(/)

Curriculum
Short Specializations
Average: **97.3%**

0x04. Files manager

Back-end **JavaScript** **ES6** **NoSQL** **MongoDB** **Redis** **NodeJS** **ExpressJS** **Kue**

By: Guillaume, CTO at Holberton School

Weight: 1

Project to be done in teams of 2 people (your team: Baraka Ngaira, Igbe Ajaga)

Project over - took place from Feb 8, 2024 6:00 AM to Feb 15, 2024 6:00 AM

Manual QA review was done by Browni Wilson on Feb 15, 2024 3:07 AM

An auto review will be launched at the deadline

In a nutshell…

- **Contribution:** 100.0%
- **Manual QA review:** 12.0/12 mandatory & 20.0/20 optional
- **Auto QA review:** 38.0/86 mandatory
- **Altogether: 102.04%**
  - Mandatory: 51.02%
  - Optional: 100.0%
  - Contribution: 100.0%
  - Calculation: 100.0% * (51.02% + (51.02% * 100.0%)) == **102.04%**

Overall comment:
Awesome work buddy. This clearly shows you are on your path as a software engineer. Bravo!!

This project is a summary of this back-end trimester: authentication, NodeJS, MongoDB, Redis, pagination and background processing.

The objective is to build a simple platform to upload and view files:

- User authentication via a token

(/)

- List all files
- Upload a new file
- Change permission of a file
- View a file
- Generate thumbnails for images

You will be guided step by step for building it, but you have some freedoms of implementation, split in more files etc… (`utils` folder will be your friend)

Of course, this kind of service already exists in the real life - it's a learning purpose to assemble each piece and build a full product.

Enjoy!

Resources

Read or watch:

- Node JS getting started (/rltoken/buFPHJYnZjtOrTd610j6Og)
- Process API doc (/rltoken/uYPplj2cPK8pcP0LtV6RuA)
- Express getting started (/rltoken/SujfeWKCWmUMomfETjETEg)
- Mocha documentation (/rltoken/FzEwplmoZiyGvkgKllZNJw)
- Nodemon documentation (/rltoken/pdNNTX0OLugbhxvP3sLgOw)
- MongoDB (/rltoken/g1x7y_3GskzVAJBTXcSjmA)
- Bull (/rltoken/NkHBpGrxnd0sK_fDPMbihg)
- Image thumbnail (/rltoken/KX6cck2nyLpQOTDMLcwxLg)
- Mime-Types (/rltoken/j9B0Kc-4HDKLUe88ShbOjQ)
- Redis (/rltoken/nqwKRszO8Tkj_ZWW1EFwGw)

Learning Objectives

At the end of this project, you are expected to be able to explain to anyone (/rltoken/88vbnogJmkEoxqu-6wAXEw), **without the help of Google**:

- how to create an API with Express
- how to authenticate a user
- how to store data in MongoDB
- how to store temporary data in Redis
- how to setup and use a background worker

Requirements

- Allowed editors: `vi`, `vim`, `emacs`, `Visual Studio Code`
- All your files will be interpreted/compiled on Ubuntu 18.04 LTS using `node` (version 12.x.x)
- All 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
- Your code will be verified against lint using ESLint

(/)

Provided files

`package.json`

Click to show/hide file contents

`.eslintrc.js`

Click to show/hide file contents

`babel.config.js`

Click to show/hide file contents

and…

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

Tasks

0. Redis utils

mandatory

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

Inside the folder `utils`, create a file `redis.js` that contains the class `RedisClient`.

`RedisClient` should have:

- the constructor that creates a client to Redis:
  - any error of the redis client must be displayed in the console (you should use `on('error')` of the redis client)
- a function `isAlive` that returns `true` when the connection to Redis is a success otherwise, `false`
- an asynchronous function `get` that takes a string key as argument and returns the Redis value stored for this key
- an asynchronous function `set` that takes a string key, a value and a duration in second as arguments to store it in Redis (with an expiration set by the duration argument)
- an asynchronous function `del` that takes a string key as argument and remove the value in Redis for this key

After the class definition, create and export an instance of `RedisClient` called `redisClient`.

```
bob@dylan:~$ cat main.js
import redisClient from './utils/redis';

(async () => {
    console.log(redisClient.isAlive());
    console.log(await redisClient.get('myKey'));
    await redisClient.set('myKey', 12, 5);
    console.log(await redisClient.get('myKey'));

    setTimeout(async () => {
        console.log(await redisClient.get('myKey'));
    }, 1000*10)
})();

bob@dylan:~$ npm run dev main.js
true
null
12
null
bob@dylan:~$
```

Repo:

- GitHub repository: `alx-files_manager`
- File: `utils/redis.js`

Done! Help Check your code Get a sandbox QA Review

1. MongoDB utils

mandatory

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

Inside the folder `utils`, create a file `db.js` that contains the class `DBClient`.

