

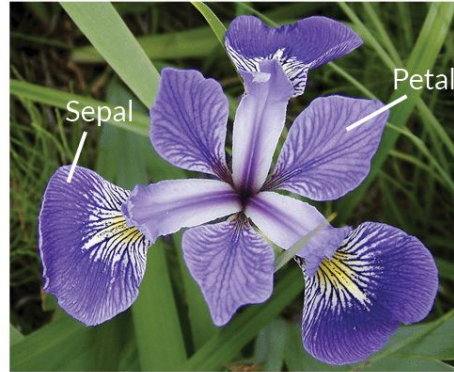
Formation API

18/10/2023

- Contexte du étude de cas
- Couches d'un logiciel
- **Partie 1:** Développement du back end
- **Partie 2:** Développement du front end
- **Partie 3:** Développement de l'API

Étude de cas

- Dataset Iris
- Collecté par le biologiste Ronald Fisher
- 150 échantillons
- 50 échantillons de chaque type de Iris



Iris Versicolor



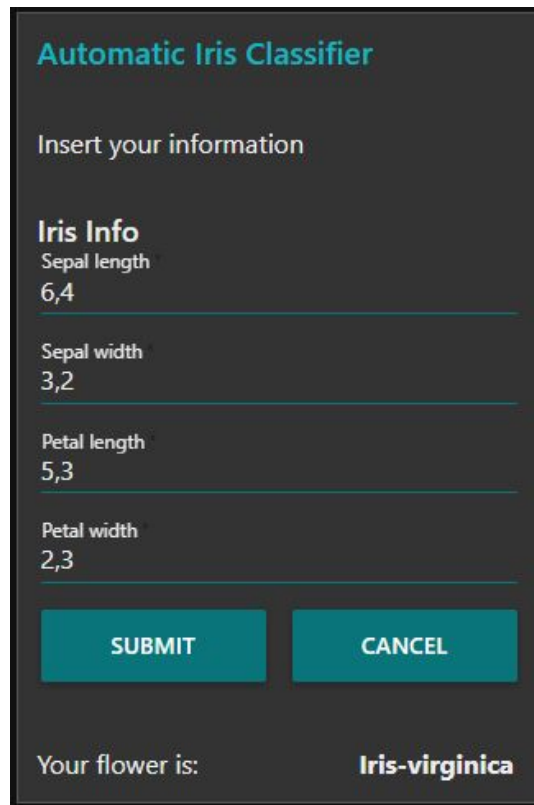
Iris Setosa



Iris Virginica

Notre objectif

- À la fin de cette formation, j'espère que vous serez capables de:
 - Développer une IA pour faire la classification des fleurs Iris en Python
 - Faire le design d'une interface graphique simple avec Node-Red
 - Faire le lien entre les deux logiciels avec une API



The image shows a web application titled "Automatic Iris Classifier". It has a dark grey background with teal-colored text and buttons. The interface prompts the user to "Insert your information" and lists four input fields: "Iris Info" (with sub-label "Sepal length" and value "6,4"), "Sepal width" (value "3,2"), "Petal length" (value "5,3"), and "Petal width" (value "2,3"). Below these fields are two teal buttons labeled "SUBMIT" and "CANCEL". At the bottom, it displays the result: "Your flower is: Iris-virginica".

