NSI première - IHM sur le web

Mini Projet serveur météo

qkzk

Serveur météo

L'objectif de ce mini projet est de mettre en oeuvre ce que vous avez appris concernant les IHM sur le web, Flask ainsi que l'utilisation d'une api très simple.

Présentation du projet

Le principe est de créer :

- une page web (html, css) avec 2 vues
- un serveur web en Python avec Flask qui sert deux pages (formulaire, résultat de la requête)
- Le serveur communique avec openweathermap et intègre la météo de la ville choisie au résultat

---

Utilisation

- L'utilisateur se connecte via le navigateur sur votre site
- Il remplit un formulaire demandant une ville, par exemple : **Lille**
- Votre serveur redirige l'utilisateur vers une page contenant l'information : **Temp : 12.3°C etc.**

Comparaison avec le formulaire obtenu à la fin du cours

Très peu de nouveautés !

- Votre serveur transforme la réponse au formulaire en une requête API pour openweathermap
- Il envoie la question
- openweathermap renvoie des données
- votre serveur en extraie les infos
- qu'il intègre à la page `resultat.html`

Étapes

Vous allez d'abord :

1. Lire et réaliser tout le cours IHM sur le web (qkzk, nsi, premiere, ihm sur le web)
2. Réaliser l'activité finale avec le formulaire. *C'est le point de départ.*

De l'activité finale à la réalisation du projet

3. Apprendre à passer des requêtes openweathermap *directement dans le navigateur*
4. Réaliser ces requêtes en Python, dans la console
5. Combiner le tout pour créer l'application complète.

Outils

IDE

- Thonny gère Flask sans problème.
- N'importe quel IDE installé sur votre machine (atom, sublime text...) pour écrire le code html (et éventuellement Python)
- On *peut* servir un site depuis colab avec Flask mais c'est **sport.** Un exemple. Ouvrez le lien de la première réponse pour découvrir un exemple.

Librairies

Flask est la seule librairie dont vous ayez besoin. Elle est déjà présente sur votre machine.

Délais

Ce projet doit être terminé **début décembre**. J'attends de vous voir travailler pour fixer précisément le délai. Je vais rater beaucoup de séances alors prédire l'avenir est délicat mais cela me semble un délai raisonnable.

Résultat attendu

Le projet final est constitué des fichiers suivants :

```
.
|-- readme.md
|-- serveur.py
+-- templates
    |-- index.html
    +-- resultats.html
```

- `readme.md` est votre compte rendu (!). Vous le rédigerez en markdown. Un éditeur en ligne
- `serveur.py` : est l'unique fichier Python qui fait tout. Partez du résultat final de l'activité IHM sur le web "Flask" (le formulaire...) `views.py`
- `index.html` et `resultats.html` sont vos pages web servies par Flask

D'autres informations sur le projet à cette page :

- L'objectif est de comprendre, pas de devenir un développeur web full stack...
- Je vous invite à créer votre propre clé API openweather map, mais vous pouvez m'en demander une. Bien sûr, je ne vais pas la publier sur mon site...

Cahier des charges détaillé

1. Servir une page (statique d'abord) avec flask
2. Lire une requête du formulaire
3. Répondre avec des données statiques (sans passer par openweathermap) et les intégrer à la page correctement
4. Transformer la requête du formulaire et envoyer une commande à l'api openweathermap
5. Lire le json en réponse via json load, renvoyer ce json
6. Extraire les infos du json et remplir un dictionnaire avec la requête
7. Remplir le template de reponse avec les vraies données
8. Extensions : css, prévisions, carte de France avec les villes, ce que vous voulez.

Sources :

- tutoriel flask
- IHM sur le web; formulaire avec flask en NSI.
- api d'openweather map
- Exemple d'utilisation de l'api OWM en Python

Extensions

- Créer une carte de France et intégrer la météo de différentes villes
- Prévisions sur une semaine, formatage des images etc.
- Utilisation de capteurs pour récupérer la météo locale avec un raspberry
- Toute autre idée inspirée d'un site météo...