`DBClient` should have:

- the constructor that creates a client to MongoDB:
  - host: from the environment variable `DB_HOST` or default: `localhost`
  - port: from the environment variable `DB_PORT` or default: `27017`
  - database: from the environment variable `DB_DATABASE` or default: `files_manager`
- a function `isAlive` that returns `true` when the connection to MongoDB is a success otherwise, `false`
- an asynchronous function `nbUsers` that returns the number of documents in the collection `users`
- an asynchronous function `nbFiles` that returns the number of documents in the collection `files`

After the class definition, create and export an instance of `DBClient` called `dbClient`.

```
bob@dylan:~$ cat main.js
import dbClient from './utils/db';

const waitConnection = () => {
    return new Promise((resolve, reject) => {
        let i = 0;
        const repeatFct = async () => {
            await setTimeout(() => {
                i += 1;
                if (i >= 10) {
                    reject()
                }
                else if(!dbClient.isAlive()) {
                    repeatFct()
                }
                else {
                    resolve()
                }
            }, 1000);
        };
        repeatFct();
    })
};

(async () => {
    console.log(dbClient.isAlive());
    await waitConnection();
    console.log(dbClient.isAlive());
    console.log(await dbClient.nbUsers());
    console.log(await dbClient.nbFiles());
})();

bob@dylan:~$ npm run dev main.js
false
true
4
30
bob@dylan:~$
```

Repo:

- GitHub repository: `alx-files_manager`
- File: `utils/db.js`

Done! Help Check your code Get a sandbox QA Review

2. First API

mandatory

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

(/)

Inside `server.js`, create the Express server:

- it should listen on the port set by the environment variable `PORT` or by default 5000
- it should load all routes from the file `routes/index.js`

Inside the folder `routes`, create a file `index.js` that contains all endpoints of our API:

- `GET /status => AppController.getStatus`
- `GET /stats => AppController.getStats`

Inside the folder `controllers`, create a file `AppController.js` that contains the definition of the 2 endpoints:

- `GET /status` should return if Redis is alive and if the DB is alive too by using the 2 utils created previously: `{ "redis": true, "db": true }` with a status code 200
- `GET /stats` should return the number of users and files in DB: `{ "users": 12, "files": 1231 }` with a status code 200
  - `users` collection must be used for counting all users
  - `files` collection must be used for counting all files

Terminal 1:

```
bob@dylan:~$ npm run start-server
Server running on port 5000
...
```

Terminal 2:

```
bob@dylan:~$ curl 0.0.0.0:5000/status ; echo ""
{"redis":true,"db":true}
bob@dylan:~$
bob@dylan:~$ curl 0.0.0.0:5000/stats ; echo ""
{"users":4,"files":30}
bob@dylan:~$
```

Repo:

- GitHub repository: `alx-files_manager`
- File: `server.js, routes/index.js, controllers/AppController.js`

Done! Help Check your code Get a sandbox QA Review

3. Create a new user

mandatory

Score: 16.67% (*Checks completed: 16.67%*)

Now that we have a simple API, it's time to add users to our database.
(/)
In the file `routes/index.js`, add a new endpoint:

- `POST /users => UsersController.postNew`

Inside `controllers`, add a file `UsersController.js` that contains the new endpoint:

`POST /users` should create a new user in DB:

