio/uploadFile?bucketName=anthonybond007
```

Server response

Code	Details
200	<div>Response headers</div> <pre>cache-control: no-cache, no-store, max-age=0, must-revalidate content-length: 0 date: Mon, 30 Dec 2019 21:02:02 GMT expires: 0 pragma: no-cache vary: Origin, Access-Control-Request-Method, Access-Control-Request-Headers x-content-type-options: nosniff x-frame-options: DENY x-xss-protection: 1; mode=block</pre>

Et après vous aurez :

MinIO Browser

anth

anthonybond007

anthonybond007 /

Used: 1.36 GB

Name

IMG-1410.jpg

**Pour Sftp-Controller :**

**sftp-controller** Allowed to upload file in a server SFTP

DELETE

/api/sftp/delete deleteFile

POST

/api/sftp/upload Upload A File - SFTP

POST

/api/sftp/upload Upload A File - SFTP

Parameters

Cancel

Name	Description
<b>address</b> * required string (query)	address <input type="text" value="127.0.0.1"/>
<b>file</b> * required file (formData)	file <div>Choose File IMG-1412.jpg</div>
<b>password</b> * required string (query)	password <input type="text" value="oussama"/>
<b>path</b> * required string (query)	path <input type="text" value="/upload"/>
<b>port</b> * required integer(\$int32) (query)	port <input type="text" value="22"/>
<b>username</b> * required string (query)	username <input type="text" value="oussama"/>

Execute

Clear

**Pour tester vos rôles :**

**test-rest-ap-is** Allowed to test user roles

GET

/api/test/admin Test 'admin' Role

GET

/api/test/user Test 'user' Role

FIN 🤖