

Licenciatura en Sistemas de Información
Bases de Datos NSQL

PRACTICA #2

Parte #1 - Node JS y Redis

Ahora que ya sabemos cómo trabajar en Docker con un contenedor de Redis veamos cómo nos podemos conectar desde NodeJS:

1. Lo primero que haremos será crear un proyecto en Node JS de la siguiente manera
mkdir api
cd mkdir
npm init
2. Luego agregamos las dependencias que necesitaremos express y redis desde la consola
npm install redis --save-dev
npm install express --save-dev
3. O las agregamos en el package.json y las instalamos todas desde la consola
npm install
4. Luego creamos un script que llamaremos app.js

```
//incluimos redis a nuestro script
var redis = require('redis');

//creamos un cliente
var redisClient = redis.createClient();

redisClient.on('connect', function() {
  console.log('Conectado a Redis Server');
});
```

5. Ahora ejecutamos el script anterior con el contenedor de redis en docker activo, debería mostrar por consola que nos hemos conectado al servidor de redis
6. Probemos ahora almacenar algunos datos

Licenciatura en Sistemas de Información Bases de Datos NSQL

```
//incluimos redis a nuestro script
var redis = require('redis');

//creamos un cliente
var redisClient = redis.createClient();

redisClient.on('connect', function() {
  console.log('Conectado a Redis Server');
});

redisClient.set("key1", "hola mundo1");
redisClient.set("key2", "hola mundo2");
redisClient.set("key3", "hola mundo2");

redisClient.get("key1", function(err, value) {
  // retornara null si la key no existe
  console.log(value);
});
```

```
redisClient.exists(key, function(err, reply){
  if(err != null){
    //error
  }
});
```

7. Ahora intentemos recuperarlos para ver si todo va bien
8. Ahora carguemos una lista y mostremos su contenido

```
redisClient.set("key1",["val1","val2","val3","val4"]);
```

```
redisClient.lrange(key, 0, -1, function(err, values) {
  console.log(values)
});
```

9. Muestre los resultados del listado anterior en el localhost

Usando Docker Compose (Haciendo una Receta)

10. Primero vamos a crear el archivo Dockerfile

```
FROM node:latest
.....

WORKDIR /api
.....
COPY api/ .
.....
```

Licenciatura en Sistemas de Información Bases de Datos NSQL

11. Segundo creamos el docker-compose.yml

```
web:
  build: .
  command: sh -c 'npm install; npm start'
  ports:
    - '3000:3000'
  volumes:
    - /home/walter/nodeProjects/star_wars/api:/api
  links:
    - "db:redis"
db:
  image: redis
  ports:
    - "6379:6379"
```

12. Ahora modifiquemos un poco el script

```
var redis = require('redis')
var express = require('express')
var app = express()
var port = 3000

var cliente = redis.createClient(6379, 'redis')
app.set('port', port)

cliente.on('connect', function(){
  console.log('conectado a redis');
})

cliente.lpush("I", "luke", "yoda", "han solo", "chewbacca", redis.print)
cliente.lrange("I", 0, -1, function(err, value){
  console.log(value)
  for (var i in value){
    console.log(value[i]);
  }
});

app.listen(app.get('port'), (err) => {
  console.log(`Server running on port ${app.get('port')}`)
})
```

13. Por último el .dockerignore

Licenciatura en Sistemas de Información Bases de Datos NSQL

```
.git
.gitignore
README.md
docker-compose.yml
node_modules
npm-debug.log
```

14. Por último en el directorio donde se encuentra la receta .yml ejecutar desde la consola
docker-compose build (para construir la imagen)
docker-compose up (para levantar todas las imágenes y dejar corriendo el proyecto)

Ahora que ya maneja los conocimientos básicos genere una lista para cada uno de los episodios de la saga de Star Wars, en los cuales deberá poder cargar los correspondientes personajes:

1. Genere una ruta agregar personajes, la cual reciba como parámetro el número episodio y el nombre del personaje.
2. Genere una ruta para quitar personajes, ídem anterior.
3. Genere una ruta para listar los personajes de un episodio, la cual reciba como parámetro el número episodio.
4. Realice las mismas actividades con componentes gráficos y añádale estilos (para no ser tan rústico).