- To create a user, you must specify an `email` and a `password`
- If the `email` is missing, return an error `Missing email` with a status code 400
- If the `password` is missing, return an error `Missing password` with a status code 400
- If the `email` already exists in DB, return an error `Already exist` with a status code 400
- The `password` must be stored after being hashed in `SHA1`
- The endpoint is returning the new user with only the `email` and the `id` (auto generated by MongoDB) with a status code 201
- The new user must be saved in the collection `users`:
  - `email`: same as the value received
  - `password`: `SHA1` value of the value received

```
bob@dylan:~$ curl 0.0.0.0:5000/users -XPOST -H "Content-Type: application/json" -d
'{ "email": "bob@dylan.com", "password": "toto1234!" }' ; echo ""
{"id":"5f1e7d35c7ba06511e683b21","email":"bob@dylan.com"}
bob@dylan:~$
bob@dylan:~$ echo 'db.users.find()' | mongo files_manager
{ "_id" : ObjectId("5f1e7d35c7ba06511e683b21"), "email" : "bob@dylan.com", "passwor
d" : "89cad29e3ebc1035b29b1478a8e70854f25fa2b2" }
bob@dylan:~$
bob@dylan:~$
bob@dylan:~$ curl 0.0.0.0:5000/users -XPOST -H "Content-Type: application/json" -d
'{ "email": "bob@dylan.com", "password": "toto1234!" }' ; echo ""
{"error":"Already exist"}
bob@dylan:~$
bob@dylan:~$ curl 0.0.0.0:5000/users -XPOST -H "Content-Type: application/json" -d
'{ "email": "bob@dylan.com" }' ; echo ""
{"error":"Missing password"}
bob@dylan:~$
```

Repo:

- GitHub repository: `alx-files_manager`
- File: `utils/, routes/index.js, controllers/UsersController.js`

☐ Done? Help Check your code Ask for a new correction >_ Get a sandbox QA Review

4. Authenticate a user

mandatory

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

(/)

In the file `routes/index.js` , add 3 new endpoints:

- `GET /connect => AuthController.getConnect`
- `GET /disconnect => AuthController.getDisconnect`
- `GET /users/me => UserController.getMe`

Inside `controllers` , add a file `AuthController.js` that contains new endpoints:

`GET /connect` should sign-in the user by generating a new authentication token:

- By using the header `Authorization` and the technique of the Basic auth (Base64 of the `<email>:<password>`), find the user associate to this email and with this password (reminder: we are storing the SHA1 of the password)
- If no user has been found, return an error `Unauthorized` with a status code 401
- Otherwise:
  - Generate a random string (using `uuidv4`) as token
  - Create a key: `auth_<token>`
  - Use this key for storing in Redis (by using the `redisClient` create previously) the user ID for 24 hours
  - Return this token: `{ "token": "155342df-2399-41da-9e8c-458b6ac52a0c" }` with a status code 200

Now, we have a way to identify a user, create a token (= avoid to store the password on any front-end) and use this token for 24h to access to the API!

Every authenticated endpoints of our API will look at this token inside the header `X-Token` .

`GET /disconnect` should sign-out the user based on the token:

- Retrieve the user based on the token:
  - If not found, return an error `Unauthorized` with a status code 401
  - Otherwise, delete the token in Redis and return nothing with a status code 204

Inside the file `controllers/UsersController.js` add a new endpoint:

`GET /users/me` should retrieve the user base on the token used:

- Retrieve the user based on the token:
  - If not found, return an error `Unauthorized` with a status code 401
  - Otherwise, return the user object (`email` and `id` only)

```
bob@dylan:~$ curl 0.0.0.0:5000/connect -H "Authorization: Basic Ym9iQGR5bGFuLmNvbTp0b3RvMTIzNCE=" ; echo ""
{"token":"031bffac-3edc-4e51-aaae-1c121317da8a"}
bob@dylan:~$
bob@dylan:~$ curl 0.0.0.0:5000/users/me -H "X-Token: 031bffac-3edc-4e51-aaae-1c121317da8a" ; echo ""
{"id":"5f1e7cda04a394508232559d","email":"bob@dylan.com"}
bob@dylan:~$
bob@dylan:~$ curl 0.0.0.0:5000/disconnect -H "X-Token: 031bffac-3edc-4e51-aaae-1c121317da8a" ; echo ""

bob@dylan:~$ curl 0.0.0.0:5000/users/me -H "X-Token: 031bffac-3edc-4e51-aaae-1c121317da8a" ; echo ""
{"error":"Unauthorized"}
bob@dylan:~$
```

