

TD4

Introduction

Au cours de ce TP nous allons reprendre la galerie d'images réalisée précédemment et y ajouter un datastore. On y stockera également l'organisation des photos et une gestion des utilisateurs.

Partie I - Redis Datastore

Nous allons nous orienter vers une base de données clé-valeur faite pour être distribuée sur plusieurs machines : Redis.

Pour commencer nous avons besoin de deux conteneurs.

Un premier demon contenant la base de donnée, lancé avec la commande :

```
orain@orain-TM1701:~/Documents/td4-antho35/td4$ sudo docker run --name redis-datastore -d redis
[sudo] Mot de passe de orain :
Unable to find image 'redis:latest' locally
latest: Pulling from library/redis
177e7ef0df69: Pull complete
66ec699db42d: Pull complete
9af6d87fd347: Pull complete
de9172cdb09c: Pull complete
27733a222e28: Pull complete
ef1ae1903ba4: Pull complete
Digest: sha256:86654d77602cbfeb873191488d176b215067549b7491364c3f84046f5753f0a0
Status: Downloaded newer image for redis:latest
83aeb905f785caaa0cab1ea6050804b6e179168acdaf270867bbae540e639b3f
```

Et la CLI :

```
orain@orain-TM1701:~/Documents/td4-antho35/td4$ sudo docker run -it --link redis-datastore:redis-cli
--rm redis redis-cli -h redis-datastore
redis-datastore:6379>
```

1)

On se familiarise avec la CLI, à l'aide de commande simples :

```
orain@orain-TM1701:~/Documents/td4-antho35/td4$ sudo docker run -it --link redis-datastore:redis-cli
--rm redis redis-cli -h redis-datastore
redis-datastore:6379> set mykey somevalue
OK
redis-datastore:6379> get mykey
"somevalue"
redis-datastore:6379> set mykey anothervalue
OK
redis-datastore:6379> get mykey
"anothervalue"
redis-datastore:6379> set mykey anothervalue nx
(nil)
redis-datastore:6379> set mykey anothervalue xx
OK
```

2)

Le seul moyen pour que les datas survivent à la suppression d'un conteneur est de créer un volume à côté.

Partie II – Architecture

Création d'une infrastructure Docker Node + Redis

1)

On duplique le projet du TP3 dans un premier temps puis on le connecte au datastore Redis à l'aide de docker-compose.

On commence par créer nouveau répertoire « myvolredis » puis on crée un volume :

```
orain@orain-TM1701:~/Documents/td4-antho35/td4/myhbsapp$ sudo docker run --name some-redis -v myvolr
edis:/data -d redis redis-server --appendonly yes
ee4001ab3e40f8e757d73e448665a36b1b94e404dd6d4dc44712f180d5353e05
orain@orain-TM1701:~/Documents/td4-antho35/td4/myhbsapp$ sudo docker run -it --link some-redis:redis
-cli --rm redis redis-cli -h some-redis
some-redis:6379> set mykey ok
OK
some-redis:6379> get mykey
"ok"
some-redis:6379>
orain@orain-TM1701:~/Documents/td4-antho35/td4/myhbsapp$ sudo docker rm -f some-redis
some-redis
orain@orain-TM1701:~/Documents/td4-antho35/td4/myhbsapp$ sudo docker run --name some-redis -v myvolr
edis:/data -d redis redis-server --appendonly yes
e5a7b77f5069c7bc2753a0bba087b1a30ae7f79315075d1bc0c3d883e064250c
orain@orain-TM1701:~/Documents/td4-antho35/td4/myhbsapp$ sudo docker run -it --link some-redis:redis
-cli --rm redis redis-cli -h some-redis
some-redis:6379> get mykey
"ok"
some-redis:6379> |
```

Les datas ont bien été sauvegardées après la suppression du conteneur grâce à l'ajout de « --appendonly yes » dans la commande de création du volume.

