

# Rapport de débogage sur le projet Rennes Traffic

## Introduction

Rennes Traffic est un mini projet permettant de prédire les zones de congestions du trafic routier sur la ville de Rennes.

Ce projet a été réalisé avec Flask pour le backend, du HTML pour l'unique page d'interface et tensorflow pour le modèle de prédiction.

Plusieurs bugs empêchent l'application de fonctionner correctement.

## Corrections des bugs

Le fichier app.py contient une seule route avec une méthode GET et une autre POST. Le processus de résolution a été de tenter de lancer l'application jusqu'à arriver à faire fonctionner les deux routes avec une mise à jour du résultat sur la page web.

### Bug #1:

Une virgule manquait entre deux paires clé-valeur d'un dictionnaire. Cette faute de syntaxe empêchait l'application de démarrer.

```
File "C:\Users\HP\debugging - Simplon\rennes_traffic_ko\src\utils.py", line 15
    zoom=10 # CORRECTION : comma was forgotten
    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
SyntaxError: invalid syntax. Perhaps you forgot a comma?
```

Correction :

```
zoom=10, # CORRECTION : comma was forgotten
```

### Bug #2:

L'input avait une longueur de 25, chaque élément représente une heure, sa longueur doit donc être de 24. Ce bug empêchait toute prédiction mais pas l'affichage de la page web..

### ValueError

```
ValueError: Exception encountered when calling Sequential.call().

[[1mInput 0 of layer "dense_93" is incompatible with the layer: expected axis -1 of input shape to have value 24, but received input with shape (1, 25)]]

Arguments received by Sequential.call():
  • inputs=tf.Tensor(shape=(1, 25), dtype=int32)
  • training=False
  • mask=None
```

Correction:

```
input_pred = np.array([0]*24) # CORRECTION : there must be 24 hours not 25
```

### Bug #3:

Les clés "latittude" et "longitude" n'existent pas, ce sont respectivement "lat" et "lon" et la clé "traffic\_status" n'existe pas, il s'agit de "trafficstatus".

Ce bug empêchait l'application de charger les données au lancement.

```
File "C:\Users\HP\debugging - Simplon\rennes_traffic_ko\src\get_data.py", line 15, in <lambda>
    temp['lat'] = temp.geo_point_2d.map(lambda x : x['latittude']) # CORRECTION : nkey is lat not lattitude
                        ~^^^^^^^^^^^^^^^^
KeyError: 'latittude'
```

Correction :

```
temp['lat'] = temp.geo_point_2d.map(lambda x : x['lat']) # CORRECTION : key is lat not lattitude
temp['lon'] = temp.geo_point_2d.map(lambda x : x['lon']) # CORRECTION : key is lon not longitude
```

```
ime', 'geo_point_2d', 'averagevehiclespeed', 'traveltime', 'trafficstatus']})
CORRECTION : idem
```

### Bug #4:

Le template "home.html" indiqué n'existe pas. Il s'agit de "index.html".

Cela rendait impossible l'accès à la page de web.

```
File "C:\Users\HP\debugging - Simplon\rennes_traffic_ko\env\Lib\site-packages\flask\templating.py", line 55, in
    _get_source_fast
    raise TemplateNotFound(template)
jinja2.exceptions.TemplateNotFound: home.html
```

Correction :

```
return render_template('index.html', graph_json=graph_json) # CORRECTION : Template is named index.html, not home.html
```

### Bug #5:

L'heure sélectionnée par l'utilisateur n'était pas passée à la fonction "prediction\_from\_model" qui l'attendait.

Cela bloquait la requête de prédiction.

## TypeError

```
TypeError: prediction_from_model() missing 1 required positional argument:
'hour_to_predict'
```

### Traceback (most recent call last)

```
File "C:\Users\HP\debugging - Simplon\rennes_traffic_ko\env\Lib\site-packages\flask\app.py", line 1498, in
    __call__
```

```
    return self.wsgi_app(environ, start_response)
    ~~~~~
```

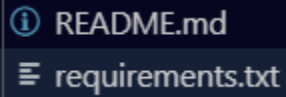
```
File "C:\Users\HP\debugging - Simplon\rennes_traffic_ko\env\Lib\site-packages\flask\app.py", line 1476, in
    wsgi_app
```

Correction :

```
cat_predict = prediction_from_model(model, selected_hour) # CORRECTION : 'selected_hour' parameter missing
```

### Ajout d'un requirements.txt

Ce n'est pas un bug à proprement parlé mais cela permettra au développeur suivant de connaître les versions des librairies utilisées.



A screenshot of a file explorer interface. It shows two files: 'README.md' with an information icon (i) and 'requirements.txt' with a list icon (≡).