Automatic Iris Classifier

Insert your information

Iris Info
Sepal length
6,4

Sepal width
3,2

Petal length
5,3

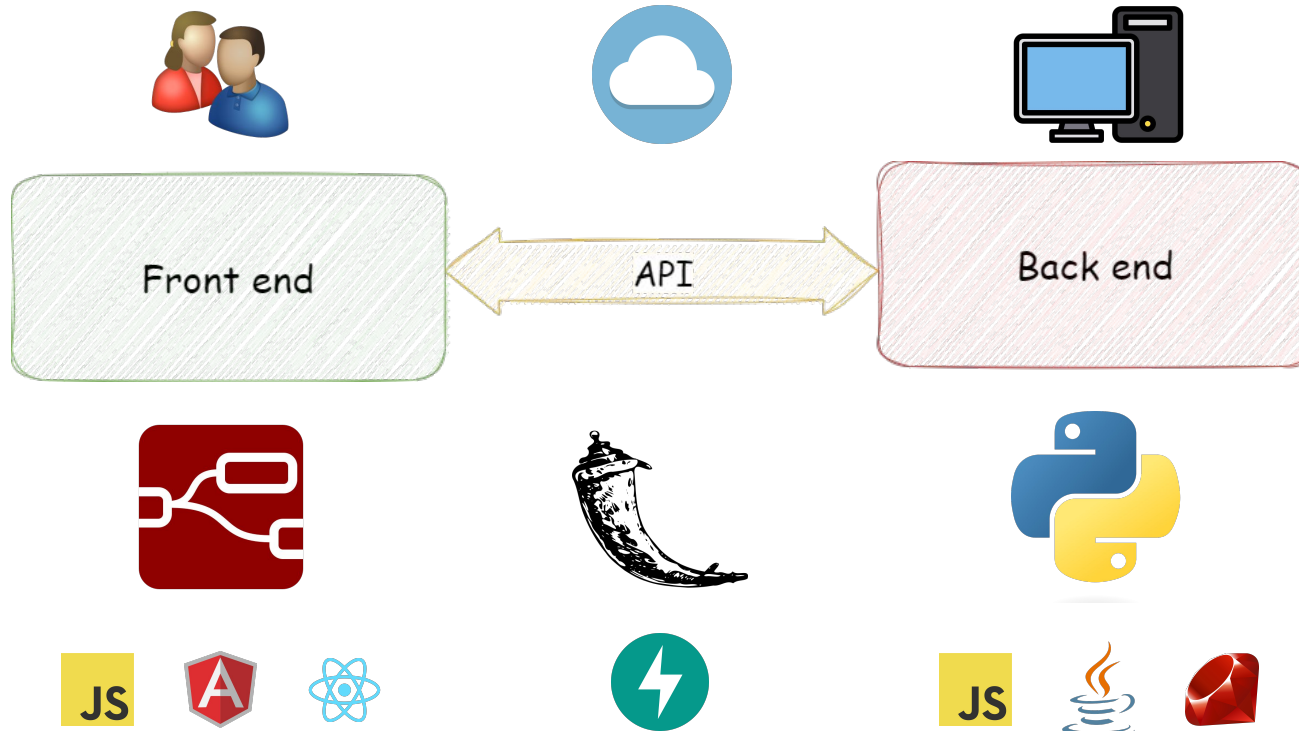
Petal width
2,3

SUBMIT **CANCEL**

Your flower is: **Iris-virginica**

Couches d'un logiciel

Vision générale