Je complète mon fichier Dockerfile pour ajouter « VOLUME /myvolredis/data » :

```
FROM node:8.15.0-alpine

# Create app directory
WORKDIR /usr/src/app

# Install app dependencies
# A wildcard is used to ensure both package.json AND package-lock.json are copied
# where available (npm@5+)
COPY package*.json ./

RUN npm install
# If you are building your code for production
# RUN npm install --only=production

# Bundle app source
COPY . .

EXPOSE 3000
VOLUME /myvolredis/data
CMD [ "npm", "start" ]
```

Je crée un fichier docker-compose.yml

```
docker-compose.yml
1 version: '3'
2 services:
3   web:
4     build: .
5     ports:
6       - "3000:3000"
7   redis:
8     image: "redis:alpine"
9     volumes:
10      - "./myvolredis:/data"
```

Dans ce fichier je n'ai pas indiqué le volume pour les images car je ne l'avais pas fait lors du TP précédent.

Puis on lance la commande suivante :

```
orain@orain-TM1701:~/Documents/td4-antho35/td4/myhbsapp$ sudo docker-compose up --build
Building web
Step 1/8 : FROM node:8.15.0-alpine
----> 5c0c5c94503f
Step 2/8 : WORKDIR /usr/src/app
----> Using cache
----> ba4f2ff94537
Step 3/8 : COPY package*.json ./
----> Using cache
----> d154d9e058e8
Step 4/8 : RUN npm install
----> Using cache
----> a458281be769
Step 5/8 : COPY . .
----> f9f6ea14601a
Step 6/8 : EXPOSE 3000
----> Running in 28c33e34fcad
Removing intermediate container 28c33e34fcad
----> e6cdc9c9b7b9
Step 7/8 : VOLUME /myvolredis/data
----> Running in c58a0ce1c9ff
Removing intermediate container c58a0ce1c9ff
----> a067ccde9097
```

```
redis_1 | 1:M 17 Jan 2019 10:38:51.190 # WARNING you have Transparent Huge Pages (THP) support enabled in your kernel. This will create latency and memory usage issues with Redis. To fix this issue run the command 'echo never > /sys/kernel/mm/transparent_hugepage/enabled' as root, and add it to your /etc/rc.local in order to retain the setting after a reboot. Redis must be restarted after THP is disabled.
redis_1 | 1:M 17 Jan 2019 10:38:51.190 * DB loaded from disk: 0.000 seconds
redis_1 | 1:M 17 Jan 2019 10:38:51.190 * Ready to accept connections
web_1 | > myhbsapp@0.0.0 start /usr/src/app
web_1 | > node ./bin/www
web_1 | node_redis: WARNING: You passed "http" as protocol instead of the "redis" protocol!
web_1 | GET / 304 56.818 ms - -
web_1 | GET /stylesheets/style.css 200 3.591 ms - 221
```

On vérifie dans le navigateur que l'app est toujours fonctionnelle, c'est bien le cas.

On s'assure que nous sommes bien connectés avec la base de données Redis :

```
var client = redis.createClient("http://redis");

client.on('connect', function(){
  console.log("connected to redis..");
});
```

```
orain@orain-TM1701:~/Documents/td4-antho35/td4/myhbsapp$ sudo docker-compose up --build
[sudo] Mot de passe de orain :
Building web
```

```
web_1      | > myhbsapp@0.0.0 start /usr/src/app
web_1      | > node ./bin/www
web_1      |
web_1      | node_redis: WARNING: You passed "http" as protocol instead of the "redis" protocol!
web_1      | connected to redis..
```

1)

Notre convention va s'organiser de la manière suivante :

Nous allons avoir pour le moment 2 albums. Chacun va être composé d'un titre, une description ainsi que 4 photos.

Les commandes pour l'ajout des 2 albums sont les suivantes :

HMSET album1 Titre « Isle Album » Description « The most beautiful islands on earth » photo1 « /images/île/île1.jpg » photo2 « /images/île/île2.jpg » photo3 « /images/île/île3.jpg » photo4 « /images/île/île4.jpg »

HMSET album2 Titre « Moutain Album » Description « The most beautiful mountains on earth » photo1 « /images/montagne/montagne1.jpg » photo2 « /images/montagne/montagne2.jpg » photo3 « /images/montagne/montagne3.jpg » photo4 « /images/montagne/montagne4.jpg »

2)

On import les données de ces deux albums dans Redis :

```
client.hmset('album_mer', ['Titre','Isle Album','Description','The most beautiful islands on earth','photo1','/images/île/île1.jpg','photo2','/images/
client.hmset('album_montagne', ['Titre','Moutain Album','Description','The most beautiful mountains on earth','photo1','/images/montagne/montagne1.jpg'
```

