

---

# Projet du cours Techniques web

---

Création d'une  
application Back-End  
et d'une application  
Front-End

---

Chinatsu Kuroiwa, Solveig  
Poder et Camille Rey

---

# Table des matières

<b>PRE-REQUIS ET LANCEMENT DE L'APPLICATION .....</b>	<b>2</b>
Lancement en local .....	2
Lancement sur Heroku .....	3
 <b>BACKEND : REQUÊTES SUR POSTMAN.....</b>	<b>4</b>
Authentification .....	4
Login (-POST-) .....	4
Logout (-GET-) .....	6
Données .....	7
Afficher les données (-GET-) .....	7
Ajouter des données (-PUT-).....	9
Modifier des données (-POST-).....	11
Supprimer des données (-DELETE-) .....	13
 <b>FRONTEND : REQUÊTES DANS UN NAVIGATEUR.....</b>	<b>14</b>
Index .....	14
Authentification .....	14
Manipulation des données .....	16
Recherche de données .....	16
Consulter une ressource spécifique .....	18
Modifier des données .....	18
Supprimer des données .....	20
Ajouter des données .....	20
 <b>DOCUMENTATIONS TECHNIQUES.....</b>	<b>22</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 du dossier config\_nginx, éditez les fichiers **backend.conf** et **frontend.conf** : Dans chaque fichier, remplacez les deux occurrences de CHEMIN\_DU\_REPERTOIRE\_PROJET par le chemin absolu de la racine du répertoire projet, et enregistrez.  
\* **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 les fichiers **backend.conf** et **frontend.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/backend.conf /etc/nginx/sites-enabled/backend.conf**

**ln -s /etc/nginx/sites-available/frontend.conf /etc/nginx/sites-enabled/frontend.conf**

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

**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 backend devrait désormais être accessible en requêtes à partir de l'URL <http://digidata.api.localhost/> ou bien <https://digidata.api.localhost/> , et le serveur frontend devrait être accessible à partir de l'URL <http://digidata.localhost/> ou bien <https://digidata.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, les serveurs backend et frontend devraient être accessible respectivement aux adresses <http://127.0.0.1:5000> et <http://127.0.0.1:8000>.

Si même gunicorn ne fonctionne pas, les serveurs devraient être accessibles aux adresses <http://127.0.0.1:5000> et <http://127.0.0.1:8000> après exécution de la commande :

```
python run.py
```

## Lancement sur Heroku

Les deux applications backend et frontend sont déjà déployées sur Heroku, respectivement aux adresses <https://projet-tecweb-backend.herokuapp.com> et <https://projet-tecweb-frontend.herokuapp.com>. On peut donc tester les requêtes à partir de ces URL à la place de `http(s)://api.digidata.api.localhost`.

