

---

# Projet du cours Techniques web

---

Création d'une  
application Back-End

---

Chinatsu Kuroiwa, Solveig  
Poder et Camille Rey

---

# Table des matières

<b>PRE-REQUIS ET LANCEMENT DE L'APPLICATION .....</b>	<b>1</b>
Lancement en local .....	2
Lancement sur Heroku .....	3
<b>REQUÊTES SUR POSTMAN .....</b>	<b>4</b>
Authentication .....	4
Login (-POST-) .....	4
Logout (-GET-) .....	9
Données .....	7
Afficher les données (-GET-) .....	7
Ajouter des données (-PUT-).....	9
Modifier des données (-POST-).....	11
Supprimer des données (-DELETE-) .....	13
<b>DOCUMENTATION DE L'API .....</b>	<b>14</b>

# Pré-requis et Lancement de l'application

---

## Lancement en local

Le lancement en local requiert d'avoir un système Linux (ex: distribution Ubuntu) ou Mac, et d'avoir installé :

- Nginx (<https://www.nginx.com>)
- Unicorn (<https://gunicorn.org/>)

1 - A partir de la racine du répertoire projet, éditez le fichier **api\_nginx.conf** : remplacez les deux occurrences de CHEMIN\_DU\_REPERTOIRE\_PROJET par le chemin absolu de la racine du répertoire projet, et enregistrez le fichier.

\* **Optionnel** : on peut retirer le # devant la ligne « #return 301 https://\$host\$request\_uri; ». Cela aura pour effet de rediriger automatiquement le serveur http vers le serveur https qui utilise un certificat ssl.

2 - Copiez le fichier **api\_nginx.conf** dans le répertoire **sites-available** de nginx

Linux : **/etc/nginx/sites-available**

Mac: **/usr/local/etc/nginx/sites-available** (si le dossier sites-available n'existe pas, créez-le et rajoutez la ligne « include /usr/local/etc/nginx/sites-enabled/\*.conf; » dans le bloc http du fichier /usr/local/etc/nginx/nginx.conf )

3 - Créez le lien symbolique avec le dossier **sites-enabled**

Linux: **ln -s /etc/nginx/sites-available/api\_nginx.conf /etc/nginx/sites-enabled/api\_nginx.conf**

Mac: **ln -s /usr/local/etc/nginx/sites-available/api\_nginx.conf /usr/local/etc/nginx/sites-enabled/api\_nginx.conf** (si le dossier sites-enabled n'existe pas, créez-le)

4 -Vérifiez la bonne écriture des fichiers de configuration de nginx :

**sudo nginx -t**

Rechargez nginx avec les nouvelles configurations :

**sudo nginx -s reload**

Lancez Nginx :

Linux : **sudo service nginx restart**

Mac : **brew services restart nginx**

5 - Installez les dépendances : depuis le répertoire du projet, créez un environnement virtuel (avec pipenv <https://pypi.org/project/pipenv/>), connectez-y vous, puis installez les dépendances depuis le fichier requirements.txt :

```
pip3 -r install requirements.txt
```

6 - Lancez l'application avec gunicorn :

```
sh launcher_gunicorn.sh
```

Sauf erreur dans le processus, le serveur devrait désormais être accessible en requêtes à partir de l'URL <http://digidata.api.localhost/> ou bien <https://digidata.api.localhost/> (il faudra autoriser les certificats self-signed dans les paramètres du client web pour le https)

Si la configuration de nginx ne fonctionne pas, le serveur devrait être accessible à l'adresse <http://127.0.0.1:8000>

Si même gunicorn ne fonctionne pas, le serveur devrait être accessible à l'adresse <http://127.0.0.1:5000> après exécution de la commande :

```
python run.py
```

## **Lancement sur Heroku**

L'application est déjà déployée sur Heroku, à l'adresse <https://projet-tecweb-backend.herokuapp.com> . On peut donc effectuer les requêtes sur à partir de cette URL à la place de http(s)://api.digidata.api.localhost .