Repo:

- GitHub repository: `alx-files_manager`
- File: `utils/, routes/index.js, controllers/UsersController.js, controllers/AuthController.js`

Done! Help Check your code Get a sandbox QA Review

5. First file

mandatory

Score: 9.09% (*Checks completed: 9.09%*)

In the file `routes/index.js`, add a new endpoint:

- `POST /files => FilesController.postUpload`

Inside `controllers`, add a file `FilesController.js` that contains the new endpoint:

`POST /files` should create a new file in DB and in disk:

- Retrieve the user based on the token:
  - If not found, return an error `Unauthorized` with a status code 401
- To create a file, you must specify:
  - `name`: as filename
  - `type`: either `folder`, `file` or `image`
  - `parentId`: (optional) as ID of the parent (default: 0 -> the root)
  - `isPublic`: (optional) as boolean to define if the file is public or not (default: false)
  - `data`: (only for `type=file|image`) as Base64 of the file content
- If the `name` is missing, return an error `Missing name` with a status code 400
- If the `type` is missing or not part of the list of accepted type, return an error `Missing type` with a status code 400

- If the `data` is missing and `type != folder`, return an error `Missing data` with a status code 400
- If the `parentId` is set:
  - If no file is present in DB for this `parentId`, return an error `Parent not found` with a status code 400
  - If the file present in DB for this `parentId` is not of type `folder`, return an error `Parent is not a folder` with a status code 400
- The user ID should be added to the document saved in DB - as owner of a file
- If the type is `folder`, add the new file document in the DB and return the new file with a status code 201
- Otherwise:
  - All file will be stored locally in a folder (to create automatically if not present):
    - The relative path of this folder is given by the environment variable `FOLDER_PATH`
    - If this variable is not present or empty, use `/tmp/files_manager` as storing folder path
  - Create a local path in the storing folder with filename a UUID
  - Store the file in clear (reminder: `data` contains the Base64 of the file) in this local path
  - Add the new file document in the collection `files` with these attributes:
    - `userId`: ID of the owner document (owner from the authentication)
    - `name`: same as the value received
    - `type`: same as the value received
    - `isPublic`: same as the value received
    - `parentId`: same as the value received - if not present: 0
    - `localPath`: for a `type=file|image`, the absolute path to the file save in local
  - Return the new file with a status code 201

```
bob@dylan:~$ curl 0.0.0.0:5000/connect -H "Authorization: Basic Ym9iQGR5bGFuLmNvbTp0
b3RvMTIzNCE=" ; echo ""
{"token":"f21fb953-16f9-46ed-8d9c-84c6450ec80f"}
bob@dylan:~$
bob@dylan:~$ curl -XPOST 0.0.0.0:5000/files -H "X-Token: f21fb953-16f9-46ed-8d9c-84c
6450ec80f" -H "Content-Type: application/json" -d '{ "name": "myText.txt", "type":
"file", "data": "SGVsbG8gV2Vic3RhY2shCg==" }' ; echo ""
{"id":"5f1e879ec7ba06511e683b22","userId":"5f1e7cda04a394508232559d","name":"myText.
txt","type":"file","isPublic":false,"parentId":0}
bob@dylan:~$
bob@dylan:~$ ls /tmp/files_manager/
2a1f4fc3-687b-491a-a3d2-5808a02942c9
bob@dylan:~$
bob@dylan:~$ cat /tmp/files_manager/2a1f4fc3-687b-491a-a3d2-5808a02942c9
Hello Webstack!
bob@dylan:~$
bob@dylan:~$ curl -XPOST 0.0.0.0:5000/files -H "X-Token: f21fb953-16f9-46ed-8d9c-84c
6450ec80f" -H "Content-Type: application/json" -d '{ "name": "images", "type": "fold
er" }' ; echo ""
{"id":"5f1e881cc7ba06511e683b23","userId":"5f1e7cda04a394508232559d","name":"image
s","type":"folder","isPublic":false,"parentId":0}
bob@dylan:~$
bob@dylan:~$ cat image_upload.py
import base64
import requests
import sys

file_path = sys.argv[1]
file_name = file_path.split('/')[-1]

file_encoded = None
with open(file_path, "rb") as image_file:
    file_encoded = base64.b64encode(image_file.read()).decode('utf-8')

r_json = { 'name': file_name, 'type': 'image', 'isPublic': True, 'data': file_encode
d, 'parentId': sys.argv[3] }
r_headers = { 'X-Token': sys.argv[2] }

r = requests.post("http://0.0.0.0:5000/files", json=r_json, headers=r_headers)
print(r.json())

bob@dylan:~$
bob@dylan:~$ python image_upload.py image.png f21fb953-16f9-46ed-8d9c-84c6450ec80f 5
f1e881cc7ba06511e683b23
{'id': '5f1e8896c7ba06511e683b25', 'userId': '5f1e7cda04a394508232559d', 'name': 'im
age.png', 'type': 'image', 'isPublic': True, 'parentId': '5f1e881cc7ba06511e683b23'}
bob@dylan:~$
bob@dylan:~$ echo 'db.files.find()' | mongo files_manager
{ "_id" : ObjectId("5f1e881cc7ba06511e683b23"), "userId" : ObjectId("5f1e7cda04a3945
08232559d"), "name" : "images", "type" : "folder", "parentId" : "0" }
{ "_id" : ObjectId("5f1e879ec7ba06511e683b22"), "userId" : ObjectId("5f1e7cda04a3945
08232559d"), "name" : "myText.txt", "type" : "file", "parentId" : "0", "isPublic" :
```

```
false, "localPath" : "/tmp/files_manager/2a1f4fc3-687b-491a-a3d2-5808a02942c9" }
{ "_id" : ObjectId("5f1e8896c7ba06511e683b25"), "userId" : ObjectId("5f1e7cda04a3945
08232559d"), "name" : "image.png", "type" : "image", "parentId" : ObjectId("5f1e881c
c7ba06511e683b23"), "isPublic" : true, "localPath" : "/tmp/files_manager/51997b88-5c
42-42c2-901e-e7f4e71bdc47" }
bob@dylan:~$
bob@dylan:~$ ls /tmp/files_manager/
2a1f4fc3-687b-491a-a3d2-5808a02942c9   51997b88-5c42-42c2-901e-e7f4e71bdc47
bob@dylan:~$
```