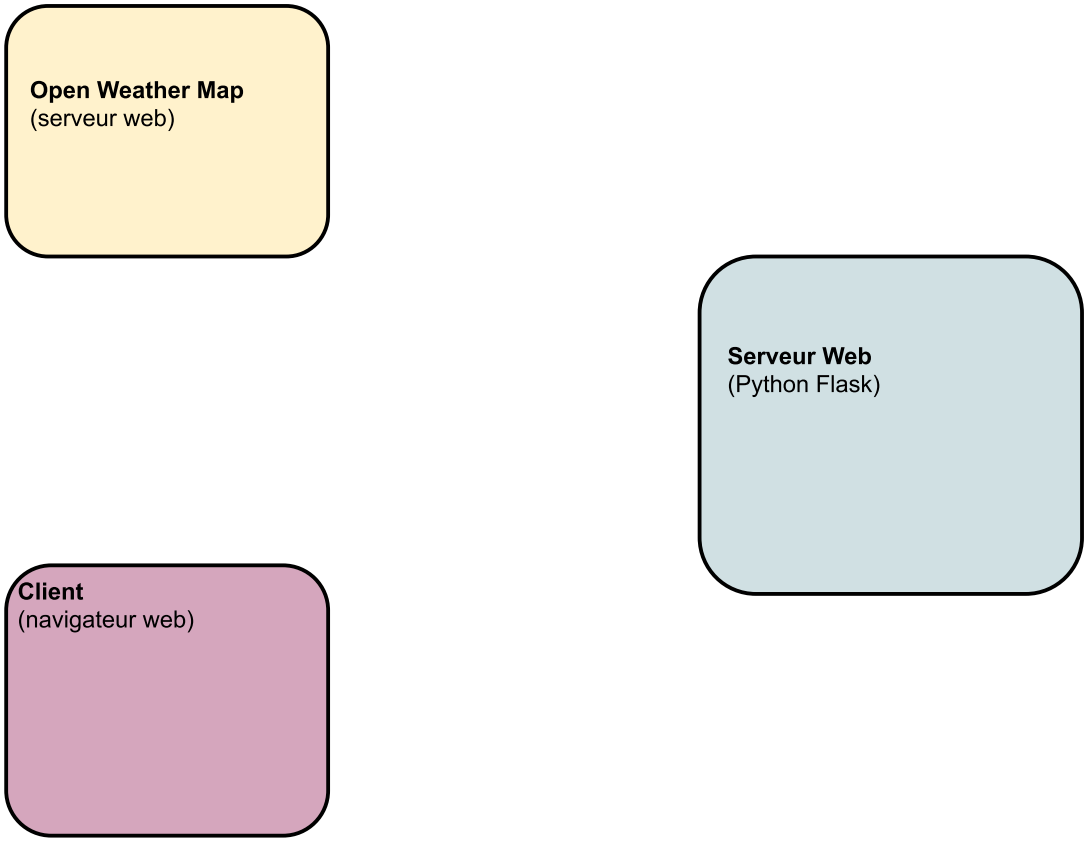
Les étapes d'une connexion client serveur

Les éléments

Le client (navigateur web)

Le serveur web (Python + Flask)