(Les fichiers runtime.txt et Procfile sont des fichiers nécessaires à Heroku pour le déploiement. Ils ne sont pas nécessaires pour lancer en local)

# Requêtes sur Postman

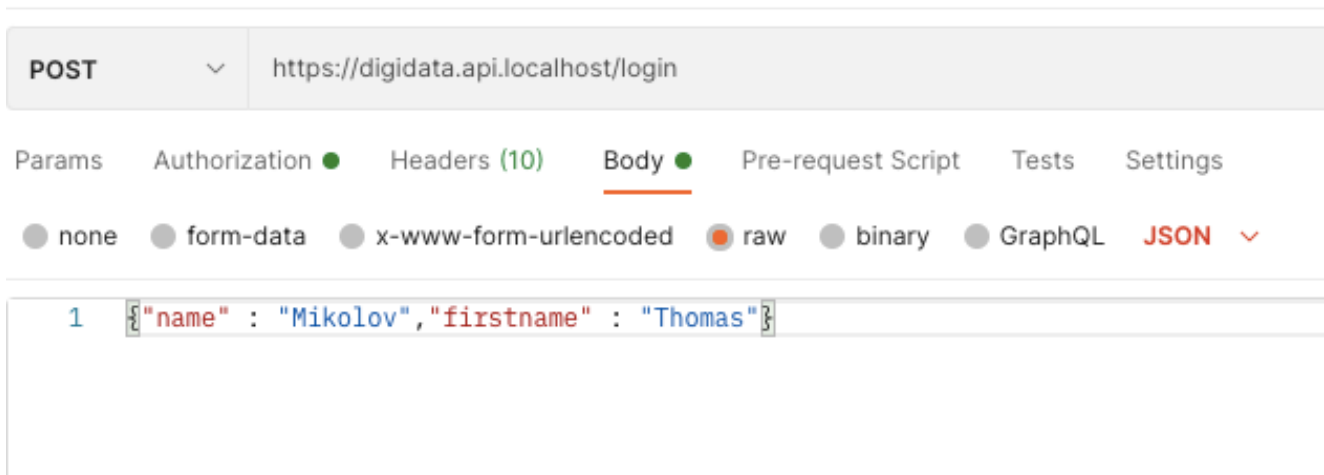
Toutes les requêtes présentées ici sont faites à partir du client web Postman (<https://www.postman.com/>), nous recommandons de l'installer pour pouvoir facilement effectuer les mêmes requêtes.

## Authentification

### Login (-POST-)

Sur Postman, créer une requête et entrer l'adresse <http://digidata.api.localhost/> (ou https)

Avant de pouvoir consulter et/ou manipuler la base de données, l'identité de l'utilisateur doit être vérifiée. Pour cela, une méthode POST a été définie pour la ressource **Login**, accessible aux URL <http://digidata.api.localhost/> ou <http://digidata.api.localhost/login>. Il faut fournir son nom et son prénom en format json (dans le Body) de cette façon pour s'authentifier :



Après envoi de la requête, si les données rentrées dans le body ne sont pas au bon format, ou que les identifiants ne sont pas corrects, différents messages et codes d'erreur sont renvoyés. Sinon, les informations de l'utilisateur sont renvoyées et un token temporaire (60 mins) est fourni :

```

1 {
2   ... "Token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpZCI6IjAwMSIsImV4cCI6MTYxNDUwODU1Mn0.
   ... d9r3tLOU6U1rkCBthbYV1pJbS39ql7OB8HuwEUa3Q9Q",
3   ... "User": {
4     ... "actif": false,
5     ... "actionnaire": true,
6     ... "anciennete": 10,
7     ... "conge": 15,
8     ... "fonction": "Directeur des representations vectorielles",
9     ... "id": "001",
10    ... "mise_a_jour": "2017-05-06 11:25:11.827000",
11    ... "missions": [
12      ... "Bruxelle",
13      ... "Paris",
14      ... "Pakistan"
15    ],
16    ... "nom": "Mikolov",
17    ... "prenom": "Thomas"
18  },
19  ... "status": "success"
20 }

```

Pour continuer, il vous suffit de copier-coller ce token en tant que *bearer token* dans l'onglet *Authorization* de la requête:

Params Auth ● Headers (11) Body ● Pre-req. Tests Settings

**Type**

Bearer Token ▼

The authorization header will be automatically generated when you send the request.

