Home(/)
My Planning(/planning/me)
Projects(/projects/current)
QA Reviews I can make(/corrections/to_review)
Evaluation quizzes(/dashboards/my_current_evaluation_quizzes)
Curriculums(/dashboards/my_curriculums)
Concepts(/concepts)
Conference rooms(/dashboards/video_rooms)
Servers(/servers)
Sandboxes(/user_containers/current)
Tools(/dashboards/my_tools)
Video on demand(/dashboards/videos)
Peers(/users/peers)
Discord(https://discord.com/app)
My Profile(/users/my_profile)

0x03. Queuing System in JS

Back-end JavaScript ES6 Redis NodeJS ExpressJS Kue

Weight: 1

Project over - took place from Aug 26, 2024 6:00 AM to Aug 29, 2024 6:00 AM

Manual QA Review was done on Aug 30, 2024 10:18 PM

In a nutshell…

- **Manual QA review:** 29.9/46 mandatory & 5.2/8 optional
- **Altogether: 107.25%**
  - Mandatory: 65.0%
  - Optional: 65.0%
  - Calculation: 65.0% + (65.0% * 65.0%) == **107.25%**

Overall comment:

good work, keep it up

I DON'T KNOW WHERE THE REDIS SERVER KEEPS GOING
AND AT THIS POINT, I'M TOO AFRAID TO ASK
made on imgur

Resources

Read or watch:

- Redis quick start (/rltoken/bD8ATSAVbine9-zEXenwyw)
- Redis client interface (/rltoken/vFJSkoXkIvLqHzQgx8DVcw)
- Redis client for Node JS (/rltoken/mRftfI67BrNvI-RM5JQfUA)
- Kue (/rltoken/yTC3Ci2IV2US24xJsBfMgQ) *deprecated but still use in the industry*

Learning Objectives

At the end of this project, you are expected to be able to explain to anyone (/rltoken/7yh7c3Zyy1RyUsdwifsyDg), **without the help of Google**:

- How to run a Redis server on your machine
- How to run simple operations with the Redis client
- How to use a Redis client with Node JS for basic operations
- How to store hash values in Redis
- How to deal with async operations with Redis
- How to use Kue as a queue system
- How to build a basic Express app interacting with a Redis server
- How to the build a basic Express app interacting with a Redis server and queue

Requirements

- All of your code will be compiled/interpreted on Ubuntu 18.04, Node 12.x, and Redis 5.0.7
- All of your files should end with a new line
- A `README.md` file, at the root of the folder of the project, is mandatory
- Your code should use the `js` extension

Required Files for the Project

`package.json`

Click to show/hide file contents

`.babelrc`

Click to show/hide file contents

and…

Don’t forget to run `$ npm install` when you have the `package.json`

Tasks

0. Install a redis instance

mandatory

Score: 65.0% (*Checks completed: 100.0%*)

Download, extract, and compile the latest stable Redis version (higher than 5.0.7 - https://redis.io/downloads/ (/rltoken/QDZIAT68ZH03R3pJY0ICOQ)):

```
$ wget http://download.redis.io/releases/redis-6.0.10.tar.gz
$ tar xzf redis-6.0.10.tar.gz
$ cd redis-6.0.10
$ make
```

- Start Redis in the background with `src/redis-server`

```
$ src/redis-server &
```

- Make sure that the server is working with a ping `src/redis-cli ping`

```
PONG
```

- Using the Redis client again, set the value `School` for the key `Holberton`

```
127.0.0.1:[Port]> set Holberton School
OK
127.0.0.1:[Port]> get Holberton
"School"
```

- Kill the server with the process id of the redis-server (hint: use `ps` and `grep`)

```
$ kill [PID_OF_Redis_Server]
```

Copy the `dump.rdb` from the `redis-5.0.7` directory into the root of the Queuing project.

Requirements:

- Running `get Holberton` in the client, should return `School`

Repo:

- GitHub repository: `alx-backend`
- Directory: `0x03-queuing_system_in_js`
- File: `README.md, dump.rdb`

Get a sandbox View results

1. Node Redis Client

mandatory

Score: 65.0% (*Checks completed: 100.0%*)

Install node_redis (/rltoken/mRftfI67BrNvl-RM5JQfUA) using npm

Using Babel and ES6, write a script named `0-redis_client.js`. It should connect to the Redis server running on your machine:

- It should log to the console the message `Redis client connected to the server` when the connection to Redis works correctly
- It should log to the console the message `Redis client not connected to the server: ERROR_MESSAGE` when the connection to Redis does not work

Requirements:

