Ressource – Connexion à une bdd via Python

Objectif :

- Se connecter à une base sqlite3 dans du code Python

Code Python

```
import flask
from flask import request, jsonify # modules permettant de requêter la bdd et de mettre la réponse dans un json
import sqlite3 # comme son nom l'indique

app = flask.Flask(__name__)
app.config["DEBUG"] = True

def dict_factory(cursor, row): # renvoie les résultats sous forme de dictionnaires plutôt que de listes - ce qui se convertit mieux au format JSON
    d = {}
    for idx, col in enumerate(cursor.description):
        d[col[0]] = row[idx]
    return d


@app.route('/', methods=['GET'])
def home():
    return '''<h1>Première base</h1>
<p>A prototype API for french birds.</p>'''


@app.route('/requetes/all', methods=['GET'])
def api_all():
    conn = sqlite3.connect('Data/birds.db') # objet représentant la connexion à la base de données
    conn.row_factory = dict_factory # dit à l'objet correspondant à la connexion d'utiliser la fonction dict_factory
    cur = conn.cursor() # objet curseur qui parcourt la base de données pour extraire les données
    all_books = cur.execute('SELECT * FROM birds_fr;').fetchall() # on exécute une requête SQL

    return jsonify(all_books) # les données récupérées sont converties au format JSON



@app.errorhandler(404)
def page_not_found(e): # Crée une page d'erreur affichée à l'utilisateur s'il spécifie une route non prise ne charge par l'API
    return "<h1>404</h1><p>The resource could not be found.</p>", 404


@app.route('/requetes', methods=['GET'])
def api_filter(): # utilise la même approche que api_all pour extraire les données de la base
    query_parameters = request.args

    speciescode = query_parameters.get('speciescode')
    sensitive_species = query_parameters.get('sensitive_species')
    popsize_etc = query_parameters.get('popsize_etc')

    query = "SELECT * FROM birds_fr WHERE"
    to_filter = []

    if speciescode:
        query += ' speciescode=? AND'
        to_filter.append(speciescode)
    if sensitive_species:
```

```
        query += ' sensitive_species=? AND'
        to_filter.append(sensitive_species)
    if popsize_etc:
        query += ' popsize_etc=? AND'
        to_filter.append(popsize_etc)
    if not (speciescode or sensitive_species or popsize_etc):
        return page_not_found(404)

    query = query[:-4] + ';'# Afin de parfaire notre requête SQL, on supprime le dernier AND et on complète la requête par le point-virgule requis par SQL

    conn = sqlite3.connect('Data/birds.db')
    conn.row_factory = dict_factory
    cur = conn.cursor()

    results = cur.execute(query, to_filter).fetchall()

    return jsonify(results)

app.run()
```

Les requêtes

http://127.0.0.1:5000/requetes?speciescode=A132

3 enregistrements ont pour code A132

http://127.0.0.1:5000/requetes/all

Liste de tous les tuples de la table