Parte #2 - Flask y Redis

Ahora veamos cómo nos podemos conectar Redis desde Python Flask:

1. Lo primero que haremos será crear un proyecto en Flask de la siguiente manera
mkdir api
cd mkdir
2. Luego instalaremos las librerías necesarias para trabajar (si aún no están instaladas)
sudo pip3 install flask
sudo pip3 install redis
3. Luego creamos un script que llamaremos app.py

Licenciatura en Sistemas de Información
Bases de Datos NSQL

```
from flask import Flask

app = Flask(__name__)

@app.route('/')
def index():
    """Retorna la pagina index."""
    return "Hola Mundo"

@app.route('/about')
def about():
    """Retorna la pagina about."""
    return 'About Python Flask'

if __name__ == '__main__':
    app.run(host='localhost', port='5000', debug=False)
```

4. Luego en la consola en la carpeta del script app.py ingresamos los siguientes comandos
set FLASK_APP = app.py
flask run
5. Ahora creemos una conexión de Flask a Redis

Licenciatura en Sistemas de Información
Bases de Datos NSQL

```
import redis

app = Flask(__name__)

def connect_db():
    """Crear conexion a base de datos."""
    conexion = redis.StrictRedis(host='127.0.0.1', port=6379, db=0)
    if(conexion.ping()):
        print("conectado al servidor de redis")
    else:
        print("error..")
    return conexion

@app.route('/')
def index():
    """Retorna la pagina index."""
    connect_db()
    return "Hola Mundo"
```

6. Probemos ahora almacenar algunos datos
7. Ahora intentemos recuperarlos para ver si todo va bien
8. Ahora carguemos una lista y mostremos su contenido
9. Veamos como cargar un template

```
from flask import Flask
from flask import render_template
import redis
```

```
<html>
<head>
  <title>Home Page Vue Flask</title>
  <link rel="stylesheet" href="https://stackpath.bootstra
  <script src="https://stackpath.bootstrapcdn.com/bootstr
  <link rel="stylesheet" href="./static/main.css">
</head>
<body>
  <h1>Hello from Template</h1>
  <h2>Python Flask - Vue js</h2>
</body>
</html>
```

Licenciatura en Sistemas de Información
Bases de Datos NSQL

```
@app.route('/')
def index():
    """Retorna la pagina index."""
    connect_db()
    return render_template('/index.html')

@app.route('/about')
def about():
    """Retorna la pagina about."""
    return 'About Python Flask'
```

10. Muestre los resultados del listado anterior en el localhost

Ahora que ya maneja los conocimientos básicos genere una lista con los capítulos de la temporada uno de The Mandalorian, los cuales podran ser alquilados para ver por una persona al mismo tiempo.

1. Genere un ruta para listar los capítulos deberá indicar si están disponibles, alquilado o reservado.
2. Cuando se alquile un capítulo deberá quedar reservado por 4 minutos, hasta que se confirme el pago, de no confirmarse el pago deberá estar disponible nuevamente una vez pasado el tiempo.
3. Genere una ruta para confirmar el pago la cual recibirá el número del capítulo y el precio, para que se pueda confirme el pago y registre el alquiler por 24 hs.
4. Realice las mismas actividades con componentes gráficos y añádale estilos (para no ser tan rústico).

Capítulos temporada 1 de The Mandalorian:

1. Chapter 1: The Mandalorian
2. Chapter 2: The Child
3. Chapter 3: The Sin
4. Chapter 4: Sanctuary
5. Chapter 5: The Gunslinger
6. Chapter 6: The Prisoner
7. Chapter 7: The Reckoning
8. Chapter 8: Redemption