Repo:

- GitHub repository: `alx-files_manager`
- File: `utils/, routes/index.js, controllers/FilesController.js`

☐ Done? Help Check your code Ask for a new correction >_ Get a sandbox QA Review

6. Get and list file

mandatory

Score: 9.09% (*Checks completed: 9.09%*)

In the file `routes/index.js`, add 2 new endpoints:

- `GET /files/:id => FilesController.getShow`
- `GET /files => FilesController.getIndex`

In the file `controllers/FilesController.js`, add the 2 new endpoints:

`GET /files/:id` should retrieve the file document based on the ID:

- Retrieve the user based on the token:
  - If not found, return an error `Unauthorized` with a status code 401
- If no file document is linked to the user and the ID passed as parameter, return an error `Not found` with a status code 404
- Otherwise, return the file document

`GET /files` should retrieve all users file documents for a specific `parentId` and with pagination:

- Retrieve the user based on the token:
  - If not found, return an error `Unauthorized` with a status code 401
- Based on the query parameters `parentId` and `page`, return the list of file document
  - `parentId`:
    - No validation of `parentId` needed - if the `parentId` is not linked to any user folder, returns an empty list
    - By default, `parentId` is equal to 0 = the root
  - Pagination:
    - Each page should be 20 items max

(/)

- `page` query parameter starts at 0 for the first page. If equals to 1, it means it's the second page (form the 20th to the 40th), etc…
- Pagination can be done directly by the `aggregate` of MongoDB

```
bob@dylan:~$ curl 0.0.0.0:5000/connect -H "Authorization: Basic Ym9iQGR5bGFuLmNvbTp0
b3RvMTIzNCE=" ; echo ""
{"token":"f21fb953-16f9-46ed-8d9c-84c6450ec80f"}
bob@dylan:~$
bob@dylan:~$ curl -XGET 0.0.0.0:5000/files -H "X-Token: f21fb953-16f9-46ed-8d9c-84c6
450ec80f" ; echo ""
[{"id":"5f1e879ec7ba06511e683b22","userId":"5f1e7cda04a394508232559d","name":"myTex
t.txt","type":"file","isPublic":false,"parentId":0},{"id":"5f1e881cc7ba06511e683b2
3","userId":"5f1e7cda04a394508232559d","name":"images","type":"folder","isPublic":fa
lse,"parentId":0},{"id":"5f1e8896c7ba06511e683b25","userId":"5f1e7cda04a394508232559
d","name":"image.png","type":"image","isPublic":true,"parentId":"5f1e881cc7ba06511e6
83b23"}]
bob@dylan:~$
bob@dylan:~$ curl -XGET 0.0.0.0:5000/files?parentId=5f1e881cc7ba06511e683b23 -H "X-T
oken: f21fb953-16f9-46ed-8d9c-84c6450ec80f" ; echo ""
[{"id":"5f1e8896c7ba06511e683b25","userId":"5f1e7cda04a394508232559d","name":"image.
png","type":"image","isPublic":true,"parentId":"5f1e881cc7ba06511e683b23"}]
bob@dylan:~$
bob@dylan:~$ curl -XGET 0.0.0.0:5000/files/5f1e8896c7ba06511e683b25 -H "X-Token: f21