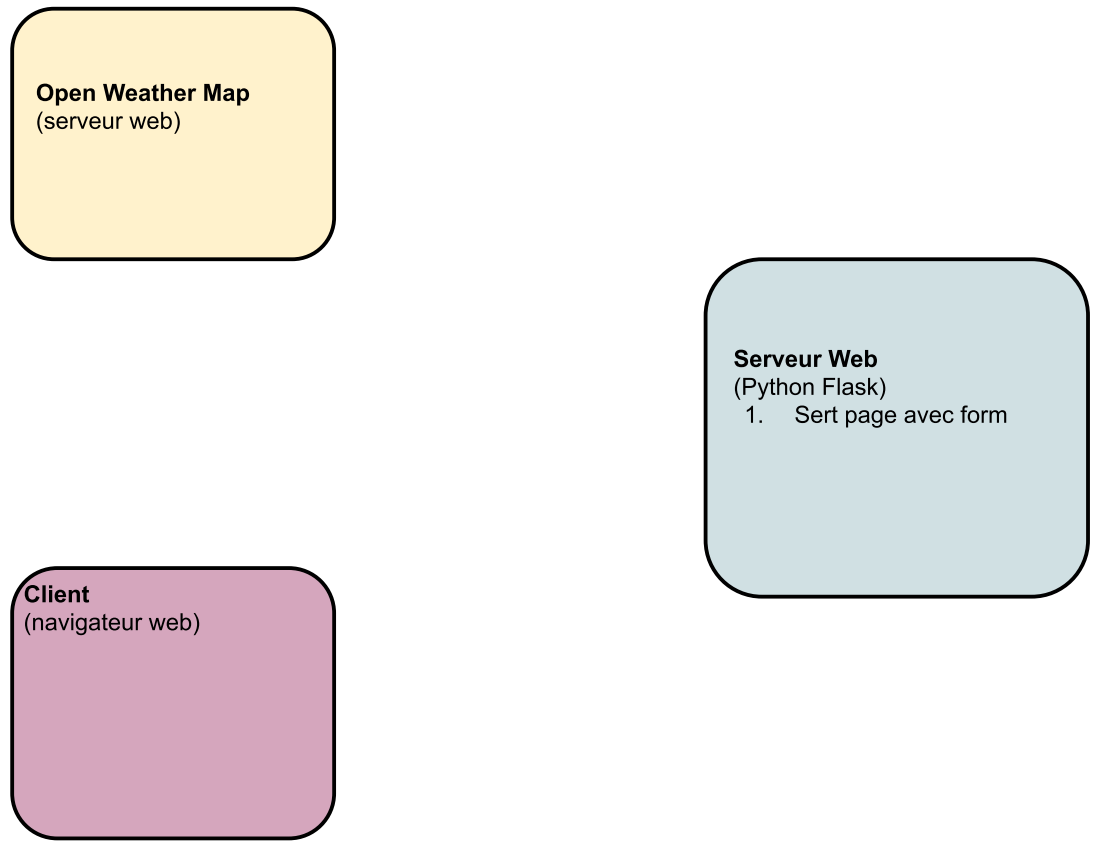
Et le site Openweathermap



-

Le serveur se lance

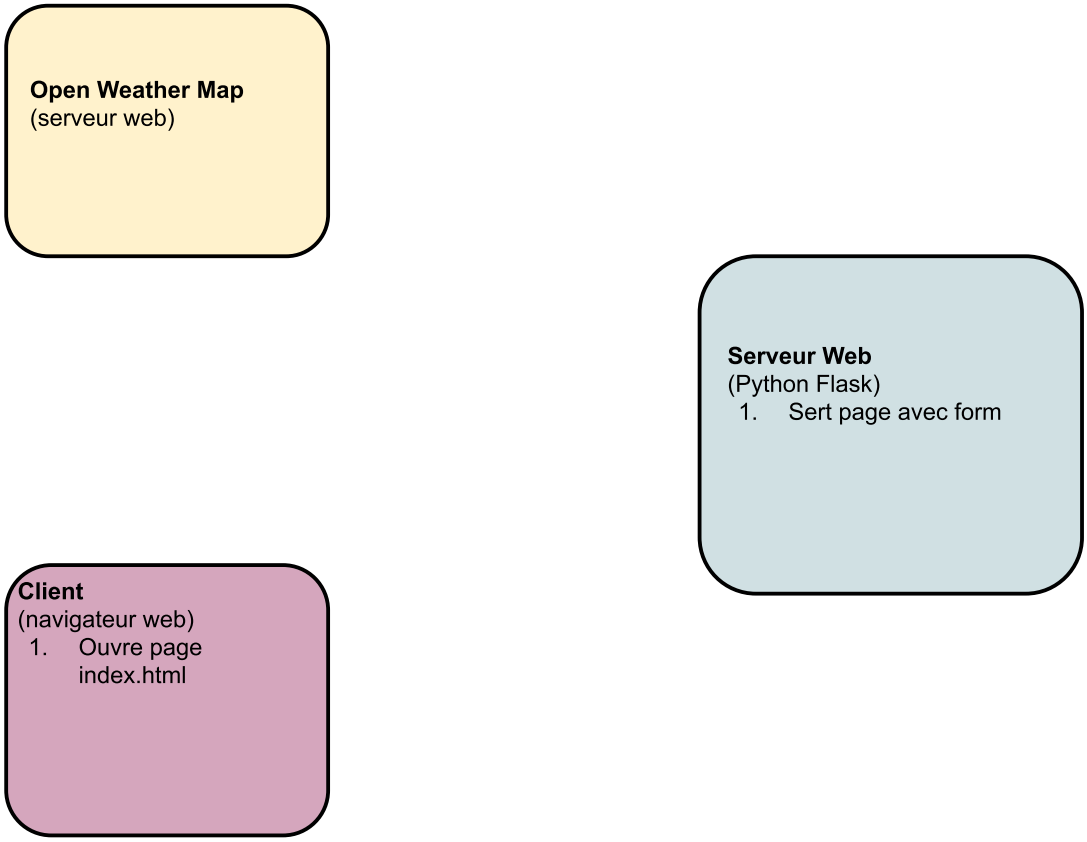
Quand on exécute le script Python du serveur il attend qu'un client se connecte...