# Backend : Requêtes sur Postman

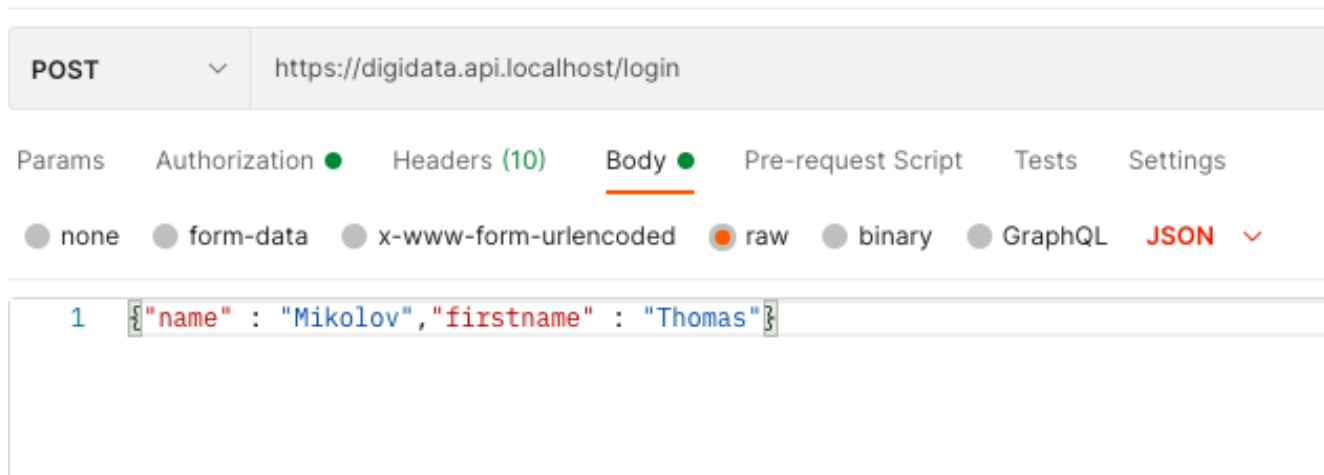
Toutes les requêtes présentées ici sont faites à partir du client web Postman (<https://www.postman.com/>), nous recommandons de l'installer pour pouvoir facilement effectuer les mêmes requêtes.

## Authentication

### 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 :

Pretty

Raw

Preview

Visualize

JSON



```

1 {
2   ... "Token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpZCI6IjAwMSIsImV4cCI6MTYxNDUwODU1Mn0.
   ... d9r3tLOU6U1rkCBthbYV1pJbS39q170B8HuwEUa3Q9Q",
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.d9r3tLOU6U1rkCBthbYV1pJbS39q170B8HuwEUa3Q9Q
```

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.d9r3tLOU6U1rkCBthbYV1pJbS39ql70B8HuwEUa3Q9Q",
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      "asciiiname": "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 renvoyée (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 displays a REST client interface with the following details:

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

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

```
1 {  
2   ... "message": "new data 7777777 successfully added",  
3   ... "status": "success"  
4 }
```

Quelques exemples de requêtes incorrectes :

- Mauvais champ :

PUT ▼ https://digidata.api.localhost/data/8888888 Send ▼

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

raw ▼ **JSON** ▼ Beautify

```
1 {"name":"Tèst", "ascii":"Test", "country_code" : "FR",  
  "timezone":"Europe/Paris"}
```

Body ▼ 400 BAD REQUEST 12 ms 235 B Save Response ▼

Pretty Raw Preview Visualize **JSON** ▼ ≡ 📄 🔍

```
1 {  
2   ... "message": "wrong format (field does not exist in db)",  
3   ... "status": "fail"  
4 }
```

- champ « name » non précisé :

PUT ▼ https://digidata.api.localhost/data/8888888 Send ▼

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

raw ▼ **JSON** ▼ Beautify

```
1 {"ascii":"Test", "country_code" : "FR", "timezone":"Europe/Paris"}
```

Body ▼ 400 BAD REQUEST 17 ms 234 B Save Response ▼

Pretty Raw Preview Visualize **JSON** ▼ ≡ 📄 🔍

```
1 {  
2   ... "message": "missing 'name' field of data to create",  
3   ... "status": "failed"  
4 }
```

## 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 générera 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 :

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

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

```
1 { "asciiname": "TestModif", "name" : "TèstModif" }
```
- Response (JSON):**

```
1 {  
2   ... "message": "7777777 successfully modified",  
3   ... "status": "success"  
4 }
```
- Status:** 200 OK, 30 ms, 217 B
- Buttons:** Send, Beautify, Save Response, Pretty, Raw, Preview, Visualize, JSON, and search icons.