## Monitoring

### Ajout des logs

Des logs ont été ajoutés afin de mieux identifier les sources de bugs.

```
logging.basicConfig(  
    level=logging.ERROR,  
    format='%(asctime)s : %(message)s',  
    # detailed format : '%(asctime)s %(levelname)s %(message)s'  
    handlers=[  
        logging.FileHandler("app.log"),  
        logging.StreamHandler()  
    ]  
)
```

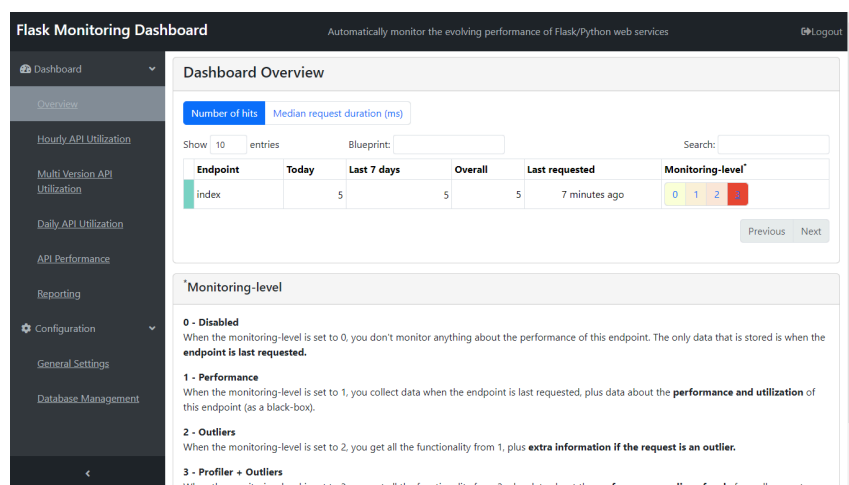
```

1 2024-08-26 14:47:33,534 DEBUG: Starting new HTTPS connection (1): data.rennesmetropole.fr:443
2 2024-08-26 14:47:35,808 DEBUG: https://data.rennesmetropole.fr:443 "GET /api/explore/v2.1/cat
3 2024-08-26 14:56:53,646 DEBUG: Starting new HTTPS connection (1): data.rennesmetropole.fr:443
4 2024-08-26 14:56:54,007 DEBUG: https://data.rennesmetropole.fr:443 "GET /api/explore/v2.1/cat
5 2024-08-26 14:56:54,640 DEBUG: dict_keys(['datetime', 'predefinedlocationreference', 'average
6 2024-08-26 15:00:41,196 DEBUG [connectionpool._new_conn]: Starting new HTTPS connection (1):
7 2024-08-26 15:00:41,609 DEBUG [connectionpool._make_request]: https://data.rennesmetropole.fr
8 2024-08-26 15:00:43,074 DEBUG [get_data.processing_one_point]: dict_keys(['datetime', 'predef
9 2024-08-26 15:08:02,499 DEBUG [connectionpool._new_conn]: Starting new HTTPS connection (1):
10 2024-08-26 15:08:02,880 DEBUG [connectionpool._make_request]: https://data.rennesmetropole.fr
11 2024-08-26 15:08:04,457 DEBUG [get_data.processing_one_point]: dict_keys(['datetime', 'predef
12 2024-08-26 15:08:04,462 DEBUG [get_data.processing_one_point]: Index(['datetime', 'geo_point_
13 | 'traffic'],
14 | dtype='object') [in C:\Users\HP\debugging - Simplon\rennes_traffic_ko\src\get_data.py:1
15 2024-08-26 15:09:17,523 DEBUG [connectionpool._new_conn]: Starting new HTTPS connection (1):
16 2024-08-26 15:09:19,303 DEBUG [connectionpool._make_request]: https://data.rennesmetropole.fr
17 2024-08-26 15:09:20,029 DEBUG [get_data.processing_one_point]: dict_keys(['datetime', 'predef
18 2024-08-26 15:09:20,037 DEBUG [get_data.processing_one_point]: 0 {'lon': -1.65021524355382
19 Name: geo_point_2d, dtype: object [in C:\Users\HP\debugging - Simplon\rennes_traffic_ko\src\g
20 2024-08-26 15:16:12,501 DEBUG [connectionpool._new_conn]: Starting new HTTPS connection (1):
21 2024-08-26 15:16:12,823 DEBUG [connectionpool._make_request]: https://data.rennesmetropole.fr

