

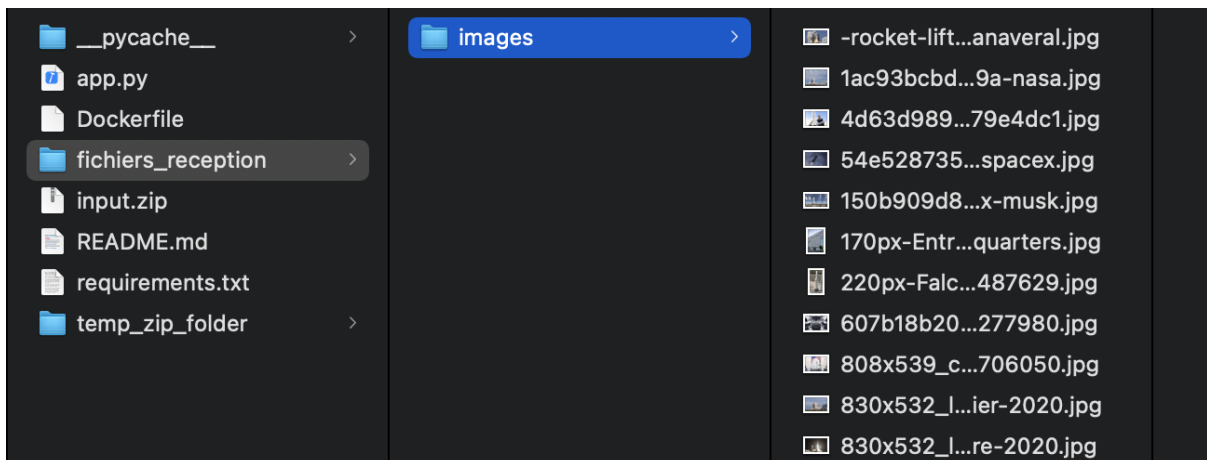
TP2 WebServers - Noah Chatelain & Ambroise Gyre

Création d'une route permettant d'envoyer un fichier .zip au serveur, le serveur le décompresse, puis le place dans le dossier *fichiers_reception*

```
root@noah:/app# python app.py 7.0.0.1:5000/extract
* Serving Flask app 'app' (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on all addresses.
  WARNING: This is a development server. Do not use it in a production deployment.
* Running on http://172.17.0.4:5000/ (Press CTRL+C to quit)
```

```
noahchatelain@MBP-de-Noah app % curl -X POST -i -F "file=@/Users/noahchatelain/Desktop/app/input.zip" 127.0.0.1:5000/extract
HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
Content-Length: 3
Server: Werkzeug/2.0.1 Python/3.8.12
Date: Tue, 05 Oct 2021 19:44:24 GMT

OK
noahchatelain@MBP-de-Noah app %
```



2ème étape, obtenir le code SHA-1 de nos photos qui seront à l'intérieur de l'archive *input.zip* à envoyer à notre serveur

```
noahchatelain@MBP-de-Noah app % shasum input/photo*
e4bc0da9501b4b23f5d5e8eaf8f0c69b105ce4db input/photo1.jpg
9d528c080e4b06af37ba8446ff69bdead9c08e6a input/photo2.jpg
e5c8184ba8f46b5c4e11fca33cf81fb1b2f5c4b7 input/photo3.jpg
noahchatelain@MBP-de-Noah app %
```

Maintenant que nous connaissons nos code SHA-1, nous pouvons vérifier en python que nous recevons bien les bonnes images

```
root@noah:/app# python app.py
* Serving Flask app 'app' (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on all addresses.
  WARNING: This is a development server. Do not use it in a production deployment.
* Running on http://172.17.0.4:5000/ (Press CTRL+C to quit)
fichiers_reception/input/.DS_Store
808c17324267fac9ab78f5f2b2fb83372ab694ef

fichiers_reception/input/photo1.jpg
e4bc0da9501b4b23f5d5e9eaf8f0c69b105ce4db

fichiers_reception/input/photo2.jpg
9d528c080e4b06af37ba8446ff69bdead9c08e6a

fichiers_reception/input/photo3.jpg
e5c8184ba8f46b5c4e11fca33cf81fb1b2f5c4b7

172.17.0.1 - - [05/Oct/2021 20:13:38] "POST /extract HTTP/1.1" 200 -
```