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

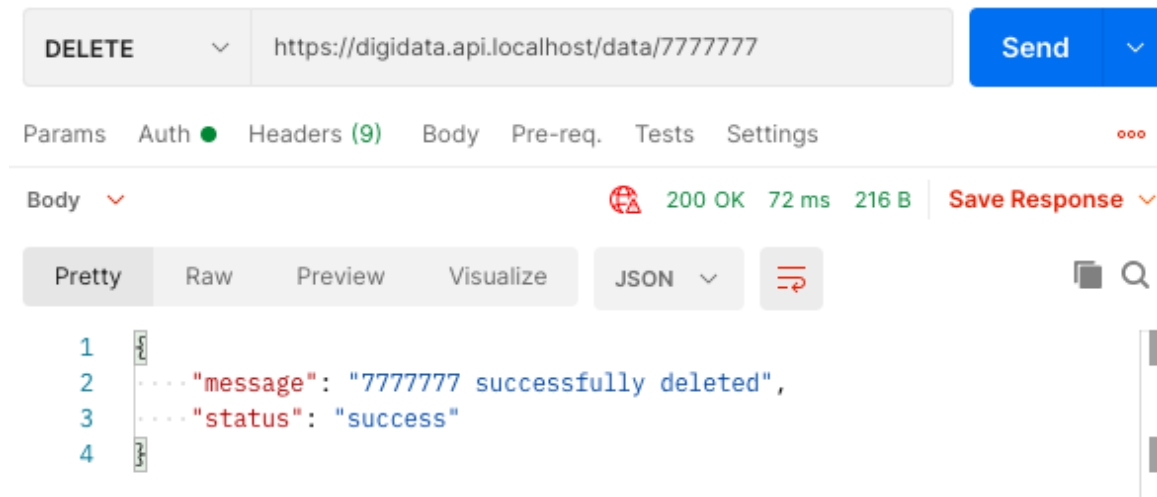
```

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.

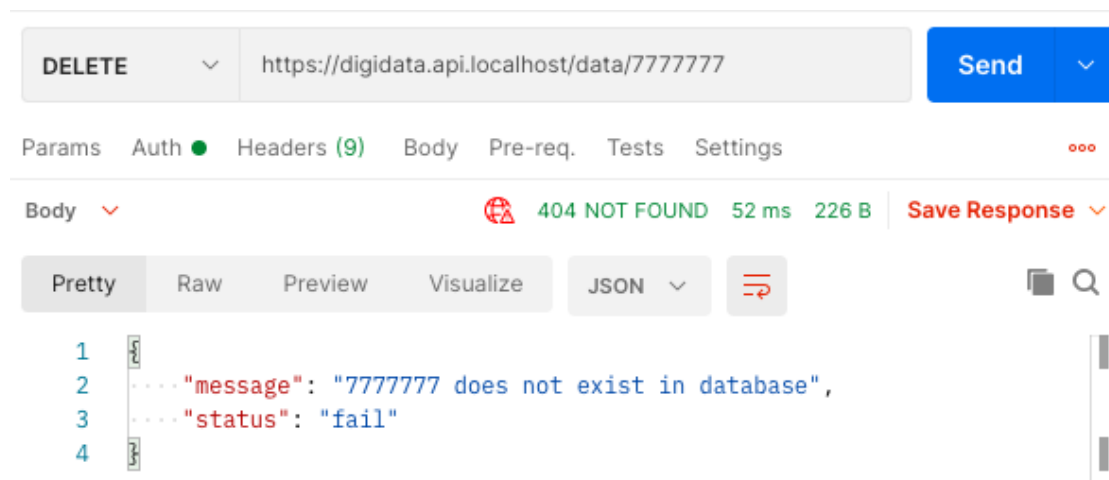


The screenshot shows a REST client interface with the following details:

- 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"}
```

# Frontend : Requêtes dans un navigateur

## Index

Rendez-vous à l'adresse <http://digidata.localhost/> (ou https) où vous tomberez sur la page d'accueil de notre site internet. Vous pourrez par la suite y retourner à tout moment de votre navigation, en cliquant sur **Digitata** en haut à gauche :



## Authentification

Pour accéder au contenu du site, il est nécessaire de s'identifier. En effet, si vous essayez d'accéder aux données sans être authentifié, vous serez dirigé vers la page de login avec un message d'erreur :