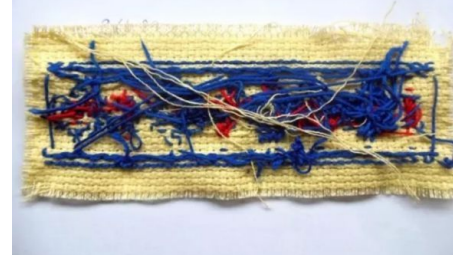


I am an It expert, but not Geek
13 hrs · 🌐
Front-end vs Back-end

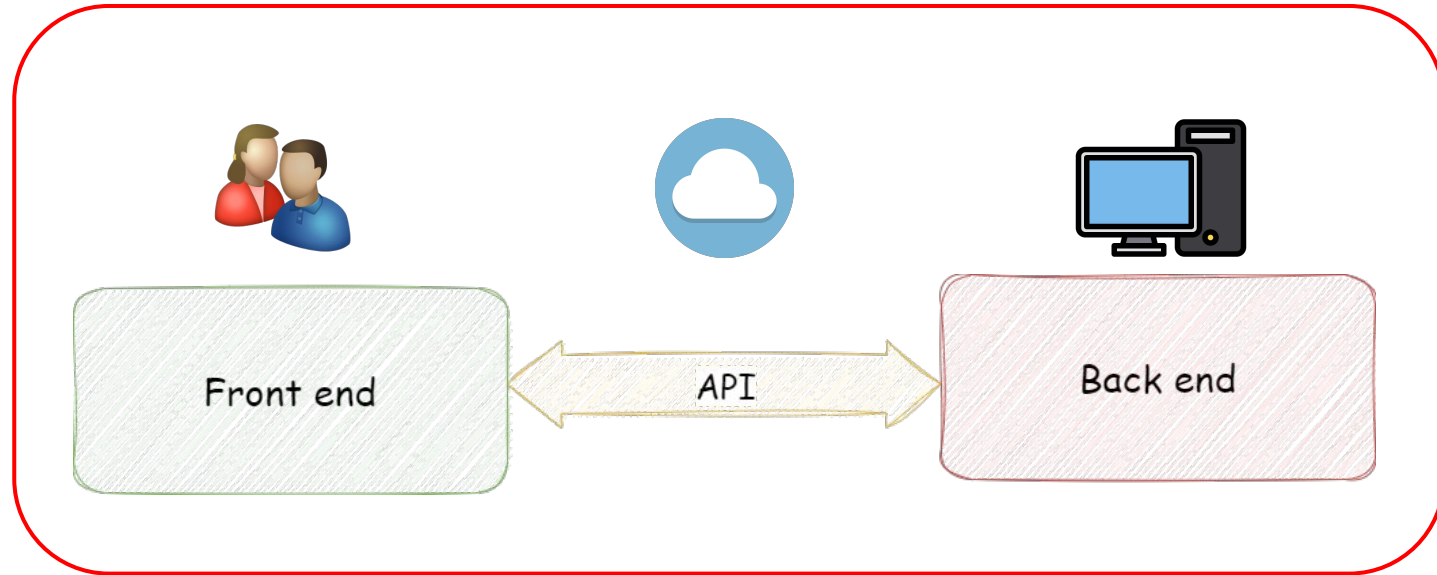
Frontend



Backend

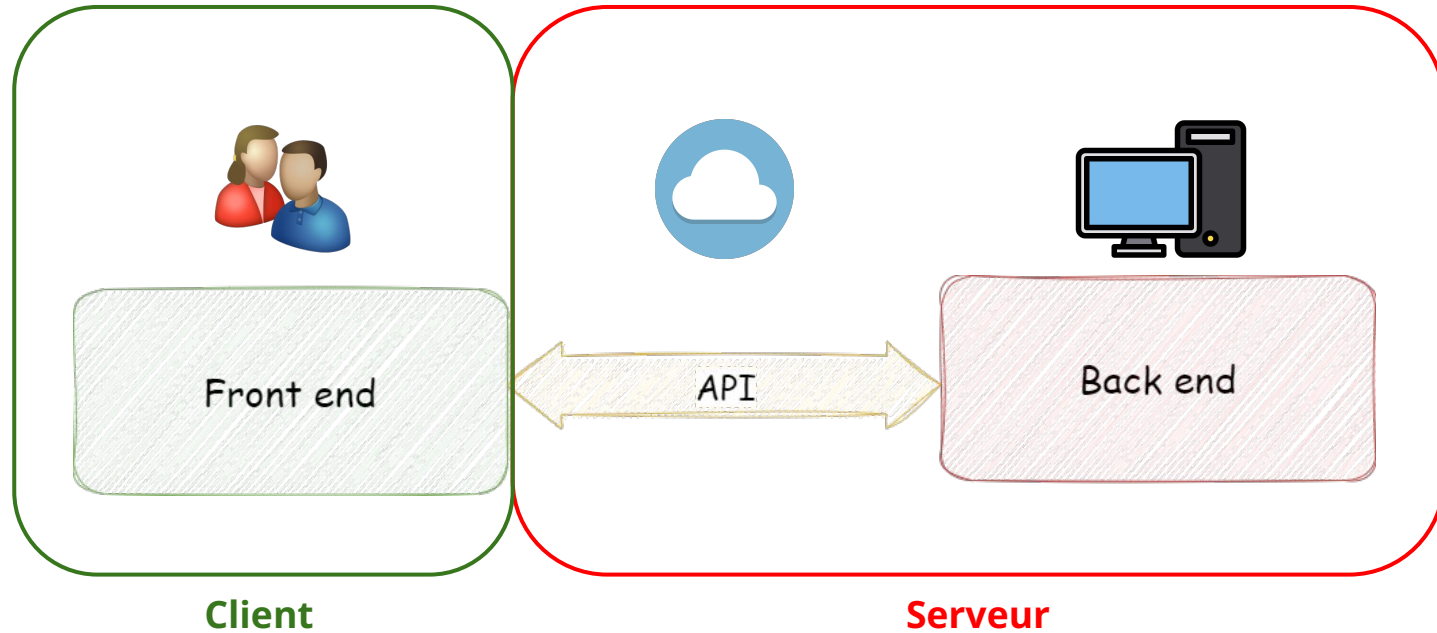


Architecture One-tier



Serveur unique

Architecture Two-tier



C'est quoi une API?