3)

On va à présent entrer un nom pour un album « album1 » dans la base de données :

```
client.set('album1', 'Isle Slide', function(err, reply){
  console.log(reply);
});
```

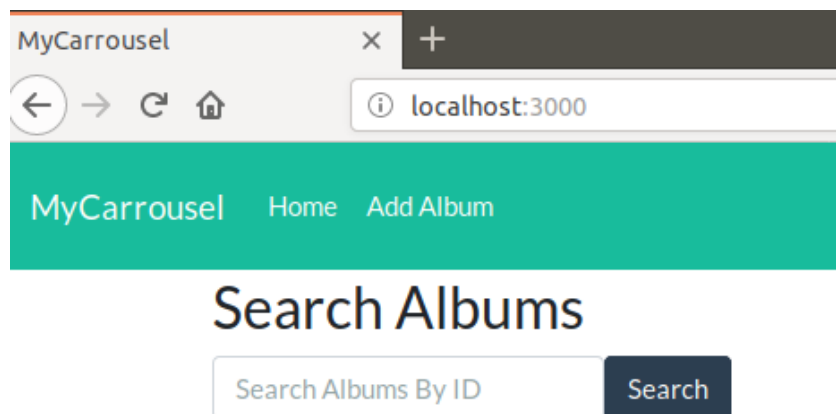
```
client.get('album1', function(err, res){
  if(err){
    console.log(err);
  } else{
    album1 = res;
    console.log('get result : '+ res);
  };
});
```

```
app.get('/', function(req, res, next){
  res.render('index', {album1:album1, album2:album2});
});
```



Isle Slide

Pour la suite j'ai décidé de faire une première page de recherche d'albums par ID pour mon application :

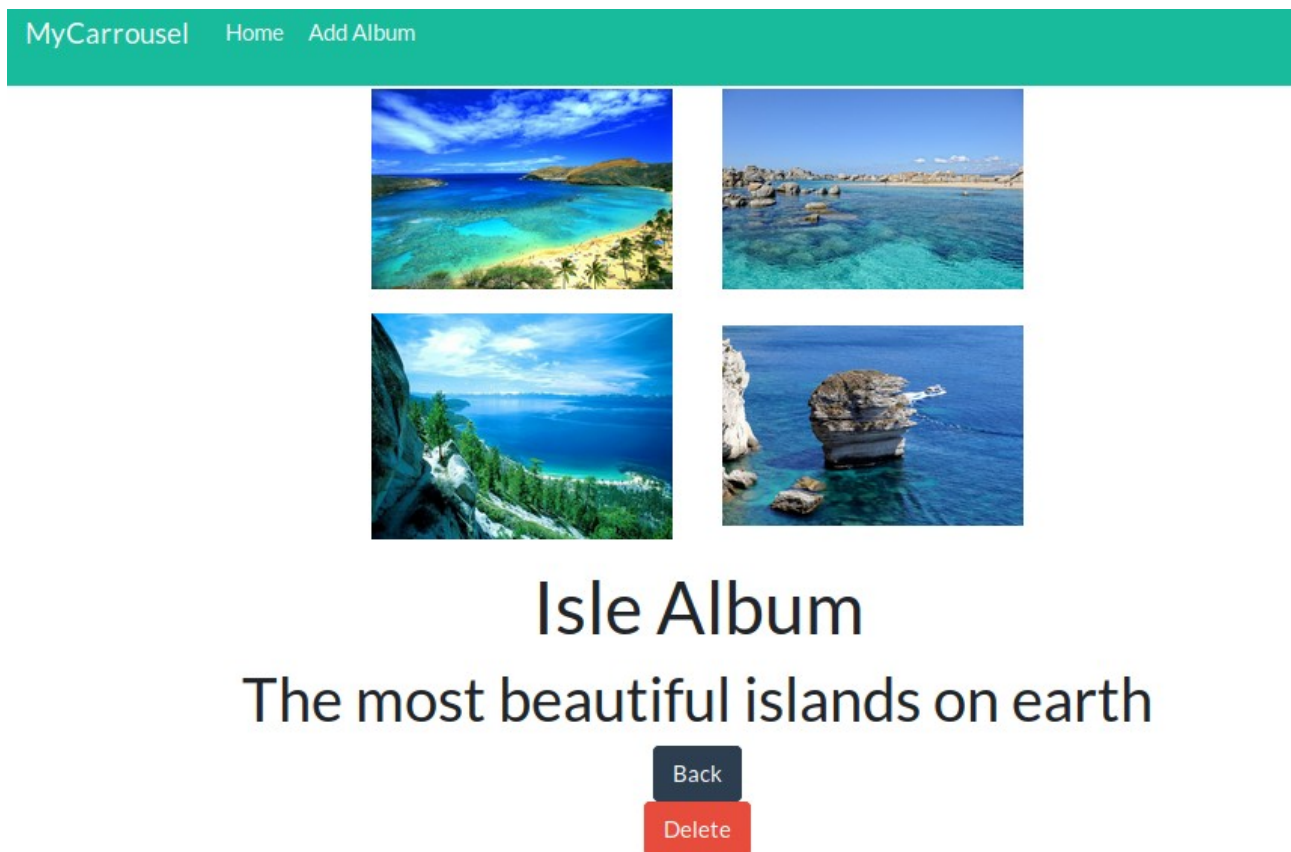



```
// Search Page
app.get('/', function(req, res, next){
  res.render('index');
});

//Search processing
app.post('/album/search', function(req, res, next){
  console.log(req.body);
  var id = req.body.id;
  client.hgetall(id, function(err, obj){
    if(!obj){
      res.render('index', {
        error: 'Album does not exist'
      });
    } else {
      obj.id = id;
      res.render('album', {
        album: obj
      });
    }
  });
});
});
```