fb953-16f9-46ed-8d9c-84c6450ec80f" ; echo ""
{"id":"5f1e8896c7ba06511e683b25","userId":"5f1e7cda04a394508232559d","name":"image.p
ng","type":"image","isPublic":true,"parentId":"5f1e881cc7ba06511e683b23"}
bob@dylan:~$
```

Repo:

- GitHub repository: `alx-files_manager`
- File: `utils/, routes/index.js, controllers/FilesController.js`

☐ Done? Help Check your code Ask for a new correction Get a sandbox QA Review

7. File publish/unpublish

mandatory

Score: 9.09% (*Checks completed: 9.09%*)

In the file `routes/index.js`, add 2 new endpoints:

- `PUT /files/:id/publish => FilesController.putPublish`
- `PUT /files/:id/publish => FilesController.putUnpublish`

In the file `controllers/FilesController.js`, add the 2 new endpoints:

`PUT /files/:id/publish` should set `isPublic` to `true` on the file document based on the ID:

- Retrieve the user based on the token:

- If not found, return an error `Unauthorized` with a status code 401
- If no file document is linked to the user and the ID passed as parameter, return an error `Not found` with a status code 404
- Otherwise:
  - Update the value of `isPublic` to `true`
  - And return the file document with a status code 200

`PUT /files/:id/unpublish` should set `isPublic` to `false` on the file document based on the ID:

- Retrieve the user based on the token:
  - If not found, return an error `Unauthorized` with a status code 401
- If no file document is linked to the user and the ID passed as parameter, return an error `Not found` with a status code 404
- Otherwise:
  - Update the value of `isPublic` to `false`
  - And return the file document with a status code 200

```
bob@dylan:~$ curl 0.0.0.0:5000/connect -H "Authorization: Basic Ym9iQGR5bGFuLmNvbTp0
b3RvMTIzNCE=" ; echo ""
{"token":"f21fb953-16f9-46ed-8d9c-84c6450ec80f"}
bob@dylan:~$
bob@dylan:~$ curl -XGET 0.0.0.0:5000/files/5f1e8896c7ba06511e683b25 -H "X-Token: f21
fb953-16f9-46ed-8d9c-84c6450ec80f" ; echo ""
{"id":"5f1e8896c7ba06511e683b25","userId":"5f1e7cda04a394508232559d","name":"image.p
ng","type":"image","isPublic":false,"parentId":"5f1e881cc7ba06511e683b23"}
bob@dylan:~$
bob@dylan:~$ curl -XPUT 0.0.0.0:5000/files/5f1e8896c7ba06511e683b25/publish -H "X-To
ken: f21fb953-16f9-46ed-8d9c-84c6450ec80f" ; echo ""
{"id":"5f1e8896c7ba06511e683b25","userId":"5f1e7cda04a394508232559d","name":"image.p
ng","type":"image","isPublic":true,"parentId":"5f1e881cc7ba06511e683b23"}
bob@dylan:~$
bob@dylan:~$ curl -XPUT 0.0.0.0:5000/files/5f1e8896c7ba06511e683b25/unpublish -H "X-
Token: f21fb953-16f9-46ed-8d9c-84c6450ec80f" ; echo ""
{"id":"5f1e8896c7ba06511e683b25","userId":"5f1e7cda04a394508232559d","name":"image.p
ng","type":"image","isPublic":false,"parentId":"5f1e881cc7ba06511e683b23"}
bob@dylan:~$
```