- **API: Application Programming Interface**
- Requêtes et réponses

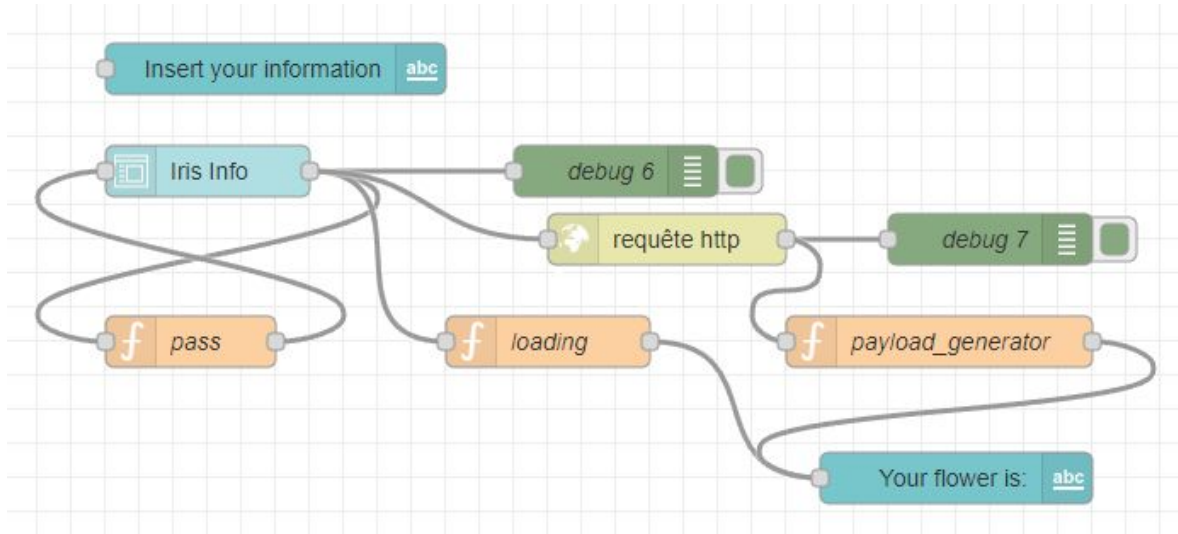
Partie 1: Développement du back end

- **Link:** <https://www.kaggle.com/vitorsilva2021/workshop-iris/>



Partie 2: Développement du front end

Node-Red



Automatic Iris Classifier

Insert your information

Iris Info
Sepal length
6,4

Sepal width
3,2

Petal length
5,3

Petal width
2,3

SUBMIT **CANCEL**

Your flower is: **Iris-virginica**

Partie 3: Développement de l'API

- Requête fait pour un client à un serveur
- Comme une conversation entre le client et le serveur
- Composé de:
 - Une ligne de requête
 - `GET /software/http/index.html HTTP/1.1`
 - Peut avoir aussi une query string:
`http://localhost:1880/myapi?val=50`
 - Une suite de entêtes
 - Contient des informations sur le client, le message, etc
 - Un corps de message, si nécessaire
 - Normalement utilisé avec POST requête