- To import the library, you need to use the keyword `import`

```
bob@dylan:~$ ps ax | grep redis-server
 2070 pts/1    S+     0:00 grep --color=auto redis-server
bob@dylan:~$
bob@dylan:~$ npm run dev 0-redis_client.js

> queuing_system_in_js@1.0.0 dev /root
> nodemon --exec babel-node --presets @babel/preset-env "0-redis_client.js"

[nodemon] 2.0.4
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `babel-node --presets @babel/preset-env 0-redis_client.js`
Redis client not connected to the server: Error: Redis connection to 127.0.0.1:63
79 failed - connect ECONNREFUSED 127.0.0.1:6379
Redis client not connected to the server: Error: Redis connection to 127.0.0.1:63
79 failed - connect ECONNREFUSED 127.0.0.1:6379
Redis client not connected to the server: Error: Redis connection to 127.0.0.1:63
79 failed - connect ECONNREFUSED 127.0.0.1:6379
^C
bob@dylan:~$
bob@dylan:~$ ./src/redis-server > /dev/null 2>&1 &
[1] 2073
bob@dylan:~$ ps ax | grep redis-server
 2073 pts/0    Sl     0:00 ./src/redis-server *:6379
 2078 pts/1    S+     0:00 grep --color=auto redis-server
bob@dylan:~$
bob@dylan:~$ npm run dev 0-redis_client.js

> queuing_system_in_js@1.0.0 dev /root
> nodemon --exec babel-node --presets @babel/preset-env "0-redis_client.js"

[nodemon] 2.0.4
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `babel-node --presets @babel/preset-env 0-redis_client.js`
Redis client connected to the server
^C
bob@dylan:~$
```

Repo:

- GitHub repository: `alx-backend`
- Directory: `0x03-queuing_system_in_js`
- File: `0-redis_client.js`

Get a sandbox   View results

2. Node Redis client and basic operations

mandatory

~~Score: 65.0% (*Checks completed: 100.0%*)~~

In a file `1-redis_op.js`, copy the code you previously wrote (`0-redis_client.js`).

Add two functions:

- `setNewSchool`:
  - It accepts two arguments `schoolName`, and `value`.
  - It should set in Redis the value for the key `schoolName`
  - It should display a confirmation message using `redis.print`
- `displaySchoolValue`:
  - It accepts one argument `schoolName`.
  - It should log to the console the value for the key passed as argument

At the end of the file, call:

- `displaySchoolValue('Holberton');`
- `setNewSchool('HolbertonSanFrancisco', '100');`
- `displaySchoolValue('HolbertonSanFrancisco');`

Requirements:

- Use callbacks for any of the operation, we will look at async operations later

```
bob@dylan:~$ npm run dev 1-redis_op.js

> queuing_system_in_js@1.0.0 dev /root
> nodemon --exec babel-node --presets @babel/preset-env "1-redis_op.js"

[nodemon] 2.0.4
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `babel-node --presets @babel/preset-env 1-redis_op.js`
Redis client connected to the server
School
Reply: OK
100
^C

bob@dylan:~$
```

Repo:

- GitHub repository: `alx-backend`
- Directory: `0x03-queuing_system_in_js`
- File: `1-redis_op.js`

Get a sandbox View results

(/)
Home(/)
My Planning(/planning/me)
Projects(/projects/current)
QA Reviews I can make(/corrections/to_review)
Evaluation quizzes(/dashboards/my_current_evaluation_quizzes)
Curriculums(/dashboards/my_curriculums)
Concepts(/concepts)
Conference rooms(/dashboards/video_rooms)
Servers(/servers)
Sandboxes(/user_containers/current)
Tools(/dashboards/my_tools)
Video on demand(/dashboards/videos)
Peers(/users/peers)
Discord(https://discord.com/app)
My Profile(/users/my_profile)

3. Node Redis client and async operations

mandatory

Score: 65.0% (*Checks completed: 100.0%*)

In a file `2-redis_op_async.js`, let's copy the code from the previous exercise (`1-redis_op.js`)

Using `promisify`, modify the function `displaySchoolValue` to use ES6 `async / await`

Same result as `1-redis_op.js`

```
bob@dylan:~$ npm run dev 2-redis_op_async.js

> queuing_system_in_js@1.0.0 dev /root
> nodemon --exec babel-node --presets @babel/preset-env "2-redis_op_async.js"

[nodemon] 2.0.4
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `babel-node --presets @babel/preset-env 2-redis_op_async.js`
Redis client connected to the server
School
Reply: OK
100
^C

bob@dylan:~$
```

Repo:

- GitHub repository: `alx-backend`
- Directory: `0x03-queuing_system_in_js`
- File: `2-redis_op_async.js`

>_ Get a sandbox   View results

4. Node Redis client and advanced operations

mandatory

Score: 65.0% (*Checks completed: 100.0%*)