Repo:

- GitHub repository: `alx-files_manager`
- File: `utils/, routes/index.js, controllers/FilesController.js`

☐ Done? Help Check your code Ask for a new correction >_ Get a sandbox QA Review

8. File data

mandatory

Score: 7.14% (*Checks completed: 7.14%*)

In the file `routes/index.js` , add one new endpoint:
(/)

- `GET /files/:id/data => FilesController.getFile`

In the file `controllers/FilesController.js` , add the new endpoint:

`GET /files/:id/data` should return the content of the file document based on the ID:

- If no file document is linked to the ID passed as parameter, return an error `Not found` with a status code 404
- If the file document (folder or file) is not public (`isPublic: false`) and no user authenticate or not the owner of the file, return an error `Not found` with a status code 404
- If the type of the file document is `folder` , return an error `A folder doesn't have content` with a status code 400
- If the file is not locally present, return an error `Not found` with a status code 404
- Otherwise:
  - By using the module `mime-types` , get the MIME-type (/rltoken/buV7HGNuNMB5ZCUH0LdECw) based on the `name` of the file
  - Return the content of the file with the correct MIME-type

```
bob@dylan:~$ curl 0.0.0.0:5000/connect -H "Authorization: Basic Ym9iQGR5bGFuLmNvbTp0
b3RvMTIzNCE=" ; echo ""
{"token":"f21fb953-16f9-46ed-8d9c-84c6450ec80f"}
bob@dylan:~$
bob@dylan:~$ curl -XPUT 0.0.0.0:5000/files/5f1e879ec7ba06511e683b22/unpublish -H "X-
Token: f21fb953-16f9-46ed-8d9c-84c6450ec80f" ; echo ""
{"id":"5f1e879ec7ba06511e683b22","userId":"5f1e7cda04a394508232559d","name":"myText.
txt","type":"file","isPublic":false,"parentId":0}
bob@dylan:~$
bob@dylan:~$ curl -XGET 0.0.0.0:5000/files/5f1e879ec7ba06511e683b22/data -H "X-Toke
n: f21fb953-16f9-46ed-8d9c-84c6450ec80f" ; echo ""
Hello Webstack!

bob@dylan:~$ curl -XGET 0.0.0.0:5000/files/5f1e879ec7ba06511e683b22/data ; echo ""
{"error":"Not found"}
bob@dylan:~$
bob@dylan:~$ curl -XPUT 0.0.0.0:5000/files/5f1e879ec7ba06511e683b22/publish -H "X-To
ken: f21fb953-16f9-46ed-8d9c-84c6450ec80f" ; echo ""
{"id":"5f1e879ec7ba06511e683b22","userId":"5f1e7cda04a394508232559d","name":"myText.
txt","type":"file","isPublic":true,"parentId":0}
bob@dylan:~$
bob@dylan:~$ curl -XGET 0.0.0.0:5000/files/5f1e879ec7ba06511e683b22/data ; echo ""
Hello Webstack!

bob@dylan:~$
```

Repo:

- GitHub repository: `alx-files_manager`

- File: `utils/, routes/index.js, controllers/FilesController.js`

(/)

Done? Help Check your code Ask for a new correction QA Review

9. Image Thumbnails

mandatory

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

Update the endpoint `POST /files` endpoint to start a background processing for generating thumbnails for a file of type `image`:

- Create a `Bull` queue `fileQueue`
- When a new image is stored (in local and in DB), add a job to this queue with the `userId` and `fileId`

Create a file `worker.js`:

- By using the module `Bull`, create a queue `fileQueue`
- Process this queue:
  - If `fileId` is not present in the job, raise an error `Missing fileId`
  - If `userId` is not present in the job, raise an error `Missing userId`
  - If no document is found in DB based on the `fileId` and `userId`, raise an error `File not found`
  - By using the module `image-thumbnail`, generate 3 thumbnails with `width` = 500, 250 and 100 - store each result on the same location of the original file by appending `_<width size>`