### LOGIN

Identification requise pour accéder aux données, veuillez vous identifier :

Depuis la page d'accueil, vous pouvez vous identifier en cliquant sur **Login/Logout** en haut à droite, puis en renseignant nom et prénom :

## LOGIN

Nom

---

Prénom

---

Se connecter

Si les identifiants sont incorrects, un message d'erreur s'affiche :

## LOGIN

Mot de passe ou identifiant incorrect. Veuillez réessayer :

Si les identifiants renseignés sont corrects, un message d'accueil apparaît et il vous est possible de vous déconnecter en cliquant sur le bouton :

## LOGIN

Bonjour Thomas Mikolov, vous êtes bien connecté(e) !

Se déconnecter

Il vous est possible de vous déconnecter à tout moment en cliquant sur **Login/Logout** dans la barre de navigation.



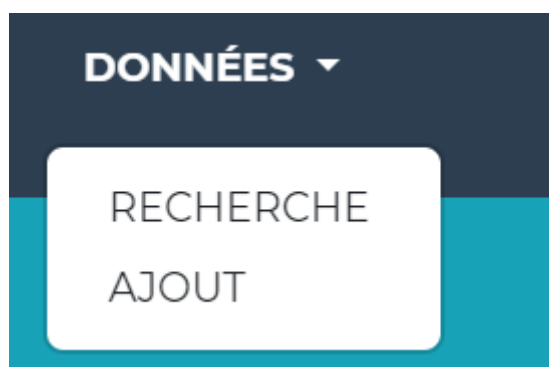
La déconnexion est automatique après une heure de navigation sur le site. Si vous essayez d'accéder à une page une fois ce temps écoulé, vous serez redirigé sur la page de login avec le message d'erreur suivant :

## LOGIN

Votre session est expirée ou invalide, veuillez vous ré-authentifier :

## Manipulation des données

Afin d'accéder aux données, cliquer sur **Données** dans la barre de navigation, qui affichera un menu déroulant. Dans un premier temps, on cliquera sur « Recherche » :



### Recherche de données

Un formulaire s'affiche et donne la possibilité de filtrer avec trois champs : geonameid, nom et asciiname :

## RECHERCHE DE DONNÉES

geonameid

nom

asciiname

En cliquant sur « Recherche avancée », vous aurez accès à tous les champs de la base de données :

geonameid	nom	asciiname
<input type="text" value="1234567"/>	<input type="text" value="La Réunion"/>	<input type="text" value="La Reunion"/>
Autre nom	latitude	longitude
<input type="text"/>	<input type="text"/>	<input type="text"/>
classe feature	code feature	code pays
<input type="text"/>	<input type="text"/>	<input type="text"/>
cc2	code admin1	code admin2
<input type="text"/>	<input type="text"/>	<input type="text"/>
code admin3	code admin4	population
<input type="text"/>	<input type="text"/>	<input type="text"/>
elevation	dem	Fuseau horaire
<input type="text"/>	<input type="text"/>	<input type="text"/>
date de modification		
<input type="text"/>		

Il est nécessaire d'en remplir au moins un pour que la recherche aboutisse. La liste des lieux correspondant aux critères sélectionnés s'affiche en bas de la page :

## RÉSULTATS

Resultat	geonameid	nom	asciiname	alternatenames	latitude	longitude	feature_class	feature_code	co
1	2659815	Lucelle	Lucelle	La Lucelle Riviere,La Lucelle Rivière,Lucelle,Lutzel,Lützel	47.41667	7.5	H	STM	FR
2	6617438	Lucelle	Lucelle	Ljusel',Lucelle,Luetzel,Lützel,Ju sai er,Люсель,吕塞尔	47.4228	7.2476	A	ADM4	FR
3	6620163	Lucelle	Lucelle	Ljusel',Lucelle JU,Iu sai er,Люсель,吕塞尔	47.4228	7.2476	P	PPL	FR

De là, vous pouvez accéder à la page d'un lieu en particulier en cliquant sur son geonameid.