In a file named `4-redis_advanced_op.js`, let's use the client to store a hash value

Create Hash:

Using `hset`, let's store the following:

- The key of the hash should be `HolbertonSchools`
- It should have a value for:
  - `Portland=50`
  - `Seattle=80`
  - `New York=20`

- `Bogota=20`
- `Cali=40`
- `Paris=2`

- Make sure you use `redis.print` for each `hset`

Display Hash:

Using `hgetall`, display the object stored in Redis. It should return the following:

Requirements:

- Use callbacks for any of the operation, we will look at async operations later

```
bob@dylan:~$ npm run dev 4-redis_advanced_op.js

> queuing_system_in_js@1.0.0 dev /root
> nodemon --exec babel-node --presets @babel/preset-env "4-redis_advanced_op.js"

[nodemon] 2.0.4
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `babel-node --presets @babel/preset-env 4-redis_advanced_op.js`
Redis client connected to the server
Reply: 1
Reply: 1
Reply: 1
Reply: 1
Reply: 1
Reply: 1
{
  Portland: '50',
  Seattle: '80',
  'New York': '20',
  Bogota: '20',
  Cali: '40',
  Paris: '2'
}
^C
bob@dylan:~$
```

Repo:

- GitHub repository: `alx-backend`
- Directory: `0x03-queuing_system_in_js`
- File: `4-redis_advanced_op.js`

Get a sandbox View results

5. Node Redis client publisher and subscriber

mandatory

(/)
Home(/)
My Planning(/planning/me)
Projects(/projects/current)
QA Reviews I can make(/corrections/to_review)
Evaluation quizzes(/dashboards/my_current_evaluation_quizzes)
Curriculums(/dashboards/my_curriculums)
Concepts(/concepts)
Conference rooms(/dashboards/video_rooms)
Servers(/servers)
Sandboxes(/user_containers/current)
Tools(/dashboards/my_tools)
Video on demand(/dashboards/videos)
Peers(/users/peers)
Discord(https://discord.com/app)
My Profile(/users/my_profile)

Score: 65.0% (*Checks completed: 100.0%*)

In a file named `5-subscriber.js`, create a redis client:

- On connect, it should log the message `Redis client connected to the server`
- On error, it should log the message `Redis client not connected to the server: ERROR MESSAGE`
- It should subscribe to the channel `holberton school channel`
- When it receives message on the channel `holberton school channel`, it should log the message to the console
- When the message is `KILL_SERVER`, it should unsubscribe and quit

In a file named `5-publisher.js`, create a redis client:

- On connect, it should log the message `Redis client connected to the server`
- On error, it should log the message `Redis client not connected to the server: ERROR MESSAGE`
- Write a function named `publishMessage`:
  - It will take two arguments: `message` (string), and `time` (integer - in ms)
  - After `time` millisecond:
    - The function should log to the console `About to send MESSAGE`
    - The function should publish to the channel `holberton school channel`, the message passed in argument after the time passed in arguments
- At the end of the file, call:

```
publishMessage("Holberton Student #1 starts course", 100);
publishMessage("Holberton Student #2 starts course", 200);
publishMessage("KILL_SERVER", 300);
publishMessage("Holberton Student #3 starts course", 400);
```

Requirements:

- You only need one Redis server to execute the program
- You will need to have two node processes to run each script at the same time

Terminal 1:

```
bob@dylan:~$ npm run dev 5-subscriber.js

> queuing_system_in_js@1.0.0 dev /root
> nodemon --exec babel-node --presets @babel/preset-env "5-subscriber.js"

[nodemon] 2.0.4
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `babel-node --presets @babel/preset-env 5-subscriber.js`
Redis client connected to the server
```

Terminal 2:

```
bob@dylan:~$ npm run dev 5-publisher.js

> queuing_system_in_js@1.0.0 dev /root
> nodemon --exec babel-node --presets @babel/preset-env "5-publisher.js"

[nodemon] 2.0.4
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `babel-node --presets @babel/preset-env 5-publisher.js`
Redis client connected to the server
About to send Holberton Student #1 starts course
About to send Holberton Student #2 starts course
About to send KILL_SERVER
About to send Holberton Student #3 starts course
^C
bob@dylan:~$
```

And in the same time in Terminal 1:

```
Redis client connected to the server
Holberton Student #1 starts course
Holberton Student #2 starts course
KILL_SERVER
[nodemon] clean exit - waiting for changes before restart
^C
bob@dylan:~$
```

Now you have a basic Redis-based queuing system where you have a process to generate job and a second one to process it. These 2 processes can be in 2 different servers, which we also call "background workers".