Nous voyons bien ici que nos codes SHA-1 sont identiques à l'étape précédente

`curl -X POST -i -F`

`json="{\"photo1\": \"e4bc0da9501b4b23f5d5e9eaf8f0c69b105ce4db\", \"photo2\": \"9d528c080e4b06af37ba8446ff69bdead9c08e6a\", \"photo3\": \"e5c8184ba8f46b5c4e11fca33cf81fb1b2f5c4b7\"}" -F "file=@/Users/noahchatelain/Desktop/app/input.zip" 127.0.0.1:5000/extract`

```
root@noah:/app# python app.py
* Serving Flask app 'app' (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on all addresses.
  WARNING: This is a development server. Do not use it in a production deployment.
* Running on http://172.17.0.4:5000/ (Press CTRL+C to quit)
{"photo1": "e4bc0da9501b4b23f5d5e9eaf8f0c69b105ce4db", "photo2": "9d528c080e4b06af37ba8446ff69bdead9c08e6a", "photo3": "e5c8184ba8f46b5c4e11fca33cf81fb1b2f5c4b7", "photo4": "4"}
photo1 fichiers_reception/input/photo1.jpg e4bc0da9501b4b23f5d5e9eaf8f0c69b105ce4db
SHA-1 Code is OK
photo2 fichiers_reception/input/photo2.jpg 9d528c080e4b06af37ba8446ff69bdead9c08e6a
SHA-1 Code is OK
photo3 fichiers_reception/input/photo3.jpg e5c8184ba8f46b5c4e11fca33cf81fb1b2f5c4b7
SHA-1 Code is OK
172.17.0.1 - - [05/Oct/2021 21:02:11] "POST /extract HTTP/1.1" 200 -
```

Les vérifications sur les checksums fonctionnent bien. Lorsque l'on teste avec une erreur de code SHA-1 dans le JSON, nous sommes bien avertis :

```

app — -zsh — 129x24
~/Desktop/app — -zsh
noahchatelain@MBP-de-Noah app % curl -X POST -i -F json="{\"photo1\": \"e4bc0da9501b4b23f5d5e9eaf8f0c69b105ce4db\", \"photo2\": \"9d528c080e4b06af37ba8446ff69bdead9c08e6a\", \"photo3\": \"SHA1CODE-FALSE\"}" -F \"file=@/Users/noahchatelain/Desktop/app/input.zip\" 127.0.0.1:5000/extract
HTTP/1.1 100 Continue

photo3 fichiers_reception/input/photo3.jpg e5
SHA-1 Code is OK
172.17.0.1 - - [05/Oct/2021 21:05:18] \"POST /
{\"photo1\": \"e4bc0da9501b4b23f5d5e9eaf8f0c69b10
E-FALSE\"}
photo1 fichiers_reception/input/photo1.jpg e4
SHA-1 Code is OK
photo2 fichiers_reception/input/photo2.jpg 9d
SHA-1 Code is OK
photo3 fichiers_reception/input/photo3.jpg e5
SHA-1 Code is not OK !
172.17.0.1 - - [05/Oct/2021 21:05:43] \"POST /

```

```

app — root@0ae5e0a0f096: / — com.docker.cli • docker exec -it tp2webserver bash — 122x24
~/Desktop/app — root@0ae5e0a0f096: / — com.docker.cli • docker exec -it tp2webserver bash
root@noah:/app# python app.py
* Serving Flask app 'app' (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on all addresses.
  WARNING: This is a development server. Do not use it in a production deployment.
* Running on http://172.17.0.4:5000/ (Press CTRL+C to quit)
{"photo1": "e4bc0da9501b4b23f5d5e9eaf8f0c69b105ce4db", "photo2": "9d528c080e4b06af37ba8446ff69bdead9c08e6a", "photo3": "SHA1CODE-FALSE"}
photo1 fichiers_reception/input/photo1.jpg e4bc0da9501b4b23f5d5e9eaf8f0c69b105ce4db
SHA-1 Code is OK
photo2 fichiers_reception/input/photo2.jpg 9d528c080e4b06af37ba8446ff69bdead9c08e6a
SHA-1 Code is OK
photo3 fichiers_reception/input/photo3.jpg e5c8184ba8f46b5c4e11fca33cf81fb1b2f5c4b7
SHA-1 Code is not OK !
172.17.0.1 - - [05/Oct/2021 21:06:31] "POST /extract HTTP/1.1" 200 -