-

Le client se connecte sur le site

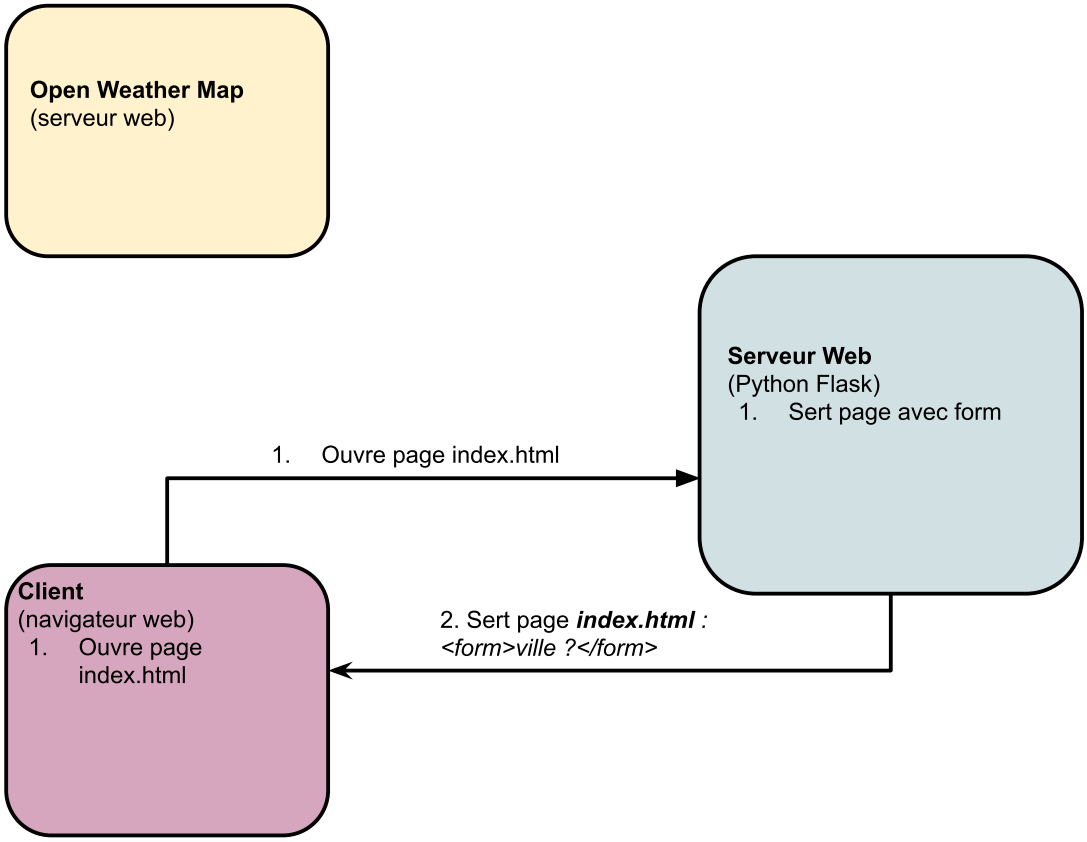
L'utilisateur a tapé l'adresse du site dans sa barre d'adresse...