Home(/)
My Planning(/planning/me)
Projects(/projects/current)
QA Reviews I can make(/corrections/to_review)
Evaluation quizzes(/dashboards/my_current_evaluation_quizzes)
Curriculums(/dashboards/my_curriculums)
Concepts(/concepts)
Conference rooms(/dashboards/video_rooms)
Servers(/servers)
Sandboxes(/user_containers/current)
Tools(/dashboards/my_tools)
Video on demand(/dashboards/videos)
Peers(/users/peers)
Discord(https://discord.com/app)
My Profile(/users/my_profile)

Repo:

- GitHub repository: `alx-backend`
- Directory: `0x03-queuing_system_in_js`
- File: `5-subscriber.js, 5-publisher.js`

>_ Get a sandbox View results

6. Create the Job creator

mandatory

Score: 65.0% (*Checks completed: 100.0%*)

In a file named `6-job_creator.js` :

- Create a queue with `Kue`
- Create an object containing the Job data with the following format:

```
{
  phoneNumber: string,
  message: string,
}
```

- Create a queue named `push_notification_code`, and create a job with the object created before
- When the job is created without error, log to the console `Notification job created: JOB ID`
- When the job is completed, log to the console `Notification job completed`
- When the job is failing, log to the console `Notification job failed`

```
bob@dylan:~$ npm run dev 6-job_creator.js

> queuing_system_in_js@1.0.0 dev /root
> nodemon --exec babel-node --presets @babel/preset-env "6-job_creator.js"

[nodemon] 2.0.4
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `babel-node --presets @babel/preset-env 6-job_creator.js`
Notification job created: 1
```

Nothing else will happen - to process the job, go to the next task!

If you execute multiple time this file, you will see the `JOB ID` increasing - it means you are storing new job to process…

Repo:

- GitHub repository: `alx-backend`
- Directory: `0x03-queuing_system_in_js`
- File: `6-job_creator.js`

Get a sandbox View results

7. Create the Job processor

mandatory

Score: 65.0% (*Checks completed: 100.0%*)

In a file named `6-job_processor.js`:

- Create a queue with `Kue`
- Create a function named `sendNotification`:
  - It will take two arguments `phoneNumber` and `message`
  - It will log to the console `Sending notification to PHONE_NUMBER, with message: MESSAGE`
- Write the queue process that will listen to new jobs on `push_notification_code`:

(/)
Home(/)
My Planning(/planning/me)
Projects(/projects/current)
QA Reviews I can make(/corrections/to_review)
Evaluation quizzes(/dashboards/my_current_evaluation_quizzes)
Curriculums(/dashboards/my_curriculums)
Concepts(/concepts)
Conference rooms(/dashboards/video_rooms)
Servers(/servers)
Sandboxes(/user_containers/current)
Tools(/dashboards/my_tools)
Video on demand(/dashboards/videos)
Peers(/users/peers)
Discord(https://discord.com/app)
My Profile(/users/my_profile)

- Every new job should call the `sendNotification` function with the phone number and the message contained within the job data

(/)
Home(/)
My Planning(/planning/me)
Projects(/projects/current)
QA Reviews I can make(/corrections/to_review)
Evaluation quizzes(/dashboards/my_current_evaluation_quizzes)
Curriculums(/dashboards/my_curriculums)
Concepts(/concepts)
Conference rooms(/dashboards/video_rooms)
Servers(/servers)
Sandboxes(/user_containers/current)
Tools(/dashboards/my_tools)
Video on demand(/dashboards/videos)
Peers(/users/peers)
Discord(https://discord.com/app)
My Profile(/users/my_profile)

Requirements:

- You only need one Redis server to execute the program
- You will need to have two node processes to run each script at the same time
- You muse use `Kue` to set up the queue

Terminal 2:

```
bob@dylan:~$ npm run dev 6-job_processor.js

> queuing_system_in_js@1.0.0 dev /root
> nodemon --exec babel-node --presets @babel/preset-env "6-job_processor.js"

[nodemon] 2.0.4
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `babel-node --presets @babel/preset-env 6-job_processor.js`
Sending notification to 4153518780, with message: This is the code to verify your
account
```

Terminal 1: let's queue a new job!

```
bob@dylan:~$ npm run dev 6-job_creator.js

> queuing_system_in_js@1.0.0 dev /root
> nodemon --exec babel-node --presets @babel/preset-env "6-job_creator.js"

[nodemon] 2.0.4
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `babel-node --presets @babel/preset-env 6-job_creator.js`
Notification job created: 2
```

And in the same time in Terminal 2:

```
Sending notification to 4153518780, with message: This is the code to verify your
account
```

BOOM! same as `5-subscriber.js` and `5-publisher.js` but with a module to manage jobs.