```

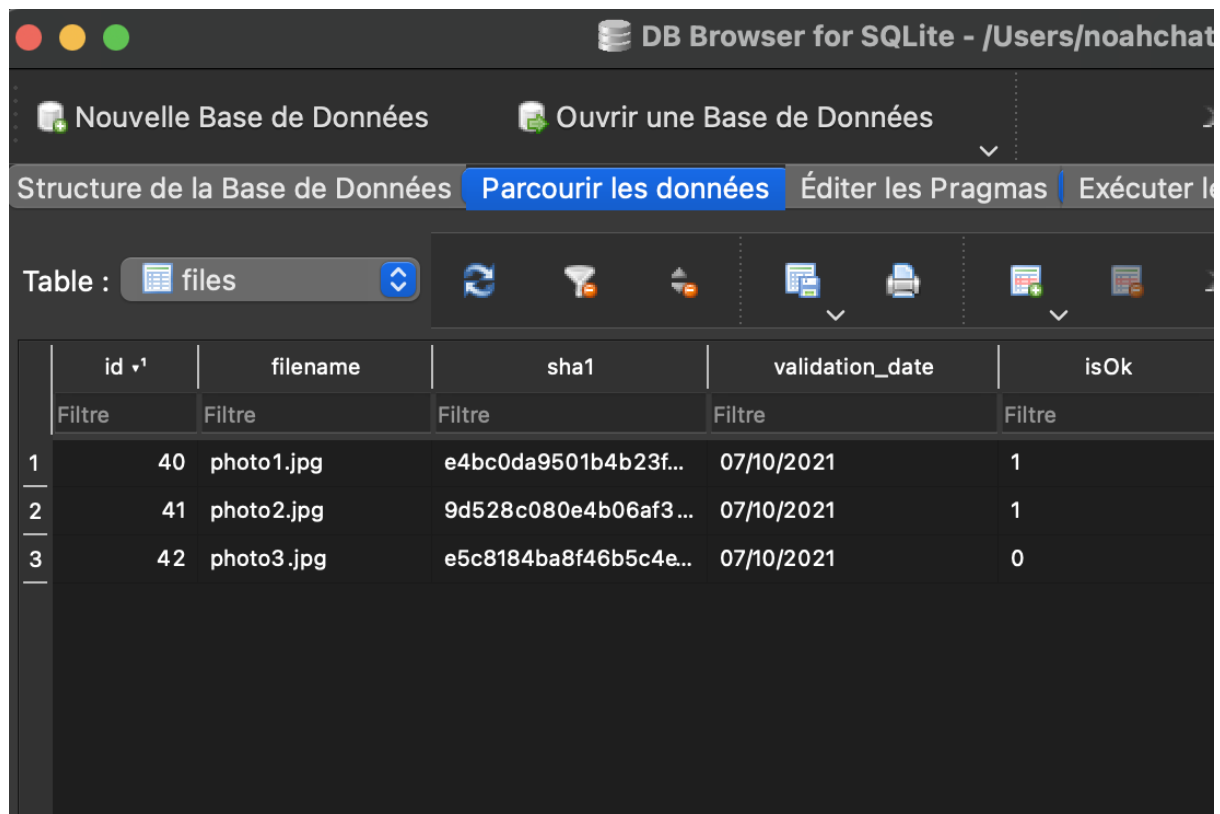
Lorsqu'un fichier possède le bon code SHA-1, alors il est ajouté au dossier `/fichiers_reception`, mais lorsqu'il n'est pas bon, il est supprimé du dossier `/fichiers_reception` et il est ajouté au fichier zip `/log_error_sha1_code_files.zip`

Essayer l'erreur avec cette commande :

`curl -X POST -i -F`

`json="{\"photo1\": \"e4bc0da9501b4b23f5d5e9eaf8f0c69b105ce4db\", \"photo2\": \"9d528c080e4b06af37ba8446ff69bdead9c08e6a\", \"photo3\": \"SHA1CODE-FALSE\"}" -F \"file=@/Users/noahchatelain/Desktop/app/input.zip\" 127.0.0.1:5000/extract`

Ajout d'une base de donnée où l'on retrouve toutes les photos, que leur code SHA-1 soit bon ou mauvais. Avec le nom, le code SHA-1, la date de validation, et s'il est valide ou non.



Ajout de la fonctionnalité CSV. Permet d'ajouter nos fichiers à une feuille CSV, exactement comme la base de données.

`curl -X POST -i -F`

`json="{\"photo1\": \"e4bc0da9501b4b23f5d5ebeaf8f0c69b105ce4db\", \"photo2\": \"9d528c080e4b06af37ba8446ff69bdead9c08e6a\", \"photo3\": \"SHA1CODE-FALSE\"}" -F`
`"file=@/Users/noahchatelain/Desktop/app/input.zip" 127.0.0.1:5000/extract`

files

photo1.jpg	e4bc0da9501b4b23f5d5ebeaf8f0c69b105ce4db	07/10/2021	True
photo2.jpg	9d528c080e4b06af37ba8446ff69bdead9c08e6a	07/10/2021	True
photo3.jpg	e5c8184ba8f46b5c4e11fca33cf81fb1b2f5c4b7	07/10/2021	False

Extrait de la feuille CSV contenant les informations des fichiers

Ajout d'une route permettant de télécharger ce fichier CSV directement pour le client.

<http://127.0.0.1:5000/download>