-

Le serveur web renvoie alors une page html

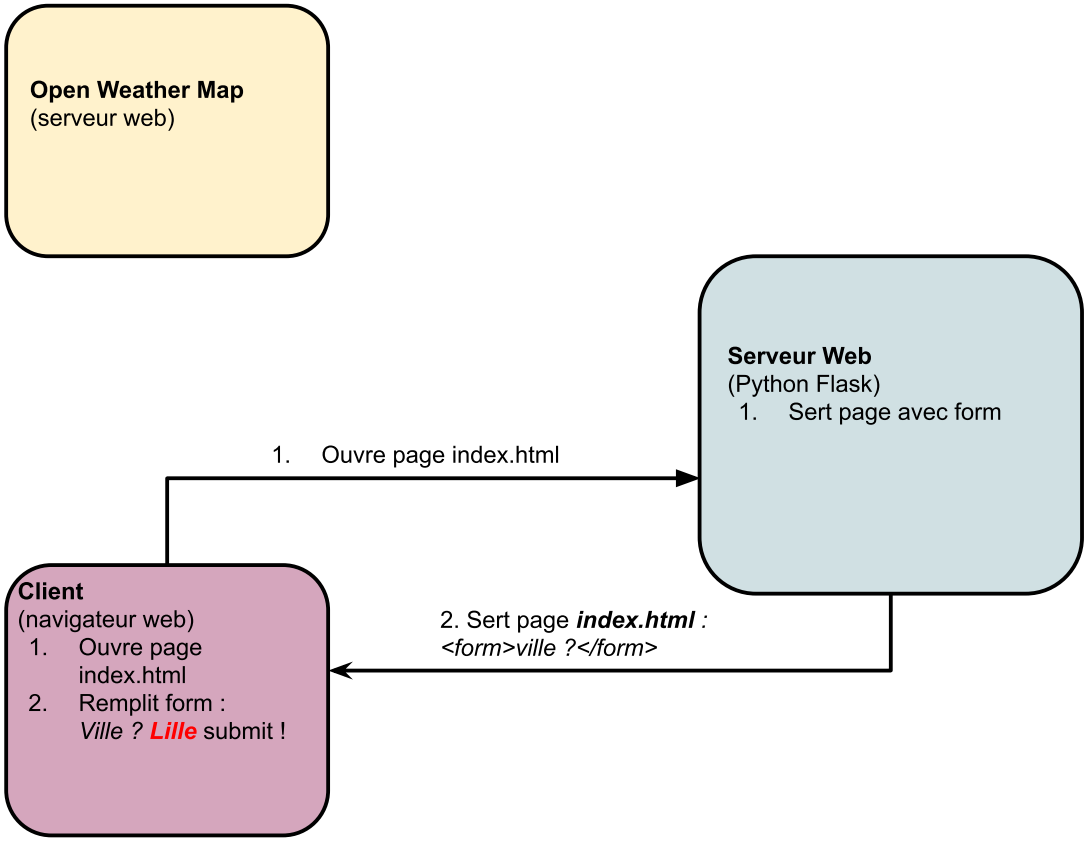
La page html est un formulaire à remplir (ville ?) avec un bouton *submit*