[Learn more about authorization](#)

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. [Learn more about variables](#)

**Token**

```
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpZCI6IjAwMSIsImV4cCI6MTYxNDUwODU1Mn0.d9r3tLOU6U1rkCBthbYV1pJbS39ql7OB8HuwEUa3Q9Q
```

Notons que si le client web est déjà « authentifié », c'est-à-dire si un token valide se trouve dans le header *Authorization*, ce token est renvoyé plutôt que de générer un nouveau token. Essayons à nouveau de s'identifier :

```
Body ▼ 200 OK 46 ms 353 B
Pretty Raw Preview Visualize JSON ▼
1 {
2   ... "Token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpZCI6IjAwMSIsImV4cCI6MTYxNDUwODU1Mn0.d9r3tLOU6U1rkCBthbYV1pJbS39q170B8HuwEUa3Q9Q",
3   ... "User_id": "001",
4   ... "message": "already logged in",
5   ... "status": "success"
6 }
```

## Logout (-GET-)

Une requête GET sur la ressource <http://digidata.api.localhost/logout> permet de se déconnecter. Cette requête ne peut se faire que si on est déjà authentifié (si un token valide est présent dans le header Authorization). Dans ce cas, le token est invalidé, et il faudra donc se reconnecter (cf requête précédente) pour obtenir un nouveau token.

```
GET ▼ https://digidata.api.localhost/logout Send ▼
Params Auth ● Headers (9) Body Pre-req. Tests Settings
Body ▼ 200 OK 41 ms 210 B Save Response ▼
Pretty Raw Preview Visualize JSON ▼
1 {
2   ... "message": "successfully logged out",
3   ... "status": "succes"
4 }
```

Si on essaie de se déconnecter une deuxième fois sans s'être ré-authentifié, on voit bien que le token n'est plus valide :


```
GET ▼ https://digidata.api.localhost/logout Send ▼
Params Auth ● Headers (9) Body Pre-req. Tests Settings
Body ▼ 401 UNAUTHORIZED 14 ms 208 B Save Response ▼
Pretty Raw Preview Visualize JSON ▼
1 {
2   ... "message": "expired token",
3   ... "status": "fail"
4 }
```

## Données

N'oubliez pas que toutes les requêtes suivantes sont réalisables **uniquement si** vous vous êtes identifié (si un token valide est présent dans le Authorization Header)


### Afficher les données (-GET-)

Pour afficher les données, il faut utiliser la méthode GET sur la ressource à l'URL <http://digidata.api.localhost/data> qui renverra la liste de tous les lieux:


GET 



https://digidata.api.localhost/data


On peut, si l'on veut, n'afficher qu'un seul lieu, en filtrant sur son attribut *geonameid* :



GET 




https://digidata.api.localhost/data/2659086

Send 

Params Auth  Headers (9) Body Pre-req. Tests Settings 

Body 

 200 OK 127 ms 587 B Save Response 

Pretty Raw Preview Visualize JSON   

```
1 {
2   "data": {
3     "admin1_code": "84",
4     "admin2_code": "74",
5     "admin3_code": "744",
6     "admin4_code": "74058",
7     "alternatenames": "Rapenaz · Col de, Recon · Col de",
8     "asciiname": "Col de Recon",
9     "cc2": "CH",
10    "country_code": "FR",
11    "dem": "1733",
12    "elevation": "",
13    "feature_class": "T",
14    "feature_code": "PASS",
15    "geonameid": "2659086",
16    "latitude": "46.30352",
17    "longitude": "6.82838",
18    "modification_date": "2019-02-15",
19    "name": "Col de Recon",
20    "population": "0",
21    "timezone": "Europe/Paris"
22  }
23 }
```

Un message d'erreur s'affichera si le *geonameid* n'est pas trouvé dans la base de données.