Repo:

- GitHub repository: `alx-backend`
- Directory: `0x03-queuing_system_in_js`
- File: `6-job_processor.js`

Get a sandbox View results

(/)

8. Track progress and errors with Kue: Create the Job creator

mandatory

Home(/)

Score: 65.0% (*Checks completed: 100.0%*)

My Planning(/planning/me)

a file named `7-job_creator.js` :

Create an array `jobs` with the following data inside:

Projects(/projects/current)

QA Reviews I can make(/corrections/to_review)

Evaluation quizzes(/dashboards/my_current_evaluation_quizzes)

Curriculums(/dashboards/my_curriculums)

Concepts(/concepts)

Conference rooms(/dashboards/video_rooms)

Servers(/servers)

Sandboxes(/user_containers/current)

Tools(/dashboards/my_tools)

Video on demand(/dashboards/videos)

Peers(/users/peers)

Discord(https://discord.com/app)

My Profile(/users/my_profile)

```
const jobs = [
  {
    phoneNumber: '4153518780',
    message: 'This is the code 1234 to verify your account'
  },
  {
    phoneNumber: '4153518781',
    message: 'This is the code 4562 to verify your account'
  },
  {
    phoneNumber: '4153518743',
    message: 'This is the code 4321 to verify your account'
  },
  {
    phoneNumber: '4153538781',
    message: 'This is the code 4562 to verify your account'
  },
  {
    phoneNumber: '4153118782',
    message: 'This is the code 4321 to verify your account'
  },
  {
    phoneNumber: '4153718781',
    message: 'This is the code 4562 to verify your account'
  },
  {
    phoneNumber: '4159518782',
    message: 'This is the code 4321 to verify your account'
  },
  {
    phoneNumber: '4158718781',
    message: 'This is the code 4562 to verify your account'
  },
  {
    phoneNumber: '4153818782',
    message: 'This is the code 4321 to verify your account'
  },
  {
    phoneNumber: '4154318781',
    message: 'This is the code 4562 to verify your account'
  },
  {
    phoneNumber: '4151218782',
    message: 'This is the code 4321 to verify your account'
  }
];
```

After this array created:

- Create a queue with `Kue`
- Write a loop that will go through the array `jobs` and for each object:
  - Create a new job to the queue `push_notification_code_2` with the current object
  - If there is no error, log to the console `Notification job created: JOB_ID`
  - On the job completion, log to the console `Notification job JOB_ID completed`
  - On the job failure, log to the console `Notification job JOB_ID failed: ERROR`

- On the job progress, log to the console `Notification job JOB_ID PERCENTAGE% complete`

(/)
Home(/)
My Planning(/planning/me)
Projects(/projects/current)
QA Reviews I can make(/corrections/to_review)
Evaluation quizzes(/dashboards/my_current_evaluation_quizzes)
Curriculums(/dashboards/my_curriculums)
Concepts(/concepts)
Conference rooms(/dashboards/video_rooms)
Servers(/servers)
Sandboxes(/user_containers/current)
Tools(/dashboards/my_tools)
Video on demand(/dashboards/videos)
Peers(/users/peers)
Discord(https://discord.com/app)
My Profile(/users/my_profile)

Terminal 1:

```
bob@dylan:~$ npm run dev 7-job_creator.js

> queuing_system_in_js@1.0.0 dev /root
> nodemon --exec babel-node --presets @babel/preset-env "7-job_creator.js"

[nodemon] 2.0.4
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `babel-node --presets @babel/preset-env 7-job_creator.js`
Notification job created: 39
Notification job created: 40
Notification job created: 41
Notification job created: 42
Notification job created: 43
Notification job created: 44
Notification job created: 45
Notification job created: 46
Notification job created: 47
Notification job created: 48
Notification job created: 49
```

Repo:

- GitHub repository: `alx-backend`
- Directory: `0x03-queuing_system_in_js`
- File: `7-job_creator.js`

Get a sandbox View results

9. Track progress and errors with Kue: Create the Job processor

mandatory

Score: 65.0% (*Checks completed: 100.0%*)

In a file named `7-job_processor.js`:

Create an array that will contain the blacklisted phone numbers. Add in it `4153518780` and `4153518781` - these 2 numbers will be blacklisted by our jobs processor.

Create a function `sendNotification` that takes 4 arguments: `phoneNumber`, `message`, `job`, and `done`:

- When the function is called, track the progress of the `job` of `0` out of `100`
- If `phoneNumber` is included in the “blacklisted array”, fail the job with an `Error` object and the message: `Phone number PHONE_NUMBER is blacklisted`
- Otherwise:

    - Track the progress to 50%
    - Log to the console `Sending notification to PHONE_NUMBER, with message: MESSAGE`

Create a queue with `Kue` that will proceed job of the queue `push_notification_code_2` with two jobs at a time.