## Consulter une ressource spécifique

La page d'un lieu se présente sous cette forme :

**2659815**

geonameid	2659815
nom	Lucelle
asciiname	Lucelle
alternatenames	La Lucelle Riviere,La Lucelle Rivière,Lucelle,Lutzel,Lützel
latitude	47.41667
longitude	7.5
feature_class	H
feature_code	STM

Tout en bas à droite de la liste des caractéristiques du lieu se trouvent deux boutons : « Modifier » et « Supprimer ».



## Modifier des données

Lorsque l'on clique sur « Modifier », un formulaire s'affiche avec tous les champs ouverts à la modification (c'est-à-dire tous les champs exceptés geonameid et la date de modification qui s'actualise automatiquement à chaque modification). Les champs sont pré-remplis avec les valeurs actuelles :

nom	asciiname	Autres noms
<input type="text" value="Lucelle"/>	<input type="text" value="Lucelle"/>	<input type="text" value="La Lucelle Riviere,La Lucelle F"/>
latitude	longitude	classe feature
<input type="text" value="47.41667"/>	<input type="text" value="7.5"/>	<input type="text" value="H"/>
code feature	code pays	cc2
<input type="text" value="STM"/>	<input type="text" value="FR"/>	<input type="text"/>
code admin1	code admin2	code admin3
<input type="text" value="00"/>	<input type="text"/>	<input type="text"/>
code admin4	population	elevation
<input type="text"/>	<input type="text" value="0"/>	<input type="text"/>
dem	Fuseau horaire	
<input type="text" value="353"/>	<input type="text" value="Europe/Paris"/>	
<input type="button" value="Modifier"/>		

Après soumission des modifications (une confirmation est demandée), un message de confirmation s'affiche ainsi que la page du lieu mise à jour :

**2659815**

L'élément **2659815** a bien été modifié.

geonameid	2659815
nom	Nouveau nom
asciiname	Lucelle
alternatenames	La Lucelle Riviere,La Lucelle Rivière,Lucelle,Lutzel,Lützel
latitude	47.41667
longitude	7.5
feature_class	H
feature_code	STM

## Supprimer des données

Lorsque l'on clique sur « Supprimer » (une confirmation est demandée), le lieu est supprimé de la base de données. Un message de confirmation apparaît alors ainsi qu'un lien vers la page de recherche :

L'élément 111111 a bien été supprimé.

[Retourner à la page des données](#)

## Ajouter des données

Pour ajouter un lieu à la base de données, cliquer sur **Données** dans la barre de navigation pour afficher le menu déroulant puis cliquer sur « Ajout ». Un formulaire similaire à celui de la recherche avancée apparaît (seul le champ concernant la date de modification est absent puisqu'il est complété automatiquement lors de l'ajout). Les champs geonameid et nom sont obligatoires.

geonameid	nom	asciiname
<input type="text" value="1234567"/>	<input type="text" value="La Réunion"/>	<input type="text"/>
Obligatoire	Obligatoire	
Autres noms	latitude	longitude
<input type="text"/>	<input type="text"/>	<input type="text"/>
classe feature	code feature	code pays
<input type="text"/>	<input type="text"/>	<input type="text"/>
cc2	code admin1	code admin2
<input type="text"/>	<input type="text"/>	<input type="text"/>
code admin3	code admin4	population
<input type="text"/>	<input type="text"/>	<input type="text"/>
elevation	dem	Fuseau horaire
<input type="text"/>	<input type="text"/>	<input type="text"/>

Ajouter

Après soumission du nouveau lieu à ajouter (une confirmation est demandée), un message de confirmation s'affiche :