Types de requête

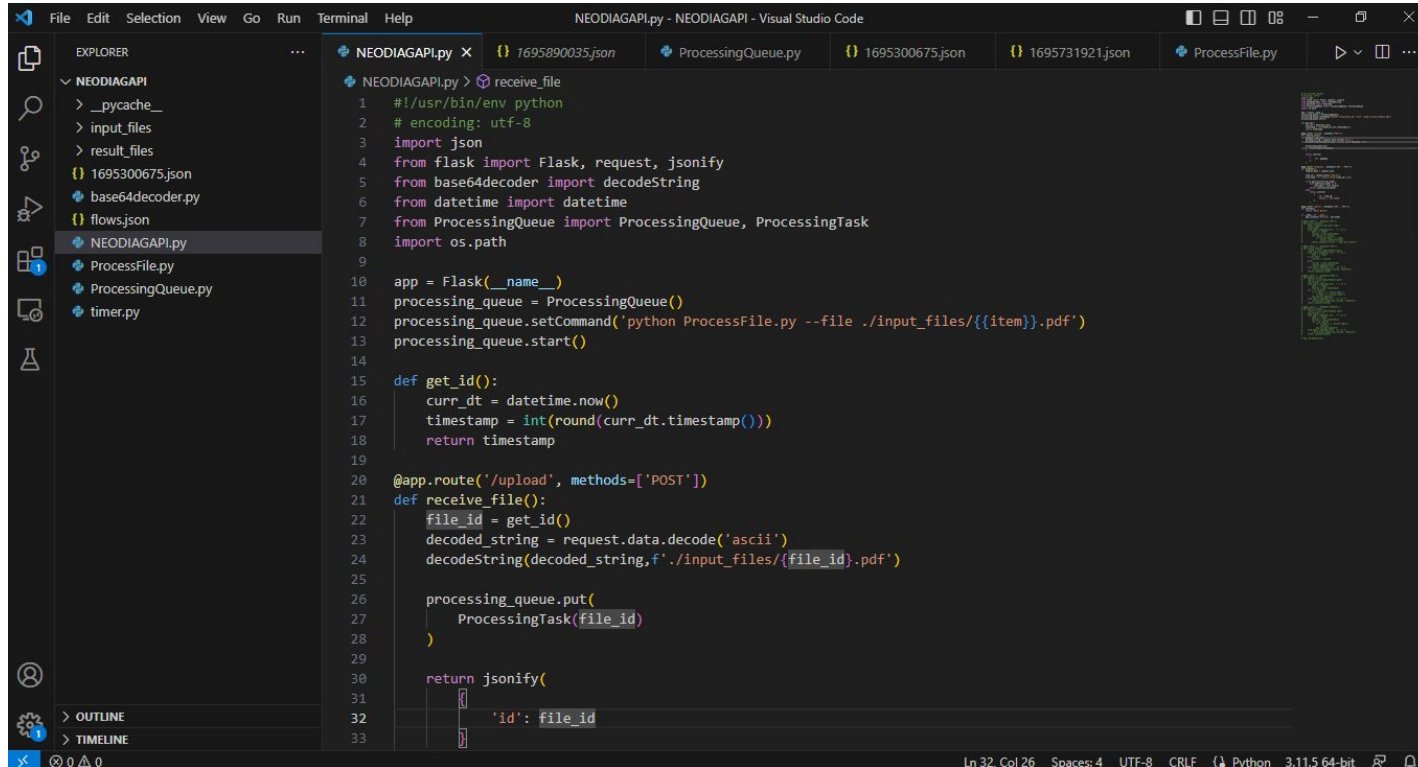
- GET
 - Pour récupérer information
- HEAD
 - Pareil, mais sans le corps de message
- POST
 - Envoi une information, et normalement change les ressources dans le serveur
- Et pas mal d'autres...

Types de réponses

- Les plus courants:
- 200 OK
 - Traité avec succès
- 400 Bad Request
 - Problème avec le requête (information invalide)
- 403 Forbidden
 - Pas de permission pour faire le requête
- 404 Not found
 - Le serveur n'a pas trouvé le ressource demandé

- Un point où l'API est accessible au client, où à un autre programme
- Point où le requête est fait
- **Example:** `http://localhost:1880/ui`
 - Où `localhost` est l'adresse, et `1880` la porte

Python



```
NEODIAGAPI.py X 1695890035.json ProcessingQueue.py 1695300675.json 1695731921.json ProcessFile.py
File Edit Selection View Go Run Terminal Help NEODIAGAPI.py - NEODIAGAPI - Visual Studio Code
EXPLORER
NEODIAGAPI
  > __pycache__
  > input_files
  > result_files
  {} 1695300675.json
  {} base64decoder.py
  {} flows.json
  {} NEODIAGAPI.py
  {} ProcessFile.py
  {} ProcessingQueue.py
  {} timer.py
OUTLINE
TIMELINE
NEODIAGAPI.py X
1  #!/usr/bin/env python
2  # encoding: utf-8
3  import json
4  from flask import Flask, request, jsonify
5  from base64decoder import decodeString
6  from datetime import datetime
7  from ProcessingQueue import ProcessingQueue, ProcessingTask
8  import os.path
9
10 app = Flask(__name__)
11 processing_queue = ProcessingQueue()
12 processing_queue.setCommand('python ProcessFile.py --file ./input_files/{{item}}.pdf')
13 processing_queue.start()
14
15 def get_id():
16     curr_dt = datetime.now()
17     timestamp = int(round(curr_dt.timestamp()))
18     return timestamp
19
20 @app.route('/upload', methods=['POST'])
21 def receive_file():
22     file_id = get_id()
23     decoded_string = request.data.decode('ascii')
24     decodeString(decoded_string, f'./input_files/{file_id}.pdf')
25
26     processing_queue.put(
27         ProcessingTask(file_id)
28     )
29
30     return jsonify(
31         {
32             'id': file_id
33         }
34     )
```