Requirements:

- You only need one Redis server to execute the program
- You will need to have two node processes to run each script at the same time
- You muse use `Kue` to set up the queue
- Executing the jobs list should log to the console the following:

Terminal 2:

```
bob@dylan:~$ npm run dev 7-job_processor.js

> queuing_system_in_js@1.0.0 dev /root
> nodemon --exec babel-node --presets @babel/preset-env "7-job_processor.js"

[nodemon] 2.0.4
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `babel-node --presets @babel/preset-env 7-job_processor.js`
Sending notification to 4153518743, with message: This is the code 4321 to verify
your account
Sending notification to 4153538781, with message: This is the code 4562 to verify
your account
Sending notification to 4153118782, with message: This is the code 4321 to verify
your account
Sending notification to 4153718781, with message: This is the code 4562 to verify
your account
Sending notification to 4159518782, with message: This is the code 4321 to verify
your account
Sending notification to 4158718781, with message: This is the code 4562 to verify
your account
Sending notification to 4153818782, with message: This is the code 4321 to verify
your account
Sending notification to 4154318781, with message: This is the code 4562 to verify
your account
Sending notification to 4151218782, with message: This is the code 4321 to verify
your account
```

And in the same time in terminal 1:

```
...
Notification job #39 0% complete
Notification job #40 0% complete
Notification job #39 failed: Phone number 4153518780 is blacklisted
Notification job #40 failed: Phone number 4153518781 is blacklisted
Notification job #41 0% complete
Notification job #41 50% complete
Notification job #42 0% complete
Notification job #42 50% complete
Notification job #41 completed
Notification job #42 completed
Notification job #43 0% complete
Notification job #43 50% complete
Notification job #44 0% complete
Notification job #44 50% complete
Notification job #43 completed
Notification job #44 completed
Notification job #45 0% complete
Notification job #45 50% complete
Notification job #46 0% complete
Notification job #46 50% complete
Notification job #45 completed
Notification job #46 completed
Notification job #47 0% complete
Notification job #47 50% complete
Notification job #48 0% complete
Notification job #48 50% complete
Notification job #47 completed
Notification job #48 completed
Notification job #49 0% complete
Notification job #49 50% complete
Notification job #49 completed
```

Repo:

- GitHub repository: `alx-backend`
- Directory: `0x03-queuing_system_in_js`
- File: `7-job_processor.js`

Get a sandbox View results

10. Writing the job creation function

mandatory

Score: 65.0% (*Checks completed: 100.0%*)

In a file named `8-job.js`, create a function named `createPushNotificationsJobs`:

- It takes into argument `jobs` (array of objects), and `queue` (`Kue` queue)
- If `jobs` is not an array, it should throw an `Error` with message: `Jobs is not an array`
- For each job in `jobs`, create a job in the queue `push_notification_code_3`
- When a job is created, it should log to the console `Notification job created: JOB_ID`

- When a job is complete, it should log to the console `Notification job JOB_ID completed`
- When a job is failed, it should log to the console `Notification job JOB_ID failed: ERROR`
- When a job is making progress, it should log to the console `Notification job JOB_ID PERCENT% complete`

```
bob@dylan:~$ cat 8-job-main.js
import kue from 'kue';

import createPushNotificationsJobs from './8-job.js';

const queue = kue.createQueue();

const list = [
    {
        phoneNumber: '4153518780',
        message: 'This is the code 1234 to verify your account'
    }
];
createPushNotificationsJobs(list, queue);

bob@dylan:~$
bob@dylan:~$ npm run dev 8-job-main.js

> queuing_system_in_js@1.0.0 dev /root
> nodemon --exec babel-node --presets @babel/preset-env "8-job-main.js"

[nodemon] 2.0.4
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `babel-node --presets @babel/preset-env 8-job-main.js`
Notification job created: 51
```

Repo:

- GitHub repository: `alx-backend`
- Directory: `0x03-queuing_system_in_js`
- File: `8-job.js`

Get a sandbox View results

11. Writing the test for job creation

mandatory

Score: 65.0% (*Checks completed: 100.0%*)

Now that you created a job creator, let's add tests:

- Import the function `createPushNotificationsJobs`
- Create a queue with `Kue`
- Write a test suite for the `createPushNotificationsJobs` function:

(/)
Home(/)
My Planning(/planning/me)
Projects(/projects/current)
QA Reviews I can make(/corrections/to_review)
Evaluation quizzes(/dashboards/my_current_evaluation_quizzes)
Curriculums(/dashboards/my_curriculums)
Concepts(/concepts)
Conference rooms(/dashboards/video_rooms)
Servers(/servers)
Sandboxes(/user_containers/current)
Tools(/dashboards/my_tools)
Video on demand(/dashboards/videos)
Peers(/users/peers)
Discord(https://discord.com/app)
My Profile(/users/my_profile)

- Use `queue.testMode` to validate which jobs are inside the queue
- etc.