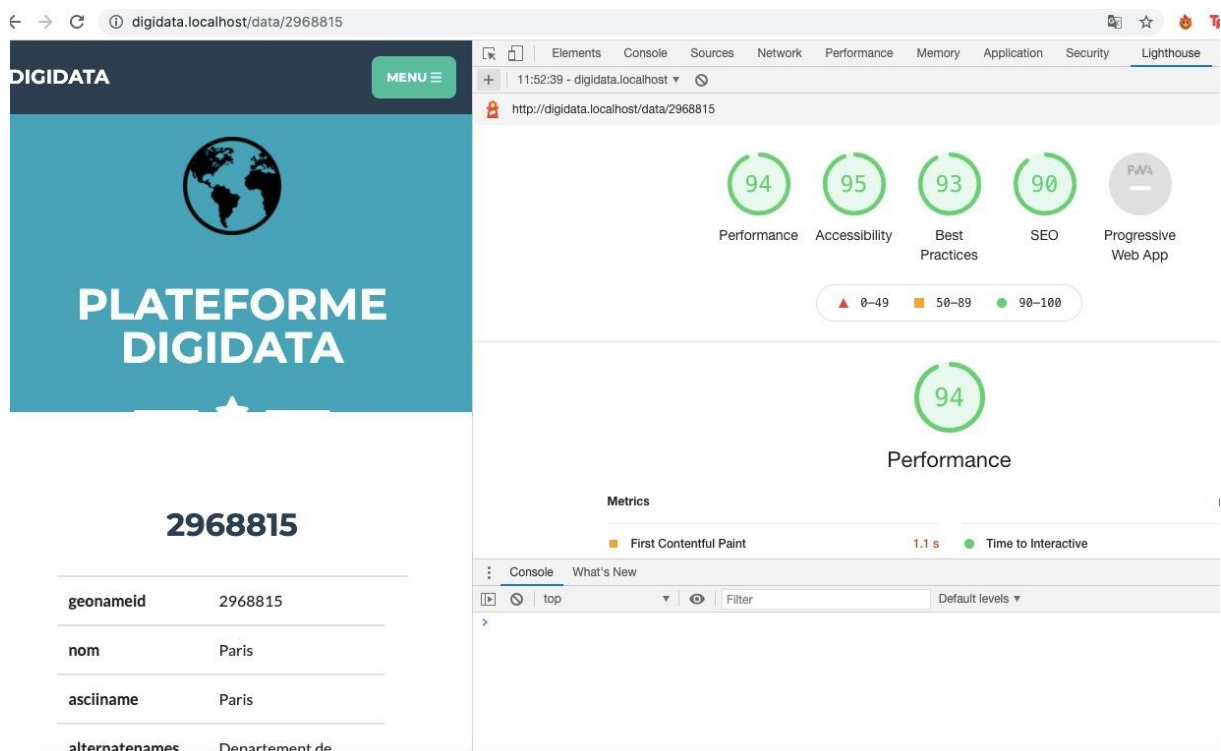
L'élément 111111 a bien été créé.

Le geonameid étant un identifiant unique, il est impossible d'ajouter un lieu dont le geonameid existe déjà dans la base de données. Cela génèrera le message suivant :

Un élément de geonameid 111111 existe déjà

## Conclusion

L'application en frontend fournit une interface intuitive et esthétique pour interagir avec le backend, sans avoir à passer par un client tel que POSTMAN. Cette application est globalement performante : les rapports fournis par l'outil Lighthouse de Google Chrome indiquent des résultats de performances globaux sur desktop entre 89 et 95% en fonction des pages :



# Documentations techniques

---

La documentation de l'API backend, 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.

Pour la documentation du front-end, nous avons utilisé l'outil Sphinx (<https://www.sphinx-doc.org/>), car il permet un format plus souple, qui nous semblait plus adapté pour documenter le front-end. Celle-ci est disponible dans le dossier doc/frontend du projet. Pour y accéder en local, il suffit d'ouvrir le fichier index.html, puis de naviguer dans la documentation.