```

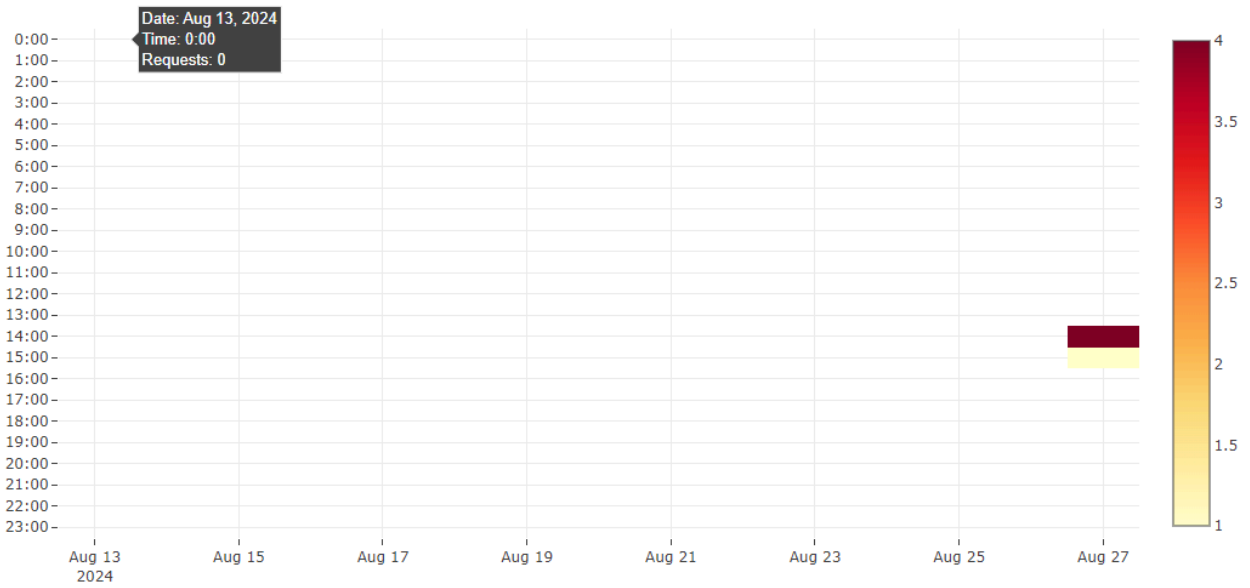
## Ajout d'un dashboard

Flask Monitoring Dashboard est une solution tout à fait adaptée pour monitorer l'utilisation de cette API.

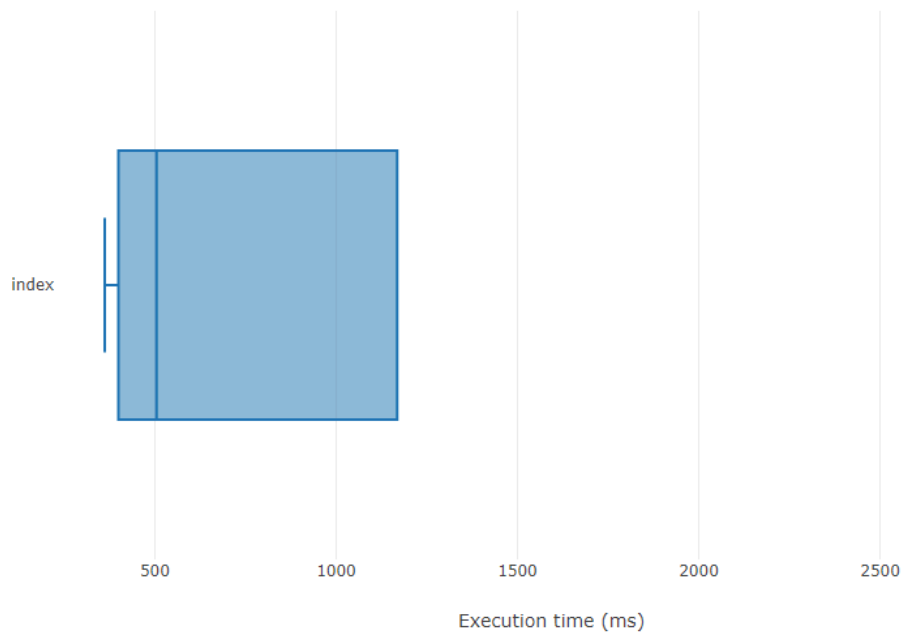


Simple d'intégration, il permet grâce à ses trois niveaux de monitoring de tracker :

*Le nombre de requêtes par heure*



*La distribution du temps d'exécution des requêtes par routes*



Le dashboard est accessible par les utilisateurs via un compte. Les administrateurs gèrent ces comptes utilisateurs et les paramètres de configuration du dashboard. Les utilisateurs par défaut peuvent seulement consulter les données.