Update the endpoint `GET /files/:id/data` to accept a query parameter `size`:

- `size` can be `500`, `250` or `100`
- Based on `size`, return the correct local file
- If the local file doesn't exist, return an error `Not found` with a status code 404

Terminal 3: (start the worker)

```
bob@dylan:~$ npm run start-worker
...
```

Terminal 2:

```
bob@dylan:~$ curl 0.0.0.0:5000/connect -H "Authorization: Basic Ym9iQGR5bGFuLmNvbTp0
b3RvMTIzNCE=" ; echo ""
{"token":"f21fb953-16f9-46ed-8d9c-84c6450ec80f"}
bob@dylan:~$
bob@dylan:~$ python image_upload.py image.png f21fb953-16f9-46ed-8d9c-84c6450ec80f 5
f1e881cc7ba06511e683b23
{'id': '5f1e8896c7ba06511e683b25', 'userId': '5f1e7cda04a394508232559d', 'name': 'im
age.png', 'type': 'image', 'isPublic': True, 'parentId': '5f1e881cc7ba06511e683b23'}
bob@dylan:~$ ls /tmp/files_manager/
2a1f4fc3-687b-491a-a3d2-5808a02942c9   51997b88-5c42-42c2-901e-e7f4e71bdc47   6dc533
97-8491-4b7c-8273-f748b1a031cb   6dc53397-8491-4b7c-8273-f748b1a031cb_100   6dc53397
-8491-4b7c-8273-f748b1a031cb_250    6dc53397-8491-4b7c-8273-f748b1a031cb_500
bob@dylan:~$
bob@dylan:~$ curl -XGET 0.0.0.0:5000/files/5f1e8896c7ba06511e683b25/data -so new_ima
ge.png ; file new_image.png
new_image.png: PNG image data, 471 x 512, 8-bit/color RGBA, non-interlaced
bob@dylan:~$
bob@dylan:~$ curl -XGET 0.0.0.0:5000/files/5f1e8896c7ba06511e683b25/data?size=100 -s
o new_image.png ; file new_image.png
new_image.png: PNG image data, 100 x 109, 8-bit/color RGBA, non-interlaced
bob@dylan:~$
bob@dylan:~$ curl -XGET 0.0.0.0:5000/files/5f1e8896c7ba06511e683b25/data?size=250 -s
o new_image.png ; file new_image.png
new_image.png: PNG image data, 250 x 272, 8-bit/color RGBA, non-interlaced
bob@dylan:~$
```

Repo:

- GitHub repository: `alx-files_manager`
- File: `utils/, controllers/FilesController.js, worker.js`

Done! Help QA Review

10. Tests!

#advanced

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

Of course, a strong and stable project can not be good without tests.

Create tests for `redisClient` and `dbClient`.

Create tests for each endpoints:

- `GET /status`
- `GET /stats`
- `POST /users`
- `GET /connect`
- `GET /disconnect`

(/)

- `GET /users/me`
- `POST /files`
- `GET /files/:id`
- `GET /files` (don’t forget the pagination)
- `PUT /files/:id/publish`
- `PUT /files/:id/unpublish`
- `GET /files/:id/data`

Repo:

- GitHub repository: `alx-files_manager`
- File: `tests/`

Done! Help QA Review

11. New user - welcome email

#advanced

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

Update the endpoint `POST /users` endpoint to start a background processing for sending a “Welcome email” to the user:

- Create a `Bull` queue `userQueue`
- When a new user is stored (in DB), add a job to this queue with the `userId`

Update the file `worker.js`:

- By using the module `Bull`, create a queue `userQueue`
- Process this queue:
  - If `userId` is not present in the job, raise an error `Missing userId`
  - If no document is found in DB based on the `userId`, raise an error `User not found`
  - Print in the console `Welcome <email>!`

In real life, you can use a third party service like Mailgun (/rltoken/Lr8MoeTgC7KRx413wpoBXg) to send real email. These API are slow, (sending via SMTP is worst!) and sending emails via a background job is important to optimize API endpoint.

Repo:

- GitHub repository: `alx-files_manager`
- File: `utils/, worker.js, controllers/UsersController.js`

Done! Help QA Review

(/)

Ready for a new manual review