Afin d'effectuer des recherches plus complexes, une ressource <http://digidata.api.localhost/data/search> a été créée avec sa propre méthode GET. Il est possible de filtrer sur plusieurs attributs en ajoutant des paramètres à la requête. Par exemple, si l'on souhaite afficher tous les lieux dont le *name* est « Lucelle » et qui contiennent « Lützel » parmi leurs *alternatenames*:

GET
https://digidata.api.localhost/data/search?alternatename=Lützel&name=Lucelle
Send

Params
Auth
Headers (9)
Body
Pre-req.
Tests
Settings
Cookies

Query Params

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	alternatename	Lützel			
<input checked="" type="checkbox"/>	name	Lucelle			

La liste des résultats sera renvoyées (liste vide si aucun match) :

Body
200 OK
152 ms
1.06 KB
Save Respor

Pretty
Raw
Preview
Visualize
JSON

```

1  {
2    "results": [
3      {
4        "admin1_code": "00",
5        "admin2_code": "",
6        "admin3_code": "",
7        "admin4_code": "",
8        "alternatenames": "La Lucelle Riviere,La Lucelle Rivière,Lucelle,Lutzel,
          Lützel",
9        "asciiname": "Lucelle",
10       "cc2": "",
11       "country_code": "FR",
12       "dem": "353",
13       "elevation": "",
14       "feature_class": "H",
15       "feature_code": "STM",
16       "geonameid": "2659815",
17       "latitude": "47.41667",
18       "longitude": "7.5",
19       "modification_date": "2014-08-05",
20       "name": "Lucelle",

```

## Ajouter des données (-PUT-)

Pour ajouter une donnée dans la base, nous avons défini une méthode PUT. La donnée à ajouter doit être communiquée au format json dans le body. Le *geonameid* doit être obligatoirement passé dans l'adresse, son absence occasionnant une erreur 400. S'il existe déjà, cela créera une erreur également. Le seul champ obligatoire dans le body est le champ « name », s'il manque les autres champs, ils seront vides dans la base de données mais cela ne générera pas d'erreur. En revanche, si des champs ne correspondant pas au format de notre base de données sont renseignés, (par exemple « ascii » au lieu de « asciiname ») un message d'erreur sera renvoyé. Voici un exemple de requête correcte :

The screenshot shows a REST client interface with the following details:

- Method:** PUT
- URL:** `https://digidata.api.localhost/data/7777777`
- Body (JSON):**

```
{  "name": "Tèst", "asciiname": "Test", "country_code": "FR",  "timezone": "Europe/Paris"}
```
- Status:** 200 OK, 50 ms, 223 B
- Response Body (JSON):**

```
{  "message": "new data 7777777 successfully added",  "status": "success"}
```

Quelques exemples de requête incorrectes :

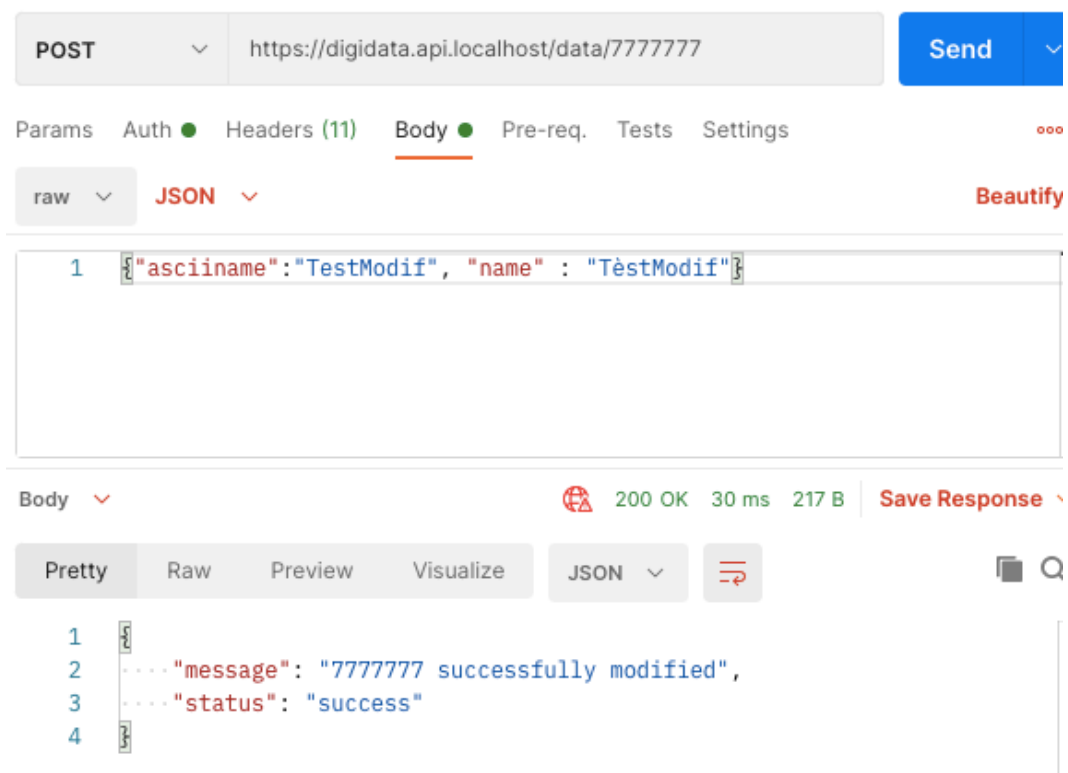
- Mauvais champ :