Requirements:

- Make sure to enter the test mode without processing the jobs before executing the tests
- Make sure to clear the queue and exit the test mode after executing the tests

```
bob@dylan:~$ npm test 8-job.test.js

> queuing_system_in_js@1.0.0 test /root
> mocha --require @babel/register --exit "8-job.test.js"



  createPushNotificationsJobs
    ✓ display a error message if jobs is not an array
Notification job created: 1
Notification job created: 2
    ✓ create two new jobs to the queue
...

  123 passing (417ms)
```

Repo:

- GitHub repository: `alx-backend`
- Directory: `0x03-queuing_system_in_js`
- File: `8-job.test.js`

View results

12. In stock?

mandatory

Score: 65.0% (*Checks completed: 100.0%*)

Data

Create an array `listProducts` containing the list of the following products:

- Id: 1, name: `Suitcase 250` , price: 50, stock: 4
- Id: 2, name: `Suitcase 450` , price: 100, stock: 10
- Id: 3, name: `Suitcase 650` , price: 350, stock: 2
- Id: 4, name: `Suitcase 1050` , price: 550, stock: 5

Data access

Create a function named `getItemById` :

- It will take `id` as argument
- It will return the item from `listProducts` with the same id

(/)
Home(/)
My Planning(/planning/me)
Projects(/projects/current)
QA Reviews I can make(/corrections/to_review)
Evaluation quizzes(/dashboards/my_current_evaluation_quizzes)
Curriculums(/dashboards/my_curriculums)
Concepts(/concepts)
Conference rooms(/dashboards/video_rooms)
Servers(/servers)
Sandboxes(/user_containers/current)
Tools(/dashboards/my_tools)
Video on demand(/dashboards/videos)
Peers(/users/peers)
Discord(https://discord.com/app)
My Profile(/users/my_profile)

Server

Create an `express` server listening on the port 1245. (You will start it via: `npm run dev 9-stock.js`)

Products

Create the route `GET /list_products` that will return the list of every available product with the following JSON format:

```
bob@dylan:~$ curl localhost:1245/list_products ; echo ""
[{"itemId":1,"itemName":"Suitcase 250","price":50,"initialAvailableQuantity":4},
{"itemId":2,"itemName":"Suitcase 450","price":100,"initialAvailableQuantity":10},
{"itemId":3,"itemName":"Suitcase 650","price":350,"initialAvailableQuantity":2},
{"itemId":4,"itemName":"Suitcase 1050","price":550,"initialAvailableQuantity":5}]
bob@dylan:~$
```

In stock in Redis

Create a client to connect to the Redis server:

- Write a function `reserveStockById` that will take `itemId` and `stock` as arguments:
  - It will set in Redis the stock for the key `item.ITEM_ID`
- Write an async function `getCurrentReservedStockById` , that will take `itemId` as an argument:
  - It will return the reserved stock for a specific item

Product detail

Create the route `GET /list_products/:itemId` , that will return the current product and the current available stock (by using `getCurrentReservedStockById`) with the following JSON format:

```
bob@dylan:~$ curl localhost:1245/list_products/1 ; echo ""
{"itemId":1,"itemName":"Suitcase 250","price":50,"initialAvailableQuantity":4,"cu
rrentQuantity":4}
bob@dylan:~$
```

If the item does not exist, it should return:

```
bob@dylan:~$ curl localhost:1245/list_products/12 ; echo ""
{"status":"Product not found"}
bob@dylan:~$
```

Reserve a product

Create the route `GET /reserve_product/:itemId` :

- If the item does not exist, it should return:

```
bob@dylan:~$ curl localhost:1245/reserve_product/12 ; echo ""
{"status":"Product not found"}
bob@dylan:~$
```

- If the item exists, it should check that there is at least one stock available. If not it should return:

```
bob@dylan:~$ curl localhost:1245/reserve_product/1 ; echo ""
{"status":"Not enough stock available","itemId":1}
bob@dylan:~$
```

- If there is enough stock available, it should reserve one item (by using `reserveStockById`), and return:

```
bob@dylan:~$ curl localhost:1245/reserve_product/1 ; echo ""
{"status":"Reservation confirmed","itemId":1}
bob@dylan:~$
```

Requirements:

- Make sure to use `promisify` with Redis
- Make sure to use the `await` / `async` keyword to get the value from Redis
- Make sure the format returned by the web application is always JSON and not text

Repo:

- GitHub repository: `alx-backend`
- Directory: `0x03-queuing_system_in_js`
- File: `9-stock.js`

View results

13. Can I have a seat?

#advanced

Score: 65.0% (*Checks completed: 100.0%*)

Redis

Create a Redic client:

- Create a function `reserveSeat`, that will take into argument `number`, and set the key `available_seats` with the number
- Create a function `getCurrentAvailableSeats`, it will return the current number of available seats (by using `promisify` for Redis)
- When launching the application, set the number of available to 50
- Initialize the boolean `reservationEnabled` to `true` - it will be turn to `false` when no seat will be available

Kue queue

Create a `Kue queue`

Server

Create an express server listening on the port 1245. (You will start it via: `npm run dev 100-seat.js`)

Add the route `GET /available_seats` that returns the number of seat available:

```
bob@dylan:~$ curl localhost:1245/available_seats ; echo ""
{"numberOfAvailableSeats":"50"}
bob@dylan:~$
```

Add the route `GET /reserve_seat` that:

(/)
Home(/)
My Planning(/planning/me)
Projects(/projects/current)
QA Reviews I can make(/corrections/to_review)
Evaluation quizzes(/dashboards/my_current_evaluation_quizzes)
Curriculums(/dashboards/my_curriculums)
Concepts(/concepts)
Conference rooms(/dashboards/video_rooms)
Servers(/servers)
Sandboxes(/user_containers/current)
Tools(/dashboards/my_tools)
Video on demand(/dashboards/videos)
Peers(/users/peers)
Discord(https://discord.com/app)
My Profile(/users/my_profile)

(/)

Home(/)

My Planning(/planning/me)

Projects(/projects/current)

QA Reviews I can make(/corrections/to_review)

Evaluation quizzes(/dashboards/my_current_evaluation_quizzes)

Curriculums(/dashboards/my_curriculums)

Concepts(/concepts)

Conference rooms(/dashboards/video_rooms)

Servers(/servers)

Sandboxes(/user_containers/current)

Tools(/dashboards/my_tools)

Video on demand(/dashboards/videos)

Peers(/users/peers)

Discord(https://discord.com/app)

My Profile(/users/my_profile)

- Returns `{ "status": "Reservation are blocked" }` if `reservationEnabled` is `false`
- Creates and queues a job in the queue `reserve_seat`:
  - Save the job and return:
    - `{ "status": "Reservation in process" }` if no error
    - Otherwise: `{ "status": "Reservation failed" }`
  - When the job is completed, print in the console: `Seat reservation job JOB_ID completed`
  - When the job failed, print in the console: `Seat reservation job JOB_ID failed: ERROR_MESSAGE`

```
bob@dylan:~$ curl localhost:1245/reserve_seat ; echo ""
{"status":"Reservation in process"}
bob@dylan:~$
```

Add the route `GET /process` that:

- Returns `{ "status": "Queue processing" }` just after:
- Process the queue `reserve_seat` (async):
  - Decrease the number of seat available by using `getCurrentAvailableSeats` and `reserveSeat`
  - If the new number of available seats is equal to 0, set `reservationEnabled` to false
  - If the new number of available seats is more or equal than 0, the job is successful
  - Otherwise, fail the job with an `Error` with the message `Not enough seats available`

```
bob@dylan:~$ curl localhost:1245/process ; echo ""
{"status":"Queue processing"}
bob@dylan:~$
bob@dylan:~$ curl localhost:1245/available_seats ; echo ""
{"numberOfAvailableSeats":"49"}
bob@dylan:~$
```

and in the server terminal:

```
Seat reservation job 52 completed
```

and you can reserve all seats:

```
bob@dylan:~$ for n in {1..50}; do curl localhost:1245/reserve_seat ; echo ""; don
e
{"status":"Reservation in process"}
{"status":"Reservation in process"}
...
{"status":"Reservation in process"}
{"status":"Reservation in process"}
{"status":"Reservation in process"}
{"status":"Reservation are blocked"}
{"status":"Reservation are blocked"}
{"status":"Reservation are blocked"}
bob@dylan:~$
```

Requirements:

- Make sure to use `promisify` with Redis

- Make sure to use the `await` / `async` keyword to get the value from Redis
- Make sure the format returned by the web application is always JSON and not text
- Make sure that only the allowed amount of seats can be reserved
- Make sure that the main route is displaying the right number of seats

Repo:

- GitHub repository: `alx-backend`
- Directory: `0x03-queuing_system_in_js`
- File: `100-seat.js`

Get a sandbox View results

Ready for a new review