Si l'album recherché ne se trouve pas dans la base de données, un message d'erreur est affiché, sinon on est redirigé vers la galerie de l'album correspondant à la recherche.

Si on entre « album_mer », on est redirigé vers la page suivante :

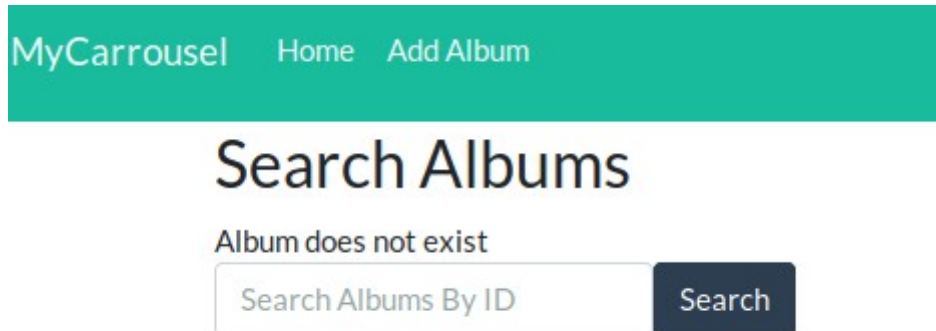


On peut revenir en arrière ou supprimer cette album.

Si on appuie sur « delete », on revient à la page de recherche d'album.

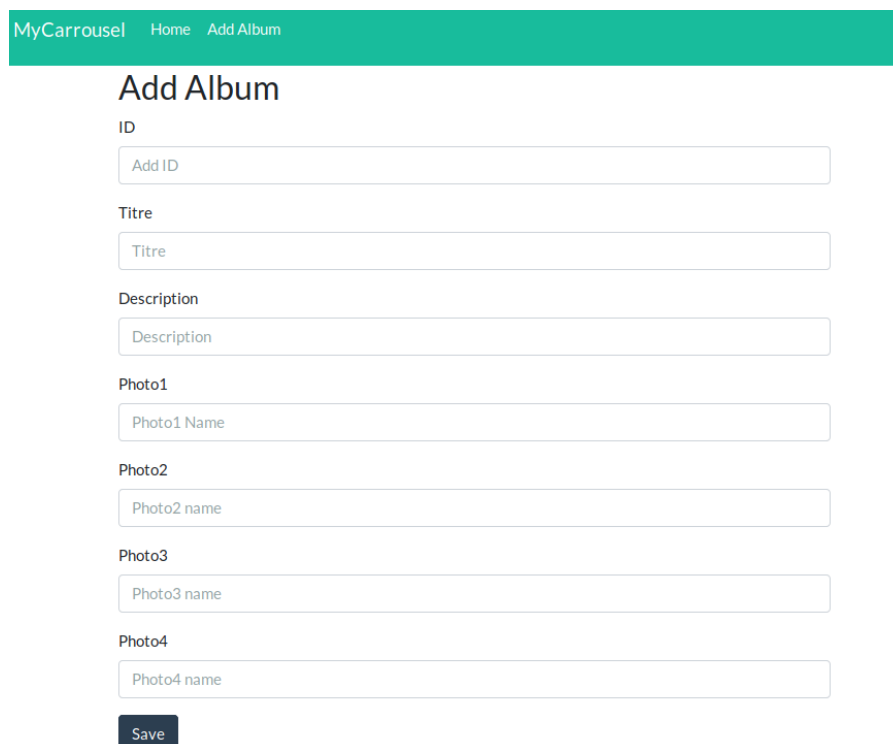
```
// Delete Album
app.delete('/album/delete/:id', function(req, res, next){
  console.log("ok");
  client.del(req.params.id);
  res.redirect('/');
});
```