## Modifier des données (-POST-)

Il est également possible de modifier des données avec la méthode POST.

Encore une fois, *geonameid* doit être obligatoirement passé dans la requête. S'il ne correspond à aucun lieu dans la base de données, cela génèrera une erreur. Les attributs à modifier et leur nouvelle valeur sont renseignés au format JSON dans le body de la requête. Si les champs ne correspondent pas au format de la BDD, un message d'erreur est renvoyé (comme pour la requête PUT). Le *geonameid* est le seul champ qui n'est pas autorisé à la modification. Voici un exemple de requête de modification :



Si on effectue une requête GET sur le lieu de *geonameid* 7777777, on voit que la modification a bien été prise en compte :

GET https://digidata.api.localhost/data/7777777 Send

Params Auth Headers (11) Body Pre-req. Tests Settings

Body 200 OK 39 ms 518 B Save Response

Pretty Raw Preview Visualize JSON

```

1  {
2    "data": {
3      "admin1_code": "",
4      "admin2_code": "",
5      "admin3_code": "",
6      "admin4_code": "",
7      "alternatenames": "",
8      "asciiname": "TestModif",
9      "cc2": "",
10     "country_code": "FR",
11     "dem": "",
12     "elevation": "",
13     "feature_class": "",
14     "feature_code": "",
15     "geonameid": "7777777",
16     "latitude": "",
17     "longitude": "",
18     "modification_date": "2021-02-28",
19     "name": "TèstModif",
20     "population": "",
21     "timezone": "Europe/Paris"
22   }
23 }

```

Exemple de requête incorrecte :

POST https://digidata.api.localhost/data/7777777 Send

Params Auth Headers (11) Body Pre-req. Tests Settings

raw JSON Beautify

```

1  {"geonameid": "8888888", "asciiname": "TestModif2"}

```

Body 400 BAD REQUEST 16 ms 224 B Save Response

Pretty Raw Preview Visualize JSON

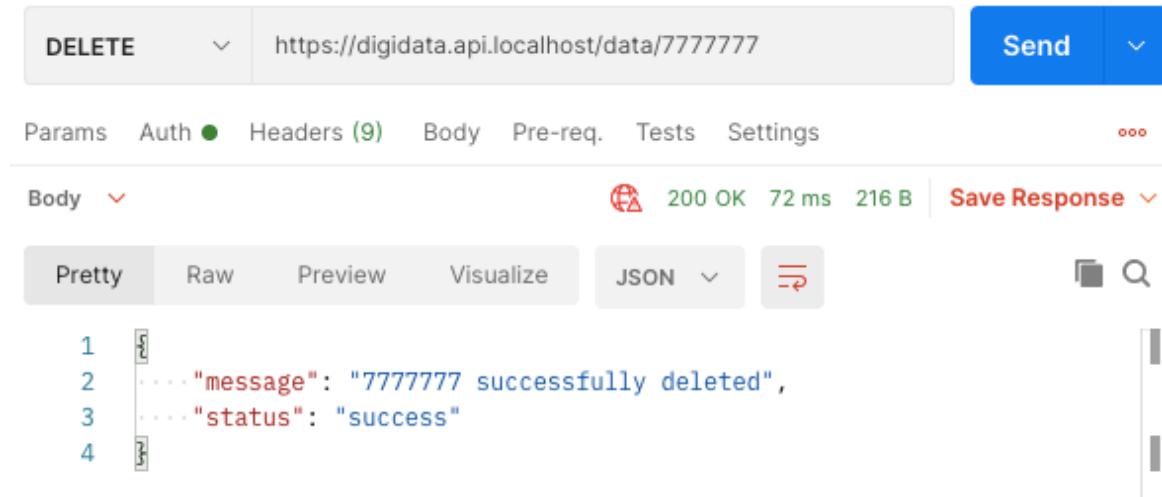
```

1  {
2    "message": "geonameid cannot be modified",
3    "status": "failed"
4  }