-

Formulaire rempli, *submit*

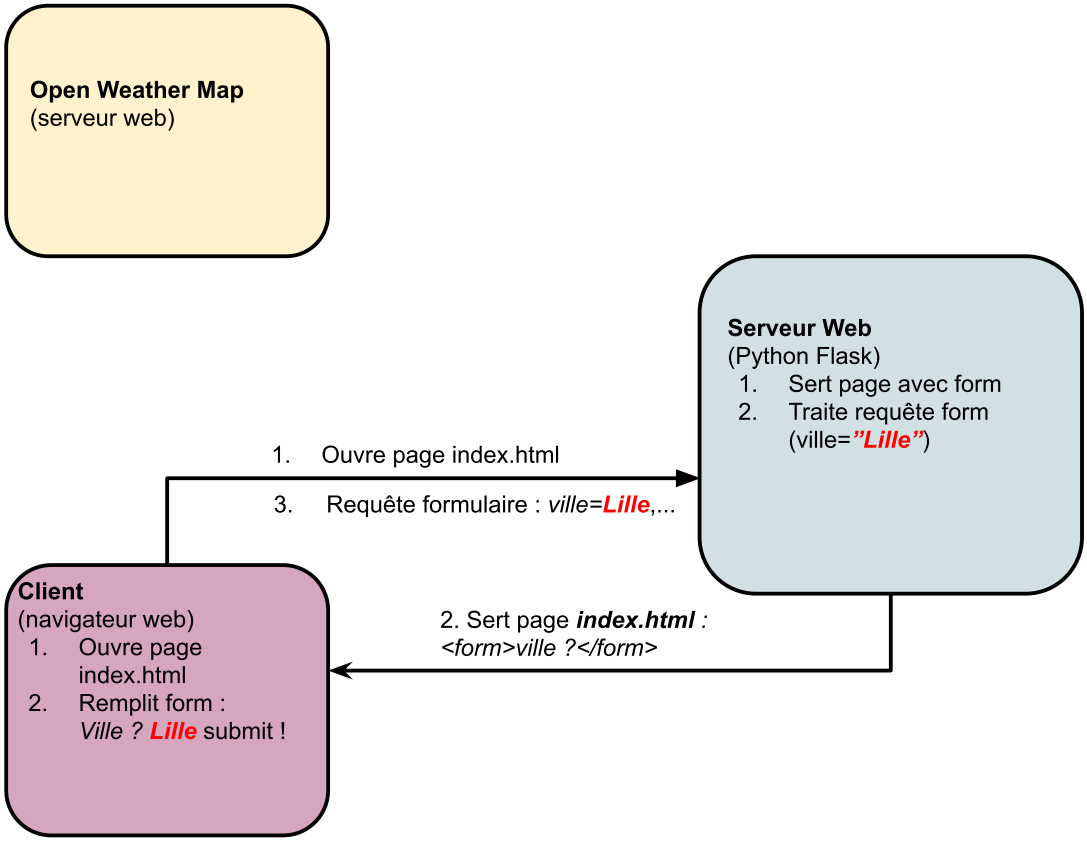
L'utilisateur a rempli son formulaire avec *Lille* et cliqué sur submit



-

Le serveur reçoit et traite la requête

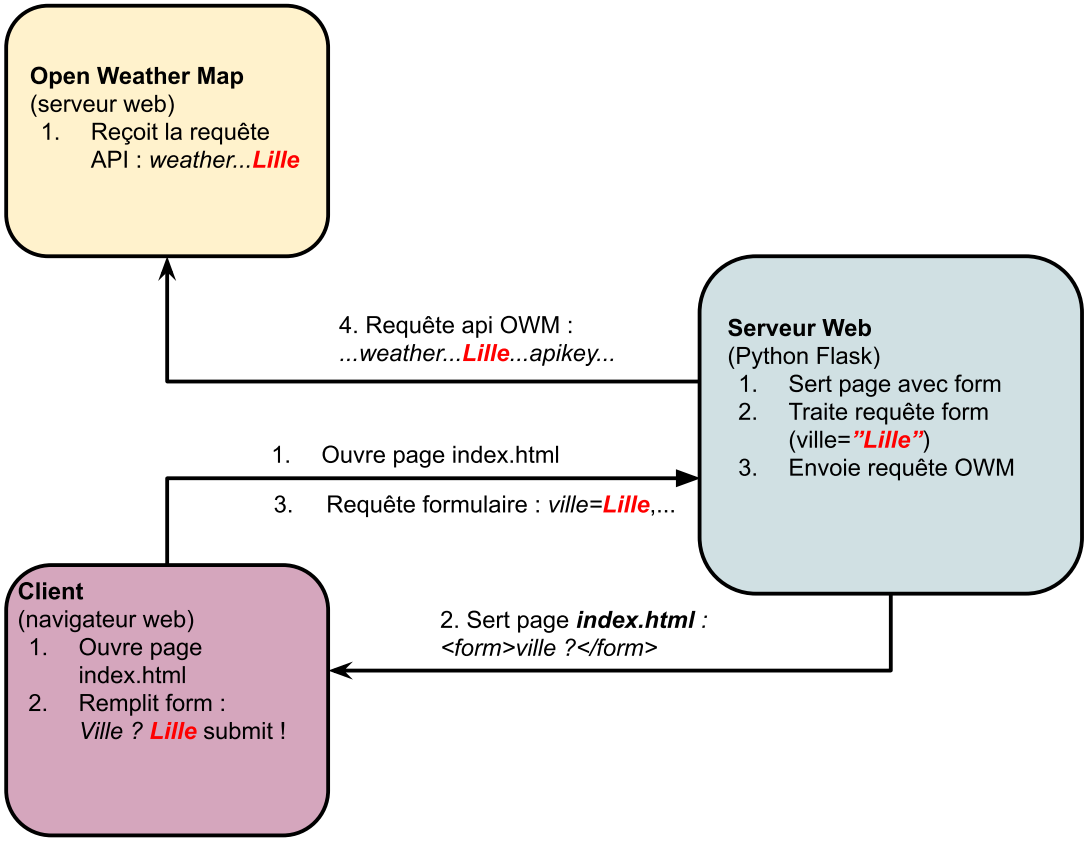
Le serveur Python récupère les données du formulaire et appelle une fonction...