Bonus: Interface avec téléversement de fichier

Encodage de fichier



ASCII Text:

H i \n

ASCII to Binary:

01001000 01101001 00001010

ASCII to Binary:

010010000110100100001010

6-bit Base64:

010010 000110 100100 001010

Base64 Encoding:

S G k K

Sauvegarde de fichier

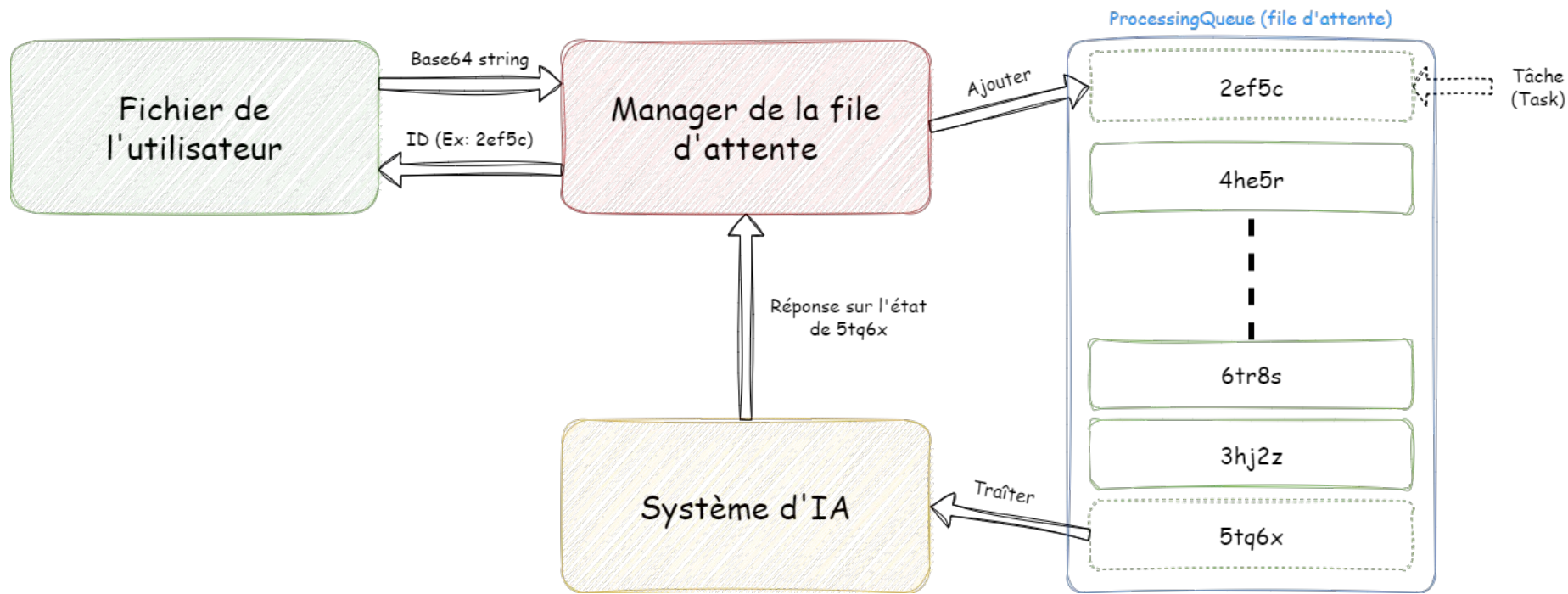
```
20 @app.route('/upload', methods=['POST'])
21 def receive_file():
22     file_id = get_id()
23     decoded_string = request.data.decode('ascii')
24     decodeString(decoded_string, f'./input_files/{file_id}.pdf')
25
26     processing_queue.put(
27         ProcessingTask(file_id)
28     )
29
30     return jsonify(
31         {
32             'id': file_id
33         }
34     )
```

Sauvegarde de fichier

```
20 @app.route('/upload', methods=['POST'])
21 def receive_file():
22     file_id = get_id()
23     decoded_string = request.data.decode('ascii')
24     decodeString(decoded_string, f'./input_files/{file_id}.pdf')
25
26     processing_queue.put(
27         ProcessingTask(file_id)
28     )
29
30     return jsonify(
31         {
32             'id': file_id
33         }
34     )
```

??

File d'attente



Merci pour votre attention !