```

## Supprimer des données (-DELETE-)

Enfin, la méthode DELETE permet de supprimer des données. Comme à l'accoutumée, passer le *geonameid* en paramètre est obligatoire.

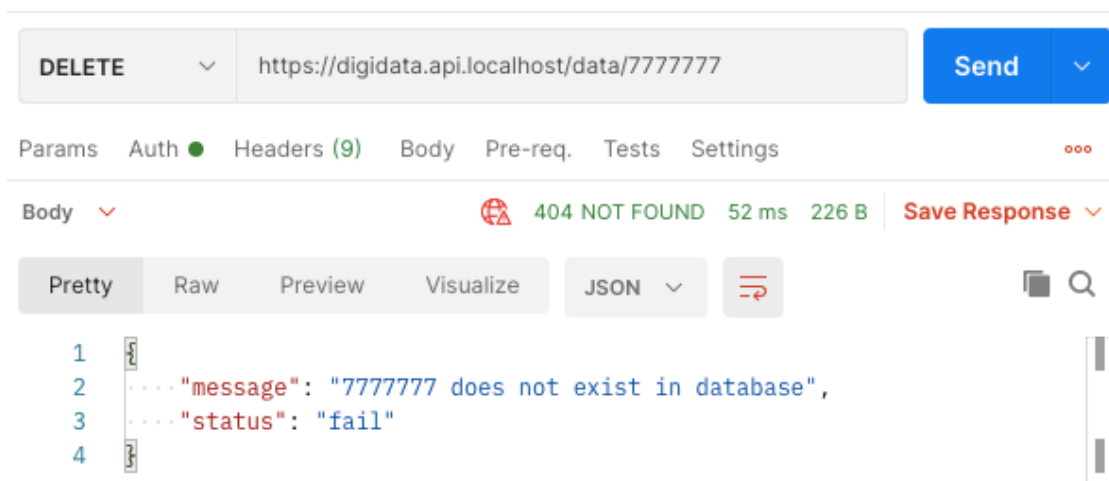


The screenshot shows a REST client interface with the following details:

- Method:** DELETE
- URL:** https://digidata.api.localhost/data/7777777
- Status:** 200 OK
- Time:** 72 ms
- Size:** 216 B
- Response Body (JSON):**

```
{
  "message": "7777777 successfully deleted",
  "status": "success"
}
```

Si ce dernier n'existe pas, une erreur est retournée. Par exemple, si on essaie de supprimer une deuxième fois l'élément 7777777, qui n'existe donc déjà plus dans la base de données :



The screenshot shows a REST client interface with the following details:

- Method:** DELETE
- URL:** https://digidata.api.localhost/data/7777777
- Status:** 404 NOT FOUND
- Time:** 52 ms
- Size:** 226 B
- Response Body (JSON):**

```
{
  "message": "7777777 does not exist in database",
  "status": "fail"
}
```

# Documentation de l'API

---

La documentation de l'API, générée à l'aide de l'outil Apidocjs (<https://apidocjs.com/>) est disponible dans le dossier doc/backend du projet. Pour y accéder en local, il suffit d'ouvrir le fichier index.html, puis de naviguer dans la documentation.