-

Le serveur envoie à son tour une requête à OWM...

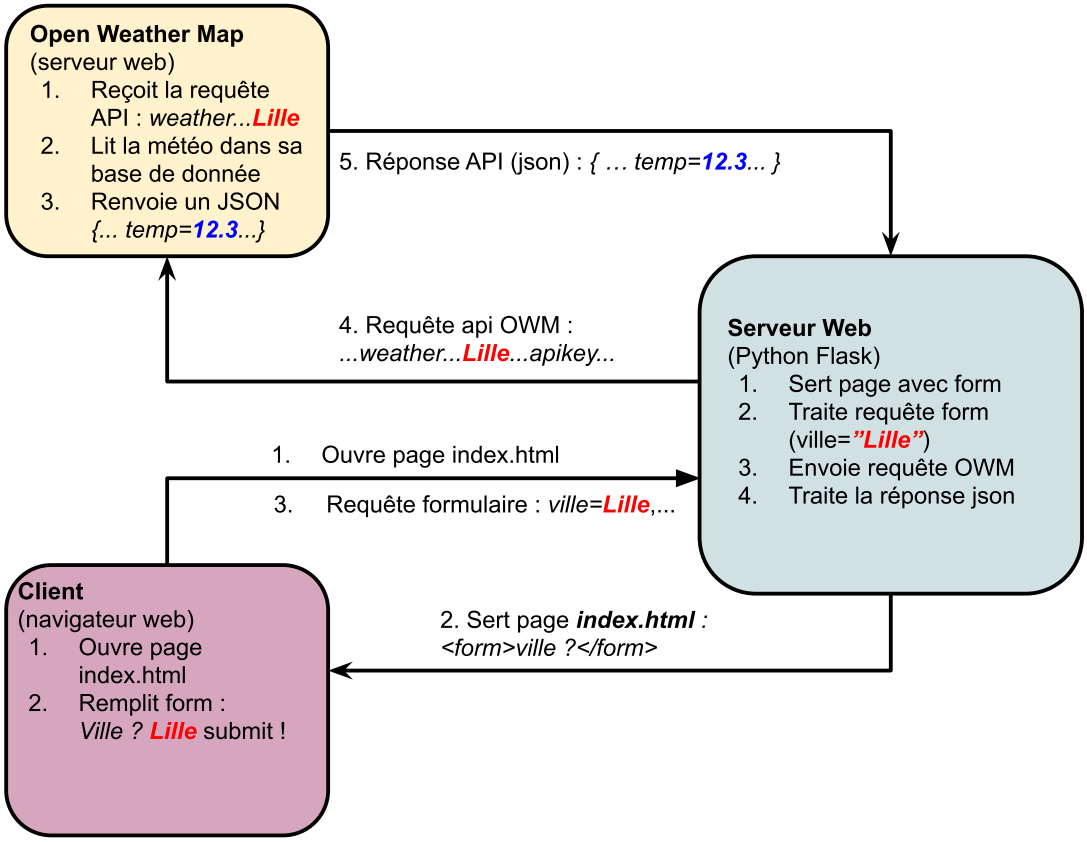
La requête est transmise à OWM via une URL (string...) qu'on formate



-

OWM lit traite la requête et renvoie la météo dans un JSON

JSON (Javascript Object Notation) est le format le plus populaire pour échanger des données sur le web. Les données ressemblent à un dictionnaire Python. Python manipule sans difficulté les JSON.