On cherche à nouveau « album_mer » :



L'album « album_mer » n'est plus présent dans la base de données.

On peut également ajouter un nouvel album en cliquant sur « Add Album » dans l'onglet :



On va ajouter un nouvel album « album_desert » :

Add Album

ID

Titre

Description

Photo1

Photo2

Photo3

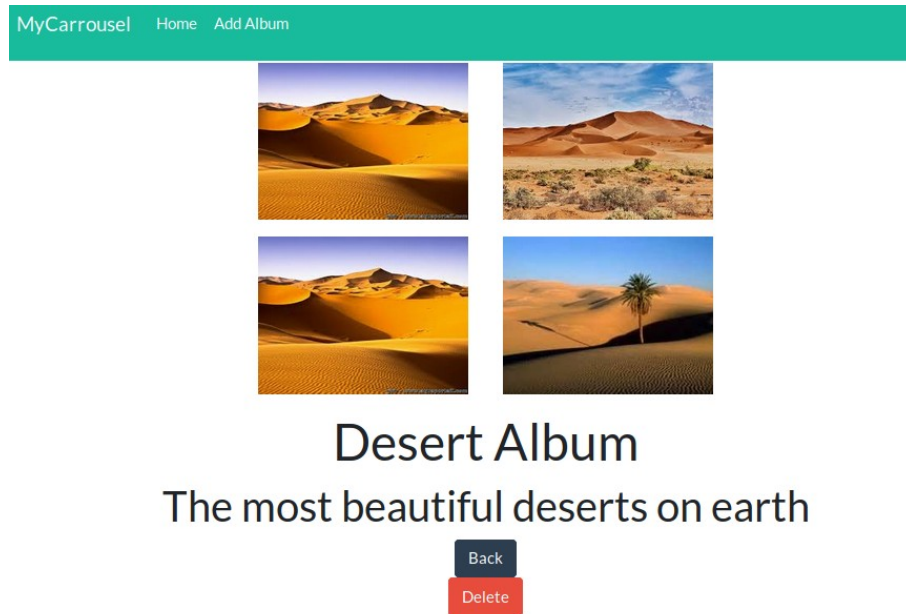
Photo4

```
// Add Album Page
app.get('/album/add', function(req, res, next){
  res.render('addalbum');
});

// Process Add Album Page
app.post('/album/add', function(req, res, next){
  let id = req.body.id;
  let Titre = req.body.Titre;
  let Description = req.body.Description;
  let photo1 = req.body.photo1;
  let photo2 = req.body.photo2;
  let photo3 = req.body.photo3;
  let photo4 = req.body.photo4;

  client.hmset(id, [
    'Titre', Titre,
    'Description', Description,
    'photo1', photo1,
    'photo2', photo2,
    'photo3', photo3,
    'photo4', photo4
  ], function(err, reply){
    if(err){
      console.log(err);
    }
    console.log(reply);
    res.redirect('/');
  });
});
```


On recherche « album_desert » dans la barre de recherche :



L'album a bien été ajouté.

Par manque de temps et ayant été bloqué pendant un certain moment je ne suis parvenu à réaliser la suite du TP. Toutefois, j'ai pu me familiariser avec la base de données Redis et comprendre son fonctionnement.