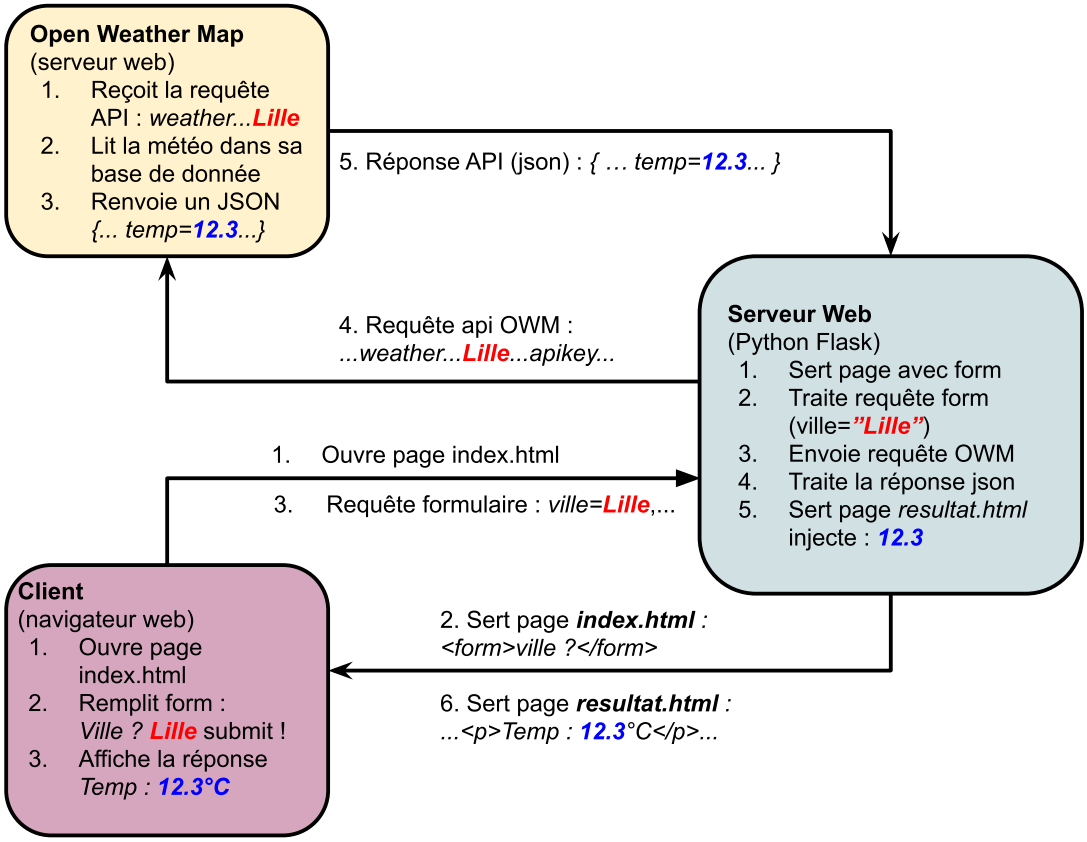
-

Le serveur traite la réponse et injecte la météo dans une page *resultat.html*

Elle est envoyée au client

Qui l'affiche dans le navigateur.

L'utilisateur lit la météo de Lille : **il fait 12.3°C** !


Open Weather Map
(serveur web)
1. Reçoit la requête API : weather...Lille
2. Lit la météo dans sa base de donnée
3. Renvoie un JSON {... temp=12.3...}
5. Réponse API (json) : { … temp=12.3... }
Serveur Web
(Python Flask)
1. Sert page avec form
2. Traite requête form (ville="Lille")
3. Envoie requête OWM
4. Traite la réponse json
5. Sert page resultat.html injecte : 12.3
4. Requête api OWM : ...weather...Lille...apikey...
1. Ouvre page index.html
3. Requête formulaire : ville=Lille,...
Client
(navigateur web)
1. Ouvre page index.html
2. Remplit form : Ville ? Lille submit !
3. Affiche la réponse Temp : 12.3°C
2. Sert page index.html : <form>ville ?</form>
6. Sert page resultat.html : ... Temp : 12.3°C ...